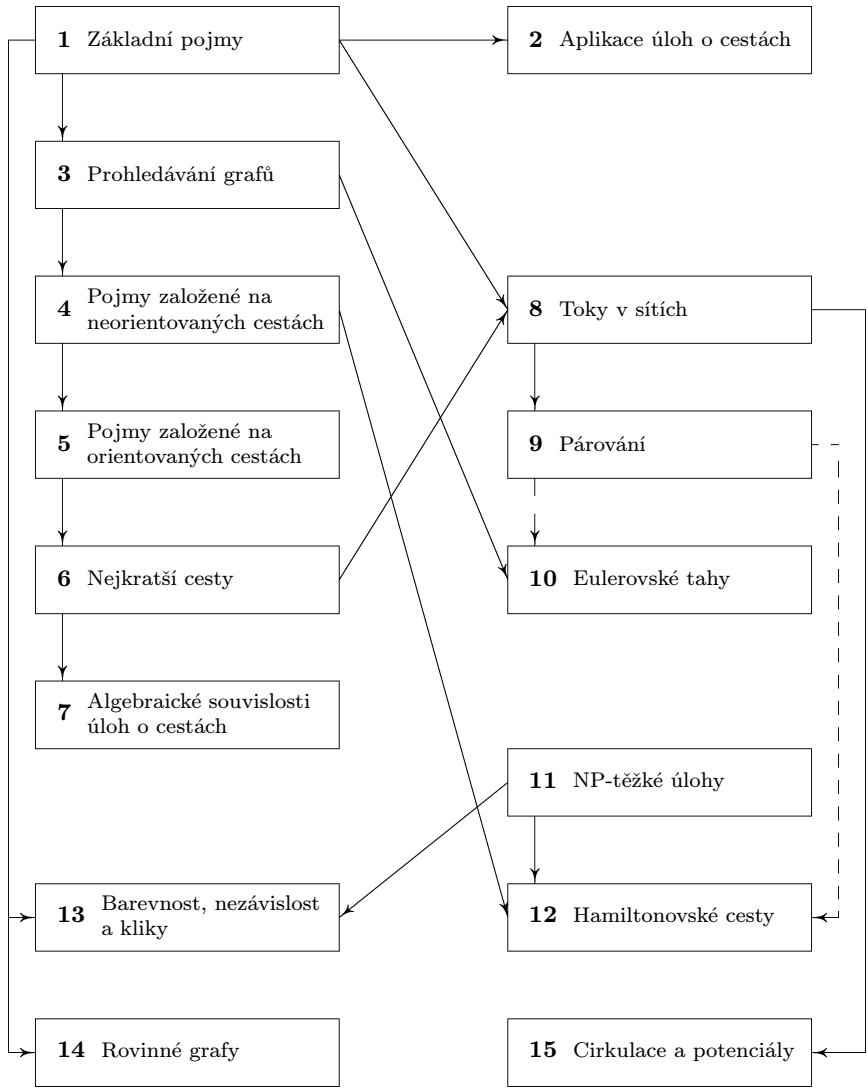
Jiří Demel

GRAFY

a jejich aplikace

Vydání třetí, elektronické (vlastním nákladem druhé), 2019

Obsah



Obrázek 1: Návaznosti kapitol (příklad orientovaného grafu).

Složité se skládá z mnoha jednoduchostí,
ovšem důmyslně sestavených.

Předmluva

Často je třeba orientovat se v komplikovaných vztazích mezi částmi nějakého celku. Řada lidí si při tom pomáhá obrázky podobnými těm, které najdete v této knize: části celku kreslíme jako puntíky, obdélníčky nebo kroužky, vztahy znázorňujeme čarami mezi nimi. Nesymetrické vztahy vyjadřujeme šipkami. Je zřejmé, že takto lze vyjádřit (a nakreslit) prakticky jakékoli vztahy mezi *dvojicemi* prvků.

Pojem grafu je přirozeným prostředkem k vyjádření párových vztahů, tj. vztahů mezi dvojicemi. Obecnější vztahy (už např. mezi trojicemi) lze nakreslit podstatně obtížněji. Snad proto se těmto obecnějším vztahům mnohdy vyhýbáme a raději volíme jiný, komplikovanější popis téže situace, založený na vztazích párových.

Díky tomu se nelze divit, že pojem grafu (a způsob uvažování, který označujeme jako grafový) je poměrně oblíbený a rozšířený.

Tato kniha je zaměřena na aplikace grafů.

Některé aplikace jsou zcela zjevné: daný problém „ze života“ se přeformuluje na grafovou úlohu, udělá se příslušný graf, který modeluje konkrétní situaci, grafová úloha se nějakým známým postupem vyřeší a získané řešení se přeloží z řeči grafů zpátky „do života“.

Často jsou však aplikace grafů skryté: nikde se žádný graf explicite nepoužije, graf zůstává jaksi v pozadí, ale použitý postup řešení má svou paralelu ve světě grafů a lze jej pomocí grafů zdůvodnit.

Umění aplikovat grafy zahrnuje dvě dílčí dovednosti. Především je třeba umět udělat (vymyslet) graf, který modeluje danou situaci (tj. jsou v něm zachyceny vztahy, o které nám jde). Někdy to je zcela triviální (např. při hledání cesty z Kolína do Rokycan), jindy to vyžaduje značnou lstivost.

Dále je třeba umět řešit alespoň ty nejdůležitější grafové úlohy, vědět, které úlohy jsou snadno a rychle řešitelné a které jsou naopak těžké. Také není k zahození, umíme-li převést nějakou (neznámou) úlohu na jinou, kterou dovedeme řešit. Toto umění převádět úlohy má mimochodem mnoho společného s uměním najít grafový model.

Kniha je určena pro technicky zaměřené čtenáře, zejména pro studenty technických vysokých škol. (Většinu textu však bez problémů přečte i středoškolák, který se nebojí používat „zdravý selský rozum“.) Vím moc dobře, že tito lidé nemívají

zálibu v matematických důkazech. Přesto v knize většinu jednoduchých tvrzení dokazuji. Jsem totiž přesvědčen, že k úspěšnému praktickému používání (nejen) teorie grafů je třeba umět samostatně vymyslet jednoduchý důsledek a *korektně* jej zdůvodnit. V knize uvedené důkazy můžete chápat jako příklady takových zdůvodnění. Pokud mohu radit, zkuste si nějaké (nejlépe každé) v knize uvedené tvrzení nejprve zdůvodnit sami a teprve potom čtěte důkaz. Konce důkazů jsou pro snazší orientaci označeny čtverečkem: □

Mnoho místa je v knize věnováno algoritmům a rozborům jejich fungování. Sleduji tím dva cíle: Při aplikacích se snadno můžete dostat do situace, kdy si musíte pro vaši speciální úlohu nějaký algoritmus vymyslet. Pokud si jej nebudete umět korektně zdůvodnit, snadno se můžete dočkat nečekaných (obvykle nepříjemných) překvapení. Druhý cíl se týká studentů informatiky. Teorie grafů je výborným cvičištěm s nepřeberným množstvím úloh pro výuku programování efektivních algoritmů.

Chci ovšem zdůraznit, že tato knížka není učebnicí navrhování a analýzy efektivních algoritmů. Výklad všelijakých důmyslných datových struktur zde není – to by knížka byla o hodně tlustší a už by nebyla tak moc o grafech. Výjimku jsem udělal v kapitole 11, kde je velice stručně vysvětlen rozdíl mezi lehkými a těžkými úlohami.

Tato knížka koncepčně vychází z mých dřívějších publikací [Demel88, Demel91]. Z nich jsem převzal hrubé členění textu na kapitoly a některé obrázky. Většina textu je však přepsána a rozšířena.

V knížce je řada cvičení jednak k počítání, jednak k samostatnému zamyšlení. Výsledky většiny cvičení jsou soustředěny na konci knihy.

Bude-li třeba po vydání této knihy zveřejnit k ní nějaké dodatky, opravy chyb apod., najdete je na Internetu na `http://kix.fsv.cvut.cz/~demel/grafy/`. Pokud byste mi chtěli napsat své připomínky, pište na adresu `demel@fsv.cvut.cz`.

Děkuji všem, kdo svými připomínkami nebo jakkoli jinak přispěli ke zlepšení tohoto textu. Zejména děkuji za cenné připomínky oběma recenzentům. Děkuji také Ediční radě Akademie věd České republiky za finanční podporu na vydání této knihy. Především však děkuji své ženě Marii, která mi při psaní byla nejen všestrannou oporou, ale také prvním kritikem. Veškeré nedostatky, které v knize zůstaly, však padají na mou hlavu.

Praha, leden 2002 Jiří Demel

Druhé vydání této knihy se liší od prvního převážně jen opravami chyb. Děkuji všem, kdo mne na chyby upozornili. Pokud by internetové adresy uvedené v původní předmluvě přestaly fungovat, zkuste `http://www.demel.name/~jiri/grafy/` a emailovou adresu `jiri@demel.name`.

Letky, únor 2015 Jiří Demel

Kapitola 1

Základní pojmy

1.1 Definice grafu

1.1.1 Nejdřív stručně a neformálně. Graf se skládá z tzv. vrcholů a tzv. hran. Hrana vždy spojuje dva vrcholy a je buď orientovaná, nebo neorientovaná. U hran orientovaných rozlišujeme počáteční a koncový vrchol a říkáme, že hrana vede z počátečního do koncového vrcholu. Neorientované hrany chápeme jako symetrické spojení dvou vrcholů.

Pokud hrana spojuje nějaký vrchol se sebou samým, nazýváme ji smyčkou.

Orientovaný graf má všechny hrany orientované, neorientovaný graf má všechny hrany neorientované. Smíšenými grafy, které mají oba druhy hran, se nebudeme zabývat.

A teď o něco formálněji:

1.1.2 Orientovaný graf. *Orientovaný graf* je trojice $G = (V, E, \varepsilon)$ tvořená neprázdnou konečnou množinou V, jejíž prvky nazýváme *vrcholy*, konečnou množinou E, jejíž prvky nazýváme *orientovanými hranami*, a zobrazením $\varepsilon : E \to V^2$, které nazýváme *vztahem incidence*. Toto zobrazení přiřazuje každé hraně $e \in E$ uspořádanou dvojici vrcholů (x, y). Prvý z nich, x, nazýváme *počátečním vrcholem hrany* a značíme jej $\textsc{Pv}(e)$. Druhý nazýváme *koncovým vrcholem hrany* a značíme jej $\textsc{Kv}(e)$.

O hraně e říkáme, že *vede z vrcholu x do vrcholu y* a také, že *spojuje* vrcholy x a y. O vrcholech x, y pak říkáme, že jsou *incidentní* (nebo že *incidují*) s hranou e a také naopak hrana e je *incidentní* s vrcholy x, y. Oba vrcholy x, y také souhrnně nazýváme *krajními vrcholy hrany e*.

Jestliže $\textsc{Pv}(e) = \textsc{Kv}(e)$, pak hranu e nazýváme (orientovanou) *smyčkou*. Vrchol, který není incidentní s žádnou hranou, nazýváme *izolovaným vrcholem*.

Je možné, aby několik hran mělo stejné počáteční a koncové vrcholy, tj. aby pro různé hrany e_1, e_2 platilo $\textsc{Pv}(e_1) = \textsc{Pv}(e_2)$ a $\textsc{Kv}(e_1) = \textsc{Kv}(e_2)$ nebo, zapsáno jinak, $\varepsilon(e_1) = \varepsilon(e_2)$. O takových hranách říkáme, že jsou *rovnoběžné* nebo též *násobné*.

Množina hran grafu může být prázdná.

1.1.3 Neorientovaný graf. Někdy je orientace hran nepodstatná. Jinými slovy, někdy nepotřebujeme či nechceme rozlišovat mezi počátečními a koncovými vrcholy hran. Proto zavádíme pojem neorientovaného grafu, který vznikne tím, že „zapomeneme“ na pořadí vrcholů v uspořádaných dvojicích.

Neorientovaný graf je trojice $G = (V, E, \varepsilon)$ tvořená neprázdnou konečnou množinou V, jejíž prvky nazýváme *vrcholy*, konečnou množinou E, jejíž prvky nazýváme *neorientovanými hranami*, a zobrazením ε, které nazýváme *vztahem incidence* a které každé hraně $e \in E$ přiřazuje jedno- nebo dvouprvkovou množinu vrcholů.

Těmto vrcholům říkáme *krajní vrcholy hrany* e. Říkáme také, že jsou *incidentní* (nebo že *incidují*) s hranou e. O hraně e pak říkáme, že je *incidentní* s těmito vrcholy nebo také že *spojuje* tyto vrcholy.

Je-li hrana e incidentní pouze s jediným vrcholem (tj. jestliže množina $\varepsilon(e)$ je jednoprvková), nazýváme hranu e (neorientovanou) *smyčkou*.

Je možné, aby několik hran spojovalo stejné (jedno- nebo dvouprvkové) množiny vrcholů, tj. aby pro různé hrany e_1, e_2 platilo $\varepsilon(e_1) = \varepsilon(e_2)$. O takových hranách říkáme, že jsou *rovnoběžné* nebo též *násobné*. Množina hran může být i zde prázdná.

1.1.4 Prázdný graf a nekonečné grafy. V literatuře, která je více teoreticky zaměřena, se lze setkat také s prázdným grafem (který nemá žádný vrchol, a tedy ani žádnou hranu) a s grafy, které mají nekonečně mnoho vrcholů. V této knize se jimi nebudeme zabývat.

1.1.5 Kreslení grafů. Grafy (orientované i neorientované) lze s výhodou znázorňovat kreslením. Vrcholy obvykle kreslíme jako body (kroužky), hrany jako čáry (úsečky, oblouky), které spojují příslušné dvojice vrcholů. Je-li hrana orientovaná, značíme orientaci šipkou od počátečního ke koncovému vrcholu. Je třeba mít na paměti, že graf lze většinou nakreslit mnoha k nepoznání různými způsoby.

Ze dvou nakreslení téhož grafu dáváme obvykle přednost takovému, kde jsou hrany kresleny pokud možno přímo s co nejmenším počtem průsečíků.

Nakreslení grafu formálně definujeme jako dvojici zobrazení φ, ψ, z nichž φ přiřazuje vrcholům grafu různé body v rovině a ψ přiřazuje každé hraně spojující vrcholy x, y jednoduchou křivku s krajními body $\varphi(x)$, $\varphi(y)$. Přitom požadujeme, aby žádná křivka $\psi(e)$ neobsahovala žádný z bodů $\varphi(x)$ jako svůj vnitřní bod.

Rovinné nakreslení je takové nakreslení grafu, že libovolné dvě křivky přiřazené různým hranám grafu mají společné nejvýše své krajní body, tj. body přiřazené vrcholům grafu. *Rovinný graf* (též *planární*) je takový graf, ke kterému existuje rovinné nakreslení.

Ne každý graf je rovinný a i rovinný graf lze nakreslit nerovinným způsobem. Rovinným grafům je věnována kapitola 14, str. 225.

1.1.6 Cvičení. V následujících příkladech grafových popisů reálných situací rozhodněte, který druh grafu (orientovaný, neorientovaný) lépe vystihuje skutečnost:

Situace:	Silniční síť
Vrcholy:	Křižovatky
Hrany:	Křižovatky i, j jsou spojeny přímou silnicí

Situace:	Silniční síť
Vrcholy:	Silnice
Hrany:	Silnice i, j mají společnou křižovatku
Situace:	Molekula
Vrcholy:	Atomy
Hrany:	Atomy i, j jsou vázány chemickou vazbou
Situace:	Elektrické zařízení
Vrcholy:	Vodiče (uzly)
Hrany:	Mezi vodiči i, j je zapojena součástka
Situace:	Šachy
Vrcholy:	Postavení figurek na šachovnici spolu s informací, kdo je na tahu
Hrany:	Z postavení i lze jedním tahem dostat postavení j
Situace:	Úřad
Vrcholy:	Úředníci
Hrany:	Úředník i je nadřízeným úředníka j
Situace:	Národní hospodářství
Vrcholy:	Produkty a služby
Hrany:	Produkt nebo služba i je zapotřebí k výrobě nebo vykonání j
Situace:	Přidělování úkolů
Vrcholy:	Pracovníci a úkoly
Hrany:	Pracovník i je schopen vykonat úkol j
Situace:	Složitá činnost, proces
Vrcholy:	Jednoduché činnosti, úkony
Hrany:	Činnost i musí být dokončena před započetím činnosti j
Situace:	Rodokmen
Vrcholy:	Lidé
Hrany:	Osoba i je otcem osoby j
Situace:	Rodinné vztahy
Vrcholy:	Lidé
Hrany:	Osoba i je rodičem osoby j
Situace:	Mnohostěn
Vrcholy:	Vrcholy mnohostěnu
Hrany:	Vrcholy i, j leží na stejné hraně mnohostěnu
Situace:	Program pro počítač
Vrcholy:	Instrukce
Hrany:	Instrukce j může být vykonána bezprostředně po vykonání instrukce i
Situace:	Program pro počítač
Vrcholy:	Podprogramy (procedury a funkce)
Hrany:	Podprogram j je volán (aktivován) podprogramem i
Situace:	Kniha o teorii grafů
Vrcholy:	Kapitoly
Hrany:	Při četbě kapitoly j se předpokládá znalost kapitoly i (viz obr. 1, str. 8)

1.1.7 Ohodnocený graf, síť. V četných aplikacích grafy samotné nepostačují k adekvátnímu popisu situace. Proto se často ke hranám či vrcholům přidávají nějaké hodnoty (obvykle číselné), které reprezentují např. doby trvání nebo náklady činností, propustnosti potrubí, pravděpodobnosti událostí apod. Graf, jehož hrany a (nebo) vrcholy jsou opatřeny takovými hodnotami, nazýváme *ohodnoceným grafem* nebo též *sítí*.

Zobrazení, které přiřazuje vrcholům nebo hranám jejich hodnoty, nazýváme *ohodnocením* vrcholů nebo hran. Často se stává, že v jednom ohodnoceném grafu máme několik různých ohodnocení současně.

1.1.8 Cvičení. U jednotlivých příkladů z 1.1.6 posuďte možnosti a vhodnost rozličných ohodnocení hran či vrcholů.

1.1.9 Poznámky k terminologii. Vrcholy grafu bývají nazývány též *uzly, prvky, styčníky, body*. Hrany grafu bývají nazývány též *vazbami, žebry, šipkami, šípy* apod.

Všeobecně platí, že ani definice grafu, ani terminologie není zdaleka jednotná, a to jak v naší, tak i ve světové literatuře. Základní myšlenky jsou sice společné, ale často se setkáte s odchylkami v důležitých detailech. Vždy je třeba prostudovat úvodní partie knihy, kde obvykle bývá stručný přehled používaných pojmů.

1.2 Orientované a neorientované grafy

Oba druhy grafů (orientované a neorientované) se studují zpravidla odděleně, bývají jim věnovány oddělené kapitoly knih apod., ovšem zdání, že se jedná o dvě samostatné teorie, je klamné.

V některých knihách lze najít i definici smíšeného grafu, který může mít oba druhy hran. Teorie se však takovými zobecněnými grafy zabývá jen okrajově, v aplikacích lze zpravidla totéž modelovat trochu složitějšími grafy orientovanými.

1.2.1 Symetrizace a orientace grafu. *Symetrizací* orientovaného grafu nazýváme neorientovaný graf, který dostaneme tím, že „zapomeneme“ na orientaci hran. Na obrázku bychom prostě umazali šipky.

Máme-li naopak nějaký neorientovaný graf G, můžeme z něj udělat graf orientovaný dvěma způsoby: Při prvém způsobu nějak (jakkoli, třeba i náhodně) zvolíme orientaci hran (v obrázku přimalujeme šipky) a získáme tím tzv. *orientaci grafu* G. Druhý způsob spočívá v tom, že každou neorientovanou hranu kromě smyček nahradíme dvěma opačně orientovanými hranami a každou neorientovanou smyčku nahradíme smyčkou orientovanou. Tím získáme tzv. *symetrickou orientaci grafu* G.

Orientovaný graf nazýváme *symetrickým*, jestliže pro každou dvojici vrcholů x, y platí, že počet hran z x do y je roven počtu hran z y do x.

1.2.2 Základem je orientovaný graf. Jak už jsme naznačili v 1.1.3, v této knize budeme pokládat orientované grafy za základní. Neorientované grafy pak lze získat tím, že abstrahujeme od orientace hran. Jsou k tomu dva důvody:

Za prvé, když někomu sdělujeme, které vrcholy jsou v grafu spojeny hranou, pak obvykle jeden z obou vrcholů uvádíme (vyslovíme, napíšeme) dříve a druhý později. Přitom ihned dodáváme, že na pořadí nezáleželo a že jsme oba vrcholy mohli vyjmenovat i v opačném pořadí.

Za druhé, při práci s orientovaným grafem se někdy stává, že s ním potřebujeme udělat něco, co nezáleží na orientaci hran. Např. orientace jednosměrných ulic je důležitá pro automobilisty, ale zcela nedůležitá pro chodce.

Bude tedy výhodné, aby pojmy, které jsou určeny pro neorientované grafy, bylo možno používat i pro grafy orientované.

Dosáhneme toho tím, že všechny grafové pojmy budeme definovat pouze pro grafy orientované. Některé z těchto pojmů (totiž ty, které nezávisí na orientaci hran) potom bude možno používat i pro grafy neorientované.

1.3 Důležité množiny hran a vrcholů

Je-li dán graf G, pak množinu všech jeho vrcholů budeme značit $V(G)$ a množinu jeho hran $E(G)$. Je-li M libovolná konečná množina, budeme počet prvků množiny M značit $|M|$.

1.3.1 Množiny hran a vrcholů v orientovaných grafech. Nechť $G = (V, E, \varepsilon)$ je orientovaný graf, nechť x a y jsou jeho libovolné vrcholy a $A \subseteq V$ nechť je libovolná množina jeho vrcholů. Pak zavedeme následující pojmy a značení:

$V_G^+(x)$ $= \{z \in V \mid (x,z) \in \varepsilon(E)\}$ = množina *následníků vrcholu* x,
tj. množina vrcholů, do nichž vede hrana z x.

$V_G^-(x)$ $= \{z \in V \mid (z,x) \in \varepsilon(E)\}$ = množina *předchůdců vrcholu* x,
tj. množina vrcholů, z nichž vede hrana do x.

$V_G(x)$ $= V_G^+(x) \cup V_G^-(x)$ = množina *sousedů vrcholu* x,
tj. množina vrcholů spojených hranou s vrcholem x.

$V_G(A)$ $= \bigcup_{x \in A} V_G(x)$,
tj. množina vrcholů spojených hranou s některým vrcholem z A.

$E_G^+(x)$ $= \{e \in E \mid \mathrm{P}\textsc{v}(e) = x\}$ = *výstupní okolí vrcholu* x,
tj. množina hran s počátečním vrcholem x.

$E_G^-(x)$ $= \{e \in E \mid \mathrm{K}\textsc{v}(e) = x\}$ = *vstupní okolí vrcholu* x,
tj. množina hran s koncovým vrcholem x.

$E_G(x)$ $= E_G^+(x) \cup E_G^-(x)$ = *okolí vrcholu* x,
tj. množina hran incidentních s vrcholem x.

$W_G^+(A) = \{e \in E \mid \mathrm{Pv}(e) \in A \ \&\ \mathrm{Kv}(e) \notin A\}$,
tj. množina všech hran, jejichž počáteční vrchol leží v A a koncový vrchol neleží v A.

$W_G^-(A) = \{e \in E \mid \mathrm{Pv}(e) \notin A \ \&\ \mathrm{Kv}(e) \in A\}$,
tj. množina všech hran, jejichž počáteční vrchol neleží v A a koncový vrchol leží v A.

$W_G(A) = W_G^+(A) \cup W_G^-(A) =$ *řez určený množinou vrcholů* A,
tj. množina všech hran, jejichž jeden (libovolný) vrchol leží v A a druhý v množině A neleží.

$m_G^+(x,y) = |E_G^+(x) \cap E_G^-(y)| =$ *násobnost hrany*,
tj. počet hran s počátečním vrcholem x a koncovým vrcholem y.

$m_G(x,y) = |E_G(x) \cap E_G(y)| =$ *násobnost hrany*,
tj. počet hran spojujících vrcholy x a y.

$d_G^+(x) = |E_G^+(x)| =$ *výstupní stupeň vrcholu* x,
tj. počet hran vycházejících z vrcholu x.

$d_G^-(x) = |E_G^-(x)| =$ *vstupní stupeň vrcholu* x,
tj. počet hran vcházejících do vrcholu x.

$d_G(x) = d_G^+(x) + d_G^-(x) =$ *stupeň vrcholu* x,
tj. počet hran incidentních s vrcholem x, přičemž smyčky počítáme dvakrát.

Vždy, když bude z kontextu zřejmé, jaký graf máme na mysli, budeme vynechávat index $_G$ a budeme psát stručněji $V(x)$ namísto $V_G(x)$ apod.

1.3.2 Cvičení.

1. Dokažte, že platí $\sum_{y \in V(G)} d(y) = 2|E(G)|$.
2. Jaký je vztah mezi čísly $d(x)$, $|E(x)|$ a $|V(x)|$?
3. Co znamenají výrazy $m(x,x)$ a $m^+(x,x)$?
4. Může řez, tj. množina $W(A)$, obsahovat smyčku?
5. Je nějaký vztah mezi čísly $|W(A)|$ a $|V(A) \setminus A|$?

1.3.3 Množiny hran a vrcholů v neorientovaných grafech. Některé z výše definovaných pojmů a značení nezávisí na orientaci hran, a proto je lze použít i pro grafy neorientované. Konkrétně jde o značení $V_G(x)$, $V_G(A)$, $E_G(x)$, $W_G(A)$, $m_G(x,y)$ a $d(x)$ a jim odpovídající pojmy. Všimněte si, že i slovní vysvětlení těchto pojmů jsou nezávislá na orientaci hran.

1.3.4 Prostý graf a multigraf. *Prostý graf* je graf, v němž násobnost každé hrany je nejvýše rovna jedné. *Multigraf* je graf, v němž násobnosti hran mohou být i větší než jedna.

Pozor, v literatuře je často slovem graf označován pouze prostý graf a slovo multigraf se pak používá k pojmenování obecnějšího případu, kdy násobnosti hran mohou být i větší než jedna.

V této knize je tedy multigraf totéž, co (obecný) graf. Slovo graf zde používáme obecněji, než je obvyklé, jednak s ohledem na aplikace, jednak proto, že řada tvrzení i algoritmů funguje bez úprav i pro multigrafy, tj. pro grafy, které nemusí být prosté. Slovo multigraf budeme používat pro zdůraznění, že graf nemusí být prostý, např. takto:

Symetrizací prostého orientovaného grafu může vzniknout multigraf.

1.3.5 Prosté grafy a relace. V prostém orientovaném grafu můžeme každou hranu e ztotožnit s uspořádanou dvojicí vrcholů $(\mathrm{P}v(e), \mathrm{K}v(e))$, neboť touto dvojicí vrcholů je hrana e jednoznačně určena (v prostém grafu nemohou být dvě takové hrany). Množinu hran prostého orientovaného grafu tedy můžeme pokládat za binární relaci na množině vrcholů. O relacích pojednáme podrobněji v 1.12.3, str. 33.

1.4 Porovnávání grafů

1.4.1 Rovnost grafů. Řekneme, že dva grafy $G = (V, E, \varepsilon)$ a $G' = (V', E', \varepsilon')$ jsou si rovny, jestliže $V = V'$, $E = E'$ a $\varepsilon = \varepsilon'$.

1.4.2 Izomorfismus. Grafy G, G' se nazývají vzájemně *izomorfní*, když existují dvě vzájemně jednoznačná zobrazení $f : V \to V'$ a $g : E \to E'$ taková, že zachovávají vztahy incidence ε a ε'. Přesněji, když pro každou hranu $e \in E$ platí:

$$\varepsilon(e) = (x, y) \qquad \Longleftrightarrow \qquad \varepsilon'(g(e)) = (f(x), f(y)) \ ,$$

popř.

$$\varepsilon(e) = \{x, y\} \qquad \Longleftrightarrow \qquad \varepsilon'(g(e)) = \{f(x), f(y)\}$$

v závislosti na tom, zda se jedná o grafy orientované nebo neorientované.

Vztah izomorfismu značíme $G \cong G'$.

Mnoho vlastností grafů se přenáší izomorfismem. Přesněji, mnoho vlastností $\mathcal{V}$ je takových, že má-li graf G_1 vlastnost $\mathcal{V}$ a platí-li $G_1 \cong G_2$, pak i graf G_2 má vlastnost $\mathcal{V}$. Teorie grafů se téměř výlučně zabývá právě takovýmito vlastnostmi.

Příkladem vlastnosti, která se zachovává izomorfismem, je třeba „počet vrcholů stupně 2, které mají alespoň jednoho souseda stupně 3“.

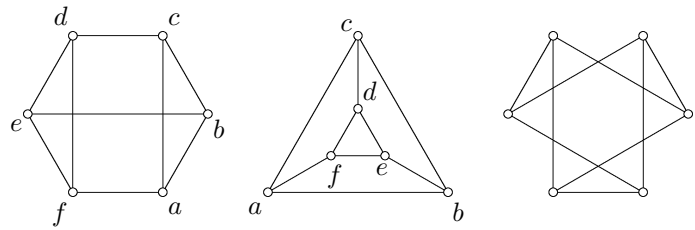
Chceme-li dokázat, že dva grafy jsou izomorfní, stačí najít (stačí uhodnout) izomorfismus (tj. zmíněná dvě zobrazení vrcholů a hran) a ověřit, že to jsou ta správná zobrazení, tj. že splňují výše uvedené podmínky. Jsou-li oba grafy prosté, stačí dokonce najít jen přiřazení vrcholů, přiřazení hran pak totiž je jednoznačně určeno. Na obrázku 1.1 na následující straně jsou tři navzájem izomorfní grafy. Pro prvé dva je izomorfismus naznačen shodným pojmenováním vrcholů.

Zjednodušeně můžeme říci, že dva grafy jsou izomorfní, když se liší pouze nakreslením a označením (pojmenováním) vrcholů a hran.

Chceme-li naopak dokázat, že dva grafy izomorfní nejsou, je třeba buď vyzkoušet všechna možná zobrazení (což obvykle je nesmírně pracné), anebo najít vlastnost, která se přenáší izomorfismem a kterou má jen jeden z obou grafů.

Rozhodnutí, zda dva dané grafy jsou izomorfní, je v řadě případů obtížným úkolem. Při zjišťování izomorfismu lze často využít tato jednoduchá pozorování:

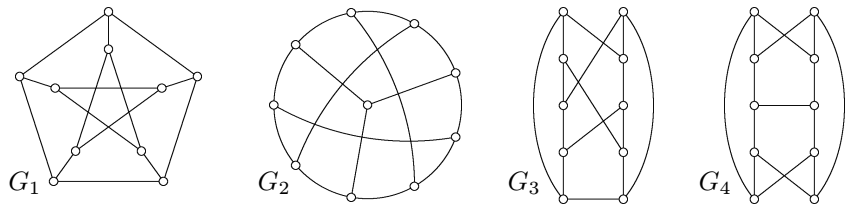
1. Izomorfní grafy musí mít stejné počty vrcholů i hran.
2. Vrcholu stupně k lze izomorfismem přiřadit pouze vrchol stejného stupně k.
3. Dvojici sousedních vrcholů může být izomorfismem přiřazena opět jen dvojice sousedních vrcholů.



Obrázek 1.1: Vzájemně izomorfní grafy.

1.4.3 Cvičení. O grafu na obr. 1.1 vpravo dokažte, že je izomorfní s předchozími dvěma. Návod: doplňte pojmenování vrcholů a hran, čímž vlastně získáte zobrazení f a g, a ověřte, že zachovávají vztahy incidence. Pokud nezachovávají, zkuste jiná zobrazení.

1.4.4 Cvičení. O grafech na obr. 1.2 rozhodněte, které z nich jsou navzájem izomorfní a které ne.



Obrázek 1.2: Jsou tyto grafy izomorfní?

1.4.5 Podgrafy. Graf G' je *podgrafem* grafu G, vznikne-li z grafu G vynecháním nějakých (nebo žádných) vrcholů a hran. Podstatné je, že podgraf musí být také grafem: spolu s každou hranou, která je v podgrafu, tam musí být i oba její krajní vrcholy. Poznamenejme, že každý graf pokládáme za podgraf sebe sama. Jsou dva speciální druhy podgrafů:

Graf G' nazýváme *faktorem* grafu G, vznikne-li z grafu G pouze vynecháním některých (nebo žádných) hran, tj. platí-li $V(G) = V(G')$.

Graf G' nazýváme *podgrafem indukovaným množinou vrcholů* $A \subseteq V(G)$ (též *úplným podgrafem na množině* A), jestliže podgraf G' má množinu vrcholů A a obsahuje všechny hrany grafu G, jejichž oba vrcholy leží v A. Indukovaný podgraf G' lze získat z grafu G tím, že vynecháme vrcholy, které neleží v množině A, a pak vynecháme všechny hrany, které byly incidentní s vynechanými vrcholy. (Tyto hrany je třeba vynechat, aby to, co zbude, byl graf.)

Poznamenejme, že obecný podgraf vždy můžeme dostat jako faktor nějakého indukovaného podgrafu a také jako indukovaný podgraf nějakého faktoru původního grafu. Rozdíl je jen v tom, zda napřed vynecháváme hrany, nebo vrcholy.

1.4.6 Příklad. Mějme graf, jehož hranami jsou všechny silnice a vrcholy jsou všechny křižovatky a slepé konce silnic. Omezíme-li se na křižovatky, které leží v nějakém okrese, a všechny silnice mezi nimi, dostaneme indukovaný podgraf. Ponecháme-li všechny křižovatky a omezíme se na silnice první třídy, dostaneme faktor s mnoha izolovanými vrcholy.

1.5 Sledy a odvozené pojmy

1.5.1 Sled. Posloupnost vrcholů a hran $v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$ nazýváme *orientovaným sledem*, jestliže pro každou hranu e_i z této posloupnosti platí $\textsc{Pv}(e_i) = v_{i-1}$ a $\textsc{Kv}(e_i) = v_i$.

Posloupnost vrcholů a hran $v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$ nazýváme *neorientovaným sledem*, jestliže každá hrana e_i z této posloupnosti spojuje vrcholy v_{i-1}, v_i.

Vrchol v_0 v obou případech nazýváme *počátečním* a vrchol v_k *koncovým vrcholem sledu*. O sledu říkáme, že *vede z vrcholu* v_0 *do vrcholu* v_k, nebo také, že *spojuje vrcholy* v_0, v_k.

V orientovaném i neorientovaném sledu na sebe vrcholy i hrany „navazují“. U orientovaného sledu však navíc požadujeme, aby všechny hrany byly orientovány „vpřed ve směru sledu“. U neorientovaného sledu na orientaci nezáleží. Proto má pojem neorientovaného sledu smysl pro orientované i neorientované grafy. Navíc, každý orientovaný sled je také sledem neorientovaným (neboť splňuje podmínky, aby se takto mohl nazývat). Konečně poznamenejme, že v obecných sledech se mohou vrcholy i hrany opakovat.

V grafu G na obr. 1.7, str. 24, je posloupnost $v_3, e_6, v_1, e_1, v_2, e_3, v_4$ orientovaným (a samozřejmě zároveň i neorientovaným) sledem, zatímco posloupnost $v_3, e_4, v_2, e_1, v_1, e_1, v_2$ je pouze neorientovaným sledem a posloupnost v_3, e_2, v_4, e_4, v_1 vůbec není sledem.

V grafu silniční sítě s jednosměrnými silnicemi smí auta jezdit jen podle orientovaných sledů, zatímco chodci smí chodit i podle sledů neorientovaných.

Triviální sled je sled, který obsahuje jediný vrchol a žádnou hranu. Triviální sled lze pokládat za orientovaný i neorientovaný.

Každý sled (kromě triviálního) je jednoznačně určen posloupností svých hran. V prostém grafu je sled určen i posloupností vrcholů.

1.5.2 Tah, cesta. Orientovaný (neorientovaný) sled, v němž se žádná hrana neopakuje, nazýváme *orientovaným (neorientovaným) tahem*. Orientovaný (neorientovaný) sled, v němž se neopakuje žádný vrchol, nazýváme *orientovanou (neorientovanou) cestou*.

Všimněte si, že z faktu, že se v cestě neopakují vrcholy, vyplývá, že se v ní neopakují ani hrany. Každá cesta je tedy zároveň také tahem a sledem, zatímco tah je vždy sledem, ale není vždy cestou.

Název tah je pravděpodobně odvozen od známé úlohy nakreslit „domeček“ nebo jiný obrázek (graf) jedním tahem.

1.5.3 Uzavřené sledy. Sled (orientovaný nebo neorientovaný), který má alespoň jednu hranu a jehož počáteční a koncový vrchol splývají, nazýváme *uzavřeným sledem*. Podobně mluvíme o *uzavřeném tahu*.

Uzavřená cesta je uzavřený sled, v němž se neopakují vrcholy (kromě toho, že $v_0 = v_k$) a navíc se neopakují ani hrany. Pro uzavřené cesty se používají speciální názvy: *kružnice* je neorientovaná uzavřená cesta a *cyklus* je orientovaná uzavřená cesta. Opět platí, že cyklus je zároveň i kružnicí, ale naopak to neplatí.

Kružnice, která má tři hrany, se nazývá *trojúhelník*.

Poznamenejme, že triviální sled nepokládáme za sled uzavřený. Pro definici uzavřených cest jsme museli zakázat kromě opakování vrcholů také opakování hran proto, aby se posloupnost v_0, e, v_1, e, v_0 nemohla nazývat uzavřenou cestou.

1.5.4 Dostupnost. Řekneme, že vrchol y je *orientovaně (neorientovaně) dostupný z vrcholu* x, jestliže existuje orientovaný (neorientovaný) sled vedoucí z vrcholu x do vrcholu y. Všimněte si, že sled spojující x a y existuje právě tehdy, když existuje cesta spojující tyto vrcholy. (Proč?)

1.5.5 Sledy v ohodnocených grafech. V ohodnocených grafech má často velmi dobrý smysl hovořit o součtu ohodnocení hran v nějakém sledu, cestě, cyklu apod. Jestliže např. ohodnocení hrany vyjadřuje vzdálenost, množství práce, času nebo peněz potřebných k průchodu hranou, pak součet ohodnocení hran má jistě dobrý praktický smysl. Přitom ohodnocení každé hrany počítáme vždy tolikrát, kolikrát se hrana ve sledu vyskytuje. Pro tento součet ohodnocení hran se vžil termín *délka sledu (cesty, tahu, kružnice, cyklu)* a v tomto smyslu se hovoří o *nejkratším* nebo *nejdelším sledu, cestě* atd. Hledání nejkratších nebo nejdelších cest patří k nejčastěji aplikovaným úlohám teorie grafů.

Nutno ovšem poznamenat, že termín délka sledu (cesty, atd.) je často užíván také pro označení počtu hran ve sledu, cestě apod. Tento rozpor ve vžité terminologii lze formálně řešit úmluvou, že počet hran ve sledu budeme pokládat za součet ohodnocení, je-li každá hrana ohodnocena jedničkou.

Dodejme, že hledání cest s nejmenším počtem hran je podstatně jednodušší než hledání cest s nejmenším součtem ohodnocení (viz 3.2 a kapitola 6).

V této knize budeme používat termíny délka sledu a nejkratší sled, cesta atd. téměř vždy ve smyslu „součet ohodnocení“ a jen výjimečně ve smyslu „počet hran“, přičemž na tyto výjimky vždy zvlášť upozorníme.

1.6 Speciální grafy

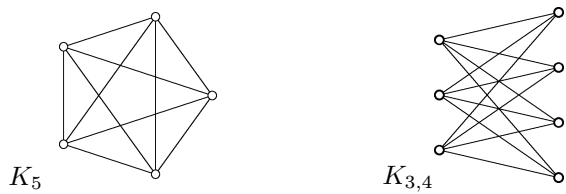
Diskrétní graf je graf, který nemá žádnou hranu. Podle potřeby jej můžeme považovat za orientovaný či neorientovaný.

Úplný orientovaný graf je prostý graf $G = (V, R)$, kde R je množina všech uspořádaných dvojic různých vrcholů z množiny V.

Úplný (neorientovaný) graf je prostý neorientovaný graf bez smyček, jehož každé dva různé vrcholy jsou spojeny hranou. Má-li n vrcholů, značíme jej K_n (obr. 1.3).

Bipartitní graf je takový graf G, jehož množina vrcholů $V(G)$ je disjunktním sjednocením dvou neprázdných množin S, T a platí $E(G) = W_G(S)$. Jinými slovy, každá hrana bipartitního grafu má jeden krajní vrchol v S a druhý v T. Množiny S, T nazýváme *stranami* bipartitního grafu. U orientovaného bipartitního grafu zpravidla požadujeme, aby všechny hrany byly orientovány souhlasně, tj. aby $E(G) = W^+(S)$.

Úplný bipartitní graf je takový bipartitní graf, kde každá dvojice vrcholů $s \in S$, $t \in T$ je spojena přesně jednou hranou. Úplný bipartitní neorientovaný graf, jehož strany S, T mají $m = |S|$ a $n = |T|$ prvků, značíme $K_{m,n}$ (viz obrázek 1.3).



Obrázek 1.3: Úplný graf K_5 a úplný bipartitní graf $K_{3,4}$.

Graf nazýváme *regulárním* (též *pravidelným*), mají-li všechny jeho vrcholy stejný stupeň. Graf, v němž všechny vrcholy mají stupeň k, nazýváme *k-regulárním* (též *k-pravidelným*). Příklady 3-regulárních grafů jsou nakresleny na obrázcích 1.1 a 1.2 na straně 18.

1.7 Kořenový strom

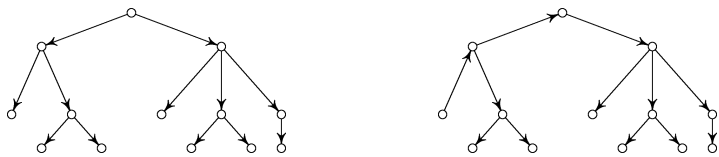
Kořenové stromy patří k nejužitečnějším a nejčastěji používaným grafům. Obrázky kořenových stromů si pro znázornění hierarchické struktury kreslí i lidé, kteří o teorii grafů nemají ani tušení.

Vzhledem k důležitosti kořenových stromů zavedeme jejich poněkud speciální terminologii již zde, třebaže podrobněji se obecnými stromy budeme zabývat v 4.2, str. 60, a kořenovými stromy pak v 5.3, str. 79.

1.7.1 Kořenový strom je orientovaný graf, v němž existuje význačný vrchol r, tzv. *kořen*, takový, že do kořene nevede žádná hrana, do každého jiného vrcholu vede

přesně jedna hrana a navíc jsou všechny vrcholy z kořene r orientovaně dostupné. Kořenový strom bývá někdy nazýván také *větvením*.

Vlastnosti kořenových stromů a definice kořene jsou uvedeny v 5.3, str. 79. Zde pouze konstatujme, že v kořenovém stromě do každého vrcholu vede z kořene přesně jedna orientovaná cesta. Na obr. 1.4 jsou dva příklady kořenových stromů. (Najděte kořeny.)

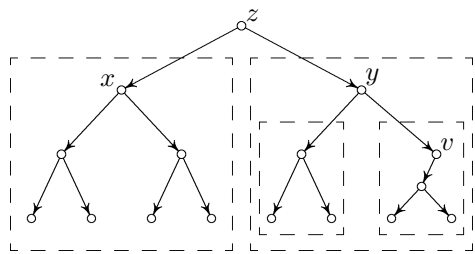


Obrázek 1.4: Kořenové stromy.

Pro kořenové stromy se často používá zvláštní „přírodopisně rodinná“ terminologie. Vede-li v kořenovém stromě hrana z vrcholu x do vrcholu y, pak vrchol x nazýváme *otcem vrcholu* y a vrchol y nazýváme *synem vrcholu* x. Někdy se dokonce o vrcholech, které mají společného otce, mluví jako o bratrech, zavádí se pojem dědečka a vnuka atd. Vrchol, který nemá žádného syna, nazýváme *listem*. Poznamenejme, že každý vrchol kořenového stromu má přesně jednoho otce s výjimkou kořene, který otce nemá.

Kořenový strom se obvykle kreslí tak, že kořen je nahoře a všechny hrany vedou shora dolů tak, aby se nekřížily. Strom na obrázku 1.4 vpravo je tedy nakreslen poněkud netradičním způsobem.

Uspořádaný kořenový strom je kořenový strom, v němž je pro každý vrchol určeno uspořádání jeho synů. Uspořádaný kořenový strom kreslíme zpravidla tak, aby synové byli uspořádáni zleva doprava.



Obrázek 1.5: Binární kořenový strom a podstromy vrcholů z a y.

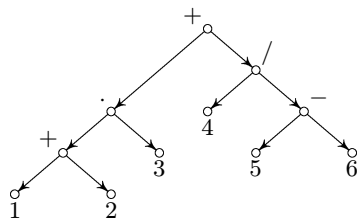
Binární kořenový strom je kořenový strom, jehož každý vrchol má nejvýše dva syny. Obvykle přitom rozlišujeme pořadí synů a ve shodě s obvyklým způsobem kreslení pak mluvíme o levém a pravém synovi. Pokud má v binárním kořenovém stromě nějaký vrchol jen jednoho syna, může to být syn pravý nebo levý. Vrchol v na obrázku 1.5 má jen levého syna.

Podstromem kořenového stromu se zpravidla nenazývá jakýkoli podgraf, který je kořenovým stromem. Je-li x vrcholem kořenového stromu G, pak *podstrom určený vrcholem* x nebo *podstrom s kořenem* x je podgraf, který je indukován množinou všech vrcholů orientovaně dostupných z vrcholu x. U binárních stromů se také mluví o *levém podstromu vrcholu* x nebo o *pravém podstromu vrcholu* x, což jsou podstromy, jejichž kořeny jsou levý nebo pravý syn vrcholu x. Na obrázku 1.5 jsou čárkovaně označeny levé a pravé podstromy vrcholů z a y. Podstrom s kořenem y je pravým podstromem vrcholu z.

Hloubku vrcholu x v kořenovém stromě definujeme jako počet hran v (jediné) cestě z kořene stromu do vrcholu x. Množinu vrcholů, které jsou ve stejné hloubce, pak nazýváme *vrstvou*. *Výšku vrcholu* x definujeme jako největší počet hran v cestě začínající v x a končící v některém listě. *Výška stromu* (jako celku) je výška jeho kořene. V kořenovém stromě na obrázku 1.5 jsou vrcholy x a y v hloubce 1, ale vrchol x má výšku 2, zatímco y má výšku 3. Výška celého stromu je 4 a všechny listy nejsou ve stejné vrstvě.

1.7.2 Aplikace kořenových stromů jsou opravdu bohaté. Znázornit kořenovým stromem rodokmen nebo nejrůznější hierarchické vztahy je velice přirozené. Důležité aplikace jsou v programování počítačů, kde se používá řada důmyslných datových struktur založených na kořenových stromech.

Na obrázku 1.6 je ukázka jiného méně triviálního použití. Aritmetický výraz je zde reprezentován uspořádaným binárním kořenovým stromem. Podobná reprezentace se používá v překladačích programovacích jazyků.



Obrázek 1.6: Reprezentace aritmetického výrazu $(1 + 2) \cdot 3 + 4/(5 - 6)$ kořenovým stromem.

V této souvislosti se lze setkat se zvláštními způsoby vyjmenování všech vrcholů kořenového stromu. Označme pro libovolný vrchol x symbolem $L(x)$ a $R(x)$ levý a pravý podstrom vrcholu x. Uspořádání vrcholů kořenového stromu se nazývá *preorder*, *postorder* nebo *inorder*, jestliže pro každý vrchol x platí, že vrcholy podstromu s kořenem x jsou v tomto uspořádání uvedeny (vyjmenovány) v tomto pořadí:

preorder:	vrchol x, všechny vrcholy z $L(x)$, všechny vrcholy z $R(x)$,
inorder:	všechny vrcholy z $L(x)$, vrchol x, všechny vrcholy z $R(x)$,
postorder:	všechny vrcholy z $L(x)$, všechny vrcholy z $R(x)$, vrchol x.

V grafu na obrázku 1.6 odpovídá pořadí inorder běžnému způsobu zápisu aritmetických výrazů. Pořadí preorder je „$+\cdot+123/4-56$“, postorder je „$12+3\cdot456-/+$“.

Pořadí postorder odpovídá pořadí při výpočtu výrazu. Jak pro daný kořenový strom tato pořadí vrcholů najít, si povíme v 3.3.13, str. 58 (prohledávání grafu do hloubky).

Dodejme, že pořadí preorder a postorder lze definovat i pro obecné kořenové stromy. Pořadí inorder má smysl jen pro stromy binární.

1.8 Matice popisující graf

1.8.1 Matice sousednosti. Nechť G je orientovaný graf. Zvolíme-li (libovolně, ale pevně) pořadí jeho vrcholů $v_1, \ldots, v_n$, můžeme grafu G přiřadit *matici sousednosti* M_G^+ řádu n předpisem

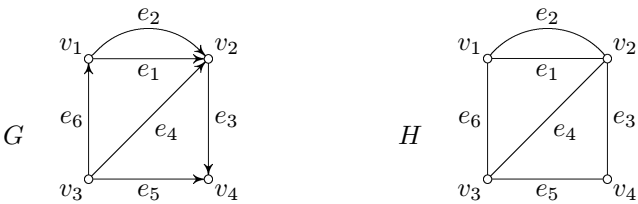
$$m_{ij}^+ = m^+(v_i, v_j) \ .$$

Pro neorientované grafy definujeme matici sousednosti M_G předpisem

$$m_{ij} = m(v_i, v_j) \ .$$

To znamená, že matice sousednosti neorientovaného grafu je vždy symetrická: $m_{ij} = m_{ji}$ pro všechna i, j. Grafy G a H na obr. 1.7 mají tyto matice sousednosti:

$$M_G^+ = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad M_H = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$



Obrázek 1.7: Grafy, jejichž matice sousednosti jsou M_G^+ a M_H.

Poznamenejme, že mají-li dva grafy, oba orientované, nebo oba neorientované, stejnou matici sousednosti, pak tyto grafy jsou navzájem izomorfní. Naopak to však neplatí: změna pořadí vrcholů má za následek stejné změny v pořadí sloupců a řádků matice sousednosti. Vzájemně izomorfní grafy tedy mohou mít různé matice sousednosti.

Neorientovaný graf a graf orientovaný, který je jeho symetrickou orientací (viz 1.2.1), mají stejné matice sousednosti. Je-li tedy matice sousednosti symetrická, pak ze samotné matice nelze poznat, zda je graf orientovaný nebo neorientovaný.

1.8.2 Cvičení. Kolik hran má orientovaný graf, který má stejnou matici sousednosti, jako graf H z obrázku 1.7? Kolik by měl hran, kdyby graf H měl navíc jednu smyčku?

1.8.3 Cvičení. Mohou dva grafy, orientovaný graf G a neorientovaný graf G', mít stejné matice sousednosti a zároveň stejné počty hran? Jak takové grafy vypadají?

1.8.4 Matice incidence. Nechť G je orientovaný graf bez smyček. Zvolíme-li (libovolně, ale pevně) nejen pořadí vrcholů $v_1, \ldots, v_n$, ale i pořadí hran $e_1, \ldots, e_m$, můžeme grafu G přiřadit *matici incidence* (též *incidenční matici*) B_G typu (n, m) předpisem

$$b_{ij} = \begin{cases} 1, & \text{jestliže } v_i \text{ je počátečním vrcholem hrany } e_j, \\ -1, & \text{jestliže } v_i \text{ je koncovým vrcholem hrany } e_j, \\ 0 & \text{v ostatních případech.} \end{cases}$$

Incidenční matice grafu G z obr. 1.7 je

$$B_G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & -1 \\ -1 & -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & -1 & 0 & -1 & 0 \end{pmatrix}.$$

Poznamenejme, že každý sloupec obsahuje přesně jednu 1 a přesně jednu -1 (proč?) a dále, že součet hodnot v i-tém řádku je roven $d^+(v_i) - d^-(v_i)$.

Pro neorientované grafy bez smyček se incidenční matice definuje předpisem

$$b_{ij} = \begin{cases} 1, & \text{jestliže } v_i \text{ je incidentní s hranou } e_j, \\ 0 & \text{v ostatních případech.} \end{cases}$$

Incidenční matice grafu H z obr. 1.7 je

$$B_H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

Poznamenejme, že součet hodnot v i-tém řádku je roven $d(v_i)$ a že každý sloupec obsahuje přesně dvě jedničky.

1.8.5 Cvičení. Předpokládejme, že graf je dán svou incidenční maticí B. Jaký význam mají jednotlivé prvky matice BB^{T}? (Symbol BB^{T} značí součin matice B s transponovanou maticí B^{T}.) Řešte zvlášť pro orientované a neorientované grafy a dejte pozor na prvky na diagonále výsledné matice.

1.9 Způsoby zadávání grafů

Pokud není graf příliš velký (zejména nemá-li mnoho hran), je zřejmě pro každého nejsrozumitelnější obrázek. Jeho snad jedinou nevýhodou je fakt, že často svádí k jednostrannému pohledu, viz např. obrázky 1.1 a 1.2 na straně 18.

Situace se ale podstatně změní, jestliže chceme pracovat s rozsáhlými grafy, které již neobsáhneme pohledem, a zejména, je-li třeba zadat graf do počítače. Používaných způsobů je celá řada a vzájemně se liší nejen vhodností pro určité typy úloh, ale i stupněm univerzálnosti. Většina těchto způsobů předpokládá, že vrcholy a popřípadě i hrany jsou očíslovány po sobě jdoucími přirozenými čísly, obvykle počínaje od jedné.

1.9.1 Seznamy vrcholů a hran. Množina vrcholů je popsána prostým výčtem (seznamem) prvků, množina hran je popsána seznamem trojic tvořených jménem hrany a jejím počátečním a koncovým vrcholem. V podstatě se jedná o úplný popis grafu podle definice. Pokud nám nezáleží na jménech hran, můžeme je vypustit a hrany popisovat pouze dvojicemi vrcholů. Neorientované grafy popisujeme formálně stejným způsobem jako orientované: pořadí vrcholů hrany volíme libovolně. To znamená, že neorientovaný graf zadáváme prostřednictvím jeho nějaké orientace.

K výhodám tohoto popisu grafu patří univerzálnost a relativní úspornost. Navíc lze tímto způsobem snadno popisovat i ohodnocené grafy: ohodnocení prostě připíšeme k hranám či vrcholům. Díky těmto výhodám je tento způsob v praxi velmi často používán, třebaže detaily provedení bývají někdy poněkud odlišné.

Příklad: Graf G z obrázku 1.7, str. 24, lze popsat např. takto:

$$\begin{array}{ll} \text{Vrcholy:} & v_1, v_2, v_3, v_4. \\ \text{Hrany:} & (e_1, v_1, v_2), \quad (e_4, v_3, v_2), \\ & (e_2, v_1, v_2), \quad (e_5, v_3, v_4), \\ & (e_3, v_2, v_4), \quad (e_6, v_3, v_1). \end{array}$$

Jestliže nám nezáleží na jménech vrcholů a hran, můžeme neorientovaný graf H z obrázku 1.7 zadat (až na izomorfismus) např. takto:

$$\begin{array}{ll} \text{Vrcholy:} & 1, 2, 3, 4. \\ \text{Hrany:} & (1, 2), \quad (2, 3), \\ & (1, 2), \quad (3, 4), \\ & (2, 4), \quad (1, 3). \end{array}$$

Všimněte si, že dvojice $(1, 2)$ se v seznamu hran vyskytuje dvakrát, neboť $m_H(1, 2) = 2$. Proto mluvíme o seznamu, a ne o množině.

1.9.2 Seznam vrcholů a seznamy okolí vrcholů. Tento způsob je vlastně úspornější variantou předchozího způsobu. Množina vrcholů je opět popsána seznamem prvků, ale hrany jsou popisovány po skupinách: pro každý vrchol x je vždy uveden seznam množiny hran $E^+(x)$. Každá hrana pak je popsána pouze svým

jménem a koncovým vrcholem, neboť počáteční vrchol x je pro celou skupinu hran společný.

Příklad popisu grafu G z obrázku 1.7:

$$\begin{array}{ll} v_1\colon & (e_1, v_2), (e_2, v_2); \\ v_2\colon & (e_3, v_4); \\ v_3\colon & (e_4, v_2), (e_5, v_4), (e_6, v_1); \\ v_4\colon & \emptyset. \end{array}$$

Je samozřejmé, že bychom místo množin $E^+(x)$ mohli uvádět též množiny $E^-(x)$. K popisu neorientovaného grafu můžeme použít popisu některé jeho orientace anebo můžeme uvádět seznamy množin $E(x)$, což je ovšem méně úsporné, neboť každá hrana (kromě smyček) je uvedena ve dvou seznamech.

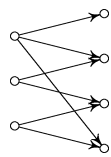
1.9.3 Matice sousednosti. Zde se využívá faktu, že matice sousednosti určuje graf až na izomorfismus. Tento způsob je matematicky elegantní, ale pro praxi méně vhodný, zejména pro grafy s relativně málo hranami, neboť matice pak obsahuje značný počet nul.

1.9.4 Matice incidence. Zde platí zhruba totéž, co v předchozím případě: Matematicky elegantní, ale neúsporné (pouze dva nenulové prvky ve sloupci). Pokud bychom udávali pouze polohy (tj. indexy) nenulových prvků ve sloupci, dostali bychom vlastně seznam hran.

1.9.5 Matice sousednosti bipartitního grafu. Tento způsob je použitelný pouze pro bipartitní grafy a je založen na faktu, že vrcholy bipartitního grafu lze uspořádat tak, aby matice sousednosti měla tvar

$$M_G^+ = \begin{pmatrix} 0 & A \\ 0 & 0 \end{pmatrix} \qquad \text{nebo} \qquad M_G = \begin{pmatrix} 0 & A \\ A^{\mathrm{T}} & 0 \end{pmatrix}$$

v závislosti na tom, zda se jedná o bipartitní graf orientovaný, nebo neorientovaný. Podmatici A pak nazýváme maticí sousednosti bipartitního grafu. Viz též obr. 1.8.



$$A = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Obrázek 1.8: Orientovaný bipartitní graf a jeho matice sousednosti.

1.9.6 Nepřímý popis grafu. V některých případech lze graf zadávat nepřímo pomocí algoritmů. Mnohdy totiž nepotřebujeme znát např. seznam hran jako celek, stačí, když pro kterýkoli vrchol v umíme nějakým algoritmem produkovat postupně všechny hrany, které z vrcholu v vycházejí. Často i množinu vrcholů nemusíme znát explicite předem: stačí, když se o existenci vrcholu dovíme díky objevení

(zpracování) hrany, která v něm končí nebo začíná. Tento přístup se používá při zpracování rozsáhlých grafů, které jsou definovány nepřímo prostřednictvím popisu nějaké situace nebo soustavy.

1.10 Zpracování grafů na počítači

Nebudeme zabíhat do detailů, ty může čtenář najít v knihách [Kučera83], [AHU74] a v některých učebnicích programování. Zmíníme se jen stručně o základních datových strukturách a uvedeme konvence pro zápis algoritmů používané v této knize.

Podotkněme, že není známa žádná univerzálně výhodná datová struktura. Pro různé algoritmy jsou různé datové struktury různě vhodné či nevhodné.

1.10.1 Datové struktury využívající matice. Matice sousednosti a matice incidence nejsou obecně příliš vhodné pro grafy s relativně málo hranami jednak pro paměťové nároky, jednak proto, že vyhledání další hrany, která vychází z téhož vrcholu, je pomalé (přeskakujeme poměrně značné množství nulových prvků v řádku).

Poněkud jiná situace je však u ohodnocených prostých grafů, které mají hodně hran, nebo také když výsledkem výpočtu má být matice: jsou-li dva vrcholy spojeny hranou, zapíšeme na příslušné místo matice ohodnocení této hrany. Pokud mezi některými dvěma vrcholy nevede hrana, zapíšeme na odpovídající místo matice nějakou hodnotu, která nemůže být ohodnocením žádné hrany, zpravidla nějakou hodně velkou nebo hodně malou konstantu.

Tuto konstantu volíme zpravidla tak, aby se s ní v příslušném algoritmu dobře pracovalo. Např. pro hledání nejkratších cest (viz 6.1.1, str. 88) použijeme k označení neexistujících hran nějaké hodně velké číslo jako náhražku nekonečna (∞). Pak vlastně algoritmus namísto s původním grafem pracuje s grafem úplným, jehož některé hrany jsou tak nepříznivě ohodnoceny, že se nemohou ve výsledném řešení vyskytnout. Vhodnost zmíněného způsobu samozřejmě velice záleží na charakteru řešené úlohy. O záludnostech těchto triků viz též poznámku 6.2.10, str. 95.

Neohodnocené grafy lze ukládat v paměti počítače také tak, že i-tý řádek matice bude obsahovat čísla následníků vrcholu i, zapsaná těsně za sebou. Pak vlastně máme pro každý vrchol tzv. kompaktní seznam jeho následníků.

Pro prosté neohodnocené grafy se také někdy používá matice sousednosti uložená jako matice bitů. Paměťové nároky jsou mnohem menší než u matice čísel, manipulace s maticí bitů je však o něco těžkopádnější.

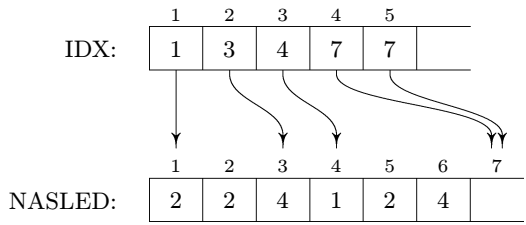
1.10.2 Seznam hran v jednorozměrných polích. Předpokládáme, že vrcholy i hrany grafu jsou očíslovány pořadovými čísly $1, 2, \ldots, n$.

Seznam hran orientovaného grafu pak lze snadno uložit do dvou polí Pv a Kv. Prvek Pv(i) pak obsahuje číslo počátečního vrcholu i-té hrany, prvek Kv(i) obsahuje koncový vrchol této hrany. Je-li graf ohodnocený, použijeme pro ohodnocení hran jednoduše další pole.

Tento způsob je vhodný zejména v programovacích jazycích, které nepodporují bohatší strukturování dat. Nevýhodou tohoto postupu je nutnost pole předem dimenzovat (určit jeho velikost) a pak pomalý přístup k následníkům či předchůdcům. Částečně si lze pomoci setříděním hran podle Pv nebo Kv. Lepší ovšem je použít ještě jedno pole Dalsi, jehož prvek Dalsi(i) bude obsahovat vždy index další hrany, která má stejný počáteční vrchol jako hrana i. Pokud další taková hrana neexistuje, může být Dalsi(i) $= -1$. Pomocí hodnot Dalsi pak lze snadno přistupovat ke všem hranám, které vycházejí z téhož vrcholu. Podobný způsob lze použít pro hrany, které končí ve stejném vrcholu a tyto způsoby lze i kombinovat.

Výše popsané použití pole Dalsi vlastně realizuje spojový seznam, kde každá položka seznamu obsahuje odkaz (spoj) na další položku.

1.10.3 Seznamy následníků v jednorozměrném poli. Tento způsob využívá dvě pole, označme je Idx a Nasled. Hodnota Idx(i) je indexem (pořadovým číslem) prvku, kterým v poli Nasled začíná seznam následníků vrcholu i. Následníci jsou v poli Nasled uloženi těsně za sebou bez jakýchkoli oddělovačů. Konec seznamu následníků vrcholu i se pozná jako začátek seznamu pro vrchol $i+1$. Kvůli ukončení posledního seznamu je v poli Idx o jeden prvek více než je vrcholů. Viz příklad na obrázku 1.9.



Obrázek 1.9: Datový popis grafu G z obr. 1.7, str. 24, pomocí seznamů následníků v jednorozměrném poli.

Je-li daný graf multigrafem, zaznamenáme rovnoběžné hrany jako opakovaný výskyt následníka. Každý prvek pole Nasled tedy můžeme pokládat za záznam o hraně, v němž je vynechán počáteční vrchol. Délka pole Nasled je rovna počtu hran. Je-li graf ohodnocený, lze pro ohodnocení použít další jednorozměrná pole (pro vrcholy i hrany).

Tento způsob uložení grafu v počítači je paměťově velmi úsporný a umožňuje rychle procházet seznamy následníků. Přístup k předchůdcům je však dost komplikovaný a pomalý.

1.10.4 Datové struktury založené na ukazatelích. Modernější programovací jazyky (např. PASCAL, C) umožňují práci s adresami datových objektů (typ ukazatel). To umožňuje pohodlnou realizaci spojových seznamů.

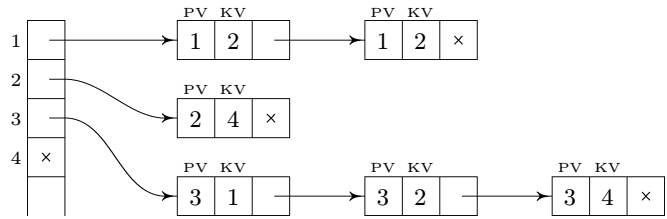
Data, která popisují vrchol či hranu, sdružujeme do záznamů (record), přičemž každý záznam obsahuje navíc ukazatel na další záznam v seznamu. Podrobnosti lze

najít v téměř každé učebnici programování.

Výhodou použití ukazatelů ve srovnání s použitím polí je fakt, že není třeba předem vědět, jak velká pole budou potřeba, a také daleko snazší změny grafu v paměti počítače. Nevýhodou je vyšší spotřeba paměti pro uložení ukazatelů. Inu, každá legrace něco stojí.

Někdy se pole a záznamy s ukazateli kombinují. Příklad je na obrázku 1.10, kde vrcholy jsou v poli, zatímco hrany jsou zaznamenány v záznamech, které jsou přístupné pomocí ukazatelů.

Stojí za zmínku, že odkazy mezi datovými objekty lze chápat jako orientovaný graf a obrázky datových struktur v učebnicích programování i obrázek 1.10 to potvrzují. Vrcholy grafu jsou záznamy (popř. jiné datové objekty) a hrany vyjadřují existenci ukazatele a odkud kam tento ukazatel ukazuje. Tento graf datové struktury je samozřejmě něco úplně jiného než graf, který je v té datové struktuře uložen.



Obrázek 1.10: Datový popis grafu G z obr. 1.7, str. 24, pomocí ukazatelů.

1.10.5 Popis algoritmů. Algoritmy v této knize jsou popisovány spíše neformálně, bez speciálního zřetele na nějaký programovací jazyk a bez použití strukturovaných příkazů. Jedinou výjimku tvoří algoritmy prohledávání do hloubky, u nichž je vedle běžného popisu uveden i zápis s využitím rekurzivní procedury (viz 3.3.9, str. 57, a 5.1.13, str. 71).

Jsou-li v algoritmech vrcholům či hranám přiřazovány nějaké hodnoty, je zápis této skutečnosti proveden tak, jako by hodnoty byly uloženy v poli (viz 1.10.2). Mluvíme tedy např. o poli VZDALENOST, přičemž VZDALENOST(v) označuje hodnotu vzdálenosti přiřazenou vrcholu v.

V žádném případě nebylo úmyslem vnucovat čtenáři implementaci založenou na polích. Spíše šlo o čtenáře, kteří nejsou zběhlí v používání ukazatelů. Zdatnější programátor si snadno zde uvedené algoritmy převede do své oblíbené datové struktury.

Jako znak přiřazení užíváme symbol :=, znak = označuje vztah rovnosti.

1.10.6 Časové nároky algoritmů a obtížnost úloh. Důležitým měřítkem kvality algoritmu je jeho rychlost. Přestože se rychlostí algoritmů nebudeme detailně zabývat, budeme u většiny algoritmů uvádět poznámky o jejich časových nárocích. Zpravidla půjde o výroky typu „čas výpočtu neroste rychleji než třetí mocnina počtu vrcholů“ apod. Budeme při tom používat toto značení:

Nechť f, g jsou dvě nezáporné funkce reálné proměnné. Jestliže existují reálné konstanty k, m takové, že pro všechna $x > m$ platí $g(x) \leq kf(x)$, pak píšeme $g = O(f)$.

Zápis $g = O(f)$ se vyslovuje jako „g je o f“. Je třeba upozornit, že v tomto zápisu má znak rovnítko trochu jiný význam než obvykle. Není to rovnost mezi čísly nebo funkcemi, je to trochu podivně zapsaný výrok o chování funkce g.

Platí např. $5x^2 + 3x = O(x^2)$ a také $x^3/1000 + 1000x^2 = O(x^3)$, ale pozor: platí také $12n \log n = O(n^2)$ nebo $2x^2 + 4 = O(x^6)$ a dokonce $2x^2 + 4 = O(x^{1000})$. Nezdá se vám to? Inu, je to tak trochu chyták, ale ověřte si, že podle výše uvedené definice je to skutečně pravda.

Řekneme-li například, že nějaký algoritmus pracuje v čase $O(n^3)$, kde n je počet vrcholů grafu, znamená to, že doba jeho práce na grafech od určité velikosti výše neroste rychleji než nějaký násobek třetí mocniny počtu vrcholů n. Přitom je však možné, že pro mnoho konkrétních případů vstupních dat (třeba i pro všechny) bude tentýž algoritmus pracovat rychleji. Dokonce i kdyby platil lepší odhad, například $O(n^2)$, byl by odhad $O(n^3)$ také pravdivý. Zápis $O(n^3)$ je prostě jen odhadem shora, který tvrdí pouze to, že to pro „dost velká n“ nebude horší než „něco“ krát n^3.

Pokud v souvislosti s časovými odhady použijeme symboly n a m, bude vždy n značit počet vrcholů a m počet hran grafu.

Algoritmy, které pracují v polynomiálním čase (tj. v čase $O(p)$, kde p je nějaký polynom), jsou obvykle pokládány za „rychlé“[1] a příslušné úlohy za „snadno řešitelné“. Je však mnoho úloh, pro které polynomiální algoritmus není znám a panuje přesvědčení, že ani žádný neexistuje. O takových úlohách budeme říkat, že jsou *NP-těžké*. Stručné vysvětlení těchto pojmů najdete v kapitole 11, str. 187.

1.11 Optimalizační úlohy

V této knize se často budeme zabývat *optimalizačními úlohami*, tj. úlohami, v nichž se hledá řešení, které je v nějakém smyslu nejlepší. V souvislosti s optimalizačními úlohami se běžně používá několik pojmů, které nyní vysvětlíme.

1.11.1 Množina přípustných řešení. V optimalizačních úlohách hledáme řešení, které je nejlepší mezi všemi tzv. *přípustnými řešeními*. Množina všech přípustných řešení však obvykle není dána seznamem svých prvků.

Obvykle máme dánu množinu, které říkáme *prostor řešení*, a seznam podmínek, kterým říkáme *omezující podmínky*. Množina přípustných řešení pak je tvořena všemi řešeními (tj. všemi prvky prostoru řešení), která splňují omezující podmínky. Je-li omezujících podmínek několik, musí je přípustné řešení splňovat všechny.

Tutéž množinu přípustných řešení lze často definovat několika způsoby.

Například při hledání nejkratších cest v grafu z vrcholu x do vrcholu y bychom mohli za prostor řešení prohlásit přímo množinu všech cest z vrcholu x do vrcholu y, omezující podmínky pak nebudeme vůbec potřebovat. Můžeme však také za prostor

[1] O záludnostech takovýchto závěrů viz 11.1.3, str. 188, nebo knihy [Kučera83, AHU74].

řešení prohlásit množinu všech sledů z x do y a pomocí omezující podmínky požadovat, že přípustné řešení musí být cestou. Nebo dokonce můžeme jako prostor řešení vzít všechny sledy (ve všech grafech) a formou omezujících podmínek požadovat, aby řešení bylo cestou v daném grafu a aby počáteční vrchol sledu byl x a koncový y.

Způsob, jakým je definována množina přípustných řešení, má velice podstatný vliv na způsob řešení úlohy, protože při řešení, chtěj nechtěj, musíme vycházet z tvaru a způsobu, jak je úloha zadána.

1.11.2 Účelová funkce. Které řešení je lepší a které horší určuje v optimalizačních úlohách tzv. *účelová funkce*, což je zobrazení, které každému přípustnému řešení přiřazuje číslo, kterému říkáme *hodnota účelové funkce*.

Optimální řešení je pak takové přípustné řešení, které má mezi všemi přípustnými řešeními nejmenší nebo naopak největší hodnotu účelové funkce. Zde záleží na charakteru úlohy – např. náklady obvykle minimalizujeme, zatímco zisk zpravidla chceme co největší.

Například při hledání nejkratších cest je účelovou funkcí součet délek všech hran, které tvoří přípustné řešení (tj. cestu z x do y), a tuto účelovou funkci minimalizujeme.

1.11.3 Typ úlohy, instance úlohy a zadání úlohy. Slovem „úloha“ bývají často označovány dvě dosti odlišné věci:

Typ úlohy určuje způsob, jak je zadána účelová funkce, zda jde o minimalizaci nebo o maximalizaci, a způsob, jak je zadána množina přípustných řešení. V tomto smyslu tedy mluvíme např. o úloze najít nejkratší cestu v grafu, čímž máme na mysli obecný typ (druh) úlohy.

Úlohy určitého typu mohou mít spousty různých *instancí* (konkrétních případů), které se liší v konkrétních hodnotách číselných parametrů, grafem, ve kterém se hledá řešení, apod.

Instance úlohy je konkrétní případ úlohy, který je zadán tak konkrétně, že má smysl ji řešit (počítat) nebo se o to alespoň pokoušet. *Zadání úlohy* jsou vlastně data (mnohdy číselná), kterými se odlišují jednotlivé instance úlohy.

Například pro úlohu (tj. typ úlohy) najít v grafu nejkratší cestu mezi dvěma vrcholy musí zadání úlohy obsahovat popis konkrétního grafu, ohodnocení hran jejich délkami a počáteční a koncový vrchol hledané cesty. Teprve máme-li k danému typu úlohy tyto konkrétní údaje, má smysl začít počítat.

Naopak obecný algoritmus pro řešení úloh daného typu (tj. pro řešení všech instancí) lze vymýšlet na základě znalosti typu úlohy, tj. i bez konkrétního zadání.

1.12 Množiny, relace a zobrazení

Jde o základní pojmy a každý čtenář se s nimi jistě už setkal. Mnohé z těchto pojmů souvisí s grafy velmi těsně (relace a zobrazení lze např. znázorňovat pomocí grafů).

1.12.1 Množiny. Je-li x prvkem množiny A, značíme to $x \in A$. Dvě množiny pokládáme za stejné, obsahují-li obě tytéž prvky.

Množinu všech reálných čísel značíme $\mathbb{R}$ a množinu všech přirozených čísel značíme $\mathbb{N}$. Nulu pokládáme za přirozené číslo, tj. $0 \in \mathbb{N}$.

Jsou dva důležité způsoby, jak popsat (sdělit někomu) množinu. Popis množiny výčtem jejích prvků lze použít k popisu konečných množin. Připomeňme, že pořadí prvků ani jejich opakovaný výskyt přitom nemají vliv. Platí tedy například $\{a, b, c, b, b, a, b\} = \{b, c, a\}$.

Popis množiny pomocí vlastnosti jejích prvků umožňuje popsat i nekonečné množiny. Zápisem $\{x \mid x \text{ má vlastnost } V\}$ označujeme množinu všech prvků, které mají vlastnost V.

Množina A je *podmnožinou* množiny B, jestliže každý prvek množiny A je také prvkem množiny B. Tento vztah značíme $A \subseteq B$. Každá množina je podmnožinou sebe sama.

Prázdná množina $\emptyset$ je množina, která nemá žádný prvek. Prázdná množina je jen jedna (neboť všechny prázdné množiny jsou si rovny) a je podmnožinou každé množiny, dokonce i podmnožinou sebe sama.

Prvkem množiny může být i jiná množina, dokonce i prázdná. Tedy například množina $\{\{a, b, c\}, b, \emptyset, \{\emptyset\}\}$ má čtyři prvky: tříprvkovou množinu $\{a, b, c\}$, prvek b, prázdnou množinu $\emptyset$ a jednoprvkovou množinu $\{\emptyset\}$.

Množinové operace průniku ($A \cap B$) a sjednocení ($A \cup B$) zná asi každý. Rozdíl množin $A \setminus B$ je množina všech prvků, které leží v množině A a neleží v B. Tedy např. $\{a, b, c, d\} \setminus \{a, c\} = \{b, d\}$, ale pozor, $\{a, b\} \setminus \{c, d\} = \{a, b\}$.

Je-li $A \cap B = \emptyset$, pak říkáme, že množiny A, B jsou *disjunktní*.

Počet prvků konečné množiny A značíme $|A|$.

1.12.2 Kartézský součin. Jsou-li A, B dvě množiny, pak symbolem $A \times B$ označujeme jejich *kartézský součin*, tj. množinu všech uspořádaných dvojic (a, b) takových, že $a \in A$ a $b \in B$. Jinak řečeno, kartézský součin $A \times B$ je množina, která se skládá ze všech možných uspořádaných dvojic vytvořených tak, že prvý prvek dvojice vezmeme z A a druhý z B.

Máme-li dvě konečné množiny A, B, které mají m a n prvků, pak počet prvků kartézského součinu $A \times B$ je roven součinu (čísel) mn.

Kartézské součiny lze vytvářet i z většího počtu množin. Například součin $A \times \times B \times C \times D$ se skládá z uspořádaných čtveřic.

Speciálním případem kartézského součinu je *kartézská mocnina*, kde násobíme množinu samu se sebou. Např. $A \times A \times A = A^3$ a $B \times B = B^2$.

Známým příkladem kartézské mocniny je rovina $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$, jejíž prvky (body) určujeme pomocí dvou souřadnic. V této knize však kartézský součin potřebujeme zejména k definici pojmu relace.

1.12.3 Relace je matematickým vyjádřením vztahu. Relaci formálně definujeme jako podmnožinu kartézského součinu.

Takto definovaný matematický pojem relace se nesnaží zachytit podstatu nebo příčinu vztahu, ale pouze eviduje, které objekty v daném vztahu jsou a které nejsou.

Např. relace „býti otcem“ pouze eviduje uspořádané dvojice lidí, z nichž prvý je otcem druhého. Vůbec se přitom nestaráme o okolnosti, např. kdo z nich má druhého rád apod. Tyto okolnosti, pokud bychom chtěli, by bylo třeba modelovat nějak jinak, např. pomocí další relace.

Je-li relace $R \subseteq A_1 \times A_2 \times \ldots \times A_n$ podmnožinou kartézského součinu n množin, pak číslo n nazýváme *aritou* relace R. Velmi důležité jsou *binární* relace, které mají aritu 2.

Skutečnost, že n-tice $(a_1, a_2, \ldots, a_n)$ je v relaci R, lze vyjádřit několika způsoby. Zápis $(a_1, a_2, \ldots, a_n) \in R$ je v souladu s matematickou definicí relace. V některých aplikacích se totéž vyjadřuje tzv. prefixním zápisem $R(a_1, a_2, \ldots, a_n)$. Pro binární relace se často používá takzvaný infixní zápis xRy. Je to obvyklé například u relací $<, >, \leq, =, \subseteq, \in$ apod., kde píšeme např. $1 < 2$ raději než $(1, 2) \in <$ nebo $<(1, 2)$, třebaže i tyto zápisy jsou správné.

1.12.4 Binární relace na množině je relace, která je podmnožinou druhé kartézské mocniny dané množiny. Je-li například A množinou lidí, pak vztah mezi rodiči a dětmi můžeme modelovat pomocí relace $R \subset A^2$. Prvky této relace budou všechny uspořádané dvojice (x, y), kde x a y jsou lidé a y je dítětem x. Jiným příkladem binární relace je relace $\leq$ na množině čísel.

Pozorný čtenář si jistě všimnul, že binárními relacemi na množině můžeme vyjádřit přesně totéž, co prostými orientovanými grafy. Proto také pro binární relace používáme stejné obrázky a stejné vyjádření pomocí matice sousednosti.

Omezení na prosté grafy je podstatné. Multigraf pomocí relace vyjádřit nejde, neboť relace je (formálně vzato) množina uspořádaných dvojic, a proto nemůže obsahovat žádnou dvojici více než jedenkrát.

Mimochodem, v mnohých knihách o teorii grafů se definuje množina hran orientovaného grafu jako binární relace na množině vrcholů (viz 1.3.5). Taková definice však neumožňuje pracovat s multigrafy.

1.12.5 Vlastnosti binárních relací. Binární relaci R na množině A nazýváme

symetrickou, jestliže $xRy \iff yRx$ pro všechna $x, y \in A$,
reflexivní, jestliže xRx platí pro všechna $x \in A$,
tranzitivní, jestliže $(xRy \;\&\; yRz) \Rightarrow xRz$ pro všechna $x, y, z \in A$,
antisymetrickou, jestliže $(xRy \;\&\; yRx) \Rightarrow x = y$ pro všechna $x, y \in A$.

Graf reflexivní relace má v každém vrcholu smyčku. Graf symetrické relace je symetrický. Symetrickou relaci tedy lze vyjádřit také neorientovaným grafem. Relace $\leq$ a relace „býti dělitelem“ na množině přirozených čísel jsou obě tranzitivní a antisymetrické.

1.12.6 Ekvivalence (též ekvivalenční relace) je relace, která je reflexivní, symetrická a tranzitivní.

Příkladem ekvivalence na množině lidí je relace „míti stejnou krevní skupinu“. Jiným příkladem je relace „míti stejnou hodnotu“ na množině zlomků, kterou definujeme takto: zlomky a/b a x/y mají stejnou hodnotu právě tehdy, když $ay = bx$

(nulový jmenovatel zde nepřipouštíme). Ověřte, že tato relace je opravdu reflexivní, symetrická i tranzitivní.

Máme-li ekvivalenční relaci R na množině A, můžeme každému prvku $x \in A$ přiřadit množinu všech prvků, které jsou s ním v relaci, tj. množinu $\{y \in A \mid xRy\}$. Tyto množiny nazýváme třídami ekvivalence R nebo též ekvivalenčními třídami.

Ekvivalenční třídy jsou navzájem disjunktní (různé třídy nemají společné prvky) a jejich sjednocením je celá množina A. Takovouto soustavu podmnožin nazýváme rozkladem množiny A na třídy ekvivalence R.

1.12.7 Rozklad množiny A je systém neprázdných podmnožin $A_1, A_2, \ldots, A_k$ množiny A takový, že množiny $A_1, A_2, \ldots, A_k$ jsou vzájemně disjunktní (nemají společné prvky) a platí $A_1 \cup A_2 \cup \ldots \cup A_k = A$. Množiny $A_1, A_2, \ldots, A_k$ nazýváme třídami rozkladu nebo rozkladovými třídami.

Máme-li rozklad množiny A, můžeme na množině A definovat relaci „býti ve stejné rozkladové třídě“. Lze dokázat (a je to snadné cvičení), že takto definovaná relace je ekvivalencí a že rozklad této relace na ekvivalenční třídy je totožný s oním výchozím rozkladem.

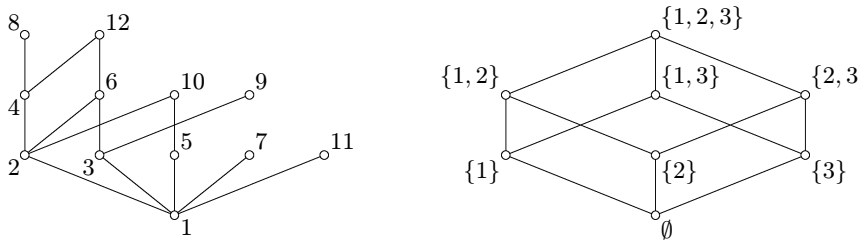
Hlavní význam ekvivalenčních relací je v jejich těsném vztahu k rozkladům.

1.12.8 Částečné uspořádání je relace, která je reflexivní, tranzitivní a antisymetrická.

Nejznámějším příkladem je uspořádání čísel podle velikosti, ale není to příklad charakteristický.

Zajímavějším příkladem částečného uspořádání na množině přirozených čísel je relace „býti dělitelem“. Všimněte si, že existují dvojice čísel, z nichž ani jedno není dělitelem druhého. Takové dvojice nazýváme neporovnatelnými.

Jiným příkladem je relace „býti podmnožinou“ na množině všech podmnožin nějaké pevně zvolené množiny A. Všimněte si, že i zde existují vzájemně neporovnatelné prvky (totiž podmnožiny množiny A).



Obrázek 1.11: Hasseovy diagramy částečných uspořádání „býti dělitelem“ a „býti podmnožinou“.

Částečné uspořádání na konečné množině lze (jako každou konečnou binární relaci) vyjádřit orientovaným grafem. Tento graf neobsahuje cykly, ale pro účely názorného kreslení má zbytečně mnoho hran. Je totiž zbytečné kreslit hrany, které lze odvodit z tranzitivity. Navíc bývá zvykem takovýto graf kreslit bez šipek, ale tak,

aby všechny hrany směřovaly nahoru. Takový obrázek se nazývá *Hasseův diagram*, příklady jsou nakresleny na obr. 1.11.

Poznamenejme, že graf, který je nakreslen Hasseovým diagramem, je ve skutečnosti orientovaný, ale orientace hran je namísto šipek vyjádřena výškou vrcholů na obrázku. Tento (orientovaný) graf je tzv. tranzitivní redukcí grafu částečného uspořádání, podrobněji viz 5.4, str. 81.

1.12.9 Lineární uspořádání je částečné uspořádání R, které má navíc tu vlastnost, že pro každé dva prvky x, y platí buď xRy, nebo yRx. V lineárním uspořádání se tedy nemůže stát, že by nějaké dva prvky byly vzájemně neporovnatelné.

Typickým příkladem lineárního uspořádání je uspořádání čísel podle velikosti nebo uspořádání slov ve slovníku podle abecedy.

Hasseův diagram lineárního uspořádání má velmi speciální tvar: všechny vrcholy leží na jediné cestě z nejnižšího vrcholu do nejvyššího.

Máme-li lineární uspořádání $\preceq$ na n-prvkové množině A, pak můžeme prvky množiny A očíslovat přirozenými čísly $\{1, 2, \ldots, n\}$ tak, že $a_i \preceq a_j \iff i \leq j$. Také naopak, máme-li prvky nějaké konečné množiny A očíslovány přirozenými čísly $\{1, 2, \ldots, n\}$, pak toto očíslování určuje lineární uspořádání na množině A.

1.12.10 Zobrazení. V této knížce vystačíme s intuitivní představou zobrazení jako předpisu f, který každému prvku nějaké množiny A přiřazuje přesně jeden prvek nějaké množiny B. Tuto skutečnost vyjadřujeme zápisem $f : A \to B$. Množinu A nazýváme *definičním oborem* a množinu B *oborem hodnot* zobrazení f. Říkáme také, že zobrazení f *zobrazuje množinu A do množiny B*.

Prvek z množiny B, který je zobrazením f přiřazen prvku $x \in A$, značíme $f(x)$ a nazýváme jej *obrazem prvku x v zobrazení f*. Je-li $X \subseteq A$ podmnožina definičního oboru A, pak symbolem $f(X)$ značíme *obraz množiny X v zobrazení f*, tj. množinu $\{y \mid \text{existuje } x \in X \text{ takové, že } y = f(x)\}$.

Slovo „funkce“ budeme používat pro zobrazení do množiny reálných čísel.

Formálně se zobrazení $f : A \to B$ definuje jako relace $f \subseteq A \times B$, která má tu vlastnost, že pro každý prvek $x \in A$ existuje přesně jeden prvek $y \in B$ takový, že $(x, y) \in f$ (nebo v infixním zápisu xfy).

Díky tomu, že zobrazení je vlastně relace, můžeme si zobrazení, které má konečný definiční obor A, kreslit jako orientovaný graf, v němž z každého vrcholu z množiny A vychází přesně jedna hrana (tj. každý vrchol z A má výstupní stupeň 1).

Zobrazení $f : A \to B$ nazýváme *prostým* zobrazením, jestliže pro libovolné dva prvky $x, y \in A$ platí $x \neq y \Rightarrow f(x) \neq f(y)$. V grafu prostého zobrazení tedy do každého vrcholu množiny B vede nejvýše jedna hrana.

Zobrazení $f : A \to B$ nazýváme zobrazením *na*, jestliže každý prvek oboru hodnot B je obrazem nějakého prvku definičního oboru A. V grafu zobrazení na tedy do každého vrcholu množiny B vede alespoň jedna hrana.

Zobrazení nazýváme *vzájemně jednoznačným*, je-li prosté a zároveň na. Ke vzájemně jednoznačnému zobrazení f vždy existuje tzv. *inverzní* zobrazení f^{-1}, které má tu vlastnost, že $f^{-1}(f(x)) = x$ pro všechna $x \in A$ a také $f(f^{-1}(y)) = y$ pro všechna $y \in B$.

Vzájemně jednoznačné zobrazení mezi dvěma množinami existuje právě tehdy, když tyto množiny mají stejnou mohutnost (stejný počet prvků). Formálně vzato, relace „míti stejnou mohutnost“ se definuje právě takto, tj. pomocí existence vzájemně jednoznačného zobrazení.

Kapitola 2

Aplikace úloh o cestách

V této kapitole uvádíme příklady úloh o cestách. Tento soubor příkladů není ani úplný ani reprezentativní. Je míněn jako inspirace při řešení vlastních úloh.

2.1 Druhy modelů a úloh

Pojem cesty v grafu je velice přirozený a má dobrý smysl ve většině reálných situací, které lze popsat grafem. Také naopak, poměrně široký okruh úloh z nejrůznějších oblastí lze s výhodou převést na úlohu najít cestu (popř. sled) s vhodnými vlastnostmi ve vhodně definovaném grafu.

Pro některé praktické úlohy se tvar grafu i vlastnosti, které má mít hledaná cesta, samy nabízejí. Typickým příkladem je hledání nejkratší, časově nejméně náročné, nejlevnější nebo vůbec jakékoli cesty v silniční síti spojující dvě daná města.

V řadě příkladů je však souvislost původního praktického problému s cestami v grafu méně zřejmá. Cílem této kapitoly je ukázat na několika příkladech, jak lze od původního problému dospět ke grafové úloze.

Přes značnou různorodost úloh, které vedou na hledání cest, lze najít typy úloh, jejichž grafový model je založen na podobných principech.

1. Hledání posloupností operací (akcí), které vedou k nějakému cíli. Jsou užívány v podstatě dva způsoby modelování těchto problémů:
   (a) Vrcholy odpovídají stavům nějakého procesu, hrany odpovídají změnám stavů, popř. operacím, akcím, které k těmto změnám vedou. Ohodnocení hran může vyjadřovat námahu (náklady, čas, spotřebovanou energii) potřebnou k provedení změny stavu.
   (b) Vrcholy odpovídají operacím, akcím a hrany vyjadřují možnost bezprostřední návaznosti akcí.
2. Časové výpočty týkající se paralelně probíhajících operací (síťové grafy).
3. Hledání statických konfigurací, které lze definovat pomocí cest v grafu.

Grafové úlohy, o jejichž aplikace nám zde půjde, se liší podle toho, zda chceme pouze zjistit, zda nějaká (jakákoli) cesta existuje, nebo nalézt nějakou cestu, nebo zjistit délku nejkratší (popř. nejdelší) cesty, nebo nejkratší (popř. nejdelší) cestu najít, a to všechno buď jen mezi dvěma danými vrcholy, nebo z daného vrcholu do všech ostatních vrcholů, nebo mezi všemi dvojicemi vrcholů grafu. Řešením těchto úloh se budeme zabývat v kapitolách 3 a 6.

Aplikace některých příbuzných úloh lze najít v kapitolách 7, 10 a 12.

Poznamenejme, že pro řešení grafových úloh není vždy nutné příslušný graf explicite sestrojit, zapsat nebo nakreslit. V některých případech je naopak vhodnější použít pouze myšlenky a metody příslušného grafového algoritmu a vlastní řešení se může odehrávat v řeči (a datové struktuře) původního praktického problému. Platí to zejména o úlohách týkajících se cest, kde mnohdy ani nemusíme celý graf předem znát, o jeho vrcholech a hranách se můžeme dovídat teprve až v průběhu řešení.

2.2 Změny stavů a posloupnosti operací

2.2.1 Odměřování vody (hlavolam). Jste na břehu jezera, máte k dispozici třílitrovou a pětilitrovou nádobu a vaším úkolem je nabrat do větší nádoby přesně čtyři litry vody. Další pomocné nádoby nemáte.

Tuto úlohu lze převést na hledání cesty ve speciálním grafu tímto způsobem: Nejprve si uvědomme, že dílčí operace má smysl provádět pouze tak, aby se některá z nádob zcela naplnila nebo zcela vyprázdnila. Jinak totiž nelze o přelitém obsahu vody vůbec nic tvrdit. Z toho plyne, že v každém stadiu přelévání vody bude v každé nádobě celočíselný počet litrů vody. To znamená, že náš „přelévací systém“ se může nacházet vždy pouze v jednom ze 24 možných stavů charakterizovaných počty litrů vody v jednotlivých nádobách. Tyto stavy budou vrcholy našeho grafu. Budeme je označovat uspořádanými dvojicemi (i, j), kde i je množství vody v pětilitrové nádobě a j množství vody v třílitrové nádobě. Hrany grafu budou vyjadřovat možnost přímého přechodu z jednoho stavu do druhého. Z vrcholu (i, j) povede hrana do vrcholu (k, l), jestliže stav (k, l) můžeme obdržet ze stavu (i, j) naplněním nebo vyprázdněním některé nádoby nebo přelitím co největšího množství z jedné nádoby do druhé (tj. vždy se musí některá nádoba vyprázdnit nebo zcela naplnit).

Je zřejmé, že do některých vrcholů nevede žádná hrana. Jsou to ty vrcholy (tj. stavy), kdy ani jedna z nádob není plná ani prázdná. Tyto vrcholy (stavy) nejsou dosažitelné z žádného jiného stavu, můžeme je tedy vypustit, aniž by to mělo vliv na řešení úlohy. Takto získaný graf je nakreslen na obr. 2.1.

Původní úlohu o přelévání vody jsme tak převedli na úlohu o nalezení (jakékoli, ale raději nejkratší) cesty z vrcholu $(0, 0)$ do vrcholu $(4, 0)$, a to v následujícím smyslu: Každému přelévání odpovídá (a to dokonce jednoznačně) nějaký sled. Cesta v grafu pak odpovídá „rozumnému“ přelévání, v němž se žádný stav neopakuje.

2.2.2 Cvičení. Následující úlohy řešte pomocí grafového modelu: najděte vhodný graf, zformulujte grafovou úlohu a promyslete, jak z řešení grafové úlohy získáte řešení původního problému.

1. Pomocí grafu na obr. 2.1 najděte nejrychlejší způsob přelévání, tj. s nejmenším počtem jednotlivých přelití.

2. Pokuste se najít způsob přelévání takový, abyste co nejméně vody vylévali zpět do jezera. Jak bude vypadat příslušný (možná ohodnocený) graf a jakou úlohu o cestách bude třeba řešit?

3. Při některých přelévacích operacích je třeba zvýšená pozornost. Jsou to ty operace, kde se jedna nádoba zcela naplní, ale druhá se tím nevyprázdní. Najděte způsob přelévání, který má nejmenší počet takovýchto „rizikových" přelití.

4. Je možno odměřit 5 litrů vody pomocí čtyřlitrové a šestilitrové nádoby, jestliže opět nemáte další pomocné nádoby?

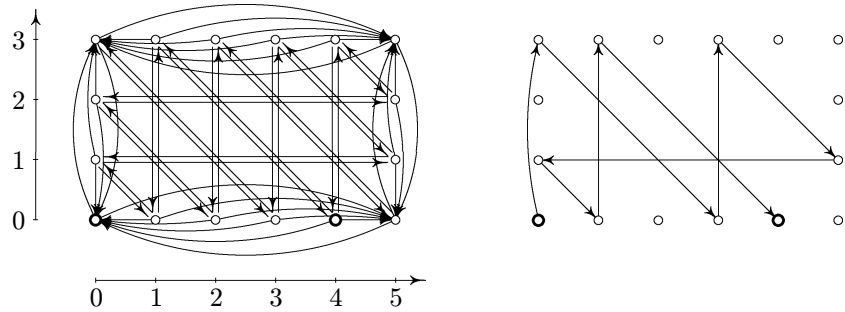
2.2.3 Hledání v jízdním řádu. Řešme následující úlohu „ze života": Najděte v jízdním řádu vlakové, autobusové či kombinované spojení z místa A do místa B. Kdy nejpozději musíme odjet z místa A, abychom byli v předem daném okamžiku v místě B? Kdy nejdříve můžeme být v místě B, jsme-li teď v místě A?

Tyto otázky můžeme řešit v následujícím „časoprostorovém" grafu, ve kterém se pro jednoduchost omezíme na železnici:

Sestrojme síť, která má jeden vrchol pro každý „zajímavý bod v časoprostoru", tj. pro každé nádraží a každý okamžik příjezdu či odjezdu vlaku na tomto nádraží. Na obr. 2.2 je příklad části takové sítě. Vrcholy, které představují totéž nádraží v různých okamžicích, jsou nakresleny vždy na vodorovné přímce.

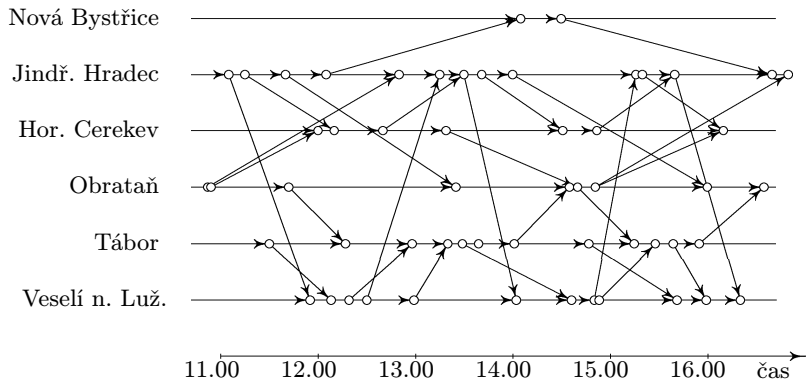
V grafu jsou dva druhy hran: Hrany, které vyjadřují možnost čekání ve stanici (kresleny vodorovně), a hrany, které vyjadřují možnost jízdy vlakem (kresleny šikmo). Uvedené úlohy o vlakovém spojení pak přirozeným způsobem odpovídají úlohám o cestách v grafu. Jiný grafový model viz 2.2.5.

Je samozřejmé, že nakreslit graf celé železniční sítě se všemi vlaky by bylo obtížné. Najít *nějaké* spojení v jízdním řádu je daleko snazší. Uvědomte si však, že i když hledáte přímo v jízdním řádu a zapomenete na teorii grafů, pokud postupujete při hledání systematicky, lze váš postup při troše dobré vůle pokládat



Obrázek 2.1: Graf k úloze o přelévání vody a jedno z možných řešení.

za hledání cesty v grafu. Přitom většinou některé trasy a priori vylučujete, což je dáno vašimi zeměpisnými znalostmi. Lze tedy tvrdit, že hledání v jízdním řádu je vlastně hledání cest ve speciálním orientovaném grafu a že jízdní řád je vlastně jakýmsi formalizovaným číselným zadáním tohoto grafu.



Obrázek 2.2: Část grafu k hledání v jízdním řádu.

Dodejme, že počítačové programy, které hledají spojení v jízdním řádu, bývají ve své podstatě založeny na grafově teoretických modelech.

2.2.4 Cvičení. Představte si, že je neděle 11 hodin dopoledne, jste na železniční stanici Nová Bystřice a máte se co nejrychleji dostat vlakem do Šluknova. Kdy nejdříve tam můžete být a kudy pojedete?

Použijte klasický „papírový“ jízdní řád, nikoli počítačový program, který najde (nějaké) spojení za vás. Hledejte libovolným způsobem, bez určité systematičnosti však máte malou naději na úspěch a žádnou záruku, že nalezené spojení je opravdu nejrychlejší. Srovnejte svou metodu s metodami, uvedenými v kapitolách 3 a 6.

2.2.5 Jiný přístup k hledání v jízdním řádu. K úloze o hledání v jízdním řádu lze přistoupit také jinak: Vrcholy grafu odpovídají vlakům a hrany vyjadřují možnost přestupování. Dále je třeba přidat dva vrcholy, jeden za výchozí a jeden za cílovou stanici, a hranami vyznačit, kterými vlaky je pro nás přijatelné odjet z výchozí stanice a kterými přijet do cílové stanice (kdy nejdříve chceme odjíždět a kdy nejpozději chceme být v cíli).

Tento druhý graf je tedy závislý na místech a časech odkud, kam a kdy chceme cestovat. Jsou-li tato místa a časy fixovány, jsou oba přístupy rovnocenné: oběma lze dosáhnout téhož. První přístup však dovoluje snadněji měnit výchozí a cílovou stanici i čas, kdy chceme cestu zahájit či ukončit. Kromě toho je graf na obr. 2.2 názornější a jeho konstrukce podle jízdního řádu je jednodušší.

Porovnejte oba přístupy na cvičeních 2.2.4 a 2.2.6. Další modely založené na podobných principech jsou uvedeny v 8.2.9, str. 140.

2.2.6 Cvičení. Jak by bylo třeba upravit grafové modely v 2.2.3 a 2.2.5, aby cesty v příslušných grafech odpovídaly pouze takovým spojením, která by nebyla ohrožena půlhodinovým zpožděním vlaku?

2.2.7 Cvičení. Nalezněte alespoň dva způsoby, jak popsat grafem silniční síť města s jednosměrnými ulicemi, zákazy odbočení a přikázanými směry jízdy tak, aby v takto vzniklém grafu bylo možno vyhledávat jízdní trasy pro osobní automobil, které respektují zákazy odbočení, přikázané směry jízdy a jednosměrnost ulic.

Graf by měl být udělán tak, aby jakýkoli (orientovaný) sled v tomto grafu odpovídal možnosti legální jízdy automobilu.

Nakreslete graf zachycující vám dobře známou část města a výsledek předložte ke kontrole zainteresovaným kolegům. Chyby, které najdou, se pokuste odstranit.

2.2.8 Nákladní parník. Máme n přístavů očíslovaných $1, 2, \ldots, n$ a nákladní parník. Pro každou dvojici přístavů i, j známe náklady $c(i, j)$ na plavbu z přístavu i do přístavu j. Dále o některých dvojicích přístavů i, j víme, že z přístavu i do přístavu j lze dopravit náklad a že za jeho převezení dostaneme zaplaceno $d(i, j)$ peněz. Předpokládáme, že náš parník má svobodnou volbu, kam popluje a co poveze. Dále předpokládáme, že v naší části oceánu neoperují žádné konkurenční lodě, které by nás mohly připravit o vyhlédnutý výdělek. Naším úkolem je dostat parník z přístavu x do přístavu y finančně co nejvýhodnějším způsobem.

Úlohu lze řešit takto: Nejprve položíme $d(i, j) = 0$, jestliže z přístavu i do přístavu j není co dovézt. Sestrojíme úplný orientovaný graf o n vrcholech $v_1, v_2, \ldots, v_n$. Délku hrany z vrcholu v_i do vrcholu v_j zvolme $a(i, j) = c(i, j) - d(i, j)$. Délky hran tedy mohou být i záporné. Nyní hledáme nejkratší cestu z vrcholu v_x do vrcholu v_y. Jsou-li zásoby zboží, které má být převezeno, dostatečně veliké, má dobrý smysl také úloha o nalezení cyklu se zápornou délkou. Opakované projíždění tohoto cyklu totiž zajistí opakované zisky.

2.2.9 Cvičení. Problémy týkající se známé Rubikovy kostky lze také převést na hledání cest v grafu. Zvažte skutečnou použitelnost tohoto přístupu.

2.3 Paralelně probíhající činnosti

2.3.1 Šíření poruch. Předpokládejme, že máme nějaké poruchové zařízení složené z n navzájem propojených dílů. Správná činnost každého dílu závisí na správné funkci některých jiných dílů. Předpokládejme, že víme o všech těchto závislostech. Navíc, závisí-li funkce dílu i na funkci dílu j, známe čas $t(i, j)$, po který nejvýše může díl i pracovat po poruše dílu j. Úkolem je zjistit, za jakou dobu po poruše dílu x přestane fungovat díl y. Další úlohy mohou být: zjistit, za jak dlouho budou porouchány všechny díly, zjistit, které díly se vůbec porouchají vlivem poruchy dílu x, které díly se určitě porouchají do 10 minut po poruše na dílu x (pak třeba přijde opravář).

Tyto úlohy lze řešit například takto: Sestrojme graf, který má n vrcholů, každý z nich odpovídá jednomu dílu našeho zařízení. Ke každému vztahu závislosti dílu

i na dílu j vytvoříme v grafu orientovanou hranu z vrcholu v_i do vrcholu v_j. Tuto hranu ohodnotíme časem $t(i,j)$, po který může díl j pracovat po poruše dílu i.

Vznikne-li v některém díle porucha, šíří se její vliv zpravidla několika směry současně. Máme-li zjistit, za jak dlouho po poruše na díle x přestane fungovat díl y, musíme vzít v úvahu všechny možné posloupnosti příčin a následků, které začínají poruchou dílu x a končí poruchou dílu y. Každé takové posloupnosti odpovídá v našem grafu cesta začínající ve vrcholu v_x a končící ve vrcholu v_y. Délka cesty, tj. součet časů, kterými jsou ohodnoceny její hrany, odpovídá času, který může nejvýše uplynout mezi poruchou dílu x a jí vyvolanou poruchou dílu y. Takových posloupností příčin a následků (a jim odpovídajících cest) může být ovšem mnoho, přičemž každá z nich shora omezuje dobu správné činnosti dílu y po poruše na dílu x. Skutečná doba fungování dílu y po poruše na dílu x je tedy nejvýše rovna délce nejkratší cesty z vrcholu v_x do vrcholu v_y v našem ohodnoceném grafu.

Ostatní uvedené otázky lze řešit podobně: Poslední díl přestane fungovat za dobu, která odpovídá maximu ze všech délek (tj. časů) nejkratších cest do všech ostatních vrcholů. Množina všech dílů, které se porouchají vlivem poruchy na dílu x, odpovídá množině všech vrcholů orientovaně dostupných z vrcholu v_x. K zodpovězení poslední otázky stačí znát množinu všech vrcholů, do nichž délka (tj. čas) nejkratší cesty nepřesáhne 10 minut.

2.3.2 Síťové grafy. Máme za úkol uskutečnit nějaký složitý projekt, který se skládá z mnoha dílčích činností, např. stavbu domu nebo továrny, vývoj a výrobu nějakého unikátního zařízení nebo let člověka na Mars. Předpokládejme, že o každé dílčí činnosti předem víme, jak dlouho bude trvat (nebo máme alespoň kvalifikovaný odhad doby trvání), a známe seznam činností, které musí být ukončeny, aby tato činnost mohla začít. Např. než začneme stavět střechu, musí být hotovy obvodové zdi, než začneme stavět zdi, musí být hotovy základy. Zajímá nás, kdy nejpozději budeme muset jednotlivé činnosti zahajovat, abychom neprodloužili dobu trvání celého projektu, a jakou tedy máme u jednotlivých činností časovou rezervu.

Popsanou situaci můžeme znázornit tzv. *síťovým grafem* dvěma způsoby: činnosti jsou znázorněny buď vrcholy, nebo hranami.

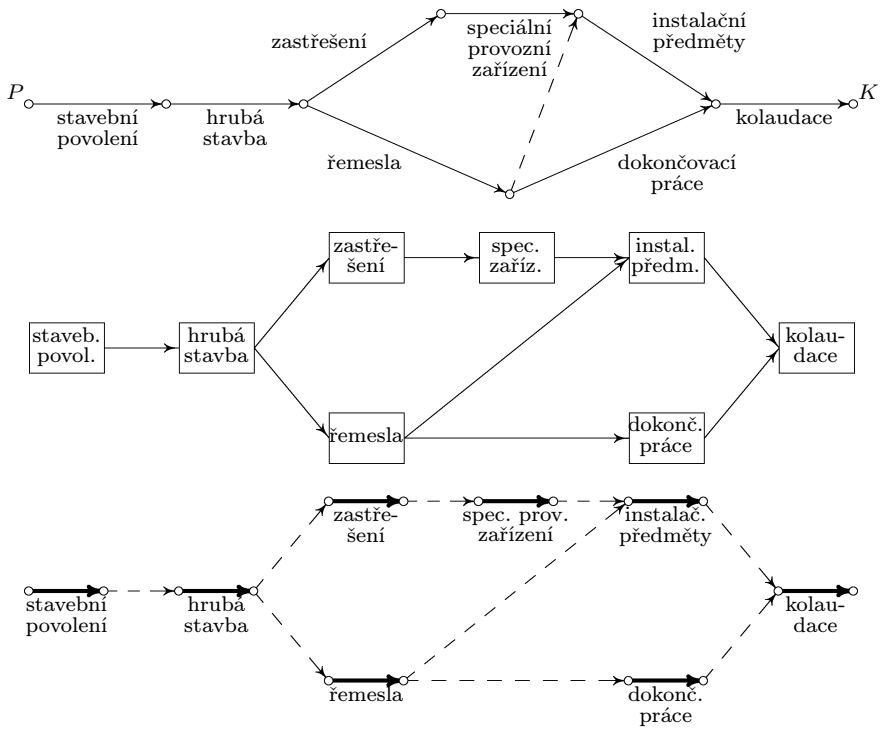
V síťovém grafu *s ohodnocenými vrcholy* jsou činnosti znázorněny vrcholy grafu, každý vrchol je ohodnocen dobou trvání činnosti. Hrany grafu vyjadřují návaznosti: z vrcholu i do vrcholu j vede hrana, jestliže ukončení činnosti i je podmínkou pro zahájení činnosti j. Viz příklad na obr. 2.3 uprostřed.

V síťovém grafu *s ohodnocenými hranami* jsou činnosti znázorněny hranami grafu, každá hrana je ohodnocena dobou trvání činnosti. Návaznosti činností jsou vyjádřeny nepřímo přes návaznosti hran ve vrcholech: jestliže dvě hrany e_1, e_2 na sebe navazují, tj. platí-li $\textsc{Kv}(e_1) = \textsc{Pv}(e_2)$, pak činnost odpovídající hraně e_1 musí být ukončena před zahájením činnosti odpovídající hraně e_2. Často se stává, že k zajištění všech požadovaných návazností je nutno použít fiktivní činnosti s nulovou dobou trvání, viz například čárkovaná hrana na obr. 2.3 nahoře.

V obou případech je síťový graf acyklický (tj. neobsahuje cykly), jinak by činnosti tvořící cyklus nebylo možno vůbec zahájit. K síťovému grafu obvykle přidáváme

(ale není to nutné) dvě pomyslné činnosti, jednu, která předchází všem ostatním, a druhou, která následuje za všemi ostatními činnostmi.

Síťový graf s ohodnocenými hranami je z matematického hlediska elegantnější. Síťový graf s ohodnocenými vrcholy je naproti tomu snáze pochopitelný a dokáže jej sestavit i pracovník bez speciálního tréninku. Oba druhy grafů však umožňují vyjádřit zhruba totéž a lze je snadno a zcela mechanickým způsobem vzájemně převádět jeden na druhý. Metody těchto převodů jsou patrné z obr. 2.3, na kterém jsou nakresleny tři různé síťové grafy popisující týž proces stavby domu.



Obrázek 2.3: Příklady síťových grafů popisující týž proces stavby domu. Čárkované hrany představují fiktivní činnosti.

Pro síťové grafy s ohodnocenými hranami uvedeme několik důležitých pojmů. Pro jednoduchost předpokládejme, že v grafu existují vrcholy P a K (počátek a konec) takové, že všechny vrcholy grafu jsou orientovaně dostupné z P a ze všech vrcholů je orientovaně dostupný vrchol K. Označme

$a(e)$ = doba trvání činnosti e,

$T_1(v)$ = nejdřívější čas, kdy mohou být dokončeny všechny činnosti z $E^-(v)$, a kdy tedy mohou být zahájeny činnosti, které ve vrcholu v začínají,

$T_2(v)$ = nejpozdější čas, kdy je třeba zahájit činnosti z $E^+(v)$, aby bylo možno dodržet termín $T_1(K)$ dokončení celého projektu.

Čas se měří od okamžiku $T_1(P) = 0$ a požadujeme, aby $T_1(K) = T_2(K)$. Jednoduchou úvahou lze zjistit, že

$$T_1(v) = \max\{T_1(\mathrm{P}v(e)) + a(e) \mid e \in E^-(v)\} \tag{2.1}$$

a že hodnota $T_1(v)$ je rovna délce nejdelší cesty z vrcholu P do vrcholu v. Podobně

$$T_2(v) = \min\{T_2(\mathrm{K}v(e)) - a(e) \mid e \in E^+(v)\} \tag{2.2}$$

a doba $T_2(K) - T_2(v)$ je délkou nejdelší cesty z v do K. K výpočtu hodnot T_1 lze použít algoritmy popsané v 6.4.5, str. 101, hodnoty T_2 se počítají podobně, ale „proti směru hran“ počínaje z vrcholu K.

Tzv. *rezerva činnosti* $R(e) = T_2(\mathrm{K}v(e)) - T_1(\mathrm{P}v(e)) - a(e)$ vyjadřuje nejvyšší přípustné zdržení činnosti e, při kterém ještě lze dodržet termín $T_1(K)$ dokončení celého projektu (předpokládá se, že ostatní činnosti budou bez zdržení). Činnosti s nulovou rezervou se nazývají *kritické*.

Lze ukázat, že kritické činnosti se v síťovém grafu nevyskytují osamoceně. Každá kritická činnost leží na tzv. *kritické cestě*, což je orientovaná cesta z P do K, jejíž všechny hrany (tj. činnosti) jsou kritické. Kritických cest může být v grafu několik, každá z nich je zároveň nejdelší cestou z P do K.

Kritickým činnostem se věnuje zvýšená pozornost, neboť jakékoli jejich prodloužení se projeví prodloužením doby trvání celého projektu. Při řízení složitých projektů se výpočet síťového grafu (tj. výpočty termínů T_1, T_2 a rezerv) pravidelně opakuje s údaji, které jsou aktualizovány podle skutečného průběhu činností. Tato metoda řízení projektů se nazývá *metoda kritické cesty* nebo také *metoda CPM*, což je zkratka z anglického „Critical Path Method“.

Dodejme, že specialisté na síťové grafy rozeznávají i další druhy rezerv a že pojem kritické cesty a rezervy lze zavést i pro síťové grafy s ohodnocenými vrcholy.

2.4 Hledání statických konfigurací

2.4.1 Nejspolehlivější spojení. Nechť ve sdělovací síti je spolehlivost přímé linky mezi místy i a j dána pravděpodobností $p(i, j)$, že linka funguje. Předpokládejme, že poruchy na jednotlivých linkách jsou navzájem nezávislé. Pak spolehlivost určitého spojení mezi místy x, y je rovna součinu pravděpodobností $p(i, j)$ všech dílčích linek tvořících toto spojení. Úloha zní: najít nejspolehlivější spojení mezi danými dvěma místy x, y.

Sestrojme neorientovaný graf, jehož vrcholy jsou uzly spojovací sítě a jehož hranami jsou linky. Hrany jsou ohodnoceny délkami $a(i, j) = -\log p(i, j) \geq 0$. Nejspolehlivější spojení pak odpovídá nejkratší cestě z vrcholu x do vrcholu y. Příslušné algoritmy jsou uvedeny v kapitole 6, str. 87. Jiný způsob řešení této úlohy bez použití logaritmů je uveden v 7.2.6, str. 117.

2.4.2 Problém batohu. Lupič má batoh, do kterého může dát nejvýše K kilogramů kořisti a ani o gram více. Jeho lup se skládá z n předmětů o hmotnosti

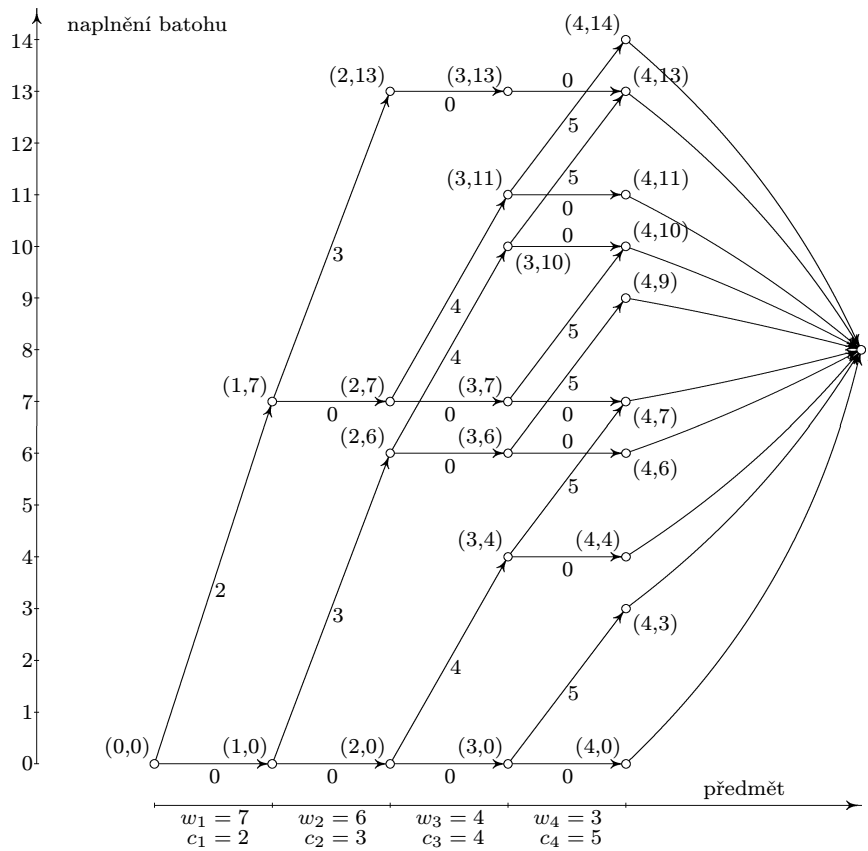
$w_1, w_2, \ldots, w_n$ kilogramů a cenách $c_1, c_2, \ldots, c_n$ Kč. Snahou lupiče je odnést v batohu co nejdražší kořist, ale nesmí batoh přetížit.

Tato úloha má mnoho poctivějších podob, zejména při sestavování rozvrhů a při plánování paralelně probíhajících procesů, které spotřebovávají nějaký materiál, energii nebo práci. Vždy se snažíme co nejužitečněji vytížit daný materiálový, energetický nebo lidský zdroj.

Ukážeme metodu, jak lze řešit problém batohu pomocí hledání nejdelší cesty ve speciálním grafu. Ovšem pozor: **není to metoda nejvhodnější**. Uvádíme ji pouze jako příklad méně zřejmého použití úloh o cestách.

Využijeme fakt, že řešení problému batohu si lze představit jako rozhodovací proces, kdy o jednotlivých předmětech postupně rozhodujeme, zda je do batohu zabalíme, nebo ne. Vrcholy odpovídají stavům rozhodovacího procesu a hrany představují možná rozhodnutí o určitém předmětu.

Na obr. 2.4 je příklad grafu pro problém batohu se čtyřmi předměty o hmotnostech 7, 6, 4 a 3 kg a cenách po řadě 2, 3, 4 a 5 Kč, kapacita batohu je 15 kg.



Obrázek 2.4: Graf k řešení problému batohu.

Hrany kreslené šikmo odpovídají rozhodnutí „zabalit“, vodorovné hrany znamenají „nezabalit“. Vrcholy jsou označeny souřadnicemi (i, j), přičemž i vyjadřuje, o kolika prvých předmětech je v tomto stavu již rozhodnuto, a druhá souřadnice vyjadřuje váhu zabalených předmětů. Šikmé hrany jsou ohodnoceny cenami zabalených předmětů, vodorovné hrany jsou ohodnoceny nulami. Orientované cesty z vrcholu $(0, 0)$ do vrcholu t vzájemně jednoznačně odpovídají přípustným výběrům předmětů do batohu. Přitom délka cesty je rovna součtu cen zabalených předmětů.

V našem příkladě je optimálním řešením výběr předmětů o hmotnosti 6, 4 a 3 kg s celkovou cenou 12 Kč. Největší hmotnost, kterou je možno do batohu uložit, je 14 kg, a to předměty o hmotnosti 7, 4 a 3 kg, jejich cena však je pouze 11 Kč. Všimněme si, že batoh nelze zcela naplnit a že 13 kg nákladu lze naložit dvěma různými způsoby, ovšem s podstatně různou cenou (5 a 12 Kč).

Znovu připomeňme, že pro řešení problému batohu existuje řada lepších algoritmů. Z nich se v této knize zmiňujeme o backtrackingu v 11.2 a o metodě větví a mezí v 11.3, str. 193.

2.4.3 Optimální umístění požární zbrojnice. Mějme silniční síť nějakého města. Naším úkolem je navrhnout umístění požární zbrojnice tak, aby její vzdálenost od nejvzdálenějšího bodu města byla co nejmenší. Silniční síť znázorníme grafem tak, aby silnice odpovídaly hranám a křižovatky vrcholům. Navíc umístíme vrcholy na všechna místa, kde lze předpokládat protipožární zásah, a také na všechna místa, kam by bylo možno postavit požární zbrojnici. Jsou-li i, j dva vrcholy, označme $u(i, j)$ jejich vzdálenost měřenou dobou jízdy požárního auta. Hledáme vrchol c, který má nejmenší hodnotu účelové funkce $\max\{u(c, j) \mid j \in V(G)\}$. Takový vrchol se nazývá *centrum grafu*.

Poněkud odlišnou úlohou je hledání nejvhodnějšího místa pro centrální sklad, ze kterého má být zásobováno k spotřebitelů, kteří denně odebírají množství $q_1, \ldots, q_k$ nějakého zboží. V takovém případě hledáme vrchol c, pro který je minimální hodnota účelové funkce $\sum u(c, i) \cdot q_i$, kde $u(i, j)$ je vzdálenost vrcholů i, j měřená jako náklady na přepravu jednotkového množství zboží.

Kapitola 3

Prohledávání grafů

Základním účelem prohledávání grafů je zjišťování dostupnosti, tj. zjišťování, do kterých vrcholů grafu vedou cesty z daného výchozího vrcholu, případně také nalezení těchto cest, tj. zjištění, kterými hranami a vrcholy procházejí. Dalším cílem prohledávání často je vykonání nějaké akce v každém dostupném vrcholu, popř. v každé dostupné hraně.

Je-li graf přehledně nakreslen a není-li příliš rozsáhlý, hravě dokážete řešit výše uvedené úlohy „pouhým pohledem“. Je-li však graf veliký nebo je-li nepřehledně nakreslen, je třeba použít nějaký systematický postup, jinak se snadno dopustíme chyby. Systematický postup prohledávání je samozřejmostí při použití počítače. A u grafů, které jsou zadány nepřímo pomocí procedur pro generování hran a vrcholů (viz 1.9.6), je systematické prohledávání základním způsobem, jak se o grafu vůbec něco dovědět.

Uvedeme tři způsoby prohledávání. Prvý z nich (značkování vrcholů) je značně obecný a nesmírně jednoduchý. Další dva způsoby, hledání do hloubky a hledání do šířky, lze pokládat za jeho speciální případy. Všechny tyto postupy prohledávání popisujeme pouze ve verzi pro orientované cesty. Modifikace pro hledání neorientovaných cest jsou snadné a jsou přenechány čtenáři jako cvičení.

3.1 Značkování vrcholů

3.1.1 Značkování vrcholů. Vrcholům grafu přiřazujeme značky. Má-li vrchol značku, znamená to, že do něj vede nějaká cesta z daného výchozího vrcholu r. Vlastní algoritmus je velmi prostý:

1. [*Inicializace.*] Označkujeme vrchol r, ostatní vrcholy jsou beze značek.

2. [*Výběr hrany.*] Vybereme libovolnou hranu e, jejíž počáteční vrchol má značku a koncový vrchol nikoli; pokračujeme podle kroku 3. Jestliže taková hrana neexistuje, výpočet končí.

3. [*Značkování.*] Označkujeme vrchol Kv(e) a pokračujeme ve výpočtu podle kroku 2.

3.1.2 Věta. Výpočet algoritmu 3.1.1 skončí nejpozději po $|V(G)|-1$ opakováních kroků 2 a 3. Po ukončení výpočtu budou označkovány právě ty vrcholy, do nichž existuje orientovaná cesta z vrcholu r.

Důkaz: Při každém vykonání kroku 3 je označkován dosud neoznačkovaný vrchol. Kroky 2 a 3 se tedy mohou opakovat nejvýše $(|V(G)|-1)$-krát.

Indukcí dokážeme, že do každého označkovaného vrcholu vede orientovaná cesta z vrcholu r taková, že prochází pouze přes již označkované vrcholy. Po provedení inicializace to triviálně platí: označkován je pouze sám vrchol r.

Předpokládejme, že toto tvrzení platí před provedením kroku 3 a že v kroku 2 byla vybrána hrana e vedoucí z vrcholu x do vrcholu y. Vrchol x má značku, podle indukčního předpokladu tedy existuje cesta C z vrcholu r do x. Tato cesta navíc prochází pouze přes označkované vrcholy, neobsahuje tedy ani hranu e, ani vrchol y. Díky tomu můžeme cestu C prodloužit o hranu e a vrchol y, čímž získáme cestu z r do y, která také vede přes označkované vrcholy. Tvrzení tedy platí i po označkování vrcholu y.

Obrácenou implikaci dokážeme sporem: Kdyby nebyl po skončení výpočtu označkován některý vrchol, do něhož vede cesta z vrcholu r, existovala by na této cestě hrana, jejíž počáteční vrchol by byl označkován, ale koncový nikoli (vrchol r totiž označkován je). To by však byl spor s krokem 2 algoritmu, neboť výpočet by ještě v takovéto situaci neměl být ukončen. □

3.1.3 Praktické provedení značkovací procedury. Asi nejjednodušší je opakovaně (cyklicky) procházet seznamem všech hran, pro každou hranu testovat, zda ji lze použít k označkování dalšího vrcholu (krok 2 algoritmu 3.1.1) a případně přidělit značku. Jestliže při celém jednom průchodu seznamem hran nebyl označkován žádný další vrchol, výpočet končí. V opačném případě pokračujeme tím, že znovu projdeme seznam hran.

Poněkud důmyslnější postupy jsou založeny na udržování seznamu vrcholů, kterým byla přidělena značka a které v tomto seznamu čekají na zpracování. Toto zpracování spočívá v postupném prozkoumání všech hran, které z vrcholu vycházejí, a v eventuelním označkování dalších vrcholů. Nově označkované vrcholy přidáváme do seznamu, vrcholy, které jsou zcela zpracovány (nebo s jejichž zpracováním právě začínáme) ze seznamu odstraňujeme. Výpočet končí, není-li co zpracovávat, tj. je-li seznam prázdný. Důležité varianty tohoto postupu uvedeme v 3.2 a 3.3.

3.1.4 Konstrukce cest. Často je třeba nejen zjistit existenci cesty, ale navíc tuto cestu explicite sestrojit. Pro tento účel lze značkovací algoritmus 3.1.1 snadno upravit:

Vždy ihned po označkování vrcholu Kv(e) přiřadíme navíc tomuto vrcholu hodnotu Odkud(Kv(e)) := e, kde e je hrana vybraná v kroku 2.

Pro takto doplněný algoritmus 3.1.1 samozřejmě stále platí věta 3.1.2, ale navíc platí:

> je-li označkován vrchol $v \neq r$, pak hodnota ODKUD(v) je jménem poslední hrany v (nějaké) cestě z vrcholu r do vrcholu v.

Cestu do označkovaného vrcholu $v \neq r$ lze po skončení značkovacího algoritmu snadno nalézt zpětným postupem pomocí hodnot ODKUD: Hodnota ODKUD(v) je poslední hranou v hledané cestě. Označme w počáteční vrchol hrany ODKUD(v). Je-li $w \neq r$, pak hodnota ODKUD(w) je předposlední hranou v cestě z vrcholu r do vrcholu w atd.

Označkované vrcholy a hrany obsažené v hodnotách ODKUD navíc tvoří kořenový strom s kořenem r (viz 1.7.1, str. 21).

3.1.5 Hledání cesty mezi danými vrcholy r a s. Zajímá-li nás pouze cesta z daného vrcholu r do daného vrcholu s, můžeme samozřejmě ukončit výpočet značkovacího algoritmu ihned po označkování vrcholu s.

3.1.6 Cvičení. Upravte značkovací proceduru 3.1.1 pro hledání neorientovaných cest. Přeformulujte příslušným způsobem větu 3.1.2 a její důkaz.

3.1.7 Cvičení. Jednoduchou úpravou algoritmu 3.1.1 lze získat algoritmus, který najde v grafu kružnici právě tehdy, když v grafu nějaká existuje. Navrhněte tuto úpravu a dokažte její správnost.

3.2 Prohledávání grafu do šířky

Prohledávání grafu do šířky lze stručně charakterizovat takto: nejprve označkujeme vrchol r, pak všechny jeho následníky, pak všechny dosud neoznačkované následníky těchto následníků atd. Lze si to představovat i tak, že graf současně prozkoumává velký počet průzkumníků, kteří „zaplavují“ postupně „po hladinách“ dostupnou část grafu.

Na této jednoduché metodě je pozoruhodné, že vede k nalezení nejkratších cest, tj. cest s nejmenším počtem hran.

3.2.1 Algoritmus hledání do šířky.

VSTUP: Orientovaný graf G a jeho vrchol r.
ÚKOL: Najít všechny vrcholy v, do nichž vede orientovaná cesta z vrcholu r, do každého z nich najít cestu o nejmenším počtu hran a zjistit délku VZD(v) této nejkratší cesty, tj. vzdálenost vrcholu v od vrcholu r.
POMOCNÉ PROMĚNNÉ: Všechny vrcholy, do nichž bude nalezena cesta, budou označkovány a budou jim přiřazeny hodnoty ODKUD a VZD. Dále použijeme seznam FRONTA, který bude obsahovat ty vrcholy, které již byly označkovány, ale z nichž jsme ještě nezkoumali možnosti dalšího postupu.

ALGORITMUS:

1. [*Inicializace.*] Označkujeme vrchol r, ostatní vrcholy jsou beze značek. Položme $\text{VZD}(r) := 0$, seznam FRONTA nechť obsahuje jen vrchol r.

2. [*Test ukončení.*] Je-li seznam FRONTA prázdný, výpočet končí.

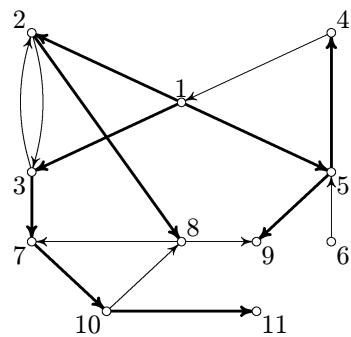
3. [*Volba vrcholu.*] Ze začátku seznamu FRONTA odebereme vrchol a označíme jej v.

4. [*Postup do šířky z vrcholu v.*] Pro každou hranu $e \in E^+(v)$ označíme $w := \text{KV}(e)$ a jestliže w dosud nemá značku, provedeme tyto operace: Označkujeme vrchol w; $\text{ODKUD}(w) := e$; $\text{VZD}(w) := \text{VZD}(v) + 1$; vrchol w připíšeme na konec seznamu FRONTA. Po zpracování všech hran z množiny $E^+(v)$ pokračujeme podle kroku 2.

ČASOVÉ NÁROKY: Je-li graf zadán ve tvaru seznamu vrcholů a seznamů výstupních okolí všech vrcholů (viz 1.9.2), proběhne hledání v čase $O(m + n)$.

POZNÁMKA: Pro hledání do šířky je podstatné, že ze seznamu FRONTA odebíráme vždy ten prvek (zde vrchol), který je v něm nejdéle. Taková datová struktura bývá označována termínem *fronta*.

3.2.2 Příklad. Použijeme algoritmus hledání do šířky na graf G (viz obr. 3.1) s množinou vrcholů $V(G) = \{1, 2, \ldots, 11\}$ a s množinou hran $E(G)$ danou seznamem $(1,2)$, $(1,3)$, $(1,5)$, $(2,3)$, $(2,8)$, $(3,2)$, $(3,7)$, $(4,1)$, $(5,4)$, $(5,9)$, $(6,5)$, $(7,10)$, $(8,7)$, $(8,9)$, $(10,8)$, $(10,11)$. Jako výchozí vrchol vezmeme vrchol 1. Na obr. 3.1 jsou silně vytaženy hrany, které jsou po skončení algoritmu obsaženy v hodnotách ODKUD a které vytvářejí systém nejkratších cest do všech dostupných vrcholů. Vrcholy byly značkovány v pořadí 1; 2, 3, 5; 8, 7, 4, 9; 10; 11. Středníkem jsou zde odděleny množiny vrcholů, kterým byla vypočtena stejná vzdálenost od vrcholu 1. Vrchol 6 nebyl označkován, neboť není orientovaně dostupný z vrcholu 1.



Obrázek 3.1: K příkladu 3.2.2 hledání do šířky.

3.2.3 Lemma (průběh hledání do šířky). Hodnoty VZD vrcholů zařazovaných algoritmem 3.2.1 do seznamu FRONTA tvoří neklesající posloupnost. V každém okamžiku se hodnoty VZD vrcholů obsažených v seznamu FRONTA liší nejvýše o jedničku.

DŮKAZ: Označme M_i množinu všech vrcholů v, kterým algoritmus přiřadil hodnotu $\text{VZD}(v) = i$. Obě tvrzení dokážeme indukcí podle doby práce algoritmu.

Zřejmě $M_0 = \{r\}$. Krok 4 algoritmu je vykonán nejprve pro vrchol r a bezprostředně poté seznam FRONTA obsahuje množinu M_1. Žádný další vrchol totiž hodnotu 1 nemůže dostat, neboť pouze vrchol r má hodnotu 0.

Dále je krok 4 vykonáván postupně pro všechny vrcholy z množiny M_1, přičemž na konec seznamu jsou zařazovány vrcholy s hodnotou 2. Po zpracování posledního vrcholu z množiny M_1 obsahuje seznam FRONTA množinu M_2. Žádný další vrchol totiž hodnotu 2 nemůže dostat, neboť pouze vrcholy z M_1 mají hodnotu 1.

Obecně, během zpracovávání vrcholů z množiny M_i obsahuje seznam FRONTA pouze vrcholy s hodnotami i a $i+1$ a těsně po zpracování posledního vrcholu z množiny M_i obsahuje seznam FRONTA množinu M_{i+1}. □

3.2.4 Věta. Algoritmus 3.2.1 skončí práci po konečném počtu kroků. Po zastavení jsou označkovány právě ty vrcholy, do nichž vede orientovaná cesta z vrcholu r. Je-li označkován vrchol $v \neq r$, pak $\text{ODKUD}(v)$ je poslední hranou v nejkratší cestě z vrcholu r do vrcholu v a $\text{VZD}(v)$ je délka (tj. počet hran) této nejkratší cesty.

DŮKAZ: Každý vrchol je ihned po označkování zařazen do seznamu FRONTA, a když je později odtud vyňat, nemůže už do něj být znovu zařazen, neboť už má značku. Proto kroky 2 až 4 mohou být provedeny celkem nejvýše $|V(G)|$-krát, algoritmus se tedy zastaví.

To, že budou označkovány právě ty vrcholy, do nichž vede z vrcholu r orientovaná cesta, se dokáže podobně jako věta 3.1.2, str. 50.

Zbývá dokázat, že algoritmus správně vypočte vzdálenosti. Je zřejmé, že do každého označkovaného vrcholu v vede nějaká cesta o délce $\text{VZD}(v)$ a že tato cesta je zakódována v hodnotách ODKUD. Ukážeme sporem, že tato cesta je nejkratší: kdyby do nějakého vrcholu v existovala cesta kratší než $\text{VZD}(v)$, pak by na této cestě musela existovat hrana e z vrcholu $x = \text{PV}(e)$ do vrcholu $y = \text{KV}(e)$ taková, že $\text{VZD}(x) + 1 < \text{VZD}(y)$. Při zpracování vrcholu x v kroku 4 vrchol y již buď byl označkován, nebo nebyl. Pokud byl, pak by podle lemmatu 3.2.3 měl hodnotu $\text{VZD}(y) \leq \text{VZD}(x)+1$, což by byl spor. Pokud nebyl, pak by byl označkován během zpracování vrcholu x a dostal by hodnotu $\text{VZD}(y) := \text{VZD}(x) + 1$, což by byl opět spor. Vzdálenosti jsou tedy vypočteny správně. □

3.2.5 Cvičení. Upravte algoritmus hledání do šířky 3.2.1 pro hledání neorientovaných cest.

3.2.6 Cvičení. V grafu z příkladu 3.2.2 proveďte hledání do šířky z výchozího vrcholu 6. Řešte bez použití obrázku!

3.2.7 Test, zda graf je bipartitní. Vezměme libovolný výchozí vrchol r a použijme hledání neorientovaných cest do šířky (viz 3.2.6).

Pokud v grafu existuje hrana e, která spojuje nějaké dva vrcholy x, y, které jsou oba v liché nebo oba v sudé vzdálenosti od výchozího vrcholu r, pak daný graf není bipartitní. Touto hranou totiž prochází kružnice liché délky (je tvořena hranou e a částmi cest z vrcholu r do vrcholů x a y) a ta se v žádném bipartitním grafu nemůže vyskytovat.

Pokud naopak hrana s uvedenými vlastnostmi neexistuje, pak prohledaná část grafu je bipartitní. Vrcholy, jejichž vzdálenost je lichá, tvoří jednu stranu, vrcholy v sudé vzdálenosti tvoří stranu druhou. Pokud zbyly v grafu nějaké neoznačkované vrcholy (tj. pokud graf není souvislý, viz 4.1), zvolíme jako výchozí vrchol r některý z neoznačkovaných vrcholů a celý postup opakujeme.

Popsaný postup lze zjednodušit: hrany lze testovat již v průběhu prohledávání do šířky a vzdálenosti stačí rozlišovat pouze na sudé a liché.

3.3 Prohledávání grafu do hloubky

Hledání do hloubky si lze představit jako průzkum grafu cestovatelem, který cestuje po hranách grafu a vrací se důsledně cestou, po které přišel.

Navíc je hledání do hloubky základem řady dalších, mnohdy velmi výkonných algoritmů, viz např. 5.1.13, str. 71, a 5.2.9, str. 78.

3.3.1 Algoritmus hledání do hloubky.

Vstup: Orientovaný graf G a jeho vrchol r.
Úkol: Najít všechny vrcholy, do nichž vede orientovaná cesta z vrcholu r.
Pomocné proměnné: Proměnná v bude obsahovat jméno aktuálního vrcholu, tj. vrcholu „kde právě teď jsme“. Seznam Trasa bude obsahovat vždy posloupnost hran tvořících tzv. aktuální cestu, tj. cestu z výchozího vrcholu r do aktuálního vrcholu v. Kromě toho budeme vrcholy, do nichž byla nalezena cesta, značkovat podobně jako v algoritmu 3.1.1.

Algoritmus:

1. [*Inicializace.*] $v := r$; Trasa $:= \emptyset$; označkujeme vrchol r, ostatní vrcholy jsou beze značek.

2. [*Volba hrany e.*] Vezmeme některou dosud nepoužitou hranu e začínající ve vrcholu v a pokračujeme podle kroku 3. Pokud taková hrana neexistuje, pokračujeme podle kroku 5.

3. [*Test vhodnosti hrany e.*] Označme w koncový vrchol hrany e. Je-li vrchol w označkován, pokračujeme podle kroku 2, v opačném případě podle kroku 4.

4. [*Postup do hloubky.*] Hranu e připíšeme na konec seznamu Trasa, položíme $v := w$ a označkujeme vrchol v. Pokračujeme krokem 2.

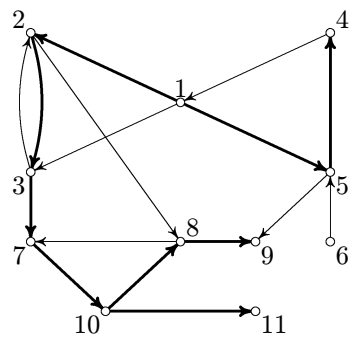
5. [*Návrat z vrcholu v.*] Je-li seznam TRASA neprázdný, odebereme z jeho konce hranu e, její počáteční vrchol označíme v a pokračujeme podle kroku 2. Je-li seznam TRASA prázdný, výpočet končí.

ČASOVÉ NÁROKY: Je-li graf zadán ve tvaru seznamu vrcholů a seznamů výstupních okolí všech vrcholů (viz 1.9.2), proběhne hledání v čase $O(m+n)$.

POZNÁMKA: Pro algoritmus hledání do hloubky je podstatné, že ze seznamu TRASA odebíráme vždy hranu, která je v seznamu nejkratší dobu. Tato datová struktura bývá označována názvem *zásobník* (anglicky stack) a je v jistém smyslu opakem fronty, která je naopak podstatná při prohledávání do šířky.

POZNÁMKA: Představa cestovatele, který chodí orientovaným grafem a vrací se, kudy přišel, má drobnou vadu na kráse: při postupu vpřed smíme jít pouze ve směru hrany, ale při návratu jdeme klidně v protisměru (ovšem při hledání do hloubky to jinak nejde).

3.3.2 Příklad. Použijeme hledání do hloubky na graf z příkladu 3.2.2, str. 52. Tento graf je znovu nakreslen na obr. 3.2, hrany, které byly použity k postupu do hloubky, jsou zde vytaženy silně. V kroku 2 jsme přitom volili hrany vycházející z vrcholu v vždy v pořadí, ve kterém jsou uvedeny v seznamu hran $E(G)$.



Obrázek 3.2: K příkladu hledání do hloubky.

Průběh výpočtu je tento: postup vpřed do vrcholů 1, 2, 3, 7, 10, 8, 9, návrat do vrcholu 8, test hrany $(8,7)$, návrat do vrcholu 10, postup vpřed do vrcholu 11, návrat do vrcholů 10, 7 a 3, test hrany $(3,2)$, návrat do vrcholu 2, test hrany $(2,8)$, návrat do vrcholu 1, test hrany $(1,3)$, postup vpřed do vrcholů 5 a 4, test hrany $(4,1)$, návrat do vrcholu 5, test hrany $(5,9)$, návrat do vrcholu 1.

3.3.3 Věta. Algoritmus hledání do hloubky 3.3.1 skončí práci po konečném počtu kroků. Po jeho zastavení jsou označkovány právě ty vrcholy, do nichž vede z vrcholu r orientovaná cesta.

DŮKAZ: Krok 3 může být proveden nejvýše jednou pro každou hranu, krok 4 nejvýše jednou pro každý vrchol. Vrchol, z něhož je proveden návrat, již nebude

zkoumán, neboť je označkován. Tedy i krok 5 bude pro každý vrchol proveden nejvýše jednou. Po kroku 2 vždy následuje krok 3 nebo 5, krok 2 tedy může být proveden nejvýše jednou pro každý vrchol a hranu.

Zbývající část věty se dokáže podobně jako věta 3.1.2, str. 50. □

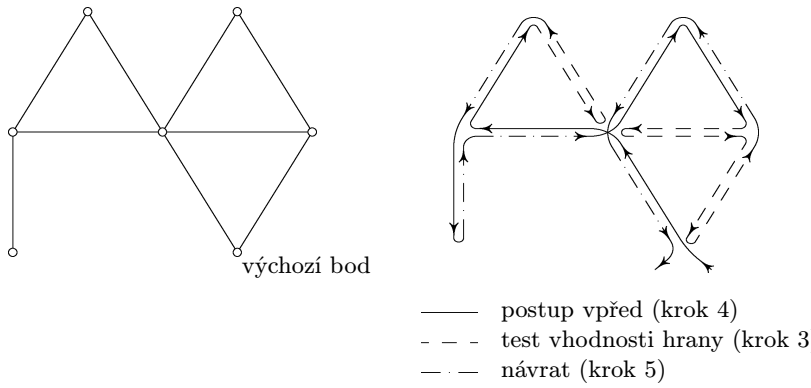
3.3.4 Konstrukce cest. Orientovaná cesta z vrcholu r do vrcholu v je obsažena v seznamu TRASA vždy ve chvíli, kdy vrchol v dostává značku. Obsah seznamu TRASA se však během výpočtu mění. Požadujeme-li zachování informací o nalezených cestách i po ukončení výpočtu, lze algoritmus 3.3.1 doplnit o vytváření hodnot ODKUD podobně, jako jsme to učinili v 3.1.4, str. 50. Po provedení této úpravy je seznam TRASA již zbytečný, neboť v kroku 5 lze k návratu z vrcholu v použít hodnotu ODKUD(v).

3.3.5 Cvičení. V grafu z příkladu 3.2.2, str. 52, proveďte hledání do hloubky z výchozího vrcholu 6. Zkuste to bez použití obrázku!

3.3.6 Hledání do hloubky v neorientovaných grafech. Algoritmus 3.3.1 lze snadno upravit pro hledání neorientovaných cest. Jediná změna se týká kroku 2, kde volíme libovolnou dosud nepoužitou hranu z množiny $E(v)$.

Jiná možnost spočívá v převodu daného grafu na symetrický orientovaný graf; každá hrana původního grafu je nahrazena dvojicí opačně orientovaných hran. V takto upraveném grafu pak lze přímo použít algoritmus 3.3.1.

3.3.7 Návod k prohledávání bludiště. Představte si, že jste v bludišti, které se skládá z neznámého počtu místností a z neznámého počtu chodeb, každá chodba spojuje vždy dvě místnosti. Vaším úkolem je prozkoumat systematicky všechny chodby a místnosti a najít východ z bludiště nebo spolehlivě zjistit, že východ neexistuje. Máte s sebou křídu, kterou si můžete dělat na stěny nebo podlahy chodeb a místností značky.



Obrázek 3.3: Příklad bludiště a jeho prohledání metodou do hloubky.

Tuto situaci lze modelovat neorientovaným grafem, jehož vrcholy odpovídají místnostem a hrany chodbám. Vy ovšem tento graf předem neznáte. Jste-li v některé místnosti (vrcholu), vidíte pouze ty chodby (hrany), které vycházejí z vaší místnosti.

K prohledávání bludiště lze použít hledání do hloubky. Při hledání je nutné označovat použité chodby a odlišit je od chodeb, po nichž se budeme teprve vracet. Nejjednodušší je kreslit při prvém průchodu chodbou jednu čáru a při návratu druhou. Z místnosti pak odcházíme pokud možno neoznačenou chodbou (krok 2). Vede-li tato chodba do dříve navštívené místnosti, nevstupujeme do ní, a okamžitě se vrátíme touž chodbou zpět (krok 3). Pokud jsme přišli do dosud nenavštívené místnosti, postupujeme z ní dále (krok 4). Jsou-li již všechny chodby vedoucí z místnosti použity (alespoň jedenkrát), vrátíme se chodbou, ve které je jen jedna čára (krok 5). Konec hledání poznáme podle toho, že jsme v místnosti, kde všechny chodby jsou označeny dvěma čarami. V takovém případě stojíme v místě, kde jsme začínali. Pokud východ z bludiště existuje a je dosažitelný, museli jsme jít kolem něj.

Příklad bludiště a možný způsob jeho prohledání je na obr. 3.3.

Vyzkoušejte na skutečném příkladě. Světla s sebou!

3.3.8 Cvičení. Rozhodněte, zda je při hledání orientovaných cest do hloubky pravdivé toto tvrzení:

> Vždy při návratu z vrcholu v jsou označkovány všechny vrcholy, do nichž vede z v orientovaná cesta.

3.3.9 Hledání do hloubky s použitím rekurzivní procedury. Mnohé programovací jazyky (např. PASCAL nebo C) dovolují, aby procedura (podprogram) volala sebe sama. Tuto možnost, tzv. rekurzivní volání, lze někdy využít k elegantnímu zápisu algoritmu. Jedním z klasických příkladů je prohledávání grafu do hloubky.

```
program HLEDANI_DO_HLOUBKY;

  procedure HLEDEJ (v : vrchol);
  begin
    označkuj vrchol v;
    pro všechny hrany e ∈ E⁺(v) proveď
      begin
        w := KV(e);
        if (w nemá značku) then HLEDEJ(w);
      end;
  end;

begin {vlastní prohledávání}
  Všechny vrcholy grafu musí být beze značek;
  HLEDEJ(r);
end.
```

Procedura HLEDEJ vezme vrchol v, který jí byl předán jako parametr, a postará se o prohledání všech dosud neoznačkovaných vrcholů, které jsou z něj dostupné. Po-

stará se o to tím, že vezme všechny neoznačkované následníky vrcholu v a u každého z nich požádá (sebe sama) o prohledání všeho, co je z něj dostupné.

Stojí za zmínku, že zde nepoužíváme žádný seznam Trasa. Je to zbytečné, protože mechanismus rekurzivního vyvolávání si vždy musí pamatovat, kam se má vrátit a v čem má pokračovat. Tento mechanismus pracuje na principu zásobníku a de facto nahrazuje seznam Trasa z algoritmu 3.3.1. Podrobnosti lze najít v učebnicích programování.

3.3.10 Konstrukce všech cest z daného vrcholu. Snadnou úpravou algoritmu hledání do hloubky 3.3.1 dosáhneme toho, že v průběhu výpočtu se v seznamu Trasa vyskytnou postupně všechny posloupnosti hran, které odpovídají všem cestám, které začínají ve vrcholu r. Lze toho využít například k tomu, abychom ze všech cest, které začínají v r, vybrali cestu, která splňuje nějakou speciální podmínku. Takto lze například hledat cestu, která začíná v r a obsahuje všechny vrcholy grafu (tzv. hamiltonovskou cestu, viz kapitola 12, str. 199) a lze dokonce snadno ze všech takových cest vybrat nejkratší.

Úprava algoritmu 3.3.1 spočívá v tom, že v kroku 5 při návratu z vrcholu v smažeme jeho značku a všechny hrany z $E^+(v)$ budeme nadále pokládat za dosud nepoužité. Přijdeme-li pak někdy znovu (ale jinou cestou) do tohoto vrcholu, budeme znovu zkoumat všechny způsoby pokračování. Značky budou mít vždy jen ty vrcholy, které leží na cestě obsažené v seznamu Trasa.

Použití tohoto algoritmu ovšem je zpravidla časově velice náročné: doba práce roste exponenciálně s počtem vrcholů grafu, přičemž tento růst je tím rychlejší, čím vyšší jsou stupně vrcholů. Přidáme-li ke grafu několik vrcholů, může se doba práce prodloužit několikanásobně. Proto lze tento algoritmus pokládat za východisko z nouze, neznáme-li lepší postup.

Tento algoritmus lze pokládat za jednoduchou formu tzv. backtrackingu, viz 11.2, str. 191.

3.3.11 Cvičení. Navrhněte varianty algoritmu 3.3.10 pro hledání neorientovaných cest a pro vyhledání všech cyklů nebo kružnic procházejících vrcholem r.

3.3.12 Prohledávání kořenového stromu do hloubky je jednodušší než prohledávání obecného grafu. Víme-li bezpečně, že prohledávaný graf je kořenovým stromem, není třeba vrcholy značkovat. V kořenovém stromě totiž do každého vrcholu vede pouze jedna cesta z kořene. Díky tomu máme jistotu, že postupem do hloubky se vždy dostaneme do nového, dosud nenavštíveného vrcholu.

3.3.13 Preorder, postorder a inorder. V uspořádaném binárním kořenovém stromě lze pořadí vrcholů preorder, postorder a inorder (viz 1.7.2, str. 23) získat během prohledávání stromu do hloubky. Do algoritmu umístíme na vhodné místo příkaz pro tisk jména vrcholu. Umístíme-li jej těsně za příchod do vrcholu, dostaneme pořadí preorder. Umístíme-li jej těsně před návrat z vrcholu, dostaneme pořadí postorder. Konečně, budeme-li tisknout jméno vrcholu po prohledání jeho levého podstromu, ale před prohledáním pravého podstromu, dostaneme pořadí inorder.

Kapitola 4

Pojmy založené na neorientovaných cestách

4.1 Souvislost

4.1.1 Souvislost. Graf nazýváme *souvislým*, jestliže každé jeho dva vrcholy jsou spojeny neorientovanou cestou.

Pro odlišení od tzv. silné souvislosti (viz 5.1) někdy mluvíme o *slabé* nebo také *obyčejné souvislosti*.

4.1.2 Komponenta souvislosti grafu G (též *souvislá komponenta* nebo i jen *komponenta*) je každý podgraf H grafu G, který je souvislý a který je maximální s touto vlastností, tj. není částí většího souvislého podgrafu.

Obrázek 4.1: Grafy k příkladu 4.1.3.

4.1.3 Příklad. Oba grafy na obr. 4.1 mají každý čtyři komponenty souvislosti s množinami vrcholů $\{1,3,5\}$, $\{2,4,6\}$, $\{7,8\}$, $\{9\}$.

4.1.4 Vlastnosti komponent souvislosti. Na množině vrcholů grafu G definujme relaci $\sim$ předpisem

$$a \sim b \quad \Longleftrightarrow \quad \text{vrcholy } a, b \text{ jsou v grafu } G \text{ spojeny neorientovanou cestou.}$$

Velmi snadno lze ověřit, že takto definovaná relace $\sim$ je ekvivalencí (tj. je reflexivní, symetrická a tranzitivní, viz 1.12.6, str. 34), a že tedy určuje na množině vrcholů

rozklad na ekvivalenční třídy. Podgrafy, které jsou indukovány těmito třídami, jsou komponentami souvislosti (neboť jsou to maximální souvislé podgrafy).

Odtud okamžitě plyne, že každý vrchol grafu leží v přesně jedné komponentě souvislosti, a také, že komponentu souvislosti, která obsahuje vrchol x, můžeme získat jako podgraf indukovaný množinou všech vrcholů, které jsou neorientovaně dostupné z vrcholu x.

4.1.5 Hledání komponent souvislosti. Zvolíme libovolně vrchol r a pomocí kteréhokoli algoritmu pro hledání neorientovaných cest (viz kap. 3, str. 49) najdeme množinu vrcholů neorientovaně dostupných z vrcholu r. Podgraf indukovaný touto množinou je komponentou souvislosti.

Pak zvolíme vrchol r mimo dosud vytvořené komponenty a celý postup opakujeme, dokud není každý vrchol zařazen do některé komponenty.

4.1.6 Cvičení. V následujících neformálně popsaných grafech objasněte význam pojmu komponenty souvislosti:

1. Vrcholy jsou atomy, hrany jsou chemické vazby mezi nimi.

2. Vrcholy jsou lidé a hrana z vrcholu x do vrcholu y vede právě tehdy, když x je otcem nebo matkou osoby y. (Všimněte si, že tento graf je orientovaný, ale my se ptáme na komponenty souvislosti, u nichž na orientaci hran nezáleží.)

4.2 Stromy a kostry

4.2.1 Les a strom. *Les* je graf, který neobsahuje kružnici. *Strom* je graf, který neobsahuje kružnici a je navíc souvislý.

Komponentami souvislosti lesa jsou tedy stromy.

4.2.2 Věta. Každý souvislý graf má faktor, který je stromem.

Důkaz: Jestliže graf obsahuje kružnici, odstraníme z grafu libovolnou hranu této kružnice. Souvislost grafu tím zůstane zachována. Takto opakovaným odstraňováním hran získáme faktor původního grafu, který je souvislý a neobsahuje již žádnou kružnici. □

4.2.3 Kostra grafu. Faktor grafu G, který je stromem, nazýváme *kostrou grafu* G (též *napnutým stromem* grafu G).

Předchozí věta 4.2.2 říká, že každý souvislý graf má kostru.

4.2.4 Napnutý les. Faktor grafu G, který je lesem a má stejný počet komponent jako graf G, nazýváme *napnutým lesem grafu* G.

4.2.5 Věta. Každý graf má napnutý les.

Důkaz je podobný jako 4.2.2: je-li v grafu kružnice, pak odstraněním některé její hrany se počet komponent souvislosti nezmění. Opakovaným trháním hran tedy dostaneme napnutý les. □

4.2.6 Věta. Každý strom s alespoň dvěma vrcholy obsahuje alespoň dva vrcholy stupně 1.

Důkaz: Žádná cesta spojující dva vrcholy stromu nemůže mít větší počet hran než $|V(G)-1|$. Vezměme tedy cestu C, která má ze všech cest největší počet hran. Označme x, y počáteční a koncový vrchol cesty C. Kdyby některý z těchto vrcholů (označme jej x) měl stupeň alespoň 2, vycházela by z vrcholu x další hrana e, která nepatří do cesty C. Její druhý krajní vrchol z nemůže ležet na cestě C, jinak by v grafu existovala kružnice. Bylo by tedy možné cestu C prodloužit o hranu e a vrchol z, což je spor, neboť C byla nejdelší. Proto vrcholy x a y mají stupeň 1. □

4.2.7 Věta. Každý strom o n vrcholech má přesně $n-1$ hran.

Důkaz: Větu dokážeme indukcí podle počtu vrcholů. Pro graf s 1, 2 nebo 3 vrcholy lze větu ověřit přímo. Předpokládejme tedy platnost věty pro stromy s k vrcholy a vezměme libovolný strom, který má $k+1$ vrcholů. Podle věty 4.2.6 má tento strom nějaký vrchol stupně 1. Odstraněním tohoto vrcholu a jediné hrany s ním incidentní získáme menší strom s přesně k vrcholy a ten podle indukčního předpokladu má přesně $k-1$ hran. Původní strom měl tedy $k+1$ vrcholů a k hran. □

Důsledek: Každý souvislý graf o n vrcholech má alespoň $n-1$ hran.

4.2.8 Věta. Každý les o n vrcholech a k komponentách souvislosti má přesně $n-k$ hran.

Důkaz: Označme $n_1, n_2, \ldots, n_k$ počty vrcholů v komponentách souvislosti daného lesa. Každá komponenta je stromem, podle věty 4.2.7 jsou tedy počty hran v těchto komponentách rovny n_1-1, n_2-1, $\ldots$, n_k-1. Poněvadž $n = \sum_{i=1}^{k} n_i$, dostaneme sečtením počet hran v celém lese $n-k$. □

4.2.9 Vlastnosti stromů. Nechť G je graf s n vrcholy, $n \geq 1$. Potom následující podmínky jsou ekvivalentní:

1. G je strom.
2. G neobsahuje kružnici a má přesně $n-1$ hran.
3. G neobsahuje kružnici a má alespoň $n-1$ hran.
4. G je souvislý a má přesně $n-1$ hran.
5. G je souvislý a má nejvýše $n-1$ hran.
6. G je souvislý a odebráním kterékoli hrany přestane být souvislý.
7. G neobsahuje kružnici a po přidání libovolné hrany bude obsahovat přesně jednu kružnici.
8. Každá dvojice vrcholů je spojena přesně jednou neorientovanou cestou.

Důkaz: $1 \Rightarrow 2, 3, 4, 5$ vyplývá přímo z definice stromu a z věty 4.2.7.

$2 \Rightarrow 3$ a $4 \Rightarrow 5$ jsou zřejmé bez důkazu.

$3 \Rightarrow 1$: Graf G je lesem. Označíme-li k počet jeho komponent souvislosti a m počet hran, pak podle věty 4.2.8 má graf G přesně $m = n - k$ hran. Podle předpokladu 3 však platí $m \geq n - 1$. Po dosazení máme $k \leq 1$, graf má tedy jen jednu komponentu souvislosti, je tedy souvislý, a proto je to strom.

$5 \Rightarrow 1$: Graf G je souvislý, obsahuje tedy kostru. Kostra je stromem, a proto má přesně $n - 1$ hran, což je přesně tolik, kolik jich má samotný graf G. Z toho plyne, že G je sám svou vlastní kostrou, a je tedy stromem.

$1 \Rightarrow 6$: Je-li $n = 1$, je tvrzení samozřejmé. Nechť tedy $n \geq 2$. Graf G je strom, má tedy přesně $n-1$ hran. Po odstranění libovolné hrany dostaneme graf H, který má jen $n - 2$ hran, a tudíž podle důsledku věty 4.2.7 není souvislý.

$6 \Rightarrow 7$: Kdyby G obsahoval kružnici, pak by zůstal souvislý i po odstranění kterékoli hrany z této kružnice. Tedy G neobsahuje kružnici. Vytvořme nyní graf H přidáním hrany e spojující vrcholy x, y. Vrcholy x, y byly ovšem spojeny cestou v G. To znamená, že v H existuje kružnice obsahující hranu e. Zbývá dokázat, že tato kružnice je jediná.

Kdyby H obsahoval dvě různé kružnice, procházely by obě hranou e. Existovaly by tedy dvě různé cesty spojující vrcholy x, y. To by ovšem znamenalo, že již graf G by musel obsahovat kružnici. Graf H tedy obsahuje přesně jednu kružnici.

$7 \Rightarrow 8$: Vezměme dvojici vrcholů x, y. Přidáním hrany e spojující x, y vznikne přesně jedna kružnice. To znamená, že už v grafu G byly x, y spojeny cestou. Kdyby byly spojeny dvěma různými cestami, existovala by již v G kružnice.

$8 \Rightarrow 1$: Souvislost grafu je zřejmá. Kdyby v G existovala kružnice, byla by některá dvojice vrcholů spojena dvěma různými cestami. □

4.2.10 Cvičení. Kořenový strom (definovaný v 1.7.1, str. 21) je stromem. Dokažte!

4.2.11 Věta. Jestliže značkovací algoritmus 3.1.1, str. 49, upravený pro neorientované cesty a s úpravou 3.1.4 označkoval všechny vrcholy grafu G, pak faktor grafu, který obsahuje pouze ty hrany, které jsou hodnotami Odkud označkovaných vrcholů různých od r, tvoří kostru grafu G.

Důkaz: Do každého vrcholu vede neorientovaná cesta tvořená hranami faktoru. Faktor je tedy souvislý. Počet hran faktoru je $|V(G) - 1|$, neboť pouze pro vrchol r není hodnota Odkud(r) definována. Podle 4.2.9 je tedy tento faktor stromem. □

Poznámka: Podobná tvrzení platí i pro další algoritmy pro hledání cest z daného výchozího vrcholu, pokud zaznamenávají poslední hrany cest způsobem popsaným v 3.1.4.

4.2.12 Cvičení. Za jakých podmínek má graf několik různých koster?

4.2.13 Cvičení. Kolik vznikne kružnic, přidáme-li ke stromu dvě hrany?

4.2.14 Cvičení. Nakreslete všechny vzájemně neizomorfní kostry úplných grafů K_4 a K_5. Kolik jich je?

4.3 Minimální kostra

4.3.1 Formulace úlohy. Je dán souvislý graf G, jehož hrany jsou ohodnoceny reálnými čísly, která budeme nazývat *cenami*. Kostru grafu G nazveme *minimální kostrou*, jestliže má nejmenší cenu, tj. nejmenší součet ohodnocení hran (mezi všemi kostrami grafu G).

4.3.2 Příklady aplikací. Předpokládejme, že máme za úkol propojit n míst $1, 2, \ldots, n$ vedením vysokého napětí. Pro každou dvojici míst i, j máme k dispozici odhad nákladů na postavení přímé linky mezi místy i, j. Naší snahou je propojit všech n míst co nejlevněji, přitom však nesmíme vedení větvit mimo místa, která propojujeme.

Je zřejmé, že vybudované linky nesmějí tvořit kružnici, jinak by bylo možno vynecháním některé linky snížit celkové náklady. Vybudované linky tedy musí tvořit minimální kostru. Všimněte si, že toto nejlevnější řešení není ideální z hlediska spolehlivosti: porucha kterékoli linky způsobí rozpad sítě na dvě komponenty souvislosti.

Jiným příkladem může být silniční síť, z níž chceme v zimě udržovat sjízdnou pouze co nejkratší část, která vzájemně propojí všechny obce v dané oblasti.

Kromě těchto přímých aplikací se s minimální kostrou setkáváme jako s dílčím problémem při řešení složitějších kombinatorických úloh.

4.3.3 Postup hledání minimální kostry.

VSTUP: Souvislý graf G a ohodnocení hran cenami $c : E(G) \to \mathbb{R}$.
ÚKOL: Najít minimální kostru grafu G.
POMOCNÉ PROMĚNNÉ: Faktor L grafu G, který neobsahuje kružnici (tj. napnutý les grafu G). K tomuto lesu budeme přidávat vhodně zvolené hrany.

ALGORITMUS:

1. [*Inicializace.*] Nechť L je diskrétní graf s množinou vrcholů $V(G)$.

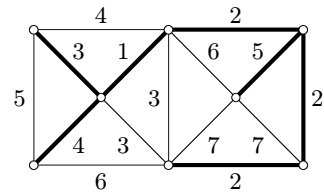
2. [*Test ukončení.*] Je-li les L stromem, výpočet končí, L je hledaná minimální kostra.

3. [*Volba hrany.*] Zvolme libovolně hranu e s touto vlastností:

$$(*)\ \begin{cases} \text{Hrana } e \text{ spojuje dvě různé komponenty lesa } L \text{ a alespoň pro} \\ \text{jednu z těchto komponent (označme ji } A\text{) platí, že cena hrany} \\ e \text{ je nejmenší ze všech cen hran z množiny } W_G(A). \end{cases}$$

4. [*Spojení komponent.*] Hranu e přidejme k lesu L (tím snížíme počet komponent) a pokračujme krokem 2.

4.3.4 Příklad. V grafu na obr. 4.2 je minimální kostra vytažena silně.



Obrázek 4.2: Minimální kostra.

4.3.5 Věta. Algoritmus 4.3.3 se po konečném počtu kroků zastaví a po zastavení je L minimální kostrou grafu G.

Důkaz: Kroky algoritmu 2, 3 a 4 se provedou přesně $(|V(G)|-1)$-krát, neboť při každém vykonání se zmenší počet komponent o jednu.

Správnost algoritmu dokážeme indukcí. Ukážeme, že po celou dobu výpočtu platí toto tvrzení: Graf G je lesem a existuje minimální kostra grafu G taková, že les L je jejím faktorem.

Po provedení kroku 1 to jistě platí. Předpokládejme tedy, že před provedením kroku 4 je les L faktorem nějaké minimální kostry K. Jestliže K obsahuje přidávanou hranu e, pak samozřejmě tvrzení platí i po jejím přidání k lesu L. Jestliže kostra K hranu e neobsahuje, pak přidáním této hrany ke kostře K vznikne kružnice. Tato kružnice obsahuje kromě hrany e ještě jednu hranu, označme ji h, takovou, že $h \in$ $\in W_G(A)$. Z podmínky $(*)$ plyne, že $c(e) \leq c(h)$. Přidejme nyní ke kostře K hranu e a odeberme hranu h. Tím dostaneme opět kostru, označme ji H, přičemž cena této nové kostry není vyšší než cena kostry K. Kostra H je tedy také minimální kostrou, navíc však obsahuje kromě lesa L i hranu e. Obsahuje tedy les L i po provedení kroku 4, tj. po přidání hrany e. □

4.3.6 Varianty postupu 4.3.3. Podmínce $(*)$ v kroku 3 obvykle vyhovuje několik hran najednou, postup 4.3.3 tedy není jednoznačný. Předchozí věta 4.3.5 nám zaručuje, že to nevadí. Ať už při výběru hrany dodržíme podmínku $(*)$ jakkoli, výsledkem bude vždy minimální kostra.

Pro praktický výpočet je ovšem vhodné výběr nějak omezit. Nejznámější jsou následující dva algoritmy.

4.3.7 Hladový algoritmus. Hrany grafu uspořádáme podle jejich cen do neklesajícího pořadí, tj. od nejlevnější po nejdražší. Pak hrany v tomto pořadí probíráme a do grafu L přidáváme jen ty z nich, které v grafu L nezpůsobí vznik kružnice. Tento algoritmus je známý také jako *Kruskalův*, pochází z roku 1956.

Správnost hladového algoritmu vyplývá z faktu, že hrana, kterou algoritmus přidává ke grafu L, je vždy nejlevnější ze všech hran, které spojují dvě různé komponenty lesa L. Přidávaná hrana tedy vždy splňuje podmínku $(*)$, a proto správnost hladového algoritmu vyplývá z věty 4.3.5.

Název „hladový algoritmus“ vystihuje fakt, že k lesu L přidáváme vždy tu nejlevnější hranu, která v dané situaci připadá v úvahu. Takovýto prostoduchý

postup, kdy se v každém kroku snažíme o maximální okamžitý zisk bez ohledu na případné budoucí problémy, je sice často používán při řešení nejrůznějších optimalizačních úloh, ale pro většinu úloh nezaručuje, že výsledkem bude skutečně optimální řešení. Jde tedy zpravidla jen o heuristický algoritmus, viz 11.4, str. 198. (Zkuste třeba „hladově“ hledat nejkratší cestu. Snadno najdete příklad, kdy hladový postup dá chybný výsledek.)

Úloha o minimální kostře je pozoruhodná tím, že pro ni hladový postup vede ke skutečně optimálnímu řešení zaručeně a vždy. Znovu upozorňujeme, že typy úloh, pro které hladový postup dává zaručeně optimální výsledky, jsou (bohužel) dosti vzácné.

4.3.8 Jarníkův-Primův algoritmus popsal v roce 1930 V. Jarník a nezávisle na něm v roce 1957 R. C. Prim:

V postupu 4.3.3 nejprve pevně zvolíme nějaký vrchol v a pak při výběru hrany, která splňuje podmínku $(*)$, volíme jako komponentu A vždy tu, která obsahuje vrchol v.

Les L tedy během výpočtu má jako komponenty souvislosti kromě izolovaných vrcholů vždy jen jeden strom, který postupně roste tím, že k němu přidáváme hrany, které jej nejlevněji spojují s některým izolovaným vrcholem.

Přidávaná hrana triviálně splňuje podmínku $(*)$, důkaz správnosti algoritmu tedy opět vyplývá z věty 4.3.5.

4.3.9 Borůvkův algoritmus je nejstarším algoritmem pro minimální kostru, pochází již z roku 1926 (historické podrobnosti viz [Šišma97] a [MN96]). Algoritmus funguje za předpokladu, že ceny všech hran jsou různé, a lze jej popsat takto:

> V postupu 4.3.3 vždy vezmeme všechny hrany, které splňují podmínku $(*)$, a přidáme je ke grafu L všechny najednou.

Borůvkův algoritmus je velmi rychlý, neboť počet komponent souvislosti lesa L se v každém kroku zmenší nejméně o polovinu (v typickém případě mnohem více), počet kroků Borůvkova algoritmu tedy bude nejvýše $\log_2 n$. Celková doba práce tedy bude $O(m \log n)$. Nejrychlejší známé algoritmy pro minimální kostru jsou založeny právě na myšlenkách Borůvkova algoritmu.

Striktně vzato, Borůvkův algoritmus není speciálním případem postupu 4.3.3, přesto lze jeho správnost pomocí postupu 4.3.3 dokázat:

4.3.10 Věta (správnost Borůvkova algoritmu). Borůvkův algoritmus 4.3.9 správně nalezne minimální kostru grafu.

Důkaz: Označme B množinu hran, které Borůvkův algoritmus přidává k lesu L v jednom kroku. Ukážeme, že obecný postup 4.3.3 může v této situaci v $|B|$ po sobě jdoucích krocích udělat totéž. Přesněji, dokud $B \neq \emptyset$, může obecný postup 4.3.3 dělat to, že vybere libovolnou hranu z množiny B, tuto hranu z množiny B odstraní a přidá ji k lesu L. Potřebujeme dokázat, že přitom všechny zbývající

hrany v množině B budou stále splňovat podmínku $(*)$, a lze je tedy k lesu L přidat v dalších krocích.

Každé komponentě A lesa L přiřaďme hranu e_A, která je nejlevnější z množiny hran $W(A)$. Ze všech hran, které splňují podmínku $(*)$, jsou některé hrany takto přiřazeny jen jedné komponentě, ostatní jsou přiřazeny dvěma komponentám. (Nakreslete si příklad!)

Je-li z množiny B vybrána hrana e_A, která je přiřazena jen jedné komponentě A, pak přidáním hrany e_A k lesu L spojíme komponentu A s nějakou jinou komponentou C. Hrana e_C, která byla dosud nejlevnější hranou z množiny $W(C)$, je přitom levnější než e_A (proč?) a nyní bude nejlevnější hranou z množiny $W(A \cup C)$, bude tedy i nadále splňovat podmínku $(*)$. Ostatní hrany z množiny B zůstávají přiřazeny ke komponentám mimo A a C, tj. ke komponentám, které se nezměnily. Proto tyto hrany i nadále splňují podmínku $(*)$.

Je-li z množiny B vybrána hrana e, která je přiřazena dvěma komponentám A, C, pak spojením těchto dvou komponent vznikne komponenta $A \cup C$. Hrana $e_{A\cup C}$, která bude nyní přiřazena této komponentě, sice možná neleží v B, ale opět platí, že zbývající hrany z množiny B zůstávají přiřazeny ke komponentám, které se nezměnily, a proto i nadále splňují podmínku $(*)$. □

4.3.11 Cvičení. Použijte všechny tři popsané algoritmy na graf z obr. 4.2 a všimněte si rozdílů v jejich činnosti. Všimněte si také, že Borůvkův algoritmus pro tento graf dá (náhodou) správný výsledek i přesto, že ceny všech hran nejsou různé.

4.3.12 Cvičení. Proč se v Borůvkově algoritmu požaduje, aby ceny všech hran byly navzájem různé? Najděte příklad, kdy by to mohlo vadit. Navrhněte úpravu Borůvkova algoritmu takovou, aby správně pracoval i na grafech, v nichž ceny některých hran jsou stejné.

4.3.13 Poznámka. Úloha o minimální kostře má ve srovnání s jinými úlohami několik pozoruhodných vlastností. Jednu jsme již zmínili v 4.3.7: hladový algoritmus dává vždy optimální řešení.

Jinou pozoruhodnou vlastností je to, že optimální řešení (tj. minimální kostra) nezávisí na absolutních velikostech cen jednotlivých hran, ale pouze na jejich uspořádání, tj. na relaci $\leq$ mezi nimi. To znamená, že pokud se ceny hran změní, ale nerovnosti mezi nimi zůstanou zachovány, minimální kostra bude stejná.

Důkaz tohoto tvrzení lze snadno odvodit ze správnosti hladového (Kruskalova) algoritmu 4.3.7. Hrany totiž budou probírány ve stejném pořadí, proto i výsledek bude stejný. Vyplývá to však i z následující věty.

4.3.14 Věta (charakterizace minimální kostry). Kostra K grafu G je minimální kostrou právě tehdy, když

$$(**) \qquad \begin{cases} \text{pro každou hranu } e \notin E(K) \text{ a pro každou hranu } h \text{ na} \\ \text{(jediné) cestě, která v kostře } K \text{ spojuje krajní vrcholy} \\ \text{hrany } e, \text{ platí } c(e) \geq c(h). \end{cases}$$

Důkaz: Předpokládejme, že kostra K je minimální. Kdyby podmínka $(**)$ neplatila, existovala by hrana $e \notin E(K)$ a hrana h ležící na cestě, která v kostře K spojuje krajní vrcholy hrany e taková, že $c(e) < c(h)$. Pak bychom však mohli získat levnější kostru tím, že bychom z kostry K odebrali hranu e a nahradili ji hranou h, což by byl spor. Podmínka $(**)$ tedy platí.

Dokážeme opačnou implikaci. Předpokládejme pro spor, že podmínka $(**)$ platí a kostra K není minimální. Graf G jistě má nějakou minimální kostru, označme ji L. Vezměme libovolnou hranu e kostry L, která neleží v K. V kostře K existuje (jediná) cesta, která spojuje krajní vrcholy hrany e. Tato cesta obsahuje alespoň jednu hranu h, která neleží v kostře L (jinak by kostra L obsahovala kružnici). Z předpokladu $(**)$ plyne, že $c(e) \geq c(h)$. Jestliže tedy z kostry K odebereme hranu h a nahradíme ji hranou e, cena kostry K tím neklesne. Tento postup (totiž výběr hran e, h a jejich výměnu v kostře K) můžeme opakovat, dokud nebudou kostry K a L stejné. Původní kostra K nebyla nejlevnější, při výměnách její cena nikdy neklesla, ale výsledná kostra L nejlevnější je, což je spor. Kostra K splňující $(**)$ tedy byla nejlevnější. □

4.3.15 Alternativní návod k hledání minimální kostry. Díky větě 4.3.14 můžeme postupovat takto: Vezmeme libovolnou kostru a testováním podmínky $(**)$ zjistíme, zda je minimální. Pokud není, tj. najdeme-li hrany e a h, díky nimž podmínka $(**)$ neplatí, pak výměnou hran e a h získáme levnější kostru. Toto opakujeme, dokud nedostaneme kostru, která je minimální.

Tento alternativní postup je zpravidla zbytečně pracný, neboť algoritmy založené na 4.3.3 jsou podstatně efektivnější. Máme-li však kostru, která již je „skoro optimální“, může být alternativní postup velmi výhodný.

4.4 Stupně souvislosti

V řadě aplikací je třeba rozlišovat mezi „více souvislými“ a „méně souvislými“ grafy. Např. u silniční sítě nás může zajímat, na kolika místech by musela být přerušena silnice (např. povodní), aby bylo přerušeno veškeré silniční spojení mezi dvěma danými místy, popř. aby se silniční síť rozpadla na několik částí, mezi nimiž není silniční spojení.

4.4.1 Hranový stupeň souvislosti grafu je definován jako minimální počet hran, jejichž odstraněním se graf stane nesouvislým. Tím je hranový stupeň souvislosti definován pro grafy s alespoň dvěma vrcholy. Pro grafy s jedním vrcholem definujeme hranový stupeň souvislosti jako nulu. Graf nazýváme *hranově k-souvislým*, je-li jeho hranový stupeň souvislosti alespoň k.

Hranový stupeň souvislosti nemůže být větší než stupeň kteréhokoli vrcholu. (Proč?)

Hranu grafu nazýváme *mostem*, jestliže jejím odstraněním zvýšíme počet komponent souvislosti. Graf, který má alespoň dva vrcholy, je tedy hranově 2-souvislý, neobsahuje-li žádný most.

4.4.2 Vrcholový stupeň souvislosti grafu je definován jako minimální počet vrcholů, jejichž odstraněním se graf stane nesouvislým. Tím je vrcholový stupeň souvislosti definován pro všechny grafy kromě úplných grafů (a kromě multigrafů, které obsahují úplný graf jako svůj faktor). Pro tyto grafy definujeme vrcholový stupeň souvislosti jako $|V(G)| - 1$. Graf nazýváme *vrcholově k-souvislým*, je-li jeho vrcholový stupeň souvislosti alespoň k.

Vrchol grafu nazýváme *artikulací*, jestliže jeho odstraněním zvýšíme počet komponent souvislosti. Graf, který má alespoň 3 vrcholy, je tedy vrcholově 2-souvislý, neobsahuje-li žádnou artikulaci.

4.4.3 Mengerova věta. Graf je vrcholově k-souvislý právě tehdy, když pro každé dva vrcholy x, y existuje k vrcholově disjunktních cest z x do y, tj. takových cest, že kromě vrcholů x, y nemají společné vrcholy.

4.4.4 Věta. (Ford a Fulkerson 1956.) Graf je hranově k-souvislý právě tehdy, když pro každé dva vrcholy x, y existuje k hranově disjunktních cest z x do y, tj. takových cest, že různé cesty nemají žádnou společnou hranu.

4.4.5 Algoritmy pro stupně souvislosti. Vrcholový a hranový stupeň souvislosti grafu lze určit pomocí toků v sítích, viz. 8.2.2. Pomocí toků v sítích lze také dokázat předchozí dvě věty. Pro zjišťování, zda daný graf je vrcholově 2-souvislý, nebo vrcholově 3-souvislý, existují speciální, velmi rychlé, ale komplikované algoritmy.

4.4.6 Cvičení. Na základě vět 4.4.3 a 4.4.4 dokažte, že vrcholový stupeň souvislosti je menší nebo roven hranovému stupni souvislosti. Najděte příklad, kdy neplatí rovnost.

Kapitola 5

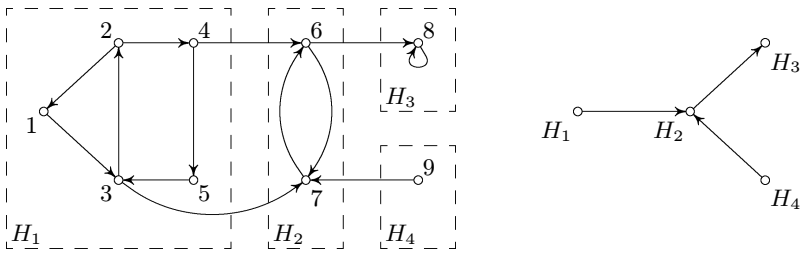
Pojmy založené na orientovaných cestách

5.1 Silná souvislost

5.1.1 Silná souvislost. Orientovaný graf G nazýváme *silně souvislým*, jestliže pro každou dvojici jeho vrcholů x, y existuje v G orientovaná cesta z x do y a také zpět z y do x.

Silně souvislou komponentou grafu G (též *silnou komponentou* nebo *komponentou silné souvislosti*) nazýváme podgraf H grafu G, který je silně souvislý a který je maximální s touto vlastností, tj. není částí většího silně souvislého podgrafu.

5.1.2 Příklad. Graf G na obr. 5.1 má čtyři silně souvislé komponenty H_1, H_2, H_3, H_4. Podgraf indukovaný množinou $\{1, 2, 3\}$ je silně souvislý, ale není silně souvislou komponentou, neboť je částí většího silně souvislého podgrafu H_1. Všimněte si, že graf G je souvislý, ale nikoli silně souvislý. Dále si všimněte, že v komponentě H_1 sice každá hrana leží na nějakém cyklu, ale neexistuje cyklus, který by procházel přes všechny vrcholy.



Obrázek 5.1: Orientovaný graf, jeho silné komponenty a kondenzace.

5.1.3 Cvičení. Uvažujme následující graf: Vrcholy grafu jsou jednotlivé podmínky $1, 2, \ldots, 8$ z věty 4.2.9, str. 61. Z vrcholu i do vrcholu j vede hrana, když v důkaze věty 4.2.9 je dokázána implikace $i \Rightarrow j$. Rozhodněte, zda tento graf je silně souvislý. Totéž proveďte s větou 5.3.4, str. 80. Nakreslete obrázky. Jaký je smysl silné souvislosti a silně souvislých komponent v těchto případech?

5.1.4 Vlastnosti silných komponent. Na množině vrcholů grafu G definujme relaci $\sim$ předpisem

$$x \sim y \quad \Longleftrightarrow \quad \text{existuje orientovaná cesta z } x \text{ do } y \text{ a také z } y \text{ do } x.$$

Velmi snadno lze ověřit, že takto definovaná relace $\sim$ je ekvivalencí (tj. je reflexivní, symetrická a tranzitivní, viz 1.12.6), a že tedy určuje na množině vrcholů rozklad na třídy ekvivalence (viz 1.12.7). Podgrafy, které jsou indukovány těmito třídami, jsou komponentami silné souvislosti (neboť jsou to maximální silně souvislé podgrafy).

Odtud okamžitě plyne, že každý vrchol grafu leží v přesně jedné komponentě silné souvislosti.

5.1.5 Věta. Hrana e je obsažena v nějakém cyklu právě tehdy, když oba její vrcholy (počáteční i koncový) leží v téže silně souvislé komponentě.

Důkaz: Vezměme libovolnou hranu e z vrcholu x do vrcholu y. Je-li $x = y$, tvoří sama hrana e cyklus. Je-li $x \neq y$ a leží-li x, y v téže silně souvislé komponentě, existuje orientovaná cesta z y do x. Tato cesta spolu s hranou e tvoří cyklus.

Je-li naopak hrana e obsažena v cyklu, pak tato hrana tvoří orientovanou cestu z x do y a ostatní hrany cyklu tvoří orientovanou cestu z y do x. Oba vrcholy x, y tedy leží ve stejné silně souvislé komponentě. □

5.1.6 Důsledek. Graf je silně souvislý právě tehdy, když je (slabě) souvislý a když každá jeho hrana leží v nějakém cyklu.

5.1.7 Důsledek. Každá silně souvislá komponenta, která obsahuje alespoň dva různé vrcholy, obsahuje cyklus.

5.1.8 Kondenzace. *Kondenzace* grafu G je prostý orientovaný graf G' bez smyček takový, že platí:

1. Vrcholy grafu G' jsou všechny silně souvislé komponenty grafu G.
2. Vrcholy H_1, H_2 (tj. silně souvislé komponenty grafu G) jsou v grafu G' spojeny hranou z H_1 do H_2 právě tehdy, když $H_1 \neq H_2$ a v grafu G existuje hrana z některého vrcholu komponenty H_1 do některého vrcholu komponenty H_2.

5.1.9 Věta. Kondenzace orientovaného grafu neobsahuje žádný cyklus.

Důkaz: Kondenzace (podle definice) neobsahuje smyčky. Kdyby kondenzace obsahovala cyklus procházející alespoň dvěma různými vrcholy (tj. silně souvislými

komponentami původního grafu), existoval by již v původním grafu cyklus procházející přes tyto komponenty. To by však bylo v rozporu s definicí silně souvislé komponenty. □

5.1.10 Dostupnost. Označme (viz 1.5.4, str. 20)

$D_G^+(x)$ = množina všech vrcholů orientovaně dostupných v grafu G z vrcholu x,

$D_G^-(x)$ = množina všech vrcholů, z nichž je v grafu G orientovaně dostupný vrchol x.

Nebude-li hrozit nedorozumění, budeme místo $D_G^+(x)$ a $D_G^-(x)$ psát stručněji $D^+(x)$ a $D^-(x)$. Dále zaveďme *matici dostupnosti* D takto:

$$d_{ij} = \begin{cases} 1, & \text{jestliže vrchol } v_j \text{ je orientovaně dostupný z vrcholu } v_i, \\ 0 & \text{v ostatních případech.} \end{cases}$$

V definici dostupnosti jsme samozřejmě mohli se stejným výsledkem místo sledu použít cestu.

Množiny $D^+(x)$ lze snadno nalézt pomocí algoritmů pro prohledávání grafu uvedených v kapitole 3, str. 49. Množiny $D^-(x)$ lze nalézt použitím týchž algoritmů na obrácený graf (graf, který získáme obrácením orientace všech hran). Jinou možností je upravit algoritmy prohledávání tak, aby hledaly cesty „proti směru hran“. Časové nároky jsou v obou případech $O(m+n)$.

Matici dostupnosti lze sestrojit po řádcích n-násobným použitím některého z těchto algoritmů v čase $O(n(m+n))$. Elegantnější postup s časovým odhadem $O(n^3)$ je založen na 5.4.2, str. 82, 7.2.3, str. 116, a 7.3.4, str. 122.

5.1.11 Věta. Silně souvislá komponenta obsahující vrchol x má množinu vrcholů rovnu $D^+(x) \cap D^-(x)$.

Důkaz bezprostředně vyplývá z 5.1.4 a z definic množin $D^+(x)$ a $D^-(x)$. □

5.1.12 Hledání silně souvislých komponent. Předchozí věta dává jednoduchý návod k sestrojení silně souvislé komponenty, která obsahuje daný vrchol r.

Všechny silně souvislé komponenty pak lze snadno sestrojit opakováním tohoto postupu: zvolíme vrchol, který neleží v žádné dosud nalezené komponentě, a najdeme silnou komponentu, která tento vrchol obsahuje. Výpočet končí, jsou-li všechny vrcholy zařazeny do komponent.

Tento primitivní postup vyžaduje čas $O(k(m+n))$, kde m je počet hran, n počet vrcholů a k počet silně souvislých komponent. V krajním případě, např. pro „téměř acyklický“ graf, to je $O(mn)$. Proto uvedeme (bez důkazu) i podstatně rychlejší algoritmus, jehož časový odhad $O(m+n)$ je lineární:

5.1.13 Tarjanův algoritmus pro hledání silně souvislých komponent uvedeme jako ukázku důmyslného postupu, který je založen na prohledávání grafu do hloubky (viz 3.3.1, str. 54). V podstatě jde o běžné prohledávání do hloubky, v němž se však navíc provádí pomocné operace vždy při příchodu do nového vrcholu, při zpracování hrany a při návratu z vrcholu.

Algoritmus přiřazuje každému vrcholu x dvě číselné hodnoty, budeme je značit $\textsc{Dfs}(x)$ a $\textsc{Vzad}(x)$. Hodnoty $\textsc{Dfs}(x)$ jsou pořadová čísla přidělovaná v pořadí, v jakém byly vrcholy poprvé navštíveny.[1]

Hodnota $\textsc{Vzad}(x)$ představuje „zpětný odkaz“: bude rovna nejnižšímu pořadovému číslu $\textsc{Dfs}(y)$ některého vrcholu y, který byl navštíven dříve než x (tj. má nižší Dfs-číslo), je orientovaně dostupný z x po hranách dříve použitých pro postup do hloubky a po nejvýše jedné další hraně a přitom existuje vrchol z, ležící ve stejné silné komponentě jako y a z něhož je dostupný jak vrchol x, tak i vrchol y. Pokud takový vrchol y neexistuje, bude $\textsc{Vzad}(x) = \textsc{Dfs}(x)$. Vypadá to složitě, ale ve skutečnosti je manipulace s čísly Vzad velmi jednoduchá a spočívá ve třech věcech:

- Přijdeme-li poprvé do vrcholu v, bude $\textsc{Vzad}(v) = \textsc{Dfs}(v)$.
- Najdeme-li hranu, která vede z vrcholu v do vrcholu w, který již byl navštíven, ale ještě není zařazen do nějaké komponenty (tj. leží v tzv. komponentovém zásobníku, viz dále), pak, pokud $\textsc{Dfs}(w) < \textsc{Vzad}(v)$, snížíme hodnotu na $\textsc{Vzad}(v) := \textsc{Dfs}(w)$.
- Při návratu z vrcholu w do vrcholu v kontrolujeme, zda $\textsc{Vzad}(v) \leq \textsc{Vzad}(w)$, a pokud neplatí, snížíme hodnotu na $\textsc{Vzad}(v) := \textsc{Vzad}(w)$.

Vtip algoritmu je v tom, že $\textsc{Vzad}(x)$ bude vždy Dfs-číslem vrcholu ze stejné silně souvislé komponenty jako x. Vždy bude $\textsc{Vzad}(x) \leq \textsc{Dfs}(x)$, a pokud bude při návratu z vrcholu x platit $\textsc{Vzad}(x) < \textsc{Dfs}(x)$, pak vrchol, do kterého se vracíme, bude ve stejné silné komponentě jako vrchol x.

Pro shromažďování vrcholů, které patří do stejné komponenty, se v tomto algoritmu používá zásobník. Je třeba zdůraznit, že jde o zcela jiný (další) zásobník, než který se používá při hledání do hloubky pro zapamatování návratů (viz 3.3.1, str. 55) a který je zde skryt v mechanismu volání rekurzivní procedury (viz 3.3.9, str. 57). Pro odlišení mu budeme říkat komponentový zásobník.

Každý vrchol bude na konec komponentového zásobníku zařazen ve chvíli, kdy bude poprvé navštíven, ale odstranění ze zásobníku budeme provádět, teprve až bude na konci zásobníku shromážděna celá silná komponenta.

Lze dokázat, že silné komponenty se v komponentovém zásobníku nikdy nepromíchají mezi sebou. Vrchol x, který byl z celé komponenty navštíven jako první (a má tedy nejnižší $\textsc{Dfs}(x)$), poznáme podle toho, že pro něj v okamžiku návratu z tohoto vrcholu platí $\textsc{Vzad}(x) = \textsc{Dfs}(x)$. Lze dokázat, že vrchol x spolu s vrcholy, které v této chvíli jsou v komponentovém zásobníku za ním, tvoří (celou) silně souvislou komponentu obsahující vrchol x. Celou komponentu tedy máme pěkně pohromadě, můžeme ji tedy zaznamenat a odstraníme ji z komponentového zásobníku.

Proměnnou *čítač_vrcholů* použijeme pro přidělování čísel Dfs a Vzad, proměnná *čítač_komponent* slouží k číslování nalezených komponent.

Hodnota $\textsc{Komponenta}(x)$ bude pro každý vrchol x nejprve rovna -1, po zařazení do komponentového zásobníku bude $\textsc{Komponenta}(x) = 0$ a konečně po

[1] Značení $\textsc{Dfs}(x)$ je odvozeno z anglického *Depth First Search*, což znamená hledání do hloubky.

odstranění vrcholu x z komponentového zásobníku bude KOMPONENTA(x) rovna kladnému pořadovému číslu silné komponenty, která vrchol x obsahuje.

```
program SilneKomponenty;
  procedure HLEDEJ (v : vrchol);
  begin
    čítač_vrcholů := čítač_vrcholů + 1;
    DFS(v) := čítač_vrcholů; VZAD(v) := DFS(v);
    zařaď v do komponentového zásobníku;
    KOMPONENTA(v) := 0;
    pro všechny následníky w ∈ V+(v) proveď
      begin
        if (KOMPONENTA(w) = −1) then
          begin
            {w je poprvé navštívený vrchol}
            HLEDEJ(w);
            {po návratu z vrcholu w upravíme hodnotu VZAD(v)}
            VZAD(v) := min(VZAD(v), VZAD(w))
          end
        else if (KOMPONENTA(w) = 0) then
          begin
            {w je v komponentovém zásobníku}
            VZAD(v) := min(VZAD(v), DFS(w));
          end;
      end;
    {před návratem z vrcholu v testujeme hranici komponenty}
    if (VZAD(v) = DFS(v)) then
      begin
        {v je prvým vrcholem silné komponenty v zásobníku}
        čítač_komponent := čítač_komponent + 1;
        repeat
          vyjmi vrchol z komponentového zásobníku a označ jej w;
          KOMPONENTA(w) := čítač_komponent;
        until w = v;
      end;
  end; {konec procedury HLEDEJ}

begin {vlastní prohledávání}
  čítač_vrcholů := 0; čítač_komponent := 0;
  pro všechny vrcholy x proveď KOMPONENTA(x) := −1;
  pro všechny vrcholy x proveď
    if (KOMPONENTA(x) = −1) then HLEDEJ(x);
end.
```

ČASOVÉ NÁROKY: $O(n + m)$.

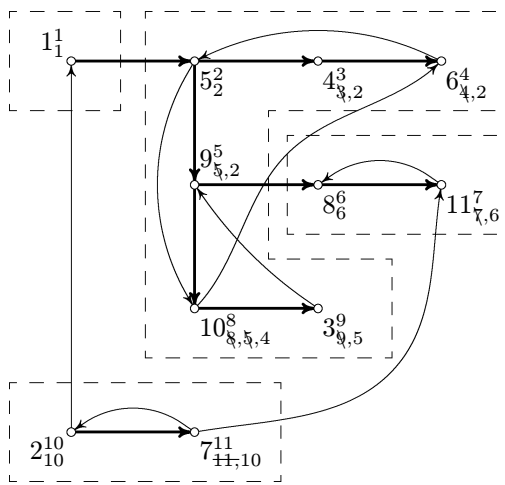
Důkaz správnosti algoritmu i časového odhadu lze najít v knize [AHU74].

5.1.14 Příklad. Použijeme Tarjanův algoritmus pro nalezení silně souvislých komponent v grafu s množinou vrcholů $\{1, \ldots, 11\}$ a množinou hran:

(1,5)	(3,9)	(5,9)	(7,2)	(9,8)	(10,6)
(2,1)	(4,6)	(5,10)	(7,11)	(9,10)	(11,8)
(2,7)	(5,4)	(6,5)	(8,11)	(10,3)	

Následníky každého vrcholu budeme probírat ve vzrůstajícím pořadí jejich čísel.

Při ručním výpočtu je vhodné si graf během prohledávání kreslit tak, aby hrany použité pro postup do hloubky byly co nejpřehlednější (např. zleva doprava a shora dolů jako na obrázku 5.2). Čísla DFS a VZAD zapisujte do obrázku. Nepopleťte číslo x, kterým je vrchol pojmenován, a hodnoty $\textsc{Dfs}(x)$ a $\textsc{Vzad}(x)$. Na našem obrázku jsou hodnoty $\textsc{Dfs}(x)$ uvedeny jako exponent a hodnoty $\textsc{Vzad}(x)$ jako index u čísla vrcholu x.



Obrázek 5.2: K příkladu hledání silně souvislých komponent 5.1.14. Vrcholy jsou označeny podle schématu x^c_z, kde $c = \textsc{Dfs}(x)$ a $z = \textsc{Vzad}(x)$.

Postup výpočtu Tarjanovým algoritmem je tento:

- zjišťujeme, že $\textsc{Komponenta}(1) = -1$, tj. vrchol 1 není v žádné komponentě;
- hledání do hloubky proto zahájíme ve vrcholu 1;
- postup do hloubky do vrcholů 1, 5, 4 a 6, přidělena pořadová čísla 1, 2, 3, 4;
- po zpracování hrany $(6, 5)$ je $\textsc{Vzad}(6) = \textsc{Dfs}(5) = 2$;
- návrat z vrcholu 6 do vrcholu 4, $\textsc{Vzad}(4) = \textsc{Vzad}(6) = 2$;
- návrat z vrcholu 4 do vrcholu 5, komponentový zásobník obsahuje posloupnost $(1, 5, 4, 6)$;
- postup do hloubky do vrcholů 9, 8 a 11, přidělena pořadová čísla 5, 6, 7, komponentový zásobník obsahuje posloupnost $(1, 5, 4, 6, 9, 8, 11)$;

- po zpracování hrany $(11, 8)$ je $\textsc{Vzad}(11) = \textsc{Dfs}(8) = 6$;
- návrat z vrcholu 11 do vrcholu 8, $\textsc{Vzad}(8) = \textsc{Dfs}(8) = 6$;
- vrcholy 11 a 8 odebrány ze zásobníku a zařazeny do komponenty 1;
- návrat z vrcholu 8 do vrcholu 9;
- postup do hloubky do vrcholů 10 a 3, přidělena pořadová čísla 8 a 9, komponentový zásobník obsahuje posloupnost $(1, 5, 4, 6, 9, 10, 3)$;
- po zpracování hrany $(3, 9)$ je $\textsc{Vzad}(3) = \textsc{Dfs}(9) = 5$;
- návrat z vrchol 3 do vrcholu 10, $\textsc{Vzad}(10) = \textsc{Vzad}(3) = 5$;
- po zpracování hrany $(10, 6)$ je $\textsc{Vzad}(10) = \textsc{Dfs}(6) = 4$;
- návrat z vrchol 10 do vrcholu 9, $\textsc{Vzad}(9) = 4$;
- návrat z vrchol 9 do vrcholu 5, $\textsc{Vzad}(5) = \textsc{Dfs}(5) = 2$;
- zpracováním hrany $(5, 10)$ se nic nezmění;
- vrcholy 3, 10, 9, 6, 4 a 5 odebrány ze zásobníku a zařazeny do komponenty 2;
- návrat z vrcholu 5 do vrcholu 1, $\textsc{Vzad}(1) = \textsc{Dfs}(1) = 1$;
- vrchol 1 odebrán ze zásobníku a zařazen do komponenty 3;
- skončilo prohledávání z vrcholu 1, pokračujeme v kontrole, zda každý vrchol je v nějaké silné komponentě;
- nové hledání do hloubky zahájíme z vrcholu 2;
- postup do hloubky do vrcholů 2 a 7, přidělena pořadová čísla 10 a 11, komponentový zásobník obsahuje posloupnost $(2, 7)$;
- po zpracování hrany $(7, 2)$ je $\textsc{Vzad}(7) = \textsc{Dfs}(2) = 10$;
- zpracováním hrany $(7, 11)$ se nic nezmění;
- vrcholy 2 a 7 odebrány ze zásobníku a zařazeny do komponenty 4;
- po návratu z vrcholu 2 pokračujeme v kontrole, zda každý vrchol je v nějaké silné komponentě. Výpočet tím končí.

5.1.15 Cvičení. Použijte Tarjanův algoritmus na graf z příkladu 5.1.14, ale začněte z jiného vrcholu a probírejte hrany v jiném pořadí. Výsledné komponenty budou samozřejmě stejné.

5.2 Acyklické grafy

5.2.1 Acyklický graf. *Acyklický graf* je orientovaný graf, který neobsahuje žádný cyklus (tedy ani smyčku).

Podle 5.1.9 je každá kondenzace acyklickým grafem. Také naopak, každá silně souvislá komponenta acyklického grafu má podle 5.1.7 pouze jeden vrchol, acyklický graf je tedy sám svou kondenzací.

5.2.2 Věta. Každý acyklický graf G obsahuje alespoň jeden vrchol x, pro který $d^+(x) = 0$, a alespoň jeden vrchol y, pro který $d^-(y) = 0$.

Důkaz: Kdyby pro všechny vrcholy platilo $d^+(x) > 0$, existovaly by v grafu orientované sledy o libovolné délce. Totiž z libovolného vrcholu vychází alespoň jedna hrana, označme ji např. e_1, z jejího koncového vrcholu vychází další hrana e_2,

z jejího koncového vrcholu vychází další hrana e_3 atd. Ve sledu delším než $|V(G)|$ se ovšem musí opakovat vrcholy. Úsek takového sledu mezi dvěma nejbližšími výskyty téhož vrcholu by však byl cyklem. To v acyklickém grafu není možné. Proto musí existovat vrchol x, pro který $d^+(x) = 0$.

Důkaz pro $d^-(y)$ je zcela analogický. □

5.2.3 Topologické uspořádání vrcholů a hran. *Topologické uspořádání vrcholů* grafu G je taková posloupnost $v_1, v_2, \ldots, v_n$ všech vrcholů grafu, že kdykoli vede orientovaná hrana z v_i do v_j, platí $i < j$. Jinými slovy: počáteční vrchol hrany je v topologickém uspořádání uveden vždy dříve než koncový vrchol téže hrany.

Topologické uspořádání hran grafu G je taková posloupnost všech hran grafu $e_1, e_2, \ldots, e_m$, že kdykoli $\mathrm{Kv}(e_i) = \mathrm{Pv}(e_j)$, platí $i < j$. Jinými slovy: pro každý vrchol v platí, že všechny hrany z $E^-(v)$ předcházejí všem hranám z $E^+(v)$.

Je zřejmé, že existuje-li topologické uspořádání vrcholů nebo hran, graf je acyklický. Dokážeme, že také naopak v každém acyklickém grafu existuje topologické uspořádání vrcholů i hran.

Často se také mluví o *topologickém očíslování* vrcholů nebo hran. Je to ohodnocení (vrcholů nebo hran) souvislou vzestupnou posloupností přirozených čísel $1, 2, 3, \ldots$ v pořadí, které je dáno topologickým uspořádáním. Seřadíme-li tedy vrcholy (hrany) vzestupně podle topologických čísel, dostaneme topologické uspořádání. Algoritmy, které hledají topologické uspořádání, často poskytují svůj výsledek ve formě topologického očíslování.

5.2.4 Aplikace topologických uspořádání. Představte si, že máte vykonat nějakou množinu činností. Činnosti musíte vykonat jednu po druhé, tj. nemůžete nebo nesmíte dělat více činností najednou a vykonávání žádné činnosti nesmí být přerušeno jinou činností. Vykonání některých činností je navíc podmíněno dokončením některých jiných činností. Například ponožku na levou nohu je třeba obout před obutím levé boty a základy domu se dělají dříve než zdi. Hledáte přípustné pořadí, ve kterém lze vykonat všechny činnosti.

Činnosti můžeme pokládat za vrcholy grafu a podmínky za jeho hrany: je-li činnost j podmíněna dokončením činnosti i, bude v grafu hrana z vrcholu i do vrcholu j. Všechna přípustná pořadí činností pak vzájemně jednoznačně odpovídají všem topologickým uspořádáním vrcholů.

Mimochodem, srovnejte tuto aplikaci se síťovými grafy 2.3.2, str. 44, a najděte podstatné odlišnosti obou úloh.

Další (a významné) použití topologických uspořádání je v algoritmech, které pracují s acyklickými grafy. Některé algoritmy se výrazně zrychlí nebo zjednoduší, pokud nějaký dílčí výpočet provádíme pro všechny vrcholy v pořadí podle topologického uspořádání vrcholů nebo pro všechny hrany v pořadí podle topologického uspořádání hran. Příkladem je výpočet síťového grafu 2.3.2, hledání jádra acyklického grafu 5.5.4 nebo hledání nejkratších cest v acyklických grafech 6.4.

5.2.5 Cvičení. Topologické uspořádání vrcholů často není určeno jednoznačně. Najděte příklad grafu, který má jen jedno topologické uspořádání vrcholů. Najděte

nutnou a postačující podmínku, aby acyklický graf měl jen jedno topologické uspořádání vrcholů.

Topologické uspořádání hran také zpravidla není určeno jednoznačně. Najděte příklad grafu, který má jen jedno topologické uspořádání hran. Najděte nutnou a postačující podmínku, aby acyklický graf měl jen jedno topologické uspořádání hran.

5.2.6 Věta (vlastnosti acyklických grafů). Nechť G je orientovaný graf. Pak následující podmínky jsou ekvivalentní:

1. G je acyklický.
2. G nemá smyčky a každá jeho silná komponenta má jen jeden vrchol.
3. Existuje topologické uspořádání vrcholů grafu G.
4. Existuje topologické uspořádání hran grafu G.

Důkaz: $1 \Leftrightarrow 2$: Vyplývá z vět 5.1.7, str. 70, a 5.1.9, str. 70.

$3 \Rightarrow 1$ a $4 \Rightarrow 1$: Existence cyklu by byla v rozporu s topologickým uspořádáním vrcholů i s topologickým uspořádáním hran.

$1 \Rightarrow 3$ a $1 \Rightarrow 4$: Důkaz plyne z věty 5.2.8 a z algoritmu 5.2.7, který, je-li graf acyklický, obě topologická uspořádání nalezne. □

5.2.7 Algoritmus pro topologické uspořádání. Obě posloupnosti (vrcholů i hran) budeme vytvářet současně.

Jako v_1 vybereme vrchol, do kterého nevede žádná hrana (viz 5.2.2). Vrchol v_1 a všechny hrany z něj vycházející z grafu odstraníme a zařadíme je do příslušných posloupností (vrchol k vrcholům, hrany k hranám). Zbývající graf je opět acyklický, existuje v něm tedy opět vrchol s nulovým vstupním stupněm, označme jej v_2. Tento vrchol a všechny hrany z něj vycházející opět z grafu odstraníme a zařadíme do posloupností. Takto postupujeme, dokud nejsou uspořádány všechny vrcholy a hrany.

Nechceme-li graf během výpočtu měnit (což obvykle nechceme), můžeme se na výše popsaný postup dívat i tak, že pro zařazení do posloupnosti vybíráme vždy vrchol, jehož všichni předchůdci jsou již do posloupnosti zařazeni.

Pro výpočet na počítači se uvedený postup zpravidla dále upravuje tak, aby výběr dalšího zařazovaného vrcholu byl co nejrychlejší. Pro každý vrchol x udržujeme číslo $C(x)$ rovné počtu hran, které do x vedou z vrcholů, které nejsou dosud zařazeny. Hodnoty $C(x)$ se postupně snižují a ty vrcholy, u nichž bylo takto dosaženo nuly, uchováváme ve zvláštním seznamu M jako kandidáty na zařazení do posloupnosti.

Algoritmus:

1. [*Výpočet vstupních stupňů.*] Pro všechny vrcholy $x \in V(G)$ položíme $C(x) :=$ $:= 0$. Pro všechny hrany $e \in E(G)$ provedeme $C(\mathrm{Kv}(e)) := C(\mathrm{Kv}(e)) + 1$.

2. [*Inicializace množiny M.*] Do množiny M vložíme všechny vrcholy x takové, že $C(x) = 0$.

3. [*Výběr vrcholu.*] Je-li $M = \emptyset$, výpočet končí. Je-li $M \neq \emptyset$, odstraníme z M některý vrchol, označíme jej x a zařadíme jej na konec posloupnosti vrcholů.

4. [*Zpracování hran vycházejících z vrcholu x.*] Pro každou hranu $e \in E^+(x)$ provedeme krok 4a a pak pokračujeme krokem 3.

   4a. [*Zpracování hrany e.*] Zařadíme hranu e na konec posloupnosti hran, označíme $y := \text{KV}(e)$ a položíme $C(y) := C(y) - 1$. Jestliže nyní platí $C(y) = 0$, zařadíme vrchol y do množiny M.

ČASOVÉ NÁROKY: $O(m + n)$, kde n je počet vrcholů a m počet hran.

POZNÁMKA: Požadujeme-li výsledek ve formě topologických očíslování vrcholů a hran, stačí při zařazování (nebo namísto zařazování) vrcholu nebo hrany do příslušné posloupnosti přiřadit vrcholu nebo hraně vždy další přirozené číslo. Jsou k tomu potřebné dva čítače, jeden pro čísla vrcholů a jeden pro čísla hran.

5.2.8 Věta. Jsou-li po zastavení algoritmu 5.2.7 zařazeny do posloupnosti vrcholů všechny vrcholy, jsou obě vytvořené posloupnosti (vrcholů i hran) topologickým uspořádáním.

Nejsou-li po zastavení algoritmu 5.2.7 zařazeny do posloupnosti vrcholů všechny vrcholy, graf není acyklický.

DŮKAZ: Během práce algoritmu vždy těsně před vykonáním kroku 3 a těsně po vykonání kroku 4 platí:

$C(x)$ = počet hran, které do x vedou z vrcholů, které dosud nejsou zařazeny do posloupnosti vrcholů,

M = množina všech vrcholů, pro které platí $C(x) = 0$ a které dosud nejsou zařazeny do posloupnosti vrcholů.

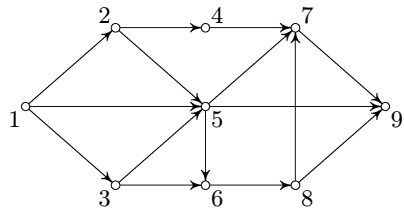
Ve chvíli, kdy je vrchol x zařazován do posloupnosti vrcholů, je $C(x) = 0$, všichni předchůdci vrcholu x jsou již tedy do posloupnosti zařazeni.

Ve chvíli, kdy je zařazována hrana e do posloupnosti hran, pro její počáteční vrchol x platí $C(x) = 0$. Všechny hrany z $E^-(x)$ již tedy byly zpracovány v kroku 4a, a byly tudíž zařazeny do posloupnosti hran.

Pokud by algoritmus nezařadil všechny vrcholy a přesto by došlo k vyprázdnění množiny M, pak by v podgrafu indukovaném nezařazenými vrcholy měly všechny vrcholy kladný vstupní stupeň $C(x)$. Tento podgraf by tedy podle věty 5.2.2 obsahoval cyklus, tudíž by ani původní graf nebyl acyklický. Algoritmus tedy zařadí všechny vrcholy, a tedy i všechny hrany. □

5.2.9 Topologické uspořádání hledáním do hloubky. Při prohledávání grafu do hloubky jsou návraty z vrcholů (tj. krok 5 algoritmu 3.3.1) prováděny v pořadí, které je obrácením topologického uspořádání vrcholů.

5.2.10 Cvičení. Použijte oba algoritmy 5.2.7 i 5.2.9 k topologickému uspořádání vrcholů grafu z obr. 5.3. Najděte všechna topologická uspořádání vrcholů. Pozor: to, co je napsáno u vrcholů, jsou jména vrcholů, nikoli topologické očíslování.



Obrázek 5.3: Acyklický graf.

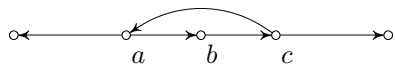
5.2.11 Cvičení. Představte si, že znáte topologické uspořádání vrcholů. Navrhněte jednoduchý způsob, jak z něj získat topologické uspořádání hran. Navrhněte i opačný postup (z topologického uspořádání hran topologické uspořádání vrcholů).

5.2.12 Topologické uspořádání a kořenové stromy. Každý kořenový strom je acyklickým grafem. Pořadí vrcholů preorder definované v 1.7.2, str. 23, je speciálním případem topologického uspořádání vrcholů. Podobně pořadí postorder je obráceným topologickým uspořádáním vrcholů. Srovnejte algoritmy 3.3.13, str. 58, s algoritmem 5.2.9.

5.3 Kořen a kořenový strom

5.3.1 Kořen grafu. Vrchol $x \in V(G)|$ je *kořen* grafu G, jestliže $D^+(x) = V(G)$, tj. jestliže každý vrchol grafu G je orientovaně dostupný z vrcholu x.

Graf na obr. 5.4 má tři kořeny a, b, c.



Obrázek 5.4: Graf se třemi kořeny.

Poznamenejme, že graf je silně souvislý právě tehdy, když každý jeho vrchol je jeho kořenem. Množina všech kořenů tvoří vždy silně souvislou komponentu.

5.3.2 Kořenový strom. Připomeňme, že v 1.7.1, str. 21, jsme definovali kořenový strom jako orientovaný graf, v němž existuje význačný vrchol r, tzv. *kořen*, takový, že do kořene nevede žádná hrana, do každého jiného vrcholu vede přesně jedna hrana a navíc jsou všechny vrcholy z kořene r orientovaně dostupné.

Ve větě 5.3.4 ukážeme, že kořenový strom jsme mohli také ekvivalentně definovat jako graf, který je stromem a má kořen.

5.3.3 Věta. V každém orientovaném grafu, který má kořen, existuje faktor, který je kořenovým stromem.

Důkaz: Vezměme kořen grafu jako výchozí vrchol a použijme některý z algoritmů pro prohledávání grafů z kapitoly 3 s úpravou 3.1.4. Hrany obsažené v hodnotách Odkud tvoří kořenový strom. (Srovnejte s důkazem tvrzení 4.2.11, str. 62.) □

5.3.4 Věta. Nechť G je orientovaný graf s n vrcholy, $n \geq 1$. Potom následující podmínky jsou ekvivalentní:

1. G má kořen a je stromem.
2. G má kořen x a do každého vrcholu vede přesně jeden orientovaný sled z x (tj. z x do x vede pouze triviální sled).
3. G má kořen a po odstranění libovolné hrany již kořen nemá.
4. G má kořen x takový, že $d^-(x) = 0$ a pro každý jiný vrchol $y \neq x$ platí $d^-(x) = 1$.
5. G neobsahuje cyklus a obsahuje vrchol x takový, že $d^-(x) = 0$ a pro každý vrchol $y \neq x$ platí $d^-(y) = 1$.
6. G má kořen a neobsahuje kružnici.
7. G má kořen a má nejvýše $n - 1$ hran.

Důkaz: Důkaz se do značné míry opírá o analogickou větu 4.2.9, str. 61, která charakterizuje stromy.

$1 \Rightarrow 2$: Z vlastnosti kořene grafu plyne, že do každého vrcholu z něj vede alespoň jeden orientovaný sled. Kdyby tam vedly dva různé sledy, bylo by možno z nich vybrat kružnici (po jednom sledu vpřed, po druhém vzad, popř. část sledu, která je cyklem). To by ale byl spor s předpokladem, že graf G je strom.

$2 \Rightarrow 3$: Vezměme libovolnou hranu e. Do jejího počátečního vrcholu u vede přesně jeden sled z kořene grafu, do jejího koncového vrcholu v také. To znamená, že onen jediný sled do vrcholu v obsahuje hranu e, jinak by existovaly sledy dva. Po odstranění hrany e již proto neexistuje žádný sled z kořene grafu do vrcholu v.

$3 \Rightarrow 4$: Z existence kořene plyne $d^-(y) \geq 1$ pro $y \neq x$. Kdyby pro nějaký vrchol y bylo $d^-(y) > 1$, existovaly by alespoň dvě hrany končící v y. Poněvadž do jejich počátečních vrcholů vedou sledy z kořene x, vrchol x by zůstal kořenem i po odstranění některé z hran končících v y. Tím je dokázáno, že $d^-(y) = 1$ pro všechna $y \neq x$. Kdyby nějaká hrana končila v kořeni x, bylo by ji možno odstranit, aniž by vrchol x přestal být kořenem. Z toho plyne, že $d^-(x) = 0$.

$4 \Rightarrow 5$: Z existence kořene plyne (slabá) souvislost. Z vlastností vstupních stupňů plyne, že graf G má přesně $n-1$ hran. G je tedy stromem, a proto neobsahuje kružnici. Nemůže tedy obsahovat ani cyklus.

$5 \Rightarrow 6$: Vyjděme z libovolného vrcholu $y \neq x$ a procházejme grafem proti směru hran. Z každého vrcholu $z \neq x$ můžeme pokračovat pouze jedním způsobem, neboť $d^-(z) = 1$. Žádným vrcholem nemůžeme projít dvakrát, neboť graf neobsahuje cyklus. Naše cesta tedy končí ve vrcholu x, pro který platí $d^-(x) = 0$. Vrchol x je tedy kořenem grafu G.

$6 \Rightarrow 7$: Z existence kořene plyne (slabá) souvislost, proto graf G je stromem, a obsahuje tedy přesně $n-1$ hran.

$7 \Rightarrow 1$: Z existence kořene plyne (slabá) souvislost, graf G je tedy stromem. □

5.3.5 Cvičení. Bylo by možné přidat k větě 5.3.4 následující tři tvrzení?

8. G je souvislý a obsahuje vrchol x takový, že $d^-(x) = 0$ a pro každý vrchol $y \neq x$ platí $d^-(y) = 1$.
9. G neobsahuje kružnici a obsahuje vrchol x takový, že $d^-(x) = 0$ a pro každý vrchol $y \neq x$ platí $d^-(y) = 1$.
10. G má kořen a neobsahuje cyklus.

Jinak řečeno, jsou i tato tři tvrzení ekvivalentní s tím, že graf G je kořenovým stromem?

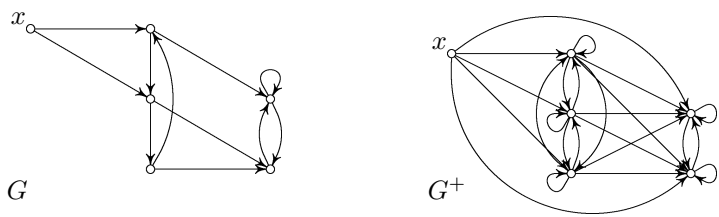
5.4 Tranzitivní uzávěr a redukce

Prostý graf můžeme pokládat za množinu s binární relací (viz 1.3.5, str. 17, a 1.12.4, str. 34). Některé užitečné vlastnosti prostých grafů proto můžeme s výhodou studovat pomocí relací. Všechny grafy, o nichž bude řeč v tomto oddíle, budou orientované a prosté.

5.4.1 Tranzitivní graf, tranzitivní uzávěr. Prostý graf $G = (V, R)$ je *tranzitivní*, jestliže relace R je tranzitivní, tj. jestliže $((x, y) \in R \,\&\, (y, z) \in R) \Rightarrow (x, z) \in \in R$. Graf je *reflexivní*, jestliže relace R je reflexivní, tj. jestliže graf obsahuje všechny smyčky.

Tranzitivní uzávěr grafu G je nejmenší tranzitivní graf, který obsahuje G jako svůj podgraf. Značíme jej G^+.

Tranzitivní a reflexivní uzávěr grafu G je nejmenší graf, který je reflexivní a tranzitivní a který obsahuje graf G jako svůj podgraf. Značíme jej G^*.



Obrázek 5.5: Orientovaný graf a jeho tranzitivní uzávěr.

Na obr. 5.5 je nakreslen graf G a jeho tranzitivní uzávěr G^+. Tranzitivní a reflexivní uzávěr G^* bychom z grafu G^+ dostali přidáním smyčky ve vrcholu x.

Je zřejmé, že tranzitivní i tranzitivní a reflexivní uzávěr existuje ke každému grafu – stačí přidat hrany, které jsou nutné pro splnění tranzitivity a případně

reflexivity. Tento postup je sice hodně neefektivní, ale je z něj patrné, že každý graf je faktorem svého tranzitivního i tranzitivního a reflexivního uzávěru.

Následující věta shrnuje nejzajímavější vlastnosti tranzitivního uzávěru.

5.4.2 Věta (vlastnosti tranzitivního uzávěru). Nechť G, H jsou prosté grafy. Pak platí:

1. V grafu G^* vede hrana z vrcholu i do vrcholu j právě tehdy, když v grafu G existuje sled z i do j.
2. V grafu G^+ vede hrana z vrcholu i do vrcholu j právě tehdy, když v grafu G existuje netriviální sled z i do j.
3. Matice sousednosti grafu G^* je rovna matici dostupnosti grafu G (viz 5.1.10, str. 71).
4. Platí $(G^*)^* = G^*$, $(G^+)^+ = G^+$.
5. Jestliže G je podgrafem grafu H, pak také G^+ je podgrafem H^+ a G^* je podgrafem H^*.

Důkaz: Tvrzení 3, 4 a 5 vyplývají okamžitě z tvrzení 1 a 2. Dokážeme pouze tvrzení 2, tvrzení 1 se dokazuje podobně.

Uvažujme graf G' definovaný takto: graf G' má stejnou množinu vrcholů jako graf G, tj. $V(G') = V(G)$, a v grafu G' vede orientovaná hrana z vrcholu i do vrcholu j právě tehdy, když v původním grafu G vede netriviální orientovaný sled z i do j. Ukážeme, že platí $G^+ = G'$.

Je zřejmé, že graf G' je tranzitivní: Existují-li sledy z i do j a z j do k, pak existuje také sled z i do k. Tedy G' obsahuje graf G^+ (neboť G^+ je nejmenší tranzitivní). Zbývá dokázat, že také G^+ obsahuje G', tj. každá hrana z G' leží také v G^+: Nechť (x, y) je hrana ležící v G'. Pak v G existuje sled procházející vrcholy $x = v_0, v_1, v_2, \ldots, v_k = y$. Pak ale v G^+ musí být díky tranzitivitě obsaženy hrany $(v_0, v_1), (v_0, v_2), (v_0, v_3), \ldots, (v_0, v_k) = (x, y)$. □

5.4.3 Hledání tranzitivního a reflexivního uzávěru spočívá ve zjišťování orientované dostupnosti. V grafu o n vrcholech to lze udělat buď n-násobným prohledáním grafu v čase $O(n(m + n))$, nebo podle 7.2.3, str. 116, a algoritmem 7.3.1 v čase $O(n^3)$.

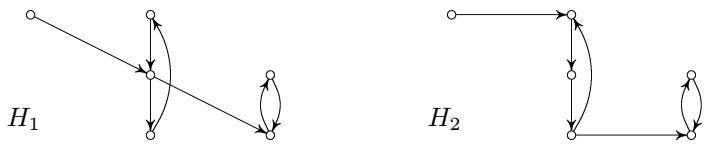
5.4.4 Tranzitivní redukce. *Tranzitivní redukce* grafu G je minimální podgraf grafu G, který má stejný tranzitivní uzávěr jako graf G.

Jinými slovy: Graf H je tranzitivní redukcí grafu G, jestliže:

1. H je podgrafem G.
2. $H^+ = G^+$.
3. Je-li H' podgrafem grafu H a $H \neq H'$, pak H' již nesplňuje podmínku 2.

Na obr. 5.6 jsou nakresleny dva grafy H_1, H_2, z nichž každý je tranzitivní redukcí grafu G z obr. 5.5.

Je zřejmé, že tranzitivní redukce není vždy určena jednoznačně.



Obrázek 5.6: Dvě různé tranzitivní redukce grafu G z obr. 5.5.

Snadno lze nahlédnout, že tranzitivní redukce existuje (alespoň jedna) pro každý graf a že tranzitivní redukce je vždy faktorem původního grafu.

5.4.5 Cvičení. Najděte graf, který má dvě různé tranzitivní redukce s různým počtem hran. (Návod: stačí tři vrcholy.)

5.4.6 Hledání tranzitivní redukce. Nejprve popíšeme velmi jednoduchý postup. Probíráme postupně všechny hrany grafu G. Pro každou hranu $e = (x, y)$ zjistíme, zda vrchol y je orientovaně dostupný z vrcholu x po hranách z množiny $R \setminus \{e\}$, tj. bez použití hrany e. Pokud je dostupný, hranu e z grafu odstraníme, neboť i bez ní bude mít graf stejný tranzitivní uzávěr. Pokud dostupný není, pak naopak hranu e vynechat nelze, neboť bez ní by byl tranzitivní uzávěr jiný (menší). Výsledkem tohoto postupu je jistě tranzitivní redukce, neboť v grafu zůstaly pouze hrany, které nelze odstranit, aniž by se změnila relace dostupnosti. Tento přímočarý postup vyžaduje čas $O(m(m + n))$, kde n je počet vrcholů a m je počet hran.

Hledání tranzitivní redukce lze urychlit několika triky:

V daném grafu G najdeme silně souvislé komponenty a nejprve se postaráme o hrany, které vedou mezi silnými komponentami. Sestrojíme kondenzaci K grafu G a najdeme tranzitivní redukci S této kondenzace. Kondenzace K je zpravidla menší než původní graf a navíc je acyklická. K hledání tranzitivní redukce S lze použít výše popsaný jednoduchý postup, rychlejší algoritmus pro acyklické grafy popíšeme dále v 5.4.9. Pokud v grafu S mezi silnými komponentami H_1, H_2 vede hrana, pak v původním grafu G ponecháme ze všech hran mezi H_1, H_2 jen jednu (kteroukoli), ostatní odstraníme, neboť se tím nezmění relace dostupnosti. Pokud v grafu S mezi komponentami H_1, H_2 žádná hrana nevede, pak v původním grafu G odstraníme dokonce všechny hrany mezi komponentami H_1, H_2 a to opět proto, že jejich odstraněním se nezmění relace dostupnosti.

Dále se budeme zabývat každou silnou komponentou zvlášť. Na komponentu použijeme Tarjanův algoritmus 5.1.13. Z hran uvnitř komponenty ponecháme všechny hrany, které byly v Tarjanově algoritmu použity pro postup do hloubky, a z ostatních hran pak pro každý vrchol x ponecháme jen tu hranu, která z něj vede do vrcholu s nejnižším číslem DFS. Ostatní hrany odstraníme. Takto redukovaná kom-

ponenta bude mít nejvýše $2n-2$ hran, má-li n vrcholů, ale bude stále silně souvislá (neboť kdybychom znovu použili Tarjanův algoritmus, vypočetl by tytéž hodnoty DFS a VZAD). Nyní tedy stačí provést výše popsaný test dostupnosti pro hrany z takto redukované komponenty. Celkový časový odhad je $O(n(m+n))$.

5.4.7 Věta. Prostý acyklický graf má vždy jen jednu tranzitivní redukci.

DŮKAZ provedeme sporem: Nechť graf $G = (V, R)$ má dvě různé tranzitivní redukce $G_1 = (V, R_1)$, $G_2 = (V, R_2)$. Ukážeme, že je to ve sporu s acykličností.

Vezměme nějaké topologické očíslování vrcholů grafu. Poněvadž $G_1 \neq G_2$, existuje hrana, která je obsažena pouze v jedné z obou redukcí G_1, G_2. Mezi všemi takovými hranami vybereme tu, jejíž vrcholy mají nejmenší rozdíl topologických čísel. Označme ji e. Předpokládejme, že $e \in R_1 \setminus R_2$ (kdyby naopak $e \in R_2 \setminus R_1$, byl by důkaz podobný). Poněvadž vrchol KV(e) je díky hraně e dostupný z PV(e) v grafech G a G_1, musí být dostupný i v grafu G_2. V G_2 tedy existuje orientovaná cesta C z vrcholu PV(e) do vrcholu KV(e), která má alespoň dvě hrany. Každá hrana z této cesty má rozdíl čísel v topologickém uspořádání ostře menší než hrana e. Všechny hrany cesty C nemohou ležet současně v R_2 i v R_1, neboť jinak by graf G_1 nebyl redukcí (bylo by možné vyřadit hranu e). Tedy některá z hran cesty C leží v G_2, ale nikoli v G_1, přitom však má menší rozdíl čísel než hrana e, což je spor s volbou hrany e.

Acyklický graf tedy nemůže mít dvě různé tranzitivní redukce. □

5.4.8 Věta. Mají-li dva acyklické grafy týž tranzitivní uzávěr, mají také tutéž tranzitivní redukci.

DŮKAZ: Vezměme libovolný acyklický graf G a jeho tranzitivní uzávěr G^+. Tranzitivní redukce obou těchto grafů jsou definovány stejně: minimální podgraf, jehož tranzitivní uzávěr je roven grafu G^+. Z věty 5.4.7 pak plyne, že oba grafy G i G^+ mají stejnou tranzitivní redukci.

Máme-li tedy dva acyklické grafy G, H, které mají stejný tranzitivní uzávěr $G^+ = H^+$, pak všechny tři grafy, totiž G, $G^+ = H^+$ a H, mají stejnou tranzitivní redukci. □

5.4.9 Hledání tranzitivní redukce acyklického grafu. Předchozí věty 5.4.7 a 5.4.8 umožňují hledat tranzitivní redukci acyklického grafu trochu efektivněji, než jsme uvedli v 5.4.4. Rozhodnutí, zda některou hranu můžeme či nemůžeme vynechat (aniž bychom narušili dostupnost), totiž nezáleží na tom, které hrany již byly vynechány dříve. Můžeme tedy postupovat např. takto:

Pro každý vrchol x nalezneme množiny $D^-(x)$ a $D^+(x)$ (viz 5.1.10) a vymažeme všechny hrany, které začínají v $D^-(x) \setminus \{x\}$ a končí v $D^+(x) \setminus \{x\}$.

Tento postup vyžaduje čas $O(n(m+n))$. V knize [Plesník83] je uveden rychlejší algoritmus s časovým odhadem $O(nm_R + m)$, kde m_R je počet hran v tranzitivní redukci.

5.5 Jádro grafu a hry

5.5.1 Jádro grafu je množina vrcholů K taková, že platí $V^+(K) \cap K = \emptyset$ a také $K \cup V^-(K) = V(G)$.

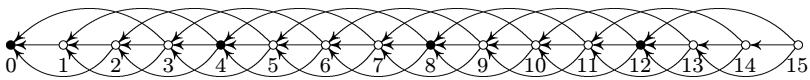
Jinými slovy, z jádra vedou hrany pouze mimo jádro a každý vrchol grafu buď sám leží v jádru, nebo z něj vede alespoň jedna hrana do jádra.

5.5.2 Hra dvou hráčů, která má konečný počet stavů a v níž se hráči střídají v tazích, může být modelována grafem. Vrcholy odpovídají stavům hry, hrany představují přípustné tahy, tj. změny stavů. Průběh hry odpovídá sledu, ve kterém jeden hráč volil vždy liché a druhý hráč sudé hrany. Jestliže hráč nemůže táhnout, tj. jestliže z příslušného vrcholu nevychází žádná hrana, pak tento hráč prohrál. Obsahuje-li graf cyklus, hra nemusí nikdy skončit.

Každého hráče jistě napadla otázka, jak hrát, aby zaručeně neprohrál. Pokud graf hry má jádro, lze neprohrávající strategii vyjádřit větou: „táhni do jádra".

Pokud se vám podaří svým tahem dostat do některého vrcholu jádra, pak sice nelze zaručit, že vyhrajete, ale lze zaručit, že neprohrajete. Pokud totiž soupeř vůbec může táhnout, jeho tah povede ven z jádra. Vy pak můžete táhnout, a to dokonce zpět do jádra. Jinak řečeno, ať soupeř udělá cokoli, vy budete moci ve hře pokračovat, a to způsobem, který vám zachová „záruku dalšího tahu".

5.5.3 Příklad. Na hromádce je 15 zápalek. Dva hráči se střídají v tazích, tah spočívá v odebrání 1–3 zápalek. Kdo nemůže táhnout, tj. kdo už nemá co odebrat, prohrál. Graf této hry je na obr. 5.7, kde jádro grafu je vyznačeno plnými tečkami.



Obrázek 5.7: Graf hry z příkladu 5.5.3.

5.5.4 Hledání jádra. Ne každý orientovaný graf má jádro. Příkladem je graf o třech vrcholech a třech hranách, které tvoří cyklus. Naopak některé grafy mají jader několik. Příkladem je graf o čtyřech vrcholech a čtyřech hranách, které tvoří cyklus. V plné obecnosti je úloha najít jádro dokonce NP-těžká. V řadě speciálních případů však je existence jádra zaručena a mnohdy lze jádro i snadno sestrojit.

Každý acyklický graf má přesně jedno jádro. Lze je sestrojit tak, že probíráme vrcholy v obráceném topologickém pořadí: vrchol x zařadíme do jádra právě tehdy, když žádný jeho následník do jádra zařazen nebyl. Takto lze jádro sestrojit v čase $O(m + n)$.

Také každý orientovaný graf, který neobsahuje cyklus liché délky, má jádro. Důkaz i s návodem k hledání jádra lze najít v knize [Plesník83]. Další podrobnosti o hrách a jádrech lze najít v knihách [Berge73, Berge58].

5.5.5 Cvičení. Ve hře z příkladu 5.5.3 změňme pravidla: prohrává hráč, který odebere poslední zápalku. Graf této úlohy je jistě acyklický, má tedy jádro. Nakreslete tento graf a jádro najděte.

Kapitola 6

Nejkratší cesty

Úlohy o nejkratších cestách patří k nejčastěji aplikovaným úlohám teorie grafů.

6.1 Typy úloh a základní fakta

6.1.1 Úlohy o nejkratších cestách. Mějme orientovaný graf G, jehož každá hrana $e \in E(G)$ je ohodnocena reálným číslem $a(e)$, které nazýváme *délkou hrany*. *Délka cesty* je součet délek jednotlivých hran tvořících cestu (viz 1.5.5, str. 20). Jsou-li x, y dva vrcholy grafu, pak *vzdálenost* $u(x, y)$ z vrcholu x do vrcholu y definujeme jako délku nejkratší cesty z x do y, pokud vůbec nějaká cesta z x do y existuje. Jestliže neexistuje, definujeme vzdálenost $u(x, y) = \infty$.

Úkolem je najít nejkratší orientovanou cestu (popř. vzdálenost):

a) z daného výchozího vrcholu do daného cílového vrcholu,
b) z daného výchozího vrcholu do každého vrcholu grafu,
c) z každého vrcholu do daného cílového vrcholu,
d) mezi všemi uspořádanými dvojicemi vrcholů.

Dále se budeme zabývat pouze řešením úloh b) a d). Úlohu c) lze snadno převést na úlohu b) obrácením orientace hran (nebo jednoduchou úpravou algoritmu). Úloha a) bývá řešena pomocí algoritmů určených pro úlohy b), c) nebo d), postup, který by byl obecně výhodnější, není znám. Některé algoritmy (a to jen v některých typech grafů) jsou však schopny rozpoznat možnost korektního ukončení výpočtu ve chvíli, kdy je vypočtena definitivní hodnota pro cílový vrchol. Na tuto možnost dále v textu vždy upozorníme (viz 6.4.3, 6.5.4 a 6.6).

Ve všech zmíněných úlohách připouštíme i záporné délky hran. Není to takový nesmysl, jak to na prvý pohled vypadá. V roli délek hran totiž mohou vystupovat například náklady na průchod hranou. Záporná délka pak odpovídá situaci, kdy za průchod hranou dostaneme naopak zaplaceno, viz 2.2.8, str. 43. Je ovšem pravdou, že úlohy, kde délky hran jsou nezáporné, jsou v praxi častější a jejich řešení je o něco snazší (viz 6.5, str. 102).

Obtížnost úlohy o nejkratších cestách podstatně závisí na tom, zda graf obsahuje cyklus se zápornou délkou. V této kapitole se převážně budeme zabývat případem, kdy v grafu cyklus se zápornou délkou není a kdy je řešení relativně snadné (existují pro ně polynomiální algoritmy). Pokud v grafu cyklus se zápornou délkou existuje, je obecná úloha o nejkratších cestách NP-těžká. Zjišťováním cyklů se zápornou délkou se budeme zabývat v 6.8.

Hledáme-li nejkratší cesty v multigrafech, můžeme předem z každé množiny rovnoběžných hran vynechat všechny kromě nejkratší z nich, aniž by to mělo vliv na délku nejkratší cesty. Také smyčky můžeme vyloučit, neboť nemohou být obsaženy v žádné cestě.

Pro $i \neq j$ označme $a(i,j)$ délku nejkratší hrany vedoucí z vrcholu i do vrcholu j. Pokud žádná hrana z i do j nevede, položme $a(i,j) = \infty$. Dále položme $a(i,i) = 0$.

Hodnoty $a(i,j)$ můžeme uspořádat do tvaru matice, budeme ji značit A a nazývat *maticí délek hran*. Tato matice obsahuje všechny informace, které jsou třeba k výpočtu vzdáleností.

Podobně můžeme výsledné vzdálenosti $u(i,j)$ uspořádat do tvaru matice, kterou značíme U a nazýváme *maticí vzdáleností*.

6.1.2 Příbuzné úlohy. Hledání nejdelších cest můžeme převést na hledání nejkratších cest obrácením znamének u délek všech hran. Častěji však používáme upravený postup, viz. např. 6.4.5.

Hledání nejkratších neorientovaných cest lze převést na hledání nejkratších orientovaných cest tím, že každou hranu nahradíme dvěma opačně orientovanými hranami téže délky. Pokud délky původních hran byly nezáporné, lze potom použít metody vyložené v této kapitole. Jestliže v původním grafu měla nějaká hrana zápornou délku, vznikne v odvozeném grafu záporný cyklus a úloha je pak podstatně obtížnější. Pokud původní graf obsahuje hrany záporné délky, ale žádnou kružnici záporné délky, lze takovou úlohu převést na hledání nejlevnějšího perfektního párování, viz [Lawler76, Demel91].

Někdy je třeba hledat nejkratší nebo nejdelší cesty v síti, ve které jsou ohodnoceny vrcholy nebo vrcholy i hrany a ve které je délka cesty definována jako součet délek všech vrcholů i hran ležících na této cestě. Hledání nejkratších či nejdelších cest v takové síti lze převést na dříve popsanou úlohu v síti, ve které jsou ohodnoceny pouze hrany. Lze při tom postupovat takto:

Každý vrchol v původního grafu nahradíme dvojicí vrcholů v_1, v_2, spojíme je hranou e_v z vrcholu v_1 do v_2 a této nové hraně dáme hodnotu (délku) původního vrcholu v. Hrany, které v původním grafu končily ve vrcholu v, přesměrujeme, aby končily ve vrcholu v_1. Hrany, které z vrcholu v vycházely, nahradíme hranami, které budou začínat ve vrcholu v_2. Příklad takové úpravy grafu je nakreslen na obr. 8.1, str. 134, viz také obr. 2.3, str. 45 (síťové grafy).

V novém grafu platí, že v každém sledu (a tedy i v každé cestě) se střídají hrany, kterým v původním grafu odpovídají vrcholy, s hranami, kterým v původním grafu odpovídají hrany. Jestliže v původním grafu cesta procházela vrcholem v, pak této cestě v novém grafu odpovídá cesta, která prochází hranou e_v.

6.1.3 Věta. Jestliže v grafu existuje nějaká cesta z vrcholu a do vrcholu b, pak v něm existuje i nejkratší cesta z a do b.

DŮKAZ: Počet hran v cestě je omezen počtem vrcholů grafu. Všech cest z a do b je tedy konečně mnoho, proto některá z nich je nejkratší. □

POZNÁMKA: Pro sledy podobná věta neplatí: Obsahuje-li graf cyklus se zápornou délkou, pak pro některé vrcholy a, b sice existuje sled z a do b, ale neexistuje nejkratší z nich. Ke každému sledu lze totiž vytvořit sled o něco kratší, a to tím, že budeme dostatečně dlouho procházet onen záporný cyklus.

6.1.4 Věta. Jestliže v grafu není cyklus se zápornou nebo nulovou délkou, pak každý nejkratší sled z vrcholu a do vrcholu b je i nejkratší cestou z a do b.

Jestliže v grafu není cyklus se zápornou délkou, pak každý nejkratší sled z a do b obsahuje nejkratší cestu z a do b a délka této cesty je stejná.

DŮKAZ: Prvé tvrzení je snadné: Kdyby nejkratší sled nebyl cestou, obsahoval by cyklus a ten by měl kladnou délku. Jeho vynecháním by bylo možno sled zkrátit, nebyl by tedy nejkratší.

Druhé tvrzení je jen maličko obtížnější: Není-li nejkratší sled již sám cestou, obsahuje cyklus. Délka tohoto cyklu nemůže být ani záporná, ani kladná, je tedy nulová. Vynecháním tohoto cyklu se délka sledu nezmění. Opakovaným vynecháváním cyklů nakonec získáme cestu téže délky jako nejkratší sled, tedy nejkratší cestu. □

6.1.5 Věta (trojúhelníková nerovnost). Jestliže graf neobsahuje cyklus se zápornou délkou, pak pro všechny trojice vrcholů i, j, k vzdálenost u splňuje nerovnost

$$u(i,j) \leq u(i,k) + u(k,j)\ . \tag{6.1}$$

POZNÁMKA: Název je odvozen od analogického vztahu mezi délkami stran trojúhelníka.

DŮKAZ: Je-li $u(i,k) = \infty$ nebo $u(k,j) = \infty$, nerovnost samozřejmě platí. Zabývejme se tedy případem, kdy $u(i,k) < \infty$ a $u(k,j) < \infty$. Spojením nejkratších cest z vrcholu i do vrcholu k a z vrcholu k do vrcholu j získáme sled z i do j. Buď tento sled je sám cestou, nebo z něj cestu vytvoříme vynecháním případných cyklů. Tyto cykly měly nezápornou délku, délka cesty je tedy nejvýše $u(i,k) + u(k,j)$. Zároveň však tato cesta má délku alespoň $u(i,j)$. □

DŮSLEDEK: Jestliže graf neobsahuje cyklus se zápornou délkou, pro každou trojici vrcholů i, j, k platí:

$$\begin{aligned} u(i,j) &\leq a(i,j)\ , \\ u(i,j) &\leq u(i,k) + a(k,j)\ , \\ u(i,j) &\leq a(i,k) + u(k,j)\ . \end{aligned} \tag{6.2}$$

6.1.6 Věta. Předpokládejme, že graf neobsahuje cyklus se zápornou délkou. Jestliže nejkratší cesta z vrcholu i do vrcholu j prochází přes vrchol k, pak počáteční úsek této cesty mezi i a k je nejkratší cestou z i do k, podobně koncový úsek cesty mezi k a j je nejkratší cestou z k do j a navíc platí $u(i,j) = u(i,k) + u(k,j)$.

Důkaz: Označme délku počátečního úseku p a koncového q. Platí tedy $u(i,j) = p + q$ a také $u(i,k) \leq p$, $u(k,j) \leq q$. Kdyby platilo $u(i,k) < p$ nebo $u(k,j) < q$, pak by $u(i,k) + u(k,j) < p + q = u(i,j)$, což by bylo v rozporu s trojúhelníkovou nerovností 6.1.5. Proto $u(i,k) = p$ a $u(k,j) = q$. Oba úseky jsou tedy nejkratšími cestami. □

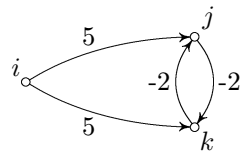
Důsledek: Nechť graf neobsahuje cyklus se zápornou délkou a nechť i, j jsou dva různé vrcholy. Jestliže k je počátečním vrcholem poslední hrany v nejkratší cestě z i do j, pak platí $u(i,j) = u(i,k) + a(k,j)$.

Důsledek: (Bellmanovy rovnice.) Jestliže graf neobsahuje cyklus se zápornou délkou, pak pro každou dvojici vrcholů i, j platí:

$$u(i,j) = \min_{k \neq j} \{u(i,k) + a(k,j)\} \ .$$

Poznámka: Věta 6.1.6 a její důsledky jsou jednou z forem tzv. *Bellmanova principu optimality*, který je základem poměrně obecné optimalizační metody, tzv. *dynamického programování*.

6.1.7 Cykly se zápornou délkou. V grafech, které obsahují cyklus se zápornou délkou, výše uvedené věty 6.1.4 až 6.1.6 ani jejich důsledky neplatí. Jednoduchý příklad je na obr. 6.1.



Obrázek 6.1: Příklad grafu, v němž neplatí trojúhelníková nerovnost.

Algoritmy pro nejkratší cesty, které dále uvádíme, však na platnost těchto vět spoléhají. Rychlost těchto algoritmů je totiž založena na tom, že se nestarají o opakování vrcholů v cestě, takže vlastně nehledají nejkratší cesty, ale nejkratší sledy. V grafech bez záporných cyklů to ovšem vyjde nastejno (viz věta 6.1.4).

Pokud graf obsahuje cyklus se zápornou délkou, nelze tento trik použít, neboť nejkratší sled vůbec nemusí existovat (viz poznámka u 6.1.3). Hledání nejkratších cest pak je NP-těžká úloha (viz kapitola 11). Jednoduchý (ale pracný) postup může být založen na algoritmu 3.3.10, str. 58, pro vyhledání všech cest začínajících v daném vrcholu. Potíž je v tom, že všech cest v grafu o n vrcholech může být řádově až $n!$.

Zjišťováním, zda graf obsahuje cyklus záporné délky, se budeme zabývat v 6.8.

6.1.8 Cvičení. Dokažte důsledky vět 6.1.5 a 6.1.6.

6.2 Základní schéma výpočtu vzdáleností

V tomto oddíle uvedeme velmi primitivní návod k hledání nejkratších cest z daného výchozího vrcholu. Všechny ostatní algoritmy, které dále pro tento typ úlohy uvádíme (kromě oddílu 6.7), jsou vlastně vylepšenými speciálními případy tohoto základního schématu. Pojmy, triky a tvrzení, které v tomto oddíle uvádíme, pak použijeme v následujících oddílech této kapitoly.

6.2.1 Základní schéma výpočtu vzdáleností. Pro každý vrchol x si budeme pamatovat hodnotu $U(x)$, která bude rovna délce nejkratší dosud nalezené cesty z vrcholu r do vrcholu x. Jestliže jsme žádnou cestu z r do x dosud nenašli, bude $U(x) = \infty$.

Kdyby hodnoty U byly skutečnými vzdálenostmi vrcholů grafu od vrcholu r, musela by pro každou hranu grafu (x, y) platit trojúhelníková nerovnost

$$U(y) \leq U(x) + a(x, y) \; . \tag{6.3}$$

Jestliže neplatí, znamená to, že $U(y)$ není délkou nejkratší cesty z r do y, protože přes vrchol x vede do y cesta kratší. Proto v takovém případě hodnotu $U(y)$ snížíme provedením příkazu

$$U(y) := U(x) + a(x, y) \; .$$

V lemmatu 6.2.4 dokážeme, že pak $U(y)$ bude opět délkou některé (a to kratší) cesty z vrcholu r do vrcholu y. Testy trojúhelníkové nerovnosti (6.3) a z nich případně vyplývající úpravy hodnot U budeme provádět tak dlouho, dokud nedosáhneme platnosti trojúhelníkové nerovnosti (6.3) pro všechny hrany grafu. Ve větě 6.2.5 dokážeme, že tento postup musí po konečně mnoha změnách skončit a že dostaneme správný výsledek.

Na začátku výpočtu volíme $U(x) := \infty$ pro $x \neq r$ a $U(r) := 0$, neboť zpočátku neznáme žádnou cestu z vrcholu r kromě cesty triviální, která má nulovou délku. Základní schéma výpočtu vzdáleností lze shrnout takto:

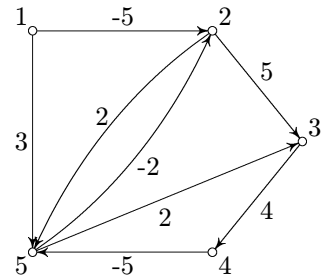
1. Polož $U(r) := 0$ a pro všechny vrcholy $j \neq r$ polož $U(j) := \infty$.

2. Vyber libovolnou hranu grafu e a zkontroluj, zda pro ni platí trojúhelníková nerovnost (6.3), tj. zda $U(\mathrm{Kv}(e)) \leq U(\mathrm{Pv}(e)) + a(e)$. Jestliže neplatí (tj. platí-li opak), proveď $U(\mathrm{Kv}(e)) := U(\mathrm{Pv}(e)) + a(e)$.

3. Jestliže nerovnost (6.3) platí pro všechny hrany grafu, výpočet končí.

Poznamenejme, že táž hrana může být použita i několikrát ke snížení hodnoty U jejího koncového vrcholu. Pořadí výběru hran k testu nerovnosti (6.3) i způsob zjišťování, zda tato nerovnost již platí pro všechny hrany, má, jak uvidíme dále, velký vliv na rychlost výpočtu. Nejprve však uvedeme příklad a vyslovíme několik tvrzení, která platí nejen pro základní schéma výpočtu vzdáleností, ale i pro algoritmy z něj odvozené.

6.2.2 Příklad. Výpočet vzdáleností předvedeme na grafu z obr. 6.2. Výchozím vrcholem je vrchol 1.



Obrázek 6.2: Graf k příkladu 6.2.2.

Uvedeme pouze ty výpočetní kroky, v nichž se hodnota U změnila. Hrany k testu nerovnosti (6.3) jsme vybírali úmyslně trochu chaoticky.

krok	Použitá hrana			hodnoty U po provedení kroku				
výpočtu	PV	KV	délka	$U(1)$	$U(2)$	$U(3)$	$U(4)$	$U(5)$
0	–	–	–	0	∞	∞	∞	∞
1	1	5	3	0	∞	∞	∞	$3\star$
2	5	2	-2	0	$1\star$	∞	∞	3
3	2	3	5	0	1	$6\star$	∞	3
4	3	4	4	0	1	6	$10\star$	3
5	5	3	2	0	1	$5\star$	10	3
6	3	4	4	0	1	5	$9\star$	3
7	1	2	-5	0	$-5\star$	5	9	3
8	2	3	5	0	-5	$0\star$	9	3
9	2	5	2	0	-5	0	9	$-3\star$
10	5	3	2	0	-5	$-1\star$	9	-3
11	3	4	4	0	-5	-1	$3\star$	-3

Hodnota U, která se snížila, je v tabulce označena hvězdičkou. Všimněte si, že hrana $(3, 4)$ byla použita ke snížení hodnoty $U(4)$ třikrát.

6.2.3 Cvičení. Ověřte si na nějakém příkladě (třeba na grafu z obrázku 6.1, str. 90), že pokud graf obsahuje cyklus se zápornou délkou a vrcholy tohoto cyklu jsou dostupné z výchozího vrcholu r, pak výpočet podle schématu 6.2.1 nikdy neskončí. Podrobněji o tom viz 6.8.2, str. 109.

6.2.4 Lemma. Jestliže graf neobsahuje cyklus se zápornou délkou, pak kdykoli během výpočtu vzdáleností podle schématu 6.2.1 pro každý vrchol v buď platí $U(v) = \infty$, nebo je $U(v)$ délkou některé cesty z vrcholu r do vrcholu v.

Důkaz: Očíslujme vzestupně všechny okamžiky, ve kterých dochází ke změnám, tj. ke snížení hodnot U. Každou hodnotu $U(v)$, která byla vypočtena v okamžiku

t, označme $U_t(v)$, hodnoty $U(v)$ platné právě teď označme $U_T(v)$. Je zřejmé, že pro každý vrchol v mohlo být vypočteno postupně několik hodnot $U_t(v)$ a tyto hodnoty tvoří klesající posloupnost.

Je-li $U_T(v) < \infty$, určitě existuje sled $v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$ z vrcholu $r = v_0$ do vrcholu $v = v_k$ a posloupnost okamžiků $0 = t_0 < t_1 < t_2 < \ldots < t_k < T$ taková, že v každém okamžiku t_i byla vypočtena hodnota $U_{t_i}(v_i) = U_{t_{i-1}}(v_{i-1}) + a(e_i)$.

Hodnota $U_T(v) = U_{t_k}(v_k)$ je přitom rovna součtu délek hran e_i, tj. délce sledu. Dokážeme sporem, že tento sled je cestou. Kdyby se v něm nějaký vrchol opakoval, vzali bychom dva k sobě nejbližší výskyty téhož vrcholu $v_i = v_j$ pro $i < j$. Část sledu tvořená hranami $e_{i+1}, e_{i+2}, \ldots, e_j$ by pak tvořila cyklus o délce $U_{t_j}(v_j) - U_{t_i}(v_j)$. Poněvadž však $U_{t_j}(v_j) < U_{t_i}(v_j)$ (neboť $t_i < t_j$), byla by délka tohoto cyklu záporná, což by byl spor. Hodnota $U_T(v)$ tedy je délkou nějaké cesty. □

6.2.5 Věta. Jestliže graf neobsahuje cyklus záporné délky, pak výpočet podle 6.2.1 skončí po konečně mnoha změnách hodnoty U a po ukončení bude pro všechny vrcholy x platit $U(x) = u(r, x)$, tj. vzdálenosti U budou vypočteny správně.

Poznámka: Tato věta nic neříká o počtu testů trojúhelníkové nerovnosti (6.3). Při nešikovné volbě hran pro testování by výpočet nemusel vůbec skončit (kdybychom např. testovali stále tutéž hranu).

Důkaz: Nejprve si uvědomme, že podle předchozího lemmatu je pro každý vrchol x vždy buď $U(x) = \infty$, nebo je $U(x)$ délkou nějaké cesty z r do x. Všech cest začínajících ve vrcholu r je konečně mnoho, proto může během výpočtu dojít jen ke konečně mnoha změnám hodnoty U.

Ukážeme sporem, že po ukončení výpočtu pro každý vrchol v bude $U(v) = u(r, v)$. Předpokládejme, že do některého vrcholu v vede nejkratší cesta o délce $u(r, v) < U(v)$. Označme y první vrchol na této cestě, pro který $U(y) > u(r, y)$. Poněvadž $y \neq r$, existuje vrchol x, který v této cestě bezprostředně předchází y. Pro něj již platí $U(x) = u(r, x)$ a navíc podle důsledku 1 věty 6.1.6 také $u(r, x) + a(x, y) = u(r, y)$. To však znamená, že výpočet nebyl ukončen, neboť $U(x) + a(x, y) < U(y)$, což je spor. □

6.2.6 Konstrukce nejkratších cest. Často potřebujeme zjistit nejen vzdálenosti, tj. délky nejkratších cest, ale také kudy nejkratší cesty vedou. K tomu lze po snadné úpravě použít základní schéma 6.2.1 nebo kterýkoli jeho speciální (a zlepšený) případ popsaný dále v 6.3 až 6.5. Úprava je tato:

Vždy, když dojde ke snížení hodnoty $U(v)$ pro některý vrchol v, zaznamenáme si pro tento vrchol hranu, která snížení způsobila. K zapamatování použijeme hodnotu Odkud(v). Na začátku výpočtu nechť není hodnota Odkud definována pro žádný vrchol.

Z důkazu lemmatu 6.2.4 vyplývá, že je-li hodnota Odkud(v) definována, pak je jménem poslední hrany v nějaké cestě o délce $U(v)$ z vrcholu r do vrcholu v. Hodnoty Odkud pak lze po ukončení výpočtu využít ke zpětné rekonstrukci nejkratších cest (podobně jako v 3.1.4, str. 50). Poznamenejme, že během výpočtu se hodnota Odkud(v) může několikrát změnit.

Dále poznamenejme, že evidování hodnot ODKUD nijak neovlivňuje postup výpočtu podle schématu 6.2.1 nebo podle kteréhokoli jeho zlepšení.

6.2.7 Věta (kořenový strom nejkratších cest). Předpokládejme, že hodnoty ODKUD(v) byly získány podle 6.2.6 a že graf neobsahuje cyklus o záporné délce. Potom po ukončení výpočtu platí:

1. Množina vrcholů $D = \{v \mid \text{ODKUD}(v) \text{ je definováno}\}$ je tvořena všemi vrcholy orientovaně dostupnými z r a různými od r.
2. Je-li ODKUD(v) definováno, pak ODKUD(v) je poslední hranou v (některé) nejkratší cestě z r do v.
3. Pro každou hranu ODKUD(y) platí $U(x) + a(x, y) = U(y)$, kde x je počáteční vrchol hrany ODKUD(y).
4. Množina hran $T = \{\text{ODKUD}(v) \mid v \in D\}$ tvoří kořenový strom s kořenem r na množině vrcholů $D \cup \{r\}$.
5. Každá cesta z kořene r do libovolného vrcholu $x \in D$ v kořenovém stromě T je zároveň nejkratší cestou z r do x v původním grafu.

DŮKAZ:

1. Hodnota ODKUD(v) je definována právě tehdy, když $v \neq r$ a $U(v) < \infty$, což nastává právě tehdy, když $v \neq r$ a vrchol v je orientovaně dostupný z vrcholu r. Hodnota ODKUD(r) není definována, neboť zůstane $U(r) = 0$.

2. Podle věty 6.2.5 je po ukončení výpočtu hodnota $U(v)$ délkou nejkratší cesty z r do v. Hrana, která způsobila poslední snížení hodnoty $U(v)$, je poslední hranou v této nejkratší cestě, jméno této hrany je tedy nakonec obsaženo v ODKUD(v).

3. Těsně po posledním snížení hodnoty $U(y)$ rovnost platila. Hodnota $U(y)$ se od té doby nezměnila a hodnota $U(x)$ také ne, neboť by se tím porušila trojúhelníková nerovnost, hrana by byla znovu zpracována a hodnota $U(y)$ by znovu klesla.

4. V každém vrcholu z množiny D končí přesně jedna hrana z množiny T, ve vrcholu r nekončí žádná. K důkazu, že jde o kořenový strom, tedy podle věty 5.3.4 stačí dokázat, že množina T neobsahuje cyklus. Kdyby obsahovala, platila by pro každou hranu (x, y) tohoto cyklu rovnost $U(x) + a(x, y) = U(y)$. Poslední hrana tohoto cyklu se však nemůže do množiny T dostat, protože všechny vrcholy cyklu již mají definitivní hodnoty U a hodnota ODKUD se může změnit jen spolu se snížením příslušné hodnoty U. Množina T tedy tvoří kořenový strom s kořenem r.

5. Z části 3 věty vyplývá, že délka cesty z r do x tvořené hranami z T je rovna $U(x)$, tato cesta je tedy nejkratší. □

6.2.8 K čemu je kořenový strom nejkratších cest. Nejkratších cest z vrcholu r do vrcholu v může být několik, v kořenovém stromě se samozřejmě objeví jen jedna z nich. Přesto kořenový strom nejkratších cest poskytuje velmi dobrý přehled o struktuře nejkratších cest z daného výchozího vrcholu r. Výborně se hodí k prezentaci výsledků výpočtu, totiž ke sdělení, kudy nejkratší cesty vedou. Například

po výpočtu nejkratších cest v pražské silniční síti stačí do plánu Prahy zakreslit hrany, které náleží ke stromu, a výsledné nejkratší cesty jsou všechny patrné na prvý pohled.

Kořenový strom nejkratších cest a větu 6.2.7 lze také využít ke kontrole správnosti vypočtených vzdáleností: stačí pro všechny hrany grafu zkontrolovat trojúhelníkovou nerovnost (6.3), přičemž pro stromové hrany v ní musí platit rovnost.

Kdybychom předem znali kořenový strom nejkratších cest, mohli bychom výpočet podle schématu 6.2.1 značně urychlit. Stačilo by použít test (6.3) pouze na hrany stromu v topologickém uspořádání (kořenový strom je acyklickým grafem) a tím by výpočet končil. Pak už by totiž byla trojúhelníková nerovnost (6.3) zaručeně splněna i pro ostatní hrany grafu. (Dokažte!)

6.2.9 Počty hran v nejkratších cestách. Někdy je třeba zjistit kromě délky cesty (tj. součtu ohodnocení) také z kolika hran se nejkratší cesta skládá. Samozřejmě to jde spočítat i dodatečně pomocí hodnot Odkud. Často se však používá tato jednoduchá úprava základního schématu 6.2.1 (a všech algoritmů z něj odvozených):

Pro každý vrchol v navíc evidujeme hodnotu $P(v)$, která je vždy rovna počtu hran v cestě, jejíž délka (tj. součet ohodnocení) je $U(v)$.

Při inicializaci položíme $P(r) := 0$ a $P(v) := \infty$ pro $v \neq r$. V průběhu výpočtu pak vždy při změně hodnoty $U(y) := U(x)+a(x,y)$ změníme také $P(y) := P(x)+1$.

V samotném algoritmu nejsou hodnoty P k ničemu dalšímu použity, a proto nijak neovlivní výpočet podle schématu 6.2.1 nebo podle kteréhokoli jeho zlepšení.

Pokud graf neobsahuje cyklus o záporné délce, pak během výpočtu i po jeho skončení platí $P(x) < n$, kde n je počet vrcholů grafu. Toho lze využít ke kontrole, zda graf záporný cyklus obsahuje, nebo nikoli, viz též 6.8.2.

6.2.10 Poznámka k počítání s nekonečnem. Algoritmy, které jsou uvedeny v této kapitole, pracují často s nekonečnem, neboť to zjednodušuje výklad. Běžné počítače a programovací jazyky ovšem s nekonečnem pracovat nedovedou. Jsou dva způsoby, jak si pomoci:

Nejjednodušší je místo symbolu ∞ použít nějakou velkou konstantu, která bude *spolehlivě* větší než dvojnásobek délek všech cest v grafu. Jsou-li např. absolutní hodnoty délek hran menší než 1000 a vrcholů je 50, můžeme místo ∞ použít číslo 100 000 (nebo větší). Čísla větší než 100 000 pak budeme pokládat za „nekonečná“.

Při použití tohoto triku musíme ovšem dávat pozor, abychom ani v mezivýsledcích nepřesáhli maximální číslo zobrazitelné v počítači. Došlo by totiž k tzv. přetečení, což vede ke zcela chybným výsledkům.

Jinou možností je využít hodnotu Odkud: Je-li Odkud(x) jménem nějaké hrany, pak $U(x)$ je délkou nějaké cesty, a tedy $U(x) < \infty$. Jestliže naopak hodnota Odkud(x) není ještě jménem žádné hrany (není dosud definována), pak $U(x) = \infty$. Abychom mohli hodnoty Odkud takto využít, musíme zvolit nějaké fiktivní jméno, odlišné od jmen všech hran, které bude znamenat „nedefinováno“, a při inicializaci toto jméno dosadit do všech hodnot Odkud. Dobře se k tomu hodí např. -1.

6.3 Algoritmy pro obecné grafy

Algoritmy, které zde uvádíme, jsou zlepšenými variantami a speciálními případy základního schématu výpočtu vzdáleností 6.2.1.

6.3.1 Algoritmus s opakovanou kontrolou všech hran je asi nejjednodušším praktickým provedením základního schématu výpočtu vzdáleností 6.2.1.

1. [*Inicializace.*] $U(r) := 0$, pro $v \neq r$ položíme $U(v) := \infty$.

2. [*Zpracování celé množiny hran.*] Pro všechny hrany grafu provedeme v libovolném pořadí krok 2a. Po zpracování všech hran pokračujeme podle kroku 3.

   2a. [*Zpracování hrany e.*] Jestliže $U(\textsc{Pv}(e)) + a(e) < U(\textsc{Kv}(e))$, provedeme $U(\textsc{Kv}(e)) := U(\textsc{Pv}(e)) + a(e)$ a $\textsc{Odkud}(\textsc{Kv}(e)) := e$.

3. [*Test ukončení.*] Jestliže během provádění kroku 2 nedošlo k žádné změně hodnoty U, výpočet končí. V opačném případě pokračujeme znovu podle kroku 2.

Z věty 6.2.5 okamžitě plyne správnost tohoto algoritmu. Platí však více:

6.3.2 Věta. Při zpracování grafu bez záporných cyklů algoritmem 6.3.1 vždy po i-tém provedení kroku 2 (pro $i = 0, 1, 2, \ldots, n-1$) platí:

> Pro každý vrchol x je hodnota $U(x)$ menší nebo rovna délce nejkratší cesty z r do x, která má nejvýše i hran.

Důsledek: Příkaz 2 algoritmu 6.3.1 může být vykonán nejvýše n-krát, doba práce algoritmu tedy je $O(mn)$.

Důkaz: Pro $i = 0$ tvrzení triviálně platí. Předpokládejme, že tvrzení platí pro nějaké i. Dokážeme, že bude platit také po dalším provedení kroku 2, tj. pro $i + 1$.

Vezměme libovolný vrchol x, označme $U_{i+1}(x)$ délku cesty, která je nejkratší ze všech cest z r do x, které mají nejvýše $i + 1$ hran, a dále označme y předposlední vrchol v této cestě. Počáteční úsek této cesty z r do y má nejvýše i hran a jeho délka $U_i(y)$ je nejkratší ze všech cest z r do y, které mají nejvýše i hran. Podle indukčního předpokladu po i-tém vykonání kroku 2 bylo $U(y) \leq U_i(y)$, a proto po dalším provedení kroku 2, kdy musela být zpracována i hrana (y, x), musí platit $U(x) \leq U_{i+1}(x)$. □

6.3.3 Algoritmus s množinou podezřelých vrcholů. Základní schéma výpočtu vzdáleností 6.2.1 lze dále zlepšit tím, že zabráníme zbytečnému testování hran, o nichž předem víme, že trojúhelníkovou nerovnost (6.3) již splňují.

Jestliže nějaká hrana e trojúhelníkovou nerovnost (6.3) již splňovala, může k porušení této nerovnosti dojít pouze snížením hodnoty U v jejím počátečním vrcholu. Naopak, došlo-li ke snížení hodnoty $U(x)$, je nutno otestovat nerovnost (6.3) pro všechny hrany vycházející z vrcholu x, neboť pro tyto hrany by mohla být

platnost nerovnosti (6.3) porušena. Budeme tedy během výpočtu udržovat množinu M „podezřelých“ vrcholů, v nichž došlo ke snížení hodnoty U a které v množině M čekají na testování hran, které z nich vycházejí.

VSTUP: Graf G, ohodnocení hran $a : E(G) \to \mathbb{R}$ a výchozí vrchol r.
VÝSTUP: Pro každý vrchol v hodnoty $U(v)$, ODKUD(v).
POMOCNÉ PROMĚNNÉ: Jako pomocnou proměnnou použijeme množinu M, která bude mít tuto vlastnost:

(6.4) Jestliže PV$(e) \notin M$, pak hrana e splňuje nerovnost (6.3).

ALGORITMUS:

1. [*Inicializace.*] $U(r) := 0$, pro $v \neq r$ položíme $U(v) := \infty$ a dále $M := \{r\}$.
2. [*Výběr vrcholu.*] Je-li $M = \emptyset$, výpočet končí. Jestliže $M \neq \emptyset$, odebereme z množiny M některý vrchol a označíme jej x.
3. [*Zpracování hran vycházejících z vrcholu x.*] Pro každou hranu $e \in E^+(x)$ provedeme krok 3a a po zpracování všech hran pokračujeme podle kroku 2.

   3a. Položíme $y :=$ KV(e). Jestliže platí $U(x) + a(e) < U(y)$, pak provedeme $U(y) := U(x) + a(e)$, ODKUD$(y) := e$ a navíc, pokud $y \notin M$, vložíme vrchol y do množiny M.

POZNÁMKA: Volbu vrcholu z množiny podezřelých M lze v kroku 2 dělat v podstatě libovolným způsobem. Tento způsob nemá vliv na správnost algoritmu (viz věta 6.3.6), ovlivňuje však jeho rychlost. Uvedeme dva důležité případy:

6.3.4 Algoritmus s frontou podezřelých vrcholů volí v kroku 2 vrchol, který je v množině M nejdéle. S množinou M tedy pracujeme jako s frontou (viz 3.2.1), tj. jako se seznamem, z něhož odebíráme na opačném konci než přidáváme. Algoritmus přitom funguje podobně jako algoritmus s opakovanou kontrolou všech hran 6.3.1, pouze jsou vynechány některé zbytečné testy trojúhelníkové nerovnosti. Zrychlení výpočtu ve srovnání s algoritmem 6.3.1 se sice neprojeví na časovém odhadu pro nejhorší případ (ten zůstává $O(mn)$, viz věta 6.3.7), průměrná doba výpočtu je však zejména u řídkých grafů výrazně kratší.

Výpočetní experimenty ukazují, že přes svou jednoduchost je tento algoritmus vhodný k hledání nejkratších cest ve dvou situacích: jednak v řídkých grafech, tj. v grafech, kde počet hran m nepřesahuje nějaký nevelký násobek počtu vrcholů n, jednak v grafech, kde je velké procento hran se zápornou délkou. Za zvláštní zmínku stojí skutečnost, že dokonce i na grafech s nezápornými délkami hran, pokud je graf řídký, je tento algoritmus v průměru rychlejší i než často doporučovaný tzv. Dijkstrův algoritmus.

6.3.5 Modifikovaný Dijkstrův algoritmus volí v kroku 2 vrchol, který má nejnižší hodnotu U. Vztah k originálnímu Dijkstrovu algoritmu je vysvětlen dále v 6.5, str. 102.

Volba vrcholu x s nejnižší hodnotou $U(x)$ se odůvodňuje nadějí, že takto zvolená hodnota $U(x)$ se již v dalším výpočtu nezmění a že už tedy tento vrchol nebude znovu zařazen do množiny M. Spolehnout se na to však lze jen v grafech, kde všechny hrany mají nezáporné délky (viz 6.5.1). V obecných grafech je tento algoritmus také poměrně dobře použitelný, ale v ojedinělých případech se může stát, že týž vrchol bude do množiny M zařazen dokonce mnohokrát (až 2^{n-1}-krát). Časový odhad tohoto algoritmu je tedy exponenciální.

Výpočetní experimenty ukazují, že modifikovaný Dijkstrův algoritmus je vhodný k hledání nejkratších cest v hustých grafech, které mají jen malé (nebo nulové) procento hran se zápornou délkou. Pokud počet vrcholů přesahuje asi tak stovku, je navíc vhodné vybírat minimum v příkaze 2 trochu chytřeji než prostým prohlížením všech hodnot, jinak většinu času promarníme vybíráním minima. Obvykle se k tomu používá datová struktura nazývaná *halda* (anglicky heap), viz např. [AHU74, Kučera83, Töpfer95]).

6.3.6 Věta (správnost algoritmu s množinou podezřelých vrcholů). Jestliže graf neobsahuje cyklus se zápornou délkou, pak se algoritmus 6.3.3 zastaví a nalezne nejkratší cesty z vrcholu r do ostatních vrcholů grafu.

Poznámka: Toto platí nezávisle na způsobu volby vrcholu z množiny M.

Důkaz: Testy trojúhelníkové nerovnosti a odpovídající změny hodnot U jsou prováděny způsobem, který je v souladu se schématem 6.2.1. Podle věty 6.2.5 tedy dojde pouze ke konečně mnoha změnám hodnot U. Každý vrchol $v \neq r$ je do množiny M zařazen pouze vzápětí po změně jeho hodnoty $U(v)$. Proto také kroky 2 a 3 algoritmu mohou být vykonány nejvýše konečný počet krát, a algoritmus se tedy zastaví.

Dále, snadno lze ověřit, že podmínka (6.4) platí před každým vykonáním kroku 2 a po každém vykonání kroku 3. Až se tedy algoritmus zastaví (tj. v kroku 2 bude $M = \emptyset$), bude trojúhelníková nerovnost (6.3) platit pro všechny hrany grafu, a výpočet tedy bude podle věty 6.2.5 také správný. □

6.3.7 Věta (časový odhad algoritmu s frontou podezřelých vrcholů). Jestliže graf neobsahuje cyklus se zápornou délkou, pak během výpočtu algoritmem 6.3.4 (s frontou podezřelých vrcholů) nebude žádný vrchol zařazen do množiny M více než $(n-1)$-krát, kde n je počet vrcholů grafu.

Důsledek: Algoritmus skončí práci v čase $O(mn)$.

Důkaz: Pro účely důkazu si algoritmus upravme podle 6.2.9 tak, aby pro každý vrchol v udržoval počet hran $P(v)$ v cestě, jejíž délka je $U(v)$. Tato úprava nijak neovlivní výpočet, použijeme ji pouze k úvahám v důkaze.

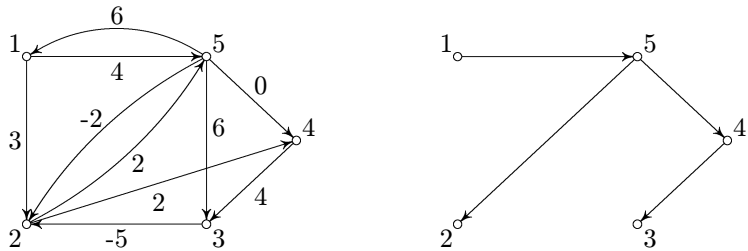
Připomeňme, že s množinou M zacházíme jako s frontou, tj. jako se seznamem, z něhož odebíráme na opačném konci než přidáváme. Při zpracování hran, které vycházejí z vrcholu x, jsou do množiny M vkládány vrcholy s hodnotou $P(x)+1$. Z toho plyne, že hodnoty P vrcholů, které jsou v nějakém okamžiku v množině M, se mohou lišit nejvýše o jedničku, a také to, že hodnoty $P(x)$ zpracovávaných vrcholů tvoří neklesající posloupnost (viz též lemma 3.2.3, str. 53).

Žádný vrchol v se nemůže dostat do množiny M dvakrát se stejnou hodnotou $P(v)$. Jakmile je totiž vrchol v vyjmut z množiny M s hodnotou $P(v) = k$, pak od tohoto okamžiku jsou do M zařazovány vrcholy pouze s hodnotami $k+1$ a vyššími.

Cesta může mít nejvýše $n-1$ hran, proto žádný vrchol nemůže mít hodnotu P vyšší než $n-1$, a tedy nemůže být do množiny M zařazen více než $(n-1)$-krát. □

Důkaz důsledku: Příkaz 3a zpracuje každou hranu nejvýše $(n-1)$-krát, neboť její počáteční vrchol bude vyjmut z množiny M nejvýše $(n-1)$-krát. □

6.3.8 Příklad. Použijeme modifikovaný Dijkstrův algoritmus 6.3.5 k vyhledání nejkratších cest z vrcholu 1 v grafu na obr. 6.3. Z množiny M tedy budeme vybírat vždy vrchol s nejnižší hodnotou U.



Obrázek 6.3: Graf k příkladu 6.3.8 a výsledný strom nejkratších cest.

Průběh výpočtu bude tento:

		$M = \{1\}$
$x = 1:$		$M = \emptyset$
	$U(2) = 3,$	$M = \{2\}$
	$U(5) = 4,$	$M = \{2, 5\}$
$x = 2:$		$M = \{5\}$
	$U(4) = 5,$	$M = \{4, 5\}$
	$U(5)$ beze změny	$M = \{4, 5\}$
$x = 5:$		$M = \{4\}$
	$U(2) = 2,$	$M = \{2, 4\}$
	$U(3) = 10,$	$M = \{2, 3, 4\}$
	$U(4) = 4,$	$M = \{2, 3, 4\}$
$x = 2:$		$M = \{3, 4\}$
	$U(4)$ beze změny	$M = \{3, 4\}$
	$U(5)$ beze změny	$M = \{3, 4\}$
$x = 4:$		$M = \{3\}$
	$U(3) = 8,$	$M = \{3\}$
$x = 3:$		$M = \emptyset$
	$U(2)$ beze změny	$M = \emptyset$
		$M = \emptyset$

Všimněte si, že vrchol 2 byl do množiny M zařazen dvakrát. Sledujte také, jak se v průběhu výpočtu mění kořenový strom nejkratších cest.

Výsledek výpočtu lze shrnout do této tabulky:

vrchol x	1	2	3	4	5
vzdálenost $U(x)$	0	2	8	4	4
ODKUD(x)	–	(5,2)	(4,3)	(5,4)	(1,5)

Počítáme-li ručně (tj. ne na počítači), je vhodné do takové tabulky zapisovat mezivýsledky a změněné hodnoty škrtat.

6.3.9 Cvičení. Najděte nejkratší cesty z vrcholu 1 do všech dostupných vrcholů a nakreslete kořenové stromy nejkratších cest v těchto grafech:

a)

Pv	Kv	a
1	2	16
1	3	6
1	7	36
2	5	10
3	2	8
3	4	4
4	3	2

Pv	Kv	a
4	5	12
4	7	21
5	3	2
5	6	6
5	9	8
6	7	3
6	10	7

Pv	Kv	a
7	8	11
7	10	6
8	5	4
8	10	2
9	8	14
10	9	0

b)

Pv	Kv	a
1	2	2
1	3	6
1	5	5
2	3	0
2	4	4
2	5	2

Pv	Kv	a
3	5	-1
3	6	-2
4	3	1
4	5	-2
4	7	7
5	6	-1

Pv	Kv	a
5	7	8
5	9	6
6	8	4
7	9	0
8	7	6
8	9	2

6.4 Algoritmy pro acyklické grafy

Pro acyklické grafy je výpočet nejkratších cest velmi snadný. Stačí v základním schématu výpočtu vzdáleností 6.2.1 probírat hrany v topologickém uspořádání:

6.4.1 Věta. Je-li graf acyklický a jsou-li jeho hrany algoritmem 6.2.1 zpracovávány v topologickém uspořádání, pak po zpracování všech hran bude trojúhelníková nerovnost (6.3) platit pro všechny hrany.

DŮKAZ: Nerovnost (6.3) platí pro každou hranu e těsně po jejím zpracování. Kdyby někdy později byla platnost nerovnosti porušena, bylo by to způsobeno zpracováním některé hrany, která končí v PV(e). To by však bylo v rozporu s topologickým uspořádáním hran. □

DŮSLEDEK: V acyklickém grafu lze nejkratší cesty z daného vrcholu najít v čase $O(n+m)$.

Důsledek: Při použití algoritmu s množinou podezřelých vrcholů 6.3.3 dosáhneme stejně rychlého průběhu výpočtu, vybíráme-li vrcholy v kroku 2 v topologickém uspořádání. Použití topologického uspořádání hran v základním schématu 6.2.1 je však jednodušší.

6.4.2 Výpočet nejkratších cest zároveň s topologickým uspořádáním. Výpočet nejkratších cest v acyklickém grafu lze s výhodou zkombinovat s hledáním topologického uspořádání algoritmem 5.2.7, str. 77. V algoritmu 5.2.7 provedeme navíc toto:

1. V kroku 1 provedeme navíc $U(r) := 0$ a $U(v) := \infty$ pro všechny vrcholy $v \neq r$.

2. V kroku 4a pro každou zpracovávanou hranu e provedeme navíc: Jestliže $U(\text{Kv}(e)) > U(\text{Pv}(e)) + a(e)$, pak položíme $U(\text{Kv}(e)) := U(\text{Pv}(e)) + a(e)$ a $\text{Odkud}(\text{Kv}(e)) := e$.

Důkaz správnosti takto upraveného algoritmu vyplývá z vět 6.2.7 a 6.4.1.

6.4.3 Nejkratší cesty do konkrétního vrcholu. Výše popsané algoritmy pro hledání nejkratších cest v acyklických grafech lze ukončit, jsou-li zpracovány všechny hrany, které končí v cílovém vrcholu. To může výpočet značně urychlit. Je zřejmé, že některá topologická uspořádání jsou z tohoto hlediska výhodnější než jiná.

6.4.4 Cvičení. Řešte znovu úlohu 6.3.9b a využijte při tom informaci, že tento graf je acyklický. Použijte oba postupy 6.4.1 i 6.4.2.

6.4.5 Hledání nejdelších cest v acyklickém grafu. Pro hledání nejdelších cest lze využít popsané algoritmy po změně znamének u délek všech hran. Tím převedeme hledání maxima na hledání minima. Délky hran sice budou záporné, ale cyklus o záporné délce v acyklickém grafu být nemůže.

Často bývá tato úloha řešena speciálním algoritmem, který získáme nepatrně odlišnou úpravou algoritmu 5.2.7 pro hledání topologického uspořádání (viz též kapitola 7):

1. V kroku 1 položíme $U(r) := 0$ a $U(v) := -\infty$ pro všechny vrcholy $v \neq r$.

2. V kroku 4a pro každou zpracovávanou hranu e provedeme navíc: Jestliže $U(\text{Kv}(e)) < U(\text{Pv}(e)) + a(e)$, pak položíme $U(\text{Kv}(e)) := U(\text{Pv}(e)) + a(e)$ a $\text{Odkud}(\text{Kv}(e)) := e$.

6.4.6 Výpočet síťového grafu (viz 2.3.2, str. 44) se obvykle skládá ze čtyř fází. V prvé fázi najdeme topologické uspořádání vrcholů nebo hran.

Ve druhé fázi počítáme časy $T_1(v)$, tj. nejdřívější možné začátky činností. K tomu lze použít vzorec (2.1) na všechny vrcholy v topologickém uspořádání.

Ve třetí fázi počítáme časy $T_2(v)$, tj. nejpozdější možné termíny, při kterých lze stihnout celý projekt v nejkratším možném čase $T_1(K)$. K tomu lze použít vzorec

(2.2) na všechny vrcholy v obráceném topologickém uspořádání, tj. vrchol P bude zpracován jako poslední.

Výpočet hodnot T_1 a T_2 lze alternativně provést podobně jako v 6.4.5, přičemž pro výpočet T_1 používáme hrany v topologickém pořadí a pro výpočet T_2 používáme hrany v obráceném topologickém pořadí.

Poslední fáze spočívá ve výpočtu rezerv činností. Celý výpočet lze provést v čase $O(m+n)$.

6.5 Nezáporné délky hran

Případ, kdy všechny délky hran jsou nezáporné, je bezesporu v praxi nejběžnější. Uvádíme zde dva algoritmy pro jeho řešení.

Prvý z nich jsme již uvedli – je jím obecně použitelný modifikovaný Dijkstrův algoritmus 6.3.5. Druhým algoritmem je tzv. *Dijkstrův algoritmus* 6.5.2.

Oba algoritmy, Dijkstrův i modifikovaný Dijkstrův, se při práci na grafech s nezápornými délkami hran liší jen nepatrně, během výpočtu dělají skoro totéž a pro oba platí stejný časový odhad $O(n^2+m)$. Pomocí programátorských triků pro výběr minima (použitím datové struktury halda, anglicky heap, viz např. [AHU74, Kučera83, Töpfer95]), lze v obou případech časový odhad zlepšit na $O((m+n)\log n)$.

Výpočetní experimenty ukazují, že tyto triky je vhodné používat u grafů s více než asi tak stovkou vrcholů. Dále poznamenejme, že oba tyto algoritmy jsou vhodné zejména pro husté grafy (které mají „hodně“ hran). Viz též poznámky o době výpočtu v 6.3.4 a 6.3.5.

6.5.1 Věta. Jsou-li v grafu délky všech hran nezáporné, pak při výpočtu modifikovaným Dijkstrovým algoritmem 6.3.5 pro každý vrchol x platí, že ve chvíli jeho vyjmutí z množiny M je již hodnota $U(x)$ definitivní, tj. $U(x)=u(r,x)$.

Důsledek: Každý vrchol bude do množiny M zařazen nejvýše jednou, tedy i každá hrana bude příkazem 3a zpracována nejvýše jednou. Dále, n-krát opakovaný výběr minima z nejvýše n hodnot vyžaduje čas $O(n^2)$. Časový odhad pro celý algoritmus tedy je $O(n^2+m)$.

Důkaz: Označme D množinu všech vrcholů x, které již byly vyjmuty z množiny M. Zřejmě hned po prvním vykonání kroku 2 bude $D=\{r\}$ a tvrzení věty platí.

Předpokládejme, že z množiny M vybíráme vrchol x a že pro množinu D tvrzení platí. Dokážeme sporem, že tvrzení věty pak platí i pro vrchol x.

Zkusme tedy pro spor předpokládat, že nejkratší cesta do x má délku $u(r,x)<$ $<U(x)$. Vezměme nejkratší cestu C z vrcholu r do vrcholu x. Označme w první vrchol této cesty, který neleží v D, a dále označme v vrchol, který v cestě C vrcholu w bezprostředně předchází, a který tedy v D leží. Všechny hrany z množiny $E^+(v)$ již v této chvíli byly algoritmem zpracovány. Navíc, nejkratší cesta z r do w je částí cesty C. Jistě tedy platí $U(w)=u(r,w)$. Proto také $w\neq x$, neboť $U(x)>u(r,x)$. Poněvadž nejkratší cesta C prochází přes w, platí $u(r,w)+u(w,x)=u(r,x)$, a tedy

$U(w)+u(w,x) = u(r,x) < U(x)$ a odtud $U(w) < U(x)$. To je však spor s pravidlem pro výběr vrcholu x (měli jsme vybrat vrchol w). Nejkratší cesta do x tedy musela mít délku $u(r,x) = U(x)$. □

6.5.2 Dijkstrův algoritmus. V literatuře bývá pro výpočet nejkratších cest v grafu s nezápornými délkami hran doporučován tzv. Dijkstrův algoritmus. Je to algoritmus téměř totožný s algoritmem, který jsme nazvali modifikovaný Dijkstrův. Odlišnost spočívá v tom, že namísto množiny M se v něm udržuje množina vrcholů D, která obsahuje vrcholy, jejichž hodnota U je již definitivní a vybírá se vrchol s nejnižší hodnotou U z množiny $V(G) \setminus D$.

Na grafech s nezápornými délkami hran fungují oba algoritmy naprosto shodně (až na manipulace s množinami M a D). Rozdíl se projeví, jsou-li délky některých hran záporné. Modifikovaný Dijkstrův algoritmus pracuje správně i se zápornými délkami, ale v tomto případě pro něj platí horší časový odhad. Dijkstrův algoritmus může dát chybné výsledky, protože nemá žádnou možnost zpracovat některý vrchol více než jednou. (To ovšem není žádná ostuda, on na to prostě není určen.)

Vzhledem k popularitě Dijkstrova algoritmu uvedeme jej v plném znění:

VSTUP: Graf G, výchozí vrchol r a ohodnocení hran $a : E(G) \to \mathbb{R}$ takové, že pro všechny hrany platí $a(e) \geq 0$.
VÝSTUP: Pro každý vrchol v hodnoty $U(v)$, ODKUD(v).
POMOCNÉ PROMĚNNÉ: Množina vrcholů D taková, že pro všechny vrcholy $v \in D$ bude hodnota $U(v)$ rovna vzdálenosti $u(r,v)$.

ALGORITMUS:

1. [*Inicializace.*] $U(r) := 0$, pro $v \neq r$ položíme $U(v) := \infty$ a dále $D := \emptyset$.

2. [*Výběr vrcholu.*] Jestliže pro všechny vrcholy $v \in (V \setminus D)$ platí $U(v) = \infty$, výpočet končí. Jinak z množiny $V \setminus D$ vybereme vrchol x, který má nejnižší hodnotu $U(x)$, zařadíme jej do množiny D a pokračujeme krokem 3.

3. [*Zpracování hran vycházejících z vrcholu x.*] Pro každou hranu $e \in E^+(x)$ provedeme krok 3a a po zpracování všech hran pokračujeme podle kroku 2.

   3a. Položíme $y :=$ KV(e). Jestliže platí $U(x) + a(e) < U(y)$, pak provedeme $U(y) := U(x) + a(e)$ a ODKUD$(y) := e$.

Dodejme, že roli množiny M podezřelých vrcholů z modifikovaného Dijkstrova algoritmu 6.3.5 zde hraje množina $(V \setminus D) \cap \{v \mid U(v) < \infty\}$.

6.5.3 Cvičení. Najděte graf, který neobsahuje cyklus se zápornou délkou, obsahuje hranu záporné délky a Dijkstrův algoritmus v něm dá chybné výsledky.

6.5.4 Nejkratší cesta do konkrétního vrcholu. Pokud nás zajímá nejkratší cesta pouze do jednoho cílového vrcholu, pak oba algoritmy (Dijkstrův i modifikovaný Dijkstrův) lze ukončit, jakmile cílový vrchol c odebereme z množiny M (nebo u Dijkstrova algoritmu zařadíme do množiny D).

Pokud máme štěstí a cílový vrchol vypočteme mezi prvními, tj. pokud jen málo vrcholů grafu leží v menší vzdálenosti než cílový vrchol, může to výpočet někdy i značně urychlit. Ke zlepšení časového odhadu to však nevede, protože se může stát, že cílový vrchol bude nejvzdálenější, a že tedy bude vypočten až jako poslední.

6.6 Využití dodatečné informace – algoritmus A*

Ve velmi rozsáhlých grafech mohou být výše uvedené postupy příliš pomalé. Námět na zrychlení výpočtu je založen na víře, že při hledání nejkratší cesty z Prahy do Brna je jistě zbytečné zkoumat cestu přes Plzeň.

Pokud bychom dovedli spolehlivě zjistit, že nejkratší cesta z vrcholu r do vrcholu c zcela jistě nepovede přes vrchol v, mohli bychom zpracování vrcholu v a hran s ním incidentních přeskočit. Pracovali bychom s menším grafem, a tedy rychleji. Kritérium pro vyloučení některých vrcholů lze chápat jako dodatečnou informaci, která může urychlit výpočet. Potíž je v tom, že takové kritérium musí být „ušito na míru“ konkrétním podmínkám úlohy (původně zpravidla negrafové), a také v tom, že ověření správnosti kritéria (tj. že nevyloučí příliš mnoho vrcholů) nemusí být nijak snadné.

Uvedeme jednoduchou modifikaci Dijkstrova algoritmu, která používá dodatečnou informaci ve formě ohodnocení vrcholů $h(v)$.

6.6.1 Heuristická funkce. Jako dodatečnou informaci použijeme ohodnocení vrcholů $h : V \to \mathbb{R}$, které splňuje pro každou hranu (x, y) podmínku

$$a(x, y) \geq h(x) - h(y) \; . \tag{6.5}$$

Splnění této podmínky lze pro všechny hrany grafu snadno ověřit.

Aby hodnoty h měly příznivý vliv na rychlost výpočtu, je třeba vzít do hry cílový vrchol c a hodnoty h pro něj „ušít na míru“. Doporučuje se, aby hodnoty $h(v)$ byly dolním odhadem vzdálenosti z vrcholu v do cílového vrcholu c, tj. aby pro každý vrchol v platilo $h(v) \leq u(v, c)$. Čím bude odhad přesnější, tj. čím bude hodnota $h(v)$ blíže skutečné vzdálenosti $u(v, c)$, tím bude výpočet rychlejší.

Jako příklad hodnot h vhodných pro hledání nejkratších cest v silniční síti se uvádí vzdálenost do cílového místa měřená vzdušnou čarou. Podmínky (6.5) v tomto případě není třeba ověřovat, protože vyplývají z geometrických vlastností prostoru.

6.6.2 Algoritmus A* (Dijkstrův algoritmus s heuristikou) se od klasického Dijkstrova algoritmu 6.5.2 liší v jediné věci: při výběru vrcholu, který má být v kroku 2 zařazen do množiny D, volíme vždy vrchol x, který má nejnižší hodnotu $U(x)+h(x)$. Stejnou úpravu lze samozřejmě udělat i pro modifikovaný Dijkstrův algoritmus 6.3.5.

Pro důkaz správnosti upravených algoritmů budeme potřebovat toto lemma:

6.6.3 Lemma. Platí-li nerovnost (6.5) pro všechny hrany grafu, pak pro každé dva vrcholy x, y platí také nerovnost $u(x, y) \geq h(x) - h(y)$.

Důkaz snadno získáme sečtením nerovností (6.5) pro všechny hrany nejkratší cesty z x do y. □

6.6.4 Věta (správnost algoritmu A*). Předpokládejme, že všechny hrany grafu mají nezáporné délky a že ohodnocení vrcholů h splňuje pro všechny hrany podmínku (6.5). Pak pro všechny vrcholy x zařazené algoritmem 6.6.2 do množiny D platí $U(x) = u(r, x)$. (Jinak řečeno, pak algoritmus 6.6.2 počítá správně.)

Důkaz je podobný důkazu klasického Dijkstrova algoritmu, viz také 6.5.1.

Na začátku je $D = \{r\}$ a tvrzení věty platí. Předpokládejme, že tvrzení platí pro množinu D dosud vybraných vrcholů a že nyní vybíráme vrchol x z množiny $M = (V \setminus D) \cap \{v \mid U(v) < \infty\}$. Ukážeme, že tvrzení věty platí i pro vrchol x.

Zkusme tedy pro spor předpokládat, že nejkratší cesta do vrcholu x má délku $u(r, x) < U(x)$. Označme tuto cestu C, dále označme w první vrchol této cesty, který neleží v D, a konečně označme v vrchol, který v cestě C vrcholu w bezprostředně předchází, a který tedy v D leží. Všechny hrany z množiny $E^+(v)$ již v této chvíli byly algoritmem zpracovány. Navíc, nejkratší cesta z r do w je částí cesty C. Jistě tedy platí $U(w) = u(r, w)$. Proto také $w \neq x$, neboť $U(x) > u(r, x)$. Poněvadž nejkratší cesta C prochází přes w, platí $u(r, w) + u(w, x) = u(r, x)$, a tedy $U(w) + u(w, x) = u(r, x) < U(x)$. (Až sem to bylo stejné jako v důkaze pro klasický Dijkstrův algoritmus.)

Z lemmatu 6.6.3 plyne $h(w) \leq u(w, x) + h(x)$ a odtud sečtením s předchozí nerovností dostaneme $U(w) + u(w, x) + h(w) < U(x) + u(w, x) + h(x)$, a tedy $U(w) + h(w) < U(x) + h(x)$. To je však spor s pravidlem pro výběr vrcholu x. Nejkratší cesta do x tedy musela mít délku $u(r, x) = U(x)$. □

6.6.5 Poznámky. Rychlost algoritmu A* je založena na tom, že výpočet ukončíme ihned po nalezení definitivní vzdálenosti do cílového vrcholu a také na tom, že vysoká hodnota h v nějakém vrcholu způsobí, že tento vrchol nebude vybrán, a tím ušetříme zpracování hran, které z něj vycházejí. V krajním případě však i tento algoritmus bude muset zpracovat všechny vrcholy grafu.

Podmínku (6.5) splňuje triviálním způsobem i nulové ohodnocení. Klasický Dijkstrův algoritmus 6.5.2 lze tedy pokládat za speciální případ algoritmu 6.6.2.

6.7 Výpočet matice vzdáleností

Samozřejmě, k výpočtu matice vzdáleností můžeme použít algoritmy popsané v předchozích oddílech 6.2 až 6.5. Tyto algoritmy poskytují vždy jeden řádek hledané matice U. Musíme tedy výpočet opakovat pro každou volbu výchozího vrcholu r, tj. v grafu s n vrcholy celkem n-krát. Oddělené provedení těchto výpočtů ovšem může být, a také obecně je, méně efektivní než speciální algoritmus poskytující celou matici najednou, neboť informace z jednoho výpočtu by měly být využity v ostatních výpočtech. Proto se dále zaměříme na speciální algoritmy poskytující celou matici U a pouze upozorníme, že pro řídké grafy je výhodnější počítat každý řádek ma-

tice U odděleně. Také je třeba zvážit, že celá matice vzdáleností může být větší než dostupná paměť počítače a že mnohdy nepotřebujeme celou matici, ale jen její část.

Prvý z obou níže uvedených postupů je zajímavý spíše teoreticky (srovnejte s kapitolou 7), druhý postup, Floydův algoritmus, je nejen rychlejší, ale navíc i jednoduchý a elegantní. Oba tyto postupy jsou zobecněny v kapitole 7.

6.7.1 Upravené násobení matic vychází z matice délek hran A a postupně počítá posloupnost matic $U_1, U_2, \ldots, U_{n-1}$, kde n je počet vrcholů. Každá matice U_p z této posloupnosti má tuto vlastnost:

$$u_p(i,j) \text{ je délka nejkratší cesty z } i \text{ do } j, \text{ která má nejvýše } p \text{ hran.}$$

V matici U_1 jsou délky nejkratších cest, které mají nejvýše jednu hranu, tedy $U_1 = A$. Poněvadž žádná cesta nemůže obsahovat více než $n-1$ hran, je $U_{n-1} = U$ hledaná matice vzdáleností.

Známe-li matici U_{p-1}, můžeme snadno vypočítat matici U_p takto:

$$u_p(i,j) = \min_k \{u_{p-1}(i,k) + a(k,j)\} \ .$$

Nejkratší cestu z i do j, která má nejvýše p hran, totiž můžeme dostat tak, že vezmeme nejkratší cestu z i do nějakého vrcholu k, která má nejvýše $p-1$ hran, k této cestě přidáme nejvýše jednu další hranu z k do j (pokud bylo $k = j$, nepřidáváme nic, proto máme v matici A na diagonále nuly) a vezmeme nejkratší z takto získaných cest.

Způsob výpočtu svou formou připomíná násobení matic: hodnota na místě (i,j) v matici U_p závisí na i-tém řádku matice U_{p-1} a na j-tém sloupci matice A. Podobnost je dokonce silnější – stačí v běžném postupu násobení matic nahradit operaci násobení sčítáním a operaci sčítání nahradit výběrem minima. Označíme-li tuto novou operaci s maticemi znakem $\otimes$, platí $U_p = U_{p-1} \otimes A$.

Přímý výpočet matice $U_{n-1} = U$ právě naznačeným postupem je sice jednoduchý na pochopení, ale zbytečně pracný: jedno upravené násobení matic $\otimes$ vyžaduje zhruba n^3 operací sčítání a výběru minima, celková spotřeba času tedy je $O(n^4)$.

Lze jej snadno zrychlit na $O(n^3 \log_2 n)$ tím, že nebudeme počítat všechny matice U_p, ale jen ty, kde p je mocninou dvojky, dokud nedosáhneme $p \geq n-1$. Lze totiž snadno dokázat, že násobení matic $\otimes$ je asociativní (tj. nezáleží na uzávorkování), a tedy $U_{2p} = U_p \otimes U_p$, a navíc, pokud v grafu není cyklus se zápornou délkou, jsou matice U_p pro $p \geq n-1$ všechny stejné a rovné hledané matici U.

Tento postup vyžaduje pouze $\log_2 n$ operací upraveného násobení matic $\otimes$, celková spotřeba času je tedy $O(n^3 \log_2 n)$. Dále však uvedeme rychlejší a jednodušší Floydův algoritmus 6.7.3, který vystačí s časem $O(n^3)$.

Poznamenejme, že v grafu se záporným cyklem by byly všechny matice U_p různé a obsahovaly by délky sledů, které by nemusely být cestami.

6.7.2 Příklad. Vypočtěme matici vzdáleností pro graf z obr. 6.2 pomocí upraveného násobení matic. Budeme počítat pouze matice U_1, U_2, U_4, U_8. Připomeňme, že $A = U_1$ a $U_8 = U$.

$$U_1 = \begin{pmatrix} 0, & -5, & \infty, & \infty, & 3 \\ \infty, & 0, & 5, & \infty, & 2 \\ \infty, & \infty, & 0, & 4, & \infty \\ \infty, & \infty, & \infty, & 0, & -5 \\ \infty, & -2, & 2, & \infty, & 0 \end{pmatrix}, \qquad U_2 = \begin{pmatrix} 0, & -5, & 0, & \infty, & -3 \\ \infty, & 0, & 4, & 9, & 2 \\ \infty, & \infty, & 0, & 4, & -1 \\ \infty, & -7, & -3, & 0, & -5 \\ \infty, & -2, & 2, & 6, & 0 \end{pmatrix},$$

$$U_4 = U_8 = \begin{pmatrix} 0, & -5, & -1, & 3, & -3 \\ \infty, & 0, & 4, & 8, & 2 \\ \infty, & -3, & 0, & 4, & -1 \\ \infty, & -7, & -3, & 0, & -5 \\ \infty, & -2, & 2, & 6, & 0 \end{pmatrix}.$$

6.7.3 Floydův algoritmus vychází z matice délek hran A a postupně počítá posloupnost matic U_0, U_1, U_2, …, U_n, kde n je počet vrcholů. Každá matice U_k z této posloupnosti má tuto vlastnost:

$u_k(i,j)$ = délka nejkratší cesty z i do j takové, že kromě vrcholů i, j cesta prochází pouze přes vrcholy z množiny $\{1, 2, \ldots, k\}$.

V matici U_0 jsou délky nejkratších cest, které nesmí procházet přes žádné vrcholy (kromě počátečního a koncového), je tedy $U_0 = A$. Naopak v matici U_n jsou délky nejkratších cest, které již nejsou nijak omezeny, proto $U_n = U$ je hledaná matice vzdáleností.

Známe-li matici U_{k-1}, můžeme snadno vypočítat matici U_k takto:

$$u_k(i,j) = \min\{u_{k-1}(i,j),\, u_{k-1}(i,k) + u_{k-1}(k,j)\}\,. \tag{6.6}$$

Nejkratší cesta z i do j přes vrcholy $\{1, 2, \ldots, k\}$ totiž buď neprochází přes vrchol k a pak má délku $u_{k-1}(i,j)$, nebo prochází přes vrchol k a skládá se ze dvou částí, z i do k a z k do j, přičemž žádná z těchto částí již přes vrchol k neprochází. Součet jejich délek tedy je $u_{k-1}(i,k) + u_{k-1}(k,j)$.

Postup výpočtu podle vzorce (6.6) je patrný z obrázku 6.4 na následující straně.

Pro ruční výpočet je tento vzorec postačujícím návodem, každou z matic U_0, U_1, U_2, …, U_n je výhodné zapisovat zvlášť.

Při výpočtu na počítači by to však bylo zbytečným plýtváním pamětí. Ukazuje se, že výpočet lze provést „na místě“, tj. v jediné matici M, která je zpočátku rovna matici délek hran A, postupně je transformována na $U_1, U_2, \ldots, U_{n-1}$ a nakonec obsahuje matici vzdáleností U. Při transformaci matice U_{k-1} na matici U_k se totiž hodnoty v k-tém řádku a v k-tém sloupci nemění a transformace ostatních prvků jsou navzájem nezávislé. Celý algoritmus lze tedy zapsat takto:

VSTUP: Matice M obsahující délky hran.
VÝSTUP: Je tvořen touž maticí M, která obsahuje vzdálenosti.

ALGORITMUS:
pro $k := 1, 2, \ldots, n$ proveď:
  pro $i := 1, 2, \ldots, n$ proveď:
    pro $j := 1, 2, \ldots, n$ proveď:
      když $M(i,j) > M(i,k) + M(k,j)$,
        pak proveď $M(i,j) := M(i,k) + M(k,j)$.

ČASOVÉ NÁROKY: $O(n^3)$.

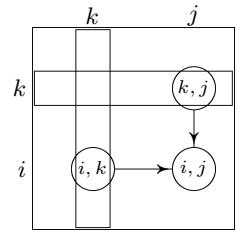
6.7.4 Příklad. Vypočtěme Floydovým algoritmem matici vzdáleností pro graf z obr. 6.2, str. 92. Připomeňme, že $U_0 = A$ a $U_5 = U$.

$$U_0 = \left(\begin{array}{c|cccc} 0, & -5, & \infty, & \infty, & 3 \\ \hline \infty, & 0, & 5, & \infty, & 2 \\ \infty, & \infty, & 0, & 4, & \infty \\ \infty, & \infty, & \infty, & 0, & -5 \\ \infty, & -2, & 2, & \infty, & 0 \end{array}\right), \quad U_1 = \left(\begin{array}{c|cccc} 0, & -5, & \infty, & \infty, & 3 \\ \hline \infty, & 0, & 5, & \infty, & 2 \\ \infty, & \infty, & 0, & 4, & \infty \\ \infty, & \infty, & \infty, & 0, & -5 \\ \infty, & -2, & 2, & \infty, & 0 \end{array}\right),$$

$$U_2 = \left(\begin{array}{cc|c|cc} 0, & -5, & 0, & \infty, & -3 \\ \infty, & 0, & 5, & \infty, & 2 \\ \hline \infty, & \infty, & 0, & 4, & \infty \\ \hline \infty, & \infty, & \infty, & 0, & -5 \\ \infty, & -2, & 2, & \infty, & 0 \end{array}\right), \quad U_3 = \left(\begin{array}{cc|c|c|c} 0, & -5, & 0, & 4, & -3 \\ \infty, & 0, & 5, & 9, & 2 \\ \infty, & \infty, & 0, & 4, & \infty \\ \hline \infty, & \infty, & \infty, & 0, & -5 \\ \infty, & -2, & 2, & 6, & 0 \end{array}\right),$$

$$U_4 = \left(\begin{array}{cccc|c} 0, & -5, & 0, & 4, & -3 \\ \infty, & 0, & 5, & 9, & 2 \\ \infty, & \infty, & 0, & 4, & -1 \\ \infty, & \infty, & \infty, & 0, & -5 \\ \hline \infty, & -2, & 2, & 6, & 0 \end{array}\right), \quad U_5 = \left(\begin{array}{ccccc} 0, & -5, & -1, & 3, & -3 \\ \infty, & 0, & 4, & 8, & 2 \\ \infty, & -3, & 0, & 4, & -1 \\ \infty, & -7, & -3, & 0, & -5 \\ \infty, & -2, & 2, & 6, & 0 \end{array}\right).$$

Zvýrazněny jsou vždy k-tý sloupec a k-tý řádek, který je použit k výpočtu následující matice U_k.



Obrázek 6.4: Schéma výpočtu Floydova algoritmu.

6.7.5 Cvičení. Vypočtěte Floydovým algoritmem matici vzdáleností pro graf z obr. 6.3, str. 99.

6.7.6 Cvičení. Je možné upravit Floydův algoritmus tak, aby navíc vytvářel matici Q, jejíž libovolný prvek $Q(i, j)$ by byl jménem vrcholu, který v nejkratší cestě z i do j těsně následuje za vrcholem i?

Motivace: Mnohé automapy obsahují matici vzdáleností vybraných měst, často však není jasné, kudy nejkratší cesty vedou. Z matice Q by mělo být okamžitě patrné, zda např. z Jindřichova Hradce do Prahy vede nejkratší cesta přes Tábor, nebo přes Pelhřimov. Další použití je při směrování paketů v počítačových sítích.

6.8 Hledání cyklů se zápornou délkou

Algoritmy uvedené v 6.2 až 6.7 pro hledání nejkratších cest pracují správně pouze za předpokladu, že graf neobsahuje cyklus se zápornou délkou. Nyní uvedeme, jak

lze platnost tohoto předpokladu ověřit. Někdy je dokonce třeba umět cyklus se zápornou délkou najít. Jako příklad lze uvést úlohu 2.2.8, str. 43, o nákladním parníku a také algoritmus 8.6.3, str. 154, pro hledání nejlevnějšího toku v síti.

Uvedeme dva způsoby zjišťování cyklů se zápornou délkou. Prvý z nich je založen na Floydově algoritmu 6.7.3 a na následující větě 6.8.1, druhý je odvozen z algoritmů oddílu 6.2 pro hledání cest z jednoho výchozího vrcholu.

6.8.1 Věta. Graf obsahuje cyklus o záporné délce právě tehdy, když matice M vypočtená Floydovým algoritmem obsahuje na diagonále zápornou hodnotu $M(i,i) < 0$.

Důkaz: Pokud graf obsahuje cyklus se zápornou délkou, označme k vrchol, který z tohoto cyklu má nejvyšší číslo, a i některý další vrchol tohoto cyklu. Platí $U(i,k) + U(k,i) < 0$, tedy $U_{k-1}(i,k) + U_{k-1}(k,i) < 0$, a proto $U_k(i,i) < 0$. Tato hodnota již nemůže stoupnout, proto bude záporná i ve výsledné matici.

Je-li na diagonále matice M záporná hodnota, označme k index matice, ve které k tomu poprvé došlo, tj. kde $U_k(i,i) < 0$. V předchozí matici U_{k-1} tedy platilo $U_{k-1}(i,k) + U_{k-1}(k,i) < 0$, cesty z vrcholu i do k a zpět tedy dohromady tvoří uzavřený sled o záporné délce. Ten je buď sám cyklem o záporné délce, anebo se skládá z cyklů, z nichž alespoň jeden má zápornou délku. □

Poznámka: Test, zda se na diagonále matice M vyskytuje záporný prvek, lze dělat již během výpočtu Floydova algoritmu. Časový odhad $O(n^3)$ se tím nezhorší.

Zbývající část tohoto oddílu se zabývá hledáním cyklů se zápornou délkou metodami, které jsou odvozeny od základního schématu výpočtu vzdáleností 6.2.1, str. 91.

6.8.2 Věta. Mějme graf, který obsahuje cyklus o záporné délce takový, že vrcholy tohoto cyklu jsou orientovaně dostupné z výchozího vrcholu r. Pak výpočet podle základního schématu výpočtu vzdáleností 6.2.1 nikdy neskončí. Dále pak platí, že během (nekonečného) výpočtu je $U(x)$ vždy délkou nějakého sledu (tj. ne nutně cesty) z vrcholu r do vrcholu x. Jsou-li navíc udržovány hodnoty P podle 6.2.9, pak hodnoty $P(x)$ pro některý vrchol x rostou do nekonečna.

Důkaz: Trojúhelníková nerovnost (6.3), str. 91 nemůže platit pro všechny hrany cyklu, který má zápornou délku. Kdyby totiž platila, dostali bychom sečtením těchto nerovností spor. To ovšem znamená, že jakmile algoritmus dosáhne vrcholu některého cyklu o záporné délce, bude v tomto cyklu vždy existovat nejméně jedna hrana, která nerovnost (6.3) nesplňuje, a která tedy vynutí další pokračování výpočtu.

Tvrzení o hodnotách $U(x)$ a $P(x)$ plyne z důkazu lemmatu 6.2.4. □

Poznámka: Tato věta samozřejmě platí pro všechny algoritmy, které jsou ze základního schématu 6.2.1 odvozeny.

Pomocí hodnot $P(x)$ lze během výpočtu poznat, že v grafu je cyklus se zápornou délkou. Chceme-li jej také najít (tj. nejen zjistit, že existuje), je třeba udržovat hodnoty Odkud podle 6.2.6, str. 93. Záporný cyklus se pak v hodnotách Odkud dříve či později objeví:

6.8.3 Věta. Graf obsahuje cyklus se zápornou délkou dostupný z vrcholu r právě tehdy, když algoritmus 6.3.1 nebo 6.3.3 dosáhne stavu, kdy hrany obsažené v hodnotách ODKUD tvoří cyklus. Pokud hodnoty ODKUD obsahují cyklus, pak tento cyklus má zápornou délku.

DŮKAZ: Důkaz, že pokud hodnoty ODKUD obsahují cyklus, má tento cyklus zápornou délku, je obsažen v důkaze lemmatu 6.2.4.

Předpokládejme tedy naopak, že v grafu existuje cyklus se zápornou délkou a že tento cyklus je dostupný z vrcholu r. V každém okamžiku (nekonečného) výpočtu pro každý vrchol v platí, že $U(v)$ je délkou sledu z vrcholu r do vrcholu v. Označme C počet všech cest, které začínají ve vrcholu r. Nejpozději po C změnách hodnot U již nemůže být pravdou, že každá hodnota $U(v)$ je délkou nějaké cesty z r do v. Některá hodnota $U(v)$ proto musí být délkou sledu, který není cestou, a který tedy obsahuje cyklus. Tento cyklus je dostupný z vrcholu r a hrany tohoto cyklu jsou obsaženy v hodnotách ODKUD těch vrcholů, kterými cyklus prochází. □

6.8.4 Hledání cyklů se zápornou délkou. Použijeme dva triky:

Abychom se nemuseli starat, zda případný záporný cyklus je dostupný z výchozího vrcholu, přidáme ke grafu jeden nový vrchol (označíme jej r) a nové hrany o nulové délce z vrcholu r do všech vrcholů původního grafu. Existence nebo neexistence cyklu o záporné délce tím není nijak ovlivněna, ale všechny vrcholy jsou nyní dostupné z přidaného výchozího vrcholu r.

Vrchol r není nutno přidávat doopravdy. Stačí, když výpočet zahájíme tak, jako kdyby tento vrchol byl ke grafu přidán a vzápětí byly zpracovány všechny hrany, které z něj vycházejí: pro všechny vrcholy původního grafu položíme $U(x) := 0$ a ODKUD$(x) := h$, kde h je fiktivní jméno, které znamená „některá z přidaných hran“. V dalším výpočtu už vrchol r nikdy nevstoupí do hry. Pozor: díky odlišné inicializaci takto upravený algoritmus již nepočítá vzdálenosti v původním grafu, ale vzdálenosti od přidaného vrcholu r. To však nevadí, teď nám jde o záporné cykly.

Další trik se týká evidence počtu hran $P(x)$ v cestě (nebo sledu), jejíž délka je $U(x)$. Uděláme to podobně jako v 6.2.9, str. 95, pouze na začátku pro každý vrchol x položíme $P(x) := 1$, abychom započítali hranu z přidaného vrcholu r.

Pokud graf neobsahuje cyklus záporné délky, výpočet skončí a pro všechny vrcholy x bude platit $P(x) \leq n$ (kde n je počet vrcholů původního grafu).

Pokud graf obsahuje cyklus záporné délky, pak se díky předchozí větě 6.8.3 některý z těchto cyklů objeví v hodnotách ODKUD a hrany tohoto cyklu potom vynutí další pokračování výpočtu, dokud pro některý vrchol x nenastane $P(x) > n$.

Během výpočtu tedy budeme pravidelně (např. vždy při každém zvýšení hodnoty $P(y)$) testovat, zda $P(y) > n$. Pokud to nastane, výpočet ukončíme, neboť pak totiž $U(y)$ je délkou a $P(y)$ je počtem hran ve sledu z r do y. Tento sled není cestou, a obsahuje tedy cyklus záporné délky. Hrany tohoto cyklu jsou obsaženy v hodnotách ODKUD. Postupem proti směru hran ODKUD pak lze záporný cyklus najít.

Test, zda hodnoty ODKUD obsahují cyklus, by bylo možno dělat i dříve, než nastane $P(y) > n$. Případný záporný cyklus bychom tak mohli najít dříve. Takový test však vyžaduje čas úměrný n, jeho časté opakování by mohlo zhoršit časový odhad.

Kapitola 7

Algebraické souvislosti úloh o sledech

Cílem této kapitoly je ukázat několik úloh, které, ač jsou na prvý pohled rozličné, mají společné jádro. Přesněji, platí pro ně analogické vztahy a lze je řešit analogickými postupy.

Předem poznamenejme, že tato kapitola klade trochu větší nároky na schopnost abstraktního myšlení a mnoha čtenářům se může zdát obtížnější. Kapitolu lze přeskočit bez ztráty souvislostí, zbytek knihy na ni nenavazuje.

V této kapitole nám půjde o dvě částečně se překrývající skupiny úloh.

Úlohy první skupiny lze charakterizovat takto: Je dán orientovaný graf s ohodnocenými hranami a dále je dán způsob, jak určit hodnotu libovolného orientovaného sledu na základě hodnot jeho hran (tuto operaci označíme $\otimes$), a způsob, jak z hodnot všech orientovaných sledů spojujících dané dva vrcholy a, b určit hodnotu spojení mezi těmito vrcholy (tuto operaci označíme $\oplus$).

Například při výpočtu délky nejkratšího sledu sečítáme délky jednotlivých hran ve sledu (což je zde operace $\otimes$) a z takto získaných hodnot sledů vybíráme minimum (což je zde operace $\oplus$). Při hledání sledu s největší propustností počítáme propustnost každého sledu jako minimum z ohodnocení hran (operace $\otimes$) a z takto získaných propustností jednotlivých sledů vybíráme maximum (operace $\oplus$). Další příklady použití jsou uvedeny v oddíle 7.2.

Jednotlivé úlohy první skupiny se liší vlastně jen algebraickými operacemi $\otimes$ a $\oplus$. Pokud tyto operace splňují podmínky pro tzv. *uzavřený polookruh* (viz 7.1), lze pro řešení těchto úloh použít postupy popsané v 7.3 a 7.4. Při splnění některých dalších podmínek lze pro řešení těchto úloh také upravit postupy popsané v kapitole 6 o nejkratších cestách.

Druhá skupina úloh těsně souvisí s řešením soustav lineárních rovnic. K soustavě rovnic lze sestrojit graf s ohodnocenými hranami, řešení soustavy budeme hledat jako ohodnocení vrcholů, které má určitý vztah k ohodnocení hran. V oddíle 7.5

uvedeme několik způsobů řešení, které podstatně využívají grafový popis úlohy. Poznamenejme, že v některých případech soustava lineárních rovnic vyjadřuje vztahy, které lze přímo znázornit grafem. Pak není nutno soustavu rovnic explicite sestavovat, její řešení lze získat přímo z grafu.

Nejprve se v oddílech 7.1 až 7.4 budeme zabývat úlohami první skupiny.

7.1 Polookruh a uzavřený polookruh

Abychom mohli v konečném čase spočítat hodnoty spojení, tj. hodnoty, které závisí na všech (tedy možná i nekonečně mnoha) sledech mezi dvěma vrcholy, musí algebraické operace $\oplus$ a $\otimes$ splňovat určité podmínky. Tyto podmínky jsou shrnuty v algebraických pojmech polookruhu a uzavřeného polookruhu. Budou-li tyto podmínky splněny, budeme moci zaručit, že algoritmy pro výpočet hodnot spojení pracují správně.

7.1.1 Polookruh. *Polookruh* je uspořádaná pětice $\mathcal{P} = (P, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, která se skládá z

- množiny prvků P (tzv. *nosné množiny*),
- binární operace sečítání $\oplus$ na množině P,
- binární operace násobení $\otimes$ na množině P,
- tzv. *nulového prvku* $\mathbf{0} \in P$,
- tzv. *jednotkového prvku* $\mathbf{1} \in P$,

jestliže pro všechna $a, b, c \in P$ platí tyto podmínky:

1. $a \oplus b = b \oplus a$,
2. $a \oplus (b \oplus c) = (a \oplus b) \oplus c$,
3. $a \oplus \mathbf{0} = \mathbf{0} \oplus a = a$,
4. $a \otimes (b \otimes c) = (a \otimes b) \otimes c$,
5. $a \otimes \mathbf{1} = \mathbf{1} \otimes a = a$,
6. $a \otimes \mathbf{0} = \mathbf{0} \otimes a = \mathbf{0}$,
7. $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$,
8. $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$.

7.1.2 Příklad: čísla. Nejběžnějším příkladem polookruhu je polookruh reálných čísel, kde nosná množina P je množinou všech čísel, operace $\oplus$ a $\otimes$ jsou běžné operace sečítání a násobení a roli nulového a jednotkového prvku hrají běžná nula a běžná jednička, tj. $\mathbf{0} = 0$ a $\mathbf{1} = 1$.

Pokud jako nosnou množinu P zvolíme všechna přirozená čísla (včetně nuly), dostaneme také polookruh a podobně tomu je i s některými dalšími množinami čísel, např. s množinou všech celých čísel, množinou všech racionálních čísel atd. Je důležité, aby nosná množina P byla tzv. uzavřená vůči operacím $\oplus$ a $\otimes$, tj. aby pro každou dvojici operandů z množiny P byl výsledek operace také prvkem množiny P.

7.1.3 Poznámky k definici polookruhu. Prvky množiny P nemusí být čísla a operace $\oplus$ a $\otimes$ nemusí být klasickými operacemi s čísly. Operace $\oplus$ a $\otimes$ mohou být jakákoli zobrazení $\oplus : P^2 \to P$, $\otimes : P^2 \to P$, která přiřazují každé uspořádané dvojici prvků z P nějaký prvek z P (totiž jejich „součet“ a „součin“), pokud tato zobrazení splňují podmínky 1 až 8.

Prvky $\mathbf{0} \in P$ a $\mathbf{1} \in P$ se od ostatních prvků množiny P liší tím, že splňují podmínky 3, 5 a 6. Tyto prvky nazýváme *nulovým* a *jednotkovým prvkem* polookruhu, ale pozor: tyto názvy nevyjadřují číselnou hodnotu, ale roli, kterou tyto prvky hrají vůči operacím $\oplus$ a $\otimes$. Říkáme, že nulový prvek $\mathbf{0}$ je neutrální vůči sečítání (podmínka 3) a agresívní vůči násobení (podmínka 6), zatímco jednotkový prvek $\mathbf{1}$ je neutrální vůči násobení.

Operaci $\otimes$ použijeme k výpočtu hodnoty sledu na základě hodnot jeho hran. Prvek $\mathbf{1}$ použijeme jako hodnotu triviálního sledu, který nemá žádnou hranu, a proto hodnotu sledu neovlivní. Prvek $\mathbf{0}$ bude vyjadřovat neexistenci sledu. Podmínky 4, 5 a 6 zaručují, že operace $\otimes$ má z tohoto hlediska „rozumné“ vlastnosti.

Operaci $\oplus$ použijeme k výpočtu hodnoty spojení mezi vrcholy na základě hodnot sledů. Podmínky 1, 2 a 3 zaručují rozumné vlastnosti samotné operace $\oplus$.

Všimněte si, že u operace $\oplus$ je v definici polookruhu požadována komutativita, tj. $a \oplus b = b \oplus a$. Je to proto, aby nezáleželo na pořadí, v jakém sečítáme hodnoty sledů. Naproti tomu u operace $\otimes$ komutativita požadována není. To znamená, že výsledek násobení obecně může záviset na pořadí činitelů. Toto pořadí odpovídá pořadí hran ve sledu, jehož hodnotu počítáme.

Podmínky 7 a 8 (distributivní zákony) jsou nejdůležitější: umožňují počítat hodnoty spojení mezi dvěma vrcholy, aniž bychom museli explicite znát a sečítat hodnoty všech jednotlivých sledů mezi těmito vrcholy.

Nyní uvedeme několik příkladů polookruhů.

7.1.4 Příklad: booleovský polookruh. Jednoduchým příkladem polookruhu je tzv. *booleovský polookruh* $\mathcal{B} = (P, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, definovaný takto: $P = \{0, 1\}$, $\mathbf{0} = 0$ a $\mathbf{1} = 1$. Operace $\oplus$ a $\otimes$ jsou definovány tabulkami

$\oplus$	0	1
0	0	1
1	1	1

$\otimes$	0	1
0	0	0
1	0	1

Platí tedy $x \oplus y = \max(x, y)$, $x \otimes y = \min(x, y)$. Interpretujeme-li hodnotu 1 jako „pravda“ a 0 jako „nepravda“, pak operace $\oplus$ je logickým součtem ($x \vee y$), operace $\otimes$ je logickým součinem ($x \,\&\, y$). Ověřte platnost podmínek 1 až 8.

7.1.5 Příklad: množiny Nosnou množinou P je množina všech podmnožin nějaké množiny M. Operace $\oplus$ je sjednocení a operace $\otimes$ je průnik podmnožin. Nulovým prvkem $\mathbf{0}$ je prázdná množina, jednotkovým prvkem $\mathbf{1}$ je celá množina M. Ověřte, že je to polookruh.

7.1.6 Uzavřený polookruh. Polookruh $\mathcal{P} = (P, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ nazveme *uzavřeným polookruhem*, jestliže kromě podmínek 1 až 8 z 7.1.1 splňuje navíc podmínky:

9. Každé posloupnosti $\{a_i\}_{i=1}^{\infty}$ prvků z množiny P je přiřazen určitý prvek z P, který nazveme *součtem* a označíme $a_1 \oplus a_2 \oplus a_3 \oplus \ldots$

10. Podmínky 1, 2, 3, 7 a 8 platí i pro nekonečné součty.

V uzavřeném polookruhu definujeme operaci *uzávěru* $*$ předpisem

$$a^* = \mathbf{1} \oplus a \oplus (a \otimes a) \oplus (a \otimes a \otimes a) \ldots$$

Z podmínek 1 až 10 vyplývají následující vlastnosti operace $*$:

$$\begin{aligned} \mathbf{0}^* &= \mathbf{1}\,, \\ a \otimes a^* &= a^* \otimes a\,, \\ a^* &= \mathbf{1} \oplus (a \otimes a^*)\,. \end{aligned} \tag{7.1}$$

Poznamenejme, že existence součtu nekonečné řady (tj. podmínka 9) nevyplývá z vlastností polookruhu 1 až 8. Vzpomeňte si na konvergenci řad reálných čísel. Reálná čísla s běžným sečítáním a násobením tedy jsou polookruhem, ale nikoli uzavřeným polookruhem.

Pokud se omezíme pouze na nezáporná čísla (reálná nebo přirozená) s běžnými operacemi sčítání a násobení a přidáme-li k množině P prvek ∞, dostaneme polookruh uzavřený, v němž platí

$$a^* = \begin{cases} 1/(1-a) & \text{pro } a < 1, \\ \infty & \text{pro } a \geq 1. \end{cases}$$

Booleovský polookruh 7.1.4 a polookruh množin 7.1.5 jsou také uzavřenými polookruhy. Příklady dalších uzavřených polookruhů jsou v tabulce 7.1 na následující straně, příklady jejich použití a několik dalších polookruhů jsou uvedeny v oddíle 7.2.

Poznamenejme, že podstatný rozdíl mezi polookruhem a uzavřeným polookruhem je ten, že v uzavřeném polookruhu zaručujeme existenci a rozumné vlastnosti *všech* nekonečných součtů.

Vlastnosti uzavřeného polookruhu využijeme tehdy, když graf obsahuje cykly, a mezi některými vrcholy tedy existuje nekonečně mnoho sledů. I v těchto případech pak lze zaručit existenci a rozumné vlastnosti hodnot spojení. Pomocí operace uzávěru $*$ budeme schopni v konečném počtu kroků sečíst hodnoty všech (nekonečně mnoha) sledů mezi dvěma vrcholy.

Konkrétně, operaci uzávěru a^* použijeme pro výpočet součtu hodnot všech sledů, které opakovaně (nula nebo vícekrát) procházejí cyklem, který má hodnotu a.

7.1.7 Polookruhy matic. Z prvků libovolného polookruhu můžeme sestavovat matice podobně, jako děláme matice čísel. Pro matice s prvky z nějakého polookruhu můžeme definovat operace sečítání a násobení způsobem, na který jsme zvyklí z oboru reálných čísel, pouze s tím rozdílem, že běžné operace $+$ a $\cdot$ nahradíme polookruhovými operacemi $\oplus$ a $\otimes$.

	P	$\oplus$	$\otimes$	$\mathbf{0}$	$\mathbf{1}$	a^*	viz
$\mathcal{B}$	$\{0,1\}$	max	min	0	1	1	7.2.3
	tranzitivní uzávěr grafu						
$\mathcal{P}_1$	$\mathbb{R}\cup\{-\infty,\infty\}$	min	$+$	∞	0	0 pro $a\geq 0$ $-\infty$ pro $a<0$	7.2.4
	nejkratší sled						
$\mathcal{P}_2$	$\mathbb{R}\cup\{-\infty,\infty\}$	max	$+$	$-\infty$	0	∞ pro $a>0$ 0 pro $a\leq 0$	7.2.5
	nejdelší sled						
$\mathcal{P}_3$	$\langle 0,1\rangle$	max	$\cdot$	0	1	1	7.2.6
	nejspolehlivější sled						
$\mathcal{P}_4$	konečná uspořádaná množina	max	min	$\min\{P\}$	$\max\{P\}$	$\max\{P\}$	7.2.7
	nejširší sled						
$\mathcal{P}_5$	podmnožiny množiny M	$\cup$	$\cap$	$\emptyset$	M	M	7.2.8
	které předměty lze dopravit						
$\mathcal{P}_6$	$\mathbb{N}\cup\{\infty\}$	$+$	$\cdot$	0	1	1 pro $a=0$ ∞ pro $a\geq 1$	7.2.2
	počet orientovaných sledů						
$\mathcal{P}_7$	$\langle 0,\infty)\cup\{\infty\}$	$+$	$\cdot$	0	1	$1/(1-a)$ pro $a<1$ ∞ pro $a\geq 1$	7.2.9
	náhodná procházka, pravděpodobnost dosažení vrcholu						

Tabulka 7.1: Příklady uzavřených polookruhů. Další příklady jsou uvedeny v textu 7.2.10 až 7.2.11.

Příkladem je upravené násobení matic, které jsme použili v 6.7.1, str. 106, pro výpočet matice vzdáleností.[1] Jednalo se o násobení matic, jejichž prvky byly z polookruhu $\mathcal{P}_1$.

Všechny matice typu $n\times n$ s prvky z polookruhu $\mathcal{P}$ tvoří polookruh. Roli nulového prvku zde hraje nulová matice (složená z nulových prvků $\mathbf{0}$ polookruhu $\mathcal{P}$), roli jednotkového prvku hraje jednotková matice E, která má na diagonále $\mathbf{1}$ a mimo diagonálu $\mathbf{0}$.

Byl-li polookruh $\mathcal{P}$ uzavřený, bude uzavřený i příslušný polookruh matic. Uzávěr matice pak je definován jako nekonečný součet

$$A^* = \mathbf{1}\oplus A\oplus(A\otimes A)\oplus(A\otimes A\otimes A)\dots$$

[1] Pozor, operace násobení je sice stejná, ale v 6.7.1 jsme násobili matice, které byly definovány trochu jinak, než jak budeme definovat matici H v 7.2.1.

7.2 Úlohy o spojeních a jejich aplikace

7.2.1 Úloha o spojeních. Předpokládejme, že je dán orientovaný graf G a uzavřený polookruh $(P, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ a že hrany grafu jsou ohodnoceny hodnotami z tohoto polookruhu, tj. je dáno zobrazení $h : E(G) \to P$. Hodnotu $h(e)$ nazvěme *hodnotou hrany* e.

Je-li Q posloupnost hran $e_1, e_2, \ldots, e_k$, která tvoří orientovaný sled, pak *hodnotu sledu* $h(Q)$ definujeme jako součin hodnot jednotlivých hran, tj.

$$h(Q) = h(e_1) \otimes h(e_2) \otimes \ldots \otimes h(e_k) \ .$$

Hodnotu triviálního sledu (tj. sledu, který neobsahuje žádnou hranu) definujeme jako $\mathbf{1}$. Poznamenejme, že pokud operace $\otimes$ není komutativní (a to být nemusí), je třeba násobit hodnoty hran ve stejném pořadí, v jakém se vyskytují ve sledu.

Jsou-li x, y dva vrcholy grafu, pak *hodnotu spojení* z vrcholu x do vrcholu y značíme $s(x, y)$ a definujeme ji jako součet $\oplus$ hodnot všech (tj. možná i nekonečně mnoha) orientovaných sledů z vrcholu x do vrcholu y. Poznamenejme, že tento součet v uzavřeném polookruhu vždy existuje. Právě proto jsme požadovali, aby polookruh byl uzavřený.

Pokud z x do y nevede žádný sled, pak hodnota spojení $s(x, y) = \mathbf{0}$, což odpovídá součtu prázdné množiny sčítanců.

Úkolem je najít hodnotu spojení $s(x, y)$ pro dané (např. všechny) uspořádané dvojice vrcholů x, y.

Metodami výpočtu hodnot spojení se budeme zabývat v 7.3 a 7.4, nejprve však uvedeme několik konkrétních příkladů.

Pro snazší vyjadřování zaveďme matici H, jejíž libovolný prvek $h(x, y)$ je roven součtu hodnot všech hran vedoucích z x do y. Nevede-li z x do y žádná hrana, klademe $h(x, y) = \mathbf{0}$.

Hledané hodnoty spojení $s(x, y)$ uspořádejme též do tvaru matice, označme ji S. Později ukážeme, že $S = H^*$, viz 7.3.8, str. 124.

7.2.2 Počet orientovaných sledů. Každou hranu grafu ohodnotíme jedničkou z polookruhu $\mathcal{P}_6$ přirozených čísel s ∞ a s běžnými operacemi sčítání a násobení. Hodnota spojení $s(x, y)$ je pak rovna počtu všech orientovaných sledů z x do y.

Matice H je zde maticí sousednosti, jejími prvky jsou násobnosti hran (viz 1.8.1, str. 24). Pokud graf obsahuje cyklus, pak některé hodnoty spojení v matici S jsou rovny ∞.

7.2.3 Tranzitivní a reflexivní uzávěr, dostupnost. Každou hranu grafu ohodnotíme jednotkou z booleovského polookruhu $\mathcal{B}$. Potom každý sled bude mít hodnotu 1 a hodnota spojení $s(x, y)$ bude rovna 1 právě tehdy, existuje-li orientovaný sled z x do y (srov. s 5.1.10, str. 71, a 5.4, str. 81).

Matici dostupnosti D_G tedy získáme jako uzávěr H^* matice hodnot hran H. Je-li graf prostý, je matice H rovna matici sousednosti M_G^+ grafu G.

7.2.4 Nejkratší sled. Délky hran budeme chápat jako hodnoty z uzavřeného polookruhu $\mathcal{P}_1$, v němž $\oplus = \min$ a $\otimes = +$. Potom hodnota spojení $s(x, y)$ je rovna délce nejkratšího sledu z vrcholu x do vrcholu y. Jestliže graf neobsahuje cyklus se zápornou délkou, pak podle 6.1.4 je matice hodnot spojení S rovna matici vzdáleností. V opačném případě budou některé hodnoty spojení rovny $-\infty$.

7.2.5 Nejdelší sled. Délky hran budeme chápat jako hodnoty z uzavřeného polookruhu $\mathcal{P}_2$, v němž $\oplus = \max$ a $\otimes = +$. Potom hodnota spojení $s(x, y)$ je rovna délce nejdelšího sledu z vrcholu x do vrcholu y. Obsahuje-li graf cyklus s kladnou délkou, budou některé hodnoty spojení rovny ∞.

7.2.6 Nejspolehlivější sled. Síť pro dálkový přenos dat znázorněme grafem tak, že jednotlivým linkám odpovídají hrany. Každá hrana je ohodnocena pravděpodobností, že příslušná linka bude po dobu přenosu (např. 10 minut) pracovat bez poruchy. Předpokládáme, že poruchy jednotlivých linek jsou navzájem nezávislé. Hledáme sled mezi danými dvěma vrcholy s největší spolehlivostí.

Pravděpodobnosti jednotlivých hran budeme chápat jako hodnoty z uzavřeného polookruhu $\mathcal{P}_3$, kde $P = \langle 0, 1\rangle$ a kde $\oplus = \max$ a $\otimes$ je běžné násobení. Hodnotou sledu je jeho spolehlivost, tj. pravděpodobnost, že všechny jeho hrany budou fungovat. Díky nezávislosti poruch je spolehlivost sledu rovna součinu pravděpodobností jeho hran. Hodnota spojení $s(x, y)$ je pak rovna spolehlivosti nejspolehlivějšího sledu z x do y. (Srovnejte s 2.4.1, str. 46, kde je pomocí logaritmů tato úloha převedena na hledání nejkratší cesty.)

7.2.7 Sled s největší propustností (nejširší sled). Spojovací síť (ale třeba i vodovodní) opět znázorníme grafem tak, že linkám odpovídají hrany. Každá hrana grafu je ohodnocena propustností (přenosovou kapacitou, šířkou). Propustnost sledu je rovna propustnosti nejméně propustné hrany. Hledáme sled s největší propustností, která spojuje dané dva vrcholy.

Propustnosti jednotlivých hran budeme pokládat za hodnoty z uzavřeného polookruhu $\mathcal{P}_4$, v němž $\oplus = \max$ a $\otimes = \min$. Potom hodnota spojení $s(x, y)$ je rovna maximální propustnosti sledu z vrcholu x do vrcholu y.

Zmínku zasluhuje volba množiny P v tomto polookruhu. Zdánlivě stačí, aby P byla množina všech hodnot propustností hran. Jako šířku sledu totiž nemůžeme dostat nic, co by neleželo v této množině. Pokud ovšem chceme, abychom neexistenci sledu z vrcholu x do vrcholu y mohli poznat podle toho, že $s(x, y) = \mathbf{0}$, je třeba, aby prvkem $\mathbf{0}$ (tj. nejmenším prvkem množiny P) nebyla ohodnocena žádná hrana. Pro větší názornost si ovšem můžeme k množině P přidat i další hodnoty, např. $-\infty$ a ∞, které pak budou plnit role nulového a jednotkového prvku.

Za zmínku také stojí fakt, že pokud množina P je dvouprvková, pak polookruh $\mathcal{P}_4$ je booleovským polookruhem.

7.2.8 Které předměty lze dopravit. Máme množinu M nebezpečných předmětů a dopravní síť, kterou pokládáme za orientovaný graf. Pro každou hranu grafu

máme seznam předmětů, které lze přes tuto hranu dopravit. Máme zjistit, které předměty lze po hranách grafu přepravit mezi dvěma danými vrcholy.

Množiny předmětů, které lze dopravovat přes jednotlivé hrany grafu, budeme pokládat za hodnoty z polookruhu $\mathcal{P}_5$ (viz 7.1.5) s operacemi $\oplus = \cup$ a $\otimes = \cap$. Hodnotou sledu je množina předmětů, které lze dopravovat po všech hranách sledu, tj. průnik množin jednotlivých hran sledu. Hodnota spojení pak je sjednocením hodnot (tj. množin) jednotlivých sledů mezi danými vrcholy.

Dodejme, že ve speciálním případě, kdy množina M je jednoprvková, je polookruh $\mathcal{P}_5$ booleovský a úloha o přepravě předmětů je pak podobná úloze 7.2.3.

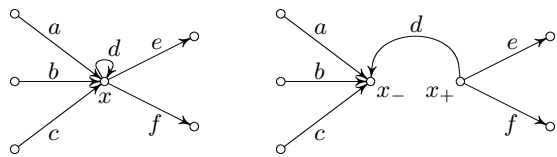
7.2.9 Náhodná procházka I – pravděpodobnost dosažení vrcholu. Představme si tuláka, který putuje podél hran grafu a v každém vrcholu si volí další postup náhodně. Pro každou hranu je známa pravděpodobnost, že tulák půjde touto hranou, přišel-li do jejího počátečního vrcholu. Pro jednoduchost předpokládejme, že každá hrana má nenulovou pravděpodobnost (hrany s nulovou pravděpodobností lze vynechat). Je zřejmé, že součet pravděpodobností hran vycházejících z téhož vrcholu musí být roven jedné. Jestliže z nějakého vrcholu nevychází žádná hrana, funguje tento vrchol jako past: pokud do něj tulák přijde, musí tam už zůstat. Podobný efekt má i smyčka, která má jednotkovou pravděpodobnost. Proto budeme předpokládat, že v grafu žádná taková smyčka není.

Úkolem je spočítat pravděpodobnost, že se tulák z daného výchozího vrcholu x někdy (lhostejno kdy) dostane do daného cílového vrcholu y. Hledaná pravděpodobnost je součtem pravděpodobností všech sledů z x do y, které neprocházejí vrcholem y. Přitom pravděpodobnost sledu je součinem pravděpodobností jeho hran.

Existují samozřejmě i serióznější aplikace této úlohy, a to všude tam, kde jde o náhodný proces s konečně mnoha stavy a kde se pravděpodobnosti změn stavů v čase nemění. V teorii náhodných procesů se jim říká markovské řetězce. V dalším výkladu budeme u náhodných procházek pojmy vrchol a stav používat jako synonyma.

Pravděpodobnosti hran chápejme jako hodnoty z uzavřeného polookruhu nezáporných reálných čísel $\mathcal{P}_7$ s běžným sečítáním a násobením. Potom hodnota spojení $s(x, y)$ je rovna součtu pravděpodobností všech sledů z x do y, bohužel však i takových sledů, které obsahují vrchol y několikrát. Pro výpočet pravděpodobnosti, že tulák dosáhne vrcholu y, musíme tyto sledy vyloučit.

Jestliže $d^+(y) = 0$, pak zřejmě hodnota spojení $s(x, y)$ je přímo hledanou pravděpodobností, že se tulák dostane z x do y.



Obrázek 7.1: Rozpůlení vrcholu (ilustrace k 7.2.9).

Jestliže $d^+(y) > 0$, je třeba před výpočtem hodnoty spojení $s(x, y)$ graf upravit tak, aby do $s(x, y)$ nebyly zahrnuty sledy, které obsahují vrchol y několikrát. Je-li $x \neq y$, stačí odstranit všechny hrany z $E^+(y)$. Je-li $x = y$, pak můžeme vrchol x nahradit dvojicí vrcholů x_+ a x_-, všechny hrany z $E^-(x)$ nahradíme hranami končícími v x_- a hrany z $E^+(x)$ nahradíme hranami vycházejícími z x^+. Toto „rozpůlení“ vrcholu je znázorněno na obrázku 7.1 na předchozí straně. Hledaná pravděpodobnost je pak rovna hodnotě spojení $s(x_+, x_-)$.

7.2.10 Náhodná procházka II – počet průchodů vrcholem. Předchozí úlohu si maličko upravme: Předpokládejme, že graf není silně souvislý, a že všechny silně souvislé komponenty, ze kterých nevychází ven žádná hrana, mají jen jeden vrchol, přičemž v tomto vrcholu není ani smyčka. Pokud do takového vrcholu tulák přijde, jeho procházka zde skončí. V teorii markovských řetězců se takovému vrcholu říká absorbující stav. Lze dokázat, že za výše uvedených předpokladů náhodná procházka s pravděpodobností jedna skončí v některém absorbujícím stavu. Jinak řečeno, pravděpodobnost, že tulák ještě nenarazil na absorbující stav, se s narůstajícím časem (počtem kroků) limitně blíží k nule.

Nyní si můžeme položit trochu těžší otázku: Když tulák zahájil náhodnou procházku ve vrcholu i, kolikrát projde vrcholem j, než jeho procházka skončí v nějakém absorbujícím stavu? Tento počet je samozřejmě náhodný, zajímá nás jeho střední hodnota.

Ukážeme, že střední počet průchodů stavem j, když procházka začala ve stavu i, je roven hodnotě spojení $s(i, j)$ v polookruhu $\mathcal{P}_7$ s běžným sečítáním a násobením. Poznamenejme, že jde o zcela stejný polookruh jako v předchozí úloze, tam jsme však mluvili o hodnotách spojení $s(x, y)$ jen v případě, kdy z vrcholu y nevycházela žádná hrana (byl to absorbující stav). Nyní je řeč o hodnotě spojení mezi libovolnými dvěma vrcholy.

Vezměme všechny sledy, které začínají ve výchozím vrcholu i. Každý takový sled představuje možný začátek náhodné procházky začínající ve vrcholu i. Pravděpodobnost, že procházka bude začínat právě takto, je rovna součinu pravděpodobností všech hran ve sledu. Vezměme nyní pouze sledy, které začínají v i a končí v j. Za každý takový sled $\mathcal{S}$ započítejme jeden průchod stavem j. Je sice pravda, že sled $\mathcal{S}$ by mohl vrcholem j procházet i vícekrát, ale tyto předchozí průchody jsou již započteny díky sledům, které jsou počátečním úsekem sledu $\mathcal{S}$ a které končí ve vrcholu j při některém dřívějším průchodu tímto vrcholem. Každý průchod stavem j ovšem musíme započítat s pravděpodobností, že k němu dojde; tato pravděpodobnost je rovna pravděpodobnosti příslušného sledu. Výsledná střední hodnota počtu průchodů vrcholem j tedy je rovna součtu hodnot (tj. pravděpodobností) všech sledů, které vedou z i do j. To je ovšem hodnota spojení $s(i, j)$.

Za zmínku stojí případ, kdy vrchol j je absorbující. Takový vrchol se může v náhodné procházce vyskytnout buď jedenkrát jako poslední, anebo vůbec ne. Pokud tedy začínáme ve vrcholu i, pak střední počet výskytů vrcholu j je roven pravděpodobnosti, že bude vrcholu j dosaženo, a to je $s(i, j)$.

Dodejme, že pokud víme, že náhodná procházka začne ve vrcholu i a známe-li střední počty průchodů jednotlivými stavy $j \in V$, pak snadno zjistíme také

střední délku náhodné procházky (tj. počet hran) než procházka skončí v některém absorbujícím stavu. Než procházka skončí, bude se tulák v každém kroku nacházet vždy v nějakém neabsorbujícím stavu (vrcholu). Stačí tedy sečíst počty, kolikrát se v jednotlivých neabsorbujících stavech vyskytne, a dostaneme střední délku procházky.

Střední délka procházky je tedy rovna součtu hodnot spojení z vrcholu i do všech neabsorbujících vrcholů j, tj. $\sum_{j\in T} s(i,j)$, kde T je množina všech neabsorbujících stavů.

7.2.11 Náhodná procházka III – čas dosažení vrcholu. Je-li v úloze o náhodné procházce 7.2.9 pro každou hranu e známa kromě pravděpodobnosti navíc očekávaná (průměrná) doba t_e průchodu touto hranou, můžeme se ptát, jaká je střední hodnota času, který tulák potřebuje, aby se svým náhodným pohybem dostal z vrcholu x poprvé do vrcholu y. Tento čas je roven váženému průměru časů průchodu všech (tj. možná i nekonečně mnoha) sledů, které začínají v x a končí v y, aniž by vrcholem y procházely.

Každá hrana e je zde ohodnocena dvojicí čísel (p_e, t_e), kterou můžeme chápat jako hodnotu z polookruhu $\mathcal{P}_8$, který definujeme takto:

$$\begin{aligned}
\mathcal{P}_8 &= (P, \oplus, \otimes, \mathbf{0}, \mathbf{1})\,, \\
P &= \{(x,y) \mid x > 0, y \geq 0\} \cup \{\mathbf{0}, \infty\}\,, \\
(p_1,t_1) \oplus (p_2,t_2) &= \left(p_1 + p_2, \frac{p_1 t_1 + p_2 t_2}{p_1 + p_2}\right), \\
(p_1,t_1) \otimes (p_2,t_2) &= (p_1 p_2, t_1 + t_2)\,, \\
\infty \oplus (p,t) &= \infty\,, \\
\infty \otimes (p,t) &= (p,t) \otimes \infty \;=\; \infty\,, \\
\mathbf{1} &= (1,0)\,.
\end{aligned}$$

Prvek $\mathbf{0}$ si můžeme představit jako náhražku za množinu všech dvojic $(0,t)$ (tj. pravděpodobnost je nulová, a na čase t tedy nezáleží). Takto definovaný polookruh je uzavřený, operaci $*$ lze pro $p < 1$ počítat podle vzorce

$$(p,t)^* = \left(\frac{1}{1-p}, \frac{pt}{1-p}\right),$$

zatímco pro $p \geq 1$ platí $(p,t)^* = \infty$.

Hodnotu ∞ potřebujeme pouze formálně, aby polookruh byl uzavřený.

Hodnota sledu, jehož hrany jsou ohodnoceny $(p_1,t_1),\ldots,(p_k,t_k)$, je dvojice (p,t), kde $p = p_1 p_2 \ldots p_k$ je pravděpodobnost, že tulák půjde podél tohoto sledu, a $t = t_1 + t_2 + \ldots + t_k$ je průměrný čas k tomu potřebný. Jsou-li nyní $(p_1,t_1),\ldots,(p_k,t_k)$ hodnoty sledů z vrcholu x do vrcholu y, pak součet jejich hodnot je dvojice (p,t), kde $p = p_1 + p_2 + \ldots + p_k$ a čas t je vážený průměr časů $t_1, t_2, \ldots, t_k$, tj. $t = (p_1 t_1 + p_2 t_2 + \ldots p_k t_k)/p$.

O vztahu hodnot spojení $s(x,y)$ k hledaným hodnotám pravděpodobnosti a času prvého příchodu do vrcholu y z vrcholu x platí totéž co v příkladě 7.2.9: Jestliže $d^+(y) = 0$, pak $s(x,y) = (p,t)$ je přímo hledaná pravděpodobnost a čas. Jestliže $d^+(y) > 0$, pak je nutno vymazávat hrany z $E^+(y)$ nebo půlit vrchol stejně jako v příkladě 7.2.9.

7.3 Algoritmy úloh o spojeních

7.3.1 Kleeneho algoritmus. K výpočtu hodnot spojení lze použít podobnou metodu jako ve Floydově algoritmu 6.7.3, str. 107: Budeme postupně počítat posloupnost matic $S_0, S_1, S_2, \ldots, S_n$, jejichž prvky $s_k(i,j)$ jsou definovány takto: $s_k(i,j)$ je součtem hodnot všech netriviálních sledů z vrcholu i do vrcholu j takových, že (kromě počátečního a koncového vrcholu) obsahují pouze vrcholy z množiny $\{1,2,\ldots,k\}$.

Matice S_0 je rovna výchozí matici hodnot hran H. Hledanou matici hodnot spojení S dostaneme připočtením hodnot triviálních sledů (tj. hodnot $\mathbf{1}$) k diagonálním prvkům poslední matice S_n, tj. jako součet $S = S_n \oplus E$, kde E je jednotková matice, tj. matice, která má na diagonále $\mathbf{1}$ a jinde $\mathbf{0}$.

Hodnoty $s_k(i,j)$ v matici S_k lze pro $k \geq 1$ vypočítat z hodnot prvků matice S_{k-1} podle vzorce

$$s_k(i,j) = s_{k-1}(i,j) \oplus (s_{k-1}(i,k) \otimes (s_{k-1}(k,k))^* \otimes s_{k-1}(k,j)) \,. \tag{7.2}$$

Sled z vrcholu i do vrcholu j, který prochází přes vrcholy $1,2,\ldots,k$ totiž buď

1. neprochází přes k, tomu odpovídá první sčítanec, nebo
2. jde z vrcholu i do vrcholu k, pak několikrát (nebo též ani jednou) prochází z vrcholu k do vrcholu k a nakonec pokračuje do j. Tomu odpovídá druhý sčítanec.

Podobně jako ve Floydově algoritmu, i zde lze výpočet vykonat v jediné matici, jejíž hodnoty postupně přepisujeme podle předpisu

$$S(i,j) := S(i,j) \oplus (S(i,k) \otimes S(k,k)^* \otimes S(k,j)) \,. \tag{7.3}$$

Pořadí provádění těchto úprav je zde ovšem trochu složitější než ve Floydově algoritmu, protože musíme zajistit, abychom žádný sled nepočítali dvakrát.

1. Do matice S dosadíme hodnoty hran z matice H.
2. Pro všechna $k = 1,2,\ldots,n$ provedeme postupně kroky 2a, 2b, 2c.
   - 2a. Pro všechny dvojice i, j takové, že $i \neq k$ a $j \neq k$, provedeme příkaz (7.3).
   - 2b. Pro všechny dvojice i, j takové, že buď $i = k \neq j$, nebo $i \neq k = j$, provedeme příkaz (7.3).
   - 2c. Pro $i = k = j$ provedeme příkaz (7.3).
3. Pro každé $i = 1,2,\ldots,n$ položíme $S(i,i) := S(i,i) \oplus \mathbf{1}$.

7.3.2 Vzorce pro výpočet hodnot spojení. Použijeme-li v algoritmu 7.3.1 namísto skutečných hodnot hran pouze jejich formální označení a budeme-li všechny výpočty podle příkazu (7.3) provádět pouze formálně, dostaneme nakonec jako hodnoty spojení vzorce, do nichž lze dosadit hodnoty jednotlivých hran a tak přímo počítat hodnoty spojení. Tyto vzorce bychom ostatně také mohli pokládat za prvky zvláštního polookruhu.

Půvab tohoto triku (totiž nejprve odvodit vzorce a pak do nich dosazovat) je v tom, že tyto vzorce závisí pouze na struktuře grafu a na označení jednotlivých hran, ale nezávisí na polookruhu, ve kterém se bude počítat. Tytéž vzorce lze použít v několika polookruzích. Například v úloze o náhodné procházce 7.2.9 můžeme podle stejných vzorců počítat pravděpodobnost dosažení vrcholu (polookruh $\mathcal{P}_7$) i počet různých způsobů, jak se tam tulák může dostat (polookruh $\mathcal{P}_6$).

Vzorce pro výpočet hodnot spojení lze získat i jinými metodami za předpokladu, že fungují v obecných uzavřených polookruzích. Zajímavé jsou zejména elementární úpravy grafu 7.4. Pozor: postupy, které využívají speciálních vlastností některých polookruhů (např. 7.3.4) takto (tj. k získání vzorců) použít nelze.

7.3.3 Slova rozpoznatelná konečným automatem. Myšlenku algoritmu 7.3.1 použil poprvé S. C. Kleene v roce 1956 k charakterizaci množiny slov rozpoznatelných konečným automatem. Konečný automat lze snadno popsat jako ohodnocený graf, jehož vrcholy odpovídají stavům automatu a hrany představují možné změny stavů. Každá hrana je ohodnocena písmenem, jehož výskyt na vstupu automatu způsobí příslušnou změnu stavu. Automat lze použít k rozpoznávání určité množiny slov takto: automat nastartujeme v pevně zvoleném výchozím stavu, necháme jej zpracovat dané slovo (posloupnost písmen) a podíváme se, v jakém stavu skončil. Pokud skončil v některém z předem zvolených koncových stavů, prohlásíme dané slovo za rozpoznané.

Kleene popsal způsob (algoritmus), jak z grafového popisu automatu sestrojit tzv. regulární výrazy, které ve zhuštěné formě popisují (často nekonečnou) množinu všech rozpoznávaných slov. Tyto regulární výrazy mají hodně společného se vzorci z předchozí poznámky. Podrobnější informaci lze najít v knihách [Chytil82] a [KŠCh89].

7.3.4 Optimální sledy. V polookruzích $\mathcal{P}_1, \ldots, \mathcal{P}_4$ a $\mathcal{B}$ spočívá operace $\oplus$ ve výběru minima nebo maxima. V úlohách 7.2.3 až 7.2.7 to chápeme tak, že výsledkem operace sečítání $\oplus$ je nejlepší z hodnot, které sečítáme. (V případě nekonečných součtů pak někdy vychází nekonečno.) Hodnota spojení je v těchto úlohách rovna hodnotě nejlepšího sledu mezi danými vrcholy (nebo nekonečno).

Přesněji, pokud pro každou dvojici $a, b \in P$ platí buď $a \oplus b = a$, nebo $a \oplus b = b$, můžeme definovat relaci $a \preceq b$ „býti lepší nebo stejně dobrý“ předpisem

$$a \preceq b \quad \Longleftrightarrow \quad a \oplus b = a$$

a relace $\preceq$ je uspořádáním.

V těchto polookruzích nevadí, když do hodnoty spojení započítáme hodnoty některých sledů několikrát. Díky tomu lze k výpočtu hodnot spojení použít některé algoritmy, které by v obecných polookruzích nefungovaly.

Budeme předpokládat, že v daném grafu nemá žádný cyklus hodnotu lepší než **1**, což je hodnota triviálního cyklu, který nemá žádnou hranu.

Pro polookruhy $\mathcal{B}$, $\mathcal{P}_3$ a $\mathcal{P}_4$ z tabulky 7.1, str. 115, je tento předpoklad automaticky splněn. Pro polookruh $\mathcal{P}_1$ tento předpoklad znamená neexistenci cyklu se zápornou délkou (viz 6.1.7, str. 90), pro polookruh $\mathcal{P}_2$ znamená naopak neexistenci cyklu s kladnou délkou.

Je-li uvedený předpoklad splněn, lze pro daný graf a polookruh přeformulovat všechny věty a algoritmy z kapitoly 6. Například trojúhelníková nerovnost 6.1.5, str. 89, přejde na tvar

$$s(i,j) \preceq s(i,k) \otimes s(k,j) \ .$$

V algoritmu 6.2.1 a v algoritmech z něj odvozených jsou úpravy tyto:

1. Při inicializaci pro $x \neq r$ položíme $U(x) := \mathbf{0}$.
2. Na jednotlivé hrany používáme příkaz:
   Jestliže $U(x) \otimes h(x,y) \preceq U(y)$, pak $U(y) := U(x) \otimes h(x,y)$.
   Nebo, což je vlastně totéž, provedeme $U(y) := U(y) \oplus (U(x) \otimes h(x,y))$.

Podobně lze upravit i Floydův algoritmus 6.7.3, v němž použijeme příkaz

$$M(i,j) := M(i,j) \oplus (M(i,k) \otimes M(k,j)) \ .$$

Je zde však malá záludnost:

7.3.5 Cvičení. Vysvětlete, proč ve Floydově algoritmu 6.7.3 není uveden závěrečný příkaz 3 z Kleeneho algoritmu 7.3.1.

7.3.6 Cvičení. Přeformulujte větu 6.2.7, str. 94, pro jednotlivé polookruhy $\mathcal{P}_1, \ldots, \mathcal{P}_4$ a $\mathcal{B}$. Srovnejte algoritmus 6.2.1 přeformulovaný pro booleovský polookruh $\mathcal{B}$ se značkovacím algoritmem 3.1.1 pro zjišťování orientované dostupnosti.

7.3.7 Cvičení. Najděte nejširší orientované sledy mezi všemi dvojicemi vrcholů v grafu, který je dán tímto seznamem hran:

Pv	Kv	šířka
1	2	4
1	4	2
1	5	7
2	1	4
2	3	5
2	5	3
2	6	1
3	1	2
3	2	6
3	3	10

Pv	Kv	šířka
4	1	1
4	3	8
4	6	4
5	2	6
5	3	5
5	6	7
6	1	2
6	4	8
6	5	1
6	6	3

Použijte všechny vyložené postupy a srovnejte je.

7.3.8 Sledy o předepsaném počtu hran. V některých aplikacích potřebujeme znát součet hodnot všech sledů, které vedou mezi danými vrcholy a mají předepsaný počet hran. Je-li předepsaný počet hran přesně k, získáme součty hodnot sledů umocněním matice hodnot H, tj. jako

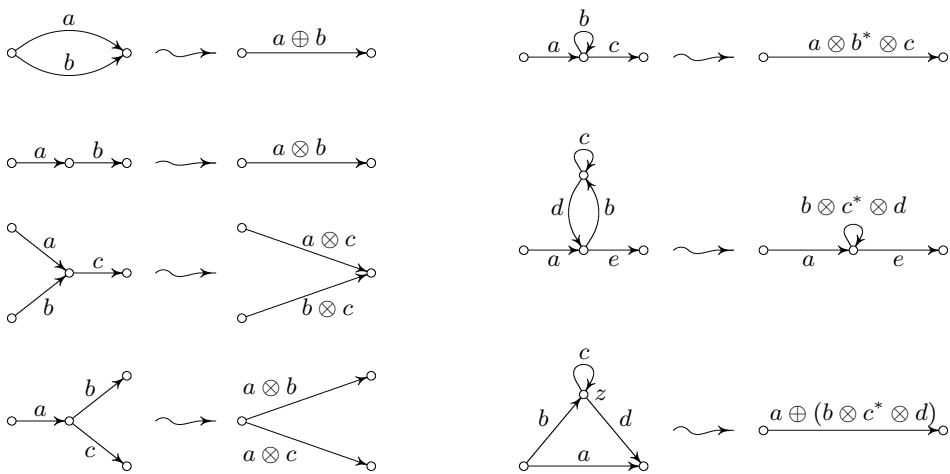
$$H^k = \underbrace{H \otimes H \otimes H \otimes \ldots \otimes H}_{k\text{-krát}} .$$

Matice přitom násobíme běžným způsobem (tj. „řádek krát sloupec“) ovšem s použitím polookruhových operací $\oplus$ a $\otimes$. Pro matici hodnot spojení platí:

$$S = H^* = E \oplus H \oplus H^2 \oplus H^3 \oplus \ldots$$

7.4 Elementární úpravy grafu

Elementární úpravy grafu se používají zejména při ručních výpočtech hodnot spojení v nevelkých grafech. Cílem elementárních úprav je graf co nejvíce zjednodušit a přitom zachovat hodnoty spojení mezi vybranými dvojicemi vrcholů. Jak se to dělá, je naznačeno na obrázku 7.2.



Obrázek 7.2: Elementární úpravy grafu.

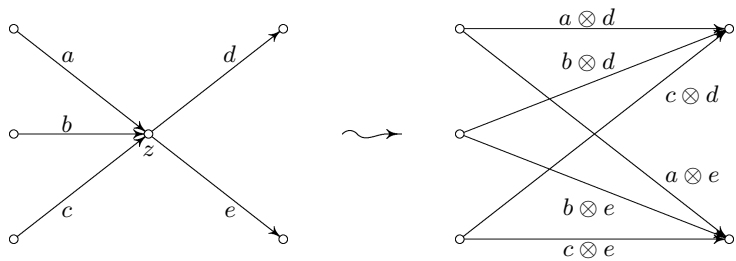
V tomto oddíle budeme předpokládat, že nás zajímají hodnoty spojení $s(x, y)$, přičemž do výchozího vrcholu x nevede žádná hrana (tedy ani smyčka). Tento předpoklad není nijak omezující, v případě potřeby lze ke grafu přidat umělý výchozí vrchol x' a hranu (x', x) o hodnotě $\mathbf{1}$. Namísto hodnot $s(x, y)$ pak budeme počítat hodnoty $s(x', y)$, které jsou stejné (tj. $s(x', y) = s(x, y)$) a přitom splňují náš předpoklad.

7.4.1 Sloučení rovnoběžných hran a jejich náhrada jedinou hranou je tou nejjednodušší elementární úpravou. Hodnota výsledné hrany je rovna součtu $\oplus$ hodnot nahrazovaných hran. Viz též obrázek 7.2 vlevo nahoře. Je-li v nějakém vrcholu několik smyček, lze je stejným způsobem nahradit jedinou smyčkou.

Slučování rovnoběžných hran a vícenásobných smyček se typicky provádí na začátku výpočtu a pak po jakékoli jiné elementární úpravě grafu tak, aby výsledkem byl prostý graf.

7.4.2 Eliminace vrcholu. Označme z vrchol, který eliminujeme (odstraňujeme). Spolu s vrcholem z odstraníme z grafu také všechny hrany s ním incidentní. V okolí odstraněného vrcholu pak přidáme hrany, které lze chápat jako „objížďku“ pro sledy, které původně procházely vrcholem z. Těmito přidanými hranami zajistíme, aby hodnoty spojení mezi všemi zbylými vrcholy zůstaly zachovány.

Pokud ve vrcholu z není smyčka, provádíme jeho eliminaci podle obrázku 7.3.

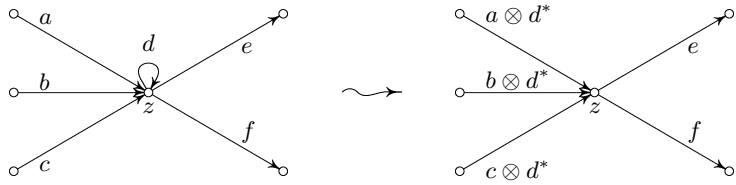


Obrázek 7.3: Eliminace vrcholu.

Je-li ve vrcholu z smyčka, je třeba ji nejprve odstranit postupem uvedeným dále v 7.4.3. Alternativně lze také obě tyto úpravy (eliminaci smyčky a vrcholu) udělat i naráz v jednom složitějším kroku, jak je naznačeno na obrázku 7.2 vpravo dole.

Za zmínku stojí jeden speciální případ. Pokud nás nezajímá hodnota spojení do nějakého vrcholu z, ze kterého nevychází žádná hrana, můžeme tento vrchol eliminovat obzvlášť jednoduše: prostě vynecháme vrchol z a všechny hrany, které jsou s ním incidentní.

7.4.3 Eliminace smyčky je nutná ve dvou situacích: buď jde o přípravu na eliminaci vrcholu, anebo v závěrečné fázi výpočtu odstraňujeme smyčku v cílovém vrcholu, do kterého chceme znát hodnotu spojení.



Obrázek 7.4: Eliminace smyčky ve vrcholu.

Eliminujeme-li smyčku o hodnotě q ve vrcholu z, pak hodnotu opakovaných průchodů smyčkou nahradíme činitelem q^*. Tímto činitelem vynásobíme zprava hodnoty všech hran, které ve vrcholu z končí. Viz obrázek 7.4.

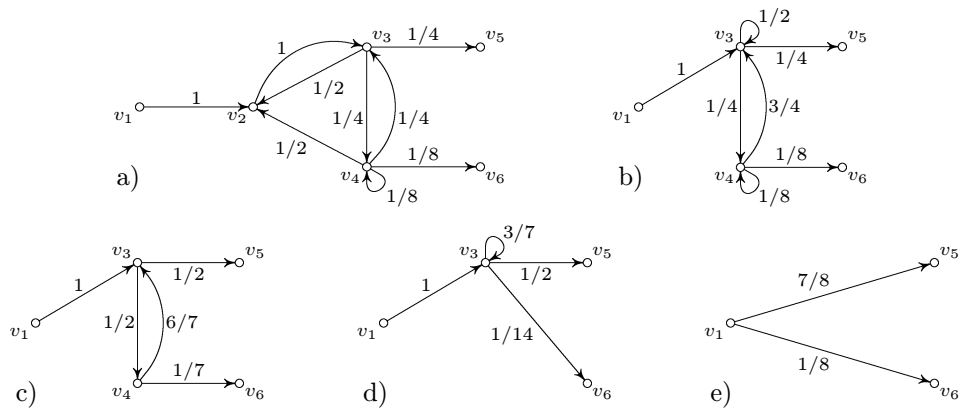
7.4.4 Alternativní eliminace smyčky. Ve vrcholech, které budeme (ihned nebo později) z grafu eliminovat, lze eliminaci smyčky dělat i jinak: činitelem q^* můžeme vynásobit zleva hodnoty všech hran, které z vrcholu z vystupují namísto hran, které do z vstupují. Snadno lze ověřit, že po eliminaci vrcholu z už bude lhostejné, zda jsme činitelem q^* násobili zprava hodnoty hran z $E^-(z)$ nebo zleva hodnoty hran z $E^+(z)$.

Pozor, předpoklad, že vrchol z někdy později vyloučíme, je podstatný. Zajímá-li nás totiž hodnota spojení $s(x,z)$ do vrcholu z, pak musíme činitelem q^* násobit hrany, které v z končí, protože v opačném případě bychom činitel q^* do hodnoty spojení $s(x,z)$ vůbec nezapočítali a dostali bychom chybný výsledek.

Tento alternativní způsob eliminace smyčky má však také jednu výhodu, která se týká úloh o náhodné procházce: Pak totiž eliminace smyčky zachovává tu vlastnost grafu, že hrany, které vycházejí z vrcholu z, mají součet pravděpodobností roven jedné. Ostatní elementární úpravy tuto vlastnost také zachovávají. Pracujeme tedy stále s grafy, ve kterých platí součty pravděpodobností, a ve kterých se tedy lze náhodně procházet. Tato výhoda však platí jen do chvíle, kdy potřebujeme eliminovat smyčku v cílovém vrcholu z, do kterého chceme znát hodnotu spojení $s(x,z)$. Takovou smyčku totiž nelze eliminovat jinak než podle 7.4.3.

7.4.5 Příklad: Náhodná procházka – pravděpodobnost dosažení vrcholu. Pomocí elementárních úprav grafu řešme úlohu o náhodné procházce (viz 7.2.9) v grafu na obrázku 7.5a. Za výchozí vrchol považujeme vrchol v_1.

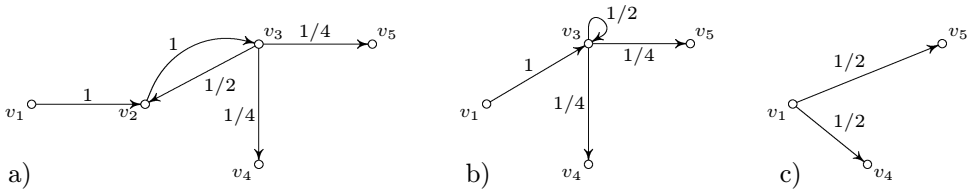
Na prvý pohled je zřejmé, že pravděpodobnosti dosažení vrcholů v_2 a v_3 jsou rovny jedné. Pravděpodobnosti dosažení vrcholů v_5 a v_6 lze určit elementárními



Obrázek 7.5: Řešení příkladu 7.4.5 – prvá část.

úpravami původního grafu, viz obr. 7.5, kde postupně byly provedeny tyto úpravy: eliminace vrcholu v_2, eliminace smyček ve vrcholech v_3 a v_4, eliminace vrcholu v_4 a konečně eliminace vrcholu v_3. Eliminace smyček jsme prováděli alternativním způsobem 7.4.4.

Pro výpočet pravděpodobnosti dosažení vrcholu v_4 odstraníme všechny hrany z $E^+(v_4)$ a úpravy provedeme s takto zmenšeným grafem, viz obr. 7.6.

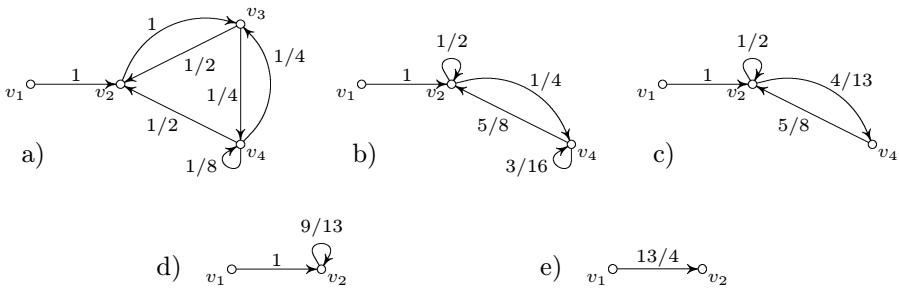


Obrázek 7.6: Řešení příkladu 7.4.5 – druhá část.

Výsledná pravděpodobnost dosažení vrcholu v_4 je rovna $1/2$. Všimněte si, že hodnota spojení $s(v_1, v_5)$ nás v tomto pomocném zmenšeném grafu nezajímá. Vrchol v_5 jsme mohli eliminovat již na začátku výpočtu.

7.4.6 Příklad: Náhodná procházka – počet průchodů vrcholem. V grafu z předchozího příkladu 7.4.5 zjistíme střední počet průchodů vrcholem v_2 (viz 7.2.10). Jinými slovy, v grafu z obrázku 7.5a zjistíme hodnotu spojení $s(v_1, v_2)$.

Výpočet je na obrázku 7.7, kde postupně byly provedeny tyto úpravy: eliminace vrcholů v_5 a v_6, eliminace vrcholu v_3, eliminace smyčky ve vrcholu v_4 (způsobem 7.4.3), eliminace vrcholu v_4 a konečně eliminace smyčky ve vrcholu v_2.



Obrázek 7.7: Řešení příkladu 7.4.6.

7.4.7 Cvičení. V příkladu 7.4.6 vypočtěte hodnoty spojení do ostatních vrcholů.

7.4.8 Cvičení. Ověřte na příkladě (třeba na obrázku 7.5), že obě metody eliminace smyčky dají po eliminaci příslušného vrcholu stejný výsledek. Podobně ověřte, že výsledky příkladů 7.4.5 a 7.4.6 nezávisí na pořadí eliminace vrcholů.

7.5 Signální toky

Teorie signálních toků se používá zejména v teorii regulace při studiu zpětné vazby, lze ji však použít i v jiných oblastech. Omezíme se na nejjednodušší fakta, podrobnosti lze nalézt např. v knihách [KŠCh89, SwTh81].

7.5.1 Grafy signálních toků. Mějme orientovaný graf, jehož hrany jsou ohodnoceny reálnými čísly, která nazýváme *přenosem hrany*. Předpokládejme, že grafem proudí blíže nespecifikovaný signál podle těchto pravidel:

1. Orientace hrany určuje směr průchodu signálu hranou.
2. Má-li hrana přenos h a vstupuje-li do ní z jejího počátečního vrcholu signál o velikosti x, pak z této hrany do jejího koncového vrcholu vystupuje signál o velikosti hx.
3. Signály vstupující z různých hran do vrcholu se sčítají, a vytvářejí tak *signál vrcholu*. Tento signál je z vrcholu vysílán v téže nezmenšené velikosti do všech hran, které z něj vycházejí.

Jestliže do některého vrcholu nevstupuje žádná hrana, pak signál tohoto vrcholu nezávisí na signálech ostatních vrcholů, ale může ovlivňovat signál v okolních vrcholech. Takový vrchol se nazývá *zdrojem signálu*, ostatní vrcholy nazýváme *závislými*. V grafu může být jeden nebo několik zdrojů signálu.

Úkolem je zjistit, jak závisí velikost signálů jednotlivých vrcholů grafu na velikostech signálů ve zdrojích.

Tuto úlohu lze řešit sestavením soustavy lineárních algebraických rovnic pro neznámé velikosti signálu a řešením této soustavy běžnými metodami lineární algebry. Pro každý závislý vrchol máme jednu rovnici, která říká, že signál vrcholu je roven součtu signálů vstupujících do něj ze všech hran, které v tomto vrcholu končí.

Např. pro graf na obr. 7.5a, v němž ohodnocení hran budeme chápat jako přenosy hran, bude mít soustava rovnic tvar

$$\begin{aligned} x_2 &= x_1 + \tfrac{1}{2}x_3 + \tfrac{1}{2}x_4\,, \\ x_3 &= x_2 + \tfrac{1}{4}x_4\,, \\ x_4 &= \tfrac{1}{4}x_3 + \tfrac{1}{8}x_4\,, \\ x_5 &= \tfrac{1}{4}x_3\,, \\ x_6 &= \tfrac{1}{8}x_4\,. \end{aligned} \tag{7.4}$$

Proměnnou x_1 zde pokládáme za známý parametr (odpovídá zdroji signálu), ostatní proměnné pokládáme za neznámé, které chceme vypočítat (odpovídají závislým vrcholům).

Tuto úlohu lze však řešit i grafovými prostředky, což je výhodné zejména tehdy, požadujeme-li hodnoty pouze některých proměnných a chceme-li znát souvislosti mezi velikostmi signálů v různých vrcholech. Další výhodou je názornost.

Snadno lze nahlédnout, že vlivy jednotlivých zdrojů signálu na signály v ostatních vrcholech se sčítají. Proto je vhodné uvažovat vliv každého zdroje zvlášť, a to tak, že ve vybraném zdroji uvažujeme jednotkový signál a v ostatních zdrojích nulový. Velikost signálu x_j, kterou v závislém vrcholu j vyvolá jednotkový signál ve zdroji i, nazýváme *přenosem grafu z vrcholu i do vrcholu j*. Jedná se vlastně o poměr signálů x_j/x_i, jsou-li signály v ostatních zdrojích nulové.

Pro výpočet přenosu grafu lze použít několik metod.

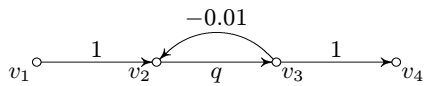
Jednou z nich je použití elementárních úprav grafu popsaných v 7.4, kde operace $\oplus$ a $\otimes$ chápeme jako běžné sečítání a násobení a operaci $*$ definujeme předpisem $x^* = 1/(1-x)$ pro všechna $x \in (-1, 1)$. Eliminace vrcholu v podstatě není ničím jiným než eliminací proměnné ze soustavy rovnic. Obrázek 7.5 může sloužit jako příklad výpočtu přenosu grafu z vrcholu v_1 do vrcholů v_5 a v_6.

Jinou možnost představuje tzv. Masonovo pravidlo popsané dále v 7.5.4. Pro výpočet přenosu lze také použít algoritmus 7.3.1, v němž příkaz (7.3) bude mít tvar

$$S(i,j) := S(i,j) + \frac{S(i,k)S(k,j)}{1 - S(k,k)} \ .$$

7.5.2 Příklad. Pro graf na obr. 7.8 vypočtěme přenos z vrcholu v_1 do v_4. Elementárními úpravami grafu dostaneme přenos $s(v_1, v_4) = q/(1 - q(-0.01)) = = 100q/(100 + q)$. Pro velká q pak máme

$$\lim_{q\to\infty} s(v_1, v_4) = \lim_{q\to\infty} \frac{100q}{100+q} = 100 \ .$$



Obrázek 7.8: Graf k příkladu 7.5.2.

Mimochodem, tento trik se používá při konstrukci zesilovačů, které mají mít předem dané zesílení z: vezmeme zesilovač s hodně velkým zesílením q a z jeho výstupu na vstup zavedeme zápornou zpětnou vazbu s přenosem $-1/z$ (což je technicky snadné). Je-li q dost velké, pak celkové zesílení je v podstatě určeno jen přenosem zpětné vazby. Záporná zpětná vazba má na zesílení stabilizující účinek.

7.5.3 Příklad: náhodná procházka – počet průchodů vrcholem. Vraťme se k úloze o náhodné procházce 7.2.10 a k příkladu 7.4.6.

Na příkladě grafu z obrázku 7.5a ukážeme, že hodnoty spojení z výchozího vrcholu v_1 do ostatních vrcholů můžeme pokládat také za přenosy v grafu signálních toků. Hodnotu spojení z vrcholu v_1 do vrcholu v_i označme $x_i = s(v_1, v_i)$. Každý sled končící ve vrcholu v_i musí mít jako předposlední vrchol některého předchůdce vrcholu v_i. Průměrný počet průchodů vrcholem v_i tedy je součtem průměrných počtů průchodů těmito předchůdci násobených pravděpodobnostmi příslušných hran.

Například pro vrchol v_2 dostaneme rovnici $x_2 = x_1 + \frac{1}{2}x_3 + \frac{1}{2}x_4$. Soustava rovnic, kterou takto získáme, je totožná se soustavou (7.4). Položíme-li $x_1 = 1$, můžeme pravděpodobnosti hran pokládat za přenosy hran a hodnoty spojení z vrcholu v_1 do ostatních vrcholů za přenosy grafu mezi těmito vrcholy.

7.5.4 Věta: Masonovo pravidlo. Nechť hrany grafu G jsou ohodnoceny jejich přenosy. Nechť x, y jsou libovolné vrcholy, nechť n je počet všech různých cest z vrcholu x do vrcholu y. Označme tyto cesty $p_1, p_2, \ldots, p_n$ a přenosy těchto cest (tj. součiny přenosů jejich hran) $P_1, P_2, \ldots, P_n$. Podobně označme $c_1, c_2, \ldots, c_m$ všechny různé cykly v grafu a $C_1, C_2, \ldots, C_m$ přenosy těchto cyklů. Označme dále

$$\begin{aligned} L \;=\; & 1 - (C_1 + C_2 + \ldots + C_m) + \\ & + (C_1C_2 + C_1C_3 + \ldots + C_1C_m + C_2C_3 + \ldots + C_{m-1}C_m) - \\ & - (C_1C_2C_3 + C_1C_2C_4 + \ldots + C_1C_3C_4 + \ldots + C_{m-2}C_{m-1}C_m) + \\ & + \ldots, \end{aligned}$$

přičemž vynecháváme vždy ty sčítance, kde z příslušných cyklů některé dva mají společný vrchol.

Konečně označme L_i výraz, který je definován stejně jako L, ale navíc v něm vynecháváme také ty sčítance, kde některý z cyklů má společný vrchol s cestou p_i.

Potom přenos grafu z vrcholu x do vrcholu y je roven

$$\frac{\sum_{i=1}^{n} P_i L_i}{L}\,.$$

Poznámka: Je-li $x = y$, pak cestou z vrcholu x do téhož vrcholu y rozumíme jednak triviální cestu, která nemá žádnou hranu a jejíž hodnota je 1, jednak cykly, které procházejí vrcholem $x = y$. Triviální sled zde chápeme jako by neobsahoval žádnou hranu *ani vrchol*. Proto, je-li p_1 triviální cesta, pak $P_1L_1 = L$, neboť žádný cyklus nemá společný vrchol s cestou p_1.

7.5.5 Příklad. Vypočtěme pomocí Masonova pravidla přenos grafu na obrázku 7.5a z vrcholu v_1 do vrcholu v_5. Z v_1 do v_5 vede jediná cesta určená vrcholy v_1, v_2, v_3, v_5. Cykly v tomto grafu jsou celkem čtyři, jsou určeny vrcholy

$$v_2v_3,\;\; v_3v_4,\;\; v_2v_3v_4 \;\text{ a smyčka ve vrcholu } v_4.$$

Pouze jediná dvojice cyklů je vrcholově disjunktní (nemá společný vrchol): v_2v_3 a v_4. Podle Masonova pravidla je přenos z vrcholu v_1 do vrcholu v_5 roven

$$s(v_1, v_5) = \frac{1 \cdot 1 \cdot \frac{1}{4} \cdot (1 - \frac{1}{8})}{1 - 1 \cdot \frac{1}{2} - \frac{1}{4} \cdot \frac{1}{4} - 1 \cdot \frac{1}{4} \cdot \frac{1}{2} - \frac{1}{8} + 1 \cdot \frac{1}{2} \cdot \frac{1}{8}} = \frac{\frac{1}{4} \cdot \frac{7}{8}}{\frac{1}{4}} = \frac{7}{8}\,.$$

7.5.6 Cvičení. V grafu na obrázku 7.5a vypočtěte hodnoty spojení z vrcholu v_1 do ostatních vrcholů pomocí Masonova pravidla a také řešením soustavy rovnic 7.4. Výsledky musí být stejné jako ve cvičení 7.4.7.

Kapitola 8

Toky v síti

Toky v sítích patří k nejčastěji aplikovaným částem teorie grafů. Pomocí toků lze modelovat řadu praktických situací, další aplikace jsou v jiných částech teorie grafů a kombinatoriky. V této kapitole se budeme zabývat zejména optimalizačními úlohami o tocích v síti a úlohami, kde jsou velikosti toků v zadání úlohy omezeny. Použitím toků v síti v elektrotechnice se budeme zabývat v kapitole 15, str. 233.

Tuto kapitolu lze studovat do značné míry nezávisle na předchozím textu.

8.1 Základní pojmy a úlohy

8.1.1 Tok a cirkulace. Mějme orientovaný graf G. *Tokem v síti* G nazýváme takové ohodnocení hran reálnými čísly $f : E(G) \to \mathbb{R}$, které pro každý vrchol v splňuje *Kirchhoffův zákon*

$$\sum_{e \in E^+(v)} f(e) \;=\; \sum_{e \in E^-(v)} f(e)\,.$$

Graf si můžeme představovat jako soustavu potrubí, kterým proudí nějaká kapalina. Tok v každé hraně je ustálený a beze ztrát, tj. tok, který do hrany vtéká je roven toku, který z hrany vytéká. Kirchhoffův zákon pak vlastně říká, že kolik tekutiny do vrcholu přitéká, tolik jí z vrcholu také odtéká. Orientace hrany určuje směr proudění; záporná velikost toku je možná a znamená proudění proti směru hrany.

Jsou studovány zejména dvě varianty toků, které se liší množinou vrcholů, pro které je požadována platnost Kirchhoffova zákona: tzv. *cirkulace* splňuje Kirchhoffův zákon pro všechny vrcholy, zatímco tzv. *tok od zdroje ke spotřebiči* splňuje Kirchhoffův zákon pro všechny vrcholy kromě dvou. Tyto vrcholy nazýváme *zdrojem* a *spotřebičem*. Neplatnost Kirchhoffova zákona ve zdroji a spotřebiči chápeme tak, že ve zdroji tok vzniká a ve spotřebiči stejné množství toku zaniká.

Tok od zdroje ke spotřebiči můžeme snadno převést na cirkulaci přidáním tzv. *návratové hrany* ze spotřebiče ke zdroji.

8.1.2 Přípustný tok. Velikost toku $f(e)$ v jednotlivých hranách bývá ve většině úloh omezena na předem daný uzavřený interval $f(e) \in \langle l(e), c(e)\rangle$. Číslo $c(e)$ nazýváme *kapacitou hrany* e nebo též *horním omezením toku v hraně* e. Číslo $l(e)$ nazýváme *dolním omezením toku v hraně* e. Tok, který splňuje nerovnosti $l(e) \leq \leq f(e) \leq c(e)$ pro všechny hrany grafu, nazýváme *přípustným tokem.*

V řadě úloh jsou všechna dolní omezení $l(e)$ rovna nule. V takovém případě je nulový tok (každou hranou teče nulové množství) tokem přípustným. V řadě dalších úloh však je nenulové dolní omezení podstatnou součástí úlohy, zde pak není a priori jasné, zda přípustný tok vůbec existuje. Existencí a hledáním přípustné cirkulace se budeme zabývat v 8.4.

Je-li v síti dán zdroj a spotřebič a jsou-li dolní omezení toku ve všech hranách nulová, nazýváme tuto síť *transportní sítí.*

8.1.3 Velikost toku od zdroje ke spotřebiči definujeme jako množství toku, které ve zdroji vzniká, tj. jako rozdíl

$$F(f) \;=\; \sum_{e\in E^+(z)} f(e) \;-\; \sum_{e\in E^-(z)} f(e)\,.$$

8.1.4 Velikost toku přes řez. Připomeňme (viz 1.3.1, str. 16), že je-li A množina vrcholů, pak řez určený množinou A (značíme jej $W(A)$) je množina hran, jejichž jeden vrchol leží v množině A a druhý nikoli. Pokud množina A obsahuje zdroj a neobsahuje spotřebič, řekneme, že řez $W(A)$ *odděluje zdroj a spotřebič.*

Velikost toku přes řez určený množinou A definujeme jako množství toku, které hranami řezu teče z množiny A ven, zmenšené o množství, které se hranami řezu zpět do množiny A vrací, tj.

$$F_A(f) \;=\; \sum_{e\in W^+(A)} f(e) \;-\; \sum_{e\in W^-(A)} f(e)\,.$$

Poznamenejme, že velikost toku od zdroje ke spotřebiči jsme vlastně definovali jako velikost toku přes speciální řez určený množinou $A = \{z\}$. Z následujícího tvrzení 8.1.5 plyne, že jsme pro tento účel mohli použít jakýkoli řez oddělující zdroj a spotřebič.

8.1.5 Věta. Nechť f je libovolný tok od zdroje z ke spotřebiči s a nechť $W(A)$ je libovolný řez, který odděluje zdroj z a spotřebič s. Potom platí $F_A(f) = F(f)$, tj. velikost $F(f)$ toku od zdroje ke spotřebiči (definovaná podle 8.1.3) je rovna velikosti $F_A(f)$ toku přes řez $W(A)$ (definované podle 8.1.4).

Důkaz je přenechán čtenáři jako cvičení 8.1.6. □

Důsledek: Je-li tok f cirkulací, pak velikost toku f přes jakýkoli řez je nulová. Cirkulaci totiž můžeme pokládat za speciální případ toku od zdroje ke spotřebiči, přičemž jako zdroj a spotřebič můžeme vzít kterékoli dva vrcholy.

8.1.6 Cvičení. Dokažte větu 8.1.5.

POZNÁMKA: Povrchní čtenář by se mohl domnívat, že zde není co dokazovat, neboť jde o triviální důsledek „zákona zachování". Matematika (a zde konkrétně teorie grafů) se ovšem nechce spoléhat na fyzikální analogie. To jednak ze zásady (chce být na fyzice nezávislá), jednak proto, že fyzikální analogie někdy zákeřně klamou (vzpomeňte třeba na trojúhelníkovou nerovnost, viz 6.1.7, str. 90).

NÁVOD: Každou množinu A, takovou, že $z \in A$, $s \notin A$, lze získat z jednoprvkové množiny $\{z\}$ postupným přidáváním vrcholů. Stačí tedy dokázat, že když $v \notin A$ a $v \neq s$, pak velikosti toků přes řezy určené množinami A a $A' = A \cup \{v\}$ jsou stejné.

8.1.7 Úloha o maximálním toku. Klasická formulace úlohy je tato: Je dána transportní síť, tj. orientovaný graf G, dva jeho vrcholy, zdroj z a spotřebič s, a kladné kapacity hran $c(e)$. Úkolem je najít maximální přípustný tok od zdroje ke spotřebiči, tj. přípustný tok f, který má největší velikost $F(f)$.

Obecnější formulace připouští i nenulová dolní omezení toku v hranách. Touto obecnější úlohou se budeme zabývat v oddíle 8.3, str. 141.

8.1.8 Kapacita řezu. Kapacitu řezu $W(A)$ definujeme jako číslo

$$C(A) \;=\; \sum_{e \in W^+(A)} c(e) \;-\; \sum_{e \in W^-(A)} l(e)\,.$$

Význam kapacity řezu je ten, že žádný přípustný tok od zdroje ke spotřebiči nemůže mít větší velikost, než je kapacita kteréhokoli řezu, který odděluje zdroj a spotřebič. Kapacita řezu tedy poskytuje dílčí informaci o možných velikostech toku.

8.1.9 Věta. Pro libovolný přípustný tok f a pro libovolný řez $W(A)$ oddělující zdroj a spotřebič platí $F(f) \leq C(A)$.

DŮKAZ: Velikost toku od zdroje ke spotřebiči $F(f)$ je rovna velikosti $F_A(f)$ toku přes řez $W(A)$. Nerovnost $F_A(f) \leq C(A)$ pak plyne z faktu, že tok je přípustný. □

8.1.10 Úloha o minimálním řezu. Je dán orientovaný graf G, zdroj, spotřebič a horní a dolní omezení toku. Hledáme řez, který odděluje zdroj a spotřebič a který má nejmenší kapacitu.

Úloha o minimálním řezu má své vlastní důležité aplikace (např. hledání úzkých míst v silniční síti, viz 8.2.2, nebo úloha o zkracování činností v síťovém grafu 8.2.7). Hlavní význam úlohy o minimálním řezu však souvisí s faktem, že velikost maximálního toku je rovna kapacitě minimálního řezu (viz věta 8.3.8).

Minimální řez lze najít algoritmem pro maximální tok 8.3.6.

8.1.11 Nejlevnější tok. Předpokládejme, že každá hrana grafu je kromě omezení toku $l(e)$ a $c(e)$ ohodnocena také číslem $a(e)$, které nazýváme *jednotkovou cenou toku v hraně* e.

Teče-li hranou e tok $f(e)$, pak součin $f(e)a(e)$ nazýváme *cenou toku v hraně* e. Konečně *cenu toku v síti* definujeme jako součet cen toků v jednotlivých hranách, tedy

$$\sum_{e \in E(G)} f(e)a(e) \ .$$

Úkolem je najít nejlevnější přípustnou cirkulaci. Poznamenejme, že jednotkové ceny toku mohou být i záporné.

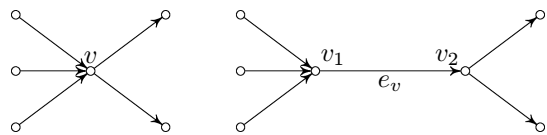
Tato úloha je poměrně obecná, zahrnuje v sobě nejen hledání (jakékoli) přípustné cirkulace, ale také hledání maximálního toku od zdroje ke spotřebiči: stačí ke grafu přidat návratovou hranu vedoucí ze spotřebiče ke zdroji, stanovit její kapacitu dost velkou, aby neomezovala tok, a jednotkovou cenu v této hraně zvolit zápornou (např. -1). Ostatní hrany budou mít jednotkové ceny nulové. Pak cirkulace s nejmenší (zápornou) cenou bude maximalizovat tok v návratové hraně.

Hledáním nejlevnější přípustné cirkulace se budeme zabývat v 8.6 a v 8.7.

8.1.12 Několik zdrojů a spotřebičů. Je-li v grafu několik zdrojů (vrcholů, kde smí tok vznikat) a spotřebičů (kde smí zanikat), lze takovou úlohu převést na standardní úlohu s jediným zdrojem a jediným spotřebičem.

K síti přidáme dva nové vrcholy, umělý zdroj z a umělý spotřebič s. Ke každému z původních zdrojů vedeme novou hranu z umělého zdroje. Podobně z každého původního spotřebiče vedeme novou hranu do umělého spotřebiče. Kapacity přidaných hran přitom stanovíme tak, aby neomezovaly tok (popř. omezovaly, ale námi zamýšleným způsobem).

8.1.13 Omezená propustnost vrcholů. Pro některé vrcholy může být v úloze předepsáno maximální i minimální množství toku, které smí vrcholem protékat. I takovou úlohu lze převést na standardní úlohu, v níž propustnosti vrcholů nejsou omezeny.



Obrázek 8.1: Převod omezené propustnosti vrcholů na omezenou propustnost hran.

Nejprve se zabývejme případem, kdy v původní úloze jsou přípustné pouze nezáporné toky (tj. pro všechny hrany platí $l(e) \geq 0$). Graf upravíme takto: Každý vrchol v, který má omezenou propustnost, nahradíme dvojicí vrcholů v_1, v_2 a orientovanou hranou e_v z v_1 do v_2. Hrany, které dosud končily ve v, přesměrujeme, aby končily ve v_1. Podobně hrany, které ve v začínaly, nahradíme hranami, které budou začínat ve v_2 (viz obr. 8.1). Omezení toků (případně další ohodnocení) v těchto hranách ponecháme. Tok, který dosud protékal vrcholem v, nyní musí

protékat vrcholy v_1, v_2 a hranou e_v. Původní omezení pro tok vrcholem v tedy můžeme snadno vyjádřit pomocí omezení pro tok hranou e_v.

Pokud v původní úloze jsou přípustné i záporné toky, je třeba nejprve upravit graf podle 8.1.15 a pak teprve zdvojit vrcholy uvedeným způsobem.

8.1.14 Cvičení. Proč nelze výše uvedený postup 8.1.13 použít i v úlohách, kde jsou přípustné záporné toky?

8.1.15 Převod na nezáporné toky. Jsou-li v původním grafu přípustné i záporné toky, tj. jestliže pro některou hranu platí $l(e) < 0$, lze tuto úlohu převést na úlohu s nezápornými toky takto:

Je-li $l(e) \leq c(e) \leq 0$, obrátíme orientaci hrany e, původní dolní omezení toku nahradíme číslem $-c(e)$ a horní omezení číslem $-l(e)$. Pokud byl původní tok hranou záporný, změní se po úpravě jeho znaménko.

Pokud $l(e) < 0 < c(e)$, přidáme ke hraně e opačně orientovanou hranu e'. Kapacitu této nové hrany zvolíme $-l(e)$ a u obou hran změníme dolní omezení toku na nulu. Pokud byl původní tok hranou e záporný, poteče nyní tento tok novou hranou e'.

8.1.16 Toky v neorientovaných grafech. Úlohy o tocích lze samozřejmě formulovat a řešit i v neorientovaných grafech. Zde zpravidla bývá předepsána jen kapacita hrany, dolní omezení toku bývají nulová. Směr toku hranou je libovolný, ale musíme jej určit.

Tyto úlohy lze snadno převést na obdobné úlohy v grafech orientovaných. Jednou z možností je zvolit (jakkoli) orientaci hran a zavést dolní omezení toku v hranách tak, aby $l(e) = -c(e)$. Druhou možností je nahradit neorientovanou hranu dvojicí opačně orientovaných hran se stejnou kapacitou.

Pokud v původní úloze šlo o nejlevnější tok a má-li být jednotková cena toku v hraně v obou směrech stejná, je vhodný pouze druhý způsob.

8.1.17 Cvičení. V úloze o nejlevnějším toku 8.1.11 jsme předpokládali, že cena toku v hraně závisí lineárně na velikosti toku. Má-li být tato závislost konvexní, spojitá a po částech lineární, lze takovou hranu nahradit několika rovnoběžnými hranami s běžnou lineární závislostí ceny na toku. Náhrada je založena na faktu, že nejlevnější tok bude přednostně procházet nejlevnější hranou, teprve po jejím nasycení druhou nejlevnější z rovnoběžných hran atd. Promyslete detaily na konkrétním příkladě i obecně.

8.1.18 Celočíselnost toků. Popsané úlohy o tocích a algoritmy pro jejich řešení mají příjemnou vlastnost: jsou-li omezení toku celočíselná, lze celý výpočet uskutečnit v oboru celých čísel. Proto se ve zbytku kapitoly omezíme na celočíselné toky. U praktických úloh to nevadí: reálná čísla lze nahradit vhodnými zlomky, ty převést na společného jmenovatele a pak tímto jmenovatelem vynásobit. Výsledný (celočíselný) tok pak ovšem musíme tímto jmenovatelem zase vydělit.

8.2 Aplikace toků v sítích

Aplikace uvedené v 8.2.1 až 8.2.5 lze pokládat za standardní. Další aplikace jsou uvedeny jako ukázky méně triviálních způsobů modelování různých problémů grafem.

8.2.1 Propustnost silniční sítě mezi danými místy lze zjišťovat triviální aplikací úlohy o maximálním toku v síti od zdroje ke spotřebiči v neorientovaném grafu (viz 8.1.16).

8.2.2 Stupně souvislosti. Hranový stupeň souvislosti (viz 4.4.1, str. 67) lze zjistit opakovaným výpočtem maximálního toku od zdroje ke spotřebiči, a to pro všechny dvojice zdroj–spotřebič, přičemž kapacity všech hran jsou jednotkové. Připomeňme, že graf je neorientovaný, je tedy třeba postupovat podle 8.1.16.

Nalezneme-li tok ze zdroje z do spotřebiče s o velikosti k, pak hrany s nenulovým tokem vytvářejí celkem k hranově disjunktních cest z vrcholu z do s. Minimální řez (který najdeme zároveň s tokem a který obsahuje podle věty 8.3.8 přesně k hran) přitom tvoří minimální množinu hran, po jejichž odstranění nebude existovat cesta ze z do s. Odtud vyplývá důkaz věty 4.4.4 o hranové k-souvislosti.

Podobně lze zjistit i vrcholový stupeň souvislosti (viz 4.4.2) opakovaným výpočtem maximálního toku v grafu, v němž mají jednotkové kapacity nejen všechny hrany, ale i všechny vrcholy (viz 8.1.13).

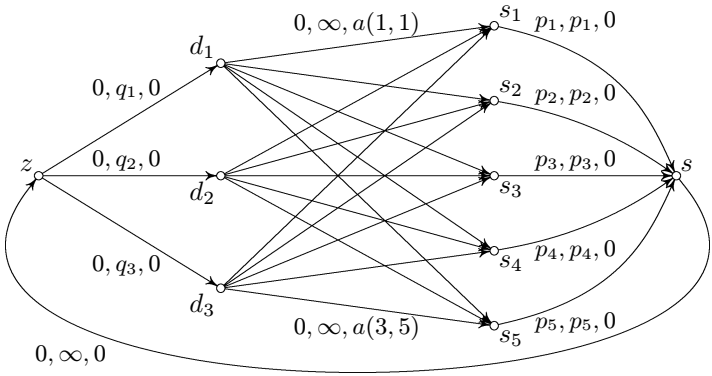
Také zde bude tok o velikosti k určovat soustavu k vrcholově disjunktních cest ze zdroje do spotřebiče. Pomocí minimálního řezu i zde určíme minimální množinu vrcholů, jejichž odstraněním přerušíme všechny cesty mezi zdrojem a spotřebičem. (Pokud ovšem graf byl úplný, pak taková množina neexistuje.) Odtud také vyplývá důkaz Mengerovy věty 4.4.3, str. 68.

8.2.3 Dopravní úlohy. Uvažujme dopravní síť s několika dodavateli nějakého (téhož) materiálu a s několika spotřebiteli. Síť si můžeme představit jako orientovaný graf. Hrany grafu odpovídají úsekům silnic, železnic, linkám lodní dopravy apod. Některé vrcholy grafu odpovídají dodavatelům a spotřebitelům, další vrcholy pak mohou představovat křižovatky silnic, železniční uzly, překladiště, přístavy apod. Předpokládejme, že jsou známy propustnosti (kapacity) jednotlivých hran (mohou být i neomezené) a také ceny za dopravu jednotkového množství materiálu po trase představované hranou. Úkolem je najít nejlevnější přípustný způsob dopravy materiálu od dodavatelů ke spotřebitelům.

Jde zřejmě o nalezení nejlevnějšího toku. K síti přidáme umělý zdroj a umělý spotřebič a spojíme je hranami se skutečnými zdroji a spotřebiči podobně jako v 8.1.12. Pomocí horního a dolního omezení toku v takto přidaných hranách lze snadno vyjádřit velikosti požadavků spotřebitelů a kapacity dodavatelů. Nestačí-li kapacity dodavatelů k uspokojení poptávky spotřebitelů a mají-li spotřebitelé různé priority, můžeme tyto priority vyjádřit pomocí jednotkových cen (i záporných) toku v hranách od spotřebitelů k umělému spotřebiči. Chceme-li k řešení použít algoritmus pro hledání nejlevnější přípustné cirkulace, musíme ještě přidat návratovou

hranu z umělého spotřebiče do umělého zdroje (viz obr. 8.2) anebo (elegantněji) můžeme umělý zdroj a umělý spotřebič ztotožnit.

8.2.4 Klasická dopravní úloha (též Hitchcockova). Je to speciální případ dopravní úlohy ve smyslu 8.2.3. Graf klasické dopravní úlohy je úplným bipartitním grafem, hrany tedy vedou od každého dodavatele přímo ke všem spotřebitelům a kapacity všech těchto hran jsou neomezené. Příklad takového grafu, doplněného o umělý zdroj, umělý spotřebič a návratovou hranu, je na obr. 8.2.



Obrázek 8.2: Síť k řešení dopravní úlohy. Hodnoty jednotlivých hran jsou l, c, a, tj. „dolní omezení, kapacita, jednotková cena“.

Zpravidla se požaduje, aby klasická dopravní úloha byla tzv. *vyvážená*, což znamená, že součet kapacit dodavatelů je roven součtu požadavků spotřebitelů. Pak má každý dodavatel i spotřebitel přesně předepsánu velikost toku, tj. dolní omezení toku je rovno hornímu.

Pro řešení klasické dopravní úlohy existují speciální postupy, které využívají velmi jednoduché struktury grafu a jsou rychlejší než obecný algoritmus pro nejlevnější tok.

8.2.5 Přiřazovací úloha. Klasická formulace je tato: Máme n pracovníků, kterým je třeba přidělit n úkolů, každému jeden. Pro každou dvojici pracovník–úkol známe náklady na vykonání úkolu tímto pracovníkem. Cílem přiřazovací úlohy je přiřadit každému pracovníkovi úkol tak, aby celkové náklady byly minimální.

Přiřazovací úlohu je možno chápat (a řešit) jako speciální případ klasické dopravní úlohy: pracovníci odpovídají dodavatelům, přiřazované úkoly odpovídají spotřebitelům, přičemž kapacity dodavatelů i spotřebitelů jsou jednotkové (jakoby jeden kus zboží).

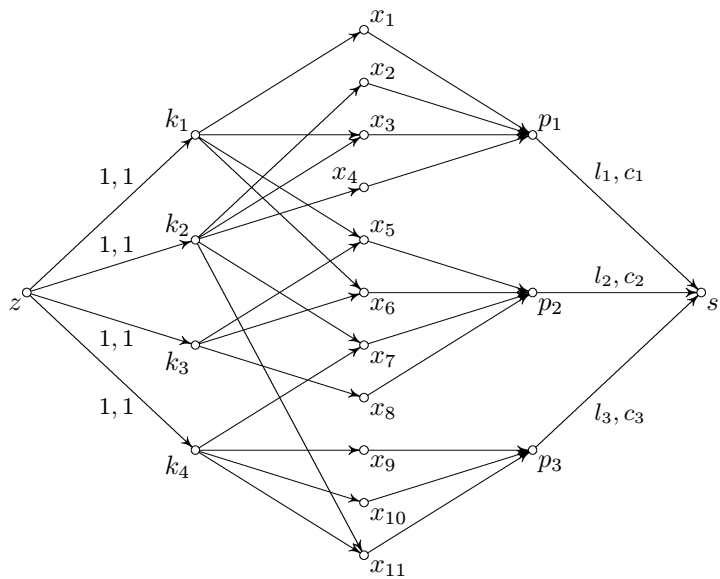
V každé hraně příslušného bipartitního grafu teče buď nulový, nebo jednotkový tok, přičemž jednotkový tok mezi určitým dodavatelem a spotřebitelem odpovídá dvojici pracovník–úkol, kteří jsou si vzájemně přiřazeni.

Přiřazovací úlohu ovšem lze chápat také jako úlohou o tzv. párování, (viz 9.1.7, str. 164), příslušný algoritmus je uveden v 9.4, str. 173.

8.2.6 Výběr reprezentantů. Tuto úlohu lze pokládat za přiřazovací úlohu s dalším omezením.

Mějme n gentlemanů, každý z nich je členem jednoho nebo několika klubů $k_1, \ldots, k_q$ a přesně jedné politické strany $p_1, \ldots, p_r$. Každý klub chce vybrat ze svých členů jednoho reprezentanta. Nikdo však nesmí reprezentovat více než jeden klub. Navíc je třeba dodržet zastoupení politických stran: pro každou stranu je stanoven minimální a maximální počet reprezentantů, kteří jsou členy této strany. Úkolem je najít takovýto výběr reprezentantů anebo prokázat, že neexistuje.

Tuto úlohu lze řešit nalezením přípustného toku v síti, jejíž příklad je na obr. 8.3. Je snadné nahlédnout, že přípustné výběry reprezentantů vzájemně jednoznačně odpovídají přípustným celočíselným tokům v této síti.



Obrázek 8.3: Graf k řešení úlohy o výběru reprezentantů pro čtyři kluby a tři strany. Hrany jsou ohodnoceny omezeními toku. Nejsou-li omezení uvedena, je $l(e) = 0$ a $c(e) = 1$.

8.2.7 Zkracování činností v síťovém grafu. Mějme síťový graf (viz 2.3.2, str. 44), v němž pro každou činnost známe náklady, které by bylo třeba vynaložit (navíc), aby se trvání činnosti zkrátilo o jednu časovou jednotku. Úkolem je zkrátit trvání celého síťového grafu nejlevnějším způsobem.

Má smysl zkracovat pouze činnosti (hrany), které leží na tzv. kritické cestě. Kritických cest ovšem může být v grafu několik. Je třeba zkrátit takovou množinu kritických činností, aby na každé kritické cestě byla zkrácena nějaká činnost. Množina zkrácených činností tedy musí v podgrafu tvořeném kritickými činnostmi tvořit řez, který odděluje počáteční a koncový vrchol grafu. Hledáme takovou množinu s minimálním součtem nákladů.

Hledáme tedy řez s minimálním součtem ohodnocení. K tomu lze použít algoritmus pro hledání (maximálního toku a) minimálního řezu. V roli kapacit hran přitom vystupují náklady na jednotkové zkrácení činností.

Zkrácením některých kritických činností se mohou některé další činnosti stát kritickými. Máme-li tedy zkrátit celý projekt o několik jednotek, musíme výše popsaný výpočet několikrát opakovat, přičemž graf, v němž hledáme minimální řez, bude obsahovat postupně více a více kritických činností.

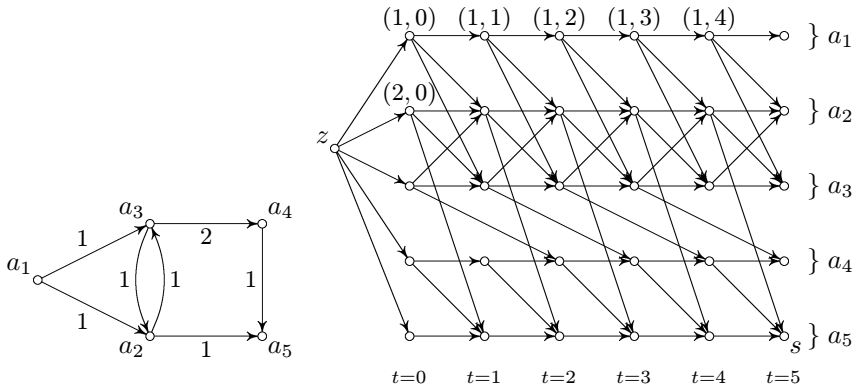
Poznamenejme, že úlohu o zkracování síťového grafu lze řešit také jako tzv. duální úlohu k úloze o nejlevnější cirkulaci pomocí algoritmu Out of Kilter 8.7.6, podrobnosti však přesahují rámec této knížky.

8.2.8 Dynamický tok je tok, který mění svou velikost v závislosti na čase. Ukážeme způsob, jak lze některé úlohy o dynamickém toku převést na vhodnou úlohu o běžném (statickém) toku ve smyslu definice 8.1.1.

Ve městech $a_1, \ldots, a_n$ je k dispozici $q_1, \ldots, q_n$ automobilů, které mají být dopraveny do města a_n během T hodin. Jsou-li města a_i, a_j spojena přímou silnicí, označme $d_{i,j}$ dobu jízdy mezi těmito městy a dále označme $c_{i,j}$ kapacitu této silnice, tj. maximální počet aut, která tudy mohou projet za hodinu. Konečně označme p_i kapacitu parkovišť ve městě a_i. Úkolem je zorganizovat pohyby aut tak, aby jich co největší počet dorazil do města a_n během T hodin.

Tuto úlohu lze převést na hledání maximálního toku v následující „časoprostorové“ transportní síti. Síť bude mít celkem $n(T+1)+1$ vrcholů: jeden zdroj z a pro každé město i bude mít $T+1$ vrcholů označených (i,t), kde $t = 0, \ldots, T$. Spotřebičem je vrchol $s = (n, T)$. (Viz obr. 8.4.)

V síti jsou tři druhy hran. Hrany od zdroje do vrcholů $(i, 0)$ mají kapacitu q_i a zajišťují počáteční stavy aut ve městech. Hrany z vrcholu (i,t) do vrcholu $(i, t+1)$ jsou na obrázku kresleny vodorovně a vyjadřují možnost parkování ve městě a_i; jejich kapacity jsou p_i. Konečně, vede-li z města i do města j přímá silnice, pak jsou v síti hrany z vrcholu (i,t) do vrcholu $(j, t+d_{i,j})$ o kapacitě $c_{i,j}$, které vyjadřují možnost jízdy, která trvá $d_{i,j}$ hodin.



Obrázek 8.4: Příklad k úloze o dynamickém toku.

8.2.9 Plánování oběhu lokomotiv. Je dána železniční síť a jízdní řád. Jaký nejmenší počet lokomotiv je třeba k zajištění jízdního řádu? Jak přitom musí lokomotivy přejíždět? Je-li známa cena za přejezd lokomotivy bez vlaku a cena prostoje lokomotivy, jaký je nejlevnější oběh lokomotiv? Pro jednoduchost předpokládejme, že jízdní řád je pro každý den stejný, tj. cyklicky se opakuje, a že každá lokomotiva může táhnout kterýkoli vlak.

Tyto úlohy lze řešit v síti, která je velmi podobná síti z obr. 2.2, str. 42, pro hledání v jízdním řádu. Za každou stanici, ze které jsou vypravovány vlaky a kde vlaky končí, je v síti řada vrcholů, které odpovídají okamžikům, kdy má v této stanici vlak vyjíždět nebo přijíždět. Vrcholy, které vyjadřují tutéž stanici v různých okamžicích, jsou pospojovány hranami (na obr. 2.2 kresleny vodorovně), které vyjadřují možnost čekání lokomotivy ve stanici. Dolní omezení jsou v těchto hranách nulová, kapacity záleží na velikosti nádraží a cena za jednotku toku je rovna ceně prostoje lokomotivy za příslušný časový úsek. Tyto hrany tvoří pro každou stanici 24-hodinový cyklus.

Další hrany (kresleny šikmo) odpovídají vlakům a jsou předepsány jízdním řádem. Dolní i horní omezení toku jsou zde rovna jedné, na ceně toku v těchto hranách tedy nezáleží.

Konečně třetí typ hran (na obr. 2.2 nejsou nakresleny) vyjadřuje možnost přejezdu samotné lokomotivy mezi stanicemi. Jednotková cena zde vyjadřuje náklady na přejezd, dolní omezení toku je nulové a kapacita záleží na tom, kolik lokomotiv lze při takové jízdě spřáhnout dohromady.

Nejlevnější přípustná cirkulace v této síti určuje nejlevnější oběh lokomotiv. Cyklické trasy jednotlivých lokomotiv tím však nemusí být určeny jednoznačně: Jestliže na nádraží čeká několik lokomotiv, může nejbližší vlak táhnout kterákoli z nich.

Poznamenejme, že prostřednictvím cen přejezdů a ceny prostoje lze výsledný tok v širokých mezích ovládat. Bude-li cena přejezdu kladná a srovnatelná s cenou prostoje, bude výsledný tok odpovídat minimálnímu možnému počtu lokomotiv. Bude-li cena prostoje zanedbatelná vzhledem k ceně přejezdu, lokomotivy nebudou přejíždět, budou vždy čekat na svůj vlak. Výsledný počet lokomotiv bude větší, ale ušetříme na přejezdech.

JINÉ ŘEŠENÍ je podobně jako 2.2.5, str. 42, založeno na zcela jiné síti. Pro každý vlak i je v síti jedna hrana e_i. Dolní i horní omezení toku v této hraně je rovno jedné. Jsou-li i, j dva vlaky (ne nutně různé), pak z vrcholu $\text{Kv}(e_i)$ vede hrana do vrcholu $\text{Pv}(e_j)$. Tato hrana vyjadřuje možnost, že lokomotiva, která táhla vlak i, může po čekání a případném přejezdu táhnout vlak j (pozor, přejezd a čekání se mohou protáhnout do dalšího dne). Dolní omezení v takové hraně je nulové, kapacita je rovna jedné a cena za jednotku toku je rovna ceně čekání a případného přejezdu lokomotivy.

Přípustná cirkulace v této síti vytváří cykly, které odpovídají cyklickým jízdám lokomotiv, nejlevnější cirkulace pak odpovídá nejlevnějšímu oběhu lokomotiv. Poznamenejme, že cyklická jízda může trvat i několik dní. To pak znamená, že po této cyklické trase musí obíhat několik lokomotiv.

8.2.10 Cvičení. Spíše pro zajímavost než pro praktickou potřebu uveďme, že úlohu najít nejkratší cestu z výchozího vrcholu r do cílového vrcholu c lze převést na hledání nejlevnějšího přípustného toku. Popište detaily. Návod: délky hran budou použity jako jednotkové ceny toku.

Co se stane, když graf obsahuje cyklus se zápornou délkou?

8.3 Maximální tok a minimální řez

Uvedeme základní algoritmus pro řešení obou uvedených úloh a odvodíme z něj důležitá tvrzení o vztahu kapacity řezu a velikosti toku. Dodejme, že existují i podstatně rychlejší, ale složitější algoritmy, které zde neuvádíme.

8.3.1 Princip algoritmu pro maximální tok. Algoritmus je založen na postupném zvětšování (zlepšování) toku při zachování jeho přípustnosti. Výchozím předpokladem tedy je nějaký (jakýkoli) přípustný tok.

Změny toku v hranách nemůžeme dělat jednotlivě, protože je třeba zachovat platnost Kirchhoffova zákona: zvýšíme-li tok v hraně končící ve vrcholu v, je třeba současně buď zvýšit tok v některé hraně z $E^+(v)$, anebo snížit tok v hraně z $E^-(v)$. To ovšem vyvolá změnu toku v další hraně atd.

Nejjednodušší změna toku se tedy týká hran, které tvoří buď cestu od zdroje ke spotřebiči (tzv. *zlepšující cestu*), anebo kružnici (tzv. *zlepšující kružnici*). Velikost změny toku (zvýšení nebo snížení) musí být ve všech hranách zlepšující cesty stejná (jinak by se cesta musela „větvit“). Dodejme, že zlepšující cesta (kružnice) je neorientovaná, neboť může procházet i proti směru hran.

Algoritmus pro maximální tok bude hledat zlepšující cestu (viz 8.3.3) a na hranách této cesty měnit tok, což se bude opakovat, dokud to půjde. To, že výsledný tok bude maximální, není samo o sobě jasné. Naštěstí zároveň s tokem získáme i řez a maximálnost toku pak vyplyne z následující věty:

8.3.2 Věta (ověření optimálnosti toku a řezu). Předpokládejme, že f je přípustný tok ze zdroje z do spotřebiče s a že $W(A)$ je řez, který odděluje zdroj a spotřebič. Jestliže velikost toku f je rovna kapacitě řezu $W(A)$, tj. $F(f) = C(A)$, pak tok f je maximálním tokem ze zdroje do spotřebiče a řez $W(A)$ je minimálním řezem.

Poznámka: Vůbec přitom nezáleží, jak jsme takovou dvojici tok–řez získali. Při notné dávce štěstí jsme je dokonce mohli i uhodnout a pak už jen ověřit, že jsme hádali správně.

Všimněte si, jak si zde tok f a řez $W(A)$ vzájemně jeden druhému prokazují, že jsou (každý ve svém oboru) nejlepší.

Důkaz: Velikost žádného přípustného toku nemůže být větší než kapacita řezu $W(A)$, a tedy ani větší než velikost toku f. Tok f je tedy maximální. Podobně, řez $W(A)$ je minimální, protože žádný jiný řez nemůže mít menší kapacitu, než je velikost toku f. □

8.3.3 Zlepšující cesta. Nejprve zaveďme dva pojmy pro neorientované cesty a kružnice v orientovaných grafech: Hranu nazveme hranou *vpřed*, je-li orientována ve směru průchodu cestou (připomeňme, že cesta je definována jako posloupnost). Naopak *hrana vzad* je orientována proti směru průchodu cestou.

Předpokládejme, že v síti s omezeními toku l a c máme přípustný tok f. *Zlepšující cesta vzhledem k toku* f je taková neorientovaná cesta ze zdroje z do spotřebiče s, jejíž každá hrana e splňuje:

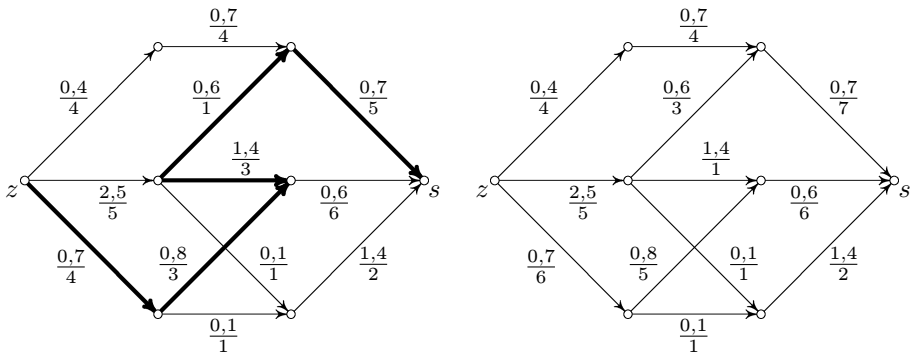
$$\begin{array}{ll} \text{je-li } e \text{ hranou vpřed,} & \text{pak } \ f(e) < c(e), \\ \text{je-li } e \text{ hranou vzad,} & \text{pak } \ f(e) > l(e). \end{array} \tag{8.1}$$

Jinými slovy, na hranách vpřed lze tok zvýšit a na hranách vzad jej lze snížit o nějakou hodnotu $d > 0$.

Kapacitu zlepšující cesty definujeme jako maximální hodnotu d, o kterou lze změnit tok na zlepšující cestě. Kapacita zlepšující cesty je tedy rovna minimu z rozdílů $c(e) - f(e)$ na hranách vpřed a $f(e) - l(e)$ na hranách vzad.

Je třeba zdůraznit, že zlepšující cesta i její kapacita se vždy vztahuje ke konkrétnímu toku. Když se tok změní, cesta už nemusí být zlepšující nebo může mít jinou kapacitu.

Na obr. 8.5 vlevo je silně vyznačena zlepšující cesta o kapacitě 2.



Obrázek 8.5: Zlepšující cesta a změna toku. V „čitateli“ jsou omezení toku, ve „jmenovateli“ je velikost toku. Kapacita zlepšující cesty je rovna 2.

8.3.4 Změna toku podél zlepšující cesty. Známe-li zlepšující cestu o kapacitě d vzhledem k přípustnému toku f, pak na hranách vpřed tok zvýšíme o d a na hranách vzad tok o stejnou hodnotu snížíme. Přípustnost toku i Kirchhoffův zákon tím zůstanou zachovány, ale celková velikost toku stoupne o hodnotu d.

Takto zvětšený tok můžeme zkusit dále zvětšovat. Je ovšem třeba najít novou zlepšující cestu (zlepšující vzhledem k novému toku). Tu starou nelze použít, protože alespoň v jedné hraně byl tok změněn „na doraz“.

Na obr. 8.5 vpravo je výsledný tok po provedení změny. Ověřte, že vzhledem k tomuto toku už neexistuje žádná zlepšující cesta.

8.3.5 Hledání zlepšující cesty – značkovací procedura. K hledání zlepšující cesty lze použít běžné postupy z kap. 3 teprve po úpravě, která zajistí, že výsledná cesta bude hranami grafu procházet ve správném směru s ohledem na stávající velikost toku.

1. [*Inicializace.*] Označkujeme vrchol z, ostatní vrcholy jsou beze značek.
2. [*Značkování vpřed.*] Existuje-li hrana e taková, že $\textsc{Pv}(e)$ má značku, $\textsc{Kv}(e)$ nemá značku a platí $f(e) < c(e)$, pak označkujeme $\textsc{Kv}(e)$.
3. [*Značkování vzad.*] Existuje-li hrana e taková, že $\textsc{Kv}(e)$ má značku, $\textsc{Pv}(e)$ nemá značku a platí $l(e) < f(e)$, pak označkujeme $\textsc{Pv}(e)$.
4. [*Test ukončení.*] Je-li označkován spotřebič s, značkování končí – zlepšující cesta je nalezena. Není-li spotřebič označkován a nelze-li označkovat další vrchol, značkování končí – zlepšující cesta neexistuje.

Aby bylo možné zlepšující cestu zpětně zkonstruovat, je vhodné zaznamenávat ke každému označkovanému vrcholu hranu, která označkování způsobila, a případně také to, zda při tom byla použita jako hrana vpřed nebo vzad.

Poznamenejme, že pro hledání zlepšující cesty lze upravit i ostatní metody prohledávání grafu, viz též 8.3.13.

8.3.6 Fordův-Fulkersonův algoritmus pro maximální tok.

Vstup: Orientovaný graf G, zdroj z, spotřebič s, celočíselná omezení toku l a c a celočíselný přípustný tok f.
Výstup: Tok f a množina $A \subset V(G)$, která určuje řez.
Úkol: Změnit tok f tak, aby byl maximálním tokem od zdroje z ke spotřebiči s a zároveň najít minimální řez oddělující z a s.
Pomocné proměnné: Značky vrcholů a hodnoty Odkud pro konstrukci cest.

Algoritmus:

1. [*Značkování.*] Použijeme značkovací proceduru 8.3.5. Pokud dojde k označkování spotřebiče s, pokračujeme krokem 2, jinak krokem 3.
2. [*Změna toku.*] Zjistíme kapacitu zlepšující cesty (viz 8.3.3) a změníme tok podle 8.3.4. Pokračujeme krokem 1.
3. [*Konec výpočtu.*] Označme A množinu označkovaných vrcholů. Výpočet končí, f je maximální tok a $W(A)$ je minimální řez.

8.3.7 Věta. Fordův-Fulkersonův algoritmus 8.3.6 se po konečně mnoha krocích zastaví. Po zastavení je f maximálním tokem ze zdroje z do spotřebiče s a řez $W(A)$ je minimálním řezem oddělujícím zdroj z a spotřebič s.

Důkaz: Každým provedením kroku 2 se velikost toku zvýší o celočíselnou hodnotu, tedy nejméně o 1. Velikost toku nemůže nabývat neomezených hodnot (díky kapacitám řezů), a proto dosáhne maxima po konečně mnoha změnách.

Po zastavení je řez $W(A)$ tvořen hranami, které nedovolují další značkování. Tyto hrany tedy splňují

$$f(e) = c(e) \quad \text{pro} \quad e \in W^+(A), \qquad f(e) = l(e) \quad \text{pro} \quad e \in W^-(A),$$

jinak by bylo možno značkovat dál. To však znamená, že velikost toku f je rovna

$$F \;=\; \sum_{e \in W^+(A)} f(e) \;-\; \sum_{e \in W^-(A)} f(e) \;=\; \sum_{e \in W^+(A)} c(e) \;-\; \sum_{e \in W^-(A)} l(e)\,,$$

což je kapacita řezu $W(A)$. Podle 8.3.2 je tedy tok maximální a řez minimální. □

Poznámka: Pro neceločíselná omezení toku a neceločíselný tok se algoritmus nemusí vůbec zastavit – tok se může zvětšovat po stále menších krocích a nikdy nedosáhnout maxima. K tomu ovšem musí být omezení toku iracionální čísla (např. $\sqrt{5}$) a ještě je třeba provádět značkování dost nešikovným způsobem. Z praktického hlediska je to taková matematická schválnost.

8.3.8 Věta (Fordova-Fulkersonova o maximálním toku a minimálním řezu). Velikost maximálního toku od zdroje ke spotřebiči je rovna kapacitě minimálního řezu oddělujícího zdroj a spotřebič.

Důkaz: Vezměme maximální tok a použijme jej jako výchozí tok ve Fordově-Fulkersonově algoritmu 8.3.6. Algoritmus se hned po prvém použití značkovací procedury zastaví (tok už nelze zvětšovat) a podle věty 8.3.7 najde minimální řez $W(A)$, který má kapacitu rovnou velikosti toku. □

8.3.9 Kde vzít výchozí přípustný tok. Nejjednodušší tok, který si lze vymyslet, je tok nulový, neboť splňuje Kirchhoffův zákon automaticky. Máme-li to štěstí, že nulový tok je přípustný (např. v transportní síti), lze jej použít jako výchozí.

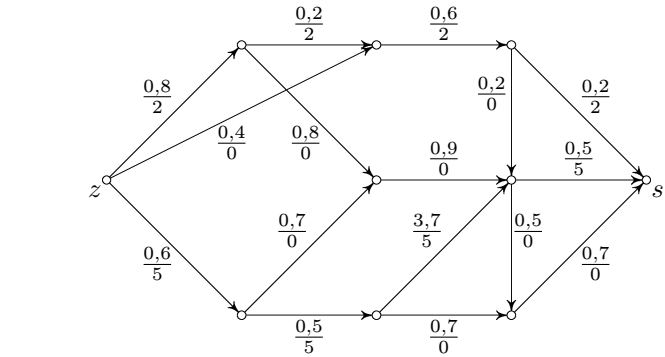
V ostatních případech máme několik možností. Často se vyplatí (poněkud hazardně) začít z nulového toku a při hledání zlepšujících cest dát přednost hranám, v nichž má být tok nenulový. Samozřejmě je pak nutno přípustnost toku ověřit.

Ve složitějších případech, zejména máme-li podezření, že přípustný tok neexistuje, je třeba použít speciální postupy popsané v oddíle 8.4.

8.3.10 Příklad. Postup hledání maximálního toku předvedeme na síti, která je nakreslena na obr. 8.6 nahoře. Jako výchozí je zde použit tok získaný změnou podél dvou snadno nalezených zlepšujících cest: jedna prochází (jedinou) hranou s nenulovým dolním omezením, druhá prochází hranami podél horního okraje obrázku.

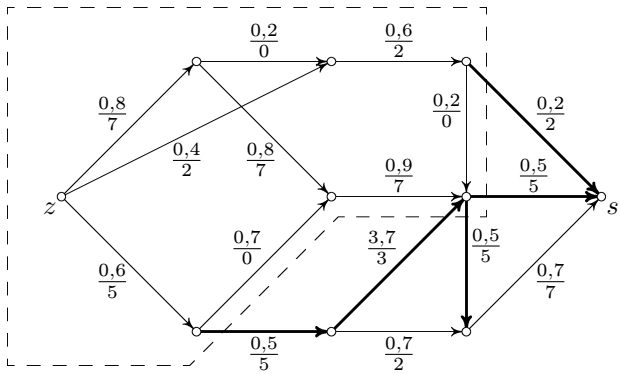
Použité zlepšující cesty jsou nakresleny uprostřed obrázku, výsledný maximální tok je nakreslen dole. Množina vrcholů označkovaných při posledním značkování je čárkovaně orámována a hrany tvořící minimální řez jsou zvýrazněny.

Všimněte si, že čtyři hrany minimálního řezu jsou tokem nasyceny, zatímco v páté, vzhledem k řezu obráceně orientované, je tok roven dolnímu omezení. Stojí za upozornění, že ne všechny nasycené hrany jsou součástí minimálního řezu.



Transportní síť a výchozí tok

Použité zlepšující cesty



Výsledný maximální tok a minimální řez

Obrázek 8.6: Příklad výpočtu maximálního toku (viz 8.3.10). Hrany jsou označeny stejným způsobem jako na obr. 8.5.

Dále si všimněte, že jsme při řešení museli ve zlepšující cestě použít i hrany orientované vzad, což je důsledkem volby výchozího toku.

8.3.11 Rady k ručnímu počítání. Graf o velikosti do cca 15 vrcholů a 30 hran obvykle jde „rozumně“ nakreslit: zdroj vlevo, spotřebič vpravo, většinu hran pokud možno zleva doprava a samozřejmě si nechejte spoustu místa pro ohodnocení hran. Několik prvých zlepšujících cest najdete pouhým pohledem. Změny toku zaznamenávejte přímo do grafu, staré hodnoty toku škrtejte. Teprve když už žádnou další zlepšující cestu nevidíte, začněte značkovat.

8.3.12 Cvičení. Najděte maximální tok a minimální řez ve dvou sítích s množinou vrcholů $\{1, 2, \ldots, 9\}$, zdrojem 1 a spotřebičem 9. Hrany a omezení toku jsou dány těmito tabulkami:

G_1:

Pv	Kv	l	c	f	Pv	Kv	l	c	f
1	2	0	6	0	5	1	0	3	0
1	3	0	6	0	5	6	0	6	0
1	6	0	4	0	5	7	0	5	4
2	3	0	2	0	5	8	0	5	0
2	4	0	3	0	6	9	0	5	0
3	4	0	4	0	7	4	2	6	4
3	5	0	4	0	7	9	0	3	0
4	5	3	5	4	8	9	0	9	0
4	8	0	3	0					

G_2:

Pv	Kv	l	c	f	Pv	Kv	l	c	f
1	2	0	6	0	5	4	0	3	0
1	3	0	5	0	5	8	0	6	0
1	6	0	3	0	6	3	0	2	0
2	3	0	3	0	6	9	0	7	0
2	4	0	5	0	7	8	0	2	0
2	5	0	2	0	7	9	0	2	0
3	5	0	4	0	8	6	0	6	0
4	7	0	3	0	8	9	0	4	0

8.3.13 Rychlost výpočtu maximálního toku. Značkovací procedura 8.3.5 umožňuje volit postup značkování dosti libovolným způsobem. Při nešikovném postupu značkování pak může být výpočet velmi pomalý, tok se může v krajním případě zvětšovat i jen po jednotkových krocích.

Doporučuje se při značkování vrcholů používat metodu hledání do šířky (viz 3.2.1, str. 51), tj. používat zlepšující cesty s co nejmenším počtem hran. Pak totiž lze zaručit, že maximální tok bude nalezen v čase $O(m^2 n)$. Všimněte si, že tento časový odhad nezávisí na kapacitách hran ani na celkové velikosti maximálního toku.

Existuje řada mnohem rychlejších postupů, ovšem podstatně složitějších. Algoritmus pracující v čase $O(n^3)$ lze najít v knize [Kučera83]. Viz též poznámku 8.5.8.

8.3.14 Poznámka o hranách vzad. Značkování proti směru hran nelze v algoritmu 8.3.6 vynechat. Jinak by totiž z některých výchozích toků vůbec nebylo možné dosáhnout toku maximálního. Příklad je na obrázku 8.5.

Dokonce i když vyjdeme z nulového toku a budeme jej zvětšovat podél zlepšujících cest o nejmenším počtu hran (podle 8.3.13), může být snížení toku v některé hraně nezbytné. Najděte příklad takové sítě.

Existují však sítě, kde se lze bez hran vzad obejít:

8.3.15 Maximální tok v rovinné transportní síti. Lze-li transportní síť nakreslit rovinným způsobem tak, že zdroj je nakreslen nejvíce vlevo a spotřebič nejvíce vpravo, přičemž žádná hrana nezasahuje nalevo od zdroje ani napravo od spotřebiče,[1] lze pro hledání maximálního toku použít jednodušší a rychlejší postup:

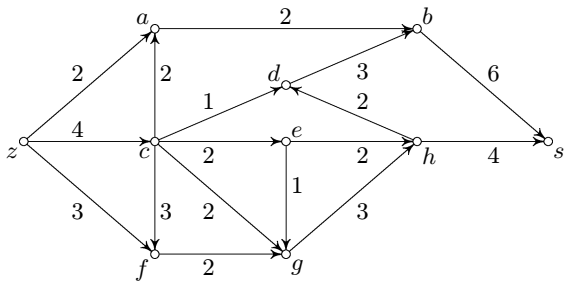
Výpočet zahájíme s nulovým tokem. Ke změnám toku používáme vždy „nejhořejší“ zlepšující cestu, která neobsahuje hranu vzad. Zlepšující cesta tedy bude cestou orientovanou, tok se bude vždy jen zvyšovat.

K nalezení zlepšující cesty lze použít orientované hledání do hloubky, při kterém z každého vrcholu postupujeme nejprve hranou, která není nasycena a která vede nejvíce vlevo. Zjišťování kapacity zlepšující cesty i změny toku se provádějí stejně jako v obecném algoritmu 8.3.6. Výpočet končí, když už neexistuje žádná *orientovaná* zlepšující cesta. Lze dokázat, že pak totiž neexistuje ani obecná zlepšující cesta, a nalezený tok je tedy maximální.

Rychlost tohoto algoritmu je založena na faktu, že v žádné hraně tok nikdy neklesne, neboť nepoužíváme hrany vzad. Každá zlepšující cesta způsobí nasycení alespoň jedné hrany, celkový počet změn toku je tedy nejvýše roven počtu hran m. Časový odhad tedy je $O(m(m+n))$, ale díky větě 14.3.3, str. 231, to pro prosté rovinné grafy bez smyček je $O(n^2)$.

Pokud jde o minimální řez, viz též 14.2.4, str. 230.

8.3.16 Příklad. Použijme algoritmus 8.3.15 k nalezení maximálního toku v rovinné transportní síti z obr. 8.7.



Obrázek 8.7: Rovinná transportní síť k příkladu 8.3.16.

Začínáme s nulovým tokem. Tok zvětšujeme podél těchto zlepšujících cest:

Cesta	Kapacita
z, a, b, s	2
z, c, d, b, s	1
z, c, e, h, d, b, s	2
z, c, g, h, s	1
z, f, g, h, s	2

[1] Tj. zdroj i spotřebič jsou na hranici tzv. neomezené stěny, viz 14.1.2, str. 225.

Celková velikost maximálního toku je 8. Najděte minimální řez (např. algoritmem 8.3.6 to už půjde rychle) a ověřte, že tok je opravdu maximální.

8.3.17 Cvičení. Graf G_2 z cvičení 8.3.12 nakreslete rovinným způsobem a najděte maximální tok algoritmem 8.3.15.

8.4 Přípustná cirkulace

Budeme se zabývat cirkulacemi, protože pro ně jsou algoritmy i teorie jednodušší. Úlohy týkající se přípustného toku od zdroje ke spotřebiči lze snadno převést na hledání přípustné cirkulace přidáním návratové hrany, viz 8.4.9.

8.4.1 Věta. V síti s omezeními toku l a c existuje přípustná cirkulace právě tehdy, když každý řez $W(A)$ má nezápornou kapacitu, tj.

$$C(A) \;=\; \sum_{e\in W^+(A)} c(e) \;-\; \sum_{e\in W^-(A)} l(e) \;\geq\; 0\,.$$

Poznámka: Jinak řečeno, přípustná cirkulace existuje právě tehdy, když každá množina vrcholů A má tu vlastnost, že tok, který do ní povinně musí vtéci kvůli dolnímu omezení l na hranách z $W^-(A)$, může z této množiny také odtéci díky hornímu omezení c na hranách z $W^+(A)$.

Motivace k této větě je poněkud alibistická: nemůžeme-li najít přípustnou cirkulaci, chceme umět alespoň jednoduše prokázat, že to není způsobeno naší neschopností nebo nedostatečným úsilím. Najdeme-li řez se zápornou kapacitou, je jasné, že žádná přípustná cirkulace nemůže existovat.

Důkaz: Existuje-li přípustná cirkulace f, pak pro každý řez platí:

$$C(A) \;=\; \sum_{e\in W^+(A)} c(e) \;-\; \sum_{e\in W^-(A)} l(e) \;\geq\; \sum_{e\in W^+(A)} f(e) \;-\; \sum_{e\in W^-(A)} f(e) \;=0\,.$$

Jestliže naopak přípustná cirkulace neexistuje, pak existence řezu se zápornou kapacitou vyplyne z následujícího algoritmu, který jej sestrojí:

8.4.2 Algoritmus pro hledání přípustné cirkulace.

Vstup: Síť G s omezeními toku l, c a libovolná cirkulace f (např. i nulová).
Úkol: Změnit cirkulaci f na přípustnou anebo najít řez se zápornou kapacitou.

1. Najdeme hranu h s nepřípustným tokem. Pokud taková hrana neexistuje, výpočet končí, cirkulace f je přípustná.

2. Jestliže $f(h) < l(h)$, položíme $z := \mathrm{Kv}(h), s := \mathrm{Pv}(h)$, v opačném případě, tj. pokud $f(h) > c(h)$, položíme $z := \mathrm{Pv}(h), s := \mathrm{Kv}(h)$.

3. Použijeme značkovací proceduru 8.3.5 pro nalezení zlepšující cesty z vrcholu z do vrcholu s. Když je cesta nalezena, pokračujeme krokem 4. Pokud cesta neexistuje, výpočet končí, přípustná cirkulace neexistuje a množina označkovaných vrcholů A určuje řez $W(A)$, který má zápornou kapacitu.

4. Zlepšující cestu doplníme o hranu h, čímž vznikne zlepšující kružnice. Vypočteme její kapacitu a změníme tok na jejích hranách podle 8.3.3 a 8.3.4. Pak pokračujeme krokem 1.

8.4.3 Věta. Algoritmus 8.4.2 se po konečně mnoha krocích zastaví. Po zastavení v kroku 1 je f přípustnou cirkulací. Po zastavení v kroku 3 určuje množina A řez se zápornou kapacitou, a přípustná cirkulace proto neexistuje.

Důkaz: K důkazu, že se algoritmus zastaví, zavedeme pomocný pojem *defekt* toku f v hraně e:

$$def_e(f) = \begin{cases} f(e) - c(e) & \text{pro } f(e) > c(e), \\ 0 & \text{pro } l(e) \le f(e) \le c(e), \\ l(e) - f(e) & \text{pro } f(e) < l(e). \end{cases}$$

Defekt je vždy nezáporný a vyjadřuje vzdálenost toku $f(e)$ od přípustného intervalu. Cirkulace je přípustná, je-li součet defektů všech hran roven nule.

Volba vrcholů z, s a hledání zlepšující cesty jsou prováděny tak, že při následné změně toku defekt hrany h klesne (o hodnotu kapacity zlepšující kružnice) a defekty ostatních hran přitom nevzrostou. Algoritmus tedy může vykonat pouze konečný počet změn toku (díky celočíselnosti).

Zbývá dokázat, že pokud se algoritmus zastavil v kroku 3, množina A určuje řez se zápornou kapacitou. Z vlastností značkovací procedury plyne:

$$\begin{aligned} f(e) &\ge c(e) \quad \text{pro hrany} \quad e \in W^+(A), \\ f(e) &\le l(e) \quad \text{pro hrany} \quad e \in W^-(A), \end{aligned}$$

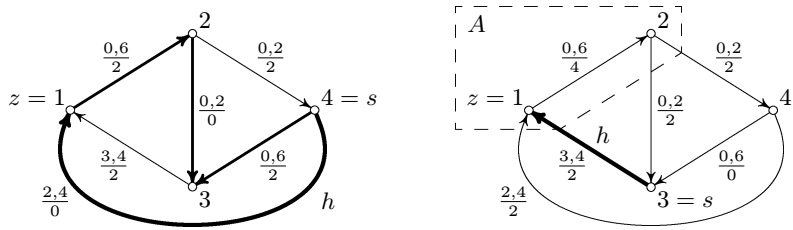
jinak by bylo možno pokračovat ve značkování. Tyto nerovnosti platí i pro hranu h, pro ni je však příslušná nerovnost ostrá, protože tok $f(h)$ je nepřípustný. Platí tedy:

$$\sum_{e \in W^+(A)} c(e) \;\le\; \sum_{e \in W^+(A)} f(e) \;=\; \sum_{e \in W^-(A)} f(e) \;\le\; \sum_{e \in W^+(A)} l(e)\,,$$

přičemž rovnost uprostřed plyne z faktu, že f je cirkulace. Díky hraně h je však zde alespoň jedna z obou nerovností ostrá, a tedy

$$\sum_{e \in W^+(A)} c(e) \;<\; \sum_{e \in W^+(A)} l(e)\,,$$

a kapacita řezu $W(A)$ je tedy záporná. □



Obrázek 8.8: Hledání přípustné cirkulace (viz 8.4.4).

8.4.4 Příklad. Činnost algoritmu 8.4.2 předvedeme na síti z obr. 8.8 vlevo. Hrany s nepřípustným tokem jsou zde dvě, použijeme hranu označenou h a nakreslenou tlustě. Zlepšující kružnice o kapacitě 2 je vyznačena středně silnou čarou. Výsledná cirkulace (je nakreslena vpravo) ještě není přípustná, ale zlepšující kružnice, která by snížila defekt v hraně h, zde již neexistuje. Množina označkovaných vrcholů A určuje řez o kapacitě -1.

8.4.5 Jiný algoritmus pro přípustnou cirkulaci. Úlohu najít přípustnou cirkulaci v síti G s omezeními toku l a c převedeme na úlohu o maximálním toku v transportní síti G' s kapacitami hran c'. Transportní síť G' sestrojíme takto:

1. Pro každou hranu $e \in E(G)$ položme $c'(e) := c(e) - l(e)$.
2. K síti G přidáme dva nové vrcholy, umělý zdroj z a umělý spotřebič s.
3. Pro každý vrchol $x \in V(G)$ vypočítáme výraz
$$L(x) \;=\; \sum_{e \in E^+(x)} l(e) \;-\; \sum_{e \in E^-(x)} l(e)\,.$$
4. Pro každý vrchol x takový, že $L(x) > 0$, přidáme hranu o kapacitě $c'(e_x) = = L(x)$ z vrcholu x do spotřebiče s.
5. Pro každý vrchol x takový, že $L(x) < 0$, přidáme hranu o kapacitě $c'(e_x) = = -L(x)$ ze zdroje z do vrcholu x.

Vtip této konstrukce je v tom, že na všech původních hranách intervaly $\langle l(e), c(e)\rangle$ pro přípustný tok posuneme na $\langle 0, c'(e)\rangle$ a chybějící nebo přebývající tok vyrovnáme v přidaných hranách.

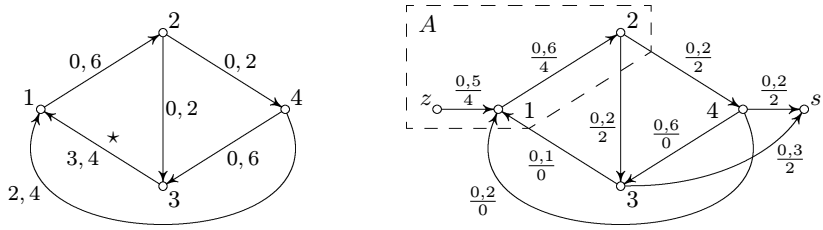
V takto vytvořené síti G' najdeme (libovolným způsobem) maximální tok od zdroje z ke spotřebiči s. Lze dokázat, že platí:

Jestliže maximální tok f' v síti G' nasycuje všechny přidané hrany vycházející ze zdroje z, pak v původní síti G existuje přípustná cirkulace f daná předpisem $f(e) := f'(e) + l(e)$.

Jestliže naopak maximální tok f' v síti G' nenasycuje všechny hrany vycházející ze zdroje z, pak přípustná cirkulace v síti G neexistuje. Označíme-li v tomto případě

A množinu označkovanou procedurou 8.3.5 při hledání maximálního toku, pak řez $W_G(A \setminus \{z\})$ má zápornou kapacitu a brání existenci přípustné cirkulace.

8.4.6 Příklad. Použijeme algoritmus 8.4.5 k hledání přípustné cirkulace ve stejné síti jako v příkladě 8.4.4. Transportní síť G' i s maximálním tokem je nakreslena na obr. 8.9 vpravo. Maximální tok nenasycuje (jedinou) hranu vycházející ze zdroje z, přípustná cirkulace tedy neexistuje. Minimální řez v G' je nakreslen čárkovaně, jemu odpovídající řez se zápornou kapacitou v G je tedy stejný jako v příkladě 8.4.4. Obecně to tak být nemusí, řezů se zápornou kapacitou by mohlo být v síti několik, přičemž každý z nich by byl dostatečným důvodem pro neexistenci přípustné cirkulace.



Obrázek 8.9: Hledání přípustné cirkulace (viz 8.4.6).

8.4.7 Cvičení. V síti na obr. 8.9 snižte dolní omezení toku v hraně označené $\star$ z hodnoty 3 na hodnotu 2. K nalezení přípustné cirkulace použijte oba algoritmy.

8.4.8 Poznámka. Výhodou algoritmu 8.4.5 je možnost použít některý z rychlých algoritmů pro maximální tok. Naopak algoritmus 8.4.2 může být výhodnější v situaci, kdy již známe nějakou „skoro přípustnou“ cirkulaci.

8.4.9 Přípustný tok od zdroje ke spotřebiči lze hledat takto: K síti přidáme tzv. návratovou hranu ze spotřebiče do zdroje. Omezení toku v návratové hraně stanovíme s dostatečnou rezervou tak, aby tato hrana neomezovala tok ostatním grafem. Ve výsledné síti pak hledáme přípustnou cirkulaci. Pokud ji najdeme, odstraněním návratové hrany dostáváme řešení původní úlohy.

8.4.10 Cvičení. Navrhněte, jak v 8.4.9 zjistit, zda omezení toku v návratové hraně jsou stanovena s dostatečnou rezervou.

8.5 Obecné vlastnosti toků

V tomto oddíle uvedeme některé základní vlastnosti toků v sítích. Tyto vlastnosti jednak umožní hlubší porozumění pojmům a metodám použitým v předchozích oddílech, jednak budou podstatným způsobem využity v 8.6, kde se budeme zabývat nejlevnějšími toky v síti.

8.5.1 Součet a rozdíl toků. Jsou-li f_1 a f_2 dva toky v téže síti, pak ohodnocení f_3, f_4 daná předpisem $f_3(e) = f_1(e) + f_2(e)$ a $f_4(e) = f_1(e) - f_2(e)$ jsou také toky (splňují Kirchoffův zákon). Nazýváme je *součtem* a *rozdílem toků* f_1 a f_2. (O přípustnosti toků zde nic neříkáme.)

8.5.2 Přírustková síť. Mějme síť G, omezení toků l a c a nějaký (třeba i nepřípustný) tok f.

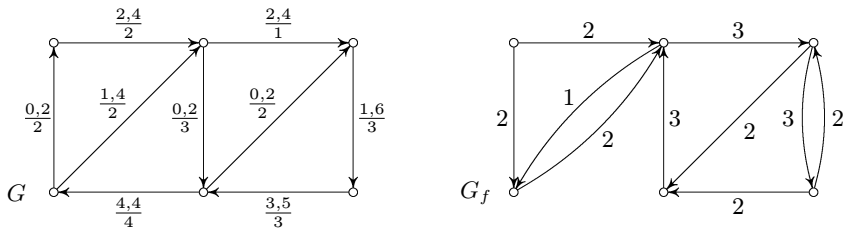
Hlavní smysl přírustkové sítě je ten, že poměrně komplikované hledání zlepšující cesty v původní síti převedeme na jednodušší hledání obyčejné orientované cesty v síti přírustkové.

Přírustková síť G_f vzhledem k toku f má stejnou množinu vrcholů jako síť G. Pro každou hranu $e \in E(G)$ definujeme hrany přírustkové sítě takto:

Jestliže $f(e) < c(e)$, pak bude v přírustkové síti hrana e_+ orientovaná stejně jako hrana e. Její kapacita bude $c(e) - f(e)$. Takovou hranu budeme nazývat hranou vpřed.

Jestliže $f(e) > l(e)$, pak bude v přírustkové síti hrana e_- orientovaná opačně než hrana e. Její kapacita bude $f(e) - l(e)$. Takovou hranu budeme nazývat hranou vzad.

Ve všech hranách přírustkové sítě je dolní omezení toku nulové. Na obr. 8.10 je příklad cirkulace a příslušná přírustková síť G_f.



Obrázek 8.10: Cirkulace a přírustková síť.

Je třeba zdůraznit, že podobně jako zlepšující cesta i přírustková síť se vždy vztahuje k nějakému toku a že po změně toku bude přírustková síť vypadat jinak.

Každé zlepšující cestě nebo kružnici v síti G odpovídá (a to jednoznačně) orientovaná cesta nebo cyklus v přírustkové síti. Na značkovací proceduru 8.3.5 se pak můžeme dívat tak, jako by hledala orientovanou cestu v přírustkové síti, aniž by ji ovšem explicite používala.

Vztah mezi toky v přírustkové síti G_f a změnami toku v G je ovšem hlubší:

8.5.3 Tok v přírustkové síti. Nechť f je tok v síti G. Každému přípustnému toku f_p v přírustkové síti G_f odpovídá v síti G tok f', který vznikne změnou původního toku f o hodnotu toku f_p takto:

Je-li hrana e_+ v přírustkové síti hranou vpřed, pak $f'(e) := f(e) + f_p(e_+)$. Je-li e_- hranou vzad, pak $f'(e) := f(e) - f_p(e_-)$.

Ze způsobu konstrukce přírustkové sítě plyne, že pokud původní tok f byl přípustný v G a pokud f_p byl přípustný v G_f, pak i výsledný tok f' je přípustný v G. Platí to i naopak: jsou-li toky f a f' přípustné v G, pak existuje „rozdílový tok“ f_p přípustný v přírustkové síti takový, že f' vznikne z f změnou o hodnotu toku f_p.

8.5.4 Cyklický tok. Tok, který má (stejnou) kladnou velikost x na všech hranách nějakého cyklu a na ostatních hranách je nulový, nazýváme *cyklickým tokem o velikosti* x.

8.5.5 Věta. Každá nezáporná cirkulace f v síti s m hranami je součtem nejvýše m cyklických toků.

Důkaz: Je-li cirkulace f nulová, věta samozřejmě platí. Je-li v některé hraně e nenulový (tedy kladný) tok $f(e) > 0$, pak z vrcholu $\text{Kv}(e)$ vychází nějaká hrana e_1, která má také nenulový tok (jinak by neplatil Kirchhoffův zákon). Z koncového vrcholu hrany e_1 vychází další hrana e_2 s nenulovým tokem atd. Díky konečnému počtu vrcholů tedy v síti existuje cyklus, v jehož hranách je nenulový tok. Označme x minimum z toků v hranách tohoto cyklu. Vytvořme cyklický tok f_1 o velikosti x na hranách tohoto cyklu a odečtěme jej od toku f. Tím vynulujeme tok v alespoň jedné hraně – v té, kde byl tok nejmenší.

Výsledný rozdílový tok je opět nezápornou cirkulací, můžeme na něj tedy znovu použít stejný postup: buď je cirkulace již nulová, nebo obsahuje cyklus s nenulovým tokem. Vytvoříme cyklický tok f_2 a odečteme jej.

Po nejvýše m opakováních tohoto postupu dostaneme nulovou cirkulaci. To znamená, že původní cirkulace f byla součtem cyklických toků. □

8.5.6 Věta. Jsou-li f_1 a f_2 dvě přípustné cirkulace, lze cirkulaci f_2 získat z cirkulace f_1 změnami podél kružnic, které jsou zlepšující vzhledem k cirkulaci f_1.

Důkaz: Cirkulaci f_2 můžeme získat z cirkulace f_1 změnou o hodnotu rozdílového toku f_p, který je přípustný v přírustkové síti vzhledem k f_1. Tok f_p je nezáporný, a je tedy součtem cyklických toků. Každý z těchto cyklických toků však představuje změnu podél zlepšující kružnice vzhledem k f_1. Provedením všech těchto změn změníme cirkulaci f_1 na cirkulaci f_2. □

8.5.7 Poznámka. Z vět 8.5.5 a 8.5.6 vyplývá, že maximální tok v síti lze získat provedením nejvýše m změn toku podél zlepšujících cest, a to dokonce bez použití hran vzad. Potíž ovšem je, že předem nevíme, které změny a o kolik máme provést.

8.5.8 Poznámka. Většina rychlých algoritmů pro výpočet maximálního toku nepoužívá pro zvětšování toku jednotlivé zlepšující cesty. Tyto algoritmy zvětšují tok po větších dávkách takto: Pro stávající přípustný tok f sestrojí přírustkovou síť G_f a z této sítě pak odstraní všechny hrany kromě těch, které v G_f leží na nějaké nejkratší cestě ze zdroje do spotřebiče (délka cesty se přitom měří jako počet hran). Tím získají tzv. *vrstvenou síť* (vrcholy jsou uspořádány do

vrstev podle vzdáleností ze zdroje a do spotřebiče, hrany vedou vždy jen do bezprostředně následující vrstvy). Ve vrstvené síti se pak hledá tzv. nasycený tok f_p, tj. takový tok, že každá cesta ze zdroje do spotřebiče obsahuje alespoň jednu nasycenou hranu. Díky speciální struktuře vrstvené sítě lze takový tok najít velmi rychle (jednotlivé algoritmy se liší ve způsobu, jak to dělají). Tok f_p se pak přičte k toku f a celý postup se opakuje, dokud v přírustkové síti G_f existuje cesta ze zdroje do spotřebiče. Podrobnosti lze najít například v knize [Kučera83].

8.6 Nejlevnější cirkulace

Uvedeme jednoduchou metodu hledání nejlevnější cirkulace. Důmyslnější postup uvedeme v 8.7.

8.6.1 Cena cirkulace, cena zlepšující kružnice. Mějme síť G s omezeními toku l a c a s jednotkovými cenami toku a. Dále mějme libovolnou cirkulaci f. Připomeňme (viz 8.1.11, str. 133), že cena toku v hraně e je rovna součinu $a(e)f(e)$ a že cena celé cirkulace je součtem cen toků ve všech hranách.

Cenu zlepšující kružnice definujeme jako součet jednotkových cen na hranách vpřed mínus součet jednotkových cen na hranách vzad. Je-li tedy cena zlepšující kružnice p a změníme-li tok podél této kružnice o hodnotu r, pak celková cena cirkulace vzroste o hodnotu rp.

Pojem jednotkové ceny toku můžeme definovat i pro přírustkovou síť G_f vzhledem k toku f. Na hranách vpřed budou v přírustkové síti ceny stejné jako na odpovídajících hranách původní sítě G, na hranách vzad budou ceny stejné jako v G, ale s opačným znaménkem. Zlepšující kružnici v původním grafu G odpovídá cyklus v grafu G_f, přičemž jejich ceny jsou stejné.

8.6.2 Věta (charakterizace nejlevnějších cirkulací). Přípustná cirkulace f je nejlevnější cirkulací v síti G právě tehdy, když v síti G neexistuje vzhledem k cirkulaci f žádná zlepšující kružnice se zápornou cenou.

Důkaz: Jedna implikace je snadná: Kdyby existovala zlepšující kružnice se zápornou cenou, bylo by možno podél ní cirkulaci změnit a tím snížit její celkovou cenu. Cirkulace f by tedy nebyla nejlevnější.

Dokážeme opačnou implikaci: Nechť f je cirkulace, vzhledem ke které neexistuje zlepšující kružnice se zápornou cenou. Označme f_1 nejlevnější přípustnou cirkulaci v síti G (určitě existuje). Cirkulace f_1 je dosažitelná z cirkulace f změnami podél zlepšujících kružnic vzhledem k cirkulaci f. Žádná z těchto kružnic nemá zápornou cenu. Cirkulace f_1 proto není levnější než f. Jinak řečeno, cirkulace f již byla (také) nejlevnější cirkulací v síti G. □

8.6.3 Algoritmus pro nalezení nejlevnější cirkulace. Předchozí věta dává jednoduchý návod k hledání nejlevnější cirkulace.

1. Najdeme přípustnou cirkulaci f (viz 8.4). Jestliže přípustná cirkulace neexistuje, výpočet končí.

2. Sestrojíme přírustkovou síť G_f vzhledem k cirkulaci f.

3. V přírustkové síti budeme jednotkové ceny toku pokládat za délky hran a najdeme zde (viz 6.8) cyklus se zápornou délkou. Jestliže takový cyklus neexistuje, výpočet končí, cirkulace f je podle věty 8.6.2 nejlevnější.

4. V síti G sestrojíme zlepšující kružnici, která odpovídá nalezenému cyklu se zápornou délkou, vypočítáme kapacitu zlepšující kružnice, změníme cirkulaci a pokračujeme podle kroku 2.

Přírustkové sítě není nutno explicite sestrojovat, algoritmy pro hledání cyklů se zápornou délkou lze upravit tak, aby pracovaly přímo v síti G s cirkulací f.

8.6.4 Příklad výpočtu nejlevnější přípustné cirkulace je znázorněn na obr. 8.11. Vlevo je vždy nakreslena síť s vyznačenou cirkulací, vpravo je odpovídající přírustková síť. Hrany, které jsou nakresleny silně, tvoří cyklus se zápornou cenou, který byl použit ke změně cirkulace. Celková cena cirkulace se snížila z hodnoty -1 na hodnotu -4. Poznamenejme, že při troše štěstí při volbě zlepšující kružnice jsme mohli získat nejlevnější cirkulaci jedinou úpravou. Najděte tuto kružnici jako cvičení.

8.7 Algoritmus Out of Kilter

Algoritmus Out of Kilter[2] pro výpočet nejlevnější přípustné cirkulace (kromě samozřejmé manipulace s tokem) přiřazuje vrcholům grafu hodnoty, které budeme nazývat *potenciály*. Rozdíly potenciálů na hranách (lze si je představit jako elektrické napětí) se v algoritmu využívají k efektivnímu hledání zlepšujících kružnic se zápornou cenou, případně k důkazu, že taková kružnice neexistuje, a že už tedy je dosaženo optima.

Algoritmus Out of Kilter byl původně odvozen z teorie duality v lineárním programování.[3] Tato teorie hodně souvisí s kombinatorickou optimalizací, ale její výklad značně přesahuje rozsah a zaměření této knížky. Následující větu lze z duality lineárního programování dokázat elegantněji, než je uvedeno zde, totiž bez odkazu na větu 8.6.2.

8.7.1 Věta. Buď dána síť G s omezeními toku l, c a s jednotkovými cenami toku a. Nechť f je cirkulace. Jestliže existuje ohodnocení vrcholů $u : V(G) \to \mathbb{R}$ takové,

[2] Slovo „kilter“ je keltského původu a znamená pořádek.
[3] Potenciály vypočtené algoritmem Out of Kilter jsou optimálním řešením duální úlohy.

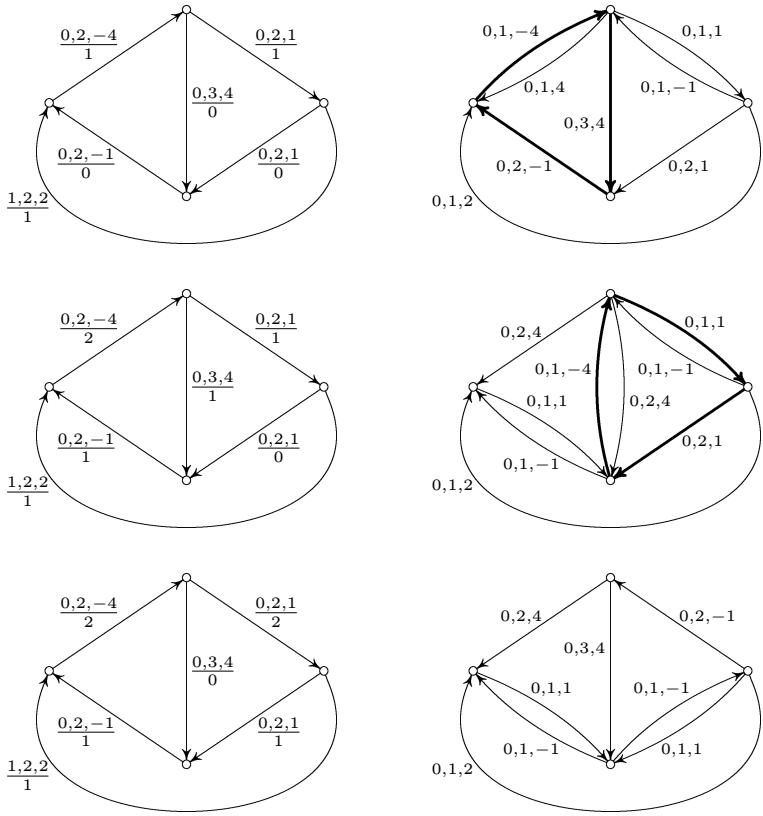
že pro každou hranu e platí tzv. *kilter-podmínky*

$$\text{(K)} \qquad \begin{cases} l(e) \le f(e) \le c(e), \\ u(\text{K}v(e)) - u(\text{P}v(e)) > a(e) \;\Rightarrow\; f(e) = c(e), \\ u(\text{K}v(e)) - u(\text{P}v(e)) < a(e) \;\Rightarrow\; f(e) = l(e), \end{cases}$$

pak cirkulace f je nejlevnější přípustnou cirkulací.

Důkaz: Předpokládejme, že cirkulace f a potenciály u splňují kilter-podmínky. Dokážeme, že žádná zlepšující kružnice vzhledem k cirkulaci f nemá zápornou cenu. Odtud již podle věty 8.6.2 plyne, že f je nejlevnější cirkulací.

Vezměme tedy libovolnou zlepšující kružnici. Z kilter-podmínek a z toho, že jde



Obrázek 8.11: Výpočet nejlevnější přípustné cirkulace (viz 8.6.4). Hrany jsou označeny $\frac{l(e),c(e),a(e)}{f(e)}$.

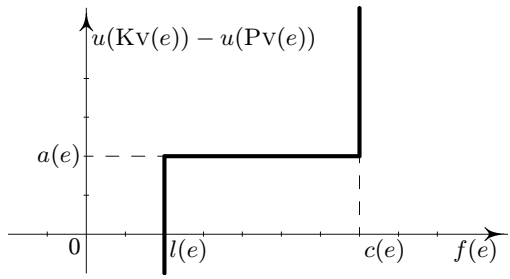
o zlepšující kružnici, pro hrany kružnice vyplývá:

pro hrany vpřed	$u(\mathrm{K}v(e)) - u(\mathrm{P}v(e))$	$\leq$	$a(e)$,
pro hrany vzad	$u(\mathrm{P}v(e)) - u(\mathrm{K}v(e))$	$\leq$	$-a(e)$,
sečtením dostaneme	0	$\leq$	cena zlepšující kružnice,

neboť suma potenciálových rozdílů podél kružnice se musí rovnat nule. □

8.7.2 Kilter-diagram. Kilter-podmínky (K) z předchozí věty lze pro každou hranu znázornit pomocí tzv. *kilter-diagramu*, viz obr. 8.12.

Velikost toku v hraně a rozdíl potenciálů mezi jejími vrcholy zde chápeme jako souřadnice bodu, který charakterizuje „stav hrany“. Hrana pak splňuje kilter-podmínky právě tehdy, když zmíněný bod leží na silně vyznačené tzv. *kilter-čáře*. V tomto smyslu říkáme, že hrana je *v kiltru* nebo *mimo kilter*.



Obrázek 8.12: Kilter-diagram.

Každá hrana má samozřejmě svůj vlastní kilter-diagram; poloha kilter-čáry je v něm určena omezeními l, c a cenou toku a a v průběhu výpočtu se nemění. Bod, který vyjadřuje „stav hrany“, se naopak během výpočtu pohybuje, a to při změně toku vodorovně a při změně potenciálů svisle.

Algoritmus Out of Kilter se snaží pomocí změn toku a potenciálů docílit, aby všechny hrany byly v kiltru.

8.7.3 Defekt. Abychom mohli měřit, jak se přibližujeme k optimálnímu řešení, zavedeme pojem defektu.[4] *Defekt hrany* definujeme jako vodorovnou vzdálenost bodu, který vyjadřuje stav hrany, od kilter-čáry. Přesněji, pro hranu e, která vede z vrcholu i do vrcholu j, definujeme defekt takto:

$$K(e) = \begin{cases} |f(e) - l(e)|, & \text{když} \quad u(j) - u(i) < a(e), \\ l(e) - f(e), & \text{když} \quad u(j) - u(i) = a(e) \ \text{ a } \ f(e) < l(e), \\ 0, & \text{když} \quad u(j) - u(i) = a(e) \ \text{ a } \ l(e) \leq f(e) \leq c(e), \\ f(e) - c(e), & \text{když} \quad u(j) - u(i) = a(e) \ \text{ a } \ c(e) < f(e), \\ |f(e) - c(e)|, & \text{když} \quad u(j) - u(i) > a(e). \end{cases}$$

[4] Jde o zcela jiný defekt než v 8.4.3.

Defekt hrany je vždy nezáporný a je roven nule právě tehdy, když hrana je v kiltru. Algoritmus Out of Kilter bude měnit tok a potenciály vždy tak, aby defekt v žádné hraně nevzrostl a aby alespoň v jedné hraně klesl.

8.7.4 Značkovací procedura. K hledání zlepšující kružnice se v algoritmu Out of Kilter používá značkovací procedura, která je podobná jako 8.3.5, ale podmínky značkování jsou složitější – závisí na polohách hran v jejich kilter-diagramech.

Značkování vždy začíná ve vrcholu z a cílem je označkovat vrchol s. Když se to podaří, je nalezena cesta ze z do s a tato cesta potom spolu s hranou mezi z a s vytvoří zlepšující kružnici.

1. [*Inicializace.*] Označkujeme vrchol z, ostatní vrcholy jsou beze značek.
2. [*Značkování vpřed.*] Existuje-li hrana e taková, že $\textsc{Pv}(e)$ má značku a $\textsc{Kv}(e)$ nikoli, pak, platí-li některá z podmínek

   (1) $$f(e) < l(e)\ ,$$

   (2) $$l(e) \le f(e) < c(e) \ \text{ a } \ u(\textsc{Kv}(e)) - u(\textsc{Pv}(e)) \ge a(e)\ ,$$

   označkujeme vrchol $\textsc{Kv}(e)$.
3. [*Značkování vzad.*] Existuje-li hrana e taková, že $\textsc{Kv}(e)$ má značku a $\textsc{Pv}(e)$ nikoli, pak, platí-li některá z podmínek

   (3) $$f(e) > c(e)\ ,$$

   (4) $$l(e) < f(e) \le c(e) \ \text{ a } \ u(\textsc{Kv}(e)) - u(\textsc{Pv}(e)) \le a(e)\ ,$$

   označkujeme vrchol $\textsc{Pv}(e)$.
4. [*Test ukončení.*] Je-li označkován vrchol s nebo nelze-li označkovat žádný další vrchol, značkování končí.

8.7.5 Řez bránící značkování. Pokud značkovací procedura 8.7.4 nemůže označkovat vrchol s, znamená to, že v řezu $W(A)$ určeném množinou označkovaných vrcholů A platí pro každou hranu jedna z podmínek

(1') $$e \in W^+(A) \ \text{ a } \ f(e) \ge c(e)\ ,$$

(2') $$e \in W^+(A) \ \text{ a } \ l(e) \le f(e) < c(e) \ \text{ a } \ u(\textsc{Kv}(e)) - u(\textsc{Pv}(e)) < a(e)\ ,$$

(3') $$e \in W^-(A) \ \text{ a } \ f(e) \le l(e)\ ,$$

(4') $$e \in W^-(A) \ \text{ a } \ l(e) < f(e) \le c(e) \ \text{ a } \ u(\textsc{Kv}(e)) - u(\textsc{Pv}(e)) > a(e)\ .$$

Algoritmus Out of Kilter se v takovém případě bude snažit změnit potenciály neoznačkovaných vrcholů tak, aby se všechny hrany řezu přiblížily (svislým pohybem) k vodorovné části kilter-čáry a přitom aby některá hrana tuto vodorovnou část dosáhla.

8.7.6 Algoritmus Out of Kilter.

Vstup: Orientovaný graf G, omezení toku l, c, jednotkové ceny toku a a výchozí cirkulace f (nemusí být přípustná, musí však splňovat Kirchhoffův zákon) a výchozí potenciály vrcholů u (zcela libovolná čísla).
Úkol: Buď změnit cirkulaci f a potenciály u tak, aby všechny hrany splňovaly kilter-podmínky (K) (pak podle věty 8.7.1 bude f nejlevnější cirkulací), anebo najít řez se zápornou kapacitou, který brání (podle věty 8.4.3) existenci přípustné cirkulace.

1. [*Test nalezení optima.*] Najdeme hranu h, která je mimo kilter. Pokud taková hrana neexistuje, výpočet končí, cirkulace f je přípustná a nejlevnější.

2. [*Určení směru značkování.*] Jestliže hrana h splňuje některou z podmínek (1), (2) (viz 8.7.4), pak položme $z := \text{Kv}(h)$ a $s := \text{Pv}(h)$. V opačném případě položme $z := \text{Pv}(h)$ a $s := \text{Kv}(h)$.

3. [*Značkování.*] Použijeme značkovací proceduru 8.7.4. Podaří-li se označkovat vrchol s, pokračujeme podle kroku 4, v opačném případě podle kroku 7.

4. [*Změna cirkulace.*] Zlepšující cestu (nalezenou ve značkovací proceduře) doplníme o hranu h, čímž vznikne zlepšující kružnice. Hranu h v ní budeme pokládat za hranu vpřed, pokud pro ni platí (1) nebo (2), nebo za hranu vzad, pokud splňuje (3) nebo (4).

5. Vypočteme kapacitu d zlepšující kružnice jako minimum přes všechny hrany kružnice z hodnot:
   $K(e)$ pro hrany s defektem $K(e) > 0$,
   $c(e) - f(e)$ pro hrany vpřed s defektem $K(e) = 0$,
   $f(e) - l(e)$ pro hrany vzad s defektem $K(e) = 0$.

6. Na hranách zlepšující kružnice změníme cirkulaci:
   $f(e) := f(e) + d$ pro hrany vpřed,
   $f(e) := f(e) - d$ pro hrany vzad.
   Pokračujeme podle kroku 1.

7. [*Pokus o změnu potenciálů.*] Prozkoumáme řez určený množinou označkovaných vrcholů A. Jestliže v řezu všechny hrany z $W^+(A)$ splňovaly (1') a současně všechny hrany z $W^-(A)$ splňovaly (3'), pak pokračujeme podle kroku 9, jinak podle kroku 8.

8. Vypočteme velikost změny potenciálů δ jako minimum z hodnot

$$|u(\text{Kv}(e)) - u(\text{Pv}(e)) - a(e)|$$

   přes všechny hrany řezu, které splňovaly (4') nebo (2'). Pokračujeme podle kroku 10.

9. Jestliže $f(h) = l(h)$ nebo $f(h) = c(h)$, vypočteme velikost změny potenciálů δ takto:
   když $f(h) = l(h)$, pak $\delta := u(z) - u(s) - a(h)$,
   když $f(h) = c(h)$, pak $\delta := u(z) - u(s) + a(h)$,
   a v obou případech pokračujeme podle kroku 10. V ostatních případech, tj. když $l(h) \neq f(h) \neq c(h)$, výpočet končí, přípustná cirkulace neexistuje a množina A označkovaných vrcholů určuje řez se zápornou kapacitou.

10. [*Změna potenciálů.*] U všech neoznačkovaných vrcholů zvýšíme potenciál o hodnotu δ, tj. provedeme pro ně $u(v) := u(v) + \delta$ a pokračujeme podle kroku 1.

8.7.7 Příklad. Výpočet algoritmem Out of Kilter předvedeme na síti z příkladu 8.6.4, obr. 8.11. Použijeme tutéž výchozí cirkulaci, výchozí potenciály zvolíme nulové: $u = (0, 0, 0, 0)$. Toky a potenciály v jednotlivých fázích výpočtu jsou znázorněny na obr. 8.13.

Kilter-diagramy jednotlivých hran jsou na obr. 8.14. Body, které v kilter--diagramech představují stav hrany v různých fázích výpočtu, jsou v diagramech očíslovány kurzívou. Změny stavů hran jsou naznačeny šipkami.

Průběh výpočtu byl tento:

1. $z = 2$, $s = 1$; označkována množina $\{2\}$; potenciál změněn na $u = (1, 0, 1, 1)$.

2. $z = 2$, $s = 1$; označkována množina $\{2, 4\}$; potenciál změněn na $u = (2, 0, 2, 1)$.

3. $z = 2$, $s = 1$; nalezena zlepšující kružnice 1–2–4–3–1 o kapacitě 1, změněn tok.

4. $z = 1$, $s = 3$; označkována množina $\{1\}$; potenciál změněn na $u = (2, 1, 3, 2)$.

5. Všechny hrany jsou v kiltru.

Všimněte si, že nejlevnější cirkulace byla nalezena již ve třetí iteraci, ale potenciály její optimálnost nepotvrzovaly. To se stalo až po jejich změně ve čtvrté iteraci. Dále si všimněte, že v prvých dvou iteracích se díky změnám potenciálů množina označkovaných vrcholů zvětšovala, dokud se ve třetí iteraci nepodařilo „prorazit“ cestu do vrcholu $s = 1$. Konečně poznamenejme, že hrana $(4, 1)$ se vrátila do svého výchozího stavu.

8.7.8 Věta. Algoritmus 8.7.6 se po konečně mnoha krocích zastaví. Zastaví-li se v kroku 1, pak f je nejlevnější přípustná cirkulace. Zastaví-li se v kroku 9, pak v síti přípustná cirkulace neexistuje.

Důkaz: K důkazu, že se algoritmus zastaví, stačí dokázat, že součet defektů všech hran klesne při každé změně toku i při každé změně potenciálů.

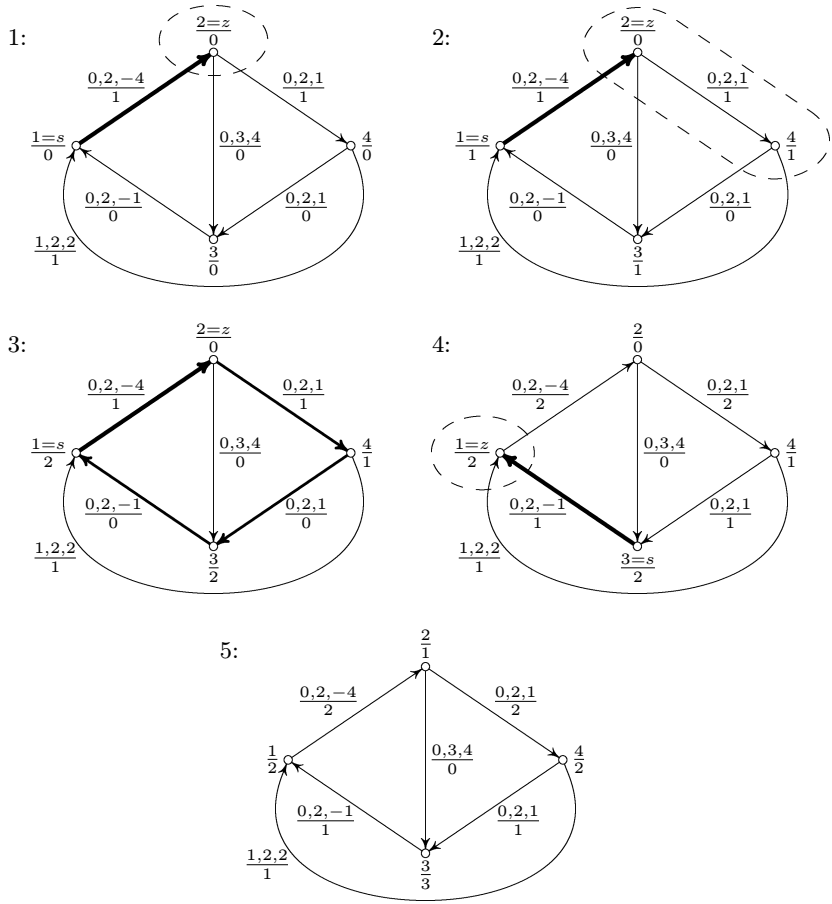
Pro změny toku prováděné v krocích 4 až 6 to zaručuje značkovací procedura: defekt hrany h klesne, defekty ostatních hran se nezvýší.

Změna potenciálů podle kroků 8 a 10 je prováděna tak, aby se alespoň jedna hrana dostala ve svém kilter-diagramu na vodorovnou část kilter-čáry, ale aby žádná

hrana tuto čáru nepřeběhla. (Ověřte!) To znamená, že alespoň pro jednu hranu defekt klesne na nulu a defekty ostatních hran se nezvýší.

Je-li vykonáván příkaz 9, pak pro všechny hrany, které jsou v řezu orientovány vpřed (tj. pro $W^+(A)$), platí $f(e) \geq c(e)$, pro všechny zbývající hrany řezu platí $f(e) \leq l(e)$. Součástí řezu přitom je i hrana h. Pokud pro ni neplatí rovnost (tj. když neplatí ani $f(h) = c(h)$, je-li h hranou vpřed, ani $f(h) = l(h)$, je-li h hranou vzad), pak má řez $W(A)$ zápornou kapacitu (srovnejte s důkazem věty 8.4.3), a proto přípustná cirkulace neexistuje. Výpočet tedy v tomto případě končí oprávněně.

Pokud pro hranu h rovnost platí, pak je ještě naděje, že přípustná cirkulace existuje. V tomto případě měníme potenciály tak, že hrany z $W^+(A)$, pro které



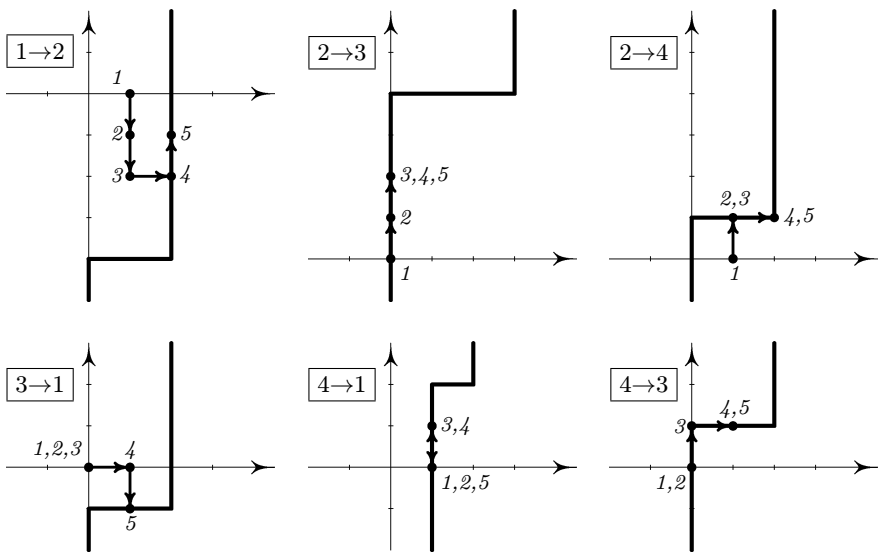
Obrázek 8.13: Postup výpočtu nejlevnější přípustné cirkulace algoritmem Out of Kilter (viz 8.7.7). Hrany jsou označeny $\frac{l(e),c(e),a(e)}{f(e)}$, vrcholy jsou označeny $\frac{i}{u(i)}$.

platí $f(e) \geq c(e)$, se budou ve svých kilter-diagramech pohybovat nahoru, zatímco hrany z $W^-(A)$, pro které platí $f(e) \leq c(e)$, se budou pohybovat dolů. Hrana h se přitom dostane přesně na „roh“ kilter-čáry. Defekt hrany h tedy klesne na nulu a defekty ostatních hran se buď nezmění, nebo klesnou o hodnotu $c(e) - l(e)$.

Po každém průchodu algoritmu krokem 2 tedy suma defektů klesne. Po konečně mnoha průchodech se tedy algoritmus musí zastavit buď v kroku 1, nebo v kroku 9. V prvém případě jsou všechny hrany v kiltru, podle věty 8.7.1 je tedy cirkulace f přípustná a nejlevnější. Druhý případ (zastavení v kroku 9) jsme již dokázali výše. □

8.7.9 Poznámky. Algoritmus Out of Kilter je výhodný v situacích, kdy řešíme posloupnost jen mírně odlišných úloh. To se vyskytuje např. při výpočtech metodou větví a mezí (viz 11.3 a 12.4.4), kdy k úloze postupně přidáváme dodatečné omezující podmínky, které činí dosavadní řešení nepřípustným. V takovém případě za výchozí cirkulaci i potenciály bereme výsledky z předchozího výpočtu.

Další výhodou algoritmu Out of Kilter je jeho univerzálnost. S jeho pomocí můžeme řešit i úlohy o maximálním toku nebo o nalezení přípustné cirkulace, je to ovšem méně efektivní než použití speciálních algoritmů.



Obrázek 8.14: Kilter-diagramy k příkladu 8.7.7.

Kapitola 9

Párování

9.1 Základní pojmy, úlohy a aplikace

9.1.1 Párování. Buď dán graf G. *Párování* v grafu G je taková množina hran $P \subseteq E(G)$, že žádné dvě hrany z množiny P nemají společný vrchol. Je zřejmé, že je-li P párování a $P' \subseteq P$, pak P' je také párování. Dokonce i prázdná množina je párováním, ale málo zajímavým.

O vrcholech, které jsou incidentní s některou hranou párování P, říkáme, že jsou párováním P *nasyceny* nebo též *pokryty*. O ostatních vrcholech říkáme, že jsou v párování P *volné*. Párování, které nasycuje všechny vrcholy grafu, nazýváme *perfektním párováním*.

9.1.2 Úlohy o párování. V souvislosti s párováním jsou studovány tři základní úlohy:

1. V daném grafu najít *maximální párování*, tj. párování, které obsahuje největší počet hran.

2. V grafu, jehož hrany jsou ohodnoceny cenami, najít *nejlevnější maximální párování*, tj. nejlevnější párování ze všech, která jsou maximální.

3. V grafu, jehož hrany jsou ohodnoceny cenami, najít *nejdražší párování*, tj. párování s největším součtem cen.

Úloha o nejdražším párování je nejobecnější. Úlohu o nejlevnějším maximálním párování lze na ni převést změnou cen, viz cvičení 9.1.8.

Úloha o nejlevnějším maximálním párování je častá v aplikacích. Jejím speciálním případem je tzv. *přiřazovací úloha*. Jde v ní o nalezení nejlevnějšího perfektního párování v úplném bipartitním grafu, jehož strany mají stejné počty vrcholů.

Všechny uvedené úlohy o párování lze řešit v polynomiálním čase (viz 1.10.6, str. 30, a 11.1.5, str. 189). Algoritmy i teoretické úvahy pro párování v bipartitních grafech jsou jednodušší, proto se jimi budeme zabývat zvlášť v oddílech 9.3 a 9.4.

9.1.3 Příklad: Letecká bitva o Anglii. V letecké bitvě o Anglii v roce 1940 bojovala na straně Anglie řada pilotů různých národností. Královské letectvo mělo letadla, která vyžadovala dva piloty. Někteří piloti spolu ovšem nemohli letět pro jazykové potíže nebo pro rozdílnost výcviku. Kolik letadel (tj. dvojic pilotů) může za těchto omezení vzlétnout najednou?

Tento problém by bylo možno řešit nalezením maximálního párování v grafu, jehož vrcholy jsou piloti a jehož hrany spojují piloty, kteří mohou letět spolu.

9.1.4 Příklad zahrádkářský. Představme si ovocnou zahradu, ve které je $2n$ stromků a jedna hromada hnoje. Naším úkolem je přivézt půl kolečka hnoje ke každému stromku. Abychom šetřili časem, chceme vždy vézt k některému stromku plné kolečko, polovinu jeho obsahu složit a pokračovat s poloprázdným kolečkem k jinému stromku. Kromě času chceme šetřit i silami a chceme, aby např. součet drah vykonaných s plným kolečkem byl co nejmenší.

Tuto úlohu lze převést na hledání nejlevnějšího perfektního párování v úplném grafu o $2n$ vrcholech. Každý vrchol odpovídá jednomu stromku. Cena hrany bude úměrná námaze spojené s jízdou plného kolečka k bližšímu z obou stromků, s následující jízdou poloprázdného kolečka ke druhému z nich a s návratem prázdného kolečka k hromadě hnoje. Je zřejmé, že každý stromek (vrchol) musí být zahrnut do přesně jedné jízdy, hrany odpovídající těmto jízdám musí tedy tvořit perfektní párování.

9.1.5 Cvičení. Jak byste řešili předchozí úlohu pro lichý počet stromků?

9.1.6 Příklad: Taneční. Předpokládejme, že v tanečních má každý hoch přesně k přítelkyň a každá dívka přesně k přátel, přičemž $k > 0$. Je možno uspořádat tanec, při kterém každý hoch i dívka tančí s některým ze svých přátel? (Zde poněkud nerealisticky předpokládáme, že vztah přátelství je symetrický.)

Vztah přátelství můžeme znázornit bipartitním grafem, v němž každý vrchol má stupeň k. V tomto grafu hledáme perfektní párování. Ukážeme, že za uvedených podmínek lze existenci perfektního párování zaručit.

9.1.7 Přiřazovací úloha. Klasická formulace přiřazovací úlohy, totiž přiřazování pracovních úkolů pracovníkům, byla uvedena v 8.2.5, str. 137. Kromě řady dalších přímých aplikací se přiřazovací úloha vyskytuje i jako podúloha při řešení složitějších úloh, viz např. úloha čínského pošťáka 10.3, str. 184, a problém obchodního cestujícího 12.4.4, str. 208.

Řešením přiřazovací úlohy se budeme zabývat v 9.4, str. 173.

9.1.8 Cvičení. Úlohu o nejlevnějším maximálním párování lze převést na úlohu o nejdražším párování změnou cen hran. Navrhněte způsob, jak to udělat.

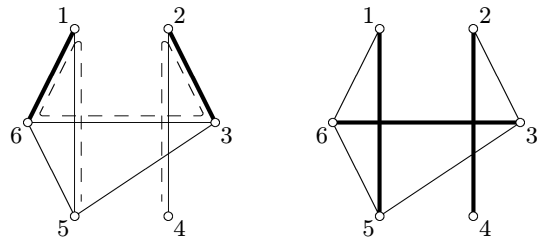
9.1.9 Poznámka. Úlohy o párování můžeme chápat jako hledání soustavy disjunktních dvojic vrcholů. Podobné úlohy týkající se disjunktních trojic jsou podstatně obtížnější (NP-těžké). Znamená to např., že kdybychom v zahrádkářské úloze

9.1.4 snížili dávku hnoje na třetinu kolečka, byl by rozvoz hnoje snazší, ale najít optimální řešení by bylo složitější.

9.2 Párování v obecných grafech

9.2.1 Střídavé cesty a kružnice. Buď dán graf G a v něm párování P. *Střídavá cesta* (někdy též *alternující cesta*) vzhledem k párování P je taková neorientovaná cesta, že její hrany střídavě leží v P a neleží v P a je-li krajní vrchol cesty nasycen v párování P, pak hrana, která jej nasycuje, je částí cesty.

Příklad střídavé cesty je na obr. 9.1 vlevo vyznačen čárkovaně. Hrany, které náleží k párování, jsou nakresleny silně. Poznamenejme, že cesta spojující vrcholy 1, 6, 3 není střídavou cestou, ale její prodloužení 1, 6, 3, 2 již střídavou cestou je.



Obrázek 9.1: Střídavá cesta a změna párování.

Střídavá kružnice vzhledem k párování P je kružnice, jejíž hrany střídavě leží a neleží v párování P. Střídavá kružnice má vždy sudý počet hran.

Pomocí střídavých cest a kružnic lze párování snadno měnit tím, že u hran, které leží na cestě nebo kružnici, změníme příslušnost k párování. Přesněji, je-li H množina hran tvořících střídavou cestu nebo kružnici vzhledem k párování P, vytvoříme nové párování P' takto:

$$\begin{aligned} &\text{jestliže} \quad e \in H, \ \text{pak} \ \ e \in P' \Leftrightarrow e \notin P, \\ &\text{jestliže} \quad e \notin H, \ \text{pak} \ \ e \in P' \Leftrightarrow e \in P. \end{aligned}$$

Takovou změnu párování budeme nazývat *změnou podél střídavé cesty* nebo *kružnice*. Výsledek změny párování podél střídavé cesty je na obr. 9.1 vpravo.

9.2.2 Věta. Buď dán prostý graf G a v něm libovolné párování P_1. Pak pro každé párování P_2 v grafu G existuje soustava vrcholově disjunktních střídavých cest a střídavých kružnic taková, že změnami podél všech těchto cest a kružnic lze z párování P_1 získat párování P_2.

Poznámka: Poněvadž střídavé cesty a kružnice nemají žádný společný vrchol (jsou vrcholově disjunktní), lze změny provádět v libovolném pořadí.

Důkaz: Uvažujme faktor G' grafu G s množinou hran $(P_1 \setminus P_2) \cup (P_2 \setminus P_1)$. Graf G' tedy obsahuje právě ty hrany grafu G, které leží přesně v jednom z obou párování (tj. nikoli v obou).

Pro každý vrchol $x \in V(G') = V(G)$ nastane jedna ze čtyř možností:

1. Jestliže x není nasycen ani v P_1, ani v P_2, je izolovaným vrcholem v G'.
2. Je-li vrchol x nasycen jen v jednom z párování P_1, P_2, má v grafu G' stupeň 1.
3. Je-li x nasycen v P_1 i P_2 touž hranou, pak tato hrana neleží v G', a vrchol x je tedy v G' izolovaným vrcholem.
4. Je-li x nasycen v P_1 i P_2 různými hranami $e_1 \in P_1$ a $e_2 \in P_2$, pak obě tyto hrany leží v G' a stupeň vrcholu x je roven 2.

Každý vrchol má tedy v grafu G' stupeň nejvýše 2. Z toho plyne, že graf G' má komponenty souvislosti tří typů:

1. izolovaný vrchol,
2. kružnice sudé délky, jejíž hrany střídavě leží v P_1 a P_2,
3. cesta, jejíž hrany střídavě leží v P_1 a P_2 a jejíž krajní vrcholy jsou různé, přičemž každý z nich je nasycen v jednom z obou párování P_1, P_2.

Komponenty souvislosti grafu G' přímo určují střídavé cesty a kružnice. Provedením změn párování P_1 podél všech těchto cest a kružnic dostaneme párování P_2. □

9.2.3 Věta. Párování P v grafu G je maximální právě tehdy, když v grafu G vzhledem k párování P neexistuje střídavá cesta spojující dva volné vrcholy.

Důkaz: Je-li párování P maximální, pak vzhledem k němu nemůže existovat střídavá cesta spojující volné vrcholy. Kdyby totiž existovala, mohli bychom párování podél ní zvětšit.

Jestliže naopak párování P není maximální, vezměme nějaké maximální párování (nějaké určitě existuje) a označme je P_1. Podle věty 9.2.2 existuje soustava střídavých cest a kružnic taková, že změnami podél nich získáme párování P_1 z párování P. Poněvadž $|P_1| > |P|$, musí některá z těchto změn zvětšovat párování, musí tedy být změnou podél střídavé cesty s volnými krajními vrcholy (jiné změny nezvětšují počet hran v párování). □

9.2.4 Cena střídavé cesty a kružnice. Uvažujme graf G, jehož hrany jsou ohodnoceny cenami $c : E(G) \to \mathbb{R}$. Dále mějme v grafu G párování P a vzhledem k němu střídavou cestu, popř. kružnici. Množinu hran této cesty, popř. kružnice, označme H.

Cenu střídavé cesty, popř. *kružnice*, definujeme jako

$$C = \sum_{e \in H \setminus P} c(e) - \sum_{e \in H \cap P} c(e) \,.$$

Má-li střídavá cesta, popř. kružnice, cenu C a provedeme-li podél ní změnu, pak cena párování vzroste o hodnotu C.

9.2.5 Věta. Párování P v grafu G je nejdražší právě tehdy, když v grafu G vzhledem k párování P neexistuje ani střídavá cesta, ani střídavá kružnice s kladnou cenou.

Důkaz je podobný důkazu věty 9.2.3 a je přenechán čtenáři jako cvičení. □

9.2.6 Algoritmy pro párování v obecných grafech. Všeobecně lze k hledání maximálního i nejdražšího párování využít metody založené na větách 9.2.3 a 9.2.5. Základní schéma algoritmů je toto:

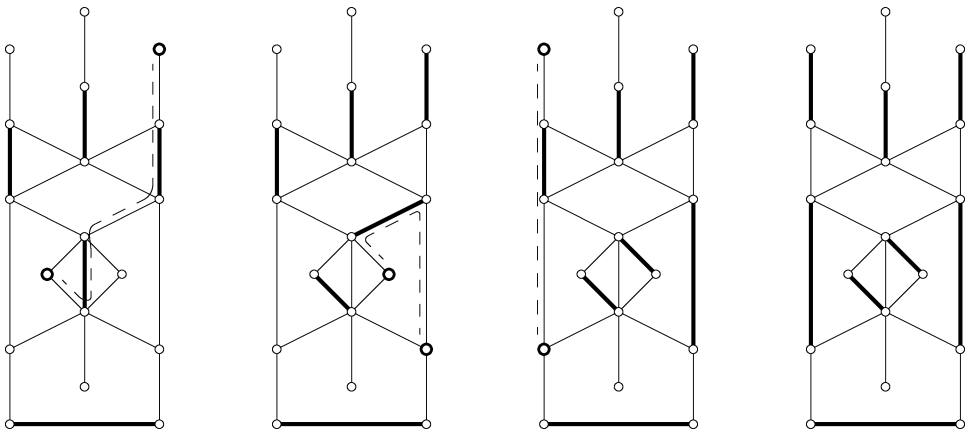
Zvolíme libovolné výchozí párování P a hledáme, zda vzhledem k němu existuje střídavá cesta, popř. kružnice, která by dovolila párování zlepšit. Jestliže neexistuje, pak podle věty 9.2.3, popř. 9.2.5, je párování P již optimální. Jestliže existuje, změníme (tj. zlepšíme) párování a znovu hledáme další střídavou cestu, popř. kružnici.

Bohužel, hledání potřebných střídavých cest a kružnic není tak jednoduché, jak se na první pohled zdá. Běžné metody prohledávání grafů popsané v kap. 3 a založené na značkování vrcholů lze jednoduše upravit pro hledání střídavých cest pouze v bipartitních grafech (viz 9.3).

V obecných grafech sice lze pro hledání střídavých cest použít backtracking (viz 11.2, str. 191), ten však je ve větších grafech neúnosně pomalý.

Rychlé (polynomiální) algoritmy pro párování v obecných grafech existují, pracují v čase $O(n^3)$, ale jsou poměrně komplikované a v případě nejdražšího párování se navíc opírají o tzv. dualitu v lineárním programování. Z tohoto důvodu nastíníme pouze hlavní myšlenky rychlých algoritmů, a to pouze pro maximální párování (viz 9.2.8). Polynomiální algoritmy pro maximální i nejdražší párování jsou popsány např. v knihách [Plesník83], [Chr75] a [Lawler76].

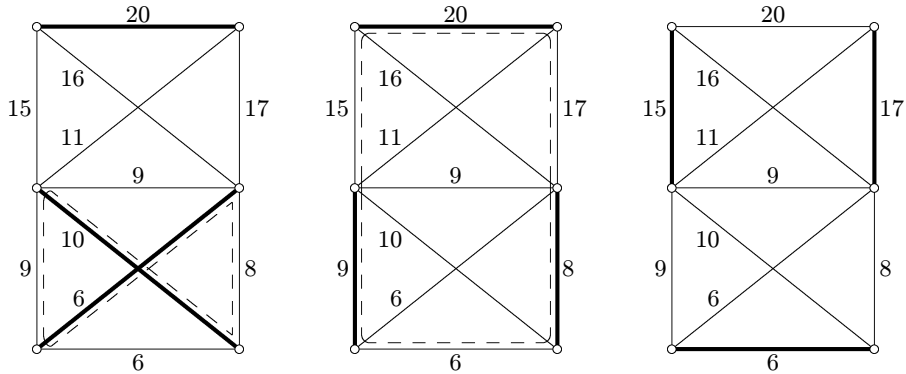
9.2.7 Příklad. Na obr. 9.2 a 9.3 jsou příklady hledání maximálního a nejdražšího párování.



Obrázek 9.2: Výpočet maximálního párování.

Výchozí párování na obr. 9.2 vlevo bylo získáno náhodným přidáváním hran. Všimněte si, že k tomuto párování už nelze přidat žádnou další hranu, přesto má dost daleko k maximálnosti. Střídavé cesty mezi volnými vrcholy jsou vyznačeny čárkovaně. Všimněte si také, že výsledné maximální párování není perfektní.

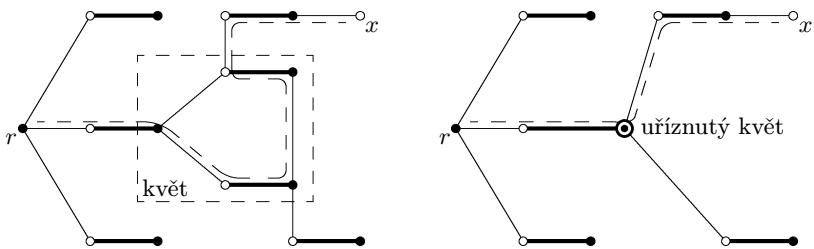
Výchozí párování na obr. 9.3 vlevo bylo získáno „hladovým“ postupem: postupně jsme přidávali vždy nejdražší přijatelnou hranu. Střídavé cesty i kružnice byly hledány ručně s využitím backtrackingu. Všimněte si, že nejdražší párování na obr. 9.3 vpravo jsme mohli získat také přímo z výchozího párování, kdybychom zvolili jinou střídavou kružnici.



Obrázek 9.3: Výpočet nejdražšího párování.

9.2.8 Rychlý algoritmus pro maximální párování. Naznačíme hlavní myšlenky rychlého algoritmu pro hledání střídavé cesty, jejíž krajní vrcholy jsou volné v párování P. Podrobnější popis nebo jiné postupy lze najít v [Kučera83], [Plesník83], [Chr75], [Lawler76], [SwTh81].

Začínáme v některém volném vrcholu r, který nazveme *kořenem*, a vytváříme tzv. *střídavý strom*, který má tu vlastnost, že každá cesta v tomto stromě, která začíná v r, je střídavou cestou. Strom vytváříme pomocí značkování vrcholů, přičemž používáme dva druhy značek: vrcholy stromu označujeme střídavě jako *vnější* a *vnitřní*, přičemž kořen r je vnější. V příkladě na obr. 9.4 jsou vnější vrcholy kresleny jako plné tečky.



Obrázek 9.4: Uříznutí květu a střídavá cesta procházející květem.

Jestliže najdeme hranu, která spojuje vnější vrchol s nějakým volným vrcholem různým od r, máme střídavou cestu s volnými krajními vrcholy a můžeme zvětšit párování.

Jestliže najdeme hranu, která spojuje dva vnější vrcholy (a která tedy neleží v párování), našli jsme kružnici liché délky, která bývá nazývána *květem*. Květ kazí vlastnost střídavého stromu, totiž že cesta začínající v r je střídavou cestou. Proto všechny vrcholy květu nahradíme jediným vrcholem a hrany, které vedly z vnějšku do květu, vedeme do tohoto vrcholu. Tato operace bývá nazývána *uříznutím květu*. Lze dokázat, že střídavá cesta z r do některého volného vrcholu x v původním grafu existuje právě tehdy, když existuje i v upraveném grafu. Dále tedy pokračujeme s upraveným grafem (viz obr. 9.4) a dále se snažíme střídavý strom rozšířit.

Jestliže střídavý strom nelze dále rozšířit a nenastává žádný z předchozích případů, tj. když z vnějších vrcholů vedou hrany pouze do vnitřních vrcholů, odstraníme z grafu celý strom a všechny hrany s ním incidentní. Lze totiž dokázat, že část párování, která ležela ve stromě, je částí některého maximálního párování. To znamená, že vynechanou částí grafu se již není třeba dále zabývat. (Ověřte na příkladě!)

Tento postup opakujeme, dokud buď nezredukujeme graf na prázdný (pak stávající párování je již maximální), nebo dokud nenalezneme střídavou cestu s volnými krajními vrcholy. V tomto druhém případě lze stávající párování zvětšit. Jestliže však střídavá cesta prochází přes vrchol, který vznikl uříznutím květu, musíme nejprve květ obnovit a najít v něm chybějící část střídavé cesty. To se může i několikrát opakovat, neboť uříznutý květ mohl být součástí dalšího květu atd.

Praktické provedení tohoto algoritmu je relativně komplikované. Uřezávání květů se neprovádí změnami grafu, ale pomocí soustavy odkazů ve vhodné datové struktuře. Detaily i důkaz správnosti lze najít v [Plesník83], [Chr75], [Lawler76], poněkud odlišný algoritmus je uveden v [Kučera83]. Časové nároky těchto algoritmů jsou $O(n^3)$, popř. až $O(n^2\sqrt{n})$.

Poznamenejme, že v grafu, který neobsahuje kružnici liché délky (tj. který je bipartitní), se květ nemůže vyskytnout.

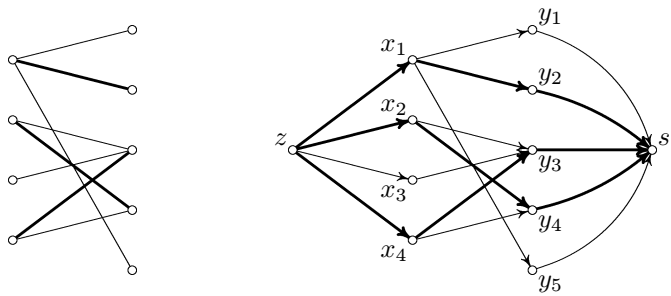
9.3 Párování v bipartitních grafech

Teorie i algoritmy pro párování v bipartitních grafech jsou podstatně jednodušší než v obecném případě, a to díky těsnému vztahu bipartitního párování k tokům v síti.

9.3.1 Převod bipartitního párování na tok v síti. Mějme bipartitní graf G se stranami X, Y. Orientaci hran tohoto grafu zvolme tak, aby všechny hrany vedly z X do Y. K takto vzniklému orientovanému grafu přidejme dva vrcholy, umělý zdroj z a umělý spotřebič s (viz obrázek 9.5 na následující straně). Dále pro každý vrchol $x \in X$ přidejme hranu ze zdroje z do x a pro každý vrchol $y \in Y$ přidejme hranu z y do spotřebiče s. Kapacity těchto přidaných hran jsou jednotkové, kapacity původních hran jsou větší nebo rovny jedné. Dolní omezení toku jsou ve všech hranách nulová.

Párování v původním grafu vzájemně jednoznačně odpovídají celočíselným tokům od zdroje ke spotřebiči: párování je tvořeno těmi hranami původního grafu,

kterými teče nenulový (tj. jednotkový) tok. Maximální párování lze tedy hledat algoritmem pro maximální tok, nejdražší párování algoritmem pro nejlevnější tok (po obrácení znamének cen a po eventuálním přidání návratové hrany).



Obrázek 9.5: Převod úlohy o párování v bipartitním grafu na úlohu o toku v síti.

9.3.2 Speciální algoritmy pro maximální párování v bipartitním grafu jsou ve své podstatě jen přeformulováním a zjednodušením algoritmů určených pro maximální tok. Zlepšující cestě, která se používá ke změnám toku v transportní síti, odpovídá v původním grafu střídavá cesta s volnými krajními vrcholy.

Přímým přeformulováním značkovací procedury 8.3.5 dostaneme tento návod:

1. [*Inicializace.*] Označkujeme všechny volné vrcholy z množiny X. Ostatní vrcholy jsou beze značek.

2. [*Test nalezení cesty.*] Je-li označkován některý volný vrchol z množiny Y, značkovací procedura končí. Je nalezena střídavá cesta s volnými krajními vrcholy a podél ní lze zvětšit počet hran v párování.

3. [*Pokus o značkování vpřed.*] Jestliže existuje hrana $e \notin P$ vedoucí z označkovaného vrcholu $x \in X$ do neoznačkovaného vrcholu $y \in Y$, pak označkujeme vrchol y a pokračujeme podle kroku 2. Neexistuje-li taková hrana, pokračujeme podle kroku 4.

4. [*Pokus o značkování vzad.*] Jestliže existuje hrana $e \in P$ z neoznačkovaného vrcholu $x \in X$ do označkovaného vrcholu $y \in Y$, pak označkujeme vrchol x a pokračujeme podle kroku 3. Neexistuje-li taková hrana, značkovací procedura končí, stávající párování je maximální.

Tento postup lze ještě dále zjednodušit takto:

1. [*Inicializace.*] Označkujeme všechny volné vrcholy z množiny X. Ostatní vrcholy jsou beze značek.

2. [*Pokus o značkování vpřed.*] Jestliže existuje hrana $e \notin P$ vedoucí z označkovaného vrcholu $x \in X$ do neoznačkovaného vrcholu $y \in Y$, pak označkujeme vrchol y a pokračujeme podle kroku 3. Neexistuje-li taková hrana, značkování končí, střídavá cesta spojující volné vrcholy neexistuje.

3. [*Test nalezení cesty.*] Je-li právě označkovaný vrchol y volný, značkovací procedura končí, je nalezena střídavá cesta s volnými krajními vrcholy. Je-li vrchol y párováním P nasycen, pokračujeme podle kroku 4.

4. [*Značkování vzad.*] Označkujeme vrchol (ležící v množině X), který je spárován s vrcholem y, a pokračujeme podle kroku 2.

Jestliže takto upravená značkovací procedura skončí, aniž nalezla střídavou cestu s volnými krajními vrcholy, pak stávající párování již je maximální. Důkaz tohoto důležitého tvrzení vyplývá z věty 9.2.3 nebo také z analogie s toky v sítích (věta 8.3.7).

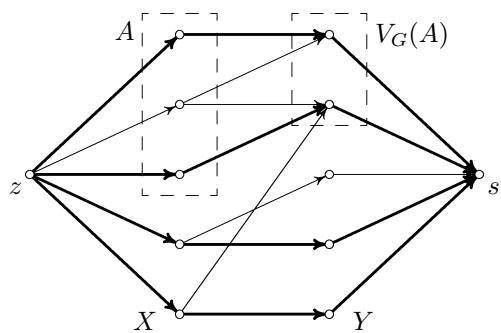
9.3.3 Königova věta. Buď dán bipartitní graf G se stranami X a Y. Počet hran v maximálním párování v grafu G je roven

$$\min_{A \subseteq X} (|X \setminus A| + |V_G(A)|) \,. \tag{9.1}$$

Poznámka: Tato věta má pro párování v bipartitních grafech podobný význam jako Fordova-Fulkersonova věta 8.3.8 pro toky v sítích. Množinu A, pro kterou ve výraze (9.1) nastává minimum, lze sestrojit pomocí značkovací procedury.

Důkaz: Ke grafu G sestrojme transportní síť T podle 9.3.1.

Vezměme libovolnou množinu $A \subseteq X$ a uvažujme řez $W(Z)$ určený množinou vrcholů $Z = \{z\} \cup A \cup V_G(A)$. (Viz obr. 9.6.) Tento řez obsahuje $|X \setminus A|$ hran z množiny $E^+(z)$ a $|V_G(A)|$ hran z množiny $E^-(s)$. Tyto hrany mají jednotkové kapacity, ostatní hrany kapacitu řezu neovlivňují. Kapacita řezu $W(Z)$ je tedy rovna $|X \setminus A| + |V_G(A)|$. Proto žádné párování nemůže mít více než $|X \setminus A| + |V_G(A)|$ hran.



Obrázek 9.6: K důkazu Königovy věty.

Zbývá dokázat, že existuje množina $A \subseteq X$, pro kterou nastane rovnost. Podle Fordovy-Fulkersonovy věty 8.3.8 je velikost maximálního toku, tj. velikost párování, rovna kapacitě minimálního řezu. Dokážeme, že minimální řez má popsaný tvar $W(Z)$, kde $Z = \{z\} \cup A \cup V_G(A)$ pro nějakou množinu $A \subseteq X$.

Uvažujme tedy některý maximální tok f a použijme na něj značkovací proceduru 8.3.5, str. 143. Výslednou množinu označkovaných vrcholů označme Z a dále označme $A = X \cap Z$. (Viz obr. 9.6 na předchozí straně.)

Uvažujme libovolný vrchol $x \in A$. Jestliže vrcholem x neprotéká žádný tok, pak vrchol x byl označkován po hraně ze zdroje z a určitě jsou označkovány všechny vrcholy množiny $V_G(x)$. Jestliže vrcholem x tok protéká, pak vrchol x nemohl být označkován po hraně ze zdroje (neboť je nasycena tokem), musel tedy být označkován po (jediné) hraně, kterou tok z vrcholu x odtéká. Pak ovšem musel být koncový vrchol této hrany označkován dříve než x. Ostatními hranami z $E^+(x)$ tok neteče, proto jejich koncové vrcholy jsou jistě označkovány (kdyby nebyly, bylo by možno dále značkovat). Tím je dokázáno, že $V_G(A) \subseteq Y \cap Z$. Žádný vrchol z množiny $Y \setminus V_G(A)$ nemůže být označkován, neboť do něj nevede žádná hrana z označkovaného vrcholu (tj. z množiny A). Platí tedy

$$V_G(A) = Y \cap Z \quad \text{a} \quad Z = \{z\} \cup A \cup V_G(A)\ ,$$

a pro množinu A tedy nastává rovnost. □

9.3.4 Hallova věta. Nechť G je bipartitní graf se stranami X a Y. V grafu G existuje párování, které nasycuje množinu X právě tehdy, když

(9.2) množina X neobsahuje žádný izolovaný vrchol a pro každou množinu $A \subseteq X$ platí $|A| \le |V_G(A)|$.

Důkaz: Pokud existuje párování nasycující množinu X, pak podmínka (9.2) triviálně platí. Zajímavá je pouze opačná implikace.

Z podmínky (9.2) plyne, že pro každou množinu $A \subseteq X$ platí $|X \setminus A| + |V_G(A)| = |X| - |A| + |V_G(A)| \ge |X|$. Podle Königovy věty je tedy počet hran v maximálním párování větší nebo roven $|X|$. Větší být ovšem nemůže, proto je roven $|X|$. □

9.3.5 Věta. Jestliže v bipartitním grafu G se stranami X, Y pro každé $x \in X$ a každé $y \in Y$ platí $d(x) \ge d(y)$ a graf G má alespoň jednu hranu, pak v G existuje párování, které nasycuje všechny vrcholy množiny X.

Důkaz: Označme

$$d_X = \min_{x \in X} d(x) \qquad \text{a} \qquad d_Y = \max_{y \in Y} d(y)\ .$$

Platí tedy $d_X \ge d_Y \ge 1$. Vezměme nyní libovolnou množinu $A \subseteq X$. O počtu hran v řezu $W(A)$ platí

$$|W(A)| \ge |A| \cdot d_X$$

a také

$$|W(A)| \le |W(V(A))| \le |V(A)| \cdot d_Y\ .$$

Odtud $|A| \cdot d_X \le |V(A)| \cdot d_Y$, a poněvadž $d_X \ge d_Y$, platí $|A| \le |V(A)|$. Toto platí pro libovolnou množinu $A \subseteq X$. Podle Hallovy věty tedy existuje párování, které nasycuje množinu X. □

9.3.6 Cvičení. Dokažte, že z předpokladů předchozí věty plyne, že $|X| \leq |Y|$. Charakterizujte grafy, pro které jsou splněny předpoklady věty a platí $|X| = |Y|$.

9.3.7 Cvičení. Pomocí předchozí věty dokažte, že v příkladu 9.1.6 (taneční) existuje řešení, tj. perfektní párování.

9.3.8 Věta. V každém bipartitním grafu, který má alespoň jednu hranu, existuje párování, které nasycuje všechny vrcholy s maximálním stupněm.

Důkaz: Označme k maximální stupeň vrcholu. Zřejmě $k > 0$. Graf doplníme novými hranami a vrcholy tak, aby byl bipartitní a všechny vrcholy měly stupeň k. V takto vzniklém grafu mají obě strany stejný počet vrcholů a podle věty 9.3.5 v něm existuje perfektní párování. Nyní ze zvětšeného grafu vypustíme všechny přidané hrany a vrcholy. Tím získáme párování v původním grafu. Toto párování nasycuje všechny vrcholy stupně k, neboť k těmto vrcholům nebyla přidána žádná hrana. □

9.3.9 Věta. Nechť v bipartitním grafu G se stranami X, Y existují dvě párování P_1, P_2. Nechť párování P_1 nasycuje množinu vrcholů $X_1 \subseteq X$ a párování P_2 nasycuje množinu $Y_2 \subseteq Y$. Potom existuje párování $P \subseteq P_1 \cup P_2$, které nasycuje obě množiny $X_1 \subseteq X$ i $Y_2 \subseteq Y$.

Důkaz: Označme G' faktor grafu G s množinou hran $P_1 \cup P_2$. V grafu G' mají všechny vrcholy stupeň 0, 1 nebo 2, komponenty souvislosti grafu G' jsou tedy izolované vrcholy, cesty a kružnice.

Položme $P = P_1$. Jestliže P nasycuje všechny vrcholy z Y_2, pak P je hledané párování. Jestliže P nenasycuje některý vrchol $y \in Y_2$, pak vrchol y je krajním vrcholem cesty, která je komponentou souvislosti grafu G'. Tuto cestu můžeme pokládat za střídavou cestu vzhledem k párování P; proveďme změnu P podél této cesty. Nyní je vrchol y nasycen v P a všechny vnitřní vrcholy cesty také zůstávají nasyceny v P. Označme v druhý krajní vrchol cesty. Jestliže $v \in X$, pak je nyní vrchol y nasycen v P. Jestliže $v \in Y$, pak sice není nasycen v P, ale není toho třeba, neboť v tomto případě byl vrchol v nasycen pouze v P_1, a tedy $v \notin Y_2$. Opakováním tohoto postupu budeme měnit párování P tak, že bude nasycovat stále větší počet prvků množiny Y_2 a zároveň celou množinu X_1. □

9.4 Přiřazovací úloha

Uvedeme tzv. maďarský algoritmus pro řešení přiřazovací úlohy, tj. pro hledání nejlevnějšího perfektního párování v úplném bipartitním grafu $K_{n,n}$, jehož hrany jsou ohodnoceny cenami. Algoritmus je založen na změnách cen, které nemají vliv na výsledné řešení.

9.4.1 Matice cen a její transformace. Buď G úplný bipartitní graf se stranami X, Y takový, že $|X| = |Y| = n$. Předpokládejme, že vrcholy obou stran X,

Y jsou očíslovány čísly $1, 2, \ldots, n$. Hrany grafu G můžeme bez újmy na obecnosti ztotožnit s uspořádanými dvojicemi vrcholů $(x, y) \in X \times Y$. Ceny hran pak můžeme přirozeným způsobem uspořádat do tvaru matice C, jejíž libovolný prvek $c(i, j)$ je cenou hrany (i, j).

Vezměme libovolné ohodnocení vrcholů grafu G reálnými čísly $p(v)$ pro $v \in \in V(G)$ a na základě tohoto ohodnocení definujme matici transformovaných cen C_p předpisem

$$c_p(i, j) = c(i, j) - p(i) - p(j) \ .$$

Transformací cen se optimální řešení přiřazovací úlohy nezmění, neboť cena každého perfektního párování se sníží o tutéž hodnotu

$$\sum_{i \in X} p(i) + \sum_{j \in Y} p(j) \ = \sum_{v \in V(G)} p(v) \ .$$

(Ověřte na příkladě!)

Ohodnocení vrcholů p nazveme *přípustným ohodnocením*, jsou-li všechny transformované ceny nezáporné, tj. $c_p(i, j) = c(i, j) - p(i) - p(j) \geq 0$.

Je-li p přípustné ohodnocení vrcholů, pak *grafem rovnosti* G_p nazveme faktor grafu G, který obsahuje právě ty hrany grafu G, jejichž transformovaná cena je nulová, tj. $c(i, j) = p(i) + p(j)$. Poznamenejme, že všechny ostatní hrany grafu G mají transformované ceny kladné. Platí tedy tato věta:

9.4.2 Věta. Jestliže graf rovnosti G_p určený ohodnocením vrcholů p obsahuje perfektní párování P, pak párování P je optimálním řešením přiřazovací úlohy.

Důkaz: Párování P má v transformovaných cenách C_p nulovou cenu. Žádné jiné párování nemůže být levnější, protože matice C_p neobsahuje záporné prvky. P je tedy nejlevnějším párováním vzhledem k transformovaným cenám C_p. Pak je ovšem nejlevnější i vzhledem k původním cenám C. □

Poznámka: Při notné dávce štěstí bychom mohli párování P a ohodnocení p uhodnout a pomocí této věty pak už jen ověřit, že jsme hádali správně.

9.4.3 Maďarský algoritmus pro přiřazovací úlohu. Hlavní myšlenka tzv. maďarského algoritmu spočívá v nalezení takového přípustného ohodnocení vrcholů, že v příslušném grafu rovnosti existuje perfektní párování.

Výchozí transformovanou matici cen a přípustné ohodnocení vrcholů získáme tím, že od každého řádku původní matice cen odečteme minimální cenu. Tím získáme v každém řádku alespoň jednu nulu. Tutéž operaci můžeme provést také se sloupci, pak i každý sloupec obsahuje alespoň jednu nulu. K takto získané transformované matici cen C_p sestrojíme graf rovnosti G_p a v něm najdeme maximální párování (viz 9.3.2, str. 170). Získáme-li takto perfektní párování, je úloha vyřešena.

Jestliže v grafu G_p perfektní párování neexistuje, pak v něm podle Hallovy věty existuje množina $A \subseteq X$ taková, že $|A| > |V_{G_p}(A)|$. Tuto množinu najde značkovací procedura 9.3.2 při neúspěšném pokusu o nalezení střídavé cesty s volnými krajními vrcholy: $A = Z \cap X$, kde Z je množina všech označkovaných vrcholů.

Abychom mohli v grafu rovnosti G_p zvětšit párování, potřebujeme označkovat další vrchol. K tomu je třeba přidat ke grafu G_p některou hranu, která vede z množiny A do neoznačkovaného vrcholu v množině Y. Potřebujeme tedy změnit ceny, ale chceme to udělat tak, abychom tuto změnu mohli pokládat za transformaci cen ve smyslu 9.4.1.

Zvýšíme tedy $p(i)$ na množině A a zároveň snížíme $p(j)$ na množině $V_{G_p}(A)$ o tutéž hodnotu d takovou, aby se vynulovala transformovaná cena na některé hraně vedoucí z A do $Y \setminus V_{G_p}(A)$ a přitom aby všechny transformované ceny zůstaly nezáporné. Prakticky to vypadá tak, že hodnota d bude rovna minimu (ze současných transformovaných) cen v průsečíku označkovaných řádků a neoznačkovaných sloupců.

Takto provedená změna ohodnocení zaručuje, že v grafu rovnosti zůstanou všechny hrany tvořící párování a také všechny hrany, které byly použity k označkování některého vrcholu. Navíc však ke grafu rovnosti přibude alespoň jedna hrana, po které bude možno označkovat alespoň jeden další vrchol. (Přitom ovšem může některá nepotřebná hrana z grafu rovnosti také zmizet.)

V takto upraveném grafu rovnosti opět hledáme maximální párování, tj. znovu použijeme značkovací proceduru a buď zvětšíme párování, nebo znovu měníme ohodnocení vrcholů, abychom mohli označkovat další vrchol. Po konečném počtu kroků nutně získáme perfektní párování, které je podle věty 9.4.2 optimálním řešením přiřazovací úlohy.

Maďarský algoritmus pro řešení přiřazovací úlohy lze shrnout takto:

1. [*Počáteční přípustné ohodnocení vrcholů.*] Pro každé $i \in X$ položíme $p(i) := \min_{j \in Y} c(i,j)$. Pro každé $j \in Y$ dále položíme $p(j) := \min_{i \in X}(c(i,j) - p(i))$.

2. [*Graf rovnosti.*] Sestrojíme faktor G_p grafu G takový, že

$$E(G_p) = \{(i,j) \in E(G) \mid c(i,j) - p(i) - p(j) = 0\} .$$

3. [*Maximální párování.*] Sestrojíme maximální párování P v grafu G_p. Je-li toto párování perfektní, výpočet končí, výsledné perfektní párování je nejlevnější.

4. [*Změna přípustného ohodnocení vrcholů.*] Není-li párování P perfektní, nalezneme množinu $A \subseteq X$ takovou, že $|A| > |V_{G_p}(A)|$, vypočteme

$$d = \min\{c(i,j) - p(i) - p(j) \mid i \in A, j \notin V_{G_p}(A)\}$$

a změníme přípustné ohodnocení vrcholů takto:

$$\begin{aligned} p(i) &:= p(i) + d \quad \text{pro všechna } i \in A, \\ p(j) &:= p(j) - d \quad \text{pro všechna } j \in V_{G_p}(A). \end{aligned}$$

Pokračujeme podle kroku 2.

Časové nároky: $O(n^4)$.

9.4.4 Příklad. Řešme přiřazovací úlohu, která je dána touto maticí cen:

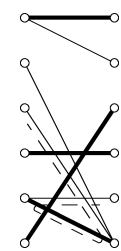
5	3	7	4	5	4
10	11	10	7	8	3
18	7	6	6	6	2
6	12	2	1	9	8
8	4	4	4	1	1
4	8	1	3	7	4

Odečtením řádkových minim od jednotlivých řádků získáme tuto transformovanou matici cen a ohodnocení $p(i)$ vrcholů strany X:

						$p(i)$
2	0	4	1	2	1	3
7	8	7	4	5	0	3
16	5	4	4	4	0	2
5	11	1	0	8	7	1
7	3	3	3	0	0	1
3	7	0	2	6	3	1

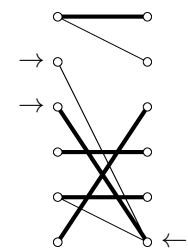
Podobně odečteme od každého sloupce sloupcové minimum. Tím získáme výchozí ohodnocení $p(j)$ vrcholů strany Y a transformovanou matici cen, která již obsahuje v každém řádku a v každém sloupci alespoň jednu nulu. Sestrojíme příslušný graf rovnosti a v něm zvolíme (v podstatě libovolným způsobem) výchozí párování.

							$p(i)$
	0	0	4	1	2	1	3
	5	8	7	4	5	0	3
	14	5	4	4	4	0	2
	3	11	1	0	8	7	1
	5	3	3	3	0	0	1
	1	7	0	2	6	3	1
$p(j)$	2	0	0	0	0	0	



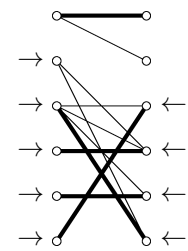
Značkováním jsme našli střídavou cestu s volnými krajními vrcholy a podél této cesty změníme (tj. zvětšíme) párování. Matice transformovaných cen ani samotný graf rovnosti se tím nemění. Poněvadž výsledné párování ještě není perfektní, pokračujeme dalším značkováním z volného vrcholu:

						↓	$p(i)$
	0	0	4	1	2	1	3
→	5	8	7	4	5	0	3
→	14	5	4	4	4	0	2
	3	11	1	0	8	7	1
	5	3	3	3	0	0	1
	1	7	0	2	6	3	1
$p(j)$	2	0	0	0	0	0	



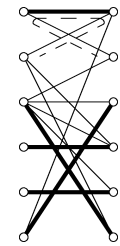
Označkované vrcholy a jim odpovídající řádky a sloupce jsou označeny šipkami. Velikost změny přípustného ohodnocení vrcholů $d = 4$ získáme jako minimum z transformovaných cen v průsečíku označených řádků a neoznačených sloupců, tj. zde druhého a třetího řádku a prvního až pátého sloupce. Po transformaci matice dostaneme:

			↓	↓	↓	↓	$p(i)$
	0	0	4	1	2	5	3
→	1	4	3	0	1	0	7
→	10	1	0	0	0	0	6
→	3	11	1	0	8	11	1
→	5	3	3	3	0	4	1
→	1	7	0	2	6	7	1
$p(j)$	2	0	0	0	0	−4	

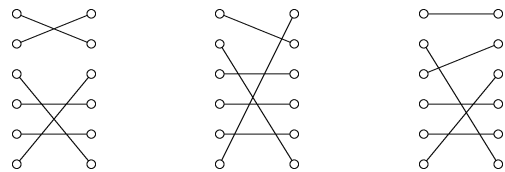


V transformované matici přibylo několik nul a v grafu rovnosti přibylo několik hran. Hrana $(5, 6)$ naopak z grafu rovnosti ubyla. Párování stále ještě nelze zvětšit, ale zvětšil se alespoň počet označkovaných vrcholů. Opět vybíráme minimum z průsečíku označkovaných řádků a neoznačkovaných sloupců, tentokrát $d = 1$:

							$p(i)$
	0	0	5	2	3	6	3
	0	3	3	0	1	0	8
	9	0	0	0	0	0	7
	2	10	1	0	8	11	2
	4	2	3	3	0	4	2
	0	6	0	2	6	7	2
$p(j)$	2	0	−1	−1	−1	−5	



Nyní již v grafu rovnosti existuje střídavá cesta, podél níž lze párování zvětšit. Dokonce jsou zde tři takové cesty, nakreslena je pro přehlednost pouze jedna z nich. Tři výsledná perfektní párování jsou na obr. 9.7. Cena každého z nich je rovna součtu ohodnocení vrcholů, tj. 18.



Obrázek 9.7: Tři rovnocenná optimální řešení příkladu 9.4.4.

9.4.5 Cvičení. Řešte tutéž přiřazovací úlohu, ale volte jinak výchozí přípustné ohodnocení vrcholů a výchozí graf rovnosti. Výsledné optimální přiřazení musí mít samozřejmě stejnou cenu, ale postup výpočtu může být hodně odlišný.

9.4.6 Poznámka. Při výpočtu na počítači se obvykle graf rovnosti explicite nesestrojuje, maximální párování lze hledat pomocí nulových prvků matice C_p, značkují se přitom řádky a sloupce matice.

9.4.7 Jiné algoritmy pro přiřazovací úlohu. Přiřazovací úlohu lze pokládat za speciální případ klasické dopravní úlohy 8.2.4, str. 137, kde X je množina dodavatelů, Y je množina spotřebitelů a kapacity všech dodavatelů i spotřebitelů jsou jednotkové. Dopravní úlohu pak lze řešit buď speciálním algoritmem, nebo jako úlohu o nejlevnější cirkulaci.

Jinou možnost představuje převod přiřazovací úlohy na úlohu o nejdražším párování v témže úplném bipartitním grafu. Tento převod spočívá ve změně cen hran (viz cvičení 9.1.8). Výslednou úlohu o nejdražším párování v bipartitním grafu pak lze řešit tak, že vyjdeme z prázdného párování, které postupně zvětšujeme, a to vždy podél střídavé cesty, která spojuje dva nenasycené vrcholy a která má vzhledem ke stávajícímu párování největší cenu (viz 9.2.4). Lze dokázat, že takto dostaneme nejdražší párování, které je zároveň optimálním řešením přiřazovací úlohy.

Kapitola 10

Eulerovské tahy

10.1 Základní pojmy a aplikace

10.1.1 Eulerovský tah, pokrývání hran. Připomeňme, že *tah* je sled, který neobsahuje žádnou hranu dvakrát (viz 1.5.2, str. 20). *Eulerovský tah* je takový tah, který obsahuje všechny hrany grafu (tedy každou hranu přesně jedenkrát). Eulerovské tahy dělíme na *orientované* a *neorientované* a na *uzavřené* a *neuzavřené*.

Řekneme, že soustava tahů *pokrývá hrany grafu*, jestliže každá hrana grafu leží přesně v jednom z nich. Eulerovský tah je tedy takový tah, který sám pokrývá všechny hrany grafu.

V souvislosti s eulerovskými tahy lze formulovat a řešit tyto úlohy:

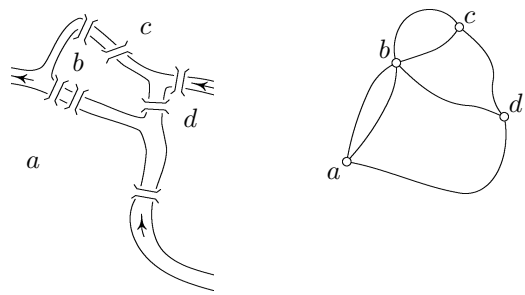
1. Rozhodnout, zda v grafu existuje eulerovský tah.
2. Sestrojit eulerovský tah.
3. Najít nejmenší počet tahů, které pokrývají hrany daného grafu.
4. V daném souvislém grafu, jehož hrany jsou ohodnoceny kladnými čísly, najít nejkratší uzavřený sled, který obsahuje všechny hrany grafu (každou hranu aspoň jedenkrát).

Pro řešení těchto úloh jsou známy rychlé (polynomiální nebo dokonce lineární) algoritmy.

Snadnost řešení těchto úloh je v pozoruhodném kontrastu s obtížností tzv. hamiltonovských úloh, které uvedeme v kapitole 12 (viz 12.1.4, str. 200).

10.1.2 Sedm mostů města Královce. Město Královec (Königsberg, dnešní Kaliningrad) leží na březích řeky Pregel a na dvou ostrovech. Břehy a ostrovy byly v 18. století spojeny sedmi mosty zhruba podle obr. 10.1 na následující straně. Obyvatelé města se tehdy bavili otázkou, zda je možno vykonat procházku, která by vedla přes každý most přesně jedenkrát. Samozřejmě se při tom nesmělo plavat přes řeku. Dokonce se o tom uzavíraly a prohrávaly sázky, dokud Leonard Euler v roce 1736 nedokázal, že taková procházka není možná. Eulerův způsob řešení této hříčky bývá označován za počátek teorie grafů (historické podrobnosti viz [Šišma97]).

Úloha o sedmi mostech je ekvivalentní úloze najít v grafu, který je nakreslen na obr. 10.1 vpravo, eulerovský tah.



Obrázek 10.1: Sedm mostů města Královce a odpovídající graf.

10.1.3 Kreslení co nejmenším počtem tahů. Na hledání eulerovského tahu lze zřejmým způsobem převést známou úlohu, nakreslit nějaký daný obrázek (třeba „domeček“) jedním tahem, aniž bychom kreslili některou čáru dvakrát a aniž bychom zvedli tužku z papíru. Od této úlohy je zřejmě odvozen termín *tah*. Každý asi ví, že na některé obrázky jeden tah nestačí.

Vzniká tedy přirozená otázka, jaký nejmenší počet tahů je k nakreslení nutný.

Nemusí přitom jít jen o zábavnou úlohu. Např. při kreslení pomocí počítače je účelné minimalizovat počet a délku přejezdů pisátka bez kreslení. Poněvadž každá čára by se měla kreslit přesně jedenkrát, je tato úloha příbuzná úloze 4. Pozor, nikoli totožná, a to ze dvou důvodů: obrázek nemusí být souvislý a navíc mohou být přejezdy pisátka přímočaré a nemusí probíhat podél kreslených čar. Je-li však kresba souvislá, lze k řešení použít postup podobný algoritmu 10.3.2.

10.1.4 Úloha čínského pošťáka. Pošťák musí při roznášce pošty projít všechny ulice svého rajónu. Jak má postupovat, aby ušel co nejméně kilometrů?

Tato úloha je pouze slovním vyjádřením úlohy 4, uvedené v 10.1.1. Jiným příkladem ze stejného soudku je optimalizace jízdy kropicího vozu, který má pokropit všechny ulice ve městě. Řešením se budeme zabývat v 10.3.

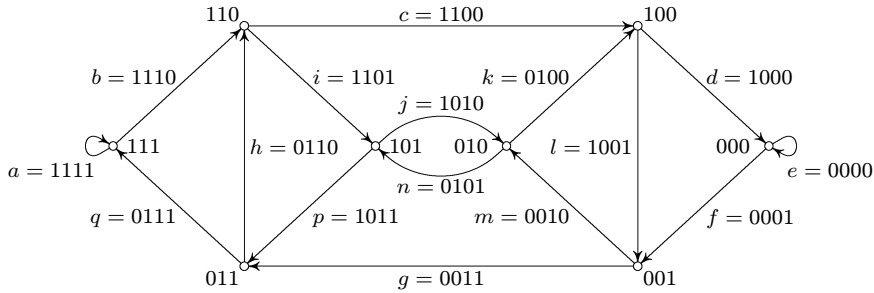
10.1.5 De Bruijnova posloupnost je příkladem důmyslné aplikace eulerovských tahů:

Máme dáno kladné číslo k. Úkolem je najít co nejdelší cyklickou posloupnost nul a jedniček takovou, že žádné dvě k-tice po sobě jdoucích cifer nejsou stejné. Taková posloupnost se nazývá *de Bruijnova posloupnost*.

Tato úloha souvisí s kódováním zhruba takto: obvod otáčejícího se kotouče má být označen nulami a jedničkami tak, aby bylo možno určit přesnou pozici kotouče přečtením pouze k po sobě jdoucích číslic.

Řešení: Počet posloupností tvořených k nulami a jedničkami je 2^k. Proto délka hledané de Bruijnovy posloupnosti nemůže být větší než 2^k. Pomocí eulerovských tahů dokážeme, že posloupnost o délce 2^k existuje, a získáme návod, jak ji sestrojit.

Vezměme graf, který má 2^{k-1} vrcholů, viz příklad na obr. 10.2 pro $k = 4$. Vrcholy označíme všemi $(k-1)$-ticemi nul a jedniček. Graf bude mít 2^k hran označených všemi k-ticemi nul a jedniček, a to tak, že hrana označená $(x_1, x_2, \ldots, x_{k-1}, x_k)$ povede z vrcholu $(x_1, x_2, \ldots, x_{k-1})$ do vrcholu $(x_2, \ldots, x_{k-1}, x_k)$. To znamená, že z každého vrcholu vychází přesně dvě hrany; jejich označení se liší pouze na posledním k-tém místě, jedna hrana tam má jedničku a druhá nulu. Podobně do každého vrcholu vchází dvě hrany, jejichž označení se liší v první číslici.



Obrázek 10.2: Graf pro konstrukci de Bruijnovy posloupnosti pro $k = 4$. Například eulerovskému tahu $abcdefghijklmnpq$ odpovídá de Bruijnova posloupnost 1111000011010010.

Libovolný sled v takovémto grafu můžeme snadno zakódovat pomocí posloupnosti nul a jedniček: každá číslice vyjadřuje jednu ze dvou možností jak pokračovat z daného vrcholu, který je označen předcházející $(k-1)$-tící.

Původní úloha o de Bruijnově posloupnosti je tedy převedena na hledání uzavřeného orientovaného eulerovského tahu v popsaném grafu. Tento graf je souvislý a pro každý jeho vrchol x platí $d^+(x) = d^-(x)$, proto podle věty 10.2.2 požadovaný eulerovský tah existuje. Dokonce je zřejmé, že eulerovských tahů (a tedy i de Bruijnových posloupností) může existovat několik.

Ve výše uvedené úloze jsme použili abecedu o dvou symbolech, totiž $\{0, 1\}$. Výše popsanou konstrukci lze snadno zobecnit i pro libovolnou větší abecedu. Má-li abeceda n prvků, bude mít de Bruijnova posloupnost délku n^k.

10.1.6 Cvičení. Najděte de Bruijnovu posloupnost pro $k = 3$ a $n = 2$. Kolik takových posloupností je? Řešte tutéž úlohu pro $k = 2$ a $n = 3$.

10.2 Existence a hledání eulerovských tahů

10.2.1 Věta. (Euler 1736.) Nechť graf G je souvislý. Pak v grafu G existuje neorientovaný uzavřený eulerovský tah právě tehdy, když každý vrchol grafu G má sudý stupeň.

Důkaz je téměř shodný s důkazem následující věty 10.2.2. Proto obě věty dokážeme společně. □

10.2.2 Věta. Nechť graf G je souvislý. Pak v grafu G existuje orientovaný uzavřený eulerovský tah právě tehdy, když pro každý vrchol v platí $d^+(v) = d^-(v)$.

Důkaz: Důkaz jedné z implikací je velmi snadný: Předpokládejme, že v grafu existuje uzavřený eulerovský tah (orientovaný nebo neorientovaný). Představme si cestovatele, který prochází grafem podél tohoto tahu. Zřejmě platí, že kolikrát přišel do nějakého vrcholu v, tolikrát z něj musel i odejít. Poněvadž při tom projde každou hranou přesně jedenkrát, plynou odtud okamžitě tvrzení o stupních vrcholů: $d(v)$ je sudý pro neorientovaný tah a $d^+(v) = d^-(v)$ pro tah orientovaný.

Opačná implikace je těžší, dokážeme ji tím, že popíšeme (a dokážeme) postup, jak eulerovský tah sestrojit, jsou-li splněny předpoklady jeho existence, tj. souvislost a vlastnosti stupňů.

Začněme z libovolného vrcholu x_0 a procházejme hranami grafu libovolně, ale tak, abychom žádnou hranou nešli dvakrát. Takto pokračujeme, dokud je to možné, tj. dokud z vrcholu, kam jsme právě přišli, vychází ještě nějaká dosud nepoužitá hrana. Tato procházka zcela jistě skončí v x_0, protože díky vlastnostem stupňů nemůže skončit nikde jinde a díky konečnosti grafu někde skončit musí. Nalezli jsme tedy nějaký uzavřený tah.

Nyní jsou dvě možnosti. Je-li tento tah eulerovský, jsme hotovi. Není-li tah eulerovský, pak někde v grafu existuje hrana, která v tahu neleží. Mezi x_0 a touto hranou existuje cesta (neboť graf je souvislý) a někde na této cestě musí existovat vrchol x_1, kterým prochází tah a z něhož ještě vychází nějaká nepoužitá hrana. Ve vrcholu x_1 tah přerušíme (rozpojíme) a začneme opět procházet nepoužitými hranami a tím tah (který je nyní otevřený) prodlužovat. Díky vlastnostem stupňů toto prodlužování skončí ve vrcholu x_1. V tomto vrcholu navážeme novou část tahu na původní, čímž získáme uzavřený tah, který má více hran.

Tento postup (přerušování a prodlužování tahu) opakujeme, dokud existuje hrana, která v tahu neleží. Díky konečnému počtu hran v grafu G nakonec jistě dostaneme tah, který je eulerovský. □

10.2.3 Poznámka. Předpoklad souvislosti v předchozích dvou větách je zbytečně silný: izolované vrcholy by nevadily.

10.2.4 Cvičení. Dokažte, že (slabě) souvislý orientovaný graf, jehož každý vrchol v splňuje $d^+(v) = d^-(v)$, je také silně souvislý.

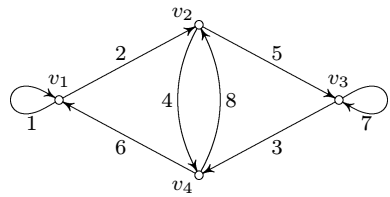
10.2.5 Algoritmus pro hledání uzavřeného eulerovského tahu. Postup byl již vysvětlen v důkaze věty 10.2.2. Zaměříme se na jeho efektivní provedení. V algoritmu se střídavě provádějí dvě fáze:

1. Existující tah prodlužujeme, dokud se nestane uzavřeným.
2. Uzavřený tah kontrolujeme, zda je eulerovský.

Při kontrole procházíme podél tahu a v každém vrcholu x testujeme, zda v množině $E(x)$, popř. $E^+(x)$ existuje hrana h, která dosud není obsažena v tahu. Pokud takovou hranu h najdeme, přerušíme kontrolu, tah ve vrcholu x rozpojíme a začneme

jej prodlužovat počínaje hranou h. Prodlužování skončí ve vrcholu x. Po propojení staré a nové části tahu pokračujeme v kontrole počínaje vrcholem x (předcházející část tahu je již zkontrolována) a postupujeme podél nové části tahu (ta doud není zkontrolována). Tím je zajištěno, že nejen při prodlužování, ale i při kontrole postupujeme podél každé hrany pouze jedenkrát. Celý postup tedy vyžaduje čas $O(n+m)$.

10.2.6 Příklad. Sestrojme uzavřený orientovaný eulerovský tah v grafu, který je nakreslen na obr. 10.3. Použijeme postup 10.2.5. Abychom se vyhnuli nejednoznačnosti, budeme k prodlužování tahu volit vždy hranu s nejnižším číslem a začneme ve vrcholu v_1.



Obrázek 10.3: Graf k příkladu 10.2.6.

V prvé fázi získáme tah tvořený hranami 1, 2, 4, 6. Při kontrole zjistíme, že ve vrcholu v_1 jsou všechny hrany obsaženy v tahu, ale z vrcholu v_2 vychází dosud nepoužitá hrana 5. Tah tedy rozpojíme mezi hranami 2 a 4, prodloužíme jej o hrany 5, 3, 8 a znovu spojíme. Tím získáme uzavřený tah 1, 2, 5, 3, 8, 4, 6. Pokračujeme v kontrole tam, kde jsme ji přerušili, tj. ve vrcholu v_2 (ten je již v pořádku) a dále ve v_3, kde zjistíme, že z v_3 vychází nepoužitá hrana 7. Tah proto rozpojíme mezi hranami 5 a 3 a prodloužením získáme tah 1, 2, 5, 7, 3, 8, 4, 6. Další kontrola ve vrcholech v_3, v_3, v_4, v_2, v_4 již ukáže, že tento tah je eulerovský.

10.2.7 Věta. Nechť graf G je souvislý a obsahuje k vrcholů lichého stupně. Pak nejmenší počet neorientovaných tahů pokrývajících hrany grafu G je roven $k/2$.

Důkaz: Nejprve si uvědomme, že číslo k je sudé: Součet všech stupňů je roven $2|E(G)|$, a je tedy sudý. Kdyby k bylo liché, musel by součet všech stupňů být lichý, což by byl spor (součet lichého počtu lichých čísel je vždy lichý).

Přidejme nyní ke grafu G celkem $k/2$ hran, které budou spojovat vždy dva vrcholy lichého stupně. Tím dostaneme souvislý graf G', v němž všechny vrcholy budou mít stupně sudé. Podle věty 10.2.1 tedy v grafu G' existuje uzavřený neorientovaný eulerovský tah.

Nyní odstraňme z grafu G' přidané hrany. Tím jednak dostaneme původní graf G, jednak se nám uzavřený eulerovský tah v G' rozpadne na celkem $k/2$ neuzavřených tahů, které dohromady pokrývají všechny hrany grafu G. □

10.2.8 Cvičení. Kolik tahů je nutných k pokrytí všech hran neorientovaného grafu, o kterém víme, že má 3 komponenty souvislosti, 2 vrcholy lichého stupně a žádný izolovaný vrchol? Bylo by možno určit potřebný počet tahů, kdyby počet vrcholů lichého stupně nebyl 2, ale 4?.

10.2.9 Věta. V orientovaném grafu G bez izolovaných vrcholů existuje orientovaný neuzavřený eulerovský tah právě tehdy, když graf G je souvislý a existují v něm dva vrcholy x, y takové, že $d^+(x) = d^-(x) + 1$ a $d^+(y) = d^-(y) - 1$ a pro všechny ostatní vrcholy v platí $d^+(v) = d^-(v)$.

Důkaz je podobný důkazu věty 10.2.7. □

10.2.10 Cvičení. Najděte způsob, jak pro (slabě) souvislý orientovaný graf zjistit na základě stupňů vrcholů nejmenší počet orientovaných tahů, které pokrývají jeho hrany. Návod: pro každý vrchol vypočtěte číslo $R(v) = d^+(v) - d^-(v)$.

10.3 Úloha čínského pošťáka

10.3.1 Úloha čínského pošťáka. Pošťák má projít všechny ulice města a vrátit se do výchozího místa, a to tak, aby ušel co nejméně kilometrů. Přeloženo do řeči grafů, jde o tuto úlohu:

Je dán neorientovaný souvislý graf, jehož hrany jsou ohodnoceny kladnými čísly. Úkolem je najít nejkratší uzavřený sled, který obsahuje všechny hrany grafu.

10.3.2 Řešení úlohy čínského pošťáka. Je zřejmé, že existuje-li v grafu eulerovský tah, je tento tah hledaným optimálním řešením.

Pokud v grafu eulerovský tah neexistuje, pak sled, který obsahuje všechny hrany, musí některými hranami procházet dvakrát nebo i vícekrát. Lze však dokázat (viz cvičení 10.3.4), že nejkratší sled prochází každou hranou pouze jedenkrát nebo dvakrát. Hledáme tedy sled, v němž je nejmenší součet délek hran, které jsou procházeny opakovaně (tj. dvakrát).

V nejkratším sledu, který obsahuje všechny hrany grafu, tvoří opakovaně procházené hrany soustavu cest, které spojují vždy dva vrcholy lichého stupně. Lze dokázat, že tyto cesty jsou navzájem hranově disjunktní, tj. že žádná hrana grafu neleží ve dvou různých takových cestách (viz cvičení 10.3.5).

Klíčem k řešení úlohy tedy je rozdělit vrcholy s lichým stupněm do dvojic, a to tak, aby součet délek nejkratších cest mezi vrcholy ve dvojicích byl nejmenší. Toto je vlastně úloha o párování, v níž je však třeba spárovat pouze vrcholy lichého stupně. Můžeme se na to dívat také tak, že hledáme nejlevnější perfektní párování v pomocném úplném grafu K, jehož vrcholy jsou všechny vrcholy lichého stupně původního grafu G a délky hran grafu K jsou délky nejkratších cest v grafu G.

Celý postup řešení lze shrnout takto:

1. V daném grafu G najdeme množinu L vrcholů s lichým stupněm.
2. Pro všechny dvojice (x, y) vrcholů množiny L vypočteme délku $u(x, y)$ nejkratší (neorientované) cesty z x do y.
3. Na množině vrcholů L definujeme úplný graf K, jehož hrany jsou ohodnoceny délkami nejkratších cest $u(x, y)$.
4. V grafu K najdeme nejlevnější perfektní párování P (viz kapitola 9).

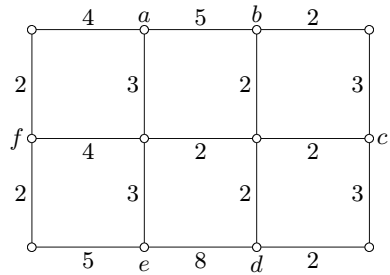
5. Pro každou hranu (x, y) grafu K, která leží v párování P, vezmeme všechny hrany původního grafu G, které tvoří nejkratší cestu z x do y, a ke grafu G přidáme kopie všech těchto hran. V grafu G', který takto získáme, budou mít všechny vrcholy sudý stupeň.

6. V grafu G' sestrojíme eulerovský tah. Tento tah prochází všemi přidanými hranami, což odpovídá opakovaným průchodům hranami původního grafu.

7. V eulerovském tahu nahradíme všechny přidané hrany jim odpovídajícími hranami grafu původního. Tím získáme hledaný nejkratší sled, který prochází všemi hranami grafu G.

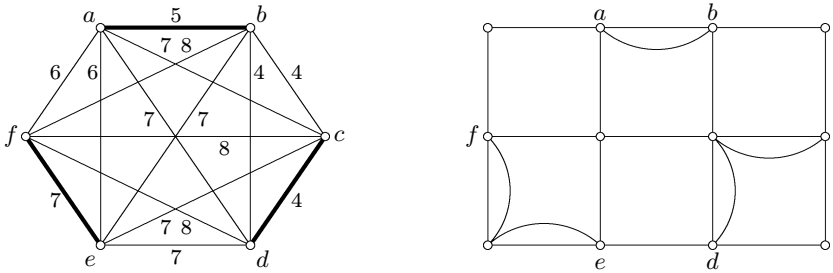
Důkaz správnosti tohoto algoritmu se opírá o fakt, že součet délek opakovaně procházených hran je roven ceně nejlevnějšího perfektního párování. Výpočet lze provést v čase $O(m + n^3)$.

10.3.3 Příklad. Řešme úlohu čínského pošťáka pro graf z obrázku 10.4.



Obrázek 10.4: Graf k úloze čínského pošťáka 10.3.3.

V grafu je šest vrcholů lichého stupně, $L = \{a, b, c, d, e, f\}$. Úplný graf K na množině vrcholů L je na obr. 10.5 vlevo. Hrany tvořící nejlevnější perfektní párování jsou v něm nakresleny silně. Na témže obrázku vpravo je nakreslen původní graf s přidanými hranami. Stupně všech vrcholů jsou v něm již sudé, není tedy problém najít v něm eulerovský tah.



Obrázek 10.5: Řešení úlohy čínského pošťáka 10.3.3.

10.3.4 Cvičení. Dokažte, že sled, který je řešením úlohy čínského pošťáka, nemůže procházet žádnou hranou více než dvakrát.

10.3.5 Cvičení. Dokažte, že žádné dvě cesty, které ke grafu přidáváme v kroku 5 algoritmu 10.3.2, nemají společnou hranu. Jinými slovy, dokažte, že v grafu G' bude ke každé hraně grafu G přidána nejvýše jedna kopie. Návod: využijte faktu, že párování P je nejlevnější.

10.3.6 Poznámka. Úlohu čínského pošťáka jsme pro jednoduchost formulovali a řešili pouze pro kladné délky hran. Algoritmus 10.3.2 lze použít i na graf, který obsahuje hrany o nulové délce. Důkaz správnosti je v tomto obecnějším případě složitější, neboť je třeba počítat s tím, že hrana o nulové délce se může ve výsledném sledu vyskytovat i více než dvakrát. To však nevadí, na délku sledu to nemá vliv.

10.3.7 Cvičení. Lze algoritmus 10.3.2 použít i na graf, který obsahuje hrany o záporné délce?

Kapitola 11

NP-těžké úlohy

V předchozích kapitolách jsme se (až na několik výjimek) zabývali úlohami, pro které jsou známy uspokojivě rychlé algoritmy. Následující dvě kapitoly se naopak věnují úlohám, které jsou (až na výjimky) „těžké“. Co to však jsou „těžké“ úlohy?

Prvý oddíl této kapitoly obsahuje velmi stručné shrnutí základních pojmů a poznatků teorie složitosti (míněno: složitosti výpočtů), zejména teorie tzv. NP-úplnosti. Uvádíme je jednak proto, aby naše výroky o obtížnosti úloh měly alespoň trochu solidní základ, jednak proto, že se v literatuře lze poměrně často setkat s výroky o NP-úplnosti některých úloh.

Výklad v tomto prvém oddíle je poněkud abstraktnější a vybočuje mimo rámec teorie grafů. Studium tohoto oddílu můžete přeskočit, smíříte-li se s vágním vyjádřením, že „NP-těžké úlohy jsou obtížně řešitelné“.

V oddílech 11.2 a 11.3 uvedeme dvě metody, které lze použít k řešení poměrně různorodých úloh (i negrafových). Obě metody se používají (zejména) k řešení NP-těžkých úloh, obě lze pokládat za „inteligentní průzkum všech myslitelných možností“ a v obou se prohledává tzv. strom řešení. Srovnání obou metod je uvedeno v 11.3.8. Obě metody využijeme v kapitolách 12 a 13.

11.1 Časová složitost

Teorie složitosti je část informatiky, která se (zjednodušeně řečeno a mimo jiné) zabývá závislostí doby řešení úlohy na velikosti instance úlohy.

11.1.1 Typ úlohy a instance úlohy. Slovo „úloha“ se používá ve dvou různých významech, mezi nimiž zde musíme pečlivě rozlišovat:

Typ úlohy určuje, jakým způsobem je úloha zadána, tj. jaká data a v jakém uspořádání budou tvořit zadání úlohy, co má být výsledkem a jaký má být vztah mezi výsledkem a zadáním. *Instance úlohy* je konkrétní případ úlohy daného typu.

Známe-li pouze typ úlohy, nemáme ještě co počítat, můžeme však přemýšlet o algoritmu, kterým bychom řešili instance tohoto typu úlohy, kdybychom nějaké

měli. Počítat má smysl, teprve když máme konkrétní zadání, tj. instanci úlohy. Viz též 1.11.3, str. 32, kde jsme tyto pojmy zavedli pro optimalizační úlohy.

11.1.2 Velikost instance úlohy. Data, která popisují instanci daného typu úlohy, tvoří vstup do algoritmu, který úlohy daného typu řeší, proto o těchto datech mluvíme jako o vstupních datech. Velikost instance pak měříme jako objem těchto vstupních dat. Obecně se používá počet bitů, ale v grafových úlohách, kde zadáním úlohy je obvykle graf (a možná nějaké jeho ohodnocení), je zvykem vyjadřovat velikost vstupních dat počtem vrcholů nebo počtem vrcholů a počtem hran (tj. dvěma čísly).

11.1.3 Doba práce algoritmu. To je tak trochu potíž. Doba výpočtu na konkrétním typu počítače se pro teoretické úvahy nehodí, protože počítače mívají různě rychlé procesory a jakékoli tvrzení by mělo velice omezenou platnost. Počet instrukcí, které je třeba při výpočtu vykonat, je trochu lepší, ale i on závisí na typu procesoru (procesory mívají různé soubory instrukcí).

Proto se v teoretických úvahách často používá tzv. *asymptotické vyjádření* doby práce algoritmu pomocí symbolu O (velké písmeno O). Připomeňme (viz 1.10.6, str. 30) jeho definici:

Nechť f, g jsou dvě nezáporné funkce reálné proměnné. Existují-li reálné konstanty k, m takové, že pro všechna $x > m$ platí $g(x) \leq kf(x)$, pak píšeme $g = O(f)$.

Zápis $g = O(f)$ se vyslovuje jako „g je o f“. Vztah $g = O(f)$ je vlastně odhadem funkce g shora. Nezávisí na chování obou funkcí g a f pro „malé“ hodnoty argumentů (přesněji, pro $x \leq m$) a nezávisí také na násobení jedné z funkcí nějakou konstantou.

Řekneme-li, že nějaký algoritmus pracuje v čase $O(f(n))$, kde n je velikost instance úlohy, znamená to, že doba jeho práce na instancích od určité velikosti výše nepřesahuje nějaký násobek funkce f. Přitom je však možné, že pro mnoho konkrétních případů vstupních dat (třeba i pro všechny) bude tentýž algoritmus pracovat rychleji. Dokonce i kdyby platil lepší odhad, např. $O(n^2)$, byl by odhad $O(n^3)$ také pravdivý.

Vyjádření doby práce algoritmu symbolem O se nezískává měřením doby skutečných výpočtů na počítači, ale teoretickým rozborem činnosti algoritmu.

Dodejme, že je třeba rozlišovat, zda máme na mysli dobu práce v nejhorším případě nebo dobu práce průměrnou. Není výjimkou, když nějaký algoritmus má odhad průměrné doby práce řádově lepší než odhad pro nejhorší případ. Odhady pro nejhorší případ jsou jednodušší (snáze odvoditelné). V této knize se téměř výhradně zabýváme jen odhady pro nejhorší případ.

Asymptotické vyjádření doby práce pomocí symbolu O má své výhody i nevýhody. Výhodou je to, že se zaměřuje na chování algoritmu na velikých instancích úloh. S instancemi malého rozsahu totiž zpravidla nejsou problémy. Další výhodou je nezávislost na typu a rychlosti počítače, na programovacím jazyce, překladači, schopnostech programátora apod.

Nevýhoda je v tom, že z asymptotického odhadu není jasné, která instance úlohy je velká. Může se snadno stát, že algoritmus s horším asymptotickým odhadem

bude na grafu o řekněme 5000 vrcholech rychlejší než jiný algoritmus, který je sice asymptoticky lepší, ale jeho rychlost se projeví až na grafech mnohem větších.

11.1.4 Rozhodovací úlohy jsou takové úlohy, jejichž výsledek je buď „ano“, nebo „ne“.

Mnohé (ovšem zdaleka ne všechny) rozhodovací úlohy jsou odvozeny z optimalizačních úloh jako jejich rozhodovací verze: k zadání úlohy se přidá konstanta K a ptáme se, zda existuje přípustné řešení s hodnotou účelové funkce, která je lepší nebo rovna K. Zvláštním případem rozhodovací verze pak je dotaz, zda existuje vůbec nějaké přípustné řešení dané úlohy (stačí vhodně zvolit konstantu K). Rozhodovací verze úlohy určitě není těžší než sama optimalizační úloha.

11.1.5 Polynomiálně řešitelné úlohy jsou úlohy, pro něž existuje algoritmus, který danou úlohu řeší, přičemž doba práce je pro nejhorší případ shora omezena asymptoticky nějakým polynomem, tj. doba práce je $O(p(n))$, kde n je velikost vstupních dat a p je nějaký polynom. Říkáme, že takový algoritmus úlohu řeší *v polynomiálním čase* nebo že je *polynomiální*.

Třída úloh P je tvořena polynomiálně řešitelnými rozhodovacími úlohami.

Polynomiálně řešitelné úlohy jsou obvykle pokládány za „snadno řešitelné“. Pro mnoho úloh však polynomiální algoritmus není znám, a to navzdory dlouhému úsilí nejlepších odborníků.

11.1.6 Třída úloh NP. Rozhodovací úloha patří do třídy NP, když pro ni existuje tzv. *ověřovací algoritmus* $\mathcal{V}$, který má tyto vlastnosti:

- jeho vstupem jsou data d popisující instanci úlohy a tzv. *certifikát*, což je jakýsi blíže neurčený kus dat, jehož velikost je shora omezena nějakým pevně daným polynomem vzhledem k délce vstupních dat d,
- pracuje v polynomiálním čase vzhledem k velikosti vstupních dat d a dává vždy odpověď „ano“ nebo „nevím“,
- pro instanci dané úlohy d, pro kterou je správná odpověď „ano“, existuje certifikát c takový, že ověřovací algoritmus $\mathcal{V}(d, c)$ dá odpověď „ano“,
- pro instanci úlohy d, pro kterou je správná odpověď „ne“, dává ověřovací algoritmus vždy (pro všechny možné certifikáty) odpověď „nevím“.

Lidově řečeno, rozhodovací úloha patří do třídy NP, když kladnou odpověď lze potvrdit tím, že uhodneme vhodný certifikát a pak v polynomiálním čase ověříme, že jsme hádali správně. Záporná odpověď se nepotvrzuje, pouze se požaduje, aby v tomto případě neexistoval žádný falešný certifikát, který by potvrzoval nesprávnou odpověď.

V typických NP-úlohách se ptáme, zda existuje něco, co se možná i těžko hledá, ale snadno se ověřuje, že jsme to našli.

Konkrétním příkladem je úloha, zda v daném grafu existuje kružnice, která prochází všemi vrcholy grafu (tzv. hamiltonovská kružnice, viz kapitola 12). Certifikátem zde je ona kružnice. Ověřovací algoritmus ověřuje, že to je opravdu kružnice a že opravdu prochází přes všechny vrcholy grafu.

Poznamenejme, že u NP-úloh je velmi důležité, jak je položena otázka, která se v úloze řeší. Je to proto, že certifikát a potvrzení požadujeme pouze pro kladnou odpověď. Obrácením otázky u nějaké NP-úlohy tedy dostaneme úlohu zcela jinou, která již do třídy NP nemusí patřit (může být těžší).

Každá úloha ze třídy P je zároveň ve třídě NP, neboť polynomiální algoritmus, který úlohu řeší, lze pokládat za algoritmus ověřovací, u něhož na certifikátu nezáleží. Platí tedy $\mathrm{P} \subseteq \mathrm{NP}$. V současné době není známo (nikdo neumí matematicky dokázat), zda platí rovnost, ale všeobecně se věří, že $\mathrm{P} \neq \mathrm{NP}$.

Ve třídě NP jsou úlohy různé obtížnosti. Než si řekneme, které jsou nejtěžší, musíme umět porovnávat obtížnost úloh.

11.1.7 Porovnávání obtížnosti, P-redukce. Úloha A je *P-redukovatelná* (též *polynomiálně redukovatelná*) na úlohu B, existuje-li polynomiální algoritmus, který ze vstupních dat úlohy A vyrobí vstupní data pro úlohu B, a to tak, že odpovědi na obě instance jsou stejné. Doba řešení úlohy B při tom nehraje žádnou roli.

Existuje-li P-redukce úlohy A na úlohu B, chápeme to jako doklad, že úloha A je lehčí nebo stejně obtížná jako úloha B. Pokud existují P-redukce v obou směrech (A na B a také B na A), pokládáme obě úlohy za stejně obtížné. Díky P-redukcím existují ve třídě NP skupiny stejně obtížných úloh. Třída P je jednou z nich.

Jestliže úloha A je P-redukovatelná na úlohu B a $B \in \mathrm{P}$, pak také $A \in \mathrm{P}$.

Několik příkladů P-redukcí je uvedeno v 12.2, str. 201.

11.1.8 NP-úplná úloha (anglicky NP-complete) je taková úloha, která patří do NP a je ve třídě NP nejtěžší v tom smyslu, že jakákoli jiná NP-úloha je na ni P-redukovatelná. Třídu všech NP-úplných úloh značíme NP-C.

NP-úplných úloh je známo mnoho (kniha [GJ79] jich uvádí přes 300), řada z nich jsou grafové úlohy. V naší knize se s NP-úplnými úlohami setkáme zejména v kapitolách 12 a 13.

Všechny NP-úplné úlohy jsou z hlediska P-redukce stejně těžké. Kdyby se podařilo najít polynomiální algoritmus pro kteroukoli z nich, měli bychom ihned polynomiální algoritmy pro všechny NP-úlohy a platilo by $\mathrm{P} = \mathrm{NP}$. V podstatě nikdo nevěří, že se takový algoritmus najde, ale dokázat to prozatím nikdo nedovede.

11.1.9 NP-těžká úloha (anglicky NP-hard) je taková úloha B, na kterou lze P-redukovat nějakou NP-úplnou úlohu A.

Na rozdíl od NP úplnosti zde nepožadujeme příslušnost úlohy B ke třídě NP. Díky tomu do třídy NP-těžkých úloh patří i optimalizační úlohy, jejichž rozhodovací verze jsou NP-úplné. Mezi NP-těžké však patří i mnohé další, daleko obtížnější úlohy.

11.1.10 Pozor na speciální případy. Je běžné, že nějaká úloha je NP-těžká, ale některé její speciální případy jsou snadno (polynomiálně) řešitelné.

Například úloha najít v daném grafu kružnici, která prochází všemi vrcholy grafu (tzv. hamiltonovská kružnice, viz kapitola 12), je notoricky známa jako NP-těžká,

ale kdybychom se omezili pouze na grafy, které mají stejný počet vrcholů i hran, dostali bychom úlohu (typ úlohy), která je triviálně řešitelná v lineárním čase.

Na tuto záludnost byste měli dát pozor, až se budete vymlouvat, že nějakou úlohu neumíte řešit, protože je NP-těžká.

11.1.11 Praktický význam teorie NP-úplnosti. Celá teorie NP-úplnosti působí dost pesimistickým dojmem: pokud věříme, že P $\neq$ NP, pak není naděje, že by existovaly polynomiální (a tedy rychlé) algoritmy pro celou řadu velmi praktických úloh.

Reálné využití této teorie je tedy v podstatě alibistické: pokud o nějaké úloze umíme dokázat, že je NP-těžká, a věříme-li, že P $\neq$ NP, pak to, že pro naši úlohu nedovedeme najít polynomiální algoritmus, není žádná ostuda, protože to neumí nikdo na světě včetně těch nejchytřejších odborníků.

To ovšem vůbec neznamená, že NP-úplné a NP-těžké úlohy nelze řešit. Musíme však buď slevit z požadavků na rychlost (algoritmus nebude polynomiální, ale stále ještě může být prakticky použitelný), nebo použijeme nějaký heuristický algoritmus, který sice bývá rychlý, ale musíme se smířit s rizikem, že nedosáhneme vždy požadovaného výsledku (např. najdeme řešení, které není optimální, nebo nenajdeme přípustné řešení, i když ve skutečnosti existuje).

Úlohy (instance) malého rozsahu lze téměř vždy řešit tzv. *metodou hrubé síly*, což je vznešený název pro jednoduché vyzkoušení všech možností, které přicházejí v úvahu.

Pro mnoho úloh lze použít backtracking 11.2 nebo metodu větví a mezí 11.3. Tyto metody lze pokládat za inteligentní vyzkoušení všech možností, kdy se dosahuje někdy i značného zrychlení tím, že se nezkoumají možnosti, které nemohou přispět k řešení.

11.2 Backtracking

11.2.1 Posloupnosti a strom řešení. Backtracking lze použít v těch situacích, kdy každé řešení úlohy (tj. každý prvek prostoru řešení) můžeme považovat za konečnou posloupnost, přičemž každý prvek posloupnosti musí být vybrán z nějaké předem dané konečné množiny. Při troše fantazie se na většinu kombinatorických úloh můžeme dívat tímto způsobem.

Kromě posloupností, které představují (úplné) řešení úlohy budeme uvažovat také všechny jejich počáteční úseky. Tyto počáteční úseky odpovídají částečným (neúplně určeným) řešením. Všechny takovéto posloupnosti můžeme pokládat za vrcholy orientovaného grafu, v němž každá částečná posloupnost má jako své následníky všechna svá prodloužení o jeden prvek. Tento graf je kořenovým stromem (kořenem je prázdná posloupnost) a nazýváme jej *stromem řešení*.

11.2.2 Backtracking. Nejjednodušší verze backtrackingu spočívá v prohledávání stromu řešení do hloubky. Vždy, když se při prohledávání dostaneme k posloup-

nosti, která odpovídá úplnému řešení, provedeme test, zda toto řešení je přípustné. U optimalizační úlohy navíc evidujeme nejlepší dosud nalezené přípustné řešení.

Takto jednoduchý backtracking bez použití dalších triků ovšem není v podstatě ničím jiným než systematicky provedenou metodou hrubé síly. Triky, kterými lze výpočet mnohdy i značně urychlit, spočívají v tom, že vynecháme prohledávání některých podstromů.

Má smysl prodlužovat jen ty posloupnosti, u nichž je naděje, že mohou být prodlouženy na přípustné řešení. Pokud jsme tedy schopni s jistotou poznat, že žádné prodloužení už nemůže být přípustné, můžeme ušetřit čas tím, že prohledávání příslušného podstromu přeskočíme (provedením návratu).

Řešíme-li optimalizační úlohu, můžeme dosáhnout další úspory času, budeme-li schopni pro každou posloupnost $x_1, \ldots, x_i$ určit číslo D, které nazveme dolním, popř. horním *odhadem* (pro minimalizační, popř. maximalizační úlohu) a které má tu vlastnost, že prodlužováním posloupnosti $x_1, \ldots, x_i$ nelze dosáhnout přípustného řešení lepšího než D. Bude-li tedy pro posloupnost $x_1, \ldots, x_i$ tento odhad horší než nějaké přípustné řešení, které už v té chvíli známe, nemá další prodlužování posloupnosti $x_1, \ldots, x_i$ smysl a prohledávání příslušného podstromu můžeme opět přeskočit.

I přes uvedené možnosti úspor bývá backtracking časově hodně náročný.

Příkladem jednoduchého backtrackingu je algoritmus 3.3.10, str. 58, další příklady uvedeme v kapitolách 12 a 13. Backtracking se však běžně používá i k řešení negrafových úloh.

11.2.3 Příklad. Metodou backtrackingu řešme tuto úlohu: Najděte maximum funkce $4x_1+3x_2+2x_3+x_4$ za podmínek $2x_1+3x_2+x_3+3x_4 \leq 6$, $x_1, x_2 \in \{0,1,2\}$, $x_3, x_4 \in \{0,1\}$.

Tuto úlohu lze samozřejmě řešit i mnoha jinými (a lepšími) postupy. Zde nám však jde o ukázku postupu, který je „ve stylu backtrackingu“.

Řešení úlohy, tj. vektor (x_1, x_2, x_3, x_4) pokládejme za posloupnost celých čísel. Dílčí posloupnosti budeme prodlužovat zleva doprava a k prodlužování budeme používat přednostně nejvyšší povolené hodnoty.

Poněvadž všechny proměnné mohou nabývat jen nezáporných hodnot, můžeme při prohledávání stromu řešení do hloubky provést návrat, jakmile součet hodnot na levé straně nerovnosti přesáhne číslo 6. Takovou posloupnost totiž nelze doplnit na přípustné řešení.

Horní odhad budeme pro každé částečné řešení počítat tak, že částečné řešení doplníme nejvyššími možnými hodnotami na délku úplného řešení a tuto posloupnost dosadíme do účelové funkce (bez ohledu na omezující podmínku). Např. posloupnost $(1,2)$ doplníme na $(1,2,1,1)$, odhad tedy je 13. Je zřejmé, že vyšší hodnoty prodlužováním posloupnosti $(1,2)$ nelze dosáhnout, ve skutečnosti nedosáhneme ani oněch 13.

Pro stručnost uvedeme jen ty posloupnosti, ze kterých provádíme návrat:

$(2,2)$ nepřípustné,
$(2,1)$ nepřípustné,

$(2,0,1,1)$	nepřípustné,
$(2,0,1,0)$	úplné přípustné řešení s hodnotou 10,
$(2,0,0)$	odhad $9 < 10$,
$(1,2)$	nepřípustné,
$(1,1,1,1)$	nepřípustné,
$(1,1,1,0)$	odhad $9 < 10$,
$(1,1,0)$	odhad $8 < 10$,
$(1,0)$	odhad $7 < 10$,
(0)	odhad $9 < 10$.

Optimálním řešením je posloupnost $(2,0,1,0)$ s hodnotou účelové funkce 10. Všimněte si, že toto řešení jsme našli poměrně brzy. Zbytek výpočtu však byl nutný pro ověření, že neexistuje lepší řešení.

11.3 Metoda větví a mezí

Metoda větví a mezí se používá pouze k řešení optimalizačních úloh. Podobně jako u backtrackingu se zde prohledává strom řešení, ten však bývá definován jiným, složitějším způsobem.

V tomto oddíle podáme obecný výklad metody větví a mezí a uvedeme příklad použití na negrafové úloze. Příklady použití na úlohy teorie grafů jsou uvedeny v 12.4, str. 205.

11.3.1 Větvení množiny přípustných řešení. Množinu přípustných řešení úlohy U označme symbolem $\text{Př}(U)$.

Mějme optimalizační úlohu U s účelovou funkcí f. Úlohu U můžeme řešit (mimo jiné) tak, že vytvoříme úlohy $U_1, \ldots, U_n$ se stejnou účelovou funkcí (tzv. *podúlohy* úlohy U) takové, že

$$\text{Př}(U) = \text{Př}(U_1) \cup \ldots \cup \text{Př}(U_n) \ .$$

Optimální řešení úlohy U pak lze získat tak, že vybereme nejlepší ze všech optimálních řešení podúloh $U_1, \ldots, U_n$. Samozřejmě se při tom snažíme (v tom je smysl větvení), aby podúlohy $U_1, \ldots, U_n$ byly v nějakém smyslu snazší než původní úloha U, tj. aby např. byly menší. Dodejme, že při tom je výhodné, když množiny $\text{Př}(U_1), \ldots, \text{Př}(U_n)$ jsou navzájem disjunktní, ale není to nutné.

Nalezení optimálního řešení úlohy U je snadné, pokud pro každou podúlohu U_i:

1. buď najdeme optimální řešení úlohy U_i,
2. nebo prokážeme, že úloha U_i nemá žádné přípustné řešení,
3. nebo prokážeme, že žádné přípustné řešení úlohy U_i není lepší než nějaké v té době již známé přípustné řešení úlohy U.

Často se ovšem stává, že pro některou podúlohu (a často ne jen jednu) nejsme schopni přímo zjistit ani jednu z výše uvedených tří možností. V tom případě pro každou takovou podúlohu, označme ji U_i, použijeme stejný postup jako pro

úlohu U, tj. rozvětvíme ji na podúlohy $U_{i_1}, \ldots, U_{i_m}$ a pro takto získané podúlohy se opět snažíme zjistit některou z výše uvedených možností. Některé z takto získaných podúloh může být nutno dále větvit.

11.3.2 Strom řešení. Jednotlivé podúlohy můžeme pokládat za vrcholy kořenového stromu, který nazýváme stromem řešení. Kořenem je výchozí úloha U a z každé úlohy, která byla větvena na podúlohy, vedou hrany ke všem jejím podúlohám. Během výpočtu se strom řešení mění (zvětšuje).

Pro výpočet jsou zajímavé pouze listy stromu řešení. Listy dělíme na tzv. *živé* a *mrtvé*. Za mrtvé pokládáme ty listy (podúlohy), u nichž se již podařilo prokázat některou ze tří možností uvedených v 11.3.1. Těmito podúlohami se již není třeba dále zabývat. Ostatní listy (podúlohy) jsou živé.

Živou podúlohu je třeba buď umrtvit (zjištěním některé ze tří možností z 11.3.1), anebo rozvětvit. Rozvětvením podúloha přestává být listem, ve stromě řešení se však místo ní objeví několik nových listů. Výpočet končí v okamžiku, kdy všechny listy stromu řešení jsou mrtvé.

11.3.3 Odhady a meze. Pokud pro některou podúlohu U_i zjistíme, že platí možnost 3, tj. pokud prokážeme, že žádné přípustné řešení úlohy U_i není lepší než nějaké v té době již známé řešení úlohy U, je to při řešení dobrá zpráva, protože tuto podúlohu není třeba větvit. Nejlepší dosud známé řešení úlohy U se však během výpočtu obvykle mění (zlepšuje), a tak se snadno stane, že možnost 3 se o nějaké podúloze podaří prokázat, teprve až najdeme dostatečně dobré řešení celé úlohy U.

Proto bývá výpočet uspořádán tak, že jednak evidujeme nejlepší dosud známé řešení a hodnotu jeho účelové funkce, jednak pro každou podúlohu U_i vypočteme číslo $\mathrm{Odh}(U_i)$ takové, že žádné přípustné řešení úlohy U_i není lepší než $\mathrm{Odh}(U_i)$.

Číslo $\mathrm{Odh}(U_i)$ bývá v literatuře nazýváno u minimalizačních úloh *dolním odhadem* a u maximalizačních úloh *horním odhadem*. Čím je odhad blíže skutečné hodnotě optimálního řešení podúlohy, tím je tento odhad užitečnější, neboť tím snáze jeho pomocí prokážeme, že pro podúlohu platí možnost 3. Odhad je tedy tím lepší, čím je méně optimistický. Někdy lze volit mezi několika metodami výpočtu odhadu, které dávají různě kvalitní výsledky.

Způsob výpočtu odhadu samozřejmě velice závisí na řešené úloze a na způsobu vytváření podúloh. Jako odhad pro podúlohu U_i se často používá hodnota optimálního řešení tzv. relaxované podúlohy:

11.3.4 Relaxace nepříjemných omezujících podmínek. Poněvadž množina přípustných řešení úlohy je vždy popsána (zadána) pomocí omezujících podmínek, provádí se i větvení úlohy na její podúlohy pomocí dodatečných omezujících podmínek, které se k větvené úloze přidávají.

Obvyklá situace je tato: Omezující podmínky výchozí úlohy U lze rozložit na dvě skupiny, nazvěme je *příjemné* a *nepříjemné*. Vynecháme-li (takzvaně *relaxujeme*) nepříjemné omezující podmínky, dostaneme tzv. relaxovanou úlohu, která má již jenom příjemné omezující podmínky a kterou dovedeme (pokud možno rychle) řešit.

Najdeme-li optimální řešení r úlohy, která vznikla relaxací z U, jsou dvě možnosti: buď řešení r splňuje, nebo nesplňuje nepříjemné omezující podmínky. Pokud splňuje, pak máme štěstí, protože řešení r je zároveň optimálním řešením i samotné úlohy U, a úlohu U tedy není třeba větvit. Obvykle však štěstí nemáme a řešení r nepříjemné omezující podmínky nesplňuje. V takové situaci uděláme dvě věci: za prvé, hodnotu optimálního řešení relaxované úlohy můžeme použít jako odhad, a za druhé, platnost oněch nepříjemných podmínek si snažíme vynutit přidáváním podmínek příjemných, a to i za cenu toho, že se nám úloha rozpadne (rozvětví) na několik podúloh.

Podstatné je, že při větvení úlohy přidáváme vždy jen příjemné omezující podmínky. Tím je totiž zaručeno, že takto vzniklé podúlohy jsou stejného typu jako větvená úloha, a můžeme tedy opět použít onen trik s relaxací a řešením relaxované podúlohy.

Které omezující podmínky můžeme pokládat za příjemné, je dáno naší schopností rozumně rychle řešit příslušné relaxované podúlohy. Zpravidla pro danou úlohu existuje několik možností, jak zvolit prostor řešení a jak přeformulovat omezující podmínky, přitom obojí má značný vliv na pracnost řešení metodou větví a mezí. Příklady uvedeme v 12.4.

11.3.5 Volba podúlohy k větvení by mohla být založena na prohledávání stromu řešení např. do šířky nebo do hloubky. Častěji se však pro každou podúlohu vypočte hodnota, nazvěme ji prioritou, a mezi živými podúlohami vybíráme k větvení tu, která má prioritu nejlepší. Běžně se v roli priority používá (dolní nebo horní) odhad. Vybíráme pak podúlohu, která má odhad nejlepší (nejnadějnější), neboť u této podúlohy lze očekávat, že zlepšíme nejlepší dosud nalezené řešení a pomocí odhadů pak umrtvíme některé dosud živé podúlohy.

11.3.6 Shrnutí metody větví a mezí. Při výpočtu udržujeme nejlepší dosud nalezené přípustné řešení $\check{r}$ dané úlohy U a jeho hodnotu účelové funkce L. Na začátku $\check{r}$ není definováno a jako hodnotu L použijeme nejhorší možnou hodnotu ($+\infty$ nebo $-\infty$).

Doporučuje se však ještě před zahájením výpočtu metodou větví a mezí získat nějakým heuristickým postupem (viz 11.4) co nejlepší přípustné řešení $\check{r}$ a jeho hodnotu L, neboť to může následující výpočet urychlit.

Dále při výpočtu udržujeme seznam živých podúloh.

Obvykle ještě před zařazením do seznamu pro každou podúlohu vypočteme odhad, nejčastěji řešením relaxované verze příslušné podúlohy. Pokud přitom dostaneme přípustné řešení, porovnáme je s řešením $\check{r}$ a dále uchováváme lepší z nich. Pokud bylo řešení $\check{r}$ nahrazeno lepším, tj. pokud se zlepšila hodnota L, projdeme seznam živých podúloh a vyřadíme ty, které jsou díky nové hodnotě L umrtveny.

Pokud podúloha nemá žádné přípustné řešení nebo pokud pro ni dostaneme odhad, který je horší než L, pak tuto podúlohu okamžitě umrtvíme a do seznamu živých podúloh ji vůbec nezapisujeme.

Ze seznamu živých podúloh vybíráme zpravidla tu, která má nejlepší odhad. Tuto podúlohu ze seznamu odstraníme, rozvětvíme ji na podúlohy, pro každou z nich

ihned vypočteme odhad a případně ji zařadíme do seznamu živých podúloh, jak bylo popsáno výše.

Tento postup opakujeme, dokud je v seznamu nějaká živá podúloha. Výsledné řešení $\check{r}$ pak je optimálním řešením úlohy.

11.3.7 Příklad. Postup výpočtu metodou větví a mezí předvedeme na problému batohu (viz 2.4.2, str. 46), tj. na negrafové úloze, aby vynikla obecná použitelnost metody. Další příklady viz 12.4, str. 205.

Mějme batoh o kapacitě 40 a pět předmětů s těmito vahami a cenami:

předmět i:	A	B	C	D	E
cena c_i:	50	107	31	31	69
váha v_i:	14	30	9	10	27

Jako prostor řešení vezmeme $\mathbb{R}^5$, jeho prvky jsou uspořádané pětice čísel $x_1, \ldots, x_5$, která určují, kolikrát zabalíme do batohu jednotlivé předměty. Omezující podmínky určují, že $0 \leq x_i \leq 1$ a $\sum_{i=1}^{5} v_i x_i \leq 40$. Nepříjemnou omezující podmínkou zde je, že všechna x_i musí být navíc celočíselná.

Pro účely snadného řešení relaxované úlohy si předměty seřadíme sestupně podle jejich „měrné ceny“, tj. podle podílu cena/váha. Ve výše uvedeném zadání je toto seřazení již provedeno.

Optimální řešení relaxované úlohy pak získáme velmi jednoduše tak, že do batohu budeme ukládat celé předměty v pořadí klesající měrné ceny, přičemž poslední z takto zařazených předmětů možná nebude zařazen celý. Pro naši úlohu tak dostaneme řešení o hodnotě 142,73 tvořené celým předmětem A a částí 26/30 předmětu B. Jako horní odhad použijeme celou část této hodnoty, tj. zde 142, neboť všechna přípustná řešení původní (nerelaxované) úlohy U mají hodnotu účelové funkce celočíselnou, a ta tedy nemůže přesáhnout 142.

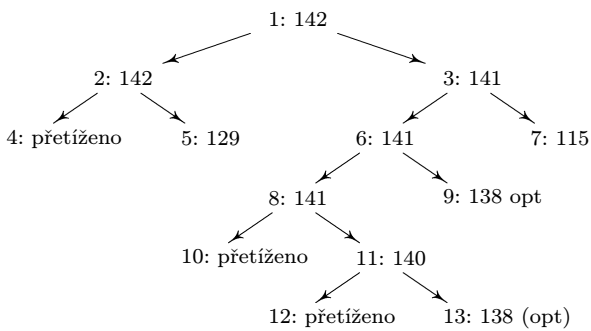
Větvení budeme provádět vždy na dvě podúlohy, a to tak, že si vybereme některý předmět, o němž v dané podúloze není ještě rozhodnuto, a v jedné z podúloh jej pevně zařadíme, zatímco ve druhé jej zakážeme. V obou podúlohách se tedy rozhoduje o menším počtu předmětů. Přidané omezující podmínky, tj. příkazy a zákazy, budeme zkráceně označovat symboly `+` a `-` tak, že např. `-+-..` bude znamenat, že 1. a 3. předmět jsou zakázány, 2. předmět je přikázán a o ostatních předmětech není v podúloze rozhodnuto.

Průběh řešení je zaznamenán v následující tabulce. Podúlohy jsou očíslovány pořadovými čísly v pořadí, jak byly vytvořeny větvením. K větvení byla vybírána vždy podúloha s nejlepším odhadem, a to v pořadí 1, 2, 3, 6, 8, 11.

číslo podúlohy	přidaná omezení	horní odhad	komentář
1	`.....`	142	větveno podle A na podúlohy 2 a 3
2	`+....`	142	větveno podle B na podúlohy 4 a 5
3	`-....`	141	větveno podle B na podúlohy 6 a 7

4	`++...`	–	přetíženo
5	`+-...`	129	umrtveno po zpracování podúlohy 9
6	`-+...`	141	větveno podle C na podúlohy 8 a 9
7	`--...`	115	umrtveno po zpracování podúlohy 9
8	`-++..`	141	větveno podle D na podúlohy 10 a 11
9	`-+-..`	138	(optimum) viz komentář v textu
10	`-+++.`	–	přetíženo
11	`-++-.`	140	větveno podle E na podúlohy 12 a 13
12	`-++-+`	–	přetíženo
13	`-++--`	138	další optimum

Strom řešení je na obrázku 11.1.



Obrázek 11.1: Strom řešení k příkladu 11.3.7. U vrcholů jsou napsána čísla podúloh a hodnoty horních odhadů.

Podúloha 9 vyžaduje podrobnější komentář: Řešením relaxované verze podúlohy 9 jsme zde (náhodou) dostali celočíselné řešení. Získali jsme tedy přípustné řešení celé úlohy, aniž by bylo nutno podúlohu větvit. V té době to bylo první přípustné řešení, které jsme nalezli. Použili jsme je ihned k umrtvení podúloh 5 a 7, které v té době byly ještě živé, ale měly ve srovnání s podúlohou 9 horší odhad.

Stojí za zmínku, že v této chvíli jsme již měli řešení, o kterém se později ukázalo, že je optimální, ale v té době jsme to ještě nevěděli, protože stále ještě zbývala živá podúloha 8 s odhadem, který dával naději na zlepšení. Bylo tedy nutno ve výpočtu pokračovat a teprve po rozvětvení podúloh 8 a 11 se ukázalo, že lepší řešení neexistuje.

11.3.8 Backtracking versus větve a meze. Obě metody mají některé rysy společné a u některých algoritmů je těžké říci, o kterou z obou metod jde.

Backtracking použitý k řešení optimalizační úlohy lze pokládat za speciální případ metody větví a mezí. Na posloupnost, která v backtrackingu tvoří částečné

řešení, se totiž můžeme dívat jako na podúlohu, v níž několik prvých členů posloupnosti je pomocí přidaných omezujících podmínek pevně určeno. Prodloužení posloupnosti pak znamená přidání nové omezující podmínky.

Při backtrackingu však prohledáváme strom řešení vždy do hloubky (jinak by to nebyl backtracking), zatímco v metodě větví a mezí lze další postup řešení volit svobodněji, a tedy zpravidla výhodněji (např. pomocí priorit, viz 11.3.5). V backtrackingu lze k zamezení zbytečného větvení použít i jiné triky než dolní, popř. horní odhady. A konečně, backtracking lze použít také k řešení úloh, které nemají optimalizační charakter (viz např. 13.3.1).

11.4 Heuristické algoritmy

V praxi mnohdy vystačíme s řešením, které je „dostatečně dobré“, které však získáme rychle. Algoritmy, které poskytují takováto řešení, nazýváme *heuristické*. Tyto algoritmy bývají založeny na nejrůznějších principech.

Mnohé heuristické algoritmy pracují tzv. *hladovým* způsobem: řešení vytváří (konstruují) postupně a v každém kroku se snaží o co největší zlepšení účelové funkce, popř. o co největší úsporu. Samozřejmě se může stát, že se „chamtivost nevyplatí“ a výsledkem je řešení, které není optimální. Úlohy, v nichž hladový postup zaručeně vede k optimálnímu řešení, jsou spíše výjimečné, viz poznámka u algoritmu 4.3.7, str. 64, pro minimální kostru.

Některé heuristické algoritmy řešení mnohokrát opakují s využitím náhody a z takto získaných řešení vybírají nejlepší.

Další heuristické algoritmy využívají tzv. *lokální průzkum*: vezmou nějaké stávající řešení (zpravidla získané nějakou jinou heuristikou) a systematicky je pozměňují, tj. v jistém smyslu prozkoumávají okolí stávajícího řešení. Pokud některé z pozměněných řešení je lepší než řešení stávající, prohlásí toto lepší řešení za stávající a pokračují průzkumem okolí tohoto nového řešení. Toto se opakuje, dokud se daří řešení zlepšovat. Samozřejmě můžeme skončit i dříve, pokud najdeme řešení, jehož kvalita nám postačuje nebo pokud už nemáme čas na další zlepšování.

Metodu lokálního průzkumu lze dále zdokonalit s využitím náhody. Jednak můžeme s nějakou pravděpodobností akceptovat i zhoršení účelové funkce v naději, že takto později objevíme celkově lepší řešení, jednak můžeme celý postup mnohokrát opakovat z jiného, náhodně zvoleného výchozího řešení.

Další, složitější heuristické algoritmy jsou „ošizené“ varianty backtrackingu nebo metody větví a mezí, v nichž používáme nepřesné (ne zcela spolehlivé) odhady. Prohledáváme pak menší strom řešení a výpočet se může podstatně urychlit, ale může se stát, že díky nepřesnému odhadu přeskočíme prohledávání podstromu, který obsahuje optimální řešení.

Obecně platí, že heuristické algoritmy bývají „šity na míru“ pro určitý typ úlohy, jejich úspěšnost závisí na tom, jak se podaří využít specifických rysů daného typu úlohy i specifických rysů instancí, které chceme řešit. Při jejich návrhu hodně záleží i na zkušenosti a citu pro řešený problém.

Kapitola 12

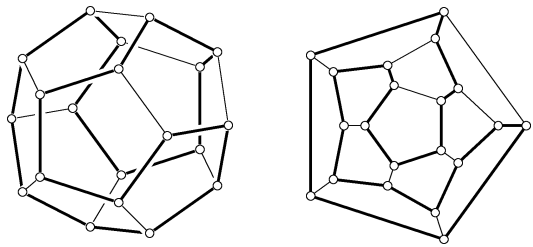
Hamiltonovské cesty a kružnice

12.1 Základní pojmy a aplikace

12.1.1 Definice. *Hamiltonovská cesta* v grafu G je taková cesta, která obsahuje všechny vrcholy grafu G.

Podobně definujeme *hamiltonovskou kružnici* a *hamiltonovský cyklus* jako kružnici nebo cyklus procházející přes všechny vrcholy grafu.

12.1.2 Hamiltonova hádanka. Výše uvedené pojmy jsou nazvány podle irského matematika W. R. Hamiltona, který v polovině 19. století zveřejnil v zeměpisném časopise hádanku, v níž šlo o nalezení uzavřené trasy vedoucí přes dvacet měst, která byla rozmístěna ve vrcholech pravidelného dvanáctistěnu, přičemž se přes žádné město nesmělo cestovat dvakrát. Příslušný graf je na obr. 12.1.



Obrázek 12.1: Dvě nakreslení grafu pravidelného dvanáctistěnu s vyznačenou hamiltonovskou kružnicí. Obrázek vpravo lze chápat jako „širokoúhlý“ pohled dovnitř skrz přední stěnu dvanáctistěnu.

12.1.3 Typy hamiltonovských úloh. Úlohy, ve kterých jde o hamiltonovské pojmy, mají mnoho variant a lze je třídit z několika různých hledisek.

Především je třeba odlišit úlohy existenční a optimalizační. V existenčních úlohách jde o zjištění existence nebo dokonce o nalezení hamiltonovské cesty, kružnice nebo cyklu v daném grafu.

V optimalizačních úlohách jsou hrany grafu ohodnoceny délkami a požaduje se nalezení hamiltonovské cesty, kružnice nebo cyklu o nejmenší délce. Optimální řešení se přitom často hledá v úplném grafu, protože obecnější úlohy lze na tento speciální případ velmi snadno převést (viz 12.2.2).

Dále úlohy dělíme na orientované a neorientované podle toho, zda hledáme orientovanou nebo neorientovanou hamiltonovskou cestu, případně zda hledáme hamiltonovský cyklus nebo kružnici.

V případě hamiltonovských cest musíme navíc rozlišit tři až čtyři varianty podle toho, zda je v zadání úlohy předepsán vrchol, ve kterém musí hamiltonovská cesta začínat, a vrchol, kde musí končit.

Ve všech těchto úlohách se můžeme omezit na prosté grafy bez smyček.

12.1.4 Obtížnost hamiltonovských úloh. Všechny výše uvedené typy hamiltonovských úloh jsou pro obecné grafy NP-těžké.

Stojí za zmínku, že kdybychom tyto úlohy „maličko“ pozměnili a namísto sledu, který obsahuje přesně jedenkrát každý vrchol, požadovali sled, který obsahuje přesně jedenkrát každou hranu, dostali bychom snadno řešitelné úlohy, kterými jsme se zabývali v kapitole 10 o eulerovských tazích.

12.1.5 Problém obchodního cestujícího. Jde o nalezení nejkratší hamiltonovské kružnice v úplném neorientovaném grafu, jehož hrany jsou ohodnoceny délkami.

Název (a popularita) této úlohy je založen na následující představě: obchodní cestující má za úkol navštívit v libovolném pořadí n měst a vrátit se zpět tak, aby jeho trasa byla co nejkratší. Přitom se předpokládá, že vzdálenosti mezi všemi dvojicemi měst jsou předem známé a symetrické (v obou směrech je vzdálenost stejná).

Často se mluví také o orientované variantě problému obchodního cestujícího – hledá se nejkratší hamiltonovský cyklus v úplném orientovaném grafu.

12.1.6 Příklad: Dopravní úlohy. Problém obchodního cestujícího má četné aplikace, v nichž jde o optimalizaci pohybu dopravního prostředku, např. při rozvozu zboží, vybírání poštovních schránek apod. Stejný problém se však vyskytuje např. při vrtání velkého počtu děr do desky s plošnými spoji pomocí číslicově řízené vrtačky (minimalizace přesunů).

V některých případech jde o nalezení (neuzavřené) hamiltonovské cesty. Např. při rozvozu pracovníků na roztroušená pracoviště můžeme požadovat, aby se dopravní prostředek nevracel prázdný, ale aby po skončení prací opět naložil v obráceném pořadí všechny pracovníky. Výchozí vrchol je přitom pevně určen, zatímco koncový obvykle nikoli.

Mnohé praktické úlohy jsou obecnější a zahrnují problém obchodního cestujícího jako částečný případ: jestliže se při rozvozu zboží nevejde všechno do jednoho auta,

je třeba vykonat několik jízd (lhostejno, zda postupně nebo několika auty najednou). Optimální rozvoz by pak měl mít tu vlastnost, že každá jízda je optimálním řešením problému obchodního cestujícího pro podmnožinu těch míst, která jsou touto jízdou zásobována.

12.1.7 Příklad: Plánování procesů. Mějme nějaké výrobní zařízení (třeba chemický reaktor), kterým máme periodicky vyrábět n produktů $p_1, p_2, \ldots, p_n$. Má-li po výrobě produktu p_i bezprostředně následovat výroba produktu p_j, pak v závislosti na těchto dvou produktech buď je, anebo není třeba zařízení vyčistit, nastavit nebo seřídit, což stojí nějaký čas nebo peníze. Úkolem je najít cyklický plán výroby všech produktů (tj. cyklické pořadí produktů) bez ztrátových časů.

Tuto úlohu lze snadno převést na hledání hamiltonovského cyklu v grafu, jehož n vrcholů odpovídá produktům $p_1, p_2, \ldots, p_n$ a jehož hrany vyjadřují možnost změn vyráběných produktů bez časových nebo peněžních ztrát.

Jestliže v této úloze hamiltonovský cyklus neexistuje, cyklická výroba bez ztrátových časů není možná. V takovém případě se můžeme snažit alespoň o minimalizaci ztrát, což vede na orientovanou variantu problému obchodního cestujícího.

Má-li být výroba jednorázová (každý produkt se má vyrábět jen jedenkrát), pak odpadnou náklady na uvedení výrobního zařízení do výchozího stavu, což vede k hledání nejkratší orientované hamiltonovské cesty.

12.2 Vzájemné převody úloh

Situace, kdy máme řešit nějakou úlohu, ale nemáme k dispozici vhodnou metodu řešení (nebo počítačový program), je bohužel častá. Mnohdy si však lze pomoci tím, že danou úlohu převedeme (transformujeme) na úlohu jinou, kterou řešit dovedeme, a z nalezeného řešení této druhé úlohy pak odvodíme řešení úlohy původní. Dovednost převádět jednu úlohu na druhou je obecně velice užitečná. V případě hamiltonovských úloh, které mají značný počet variant, je toto „umění“ tím užitečnější.

Souhrnně lze konstatovat, že všechny existenční hamiltonovské úlohy lze vzájemně převádět (každou na každou) a totéž platí i o úlohách optimalizačních. Navíc lze každou existenční úlohu převést na analogickou úlohu optimalizační. Všechny tyto převody jsou P-redukcemi, viz 11.1.7, str. 190.

Uvedeme zde jen některé převody, ostatní přenecháme čtenáři jako cvičení.

12.2.1 Převod existenční úlohy na optimalizační je snadný: Daný graf (prostý a bez smyček) doplníme na úplný graf. Délky původních hran zvolíme nulové, přidané hrany budou mít délku 1. Nyní platí, že v původním grafu existuje hamiltonovská cesta (kružnice, cyklus) právě tehdy, když v odpovídajícím úplném grafu existuje hamiltonovská cesta (kružnice, cyklus) o nulové délce. Žádná cesta (kružnice, cyklus) nemůže mít délku zápornou, proto nejkratší cesta, má-li nulovou délku, je řešením původní existenční úlohy. Pokud nejkratší cesta v úplném grafu má délku kladnou, pak v původním grafu cesta s požadovanými vlastnostmi neexistuje: kdyby existovala, musela by mít nulovou délku.

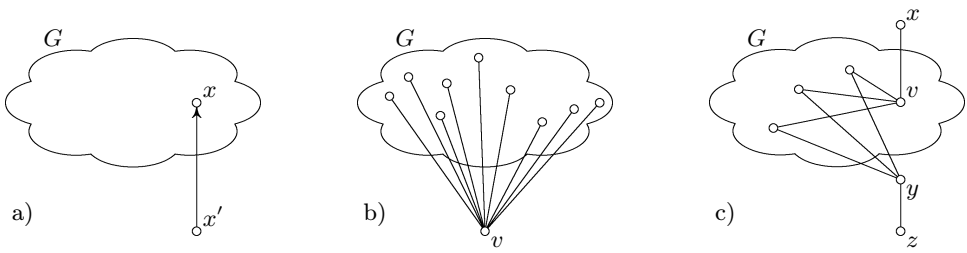
12.2.2 Převod optimalizační úlohy na úplný graf. Podobný trik se používá k převodu optimalizační úlohy v obecném grafu na analogickou optimalizační úlohu v grafu úplném. Původní graf doplníme na úplný, přičemž jako délku nově přidaných hran zvolíme nějaké dost velké kladné číslo M. Jako M lze použít např. součet absolutních hodnot délek všech hran. Potom najdeme optimální řešení v úplném grafu. Pokud toto řešení neobsahuje žádnou z přidaných hran, je toto řešení i optimálním řešením úlohy pro původní graf. Pokud naopak nějakou přidanou hranu obsahuje, pak v původním grafu přípustné řešení neexistuje.

12.2.3 Fixovaný počáteční vrchol. Hledáme-li hamiltonovskou cestu, jejíž počáteční vrchol x v grafu G je pevně dán, můžeme takovou úlohu převést na obdobnou úlohu ve větším grafu G', kde již počáteční vrchol není předepsán. Trik spočívá v tom, že ke grafu G přidáme kopii x' vrcholu x a hranu z vrcholu x' do x (viz obr. 12.2a). Poněvadž z vrcholu x' vychází v G' jen jediná hrana, nemůže žádná hamiltonovská cesta v grafu G' začínat jinde než v tomto přidaném vrcholu a odtud musí pokračovat přidanou hranou do vrcholu x. Všechny hamiltonovské cesty v G začínající v x a všechny hamiltonovské cesty v G' si vzájemně jednoznačně odpovídají, liší se pouze v hraně (x, x'). Postupujeme tedy tak, že v G' najdeme hamiltonovskou cestu (např. nejkratší), pak vynecháním přidané hrany dostaneme (stejně dlouhou) hamiltonovskou cestu v G, která začíná tam, kde to bylo předepsáno, tj. v x.

Stejný trik lze použít, je-li předepsán koncový vrchol hamiltonovské cesty.

Pro optimalizační úlohu by bylo možno postupovat stejně (délka přidané hrany by byla nulová), existuje však lepší možnost. Popíšeme ji pro orientovanou verzi úlohy: graf ponecháme beze změny, ale všem hranám z množiny $E^+(x)$ zvětšíme jejich délku o dostatečně velké číslo M a hranám z množiny $E^-(x)$ zvětšíme jejich délku o hodnotu $2M$. Tím všechny orientované hamiltonovské cesty, které začínají ve vrcholu x, prodloužíme o hodnotu M, zatímco všechny ostatní hamiltonovské cesty prodloužíme o $3M$ nebo $2M$ podle toho, zda vrcholem x prochází nebo v něm končí. Cesty, které nezačínají v x, jsou tím natolik znevýhodněny, že nejkratší hamiltonovská cesta při změněných délkách bude začínat v x (pokud taková existuje).

12.2.4 Převod neorientované cesty na kružnici. Úlohu najít neorientovanou hamiltonovskou cestu s libovolnými krajními vrcholy v grafu G lze převést na úlohu



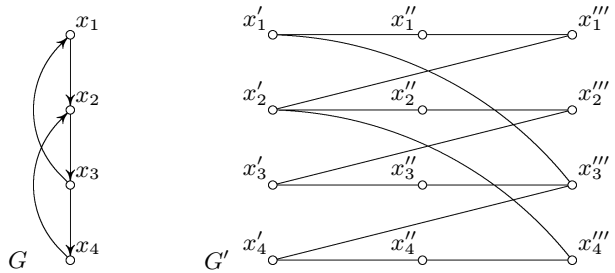
Obrázek 12.2: Ilustrace převodů hamiltonovských úloh.

najít hamiltonovskou kružnici ve větším grafu G', který získáme tak, že ke grafu G přidáme jeden nový vrchol v a spojíme jej hranami se všemi vrcholy grafu G. V optimalizační úloze zvolíme délky nových hran nulové. V grafu G' existuje hamiltonovská kružnice právě tehdy, když v G existuje hamiltonovská cesta. Viz obr. 12.2b.

12.2.5 Převod hamiltonovské kružnice na neorientovanou cestu. Úlohu najít hamiltonovskou kružnici v grafu G lze naopak převést na hledání hamiltonovské cesty s libovolnými krajními vrcholy ve větším grafu G', který získáme takto: Ke grafu G přidáme tři nové vrcholy x, y, z a v grafu G zvolíme libovolně vrchol v. Ke každé hraně mezi vrcholem v a jeho sousedem a v grafu G přidáme v grafu G' stejně dlouhou hranu mezi a a y. Navíc spojíme vrcholy y a z a vrcholy v a x hranami o nulové délce. V takto sestrojeném grafu G' existuje hamiltonovská cesta (z x do z) právě tehdy, když v G existuje (stejně dlouhá) hamiltonovská kružnice. Viz obr. 12.2c.

12.2.6 Převod orientované verze na neorientovanou. Jako poslední ukážeme převod orientovaných verzí hamiltonovských úloh na verze neorientované. Každému vrcholu x_i původního orientovaného grafu přiřadíme trojici vrcholů x'_i, x''_i, x'''_i. Každý vrchol x''_i spojíme neorientovanými hranami o nulové délce s vrcholy x'_i a x'''_i. Každé orientované hraně z x_i do x_j v grafu G přiřadíme v grafu G' stejně dlouhou neorientovanou hranu spojující vrcholy x'''_i a x'_j. Viz obr. 12.3.

Vtip této konstrukce je v tom, že všechny dvoučárkované vrcholy musí hamiltonovská cesta procházet stejným směrem, tj. buď všechny od jednočárkované ke tříčárkované, nebo všechny naopak. Orientované hamiltonovské cesty a cykly v původním grafu G tedy vzájemně jednoznačně odpovídají stejně dlouhým neorientovaným hamiltonovským cestám a kružnicím v grafu G'.



Obrázek 12.3: Převod orientované verze hamiltonovských úloh na neorientovanou.

12.2.7 Cvičení. Promyslete detaily převodu úlohy najít nejkratší neorientovanou cestu s fixovaným počátečním i koncovým vrcholem na úlohu bez fixovaných vrcholů. Použijte analogii k 12.2.3.

12.2.8 Cvičení. Jak velké musí být číslo M, které je použito v převodech 12.2.2 a 12.2.3?

12.2.9 Cvičení. Najděte příklad ohodnoceného grafu, v němž nejkratší neorientovaná hamiltonovská cesta není částí nejkratší hamiltonovské kružnice.

Jinak řečeno, najděte příklad, který ukazuje, že nejkratší hamiltonovskou cestu nelze hledat tak, že vezmeme řešení problému obchodního cestujícího a vynecháme z něj nejdelší hranu.

12.2.10 Správnost převodu je třeba dokázat. Předchozí cvičení ukazuje, že ne každý nadějně vypadající „převod“ jedné úlohy na druhou je korektní a skutečně použitelný. Chcete-li nějaký převod použít bez rizika chyby, měli byste o tomto převodu *dokázat* dvě věci:

- pokud původní úloha má řešení, pak pomocí převodu takové řešení najdeme,
- pokud původní úloha řešení nemá, pak to pomocí převodu spolehlivě zjistíme.

12.3 Existenční hamiltonovské úlohy

Existenční hamiltonovské úlohy obecně patří k NP-těžkým úlohám. V některých speciálních případech však přesto lze rychle prokázat, že řešení existuje nebo naopak neexistuje.

12.3.1 Cvičení. Graf, který obsahuje most (viz 4.4.1, str. 67), nemůže obsahovat hamiltonovskou kružnici. Najděte graf, který žádný most neobsahuje a přesto v něm hamiltonovská kružnice také neexistuje.

12.3.2 Věta. (Chvátal 1972.) Nechť G je prostý graf bez smyček s $n \geq 3$ vrcholy. Jestliže $d_1 \leq d_2 \leq \ldots \leq d_n$ je soubor stupňů jeho vrcholů a jestliže

$$d_k \leq k \;\Rightarrow\; d_{n-k} \geq n-k \qquad \text{pro} \quad 1 \leq k < n/2\,, \tag{12.1}$$

pak graf G obsahuje hamiltonovskou kružnici.

Jestliže naopak posloupnost čísel $d_1 \leq d_2 \leq \ldots \leq d_n$ nesplňuje podmínku (12.1), pak existuje graf o n vrcholech, který neobsahuje hamiltonovskou kružnici a pro jehož všechny vrcholy v_i platí $d(v_i) \geq d_i$.

Poznámka: Prvá část věty zaručuje existenci hamiltonovské kružnice, jsou-li stupně vrcholů „dost velké“. Druhá část vlastně říká, že lepší podmínka založená pouze na stupních vrcholů již neexistuje. Dodejme, že pokud není splněna podmínka (12.1) (a takových grafů je spousta), hamiltonovská kružnice může a nemusí existovat, věta o tom nic neříká.

Důkaz této věty není jednoduchý, lze jej najít např. v knize [Nešetřil79]. Poznamenejme pouze, že důkaz je proveden sporem a nedává žádný návod k hledání hamiltonovské kružnice, a to ani v případě, když věta její existenci zaručuje.

12.3.3 Věta. (Ghouilla-Houri 1960.) Nechť G je silně souvislý prostý orientovaný graf s n vrcholy a nechť pro každý vrchol x platí $d^+(x) + d^-(x) \geq n$. Potom graf G obsahuje hamiltonovský cyklus.

Poznámka: Také tato věta nedává návod, jak hamiltonovský cyklus hledat, a to ani v případě, kdy jeho existenci zaručuje.

12.3.4 Cvičení. Zjistěte, zda existuje hamiltonovská kružnice v úplném bipartitním grafu $K_{6,8}$?

12.3.5 Hledání hamiltonovských cest, cyklů a kružnic. Můžeme použít backtracking (viz 11.2, str. 191), který v tomto případě bude modifikací algoritmu 3.3.10 k prozkoumání všech cest, které začínají ve zvoleném vrcholu. Algoritmus 3.3.10 lze snadno doplnit o test, zda právě nalezená cesta je hamiltonovskou cestou, kružnicí, popř. cyklem. Takto můžeme získat dokonce úplný seznam všech hamiltonovských cest, kružnic či cyklů v daném grafu, je to však časově velice náročné i pro nepříliš velké grafy.

Celý postup lze poněkud zrychlit tím, že se v každém kroku snažíme rozpoznat hrany, které musíme nebo naopak nesmíme přidávat ke stávající cestě, aby ji bylo možno doplnit na hamiltonovskou. Dosažitelné zrychlení však je řádově jen asi dvojnásobné. Detaily lze najít v [Chr75].

Další možností je převod na optimalizační úlohu, viz 12.2.1.

12.3.6 Heuristické postupy. Pro řešení hamiltonovských úloh byla vypracována také řada poměrně rychlých algoritmů, u nichž však není zaručeno, že řešení najdou vždy, když existuje. V jednom z nich např. prodlužujeme stávající cestu, dokud je to možné. Nelze-li již cestu prodloužit, pokoušíme se závěrečnou část cesty odpojit a připojit obráceně (viz obr. 12.4) a pokračovat v prodlužování z jiného vrcholu.

Obrázek 12.4: Přepojování cest.

Pro zvýšení naděje na úspěch se doporučuje tyto algoritmy použít opakovaně na týž graf s náhodně přečíslovanými vrcholy. Podrobnosti lze najít v [Chr75] a [Kučera83].

12.4 Optimalizační hamiltonovské úlohy

Optimalizační hamiltonovské úlohy jsou přinejmenším stejně obtížné jako úlohy existenční, neboť každou existenční úlohu lze velmi jednoduše převést na řešení úlohy optimalizační (viz 12.2.1). Všechny tyto úlohy jsou pro obecné grafy NP--těžké.

Problém obchodního cestujícího lze asi nejjednodušeji řešit takzvanou metodou hrubé síly, tedy jednoduchým, ale pracným vyzkoušením všech přípustných řešení, tj. všech permutací (pořadí) vrcholů: pro každou permutaci vypočteme délku odpovídající hamiltonovské kružnice a vybereme z nich nejkratší. Počet všech přípustných řešení však roste jako faktoriál počtu vrcholů, proto přidání i jen jediného vrcholu prodlouží výpočet mnohonásobně. Tento primitivní způsob řešení je prakticky použitelný jen na grafy s nejvýše asi tak tuctem vrcholů.

Pro obecnější hamiltonovské optimalizační úlohy lze podobně použít jednoduchý backtracking (např. podle 3.3.10, str. 58) k nalezení všech hamiltonovských cest, kružnic nebo cyklů a z nich pak opět vybrat nejkratší. Pokud však při tom nepoužijeme nějaké triky pro urychlení výpočtu, je to jen jiná forma metody hrubé síly. V grafech, které mají hodně hran (zejména pak v úplných grafech), to není o mnoho rychlejší než výše zmíněné vyzkoušení všech permutací.

Dále uvedeme několik algoritmů, které lze použít na úlohy o něco rozsáhlejší (několik desítek vrcholů).

12.4.1 Hledání nejkratší neorientované hamiltonovské cesty. Popíšeme algoritmus, který je jednoduchým použitím metody větví a mezí (viz 11.3, str. 193). Úlohu přeformulujeme takto:

Hledáme nejlevnější podgraf H daného grafu G takový, že:

1. podgraf H je kostrou grafu G,
2. všechny vrcholy podgrafu H mají stupeň $d_H(x) \leq 2$.

Prvá z uvedených omezujících podmínek je příjemná ve smyslu 11.3.4, relaxováním (uvolněním) druhé (nepříjemné) podmínky totiž dostaneme známou úlohu o minimální kostře, kterou lze snadno řešit, viz 4.3, str. 63.

Pokud v nejlevnější kostře, kterou dostaneme řešením relaxované úlohy, mají všechny vrcholy stupeň nejvýše 2, pak tato kostra je nejlevnější nejen mezi kostrami, ale i mezi všemi hamiltonovskými cestami, máme tedy optimální řešení.

Pokud má některý vrchol x v kostře H stupeň $d_H(x) > 2$, pak úlohu rozvětvíme na podúlohy. Nejjednodušší verze větvení je založena na faktu, že hrany z množiny $E_H(x)$ nemohou být všechny obsaženy v téže hamiltonovské cestě. Proto pro každou hranu $e \in E_H(x)$ vytvoříme podúlohu, v níž je hrana e zakázána. Tím dostaneme $d_H(x)$ podúloh, které pak řešíme stejným postupem jako původní úlohu: hledáme nejlevnější kostru a testujeme stupně vrcholů v této kostře. Je-li třeba, podúlohu dále větvíme zakazováním dalších hran.

Dodejme, že po zákazu hrany není nutno opakovat celý výpočet nejlevnější kostry. Zákazem hrany se dosavadní minimální kostra rozpadla na dvě komponenty a tyto komponenty stačí spojit nejlevnější dosud nezakázanou hranou. Vyplývá to z věty 4.3.14, str. 66.

Právě popsaný postup řešení je předveden v příkladě 12.4.2.

Dodejme dále, že výše uvedený způsob větvení na podúlohy je sice jednoduchý, ale neefektivní, poněvadž vytvořené podúlohy nemají disjunktní množiny přípustných řešení. Snadno se pak stane, že totéž řešení se zbytečně vyskytuje v několika

větvích stromu řešení, strom se příliš rozrůstá a výpočet je zbytečně pomalý. Platí to zejména v případě, kdy stupeň vrcholu v kostře je větší než tři.

Efektivnější způsob větvení využívá nejen zákazy, ale i vynucenou přítomnost hrany v kostře:

- zakážeme hranu $e_1 \in E_H(x)$,
- vynutíme hranu e_1 a zakážeme $e_2 \in E_H(x)$,
- vynutíme hrany e_1, e_2 a zakážeme všechny ostatní hrany z množiny $E_G(x)$ (pozor, nikoli pouze z $E_H(x)$).

Všimněte si, že u třetí z těchto podúloh využíváme fakt, že nejvýše dvě hrany z množiny $E_G(x)$ mohou být obsaženy v hamiltonovské cestě. Dále si všimněte, že tímto způsobem dostaneme vždy jen tři podúlohy nezávisle na stupni vrcholu x v kostře.

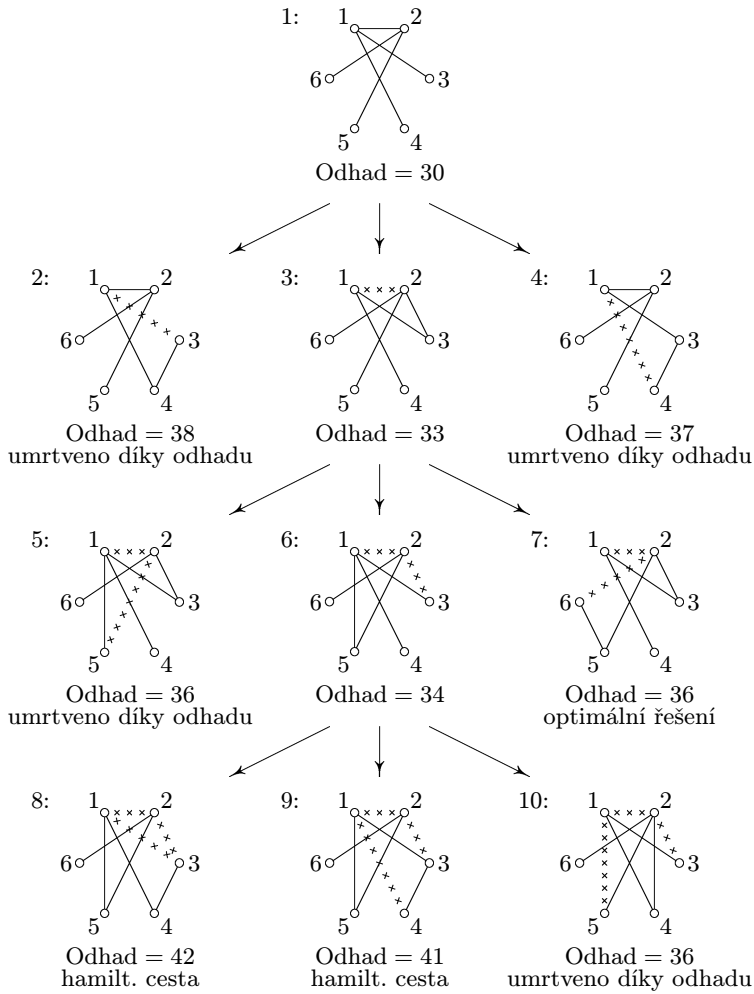
12.4.2 Příklad. Hledejme nejkratší hamiltonovskou cestu v úplném grafu, ve kterém jsou délky hran dány maticí

$$\begin{pmatrix} \infty & 7 & 2 & 3 & 11 & 17 \\ 7 & \infty & 10 & 13 & 8 & 10 \\ 2 & 10 & \infty & 10 & 19 & 15 \\ 3 & 13 & 10 & \infty & 16 & 15 \\ 11 & 8 & 19 & 16 & \infty & 13 \\ 17 & 10 & 15 & 15 & 13 & \infty \end{pmatrix}.$$

K řešení použijeme právě popsaný algoritmus 12.4.1. K větvení budeme vybírat vždy živou podúlohu s nejnadějnějším (tj. zde nejmenším) odhadem, větvení budeme dělat prvým a jednodušším z obou popsaných způsobů (tj. pouze zakazujeme jednotlivé hrany). Průběh výpočtu je patrný z následující tabulky a z obr. 12.5 na následující straně.

podúloha	zakázané hrany	odhad	komentář
1	—	30	větveno na podúlohy 2, 3, 4
2	(1, 3)	38	umrtveno díky odhadu
3	(1, 2)	33	větveno na podúlohy 5, 6, 7
4	(1, 4)	37	umrtveno díky odhadu
5	(1, 2), (2, 5)	36	umrtveno díky odhadu
6	(1, 2), (2, 3)	34	větveno na podúlohy 8, 9, 10
7	(1, 2), (2, 6)	36	optimální řešení
8	(1, 2), (2, 3), (1, 3)	42	umrtveno díky odhadu
9	(1, 2), (2, 3), (1, 4)	41	umrtveno díky odhadu
10	(1, 2), (2, 3), (1, 5)	36	umrtveno díky odhadu

12.4.3 Cvičení. Řešte znovu příklad 12.4.2, ale použijte efektivnější způsob větvení, popsaný v 12.4.1. Výsledek bude týž, ale dosáhnete jej s menším stromem



Obrázek 12.5: Strom řešení k příkladu 12.4.2.

řešení. (Obecně by výsledkem nemusela být stejná hamiltonovská cesta, ale délka výsledné cesty samozřejmě stejná být musí.)

12.4.4 Nejkratší hamiltonovský cyklus lze hledat rovněž metodou větví a mezí. Úlohu přeformulujeme takto:

Hledáme nejlevnější podgraf H daného grafu G takový, že:

1. hrany podgrafu H tvoří soustavu vrcholově disjunktních cyklů, které procházejí všemi vrcholy grafu G,
2. podgraf H je souvislý.

Prvou z podmínek opět pokládáme ve smyslu 11.3.4 za příjemnou, druhou za nepříjemnou. Relaxovanou úlohu, v níž je uvolněna (relaxována) podmínka 2, lze řešit převodem na přiřazovací úlohu 9.4, str. 173:

Každému vrcholu původního grafu je třeba přiřadit vrchol, který po něm v cyklu následuje, každý vrchol zároveň musí být přiřazen jako následník přesně jednomu vrcholu, totiž svému předchůdci. Toto přiřazení můžeme chápat jako perfektní párování v bipartitním grafu, který dostaneme tak, že každý vrchol původního grafu zdvojíme.

Jinou možností, jak hledat zmíněnou soustavu cyklů, je pomocí toků v síti. Hledáme nejlevnější cirkulaci takovou, že tok každým vrcholem má jednotkovou velikost. Tuto úlohu nejprve trikem podle 8.1.13, str. 134, převedeme na standardní úlohu hledání nejlevnější přípustné cirkulace a tu pak řešíme algoritmem 8.7.

Pokud nejkratší soustava cyklů nesplňuje podmínku 2, tj. pokud jde o soustavu více než jednoho cyklu, je zřejmé, že hrany některého (libovolného) z těchto cyklů nemohou být všechny obsaženy v cyklu hamiltonovském. Proto můžeme úlohu rozvětvit na podúlohy nejjednodušeji tak, že vybereme jeden z cyklů (doporučuje se, aby měl co nejméně hran) a pro každou jeho hranu e_i vytvoříme podúlohu U_i, v níž je hrana e_i zakázána.

V přiřazovací úloze lze zákaz hrany e_i docílit zvýšením její ceny na dostatečně vysokou hodnotu (tzv. prohibitivní cena). Přitom není nutno opakovat celý výpočet přiřazovací úlohy, stačí změnit cenu zakazované hrany a pokračovat od poslední fáze výpočtu. Při použití toku v síti docílíme zákazu změnou kapacity hrany e_i. Viz též poznámku 8.7.9 k algoritmu Out of Kilter.

Výše popsaný jednoduchý způsob větvení na podúlohy je neefektivní, neboť podúlohy nemají disjunktní množiny přípustných řešení. Disjunktnosti lze dosáhnout vynucováním hran podobně jako v 12.4.1.

Lepší výsledky (menší strom řešení) však dává následující způsob větvení: Zvolme některý cyklus a označme M množinu všech jeho vrcholů. Ve výsledném hamiltonovském cyklu musí zcela jistě být zahrnuta alespoň jedna hrana z množiny $W^+(M)$, jinými slovy, je třeba „donutit cyklus, aby opustil množinu M“. Toho lze docílit tím, že vytvoříme podúlohu pro každý vrchol $v_i \in M$, a to tak, že zakážeme všechny hrany, které z vrcholu v_i vedou do množiny M.

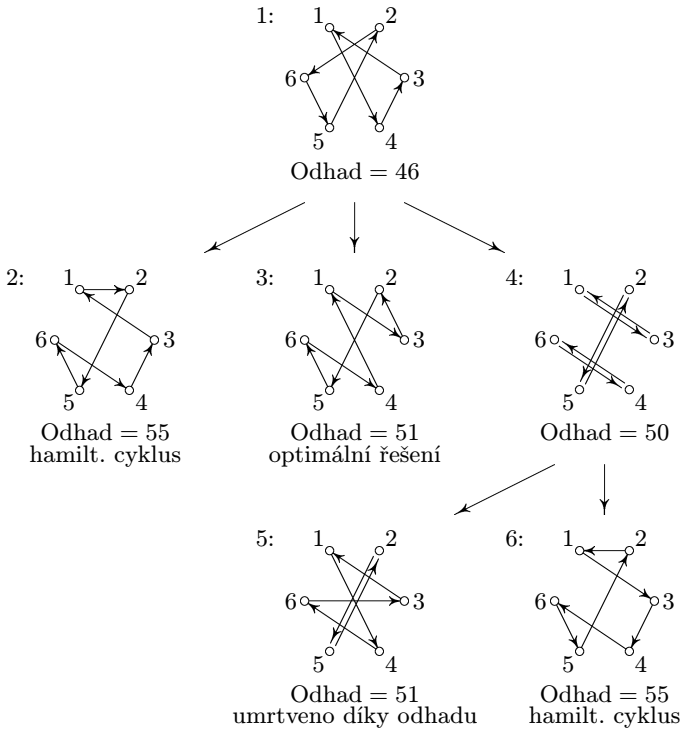
Uvedeným postupem lze řešit orientovanou verzi problému obchodního cestujícího. Lze jej samozřejmě použít i pro řešení verze neorientované (kde matice délek hran je symetrická).

12.4.5 Příklad. Použijeme právě popsanou metodu k nalezení nejkratšího hamiltonovského cyklu v úplném grafu s maticí délek hran z příkladu 12.4.2.

Průběh řešení je patrný z následující tabulky a obrázku 12.6 na následující straně. K větvení byla použita poslední z uvedených metod.

podúloha	zakázané hrany	odhad	komentář
1	–	46	větveno na podúlohy 2, 3, 4
2	(1, 3), (1, 4)	55	hamilt. cyklus
3	(3, 1), (3, 4)	51	optimální řešení

4	(4, 1), (4, 3)	50	větveno na podúlohy 5, 6
5	(4, 1), (4, 3), (1, 3)	51	umrtveno díky odhadu
6	(4, 1), (4, 3), (3, 1)	55	hamilt. cyklus



Obrázek 12.6: Strom řešení k příkladu 12.4.5.

12.4.6 1-strom a metoda penalizace. Jde o docela poučný trik, který lze použít k hledání nejkratší hamiltonovské kružnice, a tedy i k řešení problému obchodního cestujícího. Mějme graf s množinou vrcholů $V(G) = \{1, \ldots, n\}$.

1-strom je faktor H daného grafu G, který má stejný počet vrcholů a hran a vrchol 1 v něm má stupeň $d_H(1) = 2$. Jinak řečeno, libovolný 1-strom daného grafu o n vrcholech dostaneme tak, že vezmeme podgraf indukovaný množinou $\{2, \ldots, n\}$, najdeme nějakou kostru tohoto podgrafu a přidáme k ní vrchol 1 spolu s nějakými dvěma hranami z tohoto vrcholu. Každý 1-strom tedy obsahuje přesně jednu kružnici a tato kružnice prochází vrcholem 1.

Nyní můžeme úlohu najít nejkratší hamiltonovskou kružnici přeformulovat takto: hledáme podgraf H daného grafu G takový, že

1. podgraf H je 1-stromem,
2. každý vrchol x podgrafu H má stupeň $d_H(x) = 2$.

Opět platí, že prvá z obou podmínek je příjemná, protože nejlevnější 1-strom lze snadno najít pomocí algoritmu pro nejlevnější kostru (viz 4.3): najdeme nejlevnější kostru na množině $\{2, \ldots, n\}$ a přidáme dvě nejlevnější hrany z množiny $E_G(1)$.

Pokud nejlevnější 1-strom náhodou splňuje i druhou podmínku, pak tento 1-strom je zároveň hledanou nejkratší hamiltonovskou kružnicí.

Druhá z podmínek je však nepříjemná a činí úlohu obtížnou. Platnost této podmínky samozřejmě lze vynucovat metodou větví a mezí podobně jako při hledání nejkratší hamiltonovské cesty v 12.4.1. Na tuto metodu zpravidla stejně dojde, ale napřed je vhodné na podúlohu zkusit trik, kterému se říká *penalizace vrcholů*.

Každému vrcholu x grafu G přiřadíme číslo $p(x)$, tzv. *penále*. Každé hraně grafu, která spojuje vrcholy i a j, nyní změníme délku z hodnoty $a(i,j)$ na hodnotu $a(i,j)+$ $+\,p(i)+p(j)$ a v takto ohodnoceném grafu znovu vyhledáme nejlevnější 1-strom.

Vtip penalizace spočívá v tom, že délky všech hamiltonovských kružnic se vlivem penále zvětšily o stejnou hodnotu, totiž o dvojnásobek sumy všech penále, tj. o hodnotu $2\sum_{i=1}^{n} p(i)$. Délky ostatních 1-stromů se přitom mohly změnit jinými způsoby, některé mohly stoupnout, jiné klesnout. Proto, podaří-li se vhodně zvoleným penále docílit toho, že nejlevnější 1-strom bude zároveň hamiltonovskou kružnicí, budeme mít jistotu, že tento 1-strom (tj. kružnice) je také nejkratší ze všech hamiltonovských kružnic.

Penále, popř. jeho změna, se zpravidla odvozuje ze stupňů vrcholů v 1-stromu. Má-li vrchol vysoký stupeň, dostane kladné penále, tím se prodraží hrany z jeho okolí a je naděje, že se stupeň sníží. Má-li naopak vrchol stupeň 1, dostane záporné penále, ceny hran v jeho okolí klesnou a stupeň vrcholu se může zvětšit. Nejjednodušší pravidlo radí zvětšovat dosavadní penále vrcholu v o hodnotu $k(d(v)-2)$, kde konstanta k nemá být příliš velká, může být i rovna jedné. Různé způsoby volby penále a výpočetní zkušenosti jsou podrobně popsány v knize [Chr75].

Výpočet obvykle začíná s nulovým penále, tj. s původními délkami hran tím, že vypočteme nejlevnější 1-strom. Pokud není nalezena hamiltonovská kružnice, tj. pokud některý vrchol má jiný stupeň než 2, vypočteme změny (přírustky) penále pro jednotlivé vrcholy, změníme délky hran a znovu hledáme nejlevnější 1-strom.

Metoda penalizace bohužel nevede vždy k cíli. Existují grafy, kde sebedůmyslnější volba penále nevynutí vznik hamiltonovské kružnice. Výše popsaný výpočet by pak nikdy neskončil. Proto je nutno neúspěšný výpočet ukončit buď když zjistíme, že se opakuje stejná situace, nebo prostě po přiměřeném počtu neúspěšných penalizací.

Ovšem i neúspěšná penalizace není zbytečná, lze ji použít v metodě větví a mezí k získání lepšího dolního odhadu. Jestliže jsme při použití penále p dostali 1-strom o ceně C, pak číslo $C - 2\sum_{i=1}^{n} p(i)$ lze použít jako dolní odhad: délka žádné hamiltonovská kružnice (měřená v původních, tj. nepenalizovaných cenách) nemůže být kratší než toto číslo. V průběhu neúspěšné penalizace takovýchto odhadů získáme několik a jako odhad pro celou podúlohu použijeme nejlepší (tj. zde největší) z nich.

12.4.7 Cvičení. Metodu penalizace lze snadno upravit pro hledání nejkratší neorientované hamiltonovské cesty z daného počátečního vrcholu x do daného

koncového vrcholu y. V takovém případě namísto nejlevnějšího 1-stromu hledáme minimální kostru takovou, že vrcholy x a y v ní mají stupeň 1. Takovou minimální kostru lze snadno najít buď tak, že najdeme minimální kostru na množině vrcholů $V(G) \setminus \{x, y\}$ a přidáme nejlevnější hrany z vrcholů x a y, nebo (jednodušeji) vrcholům x a y dáme velmi vysoké počáteční penále.

Jak se změní délky všech hamiltonovských cest z x do y při použití penále p? Jakou hodnotu lze použít jako dolní odhad v metodě větví a mezí?

12.4.8 Heuristické algoritmy pro problém obchodního cestujícího. Poměrně dobré výsledky dává *metoda nejvzdálenějšího vkládání*:

Začneme s triviální kružnicí v nějakém vrcholu a do této kružnice postupně zařazujeme další vrcholy, dokud nedostaneme kružnici hamiltonovskou. K zařazení do kružnice vybíráme vrchol, který je od kružnice nejvíce vzdálen (odtud název metody). Přesněji, máme-li kružnici K, která prochází vrcholy $v_1, \ldots, v_k$, pak vybereme vrchol v, pro který je výraz $\min\{a(v, v_i) \mid i = 1, \ldots, k\}$ největší. Vrchol v pak zařadíme do kružnice mezi takové dva sousední vrcholy, aby se tím kružnice prodloužila co nejméně.

Stojí za zmínku, že o něco „hladovější“ postup, při němž se vkládá vrchol, který je ke kružnici nejblíže, dává v průměru o něco horší výsledky. Je vidět, že chamtivost se vždycky nevyplácí.

Dobré výsledky dává také lokální průzkum (viz 11.4, str. 198), v němž se stávající řešení snažíme vylepšovat prováděním „drobných změn“. Osvědčilo se kombinovat dva typy změn. Jedna spočívá v tom, že ze stávající hamiltonovské kružnice odstraníme dvě hrany a zbývající části kružnice propojíme „do kříže“ tak, aby znovu vznikla hamiltonovská kružnice. Druhý typ spočívá v odstranění vrcholu a jeho zařazení na jiné místo kružnice.

Kapitola 13

Barevnost, nezávislost, kliky

Pojmy uvedené v nadpise bývají někdy označovány jako kombinatorické. Třebaže mají důležité aplikace, jsou úlohy s nimi spojené většinou NP-těžké. Algoritmy pro barvení grafu a pro hledání nejpočetnějších nezávislých množin, které zde uvedeme, jsou založeny na backtrackingu.

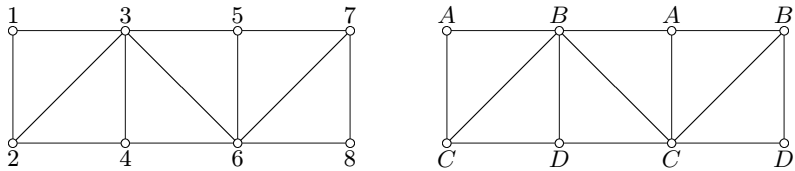
Všechny pojmy studované v této kapitole jsou nezávislé na orientaci a na násobnosti hran. Můžeme se tedy omezit na **prosté neorientované grafy.** Navíc se omezíme na grafy **bez smyček**.

13.1 Definice a základní fakta

13.1.1 Obarvení grafu G (nebo též obarvení vrcholů grafu) je ohodnocení vrcholů hodnotami z množiny B (takzvanými *barvami*), a to takové, že žádné dva sousední vrcholy nejsou ohodnoceny (obarveny) stejnou barvou. V roli barev můžeme brát libovolné prvky z libovolné množiny, často používáme např. čísla.

Graf nazýváme r-barevným, jestliže existuje jeho obarvení r barvami. *Barevnost grafu* (někdy též *chromatické číslo grafu* nebo *vrcholová barevnost*) je nejmenší počet barev, který je potřebný k obarvení grafu. Barevnost grafu G značíme $\chi(G)$.

13.1.2 Cvičení. Najděte obarvení grafu z obrázku 13.1 třemi barvami. Ověřte, že toto obarvení je až na záměnu barev jediné. Lze obarvit graf, který má smyčky?



Obrázek 13.1: Graf a jeho obarvení čtyřmi barvami A, B, C, D.

13.1.3 Nezávislá množina vrcholů v grafu G je taková množina, jejíž žádné dva vrcholy nejsou spojeny hranou. Podgraf indukovaný touto množinou je tedy diskrétní. *Maximální nezávislá množina* je nezávislá množina, která je maximální vzhledem k inkluzi, tj. nezávislá množina, ke které již nelze přidat žádný vrchol, aniž by přestala být nezávislou.

Nejpočetnější nezávislá množina je nezávislá množina, která má největší počet prvků (mezi všemi nezávislými množinami v grafu G). Každá nejpočetnější nezávislá množina je samozřejmě i maximální nezávislou množinou, ale naopak to neplatí. Jak nejpočetnějších, tak i maximálních nezávislých množin může být v témže grafu několik. Počet vrcholů v nejpočetnější nezávislé množině grafu G nazýváme *nezávislostí grafu* G. Nezávislost grafu G značíme $\alpha(G)$. V aplikacích se setkáváme také s úlohou najít nejdražší nezávislou množinu ve smyslu součtu ohodnocení vrcholů cenami.

Graf na obr. 13.1 vlevo má tyto maximální nezávislé množiny: $\{1,4,5,8\}$, $\{1,4,7\}$, $\{1,6\}$, $\{2,5,8\}$, $\{2,6\}$, $\{2,7\}$, $\{3,7\}$ a $\{3,8\}$. Nejpočetnější nezávislá množina byla uvedena jako první, nezávislost tohoto grafu je rovna 4. Množina $\{1,4,5\}$ je příkladem množiny, která je nezávislá, ale není maximální nezávislá.

Všimněte si, že množina vrcholů, které mohou být obarveny stejnou barvou, je vždy nezávislá.

13.1.4 Klika v grafu G je maximální podgraf grafu G, který je úplným (neorientovaným) grafem. Počet vrcholů v největší klice nazýváme *klikovostí grafu*. Klikovost grafu budeme značit $\omega(G)$.

V grafu na obr. 13.1 je každá klika trojúhelníkem, klikovost tohoto grafu je tedy rovna 3, v grafu je celkem šest klik.

Kliky a maximální nezávislé množiny spolu velmi úzce souvisí. Abychom to ukázali, sestrojme k danému grafu G tzv. *doplňkový graf* $-G$ takto: Oba grafy budou mít stejné vrcholy. V doplňkovém grafu $-G$ budou dva vrcholy spojeny hranou právě tehdy, když nejsou spojeny hranou v původním grafu G. Je zřejmé, že doplňkový graf $-(-G)$ k doplňkovému grafu $-G$ je shodný s původním grafem G. Stejně tak je zřejmé, že kliky v grafu G odpovídají vzájemně jednoznačně maximálním nezávislým množinám v doplňkovém grafu $-G$. Platí tedy $\alpha(G) = \omega(-G)$ a $\omega(G) = \alpha(-G)$.

13.1.5 Příklad: Skladování nebezpečných látek. Mějme n nebezpečných látek. Bezpečnostní předpisy zakazují skladovat některé dvojice látek ve stejné místnosti (skříni, polici). Kolik nejméně místností (skříní, polic) je nezbytně třeba k uskladnění všech n látek?

Řešení lze nalézt pomocí barevnosti: Sestrojme graf G, jehož vrcholy odpovídají zmíněným látkám a v němž jsou dva vrcholy spojeny hranou, jestliže odpovídající látky nesmí být skladovány společně. Přípustné umístění látek do místností pak odpovídá obarvení grafu G. Roli barev zde hrají místnosti. Potřebujeme tedy nejméně $\chi(G)$ místností.

Podívejme se nyní na jinou úlohu: Máme za úkol přepravit najednou, např. v jednom autě, co největší počet vzorků těchto látek. Přepravované vzorky musí

tvořit nezávislou množinu v grafu G, můžeme tedy najednou přepravit nejvýše $\alpha(G)$ vzorků.

Pokud by každý ze vzorků měl danou cenu, mohli bychom požadovat přepravu množiny vzorků, která bude mít co největší cenu. To by odpovídalo nejdražší nezávislé množině v grafu G. Srovnejte tuto poslední variantu s problémem batohu 2.4.2, str. 46.

13.1.6 Příklad: Plánování procesů. Máme za úkol uskutečnit n procesů $p_1, p_2, \ldots, p_n$, přičemž některé dvojice procesů nesmějí (nebo nemohou) probíhat současně např. proto, že vyžadují použití stejných strojů. Kolik časových jednotek je třeba k uskutečnění všech procesů, trvá-li každý proces stejně dlouho (např. jednotku času)? Jaký největší počet procesů může probíhat najednou?

Sestrojme graf G, jehož vrcholy odpovídají procesům. Dva vrcholy spojíme hranou právě tehdy, když odpovídající procesy nesmějí probíhat současně. Každé obarvení vrcholů grafu G rozděluje procesy na skupiny, které mohou probíhat najednou (odpovídající vrcholy jsou obarveny stejnou barvou). Nejkratší možné provedení všech procesů tedy trvá $\chi(G)$. Procesy, které mohou probíhat najednou, vždy tvoří nezávislou množinu. Maximální možný počet současně probíhajících procesů tedy je roven $\alpha(G)$.

13.1.7 Cvičení. Na škole se vyučují předměty $p_1, p_2, \ldots, p_n$ ve studijních skupinách $s_1, s_2, \ldots, s_r$. Na škole učí učitelé $u_1, u_2, \ldots, u_m$. Předpokládejme, že pro každou studijní skupinu a předmět je předem pevně určen učitel. Učební plán je tedy dán jako seznam trojic (s_i, p_j, u_k), které vyjadřují, že ve studijní skupině s_i bude předmět p_j učit učitel u_k. Kolik nejméně vyučovacích hodin bude nezbytných pro splnění učebního plánu?

NÁVOD: Vyučovací hodinu vyjádřenou trojicí (s_i, p_j, u_k) pokládejte za proces (viz 13.1.6). Dva procesy nemohou probíhat současně, vyžadují-li stejného učitele nebo stejnou studijní skupinu.

13.1.8 Věta. V prostém grafu G bez smyček platí:

$$\chi(G) \leq \max\{d(x) | x \in V(G)\} + 1 \,.$$

DŮKAZ: Uvedeme jednoduchý algoritmus, který vrcholy grafu G obarví $h+1$ barvami, kde $h = \max\{d(x) | x \in V(G)\}$. V roli barev použijeme čísla $1, 2, \ldots, h+1$.

Seřaďme vrcholy do posloupnosti $x_1, x_2, \ldots, x_n$. Vrchol x_1 obarvíme barvou 1. Pak postupně vrcholům $x_2, x_3, \ldots, x_n$ přiřadíme barvy podle tohoto pravidla: vrchol x_i obarvíme nejnižší barvou, která se nevyskytuje mezi dosud obarvenými sousedy vrcholu x_i.

Poněvadž každý vrchol x má nejvýše h sousedů, musí mezi $h+1$ barvami vždy existovat alespoň jedna barva b, kterou není obarven žádný soused vrcholu x. Vrchol x tedy lze obarvit barvou b. Takto lze postupně obarvit všechny vrcholy grafu s použitím nejvýše $h+1$ barev. □

Poznámka: Právě uvedený algoritmus je rychlý a je vhodný pro první odhad barevnosti. Algoritmus 13.2.1 pro obarvení grafu nejmenším počtem barev vychází z obarvení získaného touto metodou. Viz také 13.2.7.

13.1.9 Věta. Pro graf G bez smyček o n vrcholech platí:

1. $\chi(G) \leq n$.
2. $\chi(G) \geq \omega(G)$.
3. $\chi(G)\,\alpha(G) \geq n$.
4. $\chi(G) + \alpha(G) \leq n + 1$.

Důkaz: První tvrzení je zřejmé. Druhé tvrzení vyplývá z faktu, že všechny vrcholy kliky je nutno obarvit různými barvami. K důkazu třetího tvrzení si stačí uvědomit, že vrcholy obarvené stejnou barvou tvoří nezávislou množinu. Obarvíme-li tedy graf $\chi(G)$ barvami, získáme $\chi(G)$ nezávislých množin, z nichž každá obsahuje nejvýše $\alpha(G)$ vrcholů. Odtud pak plyne nerovnost $\chi(G)\alpha(G) \geq n$.

K důkazu čtvrtého tvrzení obarvěme nejpočetnější nezávislou množinu první barvou. K obarvení zbývajících $n - \alpha(G)$ vrcholů zcela jistě stačí $n - \alpha(G)$ barev. Platí tedy $n - \alpha(G) + 1 \geq \chi(G)$, což je jen jiný zápis dokazované nerovnosti. □

13.1.10 Věta. Nechť G je prostý graf bez smyček. Označme n počet vrcholů, m počet hran a h maximální stupeň vrcholu v grafu G. Dále označme $\lfloor x \rfloor$ a $\lceil x \rceil$ dolní a horní celou část čísla x, tj. nejbližší celé číslo takové, že $\lfloor x \rfloor \leq x$ a $\lceil x \rceil \geq x$. Potom o nezávislosti $\alpha(G)$ a barevnosti $\chi(G)$ platí:

1. $\alpha(G) \geq (2n - m)/3$, přičemž rovnost platí pouze pro úplný graf K_3.
2. $\alpha(G) \geq n^2/(2m + n)$, přičemž rovnost platí pouze tehdy, když souvislé komponenty grafu G jsou úplné grafy téže velikosti.
3. $\alpha(G) \geq \lceil n/(h+1) \rceil$.
4. $\chi(G) + \chi(-G) \leq n + 1$.
5. $\chi(G) \geq n^2/(n^2 - 2m)$, přičemž rovnost nastává právě tehdy, je-li G úplným $\chi(G)$-partitním grafem se stejně velkými stranami.
6. Vrcholy grafu G lze rozložit na $h+1$ disjunktních nezávislých množin, z nichž každá má buď $\lfloor n/(h+1) \rfloor$, nebo $\lceil n/(h+1) \rceil$ vrcholů.

Důkaz lze najít v knize [Berge73]. □

13.1.11 Věta. Pro libovolný graf G bez smyček platí:

1. $\chi(G) = 1$ právě tehdy, když graf G je diskrétní.
2. $\chi(G) \leq 2$ právě tehdy, když graf G neobsahuje kružnici liché délky.
3. $\chi(G) \leq 2$ právě tehdy, když graf G je bipartitní.

Důkaz: První tvrzení je zřejmé. Dokážeme druhé tvrzení. K obarvení kružnice liché délky potřebujeme nejméně 3 barvy. Je-li tedy $\chi(G) \leq 2$, nemůže graf G takovou kružnici obsahovat. Naopak, nechť graf G neobsahuje kružnici liché

délky. Popíšeme, jak lze graf obarvit nejvýše dvěma barvami. (Je-li graf nesouvislý, obarvíme takto každou komponentu zvlášť.)

Zvolme pevně vrchol x. Vrcholy, které mají od vrcholu x lichou vzdálenost (ve smyslu počtu hran v nejkratší cestě), obarvíme barvou 1, vrcholy, které mají sudou vzdálenost (včetně vrcholu x) obarvíme barvou 2. Musíme dokázat, že takto získáme skutečně obarvení, tj. že žádné dva stejně obarvené vrcholy nejsou spojeny hranou.

Vezměme tedy dva libovolné vrcholy u, v, které mají oba lichou (oba sudou) vzdálenost od x. Mezi vrcholy u, v vede cesta sudé délky, která je tvořena částmi nejkratších cest z vrcholu u do x a z vrcholu x do v. Kdyby byly vrcholy u, v spojeny hranou, tvořila by tato hrana spolu s cestou z u do v kružnici liché délky. Sestrojili jsme tedy skutečně obarvení dvěma barvami.

Třetí tvrzení je snadné. Každé obarvení dvěma barvami určuje rozklad množiny $V(G)$ na množiny A, B takové, že $W(A) = W(B) = E(G)$, graf je tedy bipartitní. Naopak, je-li graf bipartitní, stačí obarvit každou z jeho stran jednou barvou. □

13.1.12 Cvičení. Jakou barevnost má strom?

13.1.13 Věta. Rovinný graf bez smyček lze obarvit čtyřmi barvami.

Poznámka: Tato věta je řešením kdysi slavného tzv. *problému čtyř barev*: Stačí čtyři barvy k obarvení libovolné politické mapy tak, aby žádné dva sousední státy nebyly vybarveny stejnou barvou?

O řešení tohoto problému se marně pokoušely řady matematiků i laiků přes 100 let. Od roku 1890 byl znám důkaz, že stačí pět barev. Důkaz pro čtyři barvy však byl nesmírně komplikovaný (pomocí počítače bylo nalezeno a vyšetřeno asi 1900 dílčích případů) a byl nalezen až v roce 1976.

Důsledek: Rovinný graf G bez smyček o n vrcholech má nezávislost $\alpha(G) \geq n/4$.

13.1.14 Věta. Pro každá dvě celá kladná čísla k a r existuje graf G, který má barevnost $\chi(G) = k$ a neobsahuje kružnici délky menší nebo rovné r.

Poznámka: Tato věta vlastně říká, že existují grafy s vysokou barevností, které jsou „lokálně řídké“. Speciálním případem této věty je tvrzení, že grafy, které neobsahují trojúhelníky, mohou mít libovolně vysokou barevnost. Důkaz tohoto speciálního případu lze najít v knize [Nešetřil79].

13.1.15 Hranová barevnost. V některých aplikacích je vhodnější nebarvit vrcholy, ale hrany. U obarvení hran pak požadujeme, aby každé dvě hrany, které mají společný vrchol, byly obarveny různými barvami. *Hranová barevnost* (též *chromatický index*) je nejmenší počet barev, které stačí k obarvení hran. Hranovou barevnost budeme značit $\chi'(G)$.

Vyšetřování hranové barevnosti grafu G lze převést na vyšetřování (vrcholové) barevnosti v tzv. hranovém grafu: *Hranový graf* ∂G získáme takto: Za vrcholy grafu ∂G prohlásíme hrany původního grafu G. Dva vrcholy v hranovém grafu ∂G jsou spojeny hranou, jestliže jim odpovídající hrany původního grafu G měly společný vrchol.

Obarvení hran grafu G a obarvení vrcholů hranového grafu ∂G si vzájemně jednoznačně odpovídají, platí tedy $\chi'(G) = \chi(\partial G)$. Navíc však platí:

13.1.16 Věta. (Vizing 1964.) Nechť G je prostý neorientovaný graf bez smyček a označme d maximální stupeň vrcholu. Pak hranová barevnost je rovna buď d, nebo $d + 1$.

Poznámka: Navzdory této větě je přesné určení hranové barevnosti (přesněji, rozhodnutí, zda hranová barevnost je rovna d) v obecném případě NP-úplnou úlohou.

13.2 Zjišťování barevnosti

Věta 13.1.11 (a její důkaz) spolu s 3.2.7, str. 54, dává dobrý návod ke snadnému ověření, zda daný graf je jedno- či dvoubarevný. Pro tři a více barev již žádný tak efektivní postup není znám. Naopak, rozhodnutí, zda graf je vrcholově r-barevný pro $r \geq 3$, je NP-úplnou úlohou.

Dále uvedený algoritmus pro barvení grafu nejmenším možným počtem barev je klasickým příkladem backtrackingu (viz 11.2, str. 191).

13.2.1 Algoritmus pro zjišťování barevnosti.

Vstup: graf G, jehož vrcholy jsou očíslovány čísly $1, 2, \ldots, n$. Pro každý vrchol x je dána množina $P(x) = V(x) \cap \{1, 2, \ldots, x - 1\}$.
Pomocné proměnné: Právě zkoumané částečné obarvení budeme uchovávat v hodnotách $B(x)$. Hodnoty Barva(x) budou obsahovat nejlepší dosud nalezené obarvení. Proměnná omez bude obsahovat číslo barvy, kterou již nechceme použít ve snaze získat obarvení méně než omez barvami. S výjimkou počáteční fáze bude proměnná omez rovna počtu barev v nejlepším dosud nalezeném obarvení Barva.
Výstup: výše popsané proměnné Barva a omez.
Metoda: Vzhledem k pevnému očíslování vrcholů můžeme každé obarvení pokládat za posloupnost barev. Tyto posloupnosti budeme systematicky prodlužovat a zkracovat v duchu backtrackingu.

Při postupu vpřed, tj. při prodlužování částečné posloupnosti barev, dáme následujícímu vrcholu x jako výchozí vždy nejnižší barvu, která (momentálně) není použita u vrcholů z množiny $P(x)$. Nižší barvu nelze použít (při stávajících barvách předchozích vrcholů), vyšší barvy budou zkoušeny v dalším průběhu backtrackingu (po provedení návratu).

Prvé úplné obarvení tedy bude stejné jako v důkaze věty 13.1.8.

Návraty v backtrackingu budeme provádět jednak při dosažení úplného obarvení, jednak v situaci, kdy některý vrchol y dostal barvu omez nebo vyšší. Přidělením této barvy jsme totiž dostali částečné obarvení, jehož prodlužování již nemá smysl, neboť nemůže vylepšit nejlepší dosud známé obarvení Barva.

Cílem návratu je snížit barvu ve vrcholu y. Poněvadž však nižší barvy v y již jsou vyzkoušeny, je třeba změnit (a tedy zvýšit) barvu v některém předchozím vrcholu

$x < y$. Aby to však mělo vliv na vrchol y, musí být $x \in P(y)$, proto provedeme prvý návrat až do vrcholu $x = \max(P(y))$ a v tomto vrcholu se pokusíme zvýšit barvu.

Nelze-li barvu ve vrcholu x zvýšit (byla by použita barva OMEZ), provedeme další, tentokrát již prostý návrat do vrcholu $x - 1$, $x - 2$ atd.

Barvu vrcholu 1 nezvyšujeme, nemá to smysl. (Proč? Zvažte záměnu barev.) Z podobného důvodu nemá smysl barvit vrchol x barvou $b > x$.

Na počátku volíme OMEZ $:= n+1$, aby hodnota OMEZ nebránila nalezení prvého obarvení.

ALGORITMUS:

1. [*Inicializace.*] $x := 1$; $B(x) := 1$; OMEZ $:= n + 1$.

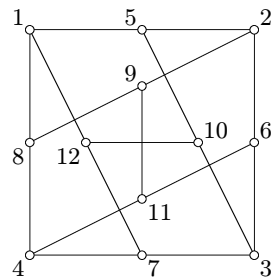
2. [*Postup vpřed – obarvení dalšího vrcholu.*] Položíme $x := x+1$. Jestliže $x > n$, pokračujeme podle kroku 3. V opačném případě položíme $B(x) :=$ minimum ze všech barev nevyskytujících se v množině $P(x)$. Jestliže $B(x) \geq$ OMEZ, položíme $y := x$ a pokračujeme podle kroku 4, jinak zopakujeme krok 2.

3. [*Všechny vrcholy obarveny, počet barev je menší než* OMEZ.] Pro všechny vrcholy x grafu G provedeme BARVA$(x) := B(x)$ a dále položíme OMEZ $:=$ $:=$ maximum z hodnot BARVA. Označme y první vrchol, který má barvu OMEZ, a pokračujme krokem 4.

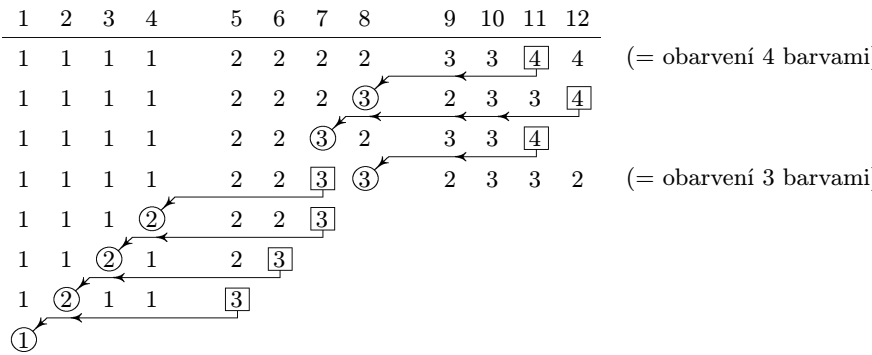
4. [*Návrat – pokus o snížení barvy ve vrcholu* y.] Označme x poslední vrchol z množiny $P(y)$ a pokračujme krokem 5.

5. [*Pokračování návratu – pokus o zvýšení barvy ve vrcholu* x.] Jestliže $x = 1$, výpočet končí. V opačném případě položíme $b :=$ minimum ze všech barev, které jsou větší než $B(x)$ a nevyskytují se v množině $P(x)$. Pokud $b <$ OMEZ a zároveň $b \leq x$, položíme $B(x) := b$ a pokračujeme podle kroku 2, v opačném případě položíme $x := x - 1$ a zopakujeme krok 5.

13.2.2 Příklad. Činnost algoritmu 13.2.1 předvedeme na grafu z obrázku 13.2. Postup výpočtu je patrný z následující tabulky, kde každý řádek zachycuje hodnoty B po vykonání kroku 4.

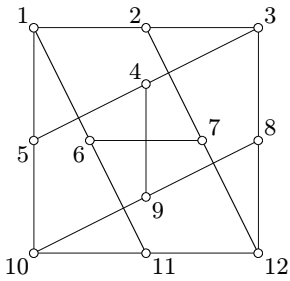


Obrázek 13.2: Graf k příkladu 13.2.2.



Čtverečkem je zde vždy vyznačen vrchol y, v němž chceme snížit barvu. Návraty v backtrackingu jsou vyznačeny šipkami a prvý vrchol, v němž se po návratu zvýšila barva, je v kroužku.

Dodejme, že v tomto příkladě jsme zvolili očíslování vrcholů úmyslně trochu nešikovně, abychom na delším výpočtu mohli demonstrovat jeho postup. Kdybychom očíslovali vrcholy podle obrázku 13.3, proběhl by výpočet velmi rychle. Ověřte!



Obrázek 13.3: Jiné očíslování vrcholů k příkladu 13.2.2.

13.2.3 Poznámka. Rychlost algoritmu 13.2.1 velice záleží na očíslování vrcholů. Doporučuje se nejnižšími čísly očíslovat co největší kliku a vrcholy s velkým stupněm číslovat nižšími čísly. Také je vhodné, aby sousední vrcholy byly číslovány pokud možno sousedními čísly.

13.2.4 Cvičení. Vezměte kořenový strom, který má jeden list v hloubce 2 a ostatní listy v hloubce 1. Určitě víte, že barevnost každého stromu je rovna 2. Přesto tento strom obarvěte algoritmem 13.2.1. Při kterém uspořádání (očíslování) vrcholů proběhne výpočet nejrychleji a při kterém nejpomaleji?

13.2.5 Test k-barevnosti. Potřebujeme-li pouze pro dané k zjistit, zda daný graf lze obarvit k barvami, lze algoritmus 13.2.1 zrychlit: při inicializaci položíme OMEZ $:= k+1$ a v kroku 3, kde je nalezeno obarvení k barvami, můžeme výpočet ukončit. Pokud výpočet skončí v kroku 5, pak obarvení k barvami neexistuje.

13.2.6 Cvičení. Snažíme-li se jednoduchý backtracking urychlit podobně jako v algoritmu 13.2.1, je třeba pečlivě ověřit (dokázat), že urychlením nevynecháme optimální řešení.

Zjistěte, co je špatně na následujícím „vylepšení“ algoritmu 13.2.1, a najděte příklad grafu, kde takto „vylepšený“ postup selže:

> Snažíme-li se snížit barvu vrcholu y a nelze-li barvu zvýšit u vrcholu $x = \max(P(y))$, pak neprovedeme další návrat do vrcholu $x - 1$, ale rovnou až do druhého nejvyššího vrcholu z množiny $P(y)$, neboť pouze vrcholy z množiny $P(y)$ ovlivňují barvu vrcholu y.

13.2.7 Heuristické postupy. Algoritmus uvedený v důkaze věty 13.1.8 lze použít pro hrubý odhad barevnosti. Jeho výhodu je fakt, že při vhodném uspořádání vrcholů máme šanci získat přímo optimální obarvení. Doporučuje se volit pořadí vrcholů tak, aby vrcholy s velkým stupněm byly použity dříve. Další možností je opakovat výpočet s náhodně změněným pořadím vrcholů. Při dostatečném počtu opakování pak poměrně rychle roste i pravděpodobnost, že se strefíme do optimálního obarvení.

13.3 Hledání maximálních nezávislých množin

Uvedeme relativně rychlý algoritmus pro vyhledání všech maximálních nezávislých množin založený na backtrackingu. Po jednoduchých úpravách jej lze použít k hledání nejpočetnější nebo nejdražší nezávislé množiny.

13.3.1 Algoritmus pro hledání maximálních nezávislých množin.

Vstup: Graf G zadaný pomocí seznamu vrcholů a seznamů sousedů $V(x)$.
Výstup: Posloupnost seznamů maximálních nezávislých množin.
Pomocné proměnné: V proměnné R budeme udržovat posloupnost vrcholů $r_1, r_2, \ldots, r_k$, přičemž její prvky budou tvořit k-prvkovou nezávislou množinu. Pro $i \leq k$ označme $R_i = \{r_1, r_2, \ldots, r_i\}$.

Dále budeme udržovat dva systémy množin vrcholů N_i a S_i pro všechna $i = 0, 1, \ldots, k$, přičemž bude platit:

1. Množiny S_i, N_i a R_i jsou po dvou disjunktní.
2. $(S_i \cup N_i)$ = množina všech vrcholů v takových, že $R_i \cup \{v\}$ je nezávislá množina, tj. vrchol v by bylo možno přidat k R_i.
3. N_i = množina všech vrcholů, které dosud k množině R_i nebyly přidány, tj. žádná nezávislá množina obsahující $R_i \cup \{v\}$ dosud nebyla zkoumána.
4. S_i = množina všech vrcholů v, jejichž přidání k množině R_i již bylo vyzkoušeno, tj. všechny maximální nezávislé množiny obsahující $R_i \cup \{v\}$ již byly prozkoumány.

METODA: Backtracking spočívá v systematickém prodlužování a zkracování posloupnosti R a v odpovídající údržbě množin S_i, N_i, R_i.

Máme-li posloupnost R o délce k, pak k jejímu prodloužení vybereme nějaký vrchol $x \in N_k$. Množiny N_k a S_k uschováme, budeme je potřebovat později, až se posloupnost R opět zkrátí na délku k. Nyní však posloupnost R prodloužíme o vrchol x tím, že položíme $r_{k+1} := x$, vytvoříme množiny N_{k+1} a S_{k+1} tak, aby splňovaly výše uvedené podmínky, a položíme $k := k + 1$.

Z podmínek 1–4 vyplývá, že posloupnost R obsahuje maximální nezávislou množinu právě tehdy, když $N_k = S_k = \emptyset$.

Nelze-li posloupnost R prodloužit, tj. je-li $N_k = \emptyset$, provedeme v backtrackingu návrat, tj. posloupnost zkrátíme o její poslední prvek r_k, položíme $k := k - 1$ a upravíme množiny N_k a S_k.

Návrat provedeme také tehdy, existuje-li vrchol $y \in S_k$ takový, že $V(y) \cap N_k = \emptyset$. V této situaci je totiž jakékoli další prodlužování posloupnosti R zbytečné, protože každé nezávislé množině, která by R obsahovala, by k maximálnosti scházel vrchol y.

ALGORITMUS:

1. [*Inicializace.*] Položíme $k := 0$, $S_0 := \emptyset$ a $N_0 := V$. Pokračujeme krokem 2.

2. [*Rozšíření nezávislé množiny.*] Vybereme libovolný vrchol $x \in N_k$ a položíme $r_{k+1} := x$, $S_{k+1} := S_k \setminus V(x)$, $N_{k+1} := N_k \setminus (V(x) \cup \{x\})$ a $k := k + 1$. Pokračujeme krokem 3.

3. [*Test maximálnosti množiny R_k.*] Jestliže $N_k = S_k = \emptyset$, vytiskneme posloupnost $R = (r_1, r_2, \ldots, r_k)$ a pokračujeme podle kroku 6, jinak pokračujeme krokem 4.

4. [*Test možnosti zvětšování.*] Jestliže $N_k = \emptyset$, pokračujeme podle kroku 6, jinak pokračujeme krokem 5.

5. [*Test, lze-li R_k doplnit na maximální nezávislou množinu.*] Jestliže existuje vrchol $y \in S_k$ takový, že $V(y) \cap N_k = \emptyset$, pokračujeme podle kroku 6, jinak pokračujeme krokem 2.

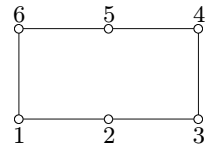
6. [*Návrat, zkrácení posloupnosti R.*] Jestliže $k = 0$, výpočet končí. V ostatních případech položíme $x := r_k$, $k := k - 1$, $N_k := N_k \setminus \{x\}$ a $S_k := S_k \cup \{x\}$. Pokračujeme krokem 4.

13.3.2 Poznámky. Při výpočtu na počítači lze systém množin S_i, N_i a R_i uchovávat v jediné matici o n sloupcích, kde n je počet vrcholů. Pro $i \geq 0$ mohou být v i-tém řádku zakódovány všechny tři zmíněné množiny, neboť jsou disjunktní. Posloupnost R je vhodné udržovat zvlášť. Víme-li, že nezávislost grafu $\alpha(G) < K$, pak stačí, aby matice měla pouze K řádků.

Jinou možností je reprezentovat množiny S_i, N_i jako pole bitů a využít toho při provádění množinových operací.

Rychlost práce tohoto algoritmu lze podstatně ovlivnit strategií pro volbu vrcholu x v kroku 2.

13.3.3 Příklad. Pomocí algoritmu 13.3.1 vyhledejme všechny maximální nezávislé množiny v grafu z obr. 13.4.



Obrázek 13.4: Graf k příkladu 13.3.3.

Průběh práce algoritmu je patrný z následující tabulky. Obsah množin N_k, S_k a R_k je vyznačen písmeny N, S a R. Znak − znamená, že příslušný vrchol nepatří do žádné z těchto množin. Situace, kdy R_k byla maximální nezávislá množina, jsou označeny hvězdičkou (∗). Znak ↙ upozorňuje, že byl proveden návrat vyvolaný testem v kroku 5.

k	R_k	N_0, S_0, R_0	N_1, S_1, R_1	N_2, S_2, R_2	N_3, S_3, R_3
		1 2 3 4 5 6	1 2 3 4 5 6	1 2 3 4 5 6	1 2 3 4 5 6
0	∅	N N N N N N			
1	1		R − N N N −		
2	13			R − R − N −	
3	135∗				R − R − R − ∗
2	13			R − R − S −	
1	1		R − S N N −		
2	14∗			R − − R − − ∗	
1	1		R − S S N − ↙		
0	∅	S N N N N N			
1	2		− R − N N N		
2	24			− R − R − N	
3	246∗				− R − R − R ∗
2	24			− R − R − S	
1	2		− R − S N N		
2	25∗			− R − − R − ∗	
1	2		− R − S S N ↙		
0	∅	S S N N N N			
1	3		S − R − N N		
2	35			S − R − R −	
1	3		S − R − S N		
2	36∗			− − R − − R ∗	
1	3		S − R − S S		
0	∅	S S S N N N ↙			

13.3.4 Heuristické postupy. Pro hledání nejpočetnějších nebo nejdražších nezávislých množin se používají také heuristické algoritmy, které bývají rychlejší, ale „bez záruky“. Nejjednodušší je zvolit některý vrchol a k němu přidávat další tak dlouho, dokud nedostaneme maximální nezávislou množinu. Pravidlo pro volbu přidávaného vrcholu má zásadní vliv na kvalitu výsledného řešení. Volba může být i náhodná, ale doporučuje se preferovat vrcholy s malým stupněm nebo s velkou ce-

nou. Výpočet je vhodné několikrát opakovat s využitím náhody a nakonec si vybrat nejlepší z dosažených řešení.

Také lze použít metodu lokálního průzkumu (viz 11.4, str. 198) k postupnému vylepšování dosaženého řešení např. tak, že jeden vrchol z nezávislé množiny odstraníme a pokoušíme se místo něj zařadit jeden nebo i několik vrcholů jiných.

Kapitola 14

Rovinné grafy

Rovinné grafy, tj. grafy, které lze nakreslit v rovině bez křížení hran, jsou důležité zejména v počítačové grafice, kde se využívá faktu, že struktura vrcholů a hran třírozměrných hranatých těles odpovídá rovinným grafům, a také v elektrotechnice při návrhu integrovaných obvodů a tištěných spojů. Další význam rovinných grafů vyplývá z jejich speciálních vlastností. Pro rovinné grafy lze například zjednodušit některé algoritmy (maximální tok v síti, prohledávání bludiště).

14.1 Nakreslení grafu a topologický rovinný graf

14.1.1 Nakreslení grafu. Formální definice říká, že *nakreslení grafu* G v rovině ρ je dvojice prostých zobrazení ϕ, ψ takových, že ϕ přiřazuje vrcholům grafu různé body roviny ρ a zobrazení ψ přiřazuje každé hraně spojující např. vrcholy $\{x, y\}$ jednoduchou křivku v rovině ρ s krajními body $\phi(x)$ a $\phi(y)$. Přitom požadujeme, aby žádná křivka, která je obrazem hrany, neobsahovala obraz žádného vrcholu jako svůj vnitřní bod.

Jednoduché křivky mohou mít rozličný tvar a mohou se vzájemně protínat, přičemž je nutno rozlišovat obrazy vrcholů od průsečíků křivek. Proto při praktickém kreslení znázorňujeme vrcholy pomocí kroužků, puntíků a podobně, abychom je odlišili od průsečíků.

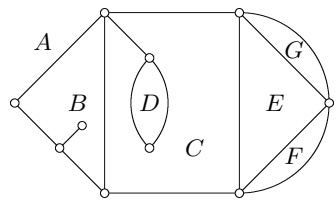
Rovinné nakreslení je takové nakreslení, v němž libovolné dvě křivky, přiřazené různým hranám, mají společné nejvýše dva – a to své krajní – body. Graf, ke kterému existuje rovinné nakreslení, nazýváme *rovinným grafem* nebo také grafem *planárním*.

14.1.2 Topologický rovinný graf je rovinný graf chápaný společně se svým nakreslením. Formálně tedy topologický rovinný graf je šestice $(V, E, \varepsilon, \rho, \phi, \psi)$, v níž (V, E, ε) je graf a (ϕ, ψ) je jeho nakreslení v rovině ρ. Rovinný graf (bez přívlastku topologický) je formálně pouze graf (tj. trojice (V, E, ε)), pro který existuje nějaké rovinné nakreslení.

Vlastnosti rovinného grafu se netýkají konkrétního nakreslení, ale pouze jeho existence nebo dokonce jen možnosti nějaké rovinné nakreslení vytvořit. Vlastnosti topologického rovinného grafu zahrnují navíc všechny vlastnosti jeho konkrétního nakreslení.

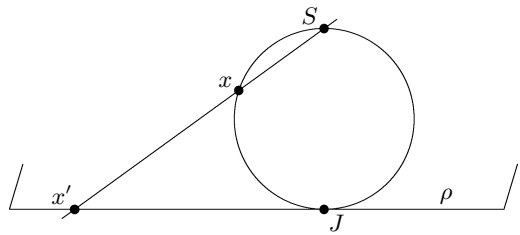
Stěna topologického rovinného grafu je část roviny ohraničená křivkami, které jsou obrazy hran. Jedna ze stěn je vždy neomezená, ostatní jsou omezené. Dvě stěny nazýváme *sousedními*, jestliže společná část jejich hranice je tvořena jednou nebo několika křivkami, které jsou obrazy hran.

Řekneme, že *stěna je incidentní s hranou*, jestliže křivka, která je obrazem hrany, tvoří část hranice (nebo celou hranici) stěny. *Stupeň stěny* definujeme jako počet hran, s nimiž je stěna incidentní, přičemž každou hranu, která je mostem, počítáme dvakrát. Například v grafu na obr. 14.1 má stěna B stupeň 6 a stěna C stupeň 8. Dvě stěny jsou sousední, jsou-li obě incidentní s touž hranou.



Obrázek 14.1: Stěny topologického rovinného grafu.

14.1.3 Kreslení grafu na kouli, stereografická projekce. Definice nakreslení grafu uvedená v 14.1.1 může být snadno zobecněna i na jiné plochy než roviny. Kreslení grafu na povrch koule má těsný vztah ke kreslení v rovině: graf lze nakreslit bez křížení hran na povrch koule právě tehdy, když je rovinný.



Obrázek 14.2: Stereografická projekce.

Nakreslení na kouli a nakreslení v rovině lze vzájemně převádět mnoha způsoby, jedním z nich je tzv. *stereografická projekce*: Na povrchu koule zvolíme dva souměrně sdružené body S a J („severní a jižní pól“) a zvolíme rovinu ρ, která se dotýká koule v bodě J (viz obr. 14.2). Nyní každému bodu $x \neq S$ z povrchu koule přiřadíme bod x' v rovině takto: body S a x vedeme přímku a ta protne rovinu ρ v bodě x'. Toto zobrazení je prosté a zobrazuje celý povrch koule kromě bodu S na celou rovinu, existuje tedy inverzní zobrazení. Obě tato zobrazení jsou spojitá, obrazem křivky je tedy opět křivka.

Převádíme-li nakreslení z koule na rovinu, musíme samozřejmě zvolit „severní pól“ S tak, aby neležel v obraze žádné hrany nebo vrcholu.

Je-li graf nakreslen na povrchu koule tak, že se jeho hrany (přesněji křivky odpovídající hranám) nekříží, můžeme i zde definovat pojmy stěna, incidence stěny s hranou, stupeň stěny a sousední stěna. Na povrchu koule jsou všechny stěny omezené, stěně obsahující „severní pól“ S odpovídá v rovinném nakreslení neomezená stěna.

Poznámka: Každý graf lze nakreslit na povrch nějakého tělesa tak, aby se obrazy hran nekřížily. Mnohdy k tomu však potřebujeme složité těleso s mnoha „dírami“ nebo „uchy“. Rovinné grafy jsou ty, které lze nakreslit bez křížení na tělesa bez „děr“ a bez „uch“, tj. na tělesa, která lze získat spojitou deformací z koule.

14.1.4 Silně ekvivalentní nakreslení. Je-li dáno rovinné nakreslení grafu G, pak je tím pro každý jeho vrchol v definováno *cyklické uspořádání hran* z množiny $E(v)$, a to tak, že budeme kolem vrcholu v obíhat ve směru hodinových ručiček. Dvě rovinná nakreslení nazýváme *silně ekvivalentní*, určují-li obě stejná cyklická uspořádání hran.

Cyklická uspořádání hran i silnou ekvivalenci lze definovat také pro grafy nakreslené na povrch koule. Lze dokázat, že jsou-li dána dvě silně ekvivalentní nakreslení, lze jedno z druhého získat spojitou transformací.

Pomocí cyklických uspořádání hran lze snadno a úsporně reprezentovat v počítači to, co je na rovinném nakreslení z grafového hlediska podstatné.

14.1.5 Věta. Nechť A je libovolná stěna topologického rovinného grafu G. Pak existuje silně ekvivalentní nakreslení grafu G takové, že stěna odpovídající A je neomezená.

Důkaz: Graf G přeneseme stereografickou projekcí na povrch koule, pootočíme jej tak, aby „severní pól“ S byl uvnitř požadované stěny a toto nakreslení promítneme zpět do roviny. □

Důkazy tří následujících vět lze najít v knize [Plesník83].

14.1.6 Věta. Ke každému rovinnému nakreslení prostého grafu bez smyček existuje silně ekvivalentní nakreslení takové, že každá hrana je reprezentována úsečkou.

14.1.7 Věta. Buď dán prostý topologický rovinný graf bez smyček s vrcholy $v_1, \ldots, v_n$. Dále nechť je v téže rovině dáno n různých bodů $p_1, \ldots, p_n$. Pak existuje silně ekvivalentní rovinné nakreslení takové, v němž každý vrchol v_i je nakreslen jako bod p_i.

14.1.8 Věta. Každá dvě rovinná nakreslení vrcholově 3-souvislého grafu buď jsou silně ekvivalentní, nebo se takovými stanou zrcadlovým převrácením jednoho z nich.

14.1.9 Mnohostěny. Vypuklé mnohostěny, tj. vypuklá[1] tělesa, jejichž stěny jsou mnohoúhelníky, přirozeným způsobem určují graf: vrcholy mnohostěnu pokládáme za vrcholy grafu a hrany mnohostěnu pokládáme za hrany grafu. Ostatně, odtud pochází naše grafová terminologie.

Je zřejmé, že je-li mnohostěn vypuklý, je jeho graf rovinný.

Lze dokázat, že graf je grafem nějakého vypuklého mnohostěnu právě tehdy, když je rovinný a vrcholově 3-souvislý.

Pravidelný mnohostěn je takový mnohostěn, jehož všechny stěny jsou shodné mnohoúhelníky a všechny vrcholy mají stejný stupeň. Graf pravidelného mnohostěnu je regulární (vrcholy mají stejný stupeň) a navíc také všechny jeho stěny mají stejný stupeň. Pomocí tzv. Eulerovy formule (viz 14.3.1) lze dokázat, že existuje (až na izomorfismus) pouze pět grafů s těmito vlastnostmi, existuje tedy pouze pět pravidelných mnohostěnů. Tyto mnohostěny znali již staří Řekové, říká se jim také *Platonova tělesa*. Grafově teoretický důkaz, že je pouze pět Platonových těles, lze najít v knize [MN96], zde uvedeme pouze číselné charakteristiky Platonových těles:

počet			stupeň	
stěn	vrcholů	hran	vrcholu	stěny
4	4	6	3	3
6	8	12	3	4
8	6	12	4	3
12	20	30	3	5
20	12	30	5	3

Pravidelný dvanáctistěn je nakreslen na obr. 12.1, str. 199. Pravidelný šestistěn asi znáte pod názvem krychle.

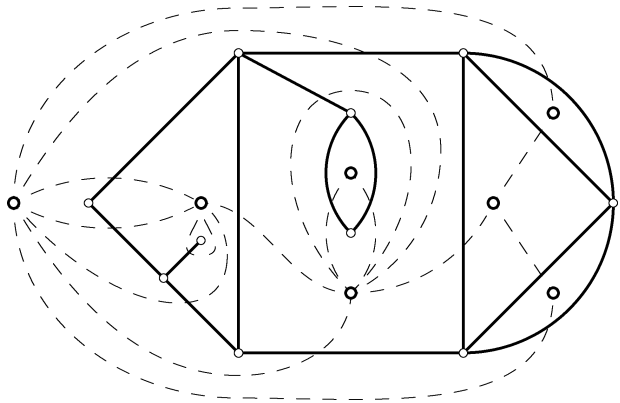
14.2 Duální grafy

14.2.1 Dualita neorientovaných rovinných grafů. Mějme souvislý neorientovaný topologický rovinný graf G. *Duální graf* G^D sestrojíme takto: Do každé stěny s grafu G umístíme jeden vrchol s^D grafu G^D. Pro každou hranu e grafu G sestrojíme hranu e^D, která bude spojovat ty vrcholy grafu G^D, které odpovídají stěnám grafu G incidentním s hranou e. Viz příklad na obr. 14.3. Platí:

1. Duální graf je také souvislý a rovinný.
2. Duální graf k duálnímu grafu je původní graf, tj. $(G^D)^D = G$.
3. Dualita převádí vrcholy grafu G na stěny grafu G^D.
4. Smyčce v grafu G odpovídá most v grafu G^D a naopak.
5. Kružnici v grafu G odpovídá v duálním grafu řez $W_{G^D}(A)$ určený množinou vrcholů A duálního grafu, přičemž vrcholy množiny A odpovídají stěnám grafu G ležícím uvnitř kružnice.

Ověřte si to alespoň nakreslením obrázku, lépe však úvahou.

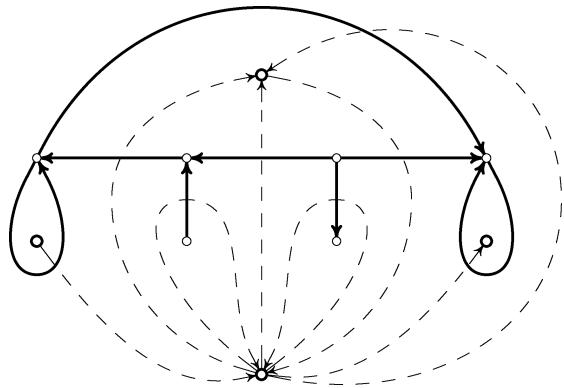
[1] Těleso je vypuklé (též *konvexní*), jestliže s každými dvěma body obsahuje i celou úsečku, která tyto dva body spojuje.



Obrázek 14.3: Duální graf ke grafu z obr. 14.1.

14.2.2 Cvičení. Nakreslete si rovinné grafy všech Platonových těles (viz 14.1.9) a sestrojte k nim grafy duální. Získáte tím opět grafy mnohostěnů – kterých?

14.2.3 Dualita orientovaných grafů. Duální graf k orientovanému souvislému topologickému grafu je definován stejně jako pro graf neorientovaný, navíc je však třeba určit orientaci hran. Orientaci hrany e^D volíme tak, aby protínala hranu e zleva doprava. Jinými slovy, orientaci hrany e^D získáme pootočením hrany e ve směru hodinových ručiček. Viz příklad na obr. 14.4.



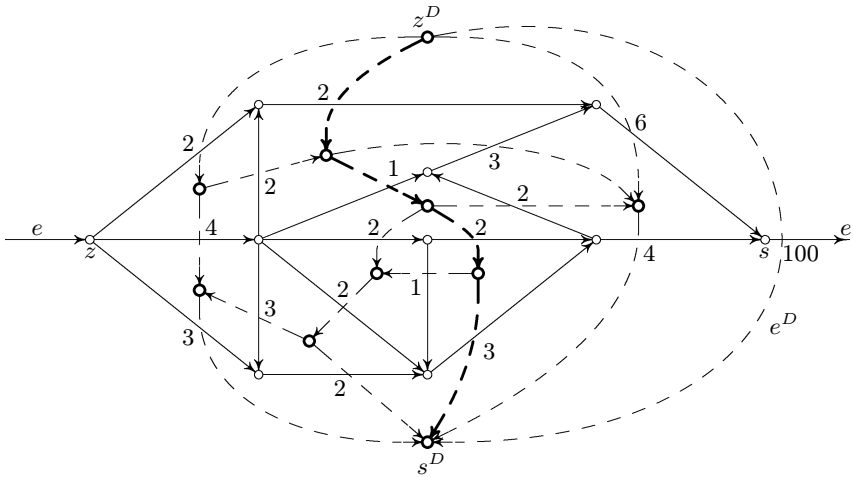
Obrázek 14.4: Duální graf k orientovanému grafu.

Tvrzení uvedená v 14.2.1 platí analogicky pro orientované grafy s výjimkou tvrzení 2: Duální graf k duálnímu grafu G^D není původní graf G, ale graf, který z něj dostaneme obrácením orientace všech hran.

Další vztahy mezi topologickým rovinným grafem a jeho duálním grafem jsou uvedeny v 15.3.7, str. 242.

14.2.4 Minimální řez a nejkratší cesta. Uvažujme orientovaný topologický rovinný graf G, jehož hrany jsou ohodnoceny kapacitami a jehož dva vrcholy, zdroj z a spotřebič s, leží na obvodě neomezené stěny. Hledáním maximálního toku v takovéto transportní síti jsme se zabývali v 8.3.15, str. 147.

Vytvořme topologický rovinný graf G' přidáním návratové hrany e ze spotřebiče do zdroje a její kapacitu zvolme tak, aby nemohla omezit tok. Ke grafu G' vytvořme duální graf. Označme z^D počáteční a s^D koncový vrchol hrany e^D v duálním grafu G^D (viz obr. 14.5). Hrany duálního grafu ohodnoťme délkami, které jsou číselně rovny kapacitám hran grafu G'.



Obrázek 14.5: Duální graf ke grafu z příkladu 8.3.16, str. 147. Nejkratší cesta odpovídá řezu s minimální kapacitou.

Každé neorientované cestě v duálním grafu z vrcholu z^D do vrcholu s^D odpovídá v grafu G řez, který odděluje zdroj z a spotřebič s. Kapacita řezu je přitom rovna délce cesty, ve které počítáme jenom hrany vpřed. Hrany, kterými cesta prochází proti jejich orientaci, totiž odpovídají hranám, které se v původním grafu díky své orientaci do kapacity řezu nepočítají.

V neorientované rovinné transportní síti je situace jednodušší. Maximální tok nasycuje vždy všechny hrany minimálního řezu, a proto je minimální kapacita řezu rovna délce nejkratší cesty v duálním grafu.

Je třeba upozornit, že v obou případech (orientovaném i neorientovaném) v grafu mohou existovat i řezy, které neodpovídají žádné cestě z vrcholu z^D do vrcholu s^D. Jsou to řezy $W(A)$ určené množinou vrcholů A takovou, že podgraf indukovaný touto množinou není souvislý. Snadno však lze nahlédnout, že žádný takový řez není minimální, neboť komponenta obsahující zdroj z určuje řez s menší kapacitou. Je-li řez určen množinou A takovou, že $z \in A$ a podgraf indukovaný množinou A je souvislý, pak hrany řezu $W(A)$ tvoří v duálním grafu cestu z vrcholu z^D do vrcholu s^D. Tato cesta vede po hranici souvislé oblasti tvořené těmi stěnami duálního grafu, které odpovídají množině A.

14.3 Vlastnosti rovinných grafů

14.3.1 Věta (Eulerova formule). Pro každý souvislý topologický rovinný graf, který má n vrcholů, m hran a s stěn, platí $n + s = m + 2$.

Důkaz: Graf G můžeme získat z jeho kostry postupným přidáváním hran, které v kostře neleží. Kostra topologického rovinného grafu má jedinou stěnu a počet hran je $m = n - 1$, proto pro kostru Eulerova formule platí. Přidáním hrany k topologickému rovinnému grafu rozdělíme vždy některou jeho stěnu na dvě. Levá i pravá strana Eulerovy formule se tedy přidáním hrany zvětší o jedničku. Po přidání všech hran tedy dostaneme Eulerovu formuli pro původní daný graf. □

14.3.2 Cvičení. Dokažte zobecnění Eulerovy formule pro nesouvislé grafy: Pro topologický rovinný graf s n vrcholy, m hranami, s stěnami a k komponentami souvislosti platí $n + s = m + k + 1$.

Následující tři věty zaručují, že prosté rovinné grafy bez smyček nemají „příliš mnoho“ hran, jsou tedy řídké.

14.3.3 Věta. Pro prostý souvislý rovinný neorientovaný graf bez smyček, který má $n \geq 3$ vrcholů a m hran, platí $m \leq 3n - 6$.

Důkaz: Poněvadž graf je prostý a bez smyček, má každá stěna (včetně stěny neomezené) stupeň alespoň 3. Součet stupňů všech stěn je roven $2m$, neboť v něm každou hranu počítáme dvakrát. Počet stěn s tedy splňuje $s \leq 2m/3$. Dosadíme-li to do Eulerovy formule, dostaneme $n + 2m/3 \geq m + 2$, a tedy $m \leq 3n - 6$. □

14.3.4 Věta. Pro prostý souvislý rovinný neorientovaný graf bez smyček a bez trojúhelníků, který má $n \geq 3$ vrcholů a m hran, platí $m \leq 2n - 4$.

Důkaz: Stupeň každé stěny grafu je větší nebo roven 4. Součet stupňů všech stěn je roven $2m$, počet stěn s tedy splňuje $s \leq 2m/4 = m/2$. Dosadíme-li to do Eulerovy formule, dostaneme $n + m/2 \geq m + 2$, a tedy $m \leq 2n - 4$. □

14.3.5 Věta. V každém prostém rovinném neorientovaném grafu bez smyček existuje vrchol, jehož stupeň je menší nebo roven 5.

Důkaz: Kdyby každý vrchol měl stupeň 6 nebo více, měl by celý graf alespoň $3n$ hran (je-li n počet vrcholů), což by bylo v rozporu s větou 14.3.3. □

14.3.6 Cvičení. Pomocí vět 14.3.3 a 14.3.4 dokažte, že grafy K_5 a $K_{3,3}$ nejsou rovinné.

14.3.7 Věta o čtyřech barvách. Rovinný graf bez smyček lze obarvit čtyřmi barvami.

Poznámka: Viz komentář k větě 13.1.13, str. 217.

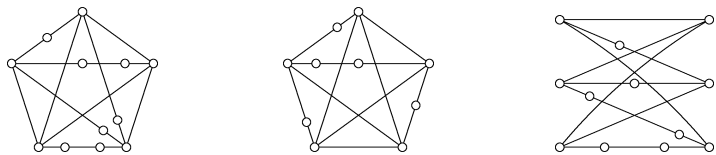
14.4 Charakterizace rovinných grafů

14.4.1 Homeomorfní grafy. Dva grafy G_1, G_2 nazýváme vzájemně *homeomorfními*, jestliže lze jeden získat z druhého těmito operacemi:

Dělení hrany. Mějme hranu e s krajními vrcholy x, y. Hranu e odstraníme a místo ní přidáme nový vrchol v a dvě nové hrany spojující nový vrchol v s vrcholy x a y.

Odstranění vrcholu stupně 2 je opačná operace: vrchol stupně 2 odstraníme spolu s oběma hranami a nahradíme hranou, která spojí bývalé sousedy odstraněného vrcholu.

Obě operace „zachovávají rovinnost“ v tom smyslu, že byl-li graf rovinný před operací, bude rovinný i po ní a naopak. Jsou-li tedy dva grafy homeomorfní, pak jsou buď oba rovinné, anebo oba nerovinné.



Obrázek 14.6: Grafy homeomorfní s grafy K_5 a $K_{3,3}$.

14.4.2 Kuratowského věta. Graf je rovinný právě tehdy, když neobsahuje podgraf homeomorfní s grafem K_5 nebo $K_{3,3}$.

Důkaz není jednoduchý, lze jej najít např. v [Berge58]. □

Poznámka: Pomocí Kuratowského věty lze např. o daném grafu dokázat, že není rovinný. Pro praktické zjišťování, zda daný graf je rovinný, to však není vhodný způsob, protože vyhledávání podgrafů je dost obtížné. Hlavní význam Kuratowského věty spočívá v tom, že rovinné grafy charakterizuje pomocí čistě grafových pojmů. Uvědomte si, že pojem rovinného nakreslení není čistě grafovým pojmem, potřebujeme k němu pojem roviny, spojitého zobrazení, křivky apod.

14.4.3 Cvičení. Pomocí Kuratowského věty dokažte, že tzv. *Petersenův graf* nakreslený vlevo na obr. 1.2, str. 18, není rovinným grafem.

14.4.4 Algoritmy pro testování rovinnosti. Kuratowského věta nedává dobrý návod k testování rovinnosti. Existují však rychlé algoritmy, které v čase $O(n)$ nejen rozhodnou, zda graf je rovinný, ale rovinné nakreslení zakódované pomocí cyklických uspořádání hran (viz 14.1.4) dokonce sestrojí. Vzhledem ke značné komplikovanosti se jimi nebudeme zabývat, jejich stručný popis lze najít např. v [Kučera83, Plesník83]. Poznamenejme pouze, že tyto algoritmy postupně přidávají kusy grafu ke vhodně zvolené kružnici a rozhodují, zda příslušná část grafu bude nakreslena uvnitř nebo vně kružnice.

Kapitola 15

Cirkulace a potenciály

Tato kapitola je věnována algebraickým vlastnostem cirkulací, potenciálových rozdílů, kružnic a řezů. Tyto vlastnosti jsou důležité zejména v elektrotechnice při analýze elektrických obvodů.

Ke studiu této kapitoly je potřebná alespoň základní znalost lineární algebry, zejména vektorových prostorů.

Poznamenejme, že následující výklad je možno dále zobecnit na vektorové prostory nad obecnými tělesy včetně těles konečných, takže pak lze mluvit např. o vektorovém prostoru kružnic v grafu apod. My se však omezíme na vektorové prostory nad tělesem reálných čísel.

V celé kapitole budeme předpokládat, že hrany grafu jsou pevně očíslovány čísly $1, 2, \ldots, m$.

15.1 Vektorový prostor cirkulací

Pojem cirkulace jsme definovali v 8.1.1, str. 131, některé vlastnosti cirkulací jsou uvedeny v 8.5, str. 151. Zde nahlédneme o něco hlouběji do algebraického zákulisí toků v síti. Na rozdíl od kapitoly 8 se zde nebudeme zabývat omezeními toku (kapacitami) a nebudeme rozlišovat toky přípustné a nepřípustné.

Díky pevnému očíslování hran čísly $1, 2, \ldots, m$ můžeme každou cirkulaci pokládat za m-členný aritmetický vektor.

15.1.1 Věta. Množina všech cirkulací v grafu G tvoří vektorový prostor, který je podprostorem prostoru $\mathbb{R}^m$, kde $m = |E(G)|$.

Důkaz vyplývá z faktu, že součet cirkulací i násobek cirkulace číslem je opět cirkulace (splňuje Kirchhoffův zákon). □

15.1.2 Kružnice a cirkulace. Připomeňme, že v 8.3.3, str. 142, jsme hrany libovolné kružnice rozdělili na hrany vpřed a hrany vzad, a to podle vztahu orientace hrany ke směru průchodu kružnicí.

Každé kružnici můžeme přiřadit tzv. *elementární cirkulaci* f předpisem:

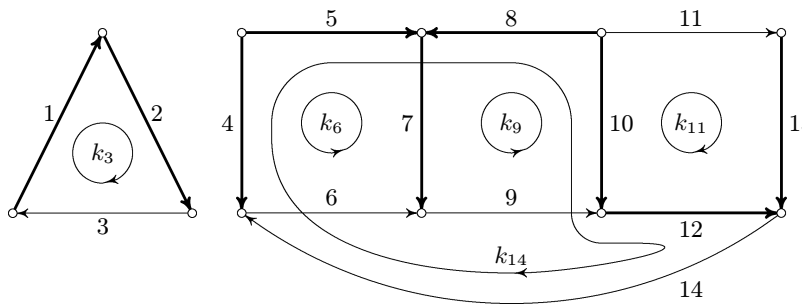
$$f(e) = \begin{cases} 0, & \text{jestliže } e \text{ neleží na kružnici,} \\ 1, & \text{je-li } e \text{ hranou vpřed,} \\ -1, & \text{je-li } e \text{ hranou vzad.} \end{cases}$$

15.1.3 Fundamentální soustava kružnic. Uvažujme graf G a nějaký jeho napnutý les K (viz 4.2.4, str. 60).

Ke každé hraně $e \in E(G) \setminus E(K)$ existuje podle věty 4.2.9 přesně jedna kružnice k_e tvořená hranou e a (některými) hranami lesa K. Tuto kružnici budeme orientovat tak, aby hrana e byla hranou vpřed. Každou takovou kružnici nazveme *fundamentální kružnicí* a množinu všech fundamentálních kružnic nazveme *fundamentální soustavou kružnic* vůči napnutému lesu K.

Každá fundamentální kružnice k_e určuje elementární cirkulaci f_e, která má v hraně e jednotkovou velikost. Takovou elementární cirkulaci nazýváme *fundamentální cirkulací*.

Fundamentální matice kružnic je matice, jejíž řádky jsou tvořeny všemi fundamentálními cirkulacemi chápanými jako řádkové vektory.



Obrázek 15.1: Fundamentální soustava kružnic.

15.1.4 Příklad. Graf na obr. 15.1 má dvě komponenty souvislosti. Napnutý les K je vyznačen silnými čarami. Tento les určuje (jednoznačně) fundamentální soustavu kružnic a fundamentální matici kružnic:

$$\begin{array}{c} \\ k_3 \\ k_6 \\ k_9 \\ k_{11} \\ k_{14} \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & e_{11} & e_{12} & e_{13} & e_{14} \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

Řádky této matice tvoří bázi vektorového prostoru cirkulací, což vyplývá z následující věty.

15.1.5 Věta. V grafu G zvolme libovolný napnutý les K. Potom množina všech fundamentálních cirkulací vzhledem k lesu K tvoří bázi vektorového prostoru cirkulací.

Důkaz: Je třeba dokázat, že fundamentální cirkulace jsou lineárně nezávislé a že generují celý vektorový prostor cirkulací.

Nezávislost vyplývá z faktu, že každá hrana $e \in E(G) \setminus E(K)$ je obsažena pouze v jediné fundamentální kružnici, totiž v k_e, a proto pouze jediná fundamentální cirkulace, totiž f_e, je na této hraně nenulová. Má-li tedy být nulová cirkulace (tj. nulový vektor) lineární kombinací fundamentálních cirkulací, musí být všechny koeficienty této lineární kombinace rovny nule. Tím je dokázána nezávislost.

Uvažujme nyní libovolnou cirkulaci f v grafu G. Ukážeme, že f je lineární kombinací fundamentálních cirkulací. Vezměme postupně všechny hrany $e \in E(G) \setminus \setminus E(K)$ a od cirkulace f odečtěme vždy $f(e)$-násobek fundamentální cirkulace f_e. Je zřejmé, že tím vynulujeme cirkulaci na všech hranách z množiny $E(G) \setminus E(k)$. Protože však zbývající hrany neobsahují žádnou kružnici (jde o hrany lesa K), musí být výsledná cirkulace nulová i na nich. Z této úvahy plyne, že pro libovolně zvolenou cirkulaci f platí

$$f = \sum_{e \in E(G) \setminus E(K)} f(e) \cdot f_e \ ,$$

a tedy f je lineární kombinací fundamentálních cirkulací. □

15.1.6 Věta. V grafu o n vrcholech, m hranách a k komponentách souvislosti je dimenze vektorového prostoru cirkulací rovna $m - n + k$.

Důkaz: Dimenze je rovna počtu prvků báze, tedy podle předchozí věty počtu fundamentálních cirkulací, tj. počtu všech hran mimo napnutý les. Podle věty 4.2.8 má každý napnutý les $n - k$ hran. Počet hran mimo napnutý les (a tedy i dimenze vektorového prostoru cirkulací) je tedy $m - n + k$. □

15.1.7 Cyklomatické číslo. Dimenzi vektorového prostoru cirkulací nazýváme *cyklomatickým číslem grafu.*

15.1.8 Poznámka. Vektorový prostor cirkulací má i jiné báze než fundamentální cirkulace určené nějakým napnutým lesem. V topologických rovinných grafech lze například získat bázi prostoru cirkulací ze soustavy kružnic určených omezenými stěnami. Ověřte na příkladě. Viz též následující cvičení.

15.1.9 Cvičení. Vezměte topologický rovinný graf z obrázku 1.1, str. 18 uprostřed, a zvolte jeho libovolnou orientaci. Ověřte, že čtyři elementární cirkulace určené kružnicemi kolem omezených stěn tvoří bázi prostoru cirkulací. Ověřte také, že tato báze není fundamentální bází vzhledem k žádnému napnutému lesu (tj. kostře) daného grafu.

15.2 Potenciály a potenciálové rozdíly

15.2.1 Potenciál a potenciálový rozdíl. Buď dán orientovaný graf G. *Potenciálem* nazveme jakékoli ohodnocení vrcholů $p : V(G) \to \mathbb{R}$ grafu G reálnými čísly.

Ohodnocení hran $\pi : E(G) \to \mathbb{R}$ reálnými čísly nazýváme *potenciálovým rozdílem*, jestliže existuje potenciál p takový, že pro každou hranu e grafu platí:

$$\pi(e) = p(\text{Kv}(e)) - p(\text{Pv}(e)) \,. \tag{15.1}$$

Všimněte si, že potenciálový rozdíl π se nezmění, zvýšíme-li potenciál p o stejnou konstantu ve všech vrcholech některé souvislé komponenty.

Zdaleka ne každé ohodnocení hran je potenciálovým rozdílem. Najděte příklad! Návod, jak ověřit, zda dané ohodnocení je potenciálovým rozdílem, podáme v následující větě 15.2.2, úspornější verzi pak ve větě 15.3.4.

Dále poznamenejme, že máme-li pevně zvoleno očíslování hran čísly $1, 2, \ldots, m$, pak každý potenciálový rozdíl můžeme pokládat za m-členný aritmetický vektor.

15.2.2 Věta. Pro libovolnou kružnici k označme C_k^+ množinu hran vpřed a C_k^- množinu hran vzad. Ohodnocení hran π je potenciálovým rozdílem právě tehdy, když pro každou kružnici k v grafu platí:

$$\sum_{e \in C_k^+} \pi(e) - \sum_{e \in C_k^-} \pi(e) = 0 \,. \tag{15.2}$$

Poznámka: Podmínka (15.2) je analogií druhého Kirchhoffova zákona (součet napětí podél obvodu je roven nule). Ve větě 15.3.4 ukážeme, že podmínku (15.2) stačí ověřit pro fundamentální soustavu kružnic.

Důkaz: Nechť π je potenciálový rozdíl a p je příslušný potenciál. Vezměme libovolnou kružnici, tj. posloupnost vrcholů a hran

$$v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k,$$

kde $v_k = v_0$. Pro každou její hranu platí:

$$\begin{aligned} p(v_i) - p(v_{i-1}) &= \pi(e_i), \quad \text{je-li } e_i \text{ hranou vpřed, a} \\ p(v_i) - p(v_{i-1}) &= -\pi(e_i), \quad \text{je-li } e_i \text{ hranou vzad.} \end{aligned}$$

Sečtením těchto rovnic pro všechny hrany kružnice dostaneme platnost podmínky (15.2):

$$\sum_{e \in C_k^+} \pi(e) - \sum_{e \in C_k^-} \pi(e) = \sum_{i=1}^{k} (p(v_i) - p(v_{i-1})) = 0 \,.$$

Nechť nyní naopak π je ohodnocení hran, které splňuje podmínku (15.2) pro každou kružnici k. Popíšeme postup, který vede k nalezení potenciálu p splňujícího podmínku (15.1).

V každé souvislé komponentě zvolíme libovolný vrchol v a ohodnotíme jej libovolným potenciálem $p(v)$. Dalším vrcholům v komponentách vypočteme jejich potenciály postupem, který připomíná značkovací proceduru: Existuje-li hrana e taková, že $\textsc{Pv}(e)$ je ohodnocen a $\textsc{Kv}(e)$ nikoli, položíme

$$p(\textsc{Kv}(e)) := p(\textsc{Pv}(e)) + \pi(e) \ .$$

Je-li naopak hrana e taková, že $\textsc{Kv}(e)$ je ohodnocen a $\textsc{Pv}(e)$ není, položíme

$$p(\textsc{Pv}(e)) := p(\textsc{Kv}(e)) - \pi(e) \ .$$

Platnost podmínky (15.2) pro všechny kružnice zaručuje, že každá hrana, jejíž oba vrcholy jsou již ohodnoceny, splňuje podmínku (15.1). □

15.2.3 Věta. Množina všech potenciálových rozdílů v grafu G tvoří vektorový prostor, který je podprostorem prostoru $\mathbb{R}^m$, kde m je počet hran.

Důkaz: Vezměme dva libovolné potenciálové rozdíly π_1, π_2. K nim existují potenciály p_1, p_2 takové, že pro každou hranu e platí:

$$\begin{aligned} \pi_1(e) &= p_1(\textsc{Kv}(e)) - p_1(\textsc{Pv}(e)) \ , \\ \pi_2(e) &= p_2(\textsc{Kv}(e)) - p_2(\textsc{Pv}(e)) \ . \end{aligned}$$

Součet potenciálů $p_1 + p_2$ je samozřejmě také potenciál, a poněvadž

$$\pi_1(e) + \pi_2(e) = p_1(\textsc{Kv}(e)) + p_2(\textsc{Kv}(e)) - p_1(\textsc{Pv}(e)) - p_2(\textsc{Pv}(e)) \ ,$$

je součet potenciálových rozdílů $\pi_1 + \pi_2$ také potenciálovým rozdílem. Podobně násobek potenciálu je potenciál, a tedy i násobek potenciálového rozdílu je potenciálový rozdíl. □

15.2.4 Řezy a potenciálové rozdíly. Každému řezu $W_G(A)$ v orientovaném grafu G můžeme přiřadit potenciál p definovaný předpisem

$$p(v) = \begin{cases} 0 & \text{pro } v \in A, \\ 1 & \text{pro } v \notin A. \end{cases}$$

Odpovídající potenciálový rozdíl π nazýváme *elementárním potenciálovým rozdílem*. Platí:

$$\pi(e) = \begin{cases} 1 & \text{pro } e \in W^+(A), \\ -1 & \text{pro } e \in W^-(A), \\ 0 & \text{pro } e \notin W(A). \end{cases}$$

Připomeňme, že zvláštním případem řezu je okolí $E(v)$ vrcholu v.

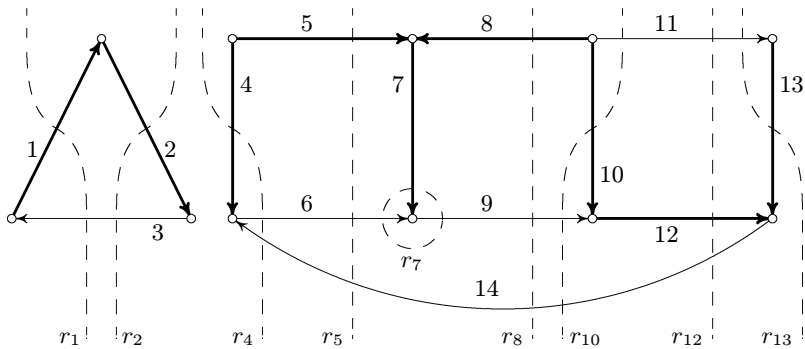
15.2.5 Fundamentální soustava řezů. Mějme graf G a jeho napnutý les K.

Odstraněním libovolné hrany $e \in E(K)$ se podle věty 4.2.9 rozpadne některá komponenta lesa K (tj. strom) na dvě komponenty. Označme A tu z nich, která obsahuje počáteční vrchol hrany e, druhou z nich označme B. *Fundamentální řez* r_e určený hranou e lesa K nyní definujeme jako řez $W_G(A)$.

Množinu všech fundamentálních řezů vůči napnutému lesu K nazýváme *fundamentální soustavou řezů.*

Fundamentální potenciálový rozdíl π_e určený hranou e napnutého lesa definujeme jako elementární potenciálový rozdíl, určený fundamentálním řezem r_e. Platí tedy $\pi_e(e) = 1$ a potenciál p_e, který tomuto potenciálovému rozdílu odpovídá, splňuje $p_e(A) = 0$ a $p_e(B) = 1$, kde A, B jsou komponenty lesa K oddělované hranou e.

Fundamentální matice řezů je matice, jejíž řádky jsou tvořeny všemi fundamentálními potenciálovými rozdíly, chápanými jako řádkové vektory.



Obrázek 15.2: Fundamentální soustava řezů.

15.2.6 Příklad. Fundamentální soustava řezů v grafu z obrázku 15.1 vzhledem k témuž napnutému lesu K je na obrázku 15.2 znázorněna čárkovaně. Fundamentální matice řezů má tvar

$$\begin{array}{c} \\ r_1 \\ r_2 \\ r_4 \\ r_5 \\ r_7 \\ r_8 \\ r_{10} \\ r_{12} \\ r_{13} \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & e_{11} & e_{12} & e_{13} & e_{14} \end{array} \\ \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Řádky této matice tvoří bázi vektorového prostoru potenciálových rozdílů, což vyplývá z následující věty.

15.2.7 Věta. V grafu G zvolme libovolný napnutý les K. Potom množina všech fundamentálních potenciálových rozdílů vzhledem k lesu K tvoří bázi vektorového prostoru potenciálových rozdílů.

Důkaz je podobný důkazu věty 15.1.5 pro cirkulace. Nezávislost fundamentálních potenciálových rozdílů vyplývá z faktu, že každá hrana $e \in E(K)$ je obsažena pouze v jediném fundamentálním řezu.

Vezměme nyní libovolný potenciálový rozdíl π. Ukážeme, že π je lineární kombinací fundamentálních potenciálových rozdílů. Uvažujme postupně všechny hrany $e \in E(K)$ a od potenciálového rozdílu π odečtěme vždy $\pi(e)$-násobek fundamentálního potenciálového rozdílu π_e. Je zřejmé, že tím získáme potenciálový rozdíl

$$\pi' = \pi - \sum_{e \in E(K)} \pi(e) \cdot \pi_e \ ,$$

který je nulový na všech hranách lesa K. Odpovídající potenciál tedy nabývá týchž hodnot vždy na celé souvislé komponentě grafu G, proto je potenciálový rozdíl π' nulový na všech hranách grafu (hrany vedou jen mezi vrcholy téže komponenty). Tím je dokázáno, že

$$\pi = \sum_{e \in E(K)} \pi(e) \cdot \pi_e$$

je lineární kombinací fundamentálních potenciálových rozdílů. □

15.2.8 Věta. V grafu o n vrcholech a k komponentách souvislosti je dimenze vektorového prostoru potenciálových rozdílů rovna $n - k$.

Důkaz: Dimenze je rovna počtu prvků báze, tedy podle předchozí věty počtu fundamentálních potenciálových rozdílů, tj. počtu hran napnutého lesa, který podle věty 4.2.8 má přesně $n - k$ hran. □

15.2.9 Poznámka. Vektorový prostor potenciálových rozdílů má i jiné báze než fundamentální soustavy určené nějakým napnutým lesem. Důležité jsou zejména báze tvořené z řezů tvaru $E(x)$, neboť příslušné elementární potenciálové rozdíly jsou shodné s řádky incidenční matice.

15.2.10 Věta. Vektorový prostor potenciálových rozdílů grafu G je generován množinou elementárních potenciálových rozdílů určených řezy tvaru $E(x)$, kde x je vrchol grafu G.

Důsledek: Vektorový prostor potenciálových rozdílů je generován řádky incidenční matice. Z řádků incidenční matice tedy lze vybrat bázi vektorového prostoru potenciálových rozdílů.

Důkaz: Dokážeme, že z elementárních potenciálových rozdílů určených okolími vrcholů, tj. řezy ve tvaru $E(x)$, lze získat jako lineární kombinaci libovolný elementární potenciálový rozdíl. Z toho pak již snadno plyne, že lze vygenerovat i jakoukoli fundamentální soustavu potenciálových rozdílů a ta již podle 15.2.7 generuje celý prostor.

Vezměme libovolný řez $W(A)$ a jím určený potenciálový rozdíl π. Označme π_x elementární potenciálový rozdíl určený okolím $E(x)$ libovolného vrcholu $x \in A$. Ukážeme, že pro všechny hrany grafu platí:

$$\pi(e) = \sum_{x \in A} \pi_x(e) \ . \tag{15.3}$$

Pro hrany z řezu $W(A)$ tato rovnost vyplývá z faktu, že hrana e je obsažena pouze v jedné z množin $E(x)$ a příslušný sčítanec má správné znaménko. Má-li hrana e oba vrcholy v množině A, je obsažena ve dvou množinách hran $E(\mathrm{P}v(e))$ a $E(\mathrm{K}v(e))$, přičemž na hraně e mají příslušné sčítance opačná znaménka, a součet je tedy nulový. Hrany, jejichž oba vrcholy leží mimo množinu A, neleží v žádném z uvažovaných okolí, a proto je na nich součet také nulový. Platí tedy:

$$\pi = \sum_{x \in A} \pi_x \ .$$

Potenciálový rozdíl π tedy je lineární kombinací elementárních potenciálových rozdílů, které jsou určeny okolími $E(x)$ vrcholů $x \in A$. □

15.2.11 Hodnost grafu. Dimenze vektorového prostoru potenciálových rozdílů je rovna hodnosti incidenční matice grafu. Tuto dimenzi nazýváme též *hodností grafu*.

15.3 Vztahy cirkulací a potenciálových rozdílů

15.3.1 Skalární součin a kolmost vektorů. Dva vektory $u, v \in \mathbb{R}^m$ jsou *kolmé*, je-li jejich skalární součin roven nule. Jsou-li P a P' dva podprostory téhož prostoru $\mathbb{R}^m$ a je-li každý vektor z P kolmý na každý vektor z P', pak prostory P a P' nazýváme vzájemně kolmými. Z vlastností skalárního součinu snadno plyne toto tvrzení:

Je-li každý vektor z množiny generátorů prostoru P kolmý na každý vektor z množiny generátorů prostoru P', pak prostory P a P' jsou vzájemně kolmé.

15.3.2 Věta. Vektorový prostor cirkulací a vektorový prostor potenciálových rozdílů jsou vzájemně kolmé.

Důkaz: Stačí dokázat, že libovolná fundamentální cirkulace je kolmá na libovolný fundamentální potenciálový rozdíl vůči témuž napnutému lesu. To je však snadné: Stačí si uvědomit, že libovolná fundamentální kružnice a libovolný fundamentální řez buď nemají žádnou společnou hranu, nebo mají přesně dvě společné hrany. Tyto dvě hrany buď jsou stejně orientovány vůči kružnici (obě vpřed nebo obě vzad) a rozdílně vůči řezu (jedna dovnitř druhá ven), nebo jsou rozdílně orientovány vzhledem ke kružnici a stejně vůči řezu. Ve všech těchto případech mají ve skalárním součinu činitelé odpovídající těmto hranám opačná znaménka a vzájemně se ruší. Skalární součin fundamentální cirkulace a fundamentálního potenciálového rozdílu je tedy nulový. □

15.3.3 Lemma. Mějme kružnici k a jí odpovídající elementární cirkulaci f. Potom ohodnocení hran π splňuje podmínku (15.2) (druhý Kirchhoffův zákon) právě tehdy, když vektory π a f jsou kolmé.

Důkaz vyplývá bezprostředně z definice elementární cirkulace a skalárního součinu. □

15.3.4 Věta. Buď G orientovaný graf a π ohodnocení jeho hran. Potom následující podmínky jsou ekvivalentní:

1. π je potenciálový rozdíl.
2. Podmínka (15.2) (druhý Kirchhoffův zákon) platí pro každou kružnici.
3. Podmínka (15.2) platí pro každou fundamentální kružnici.
4. Vektor π je kolmý na každou fundamentální cirkulaci vzhledem k nějakému napnutému lesu K.
5. Vektor π je kolmý na každou cirkulaci f v grafu G.

Důkaz: $1 \Leftrightarrow 2$ vyplývá z věty 15.2.2; $2 \Rightarrow 3$ platí triviálně; $3 \Rightarrow 4$ vyplývá z lemmatu 15.3.3; $4 \Rightarrow 5$ vyplývá z vlastností vektorových prostorů; $5 \Rightarrow 2$ vyplývá opět z lemmatu 15.3.3. □

15.3.5 Lemma. Mějme řez $W(A)$ a jemu odpovídající elementární potenciálový rozdíl π. Potom ohodnocení hran f splňuje podmínku

$$\sum_{e \in W^+(A)} f(e) - \sum_{e \in W^-(A)} f(e) = 0 \tag{15.4}$$

právě tehdy, když vektory f a π jsou kolmé.

Důkaz plyne bezprostředně z definice elementárního potenciálového rozdílu. □

15.3.6 Věta. Buď G orientovaný graf a f ohodnocení jeho hran. Potom následující podmínky jsou ekvivalentní:

1. f je cirkulace.
2. Podmínka (15.4) platí pro každý řez.
3. Podmínka (15.4) platí pro každý fundamentální řez.
4. Vektor f je kolmý na každý fundamentální potenciálový rozdíl vzhledem k nějakému napnutému lesu K.
5. Vektor f je kolmý na každý potenciálový rozdíl.

Důkaz: $1 \Leftrightarrow 2$ je zřejmé z definice cirkulace a řezu; $2 \Rightarrow 3$ platí triviálně; $3 \Rightarrow 4$ vyplývá z lemmatu 15.3.5; $4 \Rightarrow 5$ vyplývá z vlastností vektorových prostorů; $5 \Rightarrow 2$ opět vyplývá z lemmatu 15.3.5. □

15.3.7 Cirkulace a potenciálové rozdíly v rovinných grafech mají další zajímavé vlastnosti. Uvažujme orientovaný topologický rovinný graf G a k němu duální graf G^D (viz 14.2.3, str. 229). Každou cirkulaci v grafu G můžeme pokládat za potenciálový rozdíl v G^D, a také naopak každý potenciálový rozdíl v G můžeme pokládat za cirkulaci v G^D. (Ověřte na příkladě!) Z toho také plyne, že cyklomatické číslo grafu G je rovno hodnosti grafu G^D, a naopak hodnost grafu G je rovna cyklomatickému číslu grafu G^D.

15.4 Analýza elektrické sítě

V tomto oddíle naznačíme, jak lze vlastnosti vektorových prostorů cirkulací a potenciálových rozdílů použít při analýze elektrické sítě složené ze zdrojů stejnosměrného napětí a z rezistorů. Podrobnější výklad týkající se obecnějších obvodů by přesahoval rámec této knížky, zájemce odkazujeme na knihy [SwTh81, KŠCh89] a na speciální elektrotechnickou literaturu .

15.4.1 Kirchhoffovy zákony. Uvažujme elektrickou síť složenou z rezistorů a ze zdrojů napětí. Síť snadno znázorníme orientovaným grafem. Orientaci hran můžeme zvolit libovolně. Vrcholy grafu odpovídají uzlům elektrické sítě (vodičům), hrany odpovídají větvím. V každé větvi může být jeden nebo několik zdrojů napětí a jeden nebo několik rezistorů. Každou větev e ovšem můžeme zjednodušit tak, že obsahuje vždy jeden zdroj napětí ε_e (může mít i nulovou velikost) a jeden rezistor o odporu R_e: pro tento účel ve větvi sečteme napětí všech zdrojů a podobně sečteme odpory všech rezistorů a vnitřní odpory všech zdrojů.

Proud protékající větví e označíme I_e, napětí U_e na této větvi je součtem napětí zdroje ε_e a úbytku napětí na rezistoru, který podle Ohmova zákona je roven R_eI_e. Platí tedy $U_e = \varepsilon_e - R_eI_e$.

Proudy a napětí v síti musí splňovat 1. a 2. Kirchhoffův zákon. Jinými slovy, ohodnocení hran I_e musí být cirkulací ve smyslu definice 8.1.1, str. 131, a ohodnocení hran U_e musí pro každý obvod (tj. kružnici) splňovat podmínku (15.2), a tedy podle věty 15.2.2 musí být potenciálovým rozdílem.

Všechny tyto podmínky můžeme snadno vyjádřit pomocí soustavy lineárních rovnic, v nich jako neznámé vystupují proudy v jednotlivých hranách I_e. U podmínek z 1. Kirchhoffova zákona to je zcela přímočaré. Pro podmínky plynoucí z 2. Kirchhoffova zákona vyjádříme napětí U_e pomocí vztahu $U_e = \varepsilon_e - R_eI_e$, v němž ε_e pokládáme za konstantu a převedeme ji na pravou stranu. Typická rovnice pro některou kružnici tedy má tvar

$$R_1I_1 + R_2I_2 + \ldots = \varepsilon_1 + \varepsilon_2 + \ldots$$

Soustava rovnic pro všechny vrcholy a všechny kružnice má ovšem zbytečně mnoho rovnic a je lineárně závislá. Pro řešení soustavy je výhodné, aby rovnice byly lineárně nezávislé. Toho lze docílit takto:

Zvolíme kostru grafu (předpokládáme, že síť je souvislá) a vzhledem k této kostře nalezneme fundamentální soustavu řezů a fundamentální soustavu kružnic. Rovnice

pro proudy (1. Kirchhoffův zákon) sestavíme pouze pro fundamentální řezy, rovnice pro napětí (2. Kirchhoffův zákon) sestavíme pouze pro fundamentální kružnice. Z vět 15.3.4 a 15.3.6 vyplývá, že platí-li oba Kirchhoffovy zákony pro fundamentální řezy a fundamentální kružnice, platí oba v plném rozsahu, tj. pro všechny řezy i pro všechny kružnice (obvody).

Podle vět 15.1.6 a 15.2.8 má tato soustava celkem m rovnic o m neznámých, kde m je počet hran grafu. Kdybychom tuto soustavu rovnic vyřešili, získali bychom velikosti proudů v jednotlivých větvích sítě. Napětí bychom pak dopočítali podle Ohmova zákona.

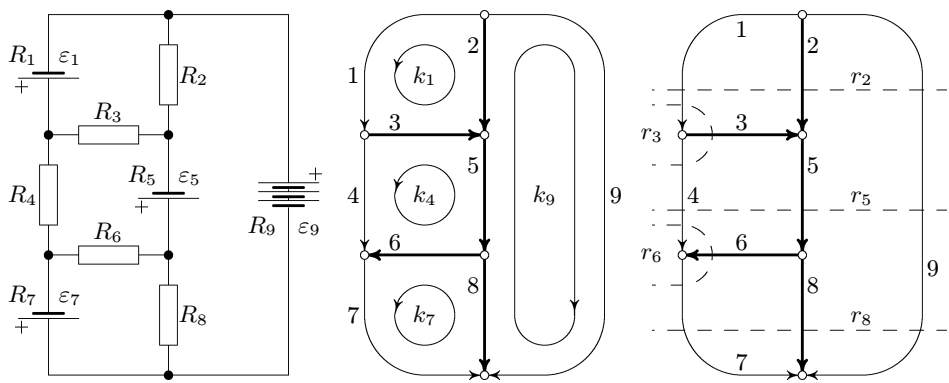
Výpočet však lze ještě dále usnadnit podstatným zmenšením velikosti soustavy rovnic. Rovnice sestavené podle 1. Kirchoffova zákona pro fundamentální řezy totiž mají velmi jednoduchý tvar a umožňují snadno vyjádřit proudy v hranách kostry jako lineární kombinace tzv. *smyčkových proudů*, tj. proudů v hranách mimo kostru. Tato vyjádření dosadíme do rovnic získaných ze 2. Kirchhoffova zákona pro fundamentální kružnice, čímž dostaneme soustavu $m - n + 1$ rovnic pro $m - n + 1$ smyčkových proudů. Řešením této soustavy získáme smyčkové proudy a z nich pak již snadno dopočítáme proudy v ostatních hranách.

Lze dokázat, že matice S soustavy rovnic pro smyčkové proudy je symetrická a že její prvky s_{ij} lze získat i zcela mechanickým způsobem podle vzorce

$$s_{ij} = \sum_{t=1}^{n} R_t k_{it} k_{jt} \ , \tag{15.5}$$

kde k_{it}, k_{jt} jsou prvky fundamentální matice kružnic K. Matici S lze získat také jako maticový součin $S = KRK^{\mathrm{T}}$, kde K je fundamentální matice kružnic, K^{T} je transponovaná fundamentální matice kružnic a R je matice, jejíž prvky na diagonále jsou rovny odporům R_i a ostatní prvky jsou nulové.

Celý postup výpočtu předvedeme na příkladě.



Obrázek 15.3: Příklad elektrické sítě a fundamentální soustavy kružnic a řezů.

15.4.2 Příklad. Nalezněme proudy a napětí v jednotlivých větvích elektrické sítě z obr. 15.3 na předchozí straně. Rovnice vyjadřující 1. Kirchhoffův zákon pro fundamentální soustavu řezů mají tvar

$$\begin{array}{rrcl} r_2: & I_1+I_2+I_9 & = & 0\,, \\ r_3: & -I_1+I_3+I_4 & = & 0\,, \\ r_5: & I_4+I_5+I_9 & = & 0\,, \\ r_6: & I_4+I_6-I_7 & = & 0\,, \\ r_8: & I_7+I_8+I_9 & = & 0 \end{array} \tag{15.6}$$

a rovnice vyjadřující 2. Kirchhoffův zákon pro fundamentální soustavu kružnic mají tvar

$$\begin{array}{rrcl} k_1: & R_1I_1-R_2I_2+R_3I_3 & = & \varepsilon_1\,, \\ k_4: & -R_3I_3+R_4I_4-R_5I_5-R_6I_6 & = & -\varepsilon_5\,, \\ k_7: & R_6I_6+R_7I_7-R_8I_8 & = & \varepsilon_7\,, \\ k_9: & -R_2I_2-R_5I_5-R_8I_8+R_9I_9 & = & -\varepsilon_9-\varepsilon_5\,. \end{array} \tag{15.7}$$

Řešením této soustavy 9 rovnic o 9 neznámých bychom mohli přímo získat proudy v jednotlivých větvích. Méně pracný však je tento postup: Proudy ve větvích, které jsou částí kostry, vyjádříme ze soustavy (15.6) jako lineární kombinace ostatních. Stejné vztahy však lze snadno vyčíst přímo z grafu na obr. 15.3 vpravo.

$$\begin{array}{rcl} I_2 & = & -I_1-I_9\,, \\ I_3 & = & I_1-I_4\,, \\ I_5 & = & -I_4-I_9\,, \\ I_6 & = & -I_4+I_7\,, \\ I_8 & = & -I_7-I_9\,. \end{array} \tag{15.8}$$

Nyní dosadíme ze vztahů (15.8) do soustavy (15.7), čímž získáme pouze $m-n+1=4$ rovnice o 4 neznámých I_1, I_4, I_7, I_9:

$$\begin{array}{rcl} (R_1+R_2+R_3)I_1-R_3I_4+R_2I_9 & = & \varepsilon_1\,, \\ -R_3I_1+(R_3+R_4+R_5+R_6)I_4-R_6I_7+R_5I_9 & = & -\varepsilon_5\,, \\ -R_6I_4+(R_6+R_7+R_8)I_7+R_8I_9 & = & \varepsilon_7\,, \\ R_2I_1+R_5I_4+R_8I_7+(R_2+R_5+R_8+R_9)I_9 & = & -\varepsilon_9-\varepsilon_5\,. \end{array} \tag{15.9}$$

Vyřešením této soustavy získáme proudy I_1, I_4, I_7, I_9 a dosazením do (15.8) zbývající proudy. Ověřte, že tytéž koeficienty soustavy rovnic (15.9) poskytuje i vzorec (15.5).

15.4.3 Cvičení. Vyřešte příklad 15.4.2 pro konkrétní hodnoty $\varepsilon_1=\varepsilon_5=\varepsilon_7=$ $=6\mathrm{V}, \varepsilon_9=12\mathrm{V}, R_1=R_5=R_7=1\,\Omega, R_9=2\,\Omega, R_2=R_3=R_4=R_6=R_8=10\,\Omega$.

Pak zvolte jinou kostru, popř. i jinou orientaci hran a řešení opakujte. Výsledek musí být samozřejmě týž.

Výsledky cvičení

V–1.3.2 (str. 16).

2. Platí $d(x) \geq |E(x)| \geq |V(x)|$.
3. Platí $m(x,x) = m^+(x,x) =$ počet smyček ve vrcholu x .
4. Nemůže. Její vrchol by musel zároveň patřit i nepatřit do množiny A .
5. Platí $|W(A)| \geq |V(A) \setminus A|$.

V–1.4.3 (str. 18). Vrcholy označte počínaje levým vrcholem ve směru hodinových ručiček d, c, e, b, f, a. Není to jediná možnost.

V–1.4.4 (str. 18). Prvé tři jsou navzájem izomorfní, čtvrtý nikoli.

V–1.5.4 (str. 20). Ze sledu, který není cestou, můžeme vynecháním částí mezi opakujícími se vrcholy získat cestu, která vede z téhož vrcholu do téhož vrcholu, a zajišťuje tedy stejnou dostupnost.

V–1.8.2 (str. 25). Dvanáct hran. Kdyby graf H měl navíc smyčku, bylo by to 13 hran.

V–1.8.3 (str. 25). Mohou. Všechny hrany obou grafů jsou smyčkami, neboť jiné hrany by se v obou maticích sousednosti projevily nestejnými hodnotami.

V–1.8.5 (str. 25). Pro orientovaný graf je

$$BB^{\mathrm{T}} = \begin{cases} -m^+(i,j) & \text{pro } i \neq j, \\ d(i) & \text{pro } i = j, \end{cases}$$

zatímco pro neorientovaný graf je

$$BB^{\mathrm{T}} = \begin{cases} m(i,j) & \text{pro } i \neq j, \\ d(i) & \text{pro } i = j. \end{cases}$$

V–2.2.9 (str. 43). Vzhledem k obrovskému počtu vrcholů (několik trilionů) je hledání cest v takovém grafu nereálné. V takových případech je vždy třeba využít speciálních vlastností konkrétní úlohy.

V–3.1.6 (str. 51). V kroku 2 vybíráme vždy hranu, jejíž jeden vrchol má značku a druhý nikoli.

V–3.1.7 (str. 51). V algoritmu 3.1.1, str. 49, upraveném podle cvičení 3.1.6, zařadíme do kroku 2 test, zda existuje hrana, která spojuje dva označkované vrcholy a která nebyla použita v kroku 3 k označkování žádného z těchto dvou vrcholů. K tomu lze použít hodnoty ODKUD.

V–3.3.8 (str. 57). Není to pravda. Například v grafu na obr. 5.2, str. 74, v době návratu z vrcholů 4 nebo 6 není označkován vrchol 9.

V–4.2.10 (str. 62). Všechny vrcholy jsou (dokonce orientovaně) dostupné z kořene, proto je graf souvislý. Z vlastností vstupních stupňů plyne, že je-li n počet vrcholů, je počet hran $n - 1$. Podle věty 4.2.9 je tedy stromem.

V–4.2.12 (str. 62). Je-li souvislý a má-li alespoň tolik hran, kolik má vrcholů.

V–4.2.13 (str. 62). Dvě nebo tři.

V–4.2.14 (str. 63). K_4 má dvě neizomorfní kostry, K_5 má tři.

V–4.3.12 (str. 66). Kdyby ceny hran nebyly různé, mohla by přidáním všech hran splňujících podmínku $(*)$ vzniknout v grafu L kružnice. Jednoduchým příkladem je graf, kde několik nejlevnějších hran má stejnou cenu a tvoří kružnici.

Úprava Borůvkova algoritmu spočívá v tom, že ceny hran učiníme různými. Lze to udělat buď zvětšením všech cen o velmi malá různá čísla, anebo lépe tak, že při porovnávání cen dvou hran v případě stejných cen rozhodneme podle pořadových čísel obou hran.

V–4.4.6 (str. 68). Cesty, které jsou vrcholově disjunktní, jsou i hranově disjunktní. Naopak to ovšem neplatí.

Rovnost neplatí např. pro graf o pěti vrcholech, který vznikne „slepením“ dvou trojúhelníků v jednom vrcholu – ten pak bude artikulací výsledného grafu.

V–5.2.10 (str. 79). V grafu na obr. 5.3 je 9 různých topologických uspořádání vrcholů: 123456879, 123546879, 123564879, 123568479, 124356879, 132456879, 132546879, 132564879, 132568479, 123456879.

V–5.2.11 (str. 79). Je-li $v_1, v_2, \ldots, v_n$ topologické uspořádání vrcholů, pak topologické uspořádání hran získáme tak, že nejprve zařadíme všechny hrany z množiny $E^+(v_1)$, pak všechny hrany z $E^+(v_2)$ atd.

Je-li $e_1, e_2, \ldots, e_m$ topologické uspořádání hran, můžeme topologické uspořádání vrcholů získat tak, že nejprve zařadíme vrcholy s kladným výstupním stupněm, a to v pořadí, v němž se vyskytují poprvé v posloupnosti $\text{PV}(e_1), \text{PV}(e_2), \ldots, \text{PV}(e_m)$ a na konec zařadíme v libovolném pořadí vrcholy s výstupním stupněm nulovým.

V–5.3.5 (str. 81). Tvrzení 8 a 9 přidat lze, viz následující důkaz. Tvrzení 10 splňují i grafy, které mají více než $n-1$ hran, proto tvrzení 10 přidat nelze.

$4 \Rightarrow 8$: z existence kořene plyne souvislost.

$8 \Rightarrow 9$: z vlastností stupňů vrcholů plyne, že graf má přesně $n-1$ hran. Ze souvislosti a počtu hran plyne, že G je stromem, a proto neobsahuje kružnici.

$9 \Rightarrow 5$: neobsahuje-li graf kružnici, neobsahuje ani cyklus.

V–5.5.5 (str. 86). Graf hry je podobný jako na obrázku 5.7, ale bez vrcholu 0. Jádrem je množina $\{1, 5, 9, 13\}$.

V–6.3.9 (str. 100).

a:

vrchol	1	2	3	4	5	6	7	8	9	10
vzdálenost	0	14	6	10	22	28	31	42	30	35
ODKUD	–	3–2	1–3	3–4	4–5	5–6	6–7 4–7	7–8	5–9	6–10

b:

vrchol	1	2	3	4	5	6	7	8	9
vzdálenost	0	2	2	6	1	0	9	4	6
ODKUD	–	1–2	2–3	2–4	3–5	5–6 3–6	5–7	6–8	8–9

V–6.5.3 (str. 103). Příkladem je graf z obr. 6.3, str. 99.

V–6.7.5 (str. 108). Výsledná matice vzdáleností:

$$\begin{pmatrix} 0 & 2 & 8 & 4 & 4 \\ 8 & 0 & 6 & 2 & 2 \\ 3 & -5 & 0 & -3 & -3 \\ 7 & -1 & 4 & 0 & 1 \\ 6 & -2 & 4 & 0 & 0 \end{pmatrix}.$$

V–6.7.6 (str. 108). Matici Q inicializujeme příkazem $Q(i,j) := j$ pro všechna i, j a výkonný příkaz uvnitř cyklů nahradíme příkazem:

když $M(i,j) > M(i,k) + M(k,j)$,
pak proveď $M(i,j) := M(i,k) + M(k,j)$ a navíc $Q(i,j) := Q(i,k)$.

V–7.3.5 (str. 123). Oba algoritmy začínají s rozdílně definovanými maticemi.

Výchozí matice A ve Floydově algoritmu 6.7.3 zahrnuje kromě hodnot hran také hodnoty triviálních sledů (nuly na diagonále), neboť jejich opakované zahrnutí do součtů $\oplus$ nevadí. Kleeneho algoritmus však musí kvůli obecným uzavřeným polookruhům započítat triviální sledy jen jednou, proto jsou hodnoty triviálních sledů z výchozí matice vyloučeny a jsou započítány až na konci výpočtu.

V–7.3.7 (str. 123). Matice šířek nejširších sledů:

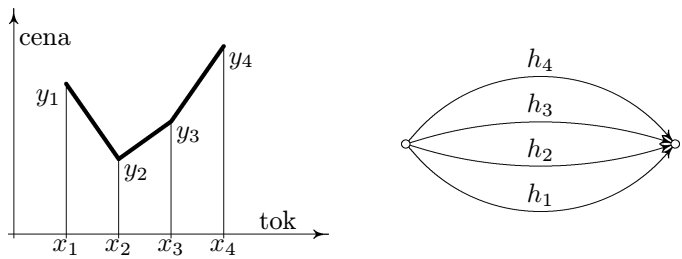
$$\begin{pmatrix} \infty & 6 & 7 & 7 & 7 & 7 \\ 4 & \infty & 5 & 4 & 4 & 4 \\ 4 & 6 & \infty & 4 & 4 & 4 \\ 4 & 6 & 8 & \infty & 4 & 4 \\ 4 & 6 & 7 & 7 & \infty & 7 \\ 4 & 6 & 8 & 8 & 4 & \infty \end{pmatrix}.$$

V–7.4.7 (str. 127). $s(v_1, v_1) = 1$, $s(v_1, v_2) = 3\frac{1}{4}$, $s(v_1, v_3) = 3\frac{1}{2}$, $s(v_1, v_4) = 1$. $s(v_1, v_5) = \frac{7}{8}$. $s(v_1, v_6) = \frac{1}{8}$.

V–8.1.14 (str. 135). Záporný tok v hraně incidentní s v by nemusel procházet hranou e_v a omezení toku v hraně e_v by bylo neúčinné.

V–8.1.17 (str. 135). Je-li např. tok v hraně h omezen na interval $\langle x_1, x_4 \rangle$ a je-li závislost ceny toku na jeho velikosti dána grafem na obr. 15.4, nahradíme hranu h čtveřicí rovnoběžných hran $h_1, \ldots, h_4$ takto:

e	$l(e)$	$c(e)$	$a(e)$
h_1	x_1	x_1	y_1/x_1
h_2	0	$x_2 - x_1$	$(y_2 - y_1)/(x2 - x1)$
h_3	0	$x_3 - x_2$	$(y_3 - y_2)/(x3 - x2)$
h_4	0	$x_4 - x_3$	$(y_4 - y_3)/(x4 - x3)$



Obrázek 15.4: Konvexní závislost ceny toku na velikosti toku.

V–8.2.10 (str. 141). Kapacity hran budou jednotkové, dolní omezení nulové. Ke grafu přidáme návratovou hranu z cílového vrcholu c do výchozího vrcholu r a v této jediné hraně bude dolní i horní omezení toku jednotkové. Nejkratší cestu budou tvořit původní hrany s jednotkovým tokem.

Cyklus se zápornou délkou může způsobit chybný výsledek, neboť jeho hrany mohou být nasyceny tokem, a tím mohou být nepoužitelné pro tok, který má určit nejkratší cestu.

V–8.4.10 (str. 151). Pokud přípustný tok najdeme, bylo omezení stanoveno dobře. (To platí pro hledání přípustného toku. Pro hledání toku maximálního by omezení mohla být i takto příliš silná.)

Pokud přípustná cirkulace neexistuje a návratová hrana je součástí řezu se zápornou kapacitou, pak je třeba omezení v návratové hraně změnit (uvolnit). Pokud přípustná cirkulace neexistuje, ale řez se zápornou kapacitou návratovou hranu neobsahuje, pak omezení v návratové hraně bylo stanoveno dobře, ale přípustný tok v původním grafu neexistuje.

V–9.1.5 (str. 164). Doprostřed hromady hnoje umístíme fiktivní stromek.

V–9.1.8 (str. 164). Nechť je dána úloha o nejlevnějším maximálním párování. Zvolíme nějaké dosti velké kladné číslo M, např. větší než součet cen všech hran, a pak pro každou hranu e její cenu $c(e)$ nahradíme cenou $M - c(e)$. V grafu s takto změněnými cenami pak řešíme úlohu o nejdražším párování.

Označme m počet hran v maximálním párování. Výsledné párování P, které je nejdražší vůči novým cenám, musí obsahovat také m hran, neboť jinak by kterékoli maximální párování mělo lepší (totiž vyšší) cenu. Párování P je ovšem nejdražší i mezi všemi maximálními párováními a jeho cena je $mM - \sum_{e\in P} c(e)$. Párování P tedy má ze všech maximálních párování nejmenší hodnotu $\sum_{e\in P} c(e)$.

V–9.3.6 (str. 173). Kdyby bylo $|X| > |Y|$, platilo by $\sum_{x\in X} d(x) > \sum_{y\in Y} d(y)$, což není možné, neboť součty stupňů na obou stranách grafu musí být stejné. Rovnost $|X| = |Y|$ je možná pouze tak, že všechny vrcholy mají stejný stupeň.

V–9.3.7 (str. 173). Předpoklady věty 9.3.5 jsou splněny, existuje tedy párování, které nasycuje jednu z obou množin. Graf je však regulární (všechny vrcholy mají stejný stupeň), proto $|X| = |Y|$. Zmíněné párování je tedy perfektní.

V–10.1.6 (str. 181). Pro $k = 3$ a $n = 2$ jsou de Bruijnovy posloupnosti dvě: 00011101 a 00010111. (Posloupnost, kterou získáme cyklickou záměnou, pokládáme za tutéž cyklickou posloupnost.)

Pro $k = 2$ a $n = 3$ má de Bruijnova posloupnost délku 9 a těchto posloupností je 22.

V–10.2.4 (str. 182). Graf splňuje předpoklady věty 10.2.2, existuje v něm tedy uzavřený orientovaný tah, který prochází všemi hranami. Díky souvislosti tento tah prochází i všemi vrcholy. Každé dva vrcholy tedy jsou spojeny orientovaným sledem.

V–10.2.8 (str. 183). Oba vrcholy lichého stupně leží v téže komponentě souvislosti. Pro každou komponentu souvislosti tedy bude stačit jeden tah, jeden z těchto tahů bude neuzavřený.

Kdyby počet vrcholů lichého stupně byl 4, byly by nutné 3 nebo 4 tahy v závislosti na tom, zda všechny vrcholy lichého stupně leží v téže komponentě, nebo dva a dva v různých komponentách.

V–10.2.10 (str. 184). Označme $K = \sum_{v \in V} |R(v)|$. Je-li $K = 0$, pokrývá hrany grafu jeden uzavřený (a eulerovský) orientovaný tah. Je-li $K > 0$, je k pokrytí hran třeba $K/2$ neuzavřených orientovaných tahů.

Důkaz: Je-li $K > 0$, označme k^+ součet všech kladných a k^- součet všech záporných hodnot R. Poněvadž $\sum_{v \in V} R(v) = 0$, platí $k^+ = -k^- = K/2$. Můžeme tedy ke grafu přidat celkem k^+ hran, které vedou z vrcholů se záporným R do vrcholů s kladným R, a to tak, aby vznikl graf G', v němž každý vrchol má stejný vstupní i výstupní stupeň. V grafu G' existuje orientovaný uzavřený eulerovský tah. Odstraněním přidaných hran se tento tah rozpadne na k^+ neuzavřených tahů, které pokrývají hrany původního grafu.

Méně než $K/2$ tahů k pokrytí hran nestačí, neboť přidáním menšího počtu hran nelze získat graf, v němž by existoval orientovaný uzavřený eulerovský tah. □

V–10.3.4 (str. 186). Každou hranu, kterou sled prochází k-krát, nahraďme k kopiemi. Kdyby pro některou hranu bylo $k > 2$, bylo by možno dvě z těchto hran odebrat. Sudost stupňů by tím zůstala zachována, ale eulerovský tah by se zkrátil.

V–10.3.5 (str. 186). Kdyby některé dvě cesty měly společnou hranu e, párování P by nebylo nejlevnější. Vynecháním hrany e z obou cest se tyto cesty rozpadnou na části, z nichž lze sestavit jiné dvě cesty o menší celkové délce. Viz obr. 15.5.

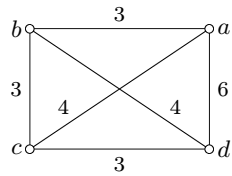
Obrázek 15.5: Náhrada dvou cest se společnou hranou e dvěma hranově disjunktními cestami.

V–10.3.7 (str. 186). Je-li v grafu hrana se zápornou délkou, úloha ztrácí smysl. Opakovaným procházením takovou hranou sem a tam lze totiž dosáhnout libovolně malé (záporné) délky sledu. Žádný sled pak není nejkratší, vždy existuje sled ještě kratší.

V–12.2.7 (str. 203). Jsou-li fixovány vrcholy x, y, pak prodloužíme o dostatečně velkou hodnotu M všechny hrany z množiny $E(x) \cup E(y)$. Hamiltonovské cesty, které začínají ve vrcholu x a končí ve vrcholu y (nebo naopak), se tím prodlouží o $2M$, zatímco všechny ostatní hamiltonovské cesty se prodlouží o $3M$ nebo o $4M$. Pokud v daném grafu existuje vůbec nějaká hamiltonovská cesta spojující x a y, pak nejkratší hamiltonovská cesta v grafu s prodlouženými hranami bude spojovat právě vrcholy x a y.

V–12.2.8 (str. 204). Stačí, aby M bylo větší než délka optimální cesty. Poněvadž ji předem neznáme, lze zvolit např. délku nejdelší hrany vynásobenou počtem vrcholů.

V–12.2.9 (str. 204). Viz obr. 15.6.



Obrázek 15.6: Příklad, že nejkratší hamiltonovská cesta nemusí být obsažena v nejkratší hamiltonovské kružnici.

V–12.3.1 (str. 204). Příkladem je graf $K_{2,3}$.

V–12.3.4 (str. 205). V bipartitním grafu musí hamiltonovská kružnice přecházet střídavě mezi oběma stranami grafu, obě strany by tedy musely mít stejný počet vrcholů.

V–12.4.3 (str. 208). V prvých třech podúlohách sice dostaneme stejné minimální kostry jako v 12.4.2, ale s podstatně silnějšími přidanými omezeními, což urychlí následující výpočet. Průběh výpočtu může být tento:

podúloha	zakázané hrany	vynucené hrany	odhad	komentář
1	–	–	30	větveno na 2, 3, 4
2	(1, 3)	–	38	umrtveno odhadem
3	(1, 4)	(1, 3)	37	umrtveno odhadem
4	(1, 2), (1, 5), (1, 6)	(1, 3), (1, 4)	33	větveno na 5, 6, 7
5	(1, 2), (1, 5), (1, 6), (2, 5)	(1, 3), (1, 4)	38	hamilt. cesta
6	(1, 2), (1, 5), (1, 6), (2, 3)	(1, 3), (1, 4), (2, 5)	36	umrtveno odhadem
7	(1, 2), (1, 5), (1, 6), (2, 4), (2, 6)	(1, 3), (1, 4), (2, 5), (2, 3)	36	optimální řešení

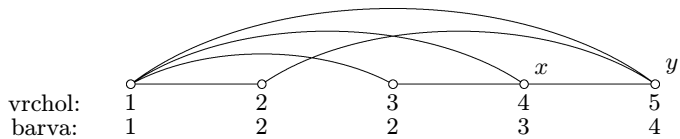
V–12.4.7 (str. 212). Délky všech hamiltonovských cest z x do y se penalizací zvýší o hodnotu $(2\sum_{i=1}^{n} p(i)) - p(x) - p(y)$. Jestliže jsme při použití penále p dostali nejlevnější 1-strom o ceně C, lze jako dolní odhad použít číslo $C + p(x) + p(y) - 2\sum_{i=1}^{n} p(i)$.

V–13.1.2 (str. 213). V obarvení podle obrázku 13.1 stačí barvu D nahradit barvou A. Graf, který má smyčku, nelze obarvit.

V–13.1.12 (str. 217). Strom o alespoň dvou vrcholech je vždy dvoubarevný.

V–13.2.4 (str. 220). Nejrychlejší výpočet bude při uspořádání vrcholů vzestupně podle jejich hloubky, tj. kořen jako první a vrchol v hloubce 2 jako poslední. Naopak výpočet bude nejpomalejší, když všechny listy budou na začátku a kořen na konci.

V–13.2.6 (str. 221). Vrcholy mezi nejvyšším a druhým nejvyšším v $P(y)$ mohou ovlivnit barvu vrcholu $x = \max(P(y))$ a tím i barvu vrcholu y. Viz graf na obr. 15.7, kde „vylepšený“ algoritmus přehlédne možnost zvýšit barvu vrcholu 3 a v důsledku toho nenajde optimální obarvení.

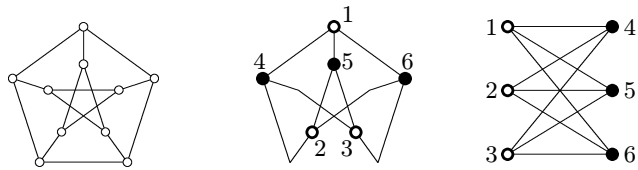


Obrázek 15.7: Protipříklad ke cvičení 13.2.6.

V–14.2.2 (str. 229). Krychle je duální k pravidelnému osmistěnu, pravidelný dvanáctistěn je duální k pravidelnému dvacetistěnu. Čtyřstěn je duální sám k sobě.

V–14.3.2 (str. 231). Důkaz je podobný důkazu věty 14.3.1, ale namísto kostry použijeme napnutý les (viz 4.2.4, str. 60, a 4.2.8, str. 61).

V–14.4.3 (str. 232). Viz obr. 15.8.



Obrázek 15.8: Petersenův graf je homeomorfní s $K_{3,3}$.

V–15.4.3 (str. 244). Rozšířená matice soustavy rovnic pro smyčkové proudy má tvar

$$\left(\begin{array}{rrrr|r} 21 & -10 & 0 & 10 & 6 \\ -10 & 31 & -10 & 1 & 6 \\ 0 & -10 & 21 & 10 & 6 \\ 10 & 1 & 10 & 22 & 18 \end{array}\right).$$

Jejím řešením dostaneme $I_1 = 2{,}56468\,\mathrm{A}, I_4 = 1{,}56498\,\mathrm{A}, I_7 = 2{,}56468\,\mathrm{A}, I_9 = -3{,}22084\,\mathrm{A}$ a dále pak $I_2 = 0{,}65616\,\mathrm{A}, I_3 = 0{,}99970\,\mathrm{A}, I_5 = 1{,}65596\,\mathrm{A}, I_6 = 0{,}99970\,\mathrm{A}, I_8 = 0{,}65616\,\mathrm{A}$.

Literatura

[AHU74] Aho, A. V., Hopcroft, J. E., Ullman, J. D.: *The Design and Analysis of Computer Algorithms.* Reading, Addison-Wesley, 1974.

[Berge58] Berge, C.: *Thèorie des Graphes et ses Applications.* Paris, Gautier--Villars, 1958.

[Berge73] Berge, C.: *Graphs and Hypergraphs.* Amsterdam, North-Holland Publ. Co., 1973.

[Demel91] Demel, J.: *Teorie grafů.* Praha, skripta ČVUT, 1982, 2. vydání 1991.

[Demel88] Demel, J.: *Grafy.* Praha, SNTL, 1988.

[FF62] Ford, L. R. Jr., Fulkerson, D. R.: *Flows in Networks.* Princeton, Princeton University Press, 1962.

[GJ79] Garrey, M. R., Johnson, D. S.: *Computers and Intractability: a Guide to the Theory of NP-Completeness.* San Francisco, W. H. Freeman, 1979.

[Harary69] Harary, F.: *Graph Theory.* Reading, Addison-Wesley, 1969.

[Chr75] Christofides, N.: *Graph theory – An Algorithnic Approach.* New York, Academic Press, 1975.

[Chytil82] Chytil, M.: *Automaty a gramatiky.* Praha, SNTL, 1982.

[KŠCh89] Kolář, J., Štěpánková, O., Chytil, M.: *Logika, algebry a grafy.* Praha, SNTL, 1989.

[Kučera83] Kučera, L.: *Kombinatorické algoritmy.* Praha, SNTL, 1983.

[Lawler76] Lawler, E., L.: *Combinatorial Optimization – Networks and Matroids.* New York, Holt, Reinhart and Winston, Inc., 1976.

[MN96] Matoušek, J., Nešetřil, J.: *Kapitoly z diskrétní matematiky.* Praha, Matfyzpress, 1996.

[Nečas78] Nečas, J.: *Grafy a jejich použití.* Praha, SNTL, 1978.

[Nešetřil79] Nešetřil, J.: *Teorie grafů.* Praha, SNTL, 1979.

[Plesník83] Plesník, J.: *Grafové algoritmy.* Bratislava, VEDA, 1983.

[Sedláček64] Sedláček, J.: *Kombinatorika v teorii a praxi.* Praha, NČSAV, 1964, 2. vydání, 1977.

[Sedláček81] Sedláček, J.: *Úvod do teorie grafů.* Praha, Academia, 1981.

[SwTh81] Swamy, M. N. S., Thulasiraman, K.: *Graphs, Networks and Algorithms.* New York, John Wiley & Sons, Inc., 1981.

[Šišma97] Šišma, P.: *Teorie grafů 1736–1963.* Praha, Prometheus, 1997.

[Töpfer95] Töpfer, P.: *Algoritmy a programovací techniky.* Praha, Prometheus, 1995.

Rejstřík

Přehled značení

$\mathbb{R}$, $\mathbb{N}$	= množina reálných a přirozených čísel (včetně nuly)	1.12.1, str. 33
$\|M\|$	= počet prvků množiny M	1.12.1, str. 33
$A \setminus B$	= rozdíl množin A a B	1.12.1, str. 33
n	= obvykle počet vrcholů grafu	1.10.6, str. 31
m	= obvykle počet hran grafu	1.10.6, str. 31
∞	= nekonečno	6.2.10, str. 95
$O(f)$	= asymptotický odhad funkce f shora	1.10.6, str. 30
$V(G)$	= množina vrcholů grafu G	1.3, str. 15
$E(G)$	= množina hran grafu G	1.3, str. 15
$\mathrm{P}v(e)$	= počáteční vrchol hrany e	1.1.2, str. 11
$\mathrm{K}v(e)$	= koncový vrchol hrany e	1.1.2, str. 11
$V(x)$	= množina sousedů vrcholu x	1.3, str. 15
$V^+(x)$	= množina následníků vrcholu x	1.3, str. 15
$V^-(x)$	= množina předchůdců vrcholu x	1.3, str. 15
$E(v),\ E^+(v),\ E^-(v)$	= okolí vrcholu	1.3, str. 15
$d(v),\ d^+(v),\ d^-(v)$	= stupeň vrcholu	1.3, str. 15
$W(A),\ W^+(A),\ W^-(A)$	= řez určený množinou A	1.3, str. 15
$m(v,w),\ m^+(v,w)$	= násobnost hrany	1.3, str. 15
$G \cong G'$	= izomorfismus grafů G a G'	1.4.2, str. 17
K_n	= úplný graf o n vrcholech	1.6, str. 21
$K_{m,n}$	= úplný bipartitní graf se stranami o m a n vrcholech	1.6, str. 21
$D^+(v)$	= množina vrcholů orientovaně dostupných z vrcholu v	5.1.10, str. 71
G^+	= tranzitivní uzávěr grafu G	5.4.1, str. 81
G^*	= reflexivní a tranzitivní uzávěr grafu G	5.4.1, str. 81
$\oplus, \otimes, {}^*$	= operace v polookruhu	7.1.1, str. 112
$\omega(G)$	= klikovost grafu	13.1.4, str. 214
$\alpha(G)$	= nezávislost grafu	13.1.3, str. 214
$\chi(G)$	= vrcholová barevnost grafu	13.1.1, str. 213
$\chi'(G)$	= hranová barevnost grafu	13.1.15, str. 217
$-G$	= doplňkový graf	13.1.4, str. 214
G^D	= duální graf	14.2, str. 228

Jiří Demel

Grafy
a jejich aplikace

Vydal Jiří Demel, Libčice nad Vltavou
Sazba a obálka Jiří Demel
Vydání třetí, elektronické (vlastním nákladem druhé), 2019