VŠB - Technická univerzita Ostrava **Fakulta elektrotechniky a informatiky**

RNDr. Eliška Ochodková

Grafové algoritmy

Průvodce studiem

Cílová skupina: Toto skriptum je určeno studentům různých studijních oborů informatického programu na Fakultě elektrotechniky a informatiky VŠB - TU Ostrava. Může být vhodnou studijní pomůckou i pro studenty jiných vysokých škol a rovněž pro posluchače kurzů celoživotního vzdělávání.

Prerekvizity: U studentů se předpokládá orientace v problematice informačních technologií a znalost matematiky na úrovni maturitního ročníku střední školy nebo lépe prvního ročníku technické vysoké školy.

Anotace modulu: Studenti se v tomto skrtiptu seznámí se základními pojmy teorie grafů a grafovými algoritmy. Grafové algoritmy jsou částí celku, který nám umožňuje jednoduše a přehledně popisovat reálné systémy. Většinou to jsou systémy, které popisujeme pomocí sítí (počítačové sítě, komunikační sítě, kanalizační sítě, . . .). V grafově reprezentovaných (popsaných) systémech pak grafovými algoritmy snadno nalezneme nejlevnější (nejkratší) cestu, maximální tok v síti, zjistíme existenci (či neexistenci) libovolné cesty apod. Samozřejmě, že se nebudeme zabývat pouze toky v sítích případně nejkratšími cestami, ale také algoritmy na nalezení eulerovského tahu (aplikováno při kreslení na plotru), maximálního párování (aplikace v tanečních), nezávislých množin atd.

Organizace skripta: Skriptum se dělí na části, kapitoly, které odpovídají logickému dělení studované látky, ale nejsou stejně obsáhlé. Předpokládaná doba ke studiu kapitoly se může výrazně lišit, proto jsou velké kapitoly děleny dále na číslované podkapitoly a těm odpovídá níže popsaná struktura.

Členění jednotlivých kapitol:

– Každá kapitola se může skládat z několika podkapitol a začíná obsahem této kapitoly a stručným popisem obsahu kapitoly.

Časová náročnost kapitoly: 10 minut

– Na začátku kapitoly (a podkapitol) je uvedena předpokládaná časová náročnost kapitoly v minutách spolu s ikonou, která na tento údaj upozorňuje. Čas je orientační a může vám sloužit jako hrubé vodítko pro rozvržení studia celého předmětu či kapitoly. Někomu se čas může zdát příliš dlouhý, někomu naopak. Jsou studenti, kteří se s uvedenou problematikou ještě nikdy nesetkali, a naopak takoví, kteří již v tomto oboru mají bohaté zkušenosti.

Cíl kapitoly:
Po prostudování kapitoly budete umět

– Ihned potom je spolu s navigační ikonou uvedeno, co je cílem této podkapitoly, cíle, kterých máte dosáhnout po prostudování této kapitoly - konkrétní dovednosti, znalosti.

– Pak následuje výklad s obrázky, tabulkami a řešenými příklady. Vlastní text obsahuje hypertextové odkazy na definované pojmy. Čtenář se jednoduchým kliknutím na modře vysvícený pojem dostane na jeho definici, nebo kapitolu, tabulku, obrázek. Zpět na místo odskoku se čtenář dostane kliknutím na šipku zpět v Acrobat Readeru (ve kterém je toto skriptum prohlíženo).

– Na začátku a konci každé kapitoly se nachází navigační lišta, pomocí které se čtenář dostane přímo na předcházející nebo následující kapitolu či obsah celého skripta. Na konci skripta se nachází seznam literatury a hypertextový rejstřík.

Animace: Prohledávání grafu do šířky

– Výklad doplňují Java applety demonstrující princip algoritmů nebo odkazy na webovské stránky, na kterých mohou zájemci nalézt doplňující informace, které se spustí kliknutím na příslušný odkaz označený touto ikonou.

Shrnutí kapitoly Pokyny

– Na závěr kapitoly jsou zopakovány hlavní pojmy, které si v ní máte osvojit. Pokud některému z nich ještě nerozumíte, vraťte se k nim ještě jednou.

Otázky ke kapitole Pokyny

– Pro ověření, že jste dobře a úplně látku kapitoly zvládli, máte k dispozici několik otázek a úkolů.

Literatura: Kromě předloženého skripta můžete rozšiřující informace nalézt v uvedené literatuře (bohužel většinou cizojazyčné):

1. Nešetřil J.: Teorie grafů, SNTL, Praha, 1979.
2. Plesník J.: Grafové algoritmy, VEDA, Bratislava, 1983.
3. Demel J.: Grafy, Academia Praha, 2002.
4. Kučera L.: Kombinatorické algoritmy, SNTL Praha, 1991.
5. Cormen, Leiserson, Rivest: Introduction to algorithms, The MIT Press, 1990.
6. McHugh J. A.: Algorithmic graph theory, PRENTICE HALL, 1990.
7. Chrtrand G., Lesniak L.: Graphs & Digraphs, Chapman & Hall/CRC, 2000.

Úspěšné a příjemné studium s touto učebnicí Vám přeje autorka.

Ostrava, prosinec 2003

Autor: Eliška Ochodková

Obsah

Seznam obrázků

Kapitola 1

Základní pojmy teorie grafů

Obsah

1.1 Definice grafu

Časová náročnost kapitoly: 40 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět definovat pojem graf a další s ním související pojmy. Dále budete umět definovat některé speciální typy grafů a uvést jejich příklady.

Orientovaný graf (viz 1.1) je trojice $G = (V, E, \varepsilon)$, kde V je konečná neprázdná ***množina vrcholů***, E je konečná ***množina hran*** (orientovaných) a zobrazení $\varepsilon : E \longrightarrow V^2$, které přiřazuje každé hraně $e \in E$ uspořádanou dvojici vrcholů $\varepsilon(e) = (x, y) \in V^2$ se nazývá ***relací incidence***.

Vrchol x nazýváme ***počátečním vrcholem*** hrany e, značíme jej $PV(e)$. Vrchol y nazýváme ***koncovým vrcholem*** hrany e, značíme jej $KV(e)$. O vrcholech x, y pak říkáme, že jsou *incidentní* s hranou e, a také naopak hrana e je *incidentní* s vrcholy x, y. Oba vrcholy x, y také nazýváme *krajními vrcholy* hrany e.

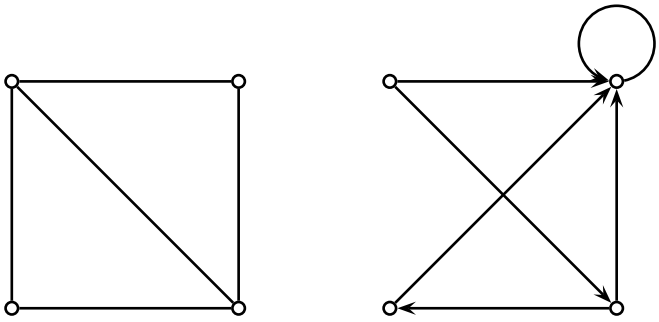
Jestliže $PV(e) = KV(e)$, nazýváme hranu e (orientovanou) ***smyčkou***. Orientovaný graf, který nemá žádnou smyčku, nazýváme orientovaným grafem bez smyček. Vrchol, který není incidentní s žádnou hranou, nazýváme ***izolovaným vrcholem***.

Neorientovaný graf (viz obr. 1.1) je trojice $G = (V, E, \varepsilon)$ tvořená konečnou neprázdnou množinou V, jejíž prvky nazýváme vrcholy, konečnou množinou E, jejíž prvky nazýváme neorientovanými hranami, a zobrazením ε (nazývaným relace incidence), které přiřazuje každé hraně $e \in E$ jedno nebo dvouprvkovou množinu vrcholů. Těmto vrcholům říkáme ***krajní vrcholy*** hrany e. Říkáme také, že jsou incidentní s hranou e (a hrana e je incidentní s těmito vrcholy).

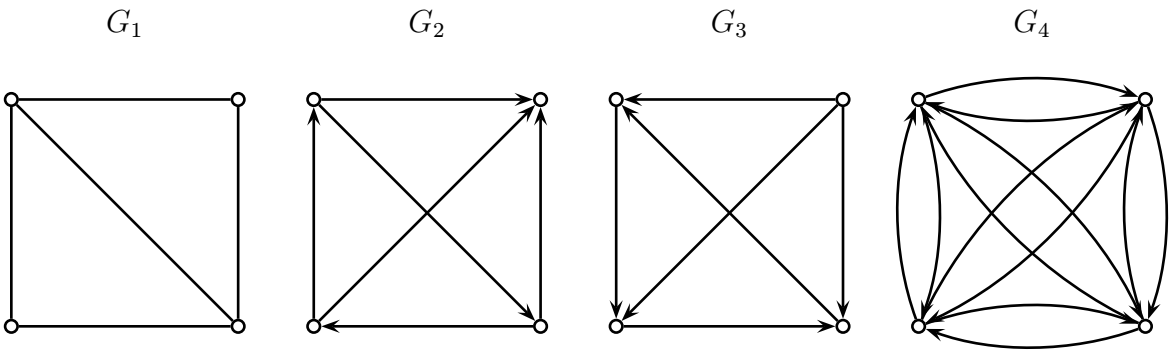
Je-li hrana e incidentní pouze s jediným vrcholem (tj. jestliže množina $\varepsilon(e)$ je jednoprvková), nazýváme hranu e (neorientovanou) smyčkou. Neorientovaný graf, který nemá smyčky, nazýváme neorientovaným grafem bez smyček. Vrchol, který není incidentní s žádnou hranou, nazýváme izolovaným vrcholem.

Vrcholy, které jsou krajními vrcholy jedné hrany, nazýváme ***sousedními***. Vrcholy, které nejsou spojeny hranou nazýváme ***nezávislými***.

Ohodnocený graf (síť) je graf, jehož hrany a (nebo) vrcholy jsou opatřeny číselnými (nebo jinými) hodnotami.



Obrázek 1.1: Neorientovaný graf bez smyček a orientovaný graf se smyčkama.



Obrázek 1.2: Vztahy mezi orientovanými a neorientovanými grafy.

Neorientovaný graf G_1, orientace G_2 grafu G_1, obrácení G_3 grafu G_2, symetrická orientace G_4 grafu G_1.

Mezi oběma druhy grafů (orientovanými i neorientovanými) existují jednoduché vzájemné převody (viz obr. 1.2). Je-li dán orientovaný graf, pak jeho ***symetrizací*** nazýváme neorientovaný graf, který dostaneme tím, že „zapomeneme“ na orientaci hran.

Je-li naopak dán neorientovaný graf G, pak z něj lze vytvořit graf orientovaný dvěma různými přirozenými způsoby:

- zvolíme orientaci hran libovolně a získáme tak tzv. ***orientaci*** grafu G,
- každou neorientovanou hranu, která není smyčkou, nahradíme dvojicí opačně orientovaných hran a každou neorientovanou smyčku nahradíme orientovanou smyčkou, čímž získáme tzv. ***symetrickou orientaci*** grafu G.

Poznamenejme, že orientace grafu G není jednoznačně určena, je to však takový graf G', jehož symetrizací je původní graf G.

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

Orientovaný graf G' nazveme ***obrácením grafu*** G, jestliže G' vznikl z G obrácením orientace všech hran. Je zřejmé, že dalším obrácením grafu G' bychom dostali původní graf G.

Jsou-li dva vrcholy spojeny více než jednou hranou, mluvíme o ***násobných hranách***.

Multigraf je graf, který obsahuje alespoň jednu hranu s násobností ≥ 1 (viz obr. 1.7). ***Prostý graf*** (jednoduchý graf) je graf, ve kterém nejsou násobné hrany ani smyčky, tj. všechny hrany mají násobnost $= 1$.

Počet hran, se kterými je daný vrchol v incidentní, se nazývá ***stupeň vrcholu*** v a označuje se d_v nebo deg_v. Jsou-li v grafu smyčky, pak smyčku započítáváme dvakrát (jakoby hrana z vrcholu vystupovala a také vstupovala).

Symboly $\delta(G)$ a $\Delta(G)$ označujeme **nejmenší** a **největší** stupeň vrcholu v grafu G.

Graf, jehož všechny vrcholy jsou stupně r nazýváme ***regulárním*** (pravidelným) grafem r-tého stupně.

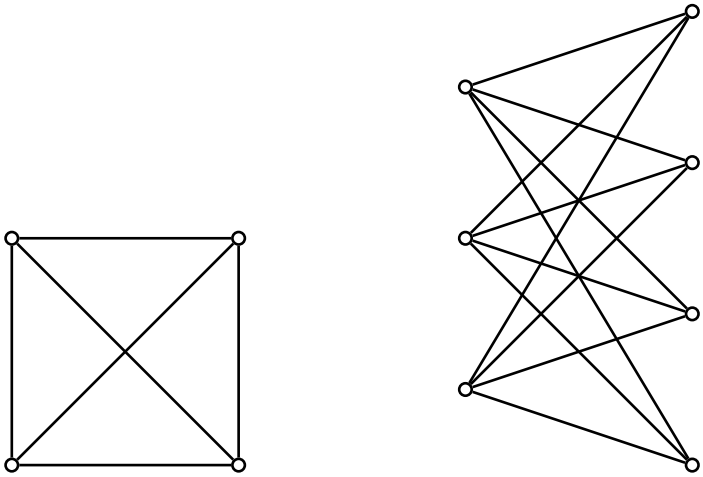
Neorientovaný ***kompletní*** nebo-li **úplný** (viz obr. 1.3) graf na n vrcholech, označovaný K_n, je graf, ve kterém každý vrchol sousedí se všemi ostatními vrcholy.

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

Bipartitní graf je graf, ve kterém se množina vrcholů V skládá ze dvou disjunktních podmnožin X, Y zvaných ***partity*** a množina hran E obsahuje hrany, jejichž jeden koncový vrchol leží v množině X a druhý koncový vrchol leží v množině Y. Žádné dva vrcholy z partity X (resp. Y) nejsou navzájem spojeny hranou.

Úplný bipartitní graf (viz obr. 1.3) je takový bipartitní graf, ve kterém ke každá dvojice vrcholů $x \in X$, $y \in Y$ spojena právě jednou hranou (značíme jej $K_{m,n}$ kde $m = |X|, n = |Y|$).

Diskrétní graf je graf, který nemá žádnou hranu.



Obrázek 1.3: Úplný graf K_4 a úplný bipartitní graf $K_{3,4}$.

Shrnutí kapitoly 1.1

V této kapitole jsme se seznámili se základními pojmy teorie grafů, a to těmito:

1. orientovaný a neorientovaný graf, vrchol, hrana, relace incidence, smyčka, ohodnocený graf,
2. symetrizace, symetrická orientace, obrácení grafu,
3. stupeň vrcholu multigraf, prostý graf, biparitiní graf, úplný graf.

Otázky ke kapitole 1.1

1. Jaký graf může vzniknout symetrizací prostého orientovaného grafu?
2. Jaký graf lépe vystihne skutečnost u uvedených reálných situací - orientovaný nebo neorientovaný?
   (a) Počítačový program se skládá z instrukcí. Instrukce j může být vykonána až po vykonání instrukce i.
   (b) Rodokmen, osoba i je otcem osoby j.
   (c) Silniční síť, silnice i, j mají společnou křižovatku.
3. Uveďte příklad reálné situace, kterou byste popsali neorientovaným ohodnoceným grafem.
4. Uveďte příklad reálné situace, kterou byste popsali bipartitním grafem.

1.2 Porovnávání grafů

Časová náročnost kapitoly: 30 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět definovat pojmy související s vzájemným vztahem dvou grafů a uvést příklady těchto vztahů.

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

Grafy G, G' se nazývají navzájem ***izomorfní***, když existuje dvojice vzájemně jednoznačných zobrazeni $f : V \longrightarrow V'$ a $g : E \longrightarrow E'$ takových, že zachovávají vztahy incidence $\varepsilon, \varepsilon'$. Jinými slovy: pro každou hranu $e \in E$ platí: Jestliže $\varepsilon(e) = (x, y)$, pak $\varepsilon'(g(e)) = (f(x), f(y))$, popř. (jestliže $\varepsilon(e) = x, y$, pak $\varepsilon'(g(e)) = f(x), f(y)$), v závislosti na tom, zda se jedná o orientované či neorientované grafy. Vztah izomorfismu grafů G, G' značíme $G \cong G'$.

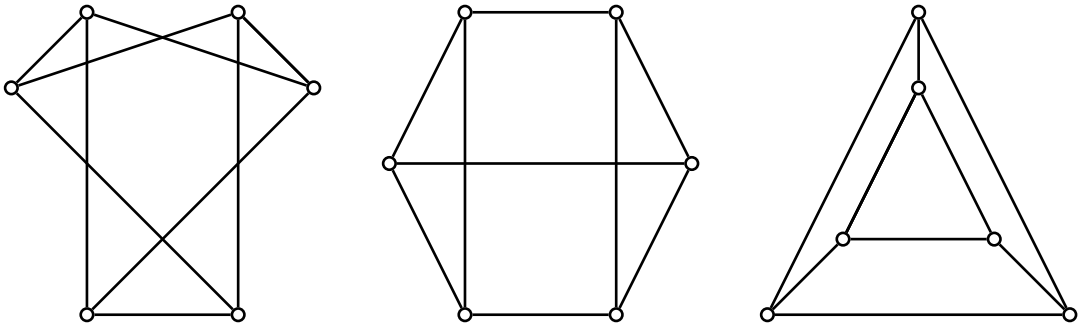
Na obrázku 1.4 jsou nakresleny tři navzájem izomorfní grafy.

Stručně bychom mohli říci, že dva grafy jsou izomorfní, když se liší pouze nakreslením a označením (pojmenováním) vrcholů a hran. Rozhodnutí, zda dva dané grafy jsou izomorfní, však je v řadě případů obtížným úkolem. Při zjišťování izomorfismu lze často využít tato jednoduchá pozorování:

- izomorfní grafy musí mít stejný počet vrcholů a hran,
- vrcholu stupně k může být izomorfismem přiřazen opět pouze vrchol stupně k,
- dvojici sousedních vrcholů může být izomorfismem přiřazena opět jen dvojice sousedních vrcholů.

Některé vlastnosti grafů se izomorfismem zachovávají - dva grafy nemohou být izomorfní, jestliže jeden z nich má takovou vlastnost a druhý nikoli. Příkladem takové vlastnosti je třeba „počet vrcholů stupně 2, které mají alespoň jednoho souseda stupně 3“. Další takové vlastnosti se týkají např. existence cest, kružnic a cyklů určité délky apod.

Graf $H = (V', E')$ je ***podgrafem*** grafu $G(V, E)$, jestliže $V' \subseteq V$ a $E' \subseteq E$ a vztah incidence ε' je zúžením zobrazení ε. Jinými slovy, graf G' je podgrafem grafu G, vznikne-li z grafu G vynecháním některých vrcholů a hran. Podstatné je, že podgraf je také grafem: s každou hranou musí patřit do podgrafu i její krajní vrcholy. Poznamenejme, že každý graf je podgrafem sebe sama.



Obrázek 1.4: Vzájemně izomorfní grafy.

Pokud $V' = V$, potom se tento podgraf nazývá ***faktor*** grafu G. Faktor tedy vznikne vynecháním některých (nebo žádných) hran.

Graf G' je ***úplným podgrafem*** grafu G, je-li podgrafem grafu G a graf G' obsahuje všechny hrany grafu G, jejichž (oba) krajní vrcholy leží ve $V(G')$. Úplný podgraf (také se někdy nazývá podgraf indukovaný množinou vrcholů $A \subseteq V(G)$) tedy vznikne vynecháním některých (nebo žádných) vrcholů a všech hran, které s nimi byly incidentní (viz obr. 1.6).

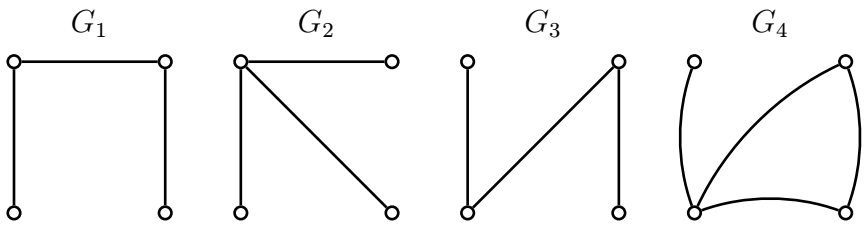
Shrnutí kapitoly 1.2

V této kapitole jsme se seznámili s pojmy teorie grafů, které se týkají vzájemných vztahů dvou grafů, a to těmito:

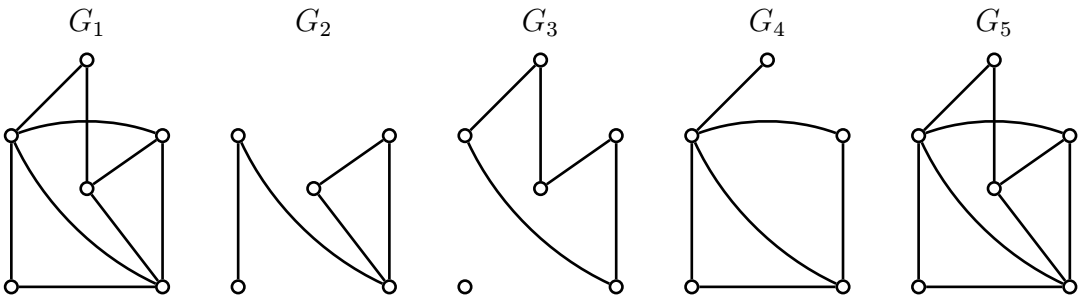
1. rovnost grafů, izomorfismus grafů,
2. podgraf, faktor, úplný podgraf.

Otázky ke kapitole 1.2

1. Rozhodněte, které grafy z obr. 1.5 jsou vzájemně izomorfní.
2. Rozhodněte, zda grafy G_2, G_3, G_4, G_5 jsou podgrafem, faktorem, nebo úplným podgrafem grafu G_1 na obr. 1.6.



Obrázek 1.5: Grafy pro procvičení znalostí z kapitoly 1.2.



Obrázek 1.6: Grafy pro procvičení znalostí z kapitoly 1.2.

1.3 Sledy a odvozené pojmy

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět definovat pojem sled, tah, cesta, cyklus a uvést jejich příklady.

Posloupnost vrcholů a hran $v_0, e_1, v_1, \ldots, e_n, v_n$ taková, že hrana e_i pro $i = 1, 2, \ldots$ má koncové vrcholy v_{i-1}, v_i, se nazývá ***sled***.

V orientovaném i neorientovaném sledu na sebe vrcholy i hrany „navazují“, u orientovaného sledu navíc požadujeme, aby všechny hrany byly orientovány „vpřed ve směru sledu“.

Sled, ve kterém se žádná hrana nevyskytuje více než jednou, se nazývá ***tah***.

Tah, ve kterém se žádný vrchol nevyskytuje více než jednou, se nazývá ***cesta***.

Každá cesta je zároveň tahem a sledem, avšak tah je vždy sledem, ale ne vždy cestou.

Sled (tah), který má alespoň jednu hranu a jehož počáteční a koncový vrchol splývají, nazýváme uzavřeným sledem (uzavřeným tahem).

Orientovanou cestu, jejíž první vrchol je totožný s posledním, nazýváme ***cyklus***.

Neorientovanou cestu, jejíž první vrchol je totožný s posledním, nazýváme ***kružnice***.

Jestliže orientovaný graf neobsahuje žádný cyklus, nazývá se ***acyklickým***.

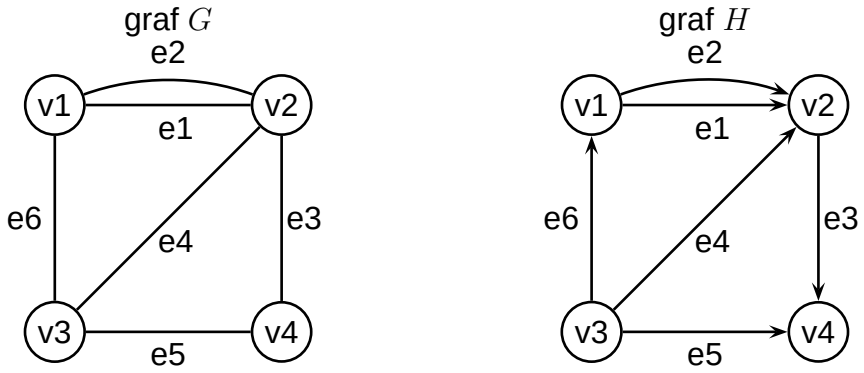
Délkou sledu, tahu, cesty nebo cyklu (kružnice) budeme rozumět počet jeho hran. Tedy cyklus na n vrcholech má délku n, kdežto cesta na n vrcholech má délku $n - 1$.

Vrchol y je orientovaně (neorientovaně) ***dostupný*** z vrcholu x, jestliže existuje orientovaný (neorientovaný) sled vedoucí z vrcholu x do vrcholu y.

Shrnutí kapitoly 1.3

V této kapitole jsme se seznámili s pojmy teorie grafů, které se týkají posloupností na sebe navazujících vrcholů a hran, a to těmito:

1. sled, tah cesta,



Obrázek 1.7: Grafy pro procvičení znalostí z kapitoly 1.3.

2. cyklus, kružnice,
3. dostupnost.

Otázky ke kapitole 1.3

1. Rozhodněte, zda je posloupnost v_3, e_2, v_4, e_4, v_1 v grafu G na obr. 1.7 sledem.
2. Rozhodněte, zda je posloupnost $v_3, e_6, v_1, e_1, v_2, e_3, v_4$ v grafu G na obr.1.7 sledem. Nalezněte v tomto grafu cyklus.
3. Rozhodněte, zda je posloupnost $v_3, e_4, v_2, e_4, v_3, e_6, v_1$ v grafu G na obr.1.7 tahem. Pokud není, nalezněte v tomto grafu tah.
4. Rozhodněte, zda je posloupnost $v_3, e_5, v_4, e_3, v_2, e_4, v_3, e_6, v_1$ v grafu H na obr. 1.7 cestou. Pokud není, nalezněte v tomto grafu cestu. Nalezněte v tomto grafu kružnici.
5. Rozhodněte, zda vrchol v_1 orientovaně dostupný z vrcholu v_2 v grafu G na obr. 1.7.

1.4 Pojmy založené na cestách

Časová náročnost kapitoly: 30 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět definovat pojem souvislý a silně souvislý graf, strom a kostra a uvést jejich příklady.

Graf, ve kterém jsou z libovolného vrcholu dostupné (po neorientované cestě) všechny ostatní vrcholy, se nazývá ***souvislý*** graf (viz obr. 1.10). Maximální souvislý podgraf daného grafu G se nazývá ***souvislá komponenta*** grafu G. Platí, že každý vrchol grafu je obsažen v právě jedné komponentě souvislosti.

Les je graf, který neobsahuje kružnici.

Strom je graf, který neobsahuje kružnici a je souvislý.

Faktor souvislého grafu G, který je ConceptUsagestromemstromStrom, se nazývá ***kostra*** grafu (viz obr. 1.8).

Vrcholová souvislost (ozn. $\kappa(G)$) je minimální počet vrcholů, jejichž vynecháním dostaneme buď nesouvislý graf nebo izolovaný vrchol. Graf G nazýváme **vrcholově k-souvislým**, jestliže $\kappa(G) \geq k$. Množinu vrcholů, jejichž vynecháním se graf G stane nesouvislým nebo tirviálním, nazýváme ***vrcholovým řezem*** G. Řez, který obsahuje jeden vrchol, se nazývá ***artikulace***.

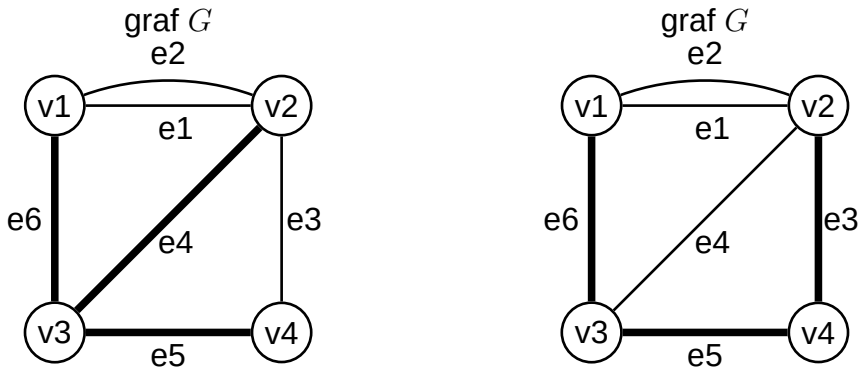
Hranová souvislost $\kappa'(G)$ grafu G je minimální počet hran, jejichž vynecháním dostaneme nesouvislý nebo triviální graf. Graf G nazýváme **hranově k-souvislým**, jestliže $\kappa'(G) \geq k$. Množinu hran, jejichž vynecháním se graf G stane nesouvislým nebo triviálním, nazýváme ***hranovým řezem*** G. Řez, obsahující jedinou hranu, se nazývá ***most***.

Silně souvislý graf (viz obr. 1.10) je takový graf, ve kterém pro každou dvojici vrcholů x, y existuje orientovaná cesta z x do y a také zpět z y do x.

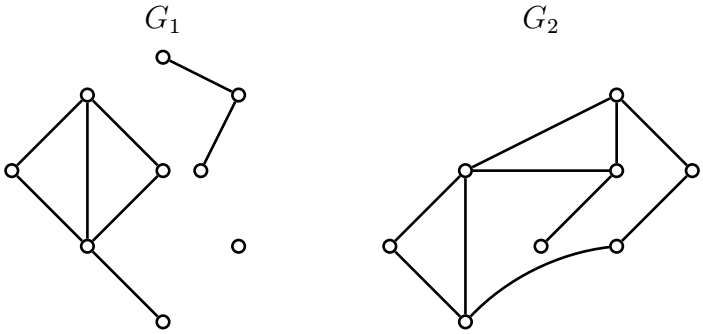
Maximální silně souvislý podgraf daného grafu G se nazývá ***silně souvislá komponenta*** grafu G.

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

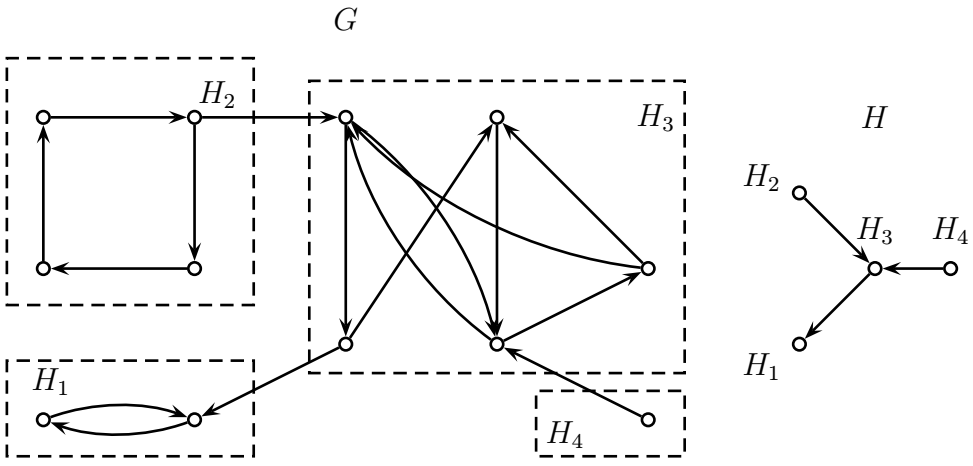
1. vrcholy grafu G' jsou všechny silně souvislé komponenty grafu G,
2. Vrcholy H_1, H_2 jsou v grafu G' spojeny hranou z H_1 do H_2 právě tehdy, když $H_1 \neq H_2$ a v grafu G existuje hrana z některého vrcholu komponenty H_1 do některého vrcholu komponenty H_2.



Obrázek 1.8: Graf G a jeho dvě kostry (silně vytažené).



Obrázek 1.9: Nesouvislý graf G_1 a souvislý graf G_2.



Obrázek 1.10: Orient. graf G, jeho silně souvislé komponenty a kondenzace H.

Shrnutí kapitoly 1.4

V této kapitole jsme si seznámili s pojmy teorie grafů, které jsou založené na pojmu cesta, a to těmito:

1. souvislý graf, silně souvislý graf,komponenta grafu, kondenzace grafu,
2. les, strom, kostra grafu, acyklický graf.

Otázky ke kapitole 1.4

1. Nalezněte kostry v grafech G_1 a G_2 na obrázku 1.10.
2. V grafu a G_2 na obrázku 1.10 nalezněte vrcholový resp. hranový řez. Je některý z řezů artikulací resp. mostem?
3. Za jakých podmínek má graf několik různých koster?

Kapitola 2

Reprezentace grafů

Časová náročnost kapitoly: 40 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět reprezentovat graf několika způsoby a pro konkrétní úlohu vybrat nejvhodnější reprezentaci daného grafu.

Snadno si představíme, že nejsrozumitelnější reprezentací grafu je jeho obrázek. Vrcholy jsou v obrázku kresleny jako kroužky, hrany jako čáry (oblouky). Orientovanou hranu znázorňujeme jako čáru s šipkou od počátečního ke koncovému vrcholu. Ze dvou nakreslení grafu preferujeme to, které obsahuje menší počet průsečíků hran.

Rovinné nakreslení grafu je takové nakreslení grafu, že libovolné dvě křivky přiřazené různým hranám grafu mají společné nejvýše své krajní body přiřazené vrcholům grafu. ***Rovinný graf*** (planární graf) je takový graf, ke kterému existuje rovinné nakreslení (úplný graf K_4 není rovinný, graf G_1 na obrázku 1.6 je rovinný, jistě k němu lze nalézt jiné nakreslení, které bude rovinným nakreslením).

Reprezentace grafu jeho nakreslením (obrázkem) je ale nevhodná v okamžiku, kdy graf je rozsáhlý a především tehdy, když jej chceme počítačově zpracovat.

Matematicky elegantní reprezentací je ***matice sousednosti*** (adjacency matrix), která určuje graf až na izomorfismus. Nevýhodou této reprezentace je, že v případě, že má matice relativně málo hran, může tato obsahovat velké množství nul (tzv. řídká matice).

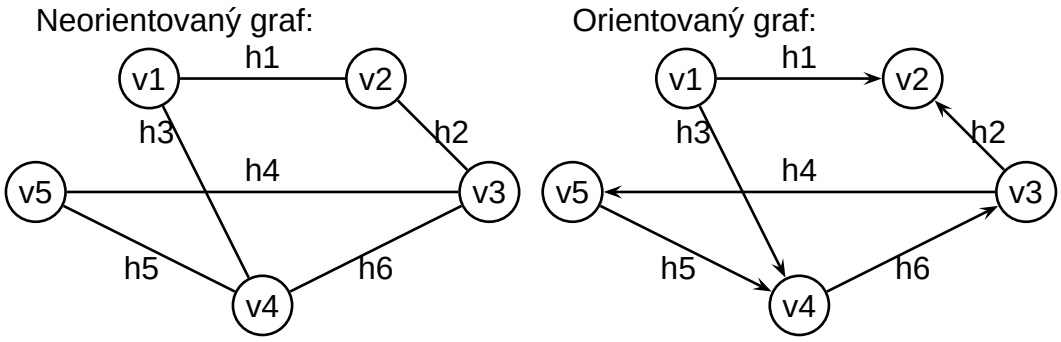
Definice 2.1 *Nechť G je graf. Zvolíme-li libovolně, ale pevně pořadí jeho vrcholů $v_1, ..., v_n$, můžeme mu přiřadit matici sousednosti takto:*

Neorientovaný graf:

$$A_{ij} = \begin{cases} 1 & v_i \; soused \; s \; v_j \\ 0 & jindy \end{cases}$$

Orientovaný graf:

$$A_{ij} = \begin{cases} 1 & v_i = PV \; a \; v_j = KV \\ 0 & jindy \end{cases}$$



Obrázek 2.1: Příklad nakreslení neorientovaného a orientovaného grafu.

■ **Příklad 2.1** *Matice sousednosti pro grafy z obrázku 2.1:*

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \qquad A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

□

Matice sousednosti neorientovaného grafu je vždy symetrická.

Mají-li dva grafy (oba orientované nebo oba neorientované) stejnou matici sousednosti, pak jsou tyto grafy navzájem izomorfní. Naopak toto tvrzení neplatí, vzájemně izomorfní grafy mohou mít různé matice sousednosti (vzhledem k jinému pořadí vrcholů).

Poznámka:
Definice je uvedená pro jednoduché grafy. Pokud bychom chtěli popsat graf s násobnými hranami maticí sousednosti, museli bychom definici mírně modifikovat (např. místo čísla 1 bychom uvedli násobnost hrany).

Dalším typem matice, kterým můžeme reprezentovat graf je ***matice incidence*** (incidence matrix). O vhodnosti jejího použití platí totéž, co pro matici sousednosti.

Definice 2.2 *Nechť G je graf. Zvolíme-li libovolně, ale pevně pořadí jeho vrcholů $v_1, ..., v_n$ a hran $h_1, ..., h_m$, můžeme mu přiřadit matici incidence (matice je typu (n, m)) takto:*

Neorientovaný graf:

$$B_{ij} = \begin{cases} 1 & v_i \; inciduje \; s \; h_j \\ 0 & jindy \end{cases}$$

Orientovaný graf:

$$B_{ij} = \begin{cases} 1 & v_i \; PV \; hrany \; h_j \\ -1 & v_i \; KV \; hrany \; h_j \\ 0 & jindy \end{cases}$$

■ **Příklad 2.2** *Matice incidence pro grafy z obrázku 2.1:*

$$B = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

□

Poznámka:
Grafy s alespoň jednou smyčkou nelze maticí incidence korektně reprezentovat.

Další možnou reprezentací grafů (vhodnou především pro počítačové zpracování) je použít **seznamy vrcholů a seznamy hran** (případně i s cenami vrcholů a /nebo hran). Zde se bude předpokládat, že jsou vrcholy a hrany grafu jednoznačně očíslovány. Možných seznamů je mnoho, záleží pro jaký typ úloh jsou určeny.

■ **Příklad 2.3** *Seznam vrcholů a jejich sousedů pro grafy z obrázku 2.1:*

Neorientovaný graf:
$v_1 \longrightarrow v_2, v_4$
$v_2 \longrightarrow v_1, v_3$
$v_3 \longrightarrow v_2, v_4, v_5$
$v_4 \longrightarrow v_1, v_3, v_5$
$v_5 \longrightarrow v_3, v_4$

Orientovaný graf:
$v_1 \longrightarrow v_2, v_4$
$v_2 \longrightarrow$
$v_3 \longrightarrow v_2, v_5$
$v_4 \longrightarrow v_3$
$v_5 \longrightarrow v_4$

□

■ **Příklad 2.4** *Seznam hran pro grafy z obrázku 2.1:*

Neorientovaný graf:
(v_1, v_2)
(v_1, v_4)
(v_2, v_3)
(v_3, v_4)
(v_3, v_5)
(v_4, v_5)

Orientovaný graf:
(v_1, v_2)
(v_1, v_4)
(v_3, v_2)
(v_3, v_5)
(v_4, v_3)
(v_4, v_4)

Poznámka:
U neorientovaného grafu jsou jednotlivé hrany neorientovány a jsou určeny svými koncovými vrcholy. U orientovaného grafu je každá hrana určena svým počátečním a koncovým uzlem

□

Pokud má být výsledkem algoritmu matice a/nebo pokud pracujeme s hranově ohodnoceným grafem prostým grafem, můžeme pro reprezentaci takovéhoto grafu zvolit tzv. matici cen (viz 4.7).

Shrnutí kapitoly 2

V této kapitole jsme si seznámili s pojmy možnostmi reprezentace grafů, a to těmito:

1. matice sousednosti, matice incidence,

2. seznamy hran, seznamy vrcholů.

Otázky ke kapitole 2

1. Daný graf G, který je zadán seznamem hran $E(G) = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_6), (v_4, v_5), (v_5, v_6), (v_6, v_4), (v_7, v_6)\}$, reprezentujte maticí sousednosti a maticí incidence a seznamem vrcholů. Na graf G se dívejte nejprve jako na neorientovaný, poté jako na orientovaný.

2. Daný orientovaný graf H, který je zadán seznamem vrcholů a jejich sousedů:
   $v_1 \longrightarrow v_2, v_3, v_6$
   $v_2 \longrightarrow v_3$
   $v_3 \longrightarrow v_5$
   $v_4 \longrightarrow v_2, v_6, v_7$
   $v_5 \longrightarrow v_6$
   $v_6 \longrightarrow v_7$
   $v_7 \longrightarrow$
   reprezentujte maticí sousednosti a maticí incidence a seznamem hran. Jak by vypadal seznam vrcholů a jejich sousedů, pokud bychom u grafu H „zapomněli“ na orientaci hran?

Kapitola 3

Prohledávání grafů

Obsah

Prohledávání grafů je systematický postup, kterým můžeme řešit např. dostupnost vrcholu nebo množiny vrcholů z daného počátečního vrcholu. Algoritmy na prohledávání grafů jsou dále rozvinuty při hledání nejkratších (nejdelších, nejlevnějších, . . .) cest. Pomocí algoritmu na prohledávání grafu jsme schopni určit jednotlivé souvislé (případně u orientovaných grafů silně souvislé) komponenty. Uvedené algoritmy jsou popsány pro případ orientovaného grafu, pro neorientovaný graf vyžadují jen mírné úpravy.

3.1 Prohledávání grafu do hloubky

Časová náročnost kapitoly: 25 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat princip algoritmu prohledávání do hloubky a na příkladu jej použít.

Tento první algoritmus si lze snadno představit jako průchod bludištěm, kdy procházíme z místnosti do místnosti po chodbách a v dosažené místnosti si vybíráme chodbu (pro postup do další místnosti), po které jsme ještě nešli. Pokud taková chodba neexistuje, musíme se z této místnosti vrátit, chodbou po které jsme do ní přišli, do předešlé místnosti. Prohledávání grafu do hloubky (depth-first search) se realizuje v těchto krocích: začneme v libovolném výchozím vrcholu v_v a přejdeme do kteréhokoliv sousedního vrcholu. Ten si označíme a testujeme, zda můžeme pokračovat v procházení grafu (tj. jestli existují sousední neoznačené vrcholy). Pokud ano, přesuneme se do dalšího vrcholu, který si také označíme. Pokud nejsou v okolí vrcholu dosud neoznačené vrcholy, vracíme se zpět a testujeme okolí dřívějších vrcholů. Tento algoritmus můžeme ukončit v různých situacích:

- když dojdeme do předem vybraného vrcholu.
- když víme, že jsme prošli již všemi vrcholy grafu.
- když se vrátíme zpět do výchozího vrcholu (prozkoumali jsme jednu souvislou komponentu).

Algoritmus:
Budeme používat pole vrcholů Uz, ve kterém bude $false$ pro neoznačené vrcholy a $true$ pro již označené vrcholy. Dále použijeme pole Z, kam budeme postupně ukládat procházené vrcholy (konkrétně jejich indexy) a zpět se po nich vracet (princip zásobníku). Poslední vrchol v zásobníku je aktuální. V celočíselné proměnné j budeme ukládat informaci o počtu vrcholů v poli Z.

1. Inicializace: $Pv_v := true$; $Pv_i := false$ pro $\forall i \in \{1, \ldots, |V(G)|\}$; $i \neq v$, kde v je index výchozího vrcholu. $Z_1 := v_v$; $j := 1$.
2. V dostupném okolí aktuálního vrcholu hledáme všechny dosud neoznačené vrcholy (tj. položka na jejich místě v poli Uz je $false$). Pokud takovýto vrchol najdeme, přejdeme k bodu 3. Pokud takovýto vrchol nenajdeme, odstraníme poslední vrchol ze zásobníku a změníme $j := j - 1$. Jestliže $j = 0$, tak jsme prošli celou komponentou grafu a přejdeme k bodu 5. Jinak pokračujeme bodem 2.

3. Jdeme do sousedního vrcholu v_k, ve kterém jsme ještě nebyli. Změníme: $j := j + 1$; $Pv_k := true$; $Z_j := v_k$.

4. Jestliže $j = |V(G)|$, tak jsme prošli všemi vrcholy grafu a přejdeme k bodu 5, jinak bod 2.

5. Konec

Složitost:
Řádová složitost algoritmu je $O(m + n)$.

Animace: Prohledávání grafu do hloubky

■ **Příklad 3.1** *V daném grafu nalezněte všechny vrcholy, které jsou dostupné z počátečního vrcholu v_1 pomocí prohledávání grafu do hloubky.*

zadaný graf:

$v_1 \longrightarrow v_2, v_3, v_4$
$v_2 \longrightarrow v_3$
$v_3 \longrightarrow v_4$
$v_4 \longrightarrow v_5, v_6$
$v_5 \longrightarrow$
$v_6 \longrightarrow v_1$

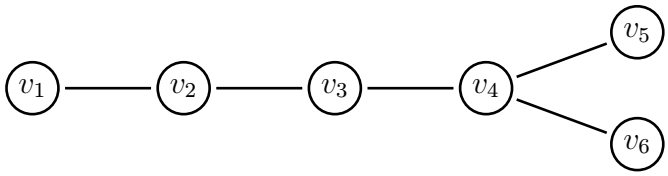
simulace algoritmu:

v_1	v_2	v_3	v_4	v_5	v_6	Z	j
-1	0	0	0	0	0	1	1
-1	1	0	0	0	0	1,2	2
-1	1	2	0	0	0	1,2,3	3
-1	1	2	3	0	0	1,2,3,4	4
-1	1	2	3	4	0	1,2,3,4,5	5
-1	1	2	3	4	0	1,2,3,4	4
-1	1	2	3	4	4	1,2,3,4,6	5

rekonstrukce cest:
v_1 je výchozí vrchol

$v_2 \longleftarrow v_1$
$v_3 \longleftarrow v_2 \longleftarrow v_1$
$v_4 \longleftarrow v_3 \longleftarrow v_2 \longleftarrow v_1$
$v_5 \longleftarrow v_4 \longleftarrow v_3 \longleftarrow v_2 \longleftarrow v_1$
$v_6 \longleftarrow v_4 \longleftarrow v_3 \longleftarrow v_2 \longleftarrow v_1$

strom dostupnosti:



□

Shrnutí kapitoly 3.1

V této kapitole jsme se seznámili s algoritmem prohledávání grafu do hloubky.

Otázky ke kapitole 3.1

1. V daném orientovaném grafu G, který je zadán seznamem hran $E(G) = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_6), (v_4, v_5), (v_5, v_6), (v_6, v_4), (v_7, v_6)\}$, nalezněte všechny vrcholy, které jsou dostupné z počátečního vrcholu v_1 pomocí algoritmu prohledávání grafu do hloubky.

2. V daném orientovaném grafu H, který je zadán seznamem vrcholů a jejich sousedů:
   $v_1 \longrightarrow v_2, v_3, v_6$
   $v_2 \longrightarrow v_3$
   $v_3 \longrightarrow v_5$
   $v_4 \longrightarrow v_2, v_6, v_7$
   $v_5 \longrightarrow v_6$
   $v_6 \longrightarrow v_7$
   $v_7 \longrightarrow$
   nalezněte všechny vrcholy, které jsou dostupné z počátečního vrcholu v_4 pomocí algoritmu prohledávání grafu do hloubky.

3.2 Prohledávání grafu do šířky

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat princip algoritmu prohledávání do šířky a na příkladu jej použít.

Odlišnost prohledávání grafu do šířky od prohledávání do hloubky spočívá v tom, že se z dosaženého vrcholu „rozhlédneme“ najednou do všech sousedních vrchulů. Tímto způsobem získáme v jednom kroku celou úroveň vrcholů, které jsou z výchozího vrcholu dostupné po stejném počtu hran.

Algoritmus:
Budeme používat pole vrcholů P_u, kam si uložíme $true$ u již označených (a tedy dosažených) vrcholů a $false$ u neoznačených vrcholů. Dále použijeme množinu F, kam budeme postupně ukládat procházené vrcholy (konkrétně jejich indexy) a zároveň je budeme z této množiny vybírat (princip fronty).

1. Inicializace: $Pv_v := true$; $Pv_i := false$ pro $\forall i \in \{1, \ldots, |V(G)|\}$; $i \neq v$, kde v je index výchozího vrcholu. $F := \{v_v\}$.
2. Vezmeme první vrchol v_k z množiny F a hledáme všechny jeho neoznačené sousedy. Pro neoznačené sousedy provedeme tyto změny: $Pv_j := true$ pro $\forall v_j \in N$, kde $N = \{\text{neoznačení sousede } v_k\}$; $F := F - \{v_k\} \cup N$.
3. Je-li $F = \emptyset$, pak přejdeme k bodu 4. Jinak jdeme k bodu 2.
4. Konec.

Poznámka:
U obou typů algoritmů můžeme využít Pv_j pro uložení předposledního vrcholu cesty, po které jsme se z výchozího vrcholu v_v do daného vrcholu v_j dostali. Nebudeme tedy ukládat $false$ a $true$, ale třeba 0 pro dosud neoznačené vrcholy a i pro označené vrcholy, kde i bude reprezentovat předposlední vrchol cesty (tj. uložíme index vrcholu, z kterého jsme přišli).

Složitost:
Řádová složitost algoritmu je $O(m + n)$.

Animace: Prohledávání grafu do šířky

■ **Příklad 3.2** *V daném grafu nalezněte všechny vrcholy, které jsou dostupné z počátečního vrcholu v_1 pomocí prohledávání grafu do šířky.*

zadaný graf:

$v_1 \longrightarrow v_2, v_3, v_4$
$v_2 \longrightarrow v_3$
$v_3 \longrightarrow v_4$
$v_4 \longrightarrow v_5, v_6$
$v_5 \longrightarrow$
$v_6 \longrightarrow v_1$

simulace algoritmu:

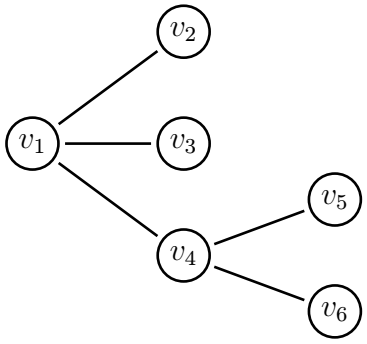
v_1	v_2	v_3	v_4	v_5	v_6	F
-1	0	0	0	0	0	1
-1	1	1	1	0	0	2,3,4
-1	1	1	1	0	0	3,4
-1	1	1	1	0	0	4
-1	1	1	1	4	4	5,6
-1	1	1	1	4	4	6
-1	1	1	1	4	4	$\emptyset$

rekonstrukce cest:

v_1 je výchozí vrchol

$v_2 \longleftarrow v_1$
$v_3 \longleftarrow v_1$
$v_4 \longleftarrow v_1$
$v_5 \longleftarrow v_4 \longleftarrow v_1$
$v_6 \longleftarrow v_4 \longleftarrow v_1$

strom dostupnosti:



□

Shrnutí kapitoly 3.2

V této kapitole jsme se seznámili s algoritmem prohledávání grafu do šířky.

Otázky ke kapitole 3.2

1. V daném orientovaném grafu G, který je zadán seznamem hran $E(G) = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_6), (v_4, v_5), (v_5, v_6), (v_6, v_4), (v_7, v_6)\}$, nalezněte všechny vrcholy, které jsou dostupné z počátečního vrcholu v_7 pomocí algoritmu prohledávání grafu do šířky.

2. V daném orientovaném grafu H, který je zadán seznamem vrcholů a jejich sousedů:

   $v_1 \longrightarrow v_2, v_3, v_6$
   $v_2 \longrightarrow v_3$
   $v_3 \longrightarrow v_5$
   $v_4 \longrightarrow v_2, v_6, v_7$
   $v_5 \longrightarrow v_6$
   $v_6 \longrightarrow v_7$
   $v_7 \longrightarrow$

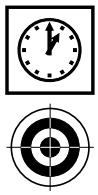
   nalezněte všechny vrcholy, které jsou dostupné z počátečního vrcholu v_1 pomocí algoritmu prohledávání grafu do šířky.

Kapitola 4

Nejkratší cesty

Obsah

4.1 Pojmy, základní algoritmus



Časová náročnost kapitoly: 25 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět vysvětlit princip určování nejkratších cest, popsat základní algoritmus pro určování nejkratších cest a na příkladech tento algoritmus použít.

Pojmy

Úkolem této skupiny grafových algoritmů je nalézt nejkratší orientovanou cestu:

- z daného vrcholu do daného vrcholu,
- z daného výchozího vrcholu do každého vrcholu grafu,
- z každého vrcholu do daného koncového vrcholu,
- mezi všemi uspořádanými dvojicemi vrcholů.

Algoritmy pro nalezení nejkratší cesty v grafu jsou ve své podstatě založeny na prohledávání grafu. Je zde přidáno podstatné kritérium při výběru hran - délka cesty (tj. součet cen všech hran, které tvoří cestu z výchozího vrcholu do koncového vrcholu).

Pokud nalezneme jakoukoliv cestu z výchozího vrcholu do libovolného dalšího vrcholu, testujeme, zda-li to není nejkratší cesta. Ověřujeme platnost ***trojúhelníkové nerovnosti***:

$$c_j > c_i + c_{ij},$$

kde c_j je cena dosud nejkratší cesty z výchozího vrcholu do vrcholu v_j, c_{ij} je cena hrany mezi vrcholy v_i a v_j. Pokud je nerovnost **splněna**, znamená to, že jsme nalezli kratší cestu do vrcholu v_j než byla stávající. Takže původní cenu změníme na nižší: $c_j := c_i + c_{ij}$. Pokud není nerovnost splněna, neměníme nic.

Vzdálenost vrcholu u od vrcholu v je délka nejkratší cesty z u do v a označuje se $dist(u, v)$. Existuje-li v grafu G cesta z vrcholu u do vrcholu v, říkáme, že vrchol v je v G dostupný z u.

Pokud je graf multigrafem, budeme pracovat pouze s hranami s nejmenší cenou (případné smyčky vynecháme). Pokud graf není ohodnocený, pak za nejkratší cestu považujeme takovou cestu, která obsahuje nejmenší počet hran (tento problém je ekvivalentní problému v grafu s jednotkovým ohodnocením hran).

Základní algoritmus

Procházíme hranami grafu a zjišťujeme, které hrany splňují trojúhelníkovou nerovnost: $c_j > c_i + c_{ij}$. Pokud je tato nerovnost splněna, přepíšeme cenu nejkratší cesty ve vrcholu v_j na $c_j := c_i + c_{ij}$. Celým seznamem hran procházíme tak dlouho, dokud dochází ke změnám v cenách cest (reprezentace pomocí seznamu hran a jejich cen se jeví výhodnější).

Algoritmus:
Budeme používat dvě pole Pu a C, jejichž prvky budou předposlední vrchol a cena nejkratší cesty do vrcholu v_i.

1. Inicializace: $Pv_i := 0$ pro $\forall i$ a $P_v := -1$. $C_v := 0$ a $C_i := \infty$ pro $\forall i \neq v$, kde v_v je výchozí vrchol.
2. Pro všechny hrany grafu provedeme: je-li $C_j > C_i + c_{ij}$, pak $\{C_j := C_i + c_{ij}$ a $Pv_j := v_i\}$
3. Jestliže během kroku 2 nedošlo k žádné změně hodnot v poli C, výpočet končí. Jinak pokračujeme krokem 2.

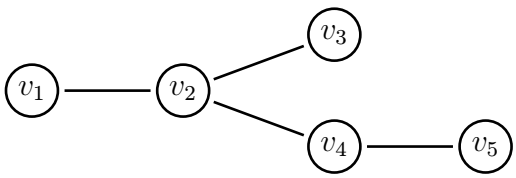
Animace: Základní algoritmus

■ **Příklad 4.1** *V ohodnoceném orientovaném grafu G_1, který je zadán následujícím seznamem hran $E(G_1) = \{(v_1, v_2, 3),\ (v_1, v_3, 5),\ (v_2, v_3, 1),\ (v_2, v_4, 2),\ (v_2, v_5, 4),\ (v_4, v_3, 9),\ (v_4, v_5, 1),\ (v_5, v_1, 2),\ (v_5, v_3, 8)\}$, určete nejkratší cesty z vrcholu v_1 základním algoritmem.*

Pv_1, C_1	Pv_2, C_2	Pv_3, C_3	Pv_4, C_4	Pv_5, C_5
$-1, 0$	$0, \infty$	$0, \infty$	$0, \infty$	$0, \infty$
$-1, 0$	$1, 3$	$0, \infty$	$0, \infty$	$0, \infty$
$-1, 0$	$1, 3$	$1, 5$	$0, \infty$	$0, \infty$
$-1, 0$	$1, 3$	$2, 4$	$0, \infty$	$0, \infty$
$-1, 0$	$1, 3$	$2, 4$	$2, 5$	$0, \infty$
$-1, 0$	$1, 3$	$2, 4$	$2, 5$	$2, 7$
$-1, 0$	$1, 3$	$2, 4$	$2, 5$	$4, 6$

Je zde zapsán jeden průchod seznamem hran. Při druhém průchodu už nedochází ke změnám a algoritmus tedy končí.

strom dostupnosti:



□

Poznámka:
Pokud nepotřebujeme znát cestu z výchozího vrcholu do ostatních dostupných vrcholů (viz. prohledávání grafů), nemusíme používat pole Pu. Tj. stačí jen pole, které reprezentuje ceny nejkratších cest.

Poznámka:
Pokud budeme pracovat s multigrafem (má násobné hrany), pak nestačí určení hrany pomocí koncových vrcholů. Musíme uvést, po které hraně jsme se do daného vrcholu dostali. Tj. v poli Pu nebudeme uvádět předposlední vrchol cesty, ale poslední hranu v cestě.

Shrnutí kapitoly 4.1

V této kapitole jsme se seznámili s principem určování nejkratších cest na základě trojúhelníkové nerovnosti. Dále jsme se seznámili se základním algoritmem pro určování nejkratších cest.

Otázky ke kapitole 4.1

1. V ohodnoceném orientovaném grafu G_2, který je zadán následujícím seznamem hran
   $h_1 \longrightarrow (v_1, v_2, 5)$
   $h_2 \longrightarrow (v_1, v_3, 1)$
   $h_3 \longrightarrow (v_1, v_6, 3)$
   $h_4 \longrightarrow (v_2, v_4, 2)$
   $h_5 \longrightarrow (v_2, v_5, 4)$
   $h_6 \longrightarrow (v_3, v_4, 1)$
   $h_7 \longrightarrow (v_3, v_6, 2)$
   $h_8 \longrightarrow (v_4, v_5, 6)$
   určete všechny nejkratší cesty z vrcholu v_1 pomocí základního algoritmu.

4.2 Zlepšený algoritmus

Časová náročnost kapitoly: 15 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat princip zlepšeného algoritmu pro určování nejkratších cest a na příkladu jej použít.

Vyjdeme nyní z algoritmu prohledávání grafu do šířky (jistě jste si povšimli, že předchozí algoritmus byl velmi podobný prohledávání grafu do hloubky). Budeme si pamatovat již dosažené vrcholy a ceny cest z výchozího vrcholu do již dosažených vrcholů. V každém kroku si z dosažených vrcholů vybereme vrchol s nejmenší cenou. Z tohoto vrcholu se pak budeme „rozhlížet“ po sousedních dostupných vrcholech. Tímto způsobem se strom dostupnosti postupně rozšiřuje (případně jen modifikuje) o hrany nejkratších cest.

Algoritmus:
Použijeme dvě pole Pu a C, do nichž budeme pro každý vrchol v_i ukládat předposlední vrchol a cenu nejkratší cesty z výchozího vrcholu do v_i. Navíc použijeme množinu M, v níž budou vrcholy, do kterých jsme v posledním kroku nalezli nejlevnější cestu.

1. Inicializace: $Pv_i := 0$, $C_v := 0$ a $C_i := \infty$ pro $\forall i \neq v$, $Pv_v := -1$, kde v_v je výchozí vrchol. $v_v \rightarrow M$ (vrchol v_v jsme přidali do množiny M).
2. Je-li $M = \emptyset$, potom 4, jinak odebereme z M vrchol v_x (takový, že $C_x = min\{C_k; v_k \in M\}$
3. Pro všechny sousedy vrcholu v_x provedeme (ozn. v_y sousedí s v_x): je-li $C_y > C_x + c_{xy}$, pak $\{C_y := C_x + c_{xy}$, $Pv_y := v_x$ a jestliže $v_y \notin M$, potom $v_y \rightarrow M\}$. Pokračujeme krokem 2.
4. Konec.

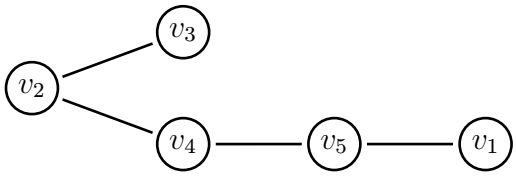
Animace: Zlepšený algoritmus

■ **Příklad 4.2** *V ohodnoceném orientovaném grafu G_1, který je zadán následujícím seznamem hran $E(G_1) = \{(v_1, v_2, 3), (v_1, v_3, 5), (v_2, v_3, 1), (v_2, v_4, 2), (v_2, v_5, 4), (v_4, v_3, 9), (v_4, v_5, 1), (v_5, v_1, 2), (v_5, v_3, 8)\}$, určete nejkratší cesty z vrcholu z vrcholu v_2 zlepšeným algoritmem.*

Pv_1, C_1	Pv_2, C_2	Pv_3, C_3	Pv_4, C_4	Pv_5, C_5	M
$0, \infty$	$-1, 0$	$0, \infty$	$0, \infty$	$0, \infty$	*2*
$0, \infty$	$-1, 0$	$2, 1$	$2, 2$	$2, 4$	*3,4,5*
$0, \infty$	$-1, 0$	$2, 1$	$2, 2$	$2, 4$	*4,5*
$0, \infty$	$-1, 0$	$2, 1$	$2, 2$	$4, 3$	*5*
$5, 5$	$-1, 0$	$2, 1$	$2, 2$	$4, 3$	*1*

V poli M jsou uloženy vrcholy, do kterých jsme již nalezli nejkratší cestu. Můžeme je vybírat postupně (jako z fronty) nebo preferovat vrchol s nejnižší cenou (použito v Dijkstrově algoritmu).

strom dostupnosti:



□

Shrnutí kapitoly 4.2

V této kapitole jsme se seznámili se zlepšeným algoritmem pro určování nejkratších cest.

Otázky ke kapitole 4.2

1. V ohodnoceném orientovaném grafu G_2, který je zadán následujícím seznamem vrcholů a jejich okolí s cenami jednotlivých hran
   $v_1 \longrightarrow (v_2, 2), (v_6, 1)$
   $v_2 \longrightarrow (v_1, 4), (v_7, 1)$
   $v_3 \longrightarrow (v_6, 6), (v_7, 3)$
   $v_4 \longrightarrow (v_3, 1), (v_6, -3)$
   $v5 \longrightarrow (v_2, 6), (v_7, 4)$
   $v_6 \longrightarrow (v_3, 7), (v_5, -1)$
   $v_7 \longrightarrow (v_1, 2), (v_6, 1)$
   nalezněte zlepšeným algoritmem nejkratší cesty z vrcholu v_1.

4.3 Dijkstrův algoritmus

Časová náročnost kapitoly: 15 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět vysvětlit princip Dijkstrova algoritmu pro určování nejkratších cest a na příkladu jej použít.

V případě, že hledáme nejkratší cesty v grafu s nezápornými cenami hran, můžeme použít Dijkstrův algoritmus. Je to prakticky předchozí algoritmus. Klade se v něm důraz na vybrání vrcholu, který má co nejlevnější cenu nejkratší cesty z výchozího vrcholu. V dalších krocích algoritmu nemusíme v tomto vrcholu testovat trojúhelníkovou nerovnost, protože se cena nejlevnější cesty již nemůže změnit (díky nezápornosti cen hran).

Algoritmus:
Použijeme dvě pole Pu a C, do nichž budeme pro každý vrchol v_i ukládat předposlední vrchol a cenu nejkratší cesty z výchozího vrcholu do v_i. Navíc použijeme dvě pomocné množiny S a S', v níchž budou dosažené a dosud nedosažené vrcholy.

1. Inicializace: $Pv_v := -1$,$Pv_i := 0$ pro $\forall i$. $C_v := 0$ a $C_i := \infty$ pro $\forall i \neq v$, kde v_v je výchozí vrchol. S je prázdné pole, $S' = V(G)$.
2. Najdeme vrchol $v_x \in S'$ takový, že $C_x = min\{C_k; v_k \in S\}$. Jestliže $c_x = \infty$, potom bod 4. Jinak změníme: $\{v_x \rightarrow S$ a $S' := S' - \{v_x\}\}$. Jestliže je nyní $S' = \emptyset$, potom bod 4.
3. Pro každý vrchol $v_i \in S'$ a zároveň incidentní s uzlem v_x provedeme: je-li $C_i > C_x + c_{xi}$, pak $\{C_i := C_x + c_{xi}$ a $Pv_i := v_x\}$. Po prozkoumání všech výše specifikovaných vrcholů přejdeme k bodu 2.
4. Konec.

Složitost:
Řádová složitost algoritmu je $O((m+n)log(n))$.

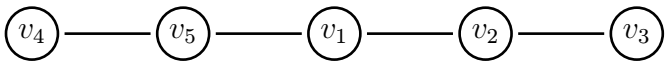
Animace: Dijkstrův algoritmus

■ **Příklad 4.3** *V ohodnoceném orientovaném grafu G_1, který je zadán následujícím seznamem hran $E(G_1) = \{(v_1, v_2, 3),\ (v_1, v_3, 5),\ (v_2, v_3, 1),\ (v_2, v_4, 2),\ (v_2, v_5, 4),\ (v_4, v_3, 9),\ (v_4, v_5, 1),\ (v_5, v_1, 2),\ (v_5, v_3, 8)\}$, určete nejkratší cesty z vrcholu v_4 Dijkstrovým algoritmem.*

Pv_1, C_1	Pv_2, C_2	Pv_3, C_3	Pv_4, C_4	Pv_5, C_5	S	S'
$0, \infty$	$0, \infty$	$0, \infty$	$-1, 0$	$0, \infty$	$\emptyset$	$1, 2, 3, 4, 5$
$0, \infty$	$0, \infty$	$4, 9$	$-1, 0$	$4, 1$	*4*	$1, 2, 3, 5$
$5, 3$	$0, \infty$	$4, 9$	$-1, 0$	$4, 1$	*4,5*	$1, 2, 3$
$5, 3$	$1, 6$	$1, 8$	$-1, 0$	$4, 1$	*4,5,1*	$2, 3$
$5, 3$	$1, 6$	$2, 7$	$-1, 0$	$4, 1$	*4,5,1,2*	3
$5, 3$	$1, 6$	$2, 7$	$-1, 0$	$4, 1$	*4,5,1,2,3*	$\emptyset$

V poli S jsou uloženy vrcholy, do kterých jsme již nalezli nejkratší cestu. Z pole S' vybíráme vrchol s nejnižší cenou. Trojúhelníkovou nerovnost testujeme jen mezi vybraným vrcholem a vrcholy z S', protože díky nezáporným cenám nemůže dojít ke změně ceny u vrcholů z pole S.

strom dostupnosti:



□

Shrnutí kapitoly 4.3

V této kapitole jsme se seznámili s Dijkstrovým algoritmem pro určování nejkratších cest.

Otázky ke kapitole 4.3

1. V ohodnoceném orientovaném grafu G_2, který je zadán následujícím seznamem vrcholů a jejich okolí s cenami jednotlivých hran
   $v_1 \longrightarrow (v_2, 2), (v_6, 1)$
   $v_2 \longrightarrow (v_1, 4), (v_7, 1)$
   $v_3 \longrightarrow (v_6, 6), (v_7, 3)$
   $v_4 \longrightarrow (v_3, 1), (v_6, -3)$
   $v5 \longrightarrow (v_2, 6), (v_7, 4)$
   $v_6 \longrightarrow (v_3, 7), (v_5, -1)$
   $v_7 \longrightarrow (v_1, 2), (v_6, 1)$
   nalezněte Dijkstrovým algoritmem nejkratší cesty z vrcholu v_3.

4.4 Bellman-Fordův algoritmus

Časová náročnost kapitoly: 15 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat princip Bellman-Fordova algoritmu pro určování nejkratších cest v grafu a na příkladu jej použít.

Tento algoritmus je v principu podobný dvěma předchozím algoritmům. Má navíc pole, ve kterém je u každého vrcholu uvedeno, kolik hran obsahuje nejkratší cesta z výchozího vrcholu do daného vrcholu. Tento algoritmus pak preferuje nejkratší cestu s nejmenším počtem hran.

Protože víme, že v grafu o n vrcholech obsahuje nejdelší cesta maximálně $(n-1)$ hran, můžeme tohoto algoritmu také využít pro detekci cyklu se zápornou cenou (pokud algoritmus korektně neskončí po $n-1$ krocích je v grafu cyklus se zápornou cenou).

Algoritmus:
Použijeme tři pole P, Pu a C, do nichž budeme pro každý vrchol v_i ukládat počet hran, předposlední vrchol a cenu nejkratší cesty z výchozího vrcholu do v_i. Navíc použijeme proměnnou k, ve které bude aktuální délka nejkratších cest.

1. Inicializace: $P_v := 0$, $P_i := -1$, $Pv_v := -1$ a $Pv_i := 0$ pro $\forall i$. $C_v := 0$ a $C_i := \infty$ pro $\forall i \neq v$, kde v_v je výchozí vrchol; $k := 0$.

2. k-tá iterace:
   Jestliže ($k = n-1$) nebo ($P_j \neq k$ pro $\forall j$), potom bod 3. Jinak, pro $\forall v_j$ takové, že $P_j = k$, provedeme: pro $\forall v_i$ takové, že v_i je incidentní s v_j testujeme - je-li $C_i > C_j + c_{ji}$, potom $\{C_i := C_j + c_{ji}$, $Pv_i := v_j$, $P_i := P_j + 1\}$; $k := k+1$ a přejdeme k bodu 2

3. Konec

Složitost:
Řádová složitost algoritmu je $O(m * n)$.

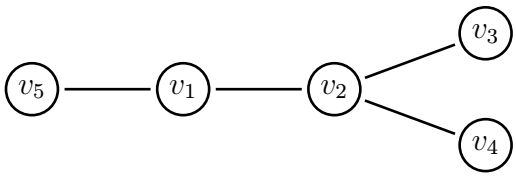
Animace: Bellman-Fordův algoritmus

■ **Příklad 4.4** *V ohodnoceném orientovaném grafu G_1, který je zadán následujícím seznamem hran $E(G_1) = \{(v_1, v_2, 3),\ (v_1, v_3, 5),\ (v_2, v_3, 1),\ (v_2, v_4, 2),\ (v_2, v_5, 4),\ (v_4, v_3, 9),\ (v_4, v_5, 1),\ (v_5, v_1, 2),\ (v_5, v_3, 8)\}$, určete nejkratší cesty z vrcholu v_5 Bellman-Fordovým algoritmem.*

P_1, Pv_1, C_1	P_2, Pv_2, C_2	P_3, Pv_3, C_3	P_4, Pv_4, C_4	P_5, Pv_5, C_5	k
$-1, 0, \infty$	$-1, 0, \infty$	$-1, 0, \infty$	$-1, 0, \infty$	$0, -1, 0$	0
$1, 5, 2$	$-1, 0, \infty$	$1, 5, 8$	$-1, 0, \infty$	$0, -1, 0$	1
$1, 5, 2$	$2, 1, 5$	$2, 1, 7$	$-1, 0, \infty$	$0, -1, 0$	2
$1, 5, 2$	$2, 1, 5$	$3, 2, 6$	$3, 2, 7$	$0, -1, 0$	3
$1, 5, 2$	$2, 1, 5$	$3, 2, 6$	$3, 2, 7$	$0, -1, 0$	4

V jednom kroku algoritmu, se prozkoumávají všechny vrcholy, pro které platí: $P_i = k$. Algoritmus končí, pokud takový vrchol neexistuje, případně pro $k = n - 1$. V našem případě byl algoritmus ukončen druhou podmínkou (a už se nemuselo ověřovat, jestli existuje vrchol, do kterého vede cesta délky $(n - 1)$ - ten také neexistuje).

strom dostupnosti:



□

Shrnutí kapitoly 4.4

V této kapitole jsme se seznámili s Bellman-Fordovým algoritmem pro určování nejkratších cest.

Otázky ke kapitole 4.4

1. V ohodnoceném orientovaném grafu G_2, který je zadán následujícím seznamem vrcholů a jejich okolí s cenami jednotlivých hran nalezněte nejkratší cesty z vrcholu v_1 pomocí Bellman-Fordova algoritmu.
   $v_1 \longrightarrow (v_2, 8), (v_6, 3)$
   $v_2 \longrightarrow (v_3, 3)$
   $v_3 \longrightarrow (v_1, 1), (v_4, -3)$
   $v_4 \longrightarrow (v_5, 6)$
   $v_5 \longrightarrow (v_2, -5)$
   $v_6 \longrightarrow (v_3, 2), (v_5, -2)$

4.5 Topologické uspořádání vrcholů a hran

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět definovat pojem toplogického uspořádání a v příslušném grafu jej určit.

Topologické uspořádání vrcholů grafu G je taková posloupnost vrcholů, že pro $\forall h \in E(G)$, $v_i = PV(h)$ a $v_j = KV(h)$, pak $i < j$. Tj. počáteční vrchol kterékoliv hrany předchází v topologickém uspořádání koncovému vrcholu této hrany.

Topologické uspořádání hran grafu G je taková posloupnost hran, že pro $\forall v \in V(G)$, $v = KV(h_i)$ a $v = PV(h_j)$, pak $i < j$. Tj. pro každý vrchol v platí, že všechny hrany, které přicházejí do v, předcházejí hranám, které z v vycházejí.

Topologického uspořádání vrcholů (případně hran) pak můžeme použít při hledání nejkratších cest. Víme, že z vrcholu s vyšším indexem neexistuje cesta do vrcholu s nižším indexem - můžeme eliminovat počet trojúhelníkových nerovností a případně můžeme ihned rozhodnot o nedostupnosti požadovaného vrcholu.

Algoritmus:
Budeme používat pole Vs, ve kterém budou vstupní stupně jednotlivých vrcholů. Dále budeme používat pole Pv a Ph pro posloupnost vrcholů a hran. Do pole M budeme ukládat vrcholy s nulovým vstupním stupněm.

1. Výpočet vstupních stupňů:
   - inicializace: pro $\forall v_i \in V(G)$; $Vs_i := 0$
   - vstupní stupně - pro $\forall h \in E(G)$; $h = (v_k, v_l)$; $Vs_l := Vs_l + 1$
2. Určíme všechny vrcholy s nulovým vstupním stupněm a přidáme je do M ($M = \{v_j; Vs_j = 0\}$).
3. Je-li $M = \emptyset$, potom bod 5. Jinak vezmeme z M libovolný vrchol v_y a zařadíme jej na konec posloupnosti vrcholů Pv.
4. Pro $\forall h \in E(G)$ takovou, že $h = (v_y, v_z)$ (tj. v_y je počáteční vrchol hrany h) provedeme: zařadíme h na konec posloupnosti hran Ph, položíme $Vs_z := Vs_z - 1$ a je-li $Vs_z = 0$, pak zařadíme v_z do množiny M. Pokračujeme bodem 3
5. Konec.

Složitost:
Řádová složitost algoritmu je $O(m+n)$.

Poznámka:
Pokud není graf acyklický, pak algoritmus skončí, aniž by do pole Pv uložil všechny vrcholy. Tímto způsobem, můžeme tedy určit i acykličnost (či cykličnost) grafu.

Animace: Topologické uspořádání

■ **Příklad 4.5** *V ohodnoceném oreintovaném grafu G_1, který je zadán následujícím seznamem hran $h_1 = (v_1, v_2, 3)$, $h_2 = (v_1, v_3, -5)$, $h_3 = (v_2, v_3, 1)$, $h_4 = (v_2, v_4, -2)$, $h_5 = (v_2, v_5, 4)$, $h_6 = (v_4, v_3, 9)$, $h_7 = (v_4, v_5, 1)$, $h_8 = (v_5, v_3, -8)$, určete topologické uspořádání vrcholů a hran.*

Vs_1	Vs_2	Vs_3	Vs_4	Vs_5	M	Pv	Ph
0	1	4	1	2	1	$\emptyset$	$\emptyset$
0	0	3	1	2	2	1	h_1, h_2
0	0	2	0	1	4	2	h_3, h_4, h_5
0	0	1	0	0	5	4	h_6, h_7
0	0	0	0	0	3	5	h_8
0	0	0	0	0	$\emptyset$	3	

V poli M jsou umístěny všechny vrcholy s právě dosaženým nulovým vstupním stupněm. Z pole M se vybere libovolný prvek (v tomto příkladu je to právě jeden, ale není to pravidlem), přesune se do pole Pv uspořádaných vrcholů a všechny hrany s ním incidentní se přidají do pole Ph uspořádaných hran.
□

Shrnutí kapitoly 4.5

V této kapitole jsme se seznámili s topologickým uspořádáním vrcholů a hran grafu a s jeho určováním.

Otázky ke kapitole 4.5

1. V ohodnoceném oreintovaném grafu G_2, který je zadán následujícím seznamem
   $v_1 \longrightarrow (v_2, 2), (v_6, 1)$
   $v_2 \longrightarrow (v_1, 4), (v_7, 1)$
   $v_3 \longrightarrow (v_6, 6), (v_7, 3)$
   $v_4 \longrightarrow (v_3, 1), (v_6, -3)$
   $v5 \longrightarrow (v_2, 6), (v_7, 4)$
   $v_6 \longrightarrow (v_3, 7), (v_5, -1)$
   $v_7 \longrightarrow (v_1, 2), (v_6, 1)$
   nalezněte topologické uspořádaní vrcholů a hran.

4.6 Nejkratší cesty v acyklických grafech

Časová náročnost kapitoly: 15 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět použít modifikovaný algoritmus pro určování nejkratších cest na základě topologického uspořádání v případě acyklických grafů.

Počet hran, pro které testujeme platnost trojúhelníkové nerovnosti, můžeme v acyklickém grafu (tj. v grafu bez cyklů) ještě snížit (v rámci urychlování algoritmů pro nalezení nejkratší cesty). Podmínkou však je, že známe topologické očíslování vrcholů a hran grafu. Použijeme-li základní algoritmus, pak stačí probírat hrany v topologickém uspořádání. Pokud použijeme zlepšený algoritmus a zpracováváme-li hrany v topologickém uspořádání, ukončíme výpočet již po prvním průchodu seznamem hran.

Neznáme-li topologické uspořádání vrcholů a víme-li, že daný graf je acyklický, pak upraveným algoritmem pro topologické uspořádání můžeme získat ceny nejkratších cest z výchozího vrcholu do všech dostupných vrcholů.

Algoritmus:
Budeme používat pole Vs, ve kterém budou vstupní stupně jednotlivých vrcholů. Dále budeme používat pole Pv a Ph pro posloupnost vrcholů a hran. Do pole M budeme ukládat vrcholy s nulovým vstupním stupněm. Jako v ostatních algoritmech pro nalezení nejkratší cesty budeme mít v poli C ceny nejkratších cest a v poli Pu předposlední vrchol nejkratší cesty k danému vrcholu.

1. Výpočet vstupních stupňů:
   - inicializace: pro $\forall i$; $Vs_i := 0$; $Pv_i := 0$; $Pv_v = -1$ a $C_i := \infty$ pro $i \neq v$; $C_v := 0$, kde v_v je výchozí vrchol.
   - vstupní stupně - pro $\forall h \in E(G)$; $h = (v_k, v_l)$; $Vs_l := Vs_l + 1$
2. Určíme všechny vrcholy s nulovým vstupním stupněm a přidáme je do M ($M = \{v_j; Vs_j = 0\}$).
3. Je-li $M = \emptyset$, potom bod 5. Jinak vezmeme z M libovolný vrchol v_y a zařadíme jej na konec posloupnosti vrcholů Pv.

4. Pro $\forall h \in E(G)$ takovou, že $h = (v_y, v_z)$ (tj. v_y je počáteční vrchol hrany h) provedeme: zařadíme h na konec posloupnosti hran Ph, položíme $Vs_z := Vs_z - 1$ a je-li $Vs_z = 0$, pak zařadíme v_z do množiny M. Jestliže $c_z > c_y + c_{yz}$, pak $\{c_z := c_y + c_{yz}$ a $Pv_z := v_y\}$. Pokračujeme bodem 3.

5. Konec.

Složitost:
Řádová složitost algoritmu je $O(m + n)$.

Animace: Nejkratší cesty v acyklických grafech

■ **Příklad 4.6** *V ohodnoceném oreintovaném grafu G_2, který je zadán následujícím seznamem hran $h_1 = (v_1, v_2, 3)$, $h_2 = (v_1, v_3, -5)$, $h_3 = (v_2, v_3, 1)$, $h_4 = (v_2, v_4, -2)$, $h_5 = (v_2, v_5, 4)$, $h_6 = (v_4, v_3, 9)$, $h_7 = (v_4, v_5, 1)$, $h_8 = (v_5, v_3, -8)$, určete topologické uspořádaní vrcholů a hran. Modifikujte algoritmus topologického uspořádání na vyhledání nejkratších cest z výchozího vrcholu v acyklickém grafu.*

Vs_1, Pv_1, C_1	Vs_2, Pv_2, C_2	Vs_3, Pv_3, C_3	Vs_4, Pv_4, C_4	Vs_5, Pv_5, C_5	M
$0, 0, \infty$	$1, 0, \infty$	$4, 0, \infty$	$1, -1, 0$	$2, 0, \infty$	1
$0, 0, \infty$	$0, 0, \infty$	$3, 0, \infty$	$1, -1, 0$	$2, 0, \infty$	2
$0, 0, \infty$	$0, 0, \infty$	$2, 0, \infty$	$0, -1, 0$	$1, 0, \infty$	4
$0, 0, \infty$	$0, 0, \infty$	$1, 4, 9$	$0, -1, 0$	$0, 4, 1$	5
$0, 0, \infty$	$0, 0, \infty$	$0, 5, -7$	$0, -1, 0$	$0, 4, 1$	3
$0, 0, \infty$	$0, 0, \infty$	$0, 5, -7$	$0, -1, 0$	$0, 4, 1$	$\emptyset$

Algoritmus se zjednoduší tím, že nemusíme testovat trojúhelníkovou nerovnost pro hrany, které předcházejí výchozímu vrcholu. Jestliže byl výchozím vrchol v_4, pak testování trojúhelníkové nerovnosti začlo až pro první hranu incidentní s v_4. Pokud by byl výchozí vrchol v_1, pak bychom museli testovat všechny hrany. Pokud by byl výchozí vrchol v_3, tak nemusíme testovat trojúhelníkovou nerovnost pro žádnou hranu.
□

Shrnutí kapitoly 4.6

V této kapitole kapitoly jsme se seznámili s tím, jak pracuje modifikovaný algoritmus pro určování nejkratších cest na základě topologického uspořádání v případě acyklických grafů.

Otázky ke kapitole 4.6

1. V ohodnoceném oreintovaném grafu G_2, který je zadán následujícím seznamem
   $v_1 \longrightarrow (v_3, 4)$
   $v_2 \longrightarrow (v_3, 5), (v_5, 1), (v_6, -5)$
   $v_3 \longrightarrow$
   $v_4 \longrightarrow (v_3, 6)$
   $v_5 \longrightarrow (v_1, 3), (v_4, 2)$
   $v_6 \longrightarrow (v_1, -4)$
   nalezněte nejkratší cesty z výchozího vrcholu v_5 v acyklickém grafu.

4.7 Floydův algoritmus

Časová náročnost kapitoly: 15 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět vysvětlit princip Floydova algoritmu pro určení vzdálenosti mezi vybraným vrcholem grafu a všemi ostatními vrcholy daného grafu.

Tento algoritmus převádí matici cen na matici Vzdáleností. Tj. výsledkem je matice, ve které jsou uvedeny ceny nejkratších cest z libovolného vrcholu do všech ostatních. Výstupem tohoto algoritmu nemůže být posloupnost vrcholů (či vrcholů a hran), která by popisovala nejkratší cestu. Ale pomocí tohoto algoritmu jsme schopni rozhodnout, zda-li v grafu existuje cyklus se zápornou cenou.

Algoritmus:
Použijeme ***matici cen*** $C = \{c_{ij}\}_{i,j=1}^{n,n}$, ve které c_{ij} jsou ceny hran, n je počet vrcholů a na diagonále jsou nuly. Pokud neexistuje hrana mezi dvojicí vrcholů, pak je místo její ceny uvedeno ∞. Algoritmus pro každou dvojici vrcholů v_i, v_j hledá nejkratší (cenově nejlevnější) cestu vedoucí přes vrchol v_k.

Pro $k = 1, 2, \ldots n$ provedeme
  Pro $i = 1, 2, \ldots n$ provedeme
    Pro $j = 1, 2, \ldots n$ provedeme
    je-li $c_{ij} > c_{ik} + c_{kj}$, pak $c_{ij} := c_{ik} + c_{kj}$

Složitost:
Řádová složitost algoritmu je $O(n^3)$.

Poznámka:
Objeví-li se v průběhu algoritmu na diagonále číslo menší než nula, pak daný graf obsahuje cyklus se zápornou cenou. Algoritmus použil hrany cyklu se zápornou cenou a nalezl kratší cestu pro některý z vrcholů tohoto cyklu. Nemá smyslu pokračovat v algoritmu, protože výsledky budou bezvýznamné.

Animace: Floydův algoritmus

■ **Příklad 4.7** *V ohodnoceném orientovaném grafu G_1, který je zadán následujícím seznamem hran $E(G_1) = \{(v_1, v_2, 3),\ (v_1, v_3, 5),\ (v_2, v_3, 1),\ (v_2, v_4, 2),\ (v_2, v_5, 4),\ (v_4, v_3, 9),\ (v_4, v_5, 1),\ (v_5, v_1, 2),\ (v_5, v_3, 8)\}$, určete ceny všech nejkratších cest ze všech vrcholů Floydovým algoritmem.*

Transformované matice, které vzniknout v průběhu Floydova algoritmu:

$$C^0 = \left[\begin{array}{c|cccc} 0 & 3 & 5 & x & x \\ \hline x & 0 & 1 & 2 & 4 \\ x & x & 0 & x & x \\ x & x & 9 & 0 & 1 \\ 2 & x & 8 & x & 0 \end{array}\right] \quad C^1 = \left[\begin{array}{c|cccc} 0 & 3 & 5 & x & x \\ \hline x & 0 & 1 & 2 & 4 \\ x & x & 0 & x & x \\ x & x & 9 & 0 & 1 \\ 2 & 5 & 7 & x & 0 \end{array}\right] \quad C^2 = \left[\begin{array}{cc|ccc} 0 & 3 & 4 & 5 & 7 \\ x & 0 & 1 & 2 & 4 \\ \hline x & x & 0 & x & x \\ \hline x & x & 9 & 0 & 1 \\ 2 & 5 & 6 & 7 & 0 \end{array}\right]$$

$$C^3 = \left[\begin{array}{ccc|cc} 0 & 3 & 4 & 5 & 7 \\ x & 0 & 1 & 2 & 4 \\ x & x & 0 & x & x \\ \hline x & x & 9 & 0 & 1 \\ \hline 2 & 5 & 6 & 7 & 0 \end{array}\right] \quad C^4 = \left[\begin{array}{cccc|c} 0 & 3 & 4 & 5 & 6 \\ x & 0 & 1 & 2 & 3 \\ x & x & 0 & x & x \\ x & x & 9 & 0 & 1 \\ \hline 2 & 5 & 6 & 7 & 0 \end{array}\right] \quad C^5 = \left[\begin{array}{ccccc} 0 & 3 & 4 & 5 & 6 \\ 5 & 0 & 1 & 2 & 3 \\ x & x & 0 & x & x \\ 3 & 6 & 7 & 0 & 1 \\ 2 & 5 & 6 & 7 & 0 \end{array}\right]$$

Matice C^0 je matice cen. Postupně jsou konstruovány matice, ve kterých jsou uvedeny nejkratší cesty vedoucí přes vrchol $1, 2, \ldots$. V matici C^5 jsou pak výsledné ceny nejkratší cest ze všech vrcholů.

□

Shrnutí kapitoly 4.7

V této kapitole jsme se seznámili s Floydovým algoritmem pro určení vzdálenosti mezi vybraným vrcholem grafu a všemi ostatními vrcholy daného grafu.

Otázky ke kapitole 4.7

1. V ohodnoceném oreintovaném grafu G_2, který je zadán následujícím seznamem
   $v_1 \longrightarrow (v_3, 4)$
   $v_2 \longrightarrow (v_3, 5), (v_5, 1), (v_6, -5)$
   $v_3 \longrightarrow$
   $v_4 \longrightarrow (v_3, 6)$
   $v_5 \longrightarrow (v_1, 3), (v_4, 2)$
   $v_6 \longrightarrow (v_1, -4)$
   nalezněte nejkratší cesty mezi všemi dvojicemi vrcholů.

4.8 Nejdelší cesty v grafech

Časová náročnost kapitoly: 10 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět vysvětlit jak modifikovat algoritmy pro určování nejkratších cest tak, abychom je mohli použít pro určování nejdelších cest.

Předchozí algoritmy řešily problém nejkratších cest. Pokud bychom ale chtěli řešit problém opačný - tedy nalézt v grafu cestu nejdelší - musíme provést určité modifikace.

1. Můžeme změnit znaménka u všech cen hran grafu a použít algoritmus pro nalezení nejkratší cesty. Cesty, které budou ve změněném grafu nejkratší, jsou v původním grafu cestami nejdelšími.
2. Nebo můžeme obrátit nerovnost v testování trojúhelníkové nerovnosti. V závislosti na této změně musíme v rámci inicializace nastavit ceny cest $C_i := -\infty$ pro $\forall i \neq v$, kde v_v je výchozí vrchol.

Poznámka:
Problém cyklu se zápornou cenou u nejkratších cest se u nejdelších cest mění na problém cyklu s kladnou cenou. Tuto překážku obejdeme tak, že ceny všech hran snížíme o cenu hrany s maximální cenou c_{max}. Potom je výhodné použít Bellman-Fordův algoritmus, který nám poskytne informaci o počtu hran cesty (samozřejmě musíme změnit i podmínku u Bellman-Fordova algoritmu, která preferuje menší počet hran). K ceně nejdražší cesty pak přičteme příslušný násobek c_{max}.

Shrnutí kapitoly 4.8

V této kapitole jsme se seznámili s tím, jak modifikovat algoritmy pro určování nejkratších cest tak, aby je bylo možno použít pro určování nejdelších cest.

Otázky ke kapitole 4.8

1. Vyberte některý s dříve uvedených algoritmů a použijte jej pro určení nejdelších cesty z vrcholu v_1 v grafu $E(G_1) = \{(v_1, v_2, 3),\ (v_1, v_3, 5),\ (v_2, v_3, 1),\ (v_2, v_4, 2),\ (v_2, v_5, 4),\ (v_4, v_3, 9),\ (v_4, v_5, 1),\ (v_5, v_1, 2),\ (v_5, v_3, 8)\}$.

Kapitola 5

Souvislost

Obsah

V této kapitole budou použity většinou již dříve uvedené algoritmy. Pomocí známých algoritmů si upřesníme způsoby určení souvislých komponent (v případě neorientovaných grafů) a silně souvislých komponent (v případě orientovaných grafů). Pro zkrácení zápisu budeme uvádět jen souvislé komponenty.

5.1 Určení souvislých komponent

Časová náročnost kapitoly: 30 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět zjistit, zda je daný graf souvislý (resp. silně souvislý) a určit jeho komponenty souvislosti.

Algoritmus:
Použijeme pole Kv, kde si pro každý vrchol v uložíme vrchol, který je s ním v souvislé komponentě.

1. $Kv_i := 0$ pro $\forall i$
2. Pro $\forall v_i \in G$ zjistíme dostupné vrcholy.
3. Je-li $Kv_i \neq 0$ pro $\forall i$, potom bod 4. Jinak vybereme libovolný vrchol v_x takový, že $Kv_x = 0$. Změníme $Kv_x := x$. Pro všechny jeho dostupné sousedy testujeme: pokud je ze sousedního vrcholu v_y dostupný vrchol v_x, jsou tyto vrcholy v jedné souvislé komponentě. Změníme $Kv_y := x$. Pokračujeme bodem 3
4. Konec.

Jak získat vrcholy dostupné z libovolného vrcholu:

1. pomocí prohledávání grafů - ve výše uvedeném algoritmu v bodu 2 použijeme pro zjištění dostupných vrcholů libovolné algoritmy na prohledávání grafů (prohledávání do hloubky nebo do šířky),
2. pomocí matice vzdáleností - matice vzdáleností, kterou získáme Floydovým algoritmem z matici cen nám také poskytuje úplnou informaci o dostupnosti vrcholů z libovolného vrcholu. Je zřejmé, že pokud je matice vzdáleností bez „nekonečna“ (tj. neexistuje nedostupný vrchol z libovolného vrcholu), pak je neorientovaný graf souvislý a orientovaný graf silně souvislý. V ostatních případech musíme rozlišit jednotlivé komponenty souvislosti podle toho, mezi kterými vrcholy existují cesty.
3. pomocí mocnin matice sousednosti - k-tá mocnina matice sousednosti A nám na místě a_{ij} udává počet sledů délky k mezi vrcholy v_i a v_j. Nás zajímají všechny sledy grafu délky l, kde $l = min\{|V(G) - 1|, |E(G)|\}$. Tuto informaci získáme ze součtové matice $S = A + A^2 + A^3 + \ldots + A^l$.

Poznámka:
Součtová matice S reprezentuje tranzitivní uzávěr binární relace, která je popsána maticí A. Pokud bychom při násobení matic používali booleovský součin a součet, byly by v matici S pouze nuly a jedničky (tj. standardní reprezentace konečné binární homogenní relace). Případně by v matici S libovolné nezáporné číslo reprezentovalo příslušnost dvojice prvků do relace.

Animace: Souvislost grafu

5.1.1 Příklady

■ **Příklad 5.1** *Určete silně souvislé komponenty grafu* G=$\{(1,2), (1,3), (2,3), (2,4), (2,5), (4,3), (5,1), (5,3), (5,4)\}$. *Informace o dostupnosti vrcholů určete algoritmem na prohledávání grafů (do hloubky nebo do šířky), Floydovým algoritmem a nebo součtem mocnin matice sousednosti.*

Při určování silně souvislých komponent potřebujeme průběžně údaje o dostupných vrcholech z daného vrcholu. Pomocí vhodného algoritmu zjistíme tyto údaje a uložíme je do čtvercové matice. V průběhu algoritmu na určení silně souvislých komponent je pak použijeme.

1. *Dostupnost pomocí algoritmu na prohledávání grafů.*

Prohledávání grafu G *do hloubky z* v_1*:*

v_1	v_2	v_3	v_4	v_5	Z
0	x	x	x	x	v_1
0	1	x	x	x	v_1, v_2
0	1	2	x	x	v_1, v_2, v_3
0	1	2	x	x	v_1, v_2
0	1	2	2	x	v_1, v_2, v_4
0	1	2	2	x	v_1, v_2
0	1	2	2	2	v_1, v_2, v_5

Prohledávání grafu G *do šířky z* v_2*:*

v_1	v_2	v_3	v_4	v_5	F
x	0	x	x	x	v_2
x	0	2	2	2	v_3, v_4, v_5
x	0	2	2	2	v_4, v_5
x	0	2	2	2	v_5
5	0	2	2	2	v_1

Výsledky prohledávání grafu G *do šířky. z* v_3, v_4 *a* v_5*:*

	v_1	v_2	v_3	v_4	v_5
v_3	x	x	0	x	x
v_4	x	x	4	0	x
v_5	5	1	5	5	0

2. *Dostupnost pomocí Floydova algoritmu.*

   Transformované matice, které vzniknou v průběhu Floydova algoritmu:

$$C^0 = \left[\begin{array}{c|cccc} 0 & 1 & 1 & x & x \\ \hline x & 0 & 1 & 1 & 1 \\ x & x & 0 & x & x \\ x & x & 1 & 0 & x \\ 1 & x & 1 & 1 & 0 \end{array}\right] \quad C^1 = \left[\begin{array}{c|cccc} 0 & 1 & 1 & x & x \\ \hline x & 0 & 1 & 1 & 1 \\ \hline x & x & 0 & x & x \\ x & x & 1 & 0 & x \\ 1 & 2 & 1 & 1 & 0 \end{array}\right] \quad C^2 = \left[\begin{array}{cc|c|cc} 0 & 1 & 1 & 2 & 2 \\ x & 0 & 1 & 1 & 1 \\ \hline x & x & 0 & x & x \\ \hline x & x & 1 & 0 & x \\ 1 & 2 & 1 & 1 & 0 \end{array}\right]$$

$$C^3 = \left[\begin{array}{ccc|c|c} 0 & 1 & 1 & 2 & 2 \\ x & 0 & 1 & 1 & 1 \\ x & x & 0 & x & x \\ \hline x & x & 1 & 0 & x \\ \hline 1 & 2 & 1 & 1 & 0 \end{array}\right] \quad C^4 = \left[\begin{array}{cccc|c} 0 & 1 & 1 & 2 & 2 \\ x & 0 & 1 & 1 & 1 \\ x & x & 0 & x & x \\ x & x & 1 & 0 & x \\ \hline 1 & 2 & 1 & 1 & 0 \end{array}\right] \quad C^5 = \left[\begin{array}{ccccc} 0 & 1 & 1 & 2 & 2 \\ 2 & 0 & 1 & 1 & 1 \\ x & x & 0 & x & x \\ x & x & 1 & 0 & x \\ 1 & 2 & 1 & 1 & 0 \end{array}\right]$$

   V poslední matici C_5 máme ceny všech nejkratších cest v zadaném grafu. Opět je můžeme použít při určování silně souvislých komponent grafu.

3. *Dostupnost pomocí matice $S = B + B^2 + B^3 + \ldots + B^l$, kde $l = min\{|V(G) - 1|, |E(G)|\}$ a B je matice sousednosti grafu G:*

$$B = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \quad B^2 = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 \end{bmatrix} \quad B^3 = \begin{bmatrix} 1 & 0 & 2 & 1 & 0 \\ 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$B^4 = \begin{bmatrix} 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 1 & 0 \end{bmatrix} \quad S = \begin{bmatrix} 1 & 2 & 6 & 2 & 1 \\ 1 & 1 & 6 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 2 & 1 & 6 & 3 & 1 \end{bmatrix}$$

   V jednotlivých maticích B^l reprezentuje nenulová hodnota prvku b^l_{ij} počet sledů délky l mezi vrcholy v_i a v_j.

Algoritmus pro určení silně souvislých komponent grafu G může použít libovolného výše uvedeného postupu, který nám poskytuje informace o dostupnosti v grafu.

Postupné zařazování vrcholů do silně souvislých komponent:

v_1	v_2	v_3	v_4	v_5
x	x	x	x	x
v_1	v_1	x	x	v_1
v_1	v_1	v_3	x	v_1
v_1	v_1	v_3	v_4	v_1

Výsledkem algoritmu jsou tedy tři silně souvislé komponenty $K_1 = \{v_1, v_2, v_5\}$, $K_2 = \{v_3\}$ *a* $K_3 = \{v_4\}$.
□

Shrnutí kapitoly 5.1

V této kapitole jsme se naučili určovat komponenty souvislosti daného grafu.

Otázky ke kapitole 5.1

1. Graf H je dán seznamem hran $E(H) = \{(v_1, v_2; 5), (v_1, v_3, 3), (v_1, v_4; 11), (v_3, v_2; -2), (v_3, v_4; 7), (v_3, v_5; 6), (v_5, v_1; 1), (v_5, v_4; 9)\}$. Určete silně souvislé komponenty grafu G:
   (a) použijte Floydův algoritmus pro určení matice vzdálenosti,
   (b) použijte součtovou matici S pro určení dostupných vrcholů z libovolného vrcholu.

Kapitola 6

Nejlevnější kostra grafu

Obsah

6.1 Pojmy, základní algoritmus

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat základní algoritmus pro určování nejlevnější kostry grafu a na příkladech tento algoritmus použít.

Kostra v neorientovaném ohodnoceném souvislém grafu (viz obr. 1.8) je faktor grafu, který je stromem. Pro nejlevnější kostru je součet cen hran tohoto stromu minimální. Například známe-li délky všech silnic mezi několika městy, pak nejlevnější kostra zajišťuje existenci cesty (a to právě jedné) z libovolného města do jiného tak, že celkově je délka silnic minimální (výhodné pro zimní údržbu komunikací).

Hledáme nejlevnější kostru grafu. Výchozí stav je množina diskrétních vrcholů L grafu, ve kterém hledáme nejlevnější kostru. Naším úkolem je postupně přidávat co nejlevnější hrany tak, aby nevznikla kružnice a aby výsledný faktor byl stromem.

Algoritmus:

1. L je diskrétní graf s množinou vrcholů $V(L) = V(G)$
2. Je-li les L stromem, výpočet končí. L je hledaná nejlevnější kostra.
3. Zvolíme libovolně hranu h s touto vlastností: hrana h spojuje dvě různé komponenty lesa L a alespoň pro jednu z těchto komponent (označme ji A) platí, že cena hrany h je nejmenší ze všech cen hran z množiny $\{e; PV(e) \in A,\ KV(e) \in V(G) - A\}$
4. Hranu h přidáme k lesu L (tím se sníží počet komponent) a pokračujeme bodem 2

Shrnutí kapitoly 6.1

V této kapitole jsme se seznámili s principem určování nejlevnějších koster grafu na základním algoritmu.

Otázky ke kapitole 6.1

1. Neorientovaný graf H je dán seznamem hran:
   $(v_1, v_2; 5)$, $(v_1, v_3, 3)$, $(v_1, v_4; 11)$, $(v_3, v_2; -2)$, $(v_3, v_4; 7)$, $(v_3, v_5; 6)$, $(v_5, v_1; 1)$, $(v_5, v_4; 9)$. Určete nejlevnější kostru a její cenu v grafu H (zanedbejte orientaci hran). Použijte základní algoritmus.

6.2 Hladový algoritmus - Borůvkův (Kruskalův)

Časová náročnost kapitoly: 15 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat hladový algoritmus pro určování nejlevnější kostry grafu a na příkladech tento algoritmus použít.

Následující postup hledání nejlevnější kostry popsal profesor Otakar Borůvka (1926), vešel ve známost na základě článku J.B.Kruskala (1956). Hrany grafu uspořádáme podle jejich cen do neklesající posloupnosti. Pak je v tomto pořadí probíráme a do postupně vytvářeného grafu L přidáváme ty z nich, které v grafu L nezpůsobí vznik kružnice.

Algoritmus:
Použijeme pole vrcholů Uz, kde si označíme vrcholy začleněné do lesu a pole hran Hr, kde si označíme hrany, začleněné do nejlevnější kostry. Do proměnné k uložíme počet hran v lese.

1. Inicializace: $Uz_i = 0$, $Hr_i := 0$ pro $\forall i$ a $k := 0$.
2. Ohodnocené hrany setřídíme do neklesající posloupnosti.
3. Je-li $k = n - 1$, potom bod 5. Jinak vezmeme nejlevnější hranu h_i takovou, že $Hr_i = 0$.
4. Testujeme, zda můžeme hranu $h_i = (v_x, v_y)$ přidat do lesu, aniž by vznikla kružnice. Je-li $(Uz_x = 1 \wedge Uz_y = 1)$ testujeme, (viz. poznámka) zda přidáním hrany h_i nevznikne v lesu L kružnice. Vznikne-li kružnice, pak hranu h_i nemůžeme přidat do kostry a změníme $Hr_i := 2$. Jinak $\{Uz_x := 1; Uz_y := 1; k := k + 1; Hr_i := 1\}$ a pokračujeme bodem 3.
5. Konec.

Poznámka:
V poli Hr máme uloženy informace o hranách, které jsme ještě nepoužili (0), které byly přidány do kostry (1), a které by způsobily vznik kružnic (2).

Poznámka:
Test vzniku kružnice po přidání hrany $h_i = (v_x, v_y)$ můžeme provést takto: je-li vrchol v_x dosažitelný z v_y (nebo naopak) jen pomocí hran, které již byly začleněny do kostry, pak přidáním hrany h_i by vznikla kružnice (tj. hranu nepřidáme a ne její pozici v poli Hr umístíme 2).

Složitost:
Řádová složitost algoritmu je $O(m\ log\ n)$.

Animace: Hladový algoritmus

■ **Příklad 6.1** *V ohodnoceném neorientovaném grafu G_4 určete nejlevnější a nejdražší kostru pomocí hladového algoritmu. Graf G_4 je dán seznamem hran:* $E(G_4)$=$\{(v_1, v_2; 1), (v_1, v_3; 5), (v_1, v_4; 3), (v_1, v_5; 7), (v_2, v_3; 4), (v_2, v_5; 6), (v_3, v_4; 2)\}$.

1. *Vzestupně setříděná posloupnost hran:* $(v_1, v_2; 1)$, $(v_3, v_4; 2)$, $(v_1, v_4; 3)$, $(v_2, v_3; 4)$, $(v_1, v_3; 5)$, $(v_2, v_5; 6)$, $(v_1, v_5; 7)$

v_1	v_2	v_3	v_4	v_5	*test hrany*	*zařazena?*
x	x	x	x	x		
○	○	x	x	x	$(v_1, v_2; 1)$	*ano*
○	○	○	○	x	$(v_3, v_4; 2)$	*ano*
○	○	○	○	x	$(v_1, v_4; 3)$	*ano*
○	○	○	○	x	$(v_2, v_3; 4)$	*ne*
○	○	○	○	x	$(v_1, v_3; 5)$	*ne*
○	○	○	○	○	$(v_2, v_5; 6)$	*ano*

Cena nejlevnější kostry je 12.

2. *Sestupně setříděná posloupnost hran:* $(v_1, v_5; 7)$, $(v_2, v_5; 6)$, $(v_1, v_3; 5)$, $(v_2, v_3; 4)$, $(v_1, v_4; 3)$, $(v_3, v_4; 2)$, $(v_1, v_2; 1)$

v_1	v_2	v_3	v_4	v_5	*test hrany*	*zařazena?*
x	x	x	x	x		
○	x	x	x	○	$(v_1, v_5; 7)$	*ano*
○	○	x	x	○	$(v_2, v_5; 6)$	*ano*
○	○	○	x	○	$(v_1, v_3; 5)$	*ano*
○	○	○	x	○	$(v_2, v_3; 4)$	*ne*
○	○	○	○	○	$(v_1, v_4; 3)$	*ano*

Cena nejdražší kostry je 21.

Poznámka:
V simulovaném algoritmu není zachycen test vzniku kružnice. Např. při hledání nejlevnější kostry jsme testovali hranu (v_2, v_3) v situaci, že v kostře již byly zahrnuty hrany (v_1, v_2), (v_3, v_4) a (v_1, v_4). Přidáním hrany (v_2, v_3) by vznikla kružnice, protože vrchol v_2 je pomocí hran kostry dosažitelný z vrcholu v_3.
□

Shrnutí kapitoly 6.2

V této kapitole jsme se seznámili s principem určování nejlevnějších koster grafu pomocí hladového algoritmu.

Otázky ke kapitole 6.2

1. Neorientovaný graf H je dán seznamem hran:
   $(v_1, v_2; 5)$, $(v_1, v_3, 3)$, $(v_1, v_4; 11)$, $(v_3, v_2; -2)$, $(v_3, v_4; 7)$, $(v_3, v_5; 6)$, $(v_5, v_1; 1)$, $(v_5, v_4; 9)$. Určete nejlevnější a nejdražší kostru a její cenu v grafu H (zanedbejte orientaci hran). Použijte hladový algoritmus.

6.3 Jarníkův - Primův algoritmus

Časová náročnost kapitoly: 15 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat Jarníkův algoritmus pro určování nejlevnější kostry grafu a na příkladech tento algoritmus použít.

Je to další modifikace základního algoritmu, tentokrát nazvaná podle známeho českého matematika profesora Vojtěcha Jarníka (1930). Začneme z libovolného vrcholu a postupně k němu přidáváme vrcholy a hrany tak, že v každé fázi výpočtu máme strom. Přidávanou hranu přitom vždy vybíráme tak, aby měla nejmenší cenu z množiny $W_G(A)$ (všechny hrany, které mají jeden krajní vrchol v množině A a druhý v $V(G) - A$), kde A je množina vrcholů stromu.

Složitost:
Řádová složitost algoritmu je $O(n^2)$.

Animace: Jarníkův algoritmus

■ **Příklad 6.2** *V ohodnoceném neorientovaném grafu G_4 určete nejlevnější a nejdražší kostru pomocí Jarníkova algoritmu. Graf G_4 je dán seznamem hran:* $E(G_4)$=$\{(v_1, v_2; 1), (v_1, v_3; 5), (v_1, v_4; 3), (v_1, v_5; 7), (v_2, v_3; 4), (v_2, v_5; 6), (v_3, v_4; 2)\}$.

1. *Vybereme libovolný vrchol, od kterého začneme budovat nejlevnější kostru grafu. V tomto příkladu byl vybrán vrchol v_1.*

v_1	v_2	v_3	v_4	v_5	*nejlevnější incid. hrana*
○	X	X	X	X	
○	○	X	X	X	$(v_1, v_2; 1)$
○	○	X	○	X	$(v_1, v_4; 3)$
○	○	○	○	X	$(v_3, v_4; 2)$
○	○	○	○	○	$(v_2, v_5; 6)$

I tímto algoritmem byla potvrzena cena nejlevnější kostry rovnající se 12.

2. *Vybereme libovolný vrchol, od kterého začneme budovat nejdražší kostru grafu. V tomto příkladu byl vybrán vrchol v_1.*

v_1	v_2	v_3	v_4	v_5	*nejdražší incid. hrana*
○	x	x	x	x	
○	x	x	x	○	$(v_1, v_5; 7)$
○	○	x	x	○	$(v_2, v_5; 6)$
○	○	○	x	○	$(v_1, v_3; 5)$
○	○	○	○	○	$(v_1, v_4; 3)$

Zde také došlo k potvrzení ceny nejdražší kostry rovnající se 21.

□

Shrnutí kapitoly 6.3

V této kapitole jsme se seznámili s principem určování nejlevnějších koster grafu pomocí Jarníkova algoritmu.

Otázky ke kapitole 6.3

1. Neorientovaný graf H je dán seznamem hran:
   $(v_1, v_2; 5)$, $(v_1, v_3, 3)$, $(v_1, v_4; 11)$, $(v_3, v_2; -2)$, $(v_3, v_4; 7)$, $(v_3, v_5; 6)$, $(v_5, v_1; 1)$, $(v_5, v_4; 9)$. Určete nejlevnější a nejdražší kostru a její cenu v grafu H. Použijte Jarníkův algoritmus.

Kapitola 7

Párování (Matching)

Obsah

7.1 Pojmy

Časová náročnost kapitoly: 15 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět definovat pojmy týkající se úloh o párování.

V souvislosti s párováním nás budou zajímat tyto úlohy:

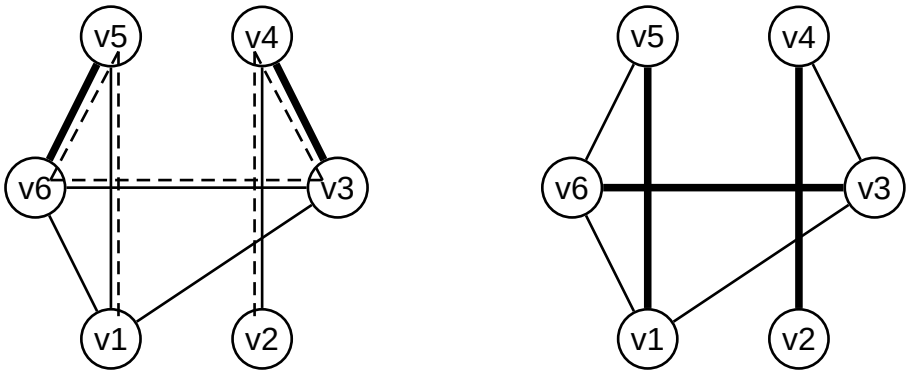
- v grafu nalézt maximální párování (párování s největším počtem hran),
- v ohodnoceném grafu nalézt nejlevnější maximální párování,
- v ohodnoceném grafu nalézt nejdražší párování.

Párování v grafu G je taková množina hran $M \subseteq E(G)$, že žádné dvě hrany z M nemají společný vrchol (nejsou incidentní). Vrchol $u \in V(G)$ je ***nasycen v párování*** M, jestliže existuje v párování M hrana s ním incidentní. ***Perfektní párování*** je takové, že nasycuje všechny vrcholy grafu. ***Maximální párování*** je takové, že obsahuje největší počet hran (pokud existuje perfektní párování, tak je i maximálním). Maximální párování je tedy takové párování M, že neexistuje párování M', které by obsahovalo M jako podgraf.

Je-li dán graf G a párování M v grafu G, pak ***střídavá (alternující) cesta*** vzhledem k párování M je taková neorientovaná cesta, že její hrany střídavě leží v M a neleží v M. Je-li krajní vrchol cesty nasycen v párování M, pak hrana, která jej nasycuje, je částí cesty. ***Střídavá kružnice*** vzhledem k párování M je kružnice, která má sudou délku a jejíž hrany střídavě leží a neleží v párování M.

Pomocí střídavých cest a kružnic lze změnit párování tak, že u hran, které leží na střídavé cestě nebo kružnici, změníme jejich příslušnost k párování. Je-li H množina hran tvořících střídavou cestu nebo kružnici vzhledem k párování P, vytvoříme párování P' takto:

$$\text{jestliže } e \in H, \text{ pak } e \in P' \iff e \notin P,$$
$$\text{jestliže } e \notin H, \text{ pak } e \in P' \iff e \in P,$$



Obrázek 7.1: Střídavá cesta a změna párování podél této střídavé cesty.

Takováto změna párování se nazývá změnou podél střídavé cesty nebo kružnice (viz obr. 7.1).

Máme-li ohodnocený graf G s párováním M a vzhledem k němu střídavou cestu, popř. střídavou kružnici, množinu hran této cesty. popř. kružnice označme H. Pak cena střídavé cesty, popř. kružnice je definována jako

$$C = \sum_{e \in H-M} c(e) - \sum_{e \in H \cap M} c(e)$$

.

Věta 7.1 *Párování M v grafu G je největší právě tehdy, když v grafu G neexistuje vzhledem k párování M střídavá cesta spojující dva nenasycené vrcholy.*

Shrnutí kapitoly 7.1

V této kapitole jsme se seznámili s pojmy, které se týkají párování, a to:

1. párování, maximální párování, perfektní párování
2. střídavá cesta, cena střídavé cesty.

Otázky ke kapitole 7.1

1. Jak se mění párování podél střídavé cesty?
2. Je perfektní párování zároveň párováním maximálním?
3. Je párování na obrázku 7.1 vpravo maximálním párováním? Pokud ano, je toto párování jediným maximálním párováním?

7.2 Maximální párování v bipartitním grafu

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět vysvětlit princip algoritmu pro nalezení maximálního párování a v příkladu jej použít.

Párování v bipartitních grafech je podstatně jednodušší než v obecném grafu. Existuje těsný vztah mezi párováním v bipartitním grafu a toky v sítích (jak později uvidíme).

Algoritmus:
Pro maximální párování v bipartitním grafu.

1. Provedeme níže uvedený algoritmus pro značení vrcholů. Pokud zjistíme, že stávající párování je maximální pokračujeme bodem 2. Jinak pokračujeme bodem 1.
2. Konec.

Algoritmus:
Pro označování vrcholů. Nechť G je bipartitní graf s partitami X a Y. Potom se následující algoritmus použije pro značení vrcholů při hledání střídavé cesty nebo zjištění, že dané párování je již maximální.

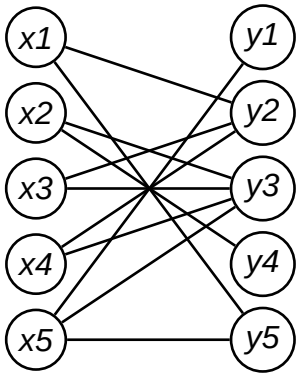
1. Označíme všechny nenasycené vrcholy z množiny X. Ostatní vrcholy jsou bez značek (i ty z partity Y).
2. Existuje-li hrana $h \notin M$ vedoucí z označeného vrcholu $x \in X$ do neoznačeného vrcholu $y \in Y$, pak označíme vrchol y a pokračujeme bodem 3. Neexistuje-li taková hrana, přejdeme k bodu 5, značkování končí, střídavá cesta spojující neoznačené vrcholy neexistuje.
3. Je-li právě označený vrchol y nenasycený, značení končí. Je nalezena střídavá cesta s nenasycenými krajními vrcholy a lze zvětšit počet hran v párování (podél této střídavé cesty). Změníme párování a přejdeme k bodu 5. Je-li však právě označený vrchol y v párování M nasycen, pokračujeme podle bodu 4.
4. Označíme vrchol z množiny X, který je spárován s vrcholem y a pokračujeme podle kroku 2.

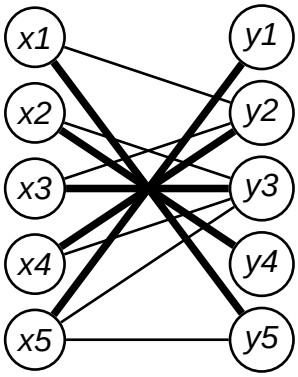
5. Konec označování vrcholů.

Poznámka:
Je vhodné si při značkovací proceduře ukládat označené vrcholy do fronty nebo zásobníku a uchovávat si tam aktuální informaci o střídavé cestě.

Animace: Maximální párování v bipartitním grafu

■ **Příklad 7.1** *V bipartitním grafu který je reprezentován seznamem*
$x1 \longrightarrow y2, y5$
$x2 \longrightarrow y3, y4$
$x3 \longrightarrow y2, y3$
$x4 \longrightarrow y2, y3$
$x5 \longrightarrow y1, y5$
určete maximální párování.





Postup (znaménko $=$ označuje hranu, která patří do párování, znaménko $-$ označuje hranu, která do párování nepatří.):

1. *Označíme $x1, x2, x3, x4, x5$ a hledáme rozšiřitelné cesty z označených vrcholů. Existuje rozšiřitelná střídavá cesta $x1 - y2$ (označíme $y2$). Změníme podél ní párování. $P = \{(x1, y2)\}$.*
2. *Označíme $x2, x3, x4, x5$. Střídavá cesta $x2 - y3$ (ozn. $y3$). Změníme podél ní párování. $P = \{(x1, y2), (x2, y3)\}$.*
3. *Označíme $x3, x4, x5$. Střídavá cesta $x3 - y2 = x1 - y5$. Změníme podél ní párování. $P = \{(x1, y5), (x2, y3), (x3, y2)\}$.*
4. *Označíme $x4, x5$. Střídavá cesta $x4 - y2 = x3 - y3 = x2 - y4$. Změníme podél ní párování. $P = \{(x1, y5), (x2, y4), (x3, y3), (x4, y2)\}$.*
5. *Označíme $x5$. Střídavá cesta $x5 - y1$. Změníme podél ní párování. $P = \{(x1, y5), (x2, y4), (x3, y3), (x4, y2), (x5, y1)\}$.*
6. *Neexistuje vrchol nenasycený v párování. Nalezli jsme maximální párování, které je i perfektní.*

□

Shrnutí kapitoly 7.2

V této kapitole jsme se seznámili s algoritmem pro nalezení maximálního párování v bipartitním grafu.

Otázky ke kapitole 7.2

1. V bipartitním grafu G_1, který je zadán seznamem hran $E(G_1) = \{(x1, y1), (x2, y1), (x2, y3), (x3, y2), (x3, y3), (x3, y4), (x4, y3), (x5, y3)\}$, určete maximální párování.
2. Určete maximální párování v grafu G_2. $E(G_2) = \{(x1, y1), (x1, y2), (x1, y3), (x2, y1), (x2, y2), (x2, y3), (x3, y1), (x3, y2), (x3, y3)\}$.

7.3 Nejlevnější maximální párování v bipartitním grafu

Časová náročnost kapitoly: 30 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět vysvětlit princip algoritmu pro nalezení nejlevnějšího perfektního párování v bipartitním grafu a v příkladu jej použít.

Máme bipartitní graf G s dvěma partitami X a Y. Představme si množinu vrcholů X jako pracovníky, množinu vrcholů Y jako jednotlivé pracovní úkony a ceny hran jako časové údaje, za jak dlouho jsou jednotliví pracovníci schopni daný úkon vykonat. Pak nalezení nejlevnějšího perfektního (případně pouze největšího) párování v grafu G je ekvivalentní tomu, že pracovníkům přiřadíme jejich práci tak, aby celkově pracovali co nejkratší dobu (kterou pak zaměstnavatel proplatí).

Nechť G je úplný bipartitní graf s množinami vrcholů X a Y takovými, že $|X| = |Y| = n$. Ceny hran můžeme uspořádat do tvaru matice cen C jejíž libovolný prvek c_{ij} je cenou hrany (x_i, y_j).

Uvažujme libovolné ohodnocení vrcholů grafu G reálnými čísly p_u pro $u \in X \cup Y$. Definujme transformovanou matici cen C^p předpisem

$$c^p_{ij} = c_{ij} - p_i - p_j.$$

Ohodnocení vrcholů p nazveme přípustným ohodnocením, jestliže pro každou hranu (x_i, y_j) grafu G platí: $c^p_{ij} = c_{ij} - p_i - p_j \geq 0$. 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. $c_{ij} = p_i + p_j$. Ostatní hrany grafu G mají transformované ceny kladné.

Algoritmus:
Maďarský. Myšlenka tohoto algoritmu spočívá v nalezení takového přípustného ohodnocení vrcholů, že v příslušném grafu rovnosti existuje perfektní párování.

Transformovanou matici cen a přípustné ohodnocení vrcholů získáme tím, že od každého řádku původní matici cen odečteme minimální cenu. Tím získáme v každém řádku alespoň jednu nulu. Tutéž operaci provedeme se sloupci (tedy i každý sloupec obsahuje alespoň jednu nulu). K takto získané transformované matici cen sestrojíme graf rovnosti G^p a v něm pomocí označovací procedury najdeme maximální párování. Získáme-li perfektní párování, je úloha vyřešena.

Jestliže v grafu G^p neexistuje perfektní párování, pak v něm existuje množina $A \subseteq X$ taková, že $|A| > V_{G^p}(A)$. Tuto množinu najde označovací procedura při neúspěšném pokusu o nalezení rozšiřitelné střídavé cesty. Množina $A = Z \cap X$, kde Z je množina označených vrcholů.

Abychom mohli zvětšit párování, musíme ke grafu rovnosti G^p přidat některou hranu incidentní s množinou A. Pro tento účel zvýšíme p_i na množině A a zároveň snížíme p_j na množině $V_{G^p}(A)$ o tutéž hodnotu takovou, aby se vynulovala transformovaná cena na některé hraně vedoucí z A do $Y - V_{G^p}(A)$ a přitom aby všechny transformované ceny zůstaly nezáporné.

1. Počáteční přípustné ohodnocení vrcholů:
   Pro $\forall i$ položíme: $p_i := min\{c_{ij}; j \in Y\}$.
   Pro $\forall j$ položíme: $p_j := min\{c_{ij} - p_i; i \in X\}$.

2. Sestrojíme graf rovnosti G^p takový, že:
   $E(G^p) = \{(x_i, y_j); c_{ij} - p_i - p_j = 0\}$ a $V(G^p) = V(G)$

3. Sestrojíme maximální párování v grafu G^p. Je-li toto párování perfektní, výpočet končí, výsledné perfektní párování je nejlevnější.

4. Není-li maximální párování perfektní, nalezneme množinu $A \subseteq U$ takovou, že $|A| > |V_{G^p}(A)|$, vypočteme $d = min\{c_{ij} - p_i - p_j; i \in A, j \notin V_{G^p}(A)\}$ a změníme přípustné ohodnocení vrcholů takto:
   $p_i := p_i + d\ pro\ \forall i \in A$ a $p_j := p_j - d\ pro\ \forall j \in V_{G^p}(A)$. Pokračujeme bodem 2

Poznámka:
Množina Z jsou vrcholy označené při hledání maximálního párování v grafu G^p. Množina $A = Z \cap X$ a $V_{G^p}(A) = Z \cap Y$.

Složitost:
Řádová složitost algoritmu je $O(n^4)$.

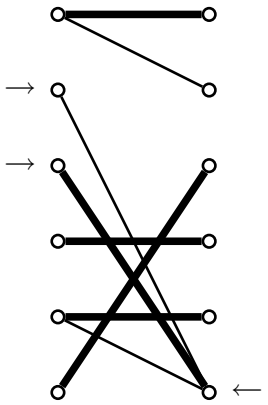
Animace: Nejlevnější maximální párování v bipartitním grafu

■ **Příklad 7.2** *V grafu G určeném maticí cen C určete nejlevnější maximální párování.*

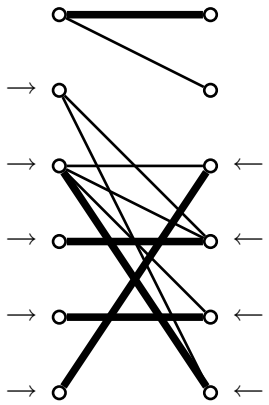
$$C = \begin{bmatrix} 5 & 3 & 7 & 4 & 5 & 4 \\ 10 & 11 & 10 & 7 & 8 & 3 \\ 18 & 7 & 6 & 6 & 6 & 2 \\ 6 & 12 & 2 & 1 & 9 & 8 \\ 8 & 4 & 4 & 4 & 1 & 1 \\ 4 & 8 & 1 & 3 & 7 & 4 \end{bmatrix}$$

$$C^1 = \begin{array}{c|cccccc} & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 3 & 2 & 0 & 4 & 1 & 2 & 1 \\ 3 & 7 & 8 & 7 & 4 & 5 & 0 \\ 2 & 16 & 5 & 4 & 4 & 4 & 0 \\ 1 & 5 & 11 & 1 & 0 & 8 & 7 \\ 1 & 7 & 3 & 3 & 3 & 0 & 0 \\ 1 & 3 & 7 & 0 & 2 & 6 & 3 \end{array}$$

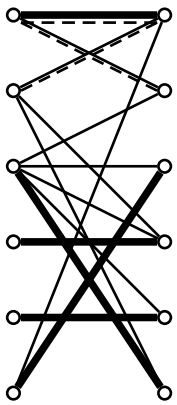
$$C^2 = \begin{array}{c|cccccc} & 2 & 0 & 0 & 0 & 0 & 0 \\ \hline 3 & 0 & 0 & 4 & 1 & 2 & 1 \\ 3 & 5 & 8 & 7 & 4 & 5 & 0 \\ 2 & 14 & 5 & 4 & 4 & 4 & 0 \\ 1 & 3 & 11 & 1 & 0 & 8 & 7 \\ 1 & 5 & 3 & 3 & 3 & 0 & 0 \\ 1 & 1 & 7 & 0 & 2 & 6 & 3 \end{array}$$



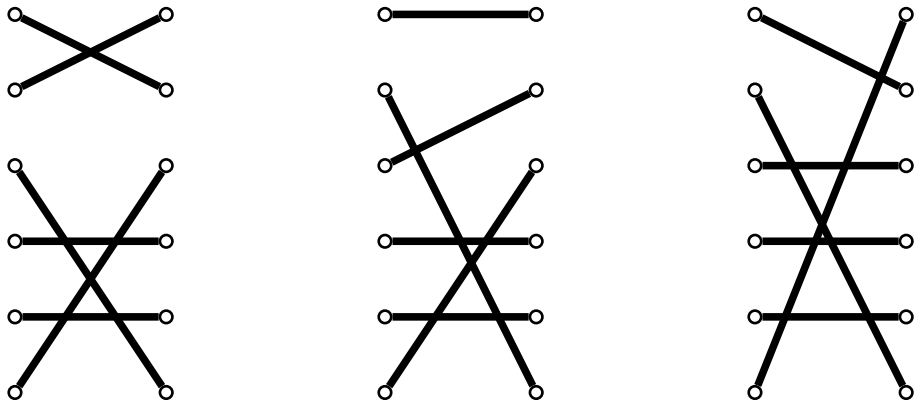
$$C^3 = \begin{array}{c|cccccc} & 2 & 0 & 0 & 0 & 0 & -4 \\ \hline 3 & 0 & 0 & 4 & 1 & 2 & 5 \\ 7 & 1 & 4 & 3 & 0 & 1 & 0 \\ 6 & 10 & 1 & 0 & 0 & 0 & 0 \\ 1 & 3 & 11 & 1 & 0 & 8 & 11 \\ 1 & 5 & 3 & 3 & 3 & 0 & 4 \\ 1 & 1 & 7 & 0 & 2 & 6 & 7 \end{array}$$



$$C^4 = \begin{array}{c|cccccc} & 2 & 0 & -1 & -1 & -1 & -5 \\ \hline 3 & 0 & 0 & 5 & 1 & 3 & 6 \\ 8 & 0 & 3 & 3 & 0 & 1 & 0 \\ 7 & 9 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 10 & 1 & 0 & 8 & 11 \\ 2 & 4 & 2 & 3 & 3 & 0 & 4 \\ 2 & 0 & 6 & 0 & 2 & 6 & 7 \end{array}$$



Tři výsledná párování, cena každého z nich je rovna součtu ohodnocení vrcholů, tj. 18*.*



□

Shrnutí kapitoly 7.3

V této kapitole jsme se seznámili s algoritmem pro nalezení nejlevnějšího perfektního párování v bipartitním grafu.

Otázky ke kapitole 7.3

1. Určete nejlevnější perfektní párování a jeho cenu v grafu G_1. Jednotlivé kroky algoritmu popište (přehledně). Jaká je ukončovací podmínka tohoto algoritmu pro neúplný ohodnocený bipartitní graf? $E(G_1) = \{(x1, y1, 1), (x1, y2, 3), (x1, y3, 5), (x2, y1, 2), (x2, y2, 3), (x2, y3, 5), (x3, y1, 6), (x3, y2, 2), (x3, y3, 4)\}$.

2. Určete nejlevnější perfektní párování v grafu G_2. Jednotlivé kroky algoritmu popište (přehledně). $E(G_2) = \{(x1, y1, 4), (x1, y2, 1), (x1, y3, 4), (x2, y1, 4), (x2, y2, 2), (x2, y3, 3), (x3, y1, 4), (x3, y2, 5), (x3, y3, 3)\}$

7.4 Maximální párování v obecném grafu

Časová náročnost kapitoly: 35 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět vysvětlit princip algoritmu pro nalezení maximálního párování v obecném grafu a na příkladu jej použít.

Máme obecný graf s počátečním párováním M. Zjišťujeme, zda vzhledem k párování M existuje vhodná rozšiřitelná střídavá cesta, popř. střídavá kružnice. Jestliže neexistuje, pak je párováni M již maximální. Jestliže existuje, změníme (tj. rozšíříme) párování a znovu hledáme další střídavou cestu, popř. kružnici.

Bohužel je hledání potřebných střídavých cest a kružnic složitější než to na první pohled vypadá. Takže si alespoň zjednodušíme označení vrcholů grafu - každý vrchol je jednoznačně určen svým indexem.

Algoritmus:
Edmondsův. Máme výchozí párování. Značením vrcholů najdeme zvětšující se cestu a párování M podél ní zvětšíme. Značení provádíme následujícím způsobem:

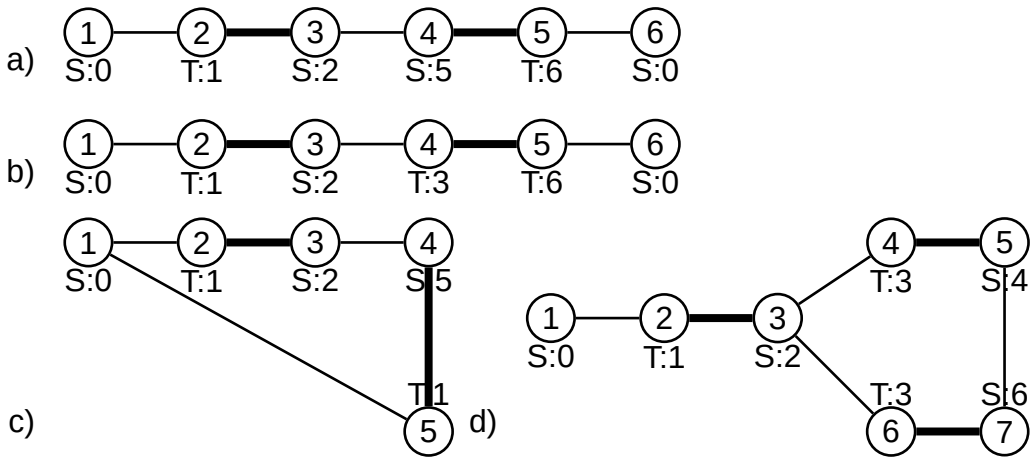
Na začátku dostanou všechny volné vrcholy u (nenasycené v párování) značku $S:0$.

Obecně, nechť už některé vrcholy mají značku. Každý označený vrchol i má značku tvaru $S:k$, resp. $T:k$. Což znamená, že existuje střídavá cesta $u-i$ začínající v nějakém volném vrcholu u, vedoucí do vrcholu i a její předposlední vrchol je k. Vrchol u se nazývá kořen pro i. Označený, ale dosud neprozkoumaný vrchol i prozkoumáme takto:

1. Jestliže i má značku S, tak každý neoznačený vrchol j takový, že hrana $ij \in E(G)-M$, dostane značku $T:i$.
2. Jestliže i má značku T, $ij \in M$ a vrchol j je neoznačený, tak j dostane značku $S:i$.

Problémy, kterou mohou nastat v případě, že vrchol j už značku má:

V uvedených případech mají i a j stejné značky (S nebo T). My vyhledáme jejich kořeny (tj. volné vrcholy ui a uj, z kterých vychází značení pro i, resp. pro j) pomocí druhých složek vrcholů - postupujeme zpětně od i a j. Najdeme střídavou $ui-i$ cestu P_i a střídavou cestu P_j (na obrázku 7.2a) je $i=3$, $j=4$, $ui=1$, $uj=6$).



Obrázek 7.2: Možnosti (problémy), které mohou nastat při značkování.

Jestliže $ui \neq uj$, pak zřejmě je $P_i - P_j^{-1}$ rozšiřitelná střídavá $ui - uj$ cesta.

Jestliže $ui = uj = u$ (viz obr. 7.2c a 7.2d), tak $P_i - P_j^{-1}$ vytváří podgraf, který se skládá ze střídavé $u - v$ cesty P a střídavého $v - v$ cyklu Z liché délky, přičemž obě hrany cyklu incidentní s v jsou nenasycené v párování. Cyklus Z se nazývá květ, cesta P stopka (květu Z) a v je báze (květu Z).

Květ Z stáhneme do jediného vrcholu, který budeme nazývat pseudovrcholem. Dáme mu značku, kterou měla báze květu Z, přičemž vzniklý pseudovrchol považujeme za neprozkoumaný. Potom pokračujeme ve značení v novém grafu G' při odpovídajícím párování M'. V dalším značení můžeme objevit nový květ Z' v G'. Po jeho stáhnutí pokračujeme ve značení v novém grafu G'' při párování M'' atd.

V nějakém G^k buď

1. objevíme rozšiřitelnou cestu, nebo
2. značení skončí prozkoumáním všech označených vrcholů a bez objevení rozšiřitelné cesty.

V prvním případě můžeme párování M^k (a tedy i párování M) zvětšit. V druhém případě tvoří sjednocení všech alternujících cest, přes které jsme označili jednotlivé vrcholy, tzv. maďarský les (v tomto lese pro každou hranu grafu platí: jestliže jeden její kraj má značku S, tak druhý patří také do lesu a má značku T). Párování M je již maximální.

Věta 7.2 *Nechť Z je květ objevený v grafu G při značkování vzhledem k párování M. Dále nechť po stažení Z na vrchol Z' vznikne z grafu G graf G' a nechť z párování M vznikne párování M'. Potom v G' existuje zlepšující cesta pro M' právě tehdy když v G existuje zvětšující cesta pro M.*

Algoritmus:

1. Najdeme nějaké (např. prázdné nebo maximální) párování M v grafu G. žádný vrchol není označený.
2. Značení:

(a) Každému volnému vrcholu (nenasycenému v párování) přiřadíme značku $S:0$. Všechny vrcholy jsou neprozkoumané.

(b) Jestliže jsou všechny označené vrcholy prozkoumané, jdeme na bod 4. Jinak zvolíme libovolný neprozkoumaný vrchol i z označených vrcholů. Jestliže i má značku S, jdeme na bod 2c. Jestliže i má značku T, jdeme na bod 2d.

(c) Pro každou hranu $ij \notin M$ provedeme:
Jestliže vrchol j má značku S, tak podle druhých složek značek najdeme kořen pro střídavé cesty začínající v i a j. Jestliže $ui \neq uj$, pak bod 3. Jestliže $ui = uj$ (cesty mají společný kořen), tak jsme objevili květ, který stáhneme na jediný vrchol se značkou báze květu; považujeme ho za neprozkoumaný a přejdeme na bod 2b.
Jestliže vrcholj j má značku T, tak ji necháme a jestliže nemá žádnou značku, tak mu přiřadíme $T:i$. Tímto je vrchol i prozkoumaný a přejdeme k bodu 2b

(d) Pro jedinou hranu $ij \in M$ provedeme:
Jestliže vrchol j má značku T, tak podle druhých složek značek najdeme kořen pro střídavé cesty začínající v i a j. Jestliže $ui \neq uj$, pak bod 3. Jestliže $ui = uj$ (cesty mají společný kořen), tak jsme objevili květ, který stáhneme na jediný vrchol se značkou báze květu; považujeme ho za neprozkoumaný a přejdeme na bod 2b.
Jestliže vrcholj j má značku S, tak ji necháme a jestliže nemá žádnou značku, tak mu přiřadíme $S:i$. Tímto je vrchol i prozkoumaný a přejdeme k bodu 2b

3. Pomocí nalezené rozšiřitelné cesty v aktuálním grafu G' najdeme rozšiřitelnou cestu v původním grafu G a párování M zvětšíme. Všechny značky setřeme a přejdeme k bodu 2a.

4. žádná rozšiřitelná cesta neexistuje a tedy párování je nejpočetnější. Konec.

Poznámka:
Pro zápis vrcholu, které jsou označené a které postupně prozkoumáváme je výhodné použít frontu. Viz. příklad.

■ **Příklad 7.3** *Máme dán graf G seznamem hran $E(G) = \{(1,2),(1,3),(2,3),(2,5),(2,6),(3,4),(3,5),(3,6)\}$ a výchozí párování $M = \{(2,3)\}$. Určíme maximální párování.*

tabulka značení vrcholů:

1	*2*	*3*	*4*	*5*	*6*	*F*
S:0			*S:0*	*S:0*	*S:0*	*1,4,5,6*
	T:1	*T:1*				*4,5,6,2,3*
						⋮
	*	*				*3*

Vrcholy nenasycené v párování jsme označili symbolem $S:0$. Uložili jsme je do fronty F. Prozkoumali jsme první vrchol, označili jeho dosud neoznačené sousedy a přidali je do F. V dalším prozkoumávání jsme došli až k vrcholu 2. Objevili jsme květ. Změna grafu.

tabulka značení vrcholů:

Z1	*4*	*5*	*6*	*F*
S:0	*S:0*	*S:0*	*S:0*	*Z1*
*	*			

$Z1$ získal značku báze květu. Zařadili jsme ho do fronty. Prozkoumali jsme ho a nalezli jsme rozšiřitelnou střídavou cestu. Změna párování, setření značek, opakování značení. $M = \{(3,4)\}$

tabulka značení vrcholů:

1	*2*	*3*	*4*	*5*	*6*	*F*
S:0	*S:0*			*S:0*	*S:0*	*1,2,5,6*
*	*					*2,5,6*

Vrcholy nenasycené v párování jsme označili symbolem $S : 0$*. Uložili jsme je do fronty* F*. Prozkoumali jsme první vrchol a objevili jsme rozšiřitelnou střídavou cestu. Změna párování, setření značek, opakování značení.* $M = \{(1,2),(3,4)\}$.

tabulka značení vrcholů:

1	*2*	*3*	*4*	*5*	*6*	*F*
				S:0	*S:0*	*5,6*
	T:5	*T:5*				*6,2,3*
						2,3
S:2						*3,1*
			S:3			*1,4*
						4

Označili jsme všechny vrcholy, aniž jsme našli rozšiřitelnou střídavou cestu. Stávající párování je maximální.
□

Shrnutí kapitoly 7.4

V této kapitole jsme se seznámili s algoritmem pro nalezení maximálního párování v obecném grafu.

Otázky ke kapitole 7.4

1. Určete maximální párování v grafu G_1, je-li dáno počáteční párování M. Uveďte jednotlivé kroky algoritmu (v přehledné formě). Popište je. Jaká je ukončovací podmínka algoritmu pro nalezení max. párování? $G_1 = \{(1,2),(1,3),(1,4),(1,5),(2,3),(3,5),(4,5),(5,6)\}$; $M = \{(2,3),(4,5)\}$;
2. Určete maximální párování v grafu G_2, je-li dáno počáteční párování M. Uveďte jednotlivé kroky algoritmu (v přehledné formě). $G_2 = \{(1,5),(1,6),(2,3),(2,5),(3,4),(3,5),(4,5),(4,6),(5,6)\}$; $M = \{(2,3),(5,6)\}$;

Kapitola 8

Eulerovské tahy

Obsah

8.1 Pojmy, konstrukce eulerovských tahů

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět definovat pojem Eulerovský tah a pojem pokrytí grafu. Dále budete umět popsat princip algoritmu pro pro nalezení uzavřeného (otevřeného) eulerovského tahu na příkladu jej použít.

Eulerovský tah v grafu G je takový tah, který obsahuje všechny hrany grafu (každou hranu právě jednou). Eulerovské tahy dělíme na orientované a neorientované (v závislosti na orientovanosti hran v grafu) a na uzavřené a neuzavřené (zda je počáteční a koncový vrchol tahu totožný vrchol nebo dva různé vrcholy).

Řekneme, že soustava tahů v grafu G ***pokrývá*** hrany grafu G, 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 můžeme formulovat a řešit následující úlohy:

1. Máme rozhodnout, zda v daném grafu existuje eulerovský tah (uzavřený či otevřený). Pokud existuje, tak ho sestrojit.
2. V daném grafu G máme najít nejmenší počet tahů (nikoliv eulerovských), které pokrývají hrany daného grafu.
3. V daném souvislém grafu, jehož hrany jsou ohodnoceny kladnými čísly máme najít nejkratší uzavřený sled, který obsahuje (alespoň jednou) každou hranu grafu.

Věta 8.1 *(Euler 1736): Nechť graf G je souvislý. Pak v grafu G existuje neorientovaný uzavřený eulerovský tah právě tehdy, když každý vrchol grafu je sudého stupně.*

Věta 8.2 *Nechť graf G je souvislý. Pak v grafu G existuje orientovaný uzavřený eulerovský tah právě tehdy, když pro $\forall v \in G$ platí: $d^+(v) = d^-(v)$ (vstupní stupeň vrcholu je roven výstupnímu stupni).*

Důkaz. (konstruktivní): $\Rightarrow$ Předpokládejme, že v grafu existuje uzavřený eulerovský tah. Procházíme grafem podél tohoto tahu. Zřejmě platí, že kolikrát jsme přišli do určitého vrcholu v, tolikrát z něj musíme také odejít. Poněvadž přitom procházíme každou hranou právě jednou, plynou odtud okamžitě podmínky o stupních vrcholů: $d(v)$ je sudý pro neorientovaný tah a $d^+(v) = d^-(v)$ pro orientovaný.
$\Leftarrow$ Začneme z libovolného vrcholu v_0 a procházíme hranami grafu (libovolně) tak, abychom žádnou hranou neprošli dvakrát. Takto pokračujeme, dokud je to možné, tj. dokud z vrcholu, kde právě jsme, vede ještě nějaká dosud nepoužitá hrana. Toto procházení určitě skončí v v_0, neboť, díky vlastnostem stupňů vrcholů, nemůže skončit jinde. Nalezli jsme nějaký uzavřený tah.

Je-li tento tah již eulerovský, konec. Není-li eulerovský, pak díky souvislosti existuje vrchol v_1, kterým prochází tah a z něhož ještě vychází nějaká nepoužitá hrana. V tomto vrcholu původní tah přerušíme a začneme opět procházet nepoužitými hranami grafu. Díky vlastnostem stupňů se zase vrátíme do vrcholu v_1. V tomto vrcholu navážeme nový tah na původní, čímž získáme uzavřený tah, který má více hran.

Není-li tento tah ještě eulerovský, najdeme vrchol v_2, kterým tento tah prochází a z něhož vychází ještě nějaká nepoužitá hrana. Tah přerušíme, prodloužíme, navážeme ... ∎

Věta 8.3 *Nechť graf G je souvislý a obsahuje k vrcholů lichého stupně. Potom nejmenší počet neor. tahů pokrývajících hrany grafu G je roven $K/2$.*

Věta 8.4 *V orientovaném (neorientovaném) grafu G existuje otevřený eulerovský tah právě tehdy, když graf G je souvislý a existují v něm dva vrcholy v_1, v_2 takové, že $d^+(v_1) = d^-(v_1) + 1$ a $d^-(v_2) = d^-(v_2) - 1$, a pro všechny ostatní vrcholy v platí $d^+(v) = d^-(v)$ (dva vrcholy jsou lichého stupně, ostatní jsou stupně sudého).*

Algoritmus:
Pro nalezení uzavřeného eulerovského tahu.

1. Začneme z libovolného vrcholu v_0 a procházíme hranami grafu tak, abychom žádnou hranou neprošli dvakrát. Pokračujeme v procházení, dokud to je možné, tj. dokud z vrcholu, kde právě jsme, vede ještě nějaká dosud nepoužitá hrana. Toto procházení skončí v v_0, díky vlastnostem stupňů vrcholů. Nalezli jsme libovolný uzavřený tah.

2. Uzavřený tah kontrolujeme, zda je eulerovský tah. Při kontrole procházíme podél tahu a v každém vrcholu v testujeme, zda v $E(v)$ (případně v $E^+(v)$) existuje hrana h, která dosud neleží v tahu. Jestliže ano, pak přerušíme kontrolu, tah ve vrcholu v rozpojíme a začneme jej prodlužovat, počínaje hranou h. Prodlužování skončí ve vrcholu v. Po propojení nového a starého tahu pokračujeme v kontrole počínaje vrcholem v a postupujeme podél nové části tahu. Takto zajistíme, že jak při prodlužování, tak i při kontrole postupujeme podél každé hrany pouze jednou.

Animace: Eulerovské tahy

■ **Příklad 8.1** *V neorientovaném grafu G_1 určete počet tahů, které tvoří pokrytí a sestrojte je.*
$E(G_1) = \{(v_1, v_2), (v_1, v_4), (v_2, v_3), (v_2, v_4), (v_2, v_5), (v_3, v_4), (v_4, v_5)\}$

vrchol	*stupeň vrcholu*
v_1	*2*
v_2	*4*
v_3	*2*
v_4	*4*
v_5	*2*

Z tabulky stupňů vrcholů vidíme, že všechny vrcholy jsou sudého stupně. V daném grafu G_1 tedy existuje uzavřený eulerovský tah.

Postupně konstruované tahy:
T_1: $v_1 \to v_2 \to v_3 \to v_4 \to v_1$
kontrolujeme první tah a zjišťujeme, zda-li jsou všechny hrany zahrnuty do tahu;
T_{v_2}: $v_2 \to v_4 \to v_5 \to v_2$
Výsledný uzavřený eulerovský tah:
T: $v_1 \to v_2 \to v_4 \to v_5 \to v_2 \to v_3 \to v_4 \to v_1$
□

■ **Příklad 8.2** *V neorientovaném grafu G_1 určete počet tahů, které tvoří pokrytí a sestrojte je.*
$E(G_1) = \{(v_1, v_2), (v_1, v_4), (v_2, v_3), (v_2, v_4), (v_2, v_5), (v_3, v_4), (v_4, v_5)\}$

vrchol	*stupeň vrcholu*
v_1	*2*
v_2	*4*
v_3	*2*
v_4	*4*
v_5	*2*

Z tabulky stupňů vrcholů vidíme, že všechny vrcholy jsou sudého stupně. V daném grafu G_1 tedy existuje uzavřený eulerovský tah.

Postupně konstruované tahy:
T_1: $v_1 \to v_2 \to v_3 \to v_4 \to v_1$
kontrolujeme první tah a zji šťujeme, zda-li jsou všechny hrany zahrnuty do tahu;
T_{v_2}: $v_2 \to v_4 \to v_5 \to v_2$
Výsledný uzavřený eulerovský tah:
T: $v_1 \to v_2 \to v_4 \to v_5 \to v_2 \to v_3 \to v_4 \to v_1$
□

Shrnutí kapitoly 8.1

V této kapitole jsme se seznámili s pojmem pojem Eulerovský tah a pojem pokrytí grafu. Dále jsme se seznámili s principem algoritmu pro pro nalezení uzavřeného (otevřeného) eulerovského tahu.

Otázky ke kapitole 8.1

1. V neorientovaném grafu H určete počet tahů, které tvoří pokrytí a sestrojte je.
   $v_1 \longrightarrow v_8$
   $v_2 \longrightarrow v_3, v_6, v_8$
   $v_3 \longrightarrow v_4, v_7$
   $v_4 \longrightarrow v_2, v_5, v_6$
   $v_5 \longrightarrow v_2, v_3$
   $v_6 \longrightarrow v_1, v_2, v_5$
   $v_7 \longrightarrow v_4, v_6$
   $v_8 \longrightarrow v_4, v_7$

8.2 Konstrukce tahů, které tvoří pokrytí grafu

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat a použít postup vedoucí k nalezení tahů, které tvoří pokrytí grafu.

Algoritmus:

1. Určíme stupně vrcholů (pro orientovaný graf určíme vstupní a výstupní stupně) a podle počtu vrcholů lichého stupně určíme minimální počet tahů, které tvoří pokrytí (v orientovaných grafech nás zajímá rozdíl mezi vstupním a výstupním stupněm).
2. Vytvoříme příslušný počet otevřených tahů.
3. Kontrolujeme vytvořené tahy a případně je rozpojujeme a prodlužujeme pomocí dosud nepoužitých hran (viz. algoritmus na konstrukci uzavřených eulerovských tahů).

Animace: Pokrytí grafu

■ **Příklad 8.3** *V neorientovaném grafu G_2 určete počet tahů, které tvoří pokrytí a sestrojte je.*
$E(G_2) = \{(v_1, v_2), (v_1, v_5), (v_2, v_3), (v_2, v_6), (v_3, v_4), (v_3, v_7), (v_4, v_7), (v_4, v_8), (v_4, v_9), (v_5, v_6), (v_5, v_9), (v_6, v_7), (v_7, v_8)\}$

Vzhledem k tomu, že v daném grafu jsou čtyři vrcholy lichého stupně (jsou to vrcholy v_2, v_3, v_5 a v_6; všechny jsou stupně 3), existují v grafu právě dva otevřené neorientované tahy, které začínají nebo končí ve výše uvedených vrcholech.

První možný způsob konstrukce těchto tahů je následující: začínáme ve vrcholu lichého stupně a tah končíme v prvním vrcholu lichého stupně, na který narazíme. Stejně postupujeme s druhým otevřeným tahem. Pak provádíme kontrolu, zda jsou všechny hrany zahrnuty do tahu. Pokud ne, zkonstruujeme uzavřený tah, který rozšiřuje jeden z otevřených tahů.

Dva otevřené tahy:
T_1: $v_2 \to v_1 \to v_5$
T_2: $v_3 \to v_2 \to v_6$
Kontrola tahů: u vrcholu v_5 jsou dosud nepoužité hrany.
T_{v_5}: $v_5 \to v_6 \to v_7 \to v_3 \to v_4 \to v_7 \to v_8 \to v_4 \to v_9 \to v_5$
Výsledné tahy tvořící pokrytí:
T_1: $v_2 \to v_1 \to v_5 \to v_6 \to v_7 \to v_3 \to v_4 \to v_7 \to v_8 \to v_4 \to v_9 \to v_5$
T_2: $v_3 \to v_2 \to v_6$

Druhá z možností, jak sestrojit tahy pokrývající graf, je založena na tom, že tah začínající ve vrcholu lichého stupně prodlužujeme tak dlouho, dokud to je možné. Opět končíme v některém z vrcholů lichého stupně (ale až po vyčerpání všech hran, které s tímto vrcholem incidují).

Dva otevřené tahy:
T_1: $v_2 \to v_1 \to v_5 \to v_6 \to v_2 \to v_3 \to v_4 \to v_7 \to v_3$
T_2: $v_5 \to v_9 \to v_4 \to v_8 \to v_7 \to v_6$
Při kontrole vrcholů zjistíme, že všechny hrany grafu již byly zahrnuty do tahů. Tj. neexistuje uzavřený tah, který by rozšiřoval některý ze stávajících otevřených tahů.
□

■ **Příklad 8.4** *V orientovaném grafu G_3 určete počet tahů, které tvoří pokrytí a sestrojte je.*
$E(G_3) = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_4), (v_3, v_5), (v_5, v_6), (v_6, v_7), (v_7, v_2), (v_7, v_3)\}$

vrchol	*vstupní stupeň*	*výstupní stupeň*	*cesta?*
v_1	*0*	*1*	*Z cesty*
v_2	*2*	*2*	
v_3	*2*	*2*	
v_4	*2*	*0*	*2*K cecty*
v_5	*1*	*1*	
v_6	*1*	*1*	
v_7	*1*	*2*	*Z cesty*

Dva otevřené tahy:
T_1: $v_1 \to v_2 \to v_3 \to v_4$
T_2: $v_7 \to v_2 \to v_4$
Kontrola vrcholů a hran s nimi sousedících:
T_{v_3}: $v_3 \to v_5 \to v_6 \to v_7 \to v_3$
Výsledné tahy tvořící pokrytí orientovaného grafu G_3:
T_1: $v_1 \to v_2 \to v_3 \to v_5 \to v_6 \to v_7 \to v_3 \to v_4$
T_2: $v_7 \to v_2 \to v_4$
□

Shrnutí kapitoly 8.2

V této kapitole jsme se seznámili s postupem vedoucím k nalezení tahů, které tvoří pokrytí grafu.

Otázky ke kapitole 8.2

1. Určete všechny tahy, které pokrývají hrany orientovaného grafu G_3. Jaká je nutná a postačující podmínka pro existenci otevřeného neorientovaného eulerovského tahu?
   $G_3 = \{(1,7),(2,1),(2,4),(2,5),(3,2),(4,7),(5,4),(5,6),(6,2),(7,2)\}$.
2. Určete všechny tahy, které pokrývají hrany orientovaného grafu G_4.
   $G_4 = \{(1,4),(1,6),(2,3),(3,1),(3,4),(4,2),(4,5),(5,6),(6,2),(6,3)\}$.

8.3 Úloha čínského pošťáka

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat postup řešení tzv. úlohy čínského pošťáka.

Pošťák musí při roznášce pošty projít alespoň jedenkrát každou ulici svého rajónu. Jak má postupovat, aby ušel co nejméně kilometrů, tj. aby dvakrát procházel co nejkratší úseky?

Řešení této úlohy je shodné s nalezením nejkratšího uzavřeného sledu v kladně ohodnoceném souvislém grafu. Tento sled musí obsahovat (alespoň jednou) každou hranu.

Algoritmus:

1. Pro každou dvojici vrcholů lichého stupně u, v vypočteme délku nejkratší (viz kapitola 4) cesty (z u do v). Označíme tuto délku $l(u, v)$.

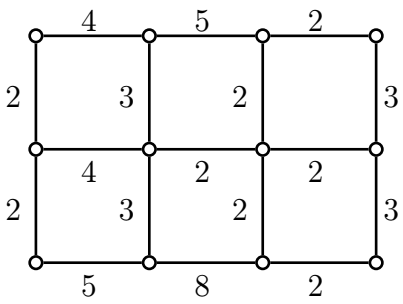
2. Na množině $L = \{$vrcholy grafu G, které jsou lichého stupně$\}$ definujeme kompletní graf K, jehož hrany jsou ohodnoceny graf délkami $l(u, v)$ pro $\forall u, v \in L$. V tomto grafu nalezneme nejlevnější perfektní párování M (viz kapitola 7.3).

3. Pro každou dvojici vrcholů u, v, která je spojena hranou z párování M, přidáme ke grafu G kopie hran, které tvoří nejkratší cestu mezi vrcholy u, v. V grafu G', který vznikne přidáním všech těchto hran, budou mít všechny vrcholy sudý stupeň.

4. V získaném grafu G' nalezneme eulerovský tah (viz kapitola 8.1). Tento tah prochází všemi přidanými hranami, což odpovídá opakovaným průchodům některými hranami původního grafu G. Nahradíme-li v eulerovském tahu všechny přidané hrany původními, získáme hledaný nejkratší sled, který pokrývá všechny hrany grafu G.

Shrnutí kapitoly 8.3

V této kapitole jsme se seznámili s postupem vedoucím k vyřešení úlohy čínského pošťáka.



Obrázek 8.1: Graf pro úlohu z kapitoly 8.3.

Otázky ke kapitole 8.3

1. Vyřešte úlohu čínského pošťáka pro graf na obrázku 8.1.

Kapitola 9

Barvení grafu

Obsah

V této kapitole se budeme zabývat vlastně jen jedním algoritmem. Bude to algoritmus vrcholového obarvení grafu G. Sice nás bude zajímat i problém hranového obarvení grafu G, ale tuto úlohu převedeme na úlohu vrcholového barvení odvozeného grafu δG.

9.1 Vrcholové barvení grafu

Časová náročnost kapitoly: 40 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat a použít algoritmy pro vrcholové a hranové barvení grafu.

Vrcholové barvení grafu G je zobrazení z množiny $V(G)$ do množiny $\{1, 2, \ldots, k\}$, nazývané **množinou barev**. (barev) tak, že žádné dva vrcholy spojené hranou nejsou ohodnoceny (obarveny) stejnou barvou. Barevnost grafu (***chromatické číslo*** grafu) je nejmenší počet barev, který je potřebný k obarvení grafu. Barevnost grafu G značíme $\chi(G)$.

Hranové barvení grafu G je zobrazení z množiny $E(G)$ do množiny $\{1, 2, \ldots, k\}$, nazývané **množinou barev**. Hranová barevnost je nejmenší počet barev potřebných k obarvení hran. U barvení hran požadujeme, aby každé dvě hrany, které mají společný vrchol, byly obarveny různými barvami. ***Chromatický index*** $\chi'(G)$ je nejmenší číslo k, pro které je G hranově obarvitelný.

Vyšetřování hranové barevnosti grafu G snadno převedeme na vyšetřování vrcholové barevnosti tzv. hranového grafu δG. Hranový graf δG získáme takto: za vrcholy grafu δG prohlásíme hrany původního grafu G. Dva vrcholy v hranovém grafu jsou spojeny hranou, jestliže jim odpovídající hrany původního grafu G měly společný vrchol.

Věta 9.1 *V prostém grafu G bez smyček platí: $\chi(G) \leq max\{d(v); v \in V(G)\} + 1$.*

Věta 9.2 *Pro každý neorientovaný prostý graf G platí: $d \leq \chi'(G) \leq d + 1$, kde d je maximální stupeň vrcholů.*

Algoritmus:
Vrcholové barvení grafu. Graf G je reprezentován seznam vrcholů a jejich sousedů s nižším indexem. Výstupem tohoto algoritmu je pole $Barva$, přičemž $Barva_v$ je barva (tj. číslo barvy), kterou je vrchol v obarven. Hodnota proměnné $Omez$, je určena barevností grafu G (počet použitých barev v poli $Barva$).

1. Obarvíme první vrchol - položíme $k := 1$, $B(k) := 1$, $Omez := n + 1$.
2. Pokus o obarvení dalšího vrcholu nejnižší možnou barvou. Položíme $k := k + 1$.
   Jestliže $k > n$, potom bod 3. Jinak $B_k :=$ {minimum ze všech barev, které se nevyskytují u sousedů vrcholu k}.
   Jestliže $B_k \geq Omez$, pak $y := k$ a pokračujeme bodem 4. Jinak opakujeme bod 2.

3. Pro $\forall v \in V(G)$ přiřadíme $Barva_v := B_v$; $Omez := max\{B_k; \forall k\}$; $y :=$ první vrchol obarvený barvou $Omez$.

4. Pokusíme se snížit barvu ve vrcholu y. Označíme si k posledního souseda vrcholu y s nižším indexem.

5. Pokusíme se zvýšit barvu ve vrcholu k. Jestliže $k = 1$, potom bod 6. Jinak položíme $b := min\{c > B_k$ nevyskytujících se v okolí vrcholu $k\}$. Jestliže $b < Omez \wedge b \leq k$, pak $B_k := b$ a pokračujeme bodem 2. Jinak položíme $k := k - 1$ a pokračujeme bodem 5.

6. Konec.

Animace: Barvení grafu

■ **Příklad 9.1** *V grafu G_1 určete vrcholové barvení.*
$E(G_1) = \{(v_1, v_2), (v_1, v_3), (v_1, v_5), (v_1, v_6), (v_2, v_4), (v_2, v_5), (v_3, v_6), (v_3, v_7), (v_4, v_5), (v_5, v_6), (v_5, v_7), (v_6, v_7)\}$

Reprezentace grafu seznamem vrcholů a jejich sousedů s nižším indexem:

$v_1 \rightarrow$
$v_2 \rightarrow v_1$
$v_3 \rightarrow v_1$
$v_4 \rightarrow v_2$
$v_5 \rightarrow v_1, v_2, v_4$
$v_6 \rightarrow v_1, v_3, v_5$
$v_7 \rightarrow v_3, v_5, v_6$

Tabulka reprezentující změny při barvení grafu:

Omez	v_1	v_2	v_3	v_4	v_5	v_6	v_7
8	*1*	*2*	*2*	*1*	*3*	*4*	*1*
4				*3*	*4*		
4			*3*	*1*	*3*	*2*	*1*
3	*1*	*2*	*3*	*1*	*3*	*2*	*1*

□

■ **Příklad 9.2** *V grafu G_1 určete hranové barvení.*

Reprezentace hranového grafu seznamem vrcholů a jejich sousedů s nižším indexem:

$h_1 \rightarrow$
$h_2 \rightarrow h_1$
$h_3 \rightarrow h_1, h_2$
$h_4 \rightarrow h_1$
$h_5 \rightarrow h_1, h_3, h_4$
$h_6 \rightarrow h_1, h_4, h_5$
$h_7 \rightarrow h_5, h_6$
$h_8 \rightarrow h_3, h_5, h_7$
$h_9 \rightarrow h_3, h_5, h_7, h_8$
$h_{10} \rightarrow h_6, h_7, h_9$
$h_{11} \rightarrow h_8, h_9, h_10$
$h_{12} \rightarrow h_2, h_4, h_6, h_7, h_10$

Tabulka reprezentující změny při barvení grafu:

Omez	*13*	*5*
h_1	*1*	
h_2	*2*	
h_3	*3*	
h_4	*2*	
h_5	*4*	
h_6	*3*	
h_7	*1*	
h_8	*2*	
h_9	*5*	
h_{10}	*2*	
h_{11}	*1*	
h_{12}	*4*	

Algoritmus byl ukončen, protože v grafu G_1 byl nejvyšší stupeň vrcholu roven 5 (takže minimální počet barev potřebný pro hranové barvení je 5).
□

Shrnutí kapitoly 9.1

V této kapitole jsme se seznámili s pojmy vrcholové a hranové barvení grafu a s algoritmy pro jejich nalezení.

Otázky ke kapitole 9.1

1. Určete vrcholové barvení grafu G_2. Postup algoritmu zapište do tabulky. Jaké jsou odhady pro $\chi(G_2)$?
   $G_2 = \{(1,2),(1,3),(1,5),(1,6),(1,8),(2,4),(2,7),(2,8),(3,5),(3,6),(4,7),(5,7),(5,8),(6,7),(6,8),(7,8)\}$
2. Určete hranové barvení grafu G_3. Postup algoritmu zapište do tabulky. Jaké jsou odhady pro $\chi'(G_3)$?
   $G_3 = \{(1,3),(1,5),(1,6),(2,3),(3,4),(3,5),(4,6)(5,6)\}$.

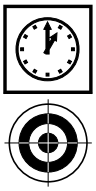
Kapitola 10

Maximální nezávislé množiny

Obsah

V této kapitole je popsán algoritmus pro vyhledání všech maximálních nezávislých množin založený na backtrackingu. Po jednoduché úpravě se může použít k hledání nejpočetnější nezávislé množiny, a tím také k výpočtu nezávislosti daného grafu. V grafech s ohodnocenými vrcholy lze tento algoritmus použít i k nalezení nejdražší nezávislé množiny.

10.1 Určení všech maximálních nezávislých množin



Časová náročnost kapitoly: 35 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět definovat pojem nezávislá a maximální nezávislá množina. Dále se seznámíte s jedním algoritmem pro nalezení všech maximálních nezávislých množin v grafu.

Metoda pro nalezení maximálních nezávislých množin spočívá ve vyzkoušení všech možností, které přicházejí v úvahu. ***Nezávislá množina*** vrcholů v grafu G je taková množina, jejíž žádné dva vrcholy nejsou spojeny hranou. ***Maximální nezávislá množina*** je taková nezávislá množina, ke které již nelze přidat další vrchol, aniž by přestala být nezávislou. ***Nejpočetnější*** nezávislá množina je taková, která má největší počet prvků (mezi všemi nezávislými množinami v grafu G). Počet prvků v největší nezávislé množině grafu G nazýváme ***nezávislost grafu*** G a značíme ji $\alpha(G)$.

V každé fázi výpočtu budeme mít k-prvkovou nezávislou množinu $S_k = \{S^1, S^2, \ldots S^k\}$. Počet prvků k v nezávislé množině bude v průběhu práce algoritmu kolísat. Budeme přidávat a ubírat vrcholy.

V algoritmu použijeme dva systémy množin vrcholů: QS_i a QN_i pro $\forall i = 0, 1, \ldots, k$. Přitom bude platit, že $[QS_i \cup QN_i] \cap S_i = \emptyset$.

QN_i = množina všech vrcholů v, které dosud k množině S_i nebyly přidány, tj. žádná nezávislá množina obsahující $S_i \cup \{v\}$ nebyla zkoumána.

QS_i = množina všech vrcholů v, jejichž přidání k S_i již bylo vyzkoušeno, tj. všechny maximální nezávislé množiny obsahující množinu $S_i \cup \{v\}$ již byly prozkoumány.

Algoritmus se vždy snaží rozšířit stávající množinu S_k tím, že vybere vrchol $u \in QN_k$ a položí $S_{k+1} := u$. Nelze-li nezávislou množinu rozšířit, vyřazuje poslední zařazený vrchol S^k a snaží se rozšířit takto vzniklou menší množinu. Přitom provádíme (dále popsané) úpravy množin QN_i a QS_i tak, aby platily výše uvedené podmínky.

Je zřejmé, že nezávislá množina S_k je maximální $\Leftrightarrow QN_k = QS_k = \emptyset$.

Je-li $QN_k = \emptyset$ a $QS_k \neq \emptyset$, potom musíme množinu S_k zmenšit, protože zvětšit ji nelze. Přidáním vrcholu z QS_k bychom dostali jenom to, co jsme již dříve prozkoumali.

Podobně, existuje-li vrchol $v \in QS_k$ takový, že $V(v) \cap QN_k = \emptyset$, je další zvětšování zbytečné, neboť každé nezávislé množině, kterou bychom takto dostali, by k maximálnosti scházel vrchol v.

Algoritmus:

1. Inicializace: Položíme $k := 0$, $QS_0 = \emptyset$, $QN_0 = V(G)$.
2. Rozšíření nezávislé množiny - Vybereme náhodně vrchol $u \in QN_k$. Položíme: $k := k + 1$, $S_k := u$, $QS_k := QS_{k-1} - V(u)$, $QN_k := QN_{k-1} - [V(u) \cup \{u\}]$
3. Test maximálnosti nezávislé množiny - Jestliže $QS_k = QN_k = \emptyset$, vytiskneme seznam maximální nezávislé množiny $S^1, S^2, \ldots S^k$ a pokračujeme bodem 6.
4. . Test možnosti zvětšování - Jestliže $QN_k = \emptyset$, pokračujeme bodem 6.
5. Test, zda lze S_k doplnit na maximální nezávislou množinu - Jestliže existuje vrchol $v \in QS_k$ takový, že $V(v) \cap QN_k = \emptyset$, pokračujeme bodem 6. Jinak pokračujeme bodem 2.
6. Návrat, tj. zmenšení o naposledy přidaný vrchol - Jestliže $k = 0$, výpočet končí. Položíme $u := S^k$, $k := k - 1$ a přeřadíme u z množiny QN_k do QS_k. Pokračujeme bodem 4

Animace: Maximální nezávislé množiny

■ **Příklad 10.1** *Určete všechny maximální nezávislé množiny v grafu* G. $E(G) = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_3, v_4), (v_3, v_5)\}$. *Značení:*

- S_k *- množina nezávislých vrcholů* $S^1, \ldots, S^k$
- k *- počet vrcholů v* S_k
- QN_i *- množina neprozkoumaných, vhodných a do* S_k *dosud nezařazených vrcholů*
- QS_i *- množina prozkoumaných a z* S_k *již vyřazených vrcholů*
- QX_i *- protože* $QS_i \cap QN_i = \emptyset$*, tak* $QS_i \cup QN_i = QX_i$
- N *- vrchol s touto značkou* $\in QN_i$
- S *- vrchol s touto značkou* $\in QS_i$
- $+$ *- vrchol s touto značkou* $\in S_k$
- *. - vrchol s touto značkou nepatří do žádné z výše uvedených množin (většinou to je vrchol sousední s vrcholem z* S_k*)*

k	S_k	QX_0	QX_1	QX_2	QX_3
0	$\emptyset$	*NNNNN*			
1	v_1		*+..NN*		
2	v_1, v_4			*+..+N*	
3	v_1, v_4, v_5				*+..++*
2	v_1, v_4			*+..+S*	
1	v_1		*+..SN*		
0	$\emptyset$	*SNNNN*			
1	v_2		*.+N.N*		
2	v_2, v_3			*.++..*	
1	v_2		*.+S.N*		
2	v_2, v_5			*.+..+*	
1	v_2		*.+S.S*		
0	$\emptyset$	*SSNNN*			
1	v_3		*.S+..*		
0	$\emptyset$	*SSSNN*			
1	v_4		*S..+N*		
0	$\emptyset$	*SSSSN*			
1	v_5		*SS.S+*		
0	$\emptyset$	*SSSSS*			

Byly nalezeny tři maximální nezávislé množiny $\{v_1, v_4, v_5\}, \{v_2, v_3\}, \{v_3, v_5\}$. První z nich je nejpočetnější nezávislou množinou v grafu G.

□

Shrnutí kapitoly 10.1

V této kapitole jsme se seznámili s pojmy nezávislá a maximální nezávislá množina a dále s algoritmem pro nalezení všech maximálních nezávislých množin v grafu.

Otázky ke kapitole 10.1

1. Určete všechny maximální nezávislé množiny v grafu H.
   $E(H) = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_6), (v_6, v_1)\}$.

Kapitola 11

Toky v sítích

Obsah

11.1 Pojmy

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět vysvětlit pojmy tok v síti, cirkulace, tok od zdroje ke spotřebiči, kapacita řezu.

Toky v sítích reprezentují širokou škálu problémů, které se často vyskytují v informatice, komunikacích a dalších oborech. Používají se zde jednoduché algoritmy, které řeší stejným způsobem propustnost (maximální vytížitelnost) počítačových sítí, dopravních či kanalizačních sítí, minimální tok danou síti (která například reprezentuje systém topných těles, kterými musí protékat alespoň minimální množství teplé kapaliny, aby nedošlo k nevratnému poškození v důsledku nízkých teplot) nebo nejlevnější tok v síti s ohodnocením hran (což jistě ocení ekonomové při svých úvahách).

Tokem v síti G nazýváme libovolné ohodnocení hran $f : E(G) \to Z$, které v každém vrcholu $v \in V(G)$ splňuje Kirchhoffův zákon:

$$\sum_{e \in E^+(v)} f(e) = \sum_{e \in E^-(v)} f(e).$$

Tato podmínka nám vlastně říká, že množství kapaliny, které do vrcholu přitéká, je rovno množství, které z vrcholu odtéká. Tok v každé hraně je ustálený a beze ztrát. Orientace hrany určuje směr proudění, záporná velikost toku pak odpovídá proudění proti směru hrany.

Toky rozlišujeme na:

- cirkulace - splňuje Kirchhoffův zákon pro všechny vrcholy,
- tok od zdroje ke spotřebiči - splňuje Kirchhoffův zákon pro všechny vrcholy kromě dvou. Tyto vrcholy nazýváme zdrojem a spotřebičem. Ve ***zdroji*** tok vzniká a ve ***spotřebiči*** stejné množství toku zaniká. Tok od zdroje ke spotřebiči lze snadno převést na cirkulaci přidáním tzv. návratové hrany ze spotřebiče ke zdroji.

Velikost toku $f(e)$ může být v úloze omezena na 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 $l(e) \leq f(e) \leq c(e)$ pro $\forall e \in E(G)$, nazýváme přípustným tokem.

Má-li síť zdroj a spotřebič a jsou-li dolní omezení toku nulová, nazýváme tuto síť *transportní sítí*.

Je-li $A \subset V(G)$ množina vrcholů, pak řez určený množinou A značíme $W(A)$. Číslo

$$\sum_{e \in W^+(A)} c(e) - \sum_{e \in W^-(A)} l(e)$$

nazýváme ***kapacitou řezu*** určeného množinou A, kde

$$W^+(A) = \{e \in E(G); PV(e) \in A, KV(e) \notin A\}$$
$$W^-(A) = \{e \in E(G); KV(e) \in A, PV(e) \notin A\}.$$

Shrnutí kapitoly 11.1

V této kapitole jsme se seznámili s pojmy tok v síti, cirkulace, tok od zdroje ke spotřebiči, kapacita řezu.

Otázky ke kapitole 11.1

1. Co je to horní a dolní omezení toku? Uveďte reálné situace, které bychom mohli potřebovat řešit pomocí toků v sítích, a jaké by byly horní a dolní omezení toku.

2. Jaký je rozdíl mezi cirkulací a tokem od zdroje ke spotřebiči?

11.2 Maximální tok v síti

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat a použít algoritmus pro nalezení maximálního toku v síti.

Je dán orientovaný graf G a dva jeho vrcholy, zdroj z a spotřebič s a kladné kapacity hran. Úkolem je najít maximální přípustný tok od zdroje ke spotřebiči, tj. takový přípustný tok f, že výraz

$$\sum_{e \in E^+(z)} f(e)$$

je maximální.

S úlohou maximálního toku souvisí pojem řezu. Řekneme, že řez určený množinou A odděluje zdroj a spotřebič, jestliže zdroj leží v A a spotřebič neleží v A.

Věta 11.1 *(Ford-Fulkersonova věta o maximálním toku a minimálním řezu): Velikost maximálního toku od zdroje ke spotřebiči je rovna kapacitě minimálního řezu oddělujícího zdroj a spotřebič.*

Důkaz. Plyne z později uvedeného algoritmu. ∎

Hranu nazveme hranou vpřed, je-li orientována ve směru průchodu cestou. Naopak hrana vzad je orientována proti směru průchodu cestou.

Předpokládejme, že v síti s omezením toku l a c máme dán přípustný tok f (např. v transportní síti to bude nulový tok). ***Zlepšující cesta*** vzhledem k toku f je taková neorientovaná cesta ze zdroje z do spotřebiče s, že každá hrana e cesty splňuje:
- je-li hranou vpřed, pak $f(e) < c(e)$,
- je-li hranou vzad, pak $f(e) > l(e)$.

Kapacitu zlepšující cesty definujeme jako minimum z rozdílů $c(e) - f(e)$ na hranách vpřed a z rozdílů $f(e) - l(e)$ na hranách vzad (je to tedy vždy kladné číslo - označme ho d).

Je zřejmé, že existuje-li zlepšující cesta vzhledem k přípustnému toku f, pak na hranách vpřed lze tok zvýšit a na hranách vzad snížit o hodnotu d rovnou kapacitě zlepšující cesty. Přípustnost toku i Kirchhoffův zákon tím zůstanou zachovány, ale celková velikost toku od zdroje ke spotřebiči se zvýší o d.

Algoritmus:
Značkovací procedura. Zlepšující cestu budeme hledat postupem odvozeným od základního algoritmu na prohledávání grafu.

1. Označíme vrchol z, ostatní vrcholy jsou beze značek.
2. Je-li označen vrchol s, značkovací procedura končí.
3. Existuje-li hrana e taková, že $PV(e)$ má značku a $KV(e)$ nemá značku a $f(e) < c(e)$, pak označíme $KV(e)$, položíme ODKUD(KV(e)):=+e a pokračujeme bodem 2. Neexistuje-li taková hrana, pokračujeme bodem 4.
4. Existuje-li hrana e taková, že $KV(e)$ má značku a $PV(e)$ nemá značku a $l(e) < f(e)$, pak označíme $PV(e)$, položíme ODKUD(PV(e)):=-e a pokračujeme bodem 2. Neexistuje-li taková hrana, značkovací procedura končí, neboť nelze označit žádný další vrchol.

Poznámka:
Označíme-li vrchol s, pak existuje zlepšující cesta z vrcholu z do vrcholu s a lze ji sestrojit zpětným postupem pomocí hodnot ODKUD.

Poznámka:
Použijeme-li ve značkovací proceduře metodu prohledávání grafu do šířky, získáme vždy zlepšující cestu s nejmenším počtem hran.

Algoritmus:
Ford-Fulkersonův algoritmus pro maximální tok. Známe nějaký přípustný tok f a cílem je nalézt maximální tok od zdroje z ke spotřebiči s (zároveň nalezneme minimální řez oddělující z a s).

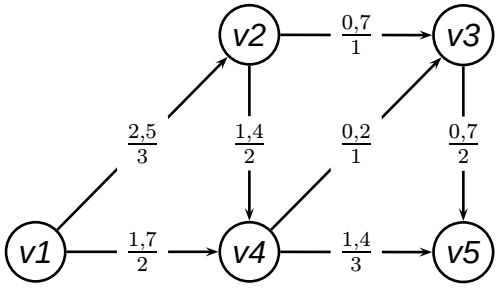
1. Použijeme značkovací proceduru k nalezení zlepšující cesty z vrcholu z do vrcholu s.
2. Jestliže zlepšující cesta neexistuje (nebyl označen spotřebič s), pak výpočet končí. Tok f je maximální a označíme-li A množinu označených vrcholů, pak $W_G(A)$ je minimální řez.
3. Jestliže zlepšující cesta existuje, vypočteme její kapacitu d.
4. Položíme $f(e) := f(e) + d$ pro hrany vpřed a $f(e) := f(e) - d$ pro hrany vzad a pokračujeme bodem 1

Poznámka:
Zajímá-li nás minimální tok od zdroje ke spotřebiči v netransportní síti, pak můžeme použít algoritmus pro nalezení maximálního toku s upravenými podmínkami. Pro průchod hranou vpřed (je-li $l(e) < f(e)$) nebo průchod hranou vzad (je-li $f(e) < c(e)$).

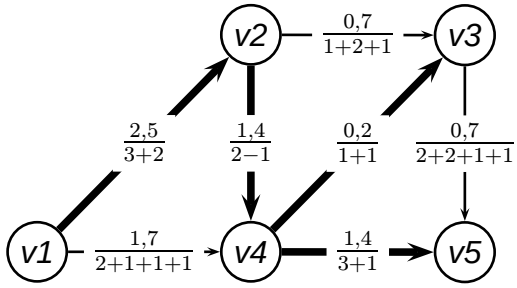
Poznámka:
Maximální párování v bipartitním grafu $G(U, V)$, můžeme převést na úlohu nalezení maximálního toku od zdroje ke spotřebiči. Síť je dána bipartitním grafem (orientace je dána z U do V), kde ke každému vrcholu $u \in U$ vede hrana s jednotkovou kapacitou od přidaného zdroje z a od každého vrcholu $v \in V$ vede hrana s jednotkovou kapacitou do přidaného spotřebiče s. Pak zlepšující cesta v síti odpovídá střídavé cestě, podél které můžeme rozšířit párování.

Animace: Ford-Fulkersonův algoritmus

■ **Příklad 11.1** *Určete maximální tok od zdroje (v_1) ke spotřebiči (v_5) v zadané síti G_1.*



Původní síť G_1.



Síť s maximálním tokem.

Zlepšující cesty ze zdroje ke spotřebiči (které se pak projeví zvýšením toku podél těchto cest - jak vidíme v obrázku sítě s maximálním tokem):

v_1 — *2* → v_2 — *6* → v_3 — *5* → v_5 *d=2*

v_1 — *5* → v_4 — *1* → v_3 — *3* → v_5 *d=1*

v_1 — *4* → v_2 — *1* → v_5 *d=1*

v_1 — *3* → v_4 — 1^- → v_2 — *4* → v_3 — *2* → v_5 *d=1*

v_1 — *2* → v_4

Silně vyznačené hrany jsou hranami nasycenými a tvoří minimální řez oddělující zdroj v1 od spotřebiče v5. Jeho kapacita je 10. Kapacita maximálního toku je rovna 10.

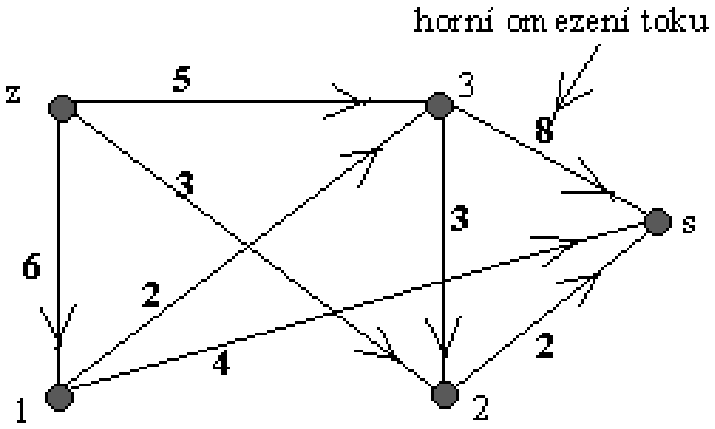
□

Shrnutí kapitoly 11.2

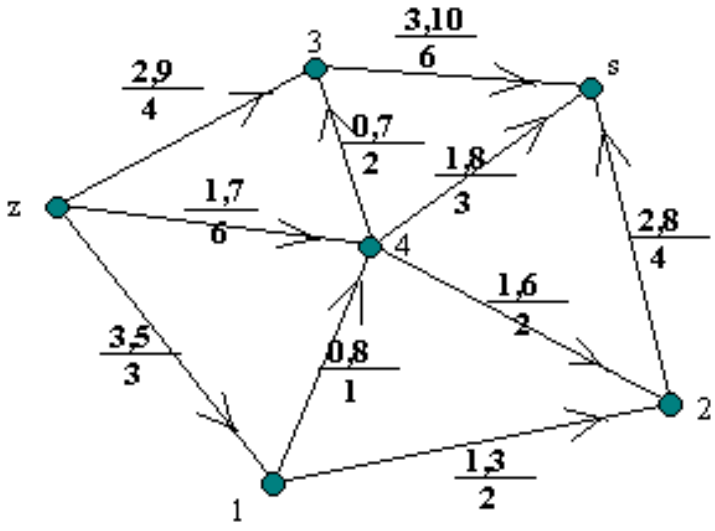
V této kapitole jsme se seznámili s Ford-Fulkersonovým algoritmem pro nalezení maximálního toku.

Otázky ke kapitole 11.2

1. Nalezněte maximální tok v grafu na obr. 11.1.
2. Nalezněte maximální tok v grafu na obr. 11.2.



Obrázek 11.1: Graf pro pro cvičení 1.



Obrázek 11.2: Graf pro pro cvičení 2.

11.3 Přípustná cirkulace

Časová náročnost kapitoly: 35 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět popsat a použít dva algoritmy pro nalezení přípustné cirkulace, jeden používá tzv. rozšířenou síť, druhý tzv. zlepšující kružnice.

V předchozí části jsme předpokládali, že známe přípustný tok (který jsme konkrétně znali pouze v případě transportní sítě - nulový tok = přípustný tok). V této části uvedeme algoritmus pro určení přípustného toku. Algoritmus používá značkovací proceduru z předchozího algoritmu.

Úloha nalezení přípustného toku je ekvivalentní úloze nalezení přípustné cirkulace. V síti pouze přidáme návratovou hranu od spotřebiče ke zdroji, která bude mít neomezující kapacitu a tím získáme cirkulaci.

Věta 11.2 *V síti s omezeními toku $l(e)$ a $c(e)$ v každé hraně existuje přípustná cirkulace právě tehdy, když každý řez $W(A)$ má nezápornou kapacitu, tj.*

$$\sum_{e \in W^+(A)} c(e) - \sum_{e \in W^-(A)} l(e) \geq 0$$

Poznámka:
Prakticky to znamená, že přípustná cirkulace existuje, když pro každou množinu vrcholů platí, že tok, který povinně vteče dovnitř díky dolnímu omezení $l(A)$, může z této množiny i odtéci díky hornímu omezení $c(A)$. Jestliže tedy přípustná cirkulace neexistuje, brání toku řez se zápornou kapacitou.

Algoritmus:
Pomocí zlepšujících kružnic. Na začátku si zvolíme libovolnou cirkulaci f (například i nulovou).

1. Najdeme hranu h s nepřípustným tokem. Neexistuje-li taková hrana, výpočet končí, tok je přípustný.
2. Jestliže $f(h) < l(h)$, pak položíme $z := KV(h)$, $s := PV(h)$. V opačném případě:
   jestliže $f(h) > c(h)$, pak položíme $z := PV(h)$, $s := KV(h)$.
3. Použijeme značkovací proceduru pro hledání zlepšující cesty z vrcholu z do vrcholu s. Jestliže není možné označit vrchol s, pak výpočet končí, přípustný tok neexistuje.

4. Zlepšující cestu doplníme o hranu h, čímž vznikne zlepšující kružnice. Hranu h v ní budeme pokládat za hranu vpřed, pokud $f(h) < l(h)$, případně za hranu vzad, pokud $f(h) > c(h)$.

5. Vypočteme kapacitu d zlepšující kružnice jako minimum z rozdílu $c(e) - f(e)$ pro hrany vpřed a $f(e) - l(e)$ pro hrany vzad.

6. Provedeme změny toku $f(e) := f(e) + d$ pro hrany vpřed a $f(e) := f(e) - d$ pro hrany vzad. Pokračujeme bodem 1.

Animace: Přípustná cirkulace pomocí zlepšujících kružnic

Algoritmus:
Nalezení přípustné cirkulace pomocí upravené sítě. Další algoritmus pro nalezení přípustné cirkulace pracuje s upravenou síti. Hledání přípustné cirkulace v síti G převedeme na hledání maximálního toku od zdroje ke spotřebiči v síti G' s horním omezením hran $c'(e)$ a s nulovým dolním omezením toku. Síť G' vznikne takto:

1. Pro $\forall e \in E(G)$ $c'(e) := c(e) - l(e)$, $l(e) = 0$

2. K síti G přidáme dva nové vrcholy, umělý zdroj z a umělý spotřebič s.

3. Pro $\forall v \in V(G)$ vypočteme

$$L(v) = \sum_{e \in E^+(v)} l(e) - \sum_{e \in E^-(v)} l(e)$$

   (počítá se vzhledem k l(e) v původním grafu G).

4. Pro $\forall v$ takové, že $L(v) > 0$ přidáme hranu o kapacitě $L(v)$ z vrcholu v do spotřebiče.

5. Pro $\forall v$ takové, že $L(v) < 0$ přidáme hranu o kapacitě $-L(v)$ ze zdroje do vrcholu v.

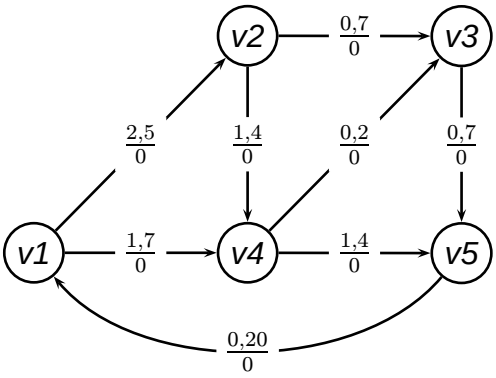
Poznámka:
Intervaly $\langle l(e), c(e) \rangle$ pro přípustný tok se posunou na $\langle 0, c'(e) \rangle$ a chybějící nebo přebývající tok se vyrovná v přidaných hranách. V takto vytvořené síti G' nalezneme libovolným způsobme maximální tok f' od zdroje z ke spotřebiči s. Lze ukázat, že platí: jestliže maximální tok f' nasycuje všechny hrany vycházející ze zdroje z, pak v síti G existuje přípustná cirkulace f, splňující $f(e) = f'(e) + l(e)$. Jestliže maximální tok f' nenasycuje všechny hrany vycházející ze zdroje , pak přípustná cirkulace v šíti G neexistuje.

Označíme-li A množinu vrcholů označených v algoritmu pro nalezení maximálního toku, pak řez $W(A - z)$ je řezem, který brání existenci přípustné cirkulace.

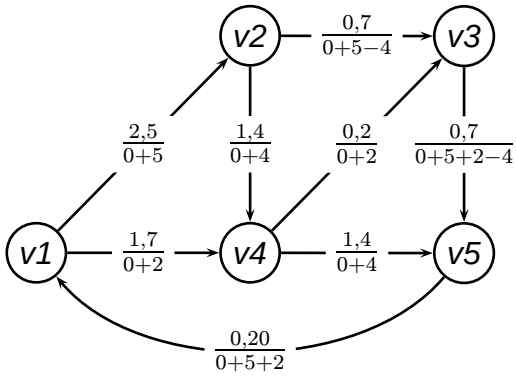
Animace: Přípustná cirkulace pomocí upravené sítě

■ **Příklad 11.2** *Určete přípustnou cirkulaci v dané síti G_2.*

1. *Pomocí zlepšujících kružnic.*



Původní síť G_2.



Síť s přípustnou cirkulací.

Zlepšující cesty z pomocného zdroje k pomocnému spotřebiči + hrana mezi pom. spotřebičem a pom. zdrojem (vznikne kružnice, podél které zvyšujeme tok - přesun nepřípustných toků do mezí přípustnosti):

$z_1 = v_2 \xrightarrow{7} v_3 \xrightarrow{7} v_5 \xrightarrow{20} v_1 = s_1$ *d=5*

5

$z_2 = v_4 \xrightarrow{2} v_3 \xrightarrow{2} v_5 \xrightarrow{15} v_1 = s_2$ *d=2*

7

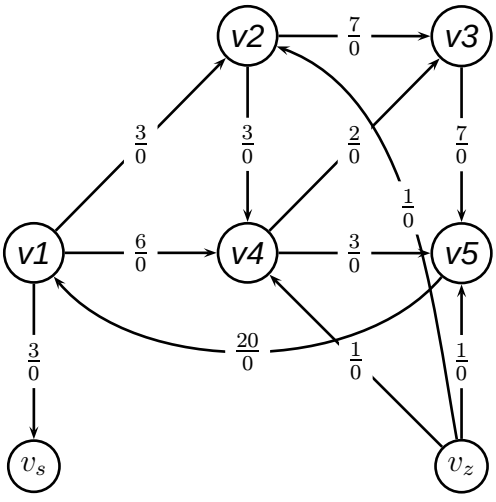
$z_3 = v_4 \xrightarrow{4} v_5 \xrightarrow{13} v_1$

7^-

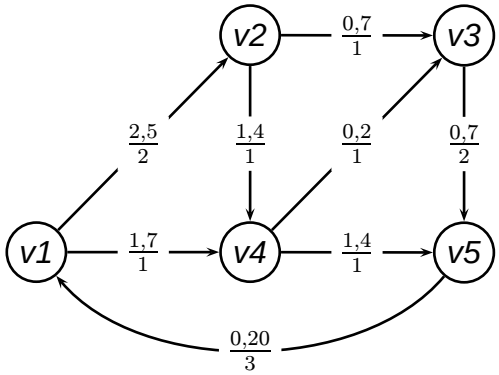
$v_3 \xrightarrow{5^-} v_2 = s_3$ *d=4*

4

2. *Pomocí pomocné rozšířené sítě.*



Rozšířená transportní síť.



Síť s přípustnou cirkulací.

Ceny doplňujících hran vedoucích od umělého zdroje (nebo do umělého spotřebiče) k příslušnému vrcholu (z příslušného vrcholu): $L_1 = 3$, $L_2 = -1$, $L_3 = 0$, $L_4 = -1$, $L_5 = -1$.

Zlepšující cesty z umělého zdroje do umělého spotřebiče:

v_z — 1 → v_2 — 7 → v_3 — 7 → v_5 — 20 → v_1 — 3 → v_s d=1

v_z — 1 → v_4 — 2 → v_3 — 6 → v_5 — 19 → v_1 — 2 → v_s d=1

v_z — 1 → v_5 — 18 → v_1 — 1 → v_s d=1

□

Shrnutí kapitoly 11.3

V této kapitole jsme se seznámili se dvěma algoritmy pro nalezení přípustné cirkulace, jeden používá tzv. rozšířenou síť, druhý tzv. zlepšující kružnice.

Otázky ke kapitole 11.3

1. Určete přípustný tok v síti G_2 (pokud existuje) pomocí upravené (rozšířené) sítě. Postup algoritmu vhodně zapište. Jaká je podmínka existence přípustného toku? Jaký bude výsledek tohoto algoritmu, pokud v síti neexistuje přípustný tok?
   $G_2 = \{(v1, v2; 0, 5), (1, 3; 3, 9), (2, 4; 0, 3), (2, 5; 1, 3), (3, 4; 2, 8), (3, 5; 0, 5), (4, 5; 4, 6)\}$
2. Určete přípustný tok v síti G_2 (pokud existuje) pomocí zlepšujících kružnic. Postup algoritmu vhodně zapište.
   $G_2 = \{(v1, v2; 0, 5), (1, 3; 3, 9), (2, 4; 0, 3), (2, 5; 1, 3), (3, 4; 2, 8), (3, 5; 0, 5), (4, 5; 4, 6)\}$

11.4 Nejlevnější cirkulace

Časová náročnost kapitoly: 20 minut

Cíl kapitoly:
Po prostudování této kapitoly budete umět vysvětlit popsat apoužít postu pro nalezení nejlevnější cirkulace.

Zatím zde byly uváděny algoritmy, které pracují se síti s dolním a horním omezením toku. Ale některé problémy mohou být zadány i s cenou, kterou musíme vynaložit na přepravu jednotkového množství toku. Pak nás může zajímat nejlevnější cirkulace v daném ohodnoceném grafu.

Je-li každá hrana grafu kromě omezení toku $l(e)$ a $c(e)$ ohodnocena také číslem $a(e)$, pak číslo $a(e)$ 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. Cena toku v síti je definována jako součet cen toků v jednotlivých hranách sítě.

Buď dána síť G s omezeními toku l a c a dále je dán libovolný (třeba i nepřípustný) tok f. Pak definujeme přírůstkovou síť vzhledem k toku f. Množina vrcholů je stejná a pro každou hranu $e \in E(G)$ definujeme hrany v přírůstkové síti (viz obr. 11.3) takto:

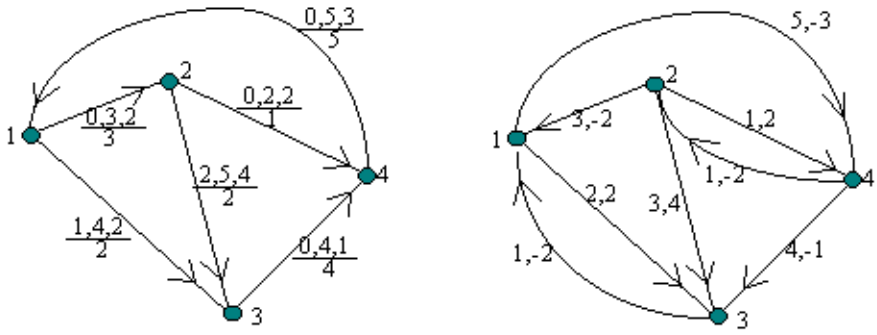
- Jestliže $f(e) < c(e)$, bude v přírůstkové síti hrana e_+ orientovaná stejně jako hrana e - hrana vpřed. Její kapacita bude $c(e) - f(e)$.
- Jestliže $f(e) > l(e)$, bude v přírůstkové síti hrana e_- orientovaná opačně než hrana e - hrana vzad. Její kapacita bude $f(e) - l(e)$.

dolním omezení toku bude u všech hran v přírůstkové síti nulové.

Vidíme (pokud si něco nakreslíme), že každé zlepšující cestě nebo kružnici v síti G odpovídá (a to jednoznačně) orientovaná cesta nebo cyklus v přírůstkové síti.

Je-li dána síť G s omezeními toku l, c a s jednotkovými cenami toku a a dále je dána libovolná cirkulace f, pak cena zlepšující kružnice je definována jako součet jednotkových cen toků na hranách vpřed minus součet jednotkových cen toků na hranách vzad.

V přírůstkové síti je pojem jednotkové ceny toku dán takto: na hranách vpřed jsou stejné ceny jako na odpovídajících hranách v původní síti a na hranách vzad jsou ceny s opačným znaménkem než v G. Cena zlepšující kružnice v přírůstkové síti je tedy přímo rovna součtu ohodnocení hran.



Obrázek 11.3: Graf a příslušná přírůstková síť.

Věta 11.3 *(charakterizace nejlevnějších cirkulací): Přípustná cirkulace f je nejlevnější přípustnou cirkulací v síti G právě tehdy, když v síti G neexistuje vzhledem k cirkulaci f zlepšující kružnice se zápornou cenou.*

Algoritmus:
Nalezení nejlevnější cirkulace.

1. Nalezneme přípustnou cirkulaci. Jestliže přípustná cirkulace neexistuje, výpočet končí.
2. Sestrojíme přírůstkovou síť vzhledem k cirkulaci f.
3. V přírůstkové síti nalezneme cyklus se zápornou cenou (jednotkové ceny toku pokládáme za délky hrany - cyklus se zápornou cenou se testuje vzhledem k jednotkovým cenám toků). Jestliže takový cyklus neexistuje, výpočet končí, cirkulace f je již podle předchozí věty nejlevnější přípustnou cirkulací. Jestliže cyklus existuje, přejdeme k bodu 4.
4. Sestrojíme zlepšující kružnici, která odpovídá nalezenému cyklu, vypočteme její kapacitu (vzhledem ke kapacitám hran v přírůstkové síti), změníme cirkulaci f o vypočtenou kapacitu a pokračujeme bodem 2.

■ **Příklad 11.3** *Určete nejlevnější přípustný tok v síti G_2 se známým přípustným tokem z předchozího cvičného příkladu.* □

Shrnutí kapitoly 11.4

V této kapitole jsme se seznámili s postupem pro nalezení nejlevnější cirkulace.

Otázky ke kapitole 11.4

1. Nalezněte nejlevnější cirkulaci pro graf z obrázku 11.3.

Rejstřík