ALGORITMY NA SIEŤACH

prednášky v letnom semestri 2017

Odporúčaná literatúra:

(1) Plesník J.: Grafové algoritmy. Veda, Bratislava, 1983.

(2) Jungnickel D.: Graphs, Networks and Algorithms (3rd ed.). Springer, Berlin, 2008.

Obsah

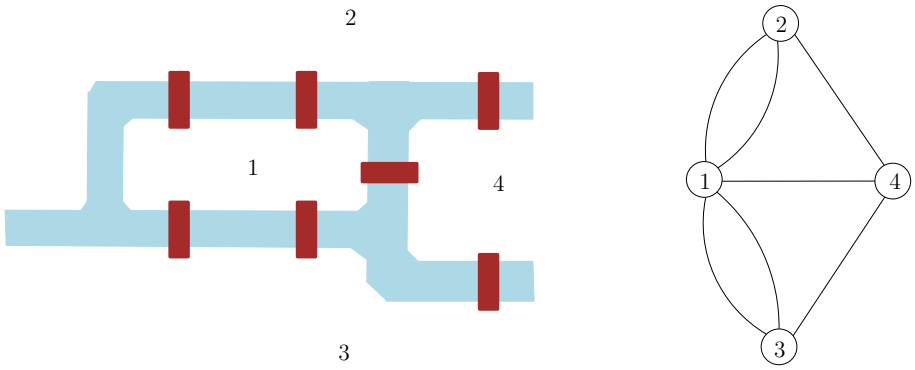
Kapitola 1

ÚVOD

1.1 Teória grafov a siete

1.1.1 História a motivácia

Zatiaľčo pri vzniku mnohých matematických disciplín stáli vážne problémy z takých odborov ako fyzika, chémia a pod., začiatky teórie grafov majú skôr zábavný charakter. Postupne sa však objavujú seriózne praktické problémy. Uveďme niekoľko motivačných príkladov vzniku grafových štruktúr.



Obr. 1.1: Ilustrácia k úlohe o 7 mostoch

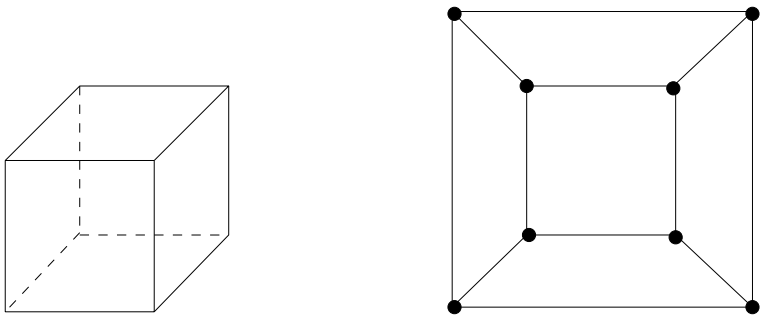
ÚLOHA O 7 MOSTOCH

V r. 1736 v meste Königsberg (dnes Kaliningrad v Rusku) tiekla rieka Pregeľ, rozdeľujúca mesto na 4 časti, ktoré však boli pospájané 7 mostami tak, ako na obr. 1.1. L. Euler, ktorý tam vtedy žil, sa stretol s nasledujúcou zábavnou úlohou: Navrhnúť okružnú prechádzku, pri ktorej prejdeme po každom moste práve raz. Euler ukázal, že taká prechádzka neexistuje a vytvoril aj teóriu pre všeobecný prípad. Implicitne v jeho úvahách nachádzame priradenie diagramu

(schémy) ako na obr. 1.1, kde sú len údaje podstatné pre úlohu. Krúžky zodpovedajú častiam mesta a dva takéto krúžky sú spojené čiarou ak príslušné časti sú priamo spojené mostom. Takto sa stačí zaoberať cestovaním po uvedenpm diagrame. Poznamenajme, že táto úloha súvisí so známou zábavnou úlohou kreslenia obrázkov jedným ťahom.

POLYÉDRE

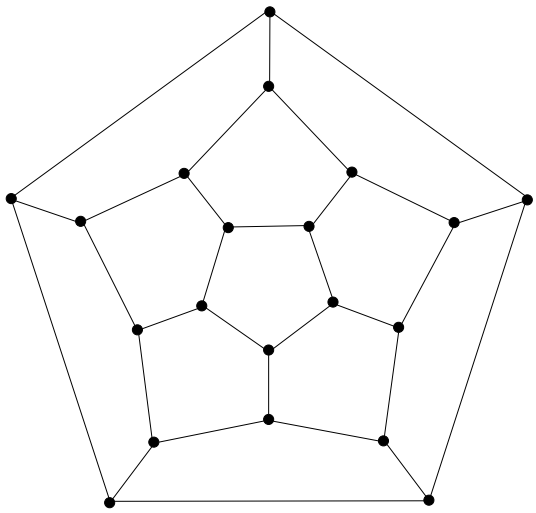
Okolo r. 1750 sa Euler snažil nájsť nejaký podobný vzťah ako je medzi počtom uhlov a hrán v mnohouholníku aj pre 3-rozmerné (konvexné) polyédre. Objavil (Eulerov) vzorec $n - m + r = 2$, kde n je počet vrcholov, m je počet hrán a r je počet 2-rozmerných stien. Napr. pre kocku z obr. 1.2 máme $n = 8$, $m = 12$ a $r = 6$. Eulerovi sa nepodarilo urobiť uspokojivý dôkaz; taký bol urobený až neskôr. Elegantný dôkaz dal Cauchy v r. 1813 prechodom k rovinným diagramom. To možno urobiť napr. tak, že polyéder zobrazíme najprv na guli obsahujúcej tento polyéder (jej stred je vo vnútornom bode polyédra a z neho premietneme hranicu polyédra na guľu) a potom odtiaľ do roviny (premietnutím z vnútorného bodu niektorej oblasti do diametrálne vzdialenej dotykovej roviny ku guli; tzv. stereografická projekcia). Pre kocku je príslušný rovinný diagram v pravej časti obr. 1.2, kde vrcholy sú zdôraznené plnými krúžkami.



Obr. 1.2: Kocka a jej rovinný diagram

Poznamenajme, že o pravidelné konvexné polyédre sa zaujímali už starí Gréci, ktorí študovali tzv. platónske telesá. Takéto teleso má tú vlastnosť, že steny sú zhodné pravidelné mnohouholníky a pri každom vrchole je rovnaký počet hrán. Geometrickými úvahami zistili, že existuje práve 5 platónskych telies. Jednoduchý dôkaz tohoto faktu bol daný až na základe Eulerovho vzorca. Tieto telesá majú 4, 6, 8, 12 a 20 stien (štvorsten, kocka, duál ku kocke, 12-sten a jeho duál). Na obr. 1.3 je rovinný diagram pre 12-sten.

S týmto mnohostenom súvisí aj hra z r. 1859, ktorej autorom bol W. R. Hamilton. V tejto hre bolo treba urobiť rôzne prechádzky. Najjednoduchšia verzia úlohy bola nájsť na pravidelnom 12-stene okružnú prechádzku obsahujúcu každý vrchol práve raz. V tejto hre to bolo veľmi ľahké, ale vo všeobecnosti sa takéto prechádzky hľadajú veľmi ťažko.

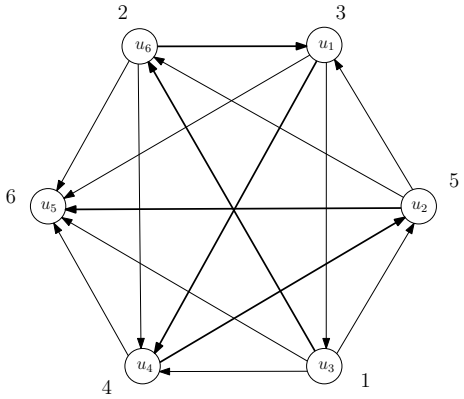


Obr. 1.3: Rovinný diagram pre pravidelný 12-sten

TURNAJE

Uvažujme taký športový turnaj, kde hrá každý s každým a z každého súboja výjde víťaz a porazený. Výsledok takéhoto turnaja možno zobraziť tak ako na obr. 1.4, kde 6 účastníci turnaja $u_1, u_2, u_3, u_4, u_5, u_6$ sú zobrazení krúžkami a šípka smeruje od víťaza k porazenému. V danom príklade možno usporiadať účastníkov, t.j. očíslovať číslami 1, 2, 3, 4, 5, 6 tak, že 1 porazila 2, 2 porazila 3, ..., 5 porazila 6.

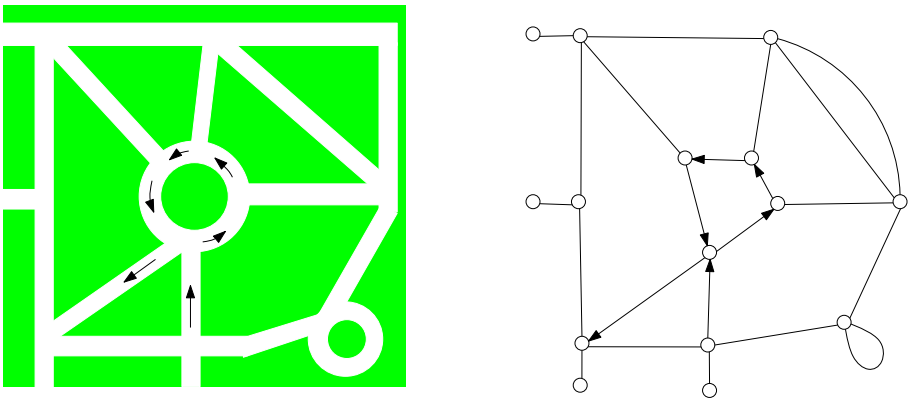
Cvičenie 1.1. Ukážte, že to platí vo šeobecnosti pre ľubovoľný turnaj s n účastníkmi.



Obr. 1.4: Turnaj s usporiadanými účastníkmi

CESTNÁ SIEŤ

Na obr. 1.5 vľavo je zobrazená cestná sieť. Vpravo je korešpondujúca schéma, kde rázcestia a konce ulíc sú zobrazené krúžkami a cestné úseky čiarami. Táto schéma ukazuje možnosti cestovania. Oproti realite sa tu nedozvieme, či je vedľa cesty trávnik, aká je cesta široká a pod. Podobné informácie niekedy potrebujeme, a preto ich dodávame najčastejšie v podobe ohodnotení (napr. na čiaru spájajúcu 2 križovatky napíšeme dĺžku príslušného cestného úseku).



Obr. 1.5: Cestná sieť

1.1.2 Rôzne grafové štruktúry

Horeuvedené príklady nás vedú k formálnej definícii "grafov". Teória grafov nemá jednotnú terminológiu (v angličtine, ale ani v slovenčine). Existujú 2 ex-

trémy pri tvorbe terminológie. Jeden z nich preferuje nazývať grafom najvšeobecnejšiu štruktúru a potom rôznymi prívlastkami vymedziť špeciálne štruktúry. Druhý extrém používa slovo graf pre najčastejšie používanú triedu a rôznymi predponami sú utvorené názvy pre iné triedy. My sa prikláňame k tomuto druhému extrému. Základom je vždy množina tzv. **vrcholov**. Vrchol na obrázkoch zakresľujeme obyčajne krúžkom, štvorčekom a pod. Tak ako v mnohostene, niektoré vrcholy sú spojené **hranou**. Hranu zakresľujeme ako čiaru spájajúcu body zodpovedajúce vrcholom. Vo všeobecnosti, niekedy môžu byť 2 vrcholy spojené aj viacerými hranami, ako napr. v úlohe o 7 mostoch. Také hrany nazývame **paralelné**, alebo **násobné**. V niektorých situáciach potrebujeme spojiť vrchol sám so sebou; vtedy príslušnej hrane hovoríme **slučka**. Vždy platí, že hrana spája buď 2 rôzne vrcholy, alebo vrchol sám so sebou. (Existujú aj také štruktúry kde hrana spája viac vrcholov - tzv. hypergrafy; tými sa však nebudeme zaoberať.) Niekedy potrebujeme vystihnúť poradie vrcholov v hrane (ako napr. v turnaji). Vtedy na hranu zakreslíme šípku a takto **orientovanú hranu** nazývame aj **šíp**. Pre naše potreby bude množina vrcholov V vždy konečná a tiež hrán bude len konečne mnoho. Klasifikácia vychádza z obr. 1.6. Takto napr. graf G môžeme definovať ako usporiadanú dvojicu (V, E), kde V je konečná množina vrcholov a E je podmnožina množiny všetkých dvojprvkových podmnožín množiny V. Často budeme písať $V(G) = V$, $E(G) = E$ a pre počty vrcholov a hrán budeme dodržiavať označenie: $n = |V|$ a $m = |E|$. **Digraf** sa líši iba tým, že jeho hranová množina je podmnožinou množiny všetkých usporiadaných dvojíc rôznych prvkov množiny V. Pri definovaní multigrafu, multidigrafu, ... je výhodné použiť pojem multimnožiny (niekedy sa multimnožina nazýva sústava alebo systém). V praxi sa však násobné (paralelné) hrany môžu odlišovať svojimi parametrami, a preto ich treba rozlíšiť. Vtedy je nutné pristúpiť k všeobecnejšej definícii kde aj E bude množina. Nech V je konečná množina (vrcholov), $V \circ V$ je množina všetkých neusporiadaných dvojíc prvkov z V a $V \times V$ je množina všetkých usporiadaných dvojíc prvkov z V. Potom pseudomigraf môžeme definovať ako usporiadanú trojicu (V, E, φ), kde V a E sú konečné disjunktné množiny a φ je zobrazenie $E \mapsto (V \circ V) \cup (V \times V)$.

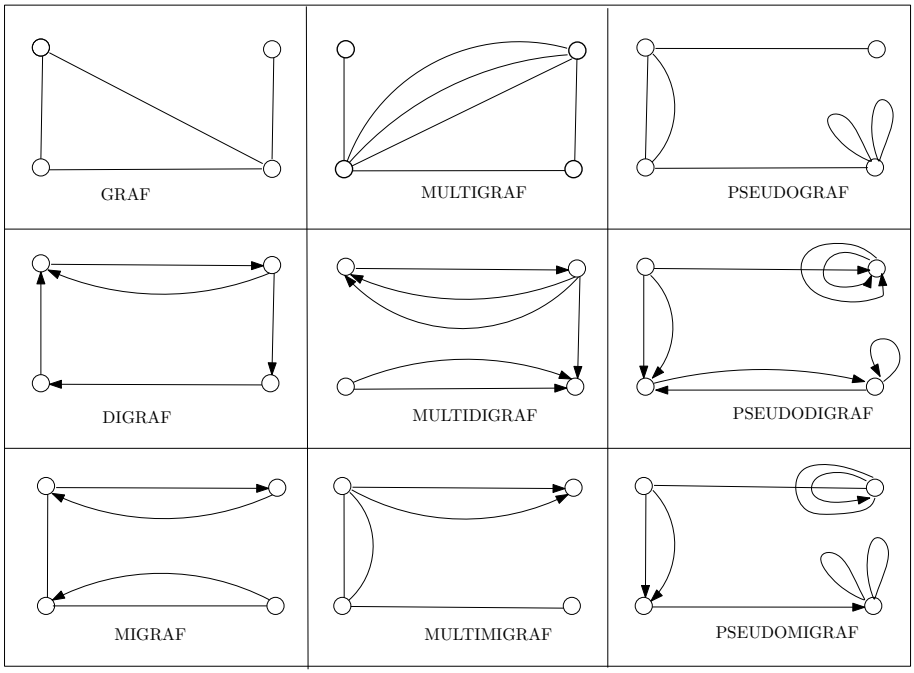
Základná klasifikácia grafových štruktúr je zrejmá z obr. 1.6. **Graf** má len neorientované hrany a nemá násobné hrany ani slučky. **Multigraf** môže mať násobné hrany, ale nemá slučky. Nakoniec **pseudograf** môže mať aj slučky. **Digraf** má len orientované hrany (šípy) a nemá násobné hrany ani slučky. **Multidigraf** môže mať násobné hrany, ale neobsahuje slučky. Nakoniec **pseudodigraf** môže mať aj slučky. **Migraf** môže mať neorientované aj orientované hrany, ale nie násobné a ani slučky. **Multimigraf** nemá slučky a **pseudomigraf** môže mať všetko.

Najčastejšie sa budú vyskytovať grafy a digrafy. Takto napr. pri skeletoch polyédrov vystačíme s grafmi. Turnaje sú špeciálne digrafy. Ale v probléme o 7 mostoch máme multigraf a v príklade cestnej siete máme dokonca pseudomigraf. Ukazuje sa však, že v mnohých úlohách vieme zložitejšiu štruktúru nahradiť jednoduchšou (čo závisí na úlohe) a tak vystačíme často s grafmi, resp. digrafmi. Napr. v probléme o 7 mostoch môžeme priradiť vrchol (krúžok) aj jednému z mostov medzi 1 a 2 a tiež jednému z mostov medzi 1 a 3. Tým sa úloha nezmení

a príslušná schéma bude grafom.

Ak v digrafe G dezorientujeme každú hranu, t.j. nahradíme každú orientovanú hranu (u, v) neorientovanou hranou $[u, v]$, tak získame nejaký multigraf, ktorému hovoríme **dezorientácia** digrafu G. Obrátene, ak v nejakom grafe H zorientujeme každú hranu, t.j. každú hranu $[u, v]$ nahradíme buď orientovanou hranou (u, v), alebo orientovanou hranou (v, u), tak získame nejaký digraf, ktorému hovoríme **orientácia** grafu H.

Hoci sa v názve tejto prednášky vyskytuje slovo **sieť**, nebudeme ho používať ako matematický termín. Dôvodom je terminologická nejednotnosť. V literatúre sa obyčajne pod sieťou rozumie ohodnotený graf, digraf, atď. , ale niekedy napr. aj špeciálny ohodnotený digraf. Slovo sieť budeme používať voľne v bežnom význame (napr. vodovodná sieť, elektrická sieť a pod.).



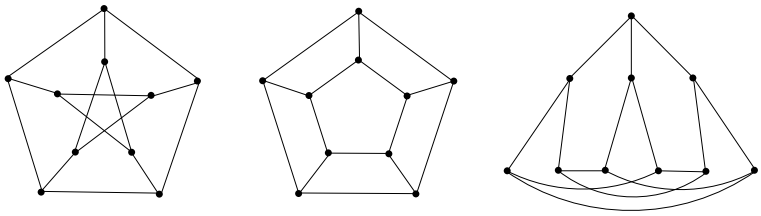
Obr. 1.6: Klasifikácia grafových štruktúr

1.1.3 Niektoré základné pojmy

Vo všeobecnosti hovoríme, že rôzne vrcholy spojené hranou sú **susedné**. Ak je hrana orientovaná, tak rozlišujeme jej **začiatok** a **koniec**. Hovoríme, že táto hrana je **prichádzajúca** pre koniec a **odchádzajúca** pre začiatok. Tiež hovoríme, že začiatok je **predchodca** konca a koniec je **nasledovník** začiatku. Podobne 2 rôzne hrany incidentné s tým istým vrcholom sa nazývajú **susedné**.

Vrchol a hrana pri tomto vrchole sa nazývajú **incidentné**. Pojmy vyjadrujúce podštruktúru, alebo nadštruktúru sú známe z matematiky a tu ich tiež používame. Napr. ak pre daný graf $G = (V, E)$ je $V' \subseteq V$, $E' \subseteq E$ a $G' = (V', E')$ je graf, tak hovoríme, že G' je **podgraf** grafu G a píšeme $G' \subseteq G$. Tiež pojem izomorfizmu je užitočný: Hovoríme, že graf $G = (V, E)$ a graf $G' = (V', E')$ sú **izomorfné**, ak existuje bijekcia množiny V na množinu V' zachovávajúca susednosť. Pri digrafoch sa musí zachovať aj orientácia hrany.

Cvičenie 1.2. Zistite, ktoré dvojice z 3 grafov na obr. 1.7 sú izomorfné a ktoré nie.



Obr. 1.7: Grafy pre cvičenie o izomorfizme. Prvý z nich je známy ako Petersenov graf

STUPNE VRCHOLOV

V grafe definujeme **stupeň** $deg(v)$ vrchola v ako počet hrán incidentných s vrcholom v. V digrafe rozlišujeme **odchádzajúci stupeň** $deg^+(v)$, čo je počet odchádzajúcich šípov z vrchola v a **prichádzajúci stupeň** $deg^-(v)$, čo je počet hrán prichádzjúcich do v. Priamo z definície máme:

VETA 1.1. *Pre počet hrán m platí:*

(1) Ak G je graf, tak $2m = \sum_{v \in V(G)} deg(v)$.

(2) Ak G je digraf, tak $m = \sum_{v \in V(G)} deg^+(v) = \sum_{v \in V(G)} deg^-(v)$

Cvičenie 1.3. Dokážte, že v každom grafe je počet vrcholov nepárneho stupňa párny.

CESTOVANIE

Najprv uvažujeme grafy. Uvádzané formálne definície sa však rovnako dajú použiť i v multigrafoch, resp. pseudografoch. **Sled** je alternujúca postupnosť vrcholov a hrán grafu začínajúca a končiaca vrcholom

$$v_0, h_1, v_1, h_2, v_2, \ldots, v_{k-1}, h_k, v_k$$

pričom pre každú hranu platí, že jej predchodca a nasledovník v tejto postupnosti sú vrcholy v grafe spojené touto hranou (teda susedné). Číslo $k \geq 0$ je

dĺžka sledu, v_0 je **začiatok** a v_k **koniec** sledu. Stručne takýto sled pomenujeme ako v_0-v_k **sled**. Ak je začiatok a koniec ten istý vrchol, tak hovoríme, že sled je **uzavretý**, inak **otvorený**.

Ťah je taký sled, v ktorom sa hrany neopakujú. Ak ťah obsahuje všetky hrany grafu aj všetky vrcholy, tak sa nazýva **eulerovský**. (V úlohe o 7 mostoch treba nájsť uzavretý eulerovský ťah v príslušnom multigrafe. Pri kreslení obrázkov „jedným ťahom“ môže byť ťah aj otvorený.)

Sled, v ktorom sa žiadny vrchol neopakuje, sa nazýva **cesta**. S výnimkou triviálnej (t.j. nulovej dĺžky), každá cesta je otvorená. Uzavretý ťah nenulovej dĺžky sa nazýva **cyklus**, ak sa žiaden vrchol neopakuje s jedinou výnimkou, že začiatok sa rovná koncu ťahu. Ak cyklus obsahuje všetky vrcholy grafu, tak sa nazýva **hamiltonovský**. (V Hamiltonovej hre bolo treba nájsť hamiltonovský cyklus.) Analogicky sa definuje aj **hamiltonovská cesta**, ako cesta obsahujúca všetky vrcholy grafu.

VETA 1.2. *V ľubovoľnom grafe platí:*

(a) Z každého u-v sledu možno vybrať u-v cestu.

(b) Z každého uzavretého sledu nepárnej dĺžky možno vybrať cyklus nepárnej dĺžky.

Dôkaz:

Cvičenie 1.4.

■

Poznamenajme, že nie vždy je možné z uzavretého sledu nenulovej dĺžky vybrať cyklus. Napr. zo sledu (u, v, w, v, u) to nie je možné.

Cvičenie 1.5. Dokážte, alebo vyvráťte: Ľubovoľné 2 najdlhšie cesty v súvislom grafe majú spoločný vrchol. (Uvažujte cesty ako podgrafy tvorené jej prvkami a tak x-y cesta a k nej obrátená y-x cesta je považovaná za tú istú. Preto, ak najdlhšia cesta má dĺžku L, tak druhá najdlhšia cesta môže mať dĺžku aj $L - 1$.)

Cvičenie 1.6. Dokážte, že ak v grafe s $n \geq 3$ vrcholmi má každý vrchol stupeň aspoň $\frac{n}{2}$, tak tam existuje hamiltonovský cyklus.

Pre digrafy sa definujú formálne rovnako všetky horeuvedené pojmy s tým dodatkom, že vždy musíme cestovať v smere orientácie hrany, t.j. v slede každej hrane musí bezprostredne predchádzať jej začiatok a za ňou hneď nasledovať jej koniec. (Horeuvedená vlastnosť turnajov sa teraz dá sformulovať takto: Každý turnaj má hamiltonovskú cestu.) Ak povolíme cestovanie aj proti smeru orientácie hrany, tak hovoríme o **poloslede, poloťahu, poloceste** atď.

VETA 1.3. *V ľubovoľnom digrafe platí:*

(a) Z každého u-v sledu možno vybrať u-v cestu.

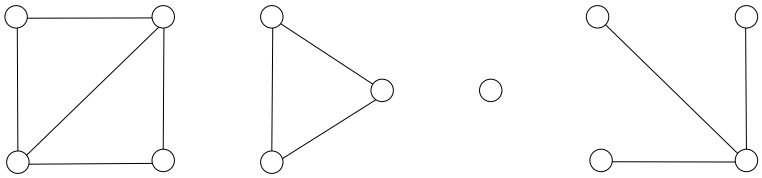
(b) Z každého uzavretého sledu nenulovej dĺžky možno vybrať cyklus.

Dôkaz:

Cvičenie 1.7.

■

Graf sa nazýva **súvislý**, ak pre každé 2 vrcholy u a v existuje u-v sled; inak sa nazýva **nesúvislý**. Maximálny, v zmysle inklúzie, súvislý podgraf sa nazýva **komponent** grafu. Teda graf je súvislý práve vtedy , keď má jediný komponent. Tiež je zrejmé, že dva rôzne komponenty sú vrcholovo-disjunkné podgrafy. Preto môžeme komponenty grafu kresliť oddelene tak, ako na obr. 1.8 a pre väčšinu úloh skúmať nezávisle.



Obr. 1.8: Graf, ktorý má 4 komponenty

Hrana e grafu G sa nazýva **most**, ak sa po jej vynechaní zväčší počet komponentov. Podobne definovaný vrchol sa nazýva **artikulácia**. Napr. v grafe tvaru cesty sú všetky nekoncové vrcholy artikuláciami.

Cvičenie 1.8. Dokážte, že v grafe s $n \geq 2$ vrcholmi sú aspoň 2 také vrcholy, ktoré nie sú artikuláciami.

VETA 1.4. *Hrana je mostom* $\Leftrightarrow$ *neleží v cykle.*

Dôkaz:

Cvičenie 1.9.

■

Cvičenie 1.10. Nech graf má práve 2 vrcholy x, y nepárneho stupňa. Rozhodnite, či existuje x-y cesta.

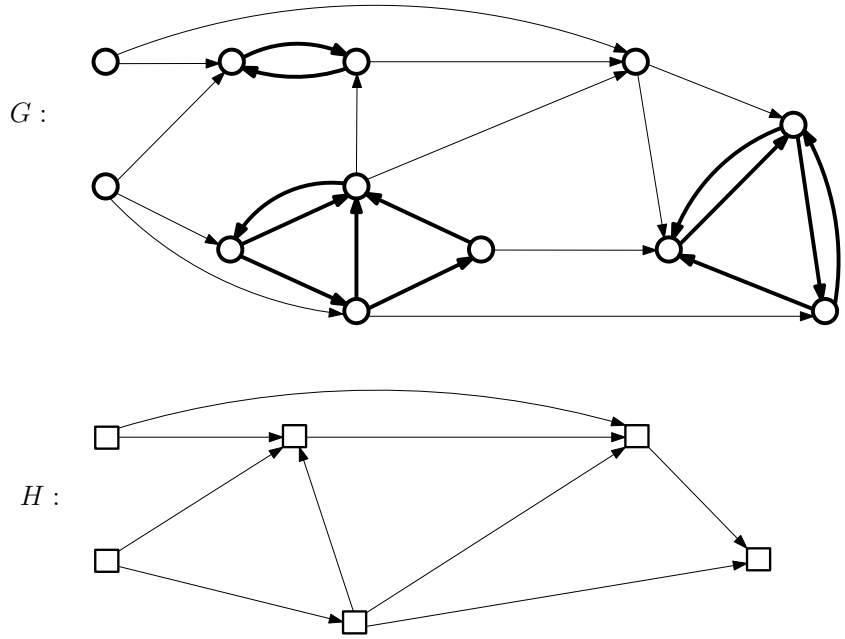
Cvičenie 1.11. Ak v grafe má každý vrchol stupeň aspoň 2, tak tam existuje cyklus. Dokážte.

Cvičenie 1.12. Ak v digrafe má každý vrchol odchádzajúci stupeň aspoň 1, tak tam existuje cyklus.

Cvičenie 1.13. Nech v grafe je stupeň každého vrchola aspoň 3. Dokážte, že potom tam existuje cyklus párnej dĺžky. [Návod: Uvažujte najdlhšiu cestu a susedov jej posledného vrchola.]

Cvičenie 1.14. Nájdite chybu v nasledujúcom dôkaze indukciou, kde dokazujeme falošné tvrdenie: Každý graf s $n \geq 3$ vrcholmi, kde každý vrchol má stupeň aspoň 2, má 3-cyklus (cyklus dĺžky 3). Pre $n = 3$ je tvrdenie pravdivé. Pre $n > 3$ zoberme graf s $n - 1$ vrcholmi, pre ktorý to platí. Pridajme nový vrchol a spojme ho aspoň 2 hranami s týmto grafom. Je zrejmé, že získaný graf bude mať n vrcholov, stupeň každého vrchola aspoň 2 a zrejme bude obsahovať 3-cyklus, lebo pôvodný ho obsahoval (indukčný predpoklad).

V digrafoch rozlišujeme silnú a slabú súvislosť. **Silná** je definovaná formálne rovnako ako súvislosť grafu. Digraf je **slabo súvislý**, ak jeho dezorientácia je súvislá. **Silný komponent** digrafu je maximálny, v zmysle inklúzie, silne súvislý poddigraf. Obr. 1.9 ilustruje tieto pojmy. Ak v grafe alebo digrafe existuje u-v sled, tak hovoríme, že vrchol v je **dosiahnuteľný** z vrchola u. Vidíme, že v komponente grafu, a tiež v silnom komponente digrafu, sú každé 2 vrcholy vzájomne dosiahnuteľné.



Obr. 1.9: Digraf G, ktorý má 6 silných komponentov (hrubo vyznačené) a jeho kondenzácia H

Cvičenie 1.15. Nech v turnaji má vrchol u maximálny odchádzajúci stupeň. Dokážte, že potom pre každý vrchol v existuje u-v cesta dĺžky nanajvýš 2.

Prv uvedené tvrdenie o usporiadaní účastníkov turnaja možno sformulovať takto: Každý turnaj má hamiltonovskú cestu.

Cvičenie 1.16. Dokážte, že každý silne súvislý turnaj s $n \geq 3$ vrcholmi má hamiltonovský cyklus.

VZDIALENOSTI

V grafe definujeme **vzdialenosť** $d(x, y)$ vrcholov x a y ako dĺžku najkratšej x-y cesty, ak tieto vrcholy súvisia; inak kladieme $d(x, y) = \infty$. Vidíme, že funkcia vzdialenosti je v súvislom grafe metrikou. **Diameter** $diam(G)$ grafu G je maximálna vzdialenosť vrcholov grafu. Pre každý vrchol x definujeme **excentricitu** $ec(x)$ ako maximálnu vzdialenosť z vrcholu x. Potom diameter je vlastne maximálna excentricita. Na druhej strane, **rádius** $rad(G)$ je definovaný ako minimálna excentricita. Všetky vrcholy s minimálnou excentricitou tvoria tzv. **centrum** grafu.

Cvičenie 1.17. $rad(G) \leq diam(G) \leq 2rad(G)$.

Formálne rovnako definujeme vzdialenosť aj v digrafe, ale tam už nemusí byť metrikou, lebo nemusí byť symetrická.

NIEKTORÉ ŠPECIÁLNE GRAFY A DIGRAFY

Graf sa nazýva **kompletný**, ak sú každé 2 rôzne vrcholy susedné (teda má $\binom{n}{2}$ hrán). Na druhej strane máme tzv. **nulový graf**, t.j. graf bez hrán. No zaujímavejším grafom, ktorý je súvislý a v istom zmysle minimálny, je **strom**, ktorý definujeme ako súvislý graf bez cyklov.

VETA 1.5. *Každý netriviálny strom má aspoň 2 koncové (t.j. stupňa 1) vrcholy.*

Dôkaz:

Cvičenie 1.18.

■

VETA 1.6. *Pre ľubovoľný graf G sú nasledujúce tvrdenia ekvivalentné:*
(1) G je strom.
(2) G je súvislý a každá jeho hrana je mostom.
(3) Pre každé 2 vrcholy $x, y \in V(G)$ existuje práve jedna x-y cesta.
(4) G je súvislý a $m = n - 1$.
(5) G je bez cyklov a $m = n - 1$.

Dôkaz:

Cvičenie 1.19.

■

Cvičenie 1.20. Nájdite minimálne $f(n)$, pre ktoré platí: Ak má graf s n vrcholmi aspoň $f(n)$ hrán, tak je súvislý.

Zaujímavý je nasledujúci výsledok.

VETA 1.7. *V každom strome centrum pozostáva buď z jediného vrcholu, alebo z dvoch susedných vrcholov.*

Dôkaz:

Cvičenie 1.21.

■

Graf sa nazýva **bipartitný**, ak jeho vrcholovú množinu možno rozložiť na 2 partie tak, že každá hrana grafu má jeden koniec v jednej partii a druhý v druhej (t.j. ak neexistuje hrana, ktorej oba konce ležia v tej istej partii). Napr. každý strom je bipartitný. Vo všeobecnosti máme:

VETA 1.8. *Graf je bipartitný $\Leftrightarrow$ neobsahuje cyklus nepárnej dĺžky.*

Dôkaz:

Cvičenie 1.22.

■

Graf, alebo digraf, sa nazýva **acyklický**, ak neobsahuje cyklus. Ako už vieme, súvislý acyklický graf je strom. Vo všeobecnosti acyklický graf pozostáva z niekoľkých stromov, a preto sa nazýva **les**.

Chvíľu sa teraz venujeme digrafom. Vrchol digrafu sa nazýva **prameň [ústie]**, ak má prichádzajúci [odchádzajúci] stupeň 0. (Izolovaný vrchol je súčasne prameňom i ústim.) Hovoríme, že vrcholy digrafu možno **topologicky usporiadať**, ak ich možno očíslovať číslami 1, 2, ..., n tak, že každý šíp ide z menšieho do väčšieho čísla.

VETA 1.9. *Každý acyklický digraf má prameň i ústie.*

Dôkaz:

Cvičenie 1.23.

■

VETA 1.10. *Digraf je acyklický $\Leftrightarrow$ vrcholy digrafu možno topologicky usporiadať.*

Dôkaz:

Cvičenie 1.24.

■

Informáciu o dosiahnuteľnosti v digrafe nám prehľadne poskytuje tzv. **digraf silných komponentov**, alebo stručne, **kondenzácia** digrafu. To je nový digraf, ktorého vrcholmi sú silné komponenty pôvodného digrafu a z vrcholu ide šíp do druhého vrcholu práve vtedy, ak v pôvodnom digrafe existoval aspoň jeden šíp idúci zo silného komponentu korešpondujúcemu tomuto vrcholu do silného komponentu korešpondujúcemu druhému vrcholu. Táto konštrukcia je ilustrovaná na obr. 1.9.

VETA 1.11. *Kondenzácia každého digrafu je acyklický digraf.*

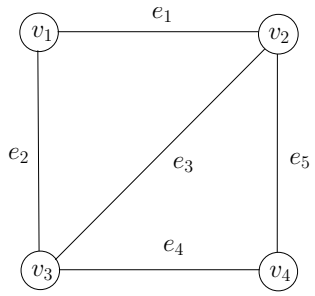
Dôkaz:

Cvičenie 1.25.

■

1.2 Ako zadať grafové štruktúry ?

Malé objekty ľudom často zadávame obrázkami, ale tu ide o informácie, ktoré potrebujeme dostať do počítača. Z teórie grafov sú známe rôzne matice popisujúce graf, alebo digraf. Napr. matica susednosti, matica incidencie a i. Tieto matice majú teoretický význam, ale na zadávanie sú často zbytočne veľké. Preto sa uprednostňujú úspornejšie spôsoby, ako napr. zoznam hrán a zoznam susedných vrcholov. Ilustrujeme to na príklade grafu z obr. 1.10.



Obr. 1.10: Príklad grafu pre zadávanie

Matica susednosti A je $V \times V$ matica a má elementy

$$a_{ij} = \begin{cases} 1 & \text{ak vrcholy } v_i \text{ a } v_j \text{ sú susedné} \\ 0 & \text{inak} \end{cases}$$

Teda pre príklad z obr. 1.10 máme:

	v_1	v_2	v_3	v_4
v_1	0	1	1	0
v_2	1	0	1	1
v_3	1	1	0	1
v_4	0	1	1	0

Cvičenie 1.26. Nech G je graf alebo digraf s vrcholmi $1, 2, \ldots, n$ a nech A je jeho matica susednosti. Dokážte, že pre $k = 0, 1, 2, \ldots$ prvok matice A^k (k-ta mocnina matice A) na mieste (i, j) sa rovná počtu i-j sledov dĺžky k.

Matica incidencie B je $V \times E$ matica a má elementy

$$b_{ij} = \begin{cases} 1 & \text{ak vrchol } i \text{ a hrana } j \text{ sú incidentné} \\ 0 & \text{inak} \end{cases}$$

Pre príklad z obr. 1.10 máme:

	e_1	e_2	e_3	e_4	e_5
v_1	1	1	0	0	0
v_2	1	0	1	0	1
v_3	0	1	1	1	0
v_4	0	0	0	1	1

Horeuvedené matice sú pomerne veľké aj pre menšie grafy preto, že obsahujú veľa nulových prvkov. Z tohoto dôvodu sa málo používajú. Úspornejšie sú **zoznam hrán** a **zoznam susedných vrcholov**. Oba spôsoby ilustrujeme na príklade z obr. 1.10.

V zozname hrán je v skutočnosti aj zoznam vrcholov a každá hrana je uvedená ako dvojica vrcholov:

vrcholy	$v_1,\quad v_2,\quad v_3,\quad v_4$
hrany	$v_1\ v_2,\quad v_1\ v_3,\quad v_2\ v_3,\quad v_2\ v_4,\quad v_3\ v_4$

Pre náš graf máme nasledujúci zoznam susedných vrcholov:

vrchol	susedné vrcholy
v_1	v_2 v_3
v_2	v_1 v_3 v_4
v_3	v_1 v_2 v_4
v_4	v_2 v_3

Tento zoznam obsahuje každú hranu dvakrát, čo by bolo možné zredukovať tak, že by sme ju uvádzali iba v okolí menšieho (incidentného) vrchola. V praxi sa však takáto duplicita často ponecháva kvôli kontrole údajov.

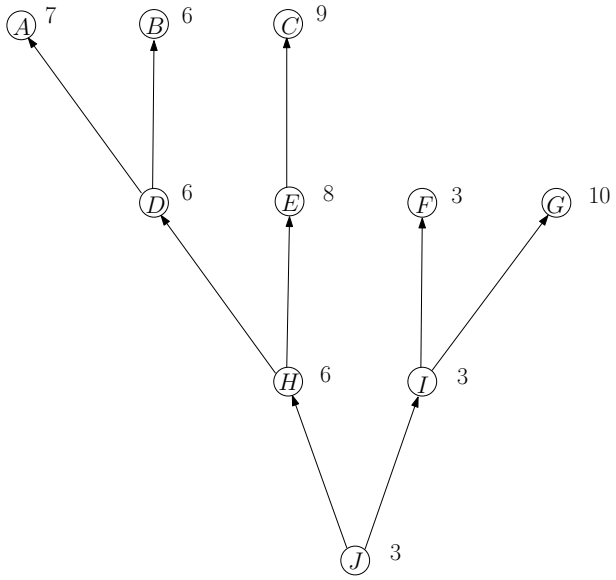
Všetky uvádzané spôsoby sa ľahko prispôsobia pre digrafy a rozšíria pre ohodnotené štruktúry.

Cvičenie 1.27. Urobte to.

1.3 Usporiadanie pomocou haldy

V mnohých algoritmoch je potrebné vybrať najmenší alebo najväčší prvok, resp. je vhodné ak máme prvky usporiadané podľa veľkosti. To často umožňuje znížiť

výpočtovú zložitosť. Ilustrujeme to na probléme usporiadať n reálnych čísel $a_1, a_2, \ldots, a_n$ do neklesajúcej postupnosti. Vo všeobecnosti nech $h(a_i)$ je veľkosť (hodnota, hmotnosť, výška a pod.) prvku a_i. Klasický prístup k tejto úlohe je jednoduchý: V čase $O(n)$ nájdeme najmenší prvok a dáme ho na prvé miesto (výmenou s prvým elementom). Potom v tom istom čase nájdeme najmenší prvok medzi zvyšnými $n-1$ prvkami a dáme ho na druhé miesto (výmenou za druhý element), atď. Celkove takto v čase $O(n^2)$ dostaneme požadované usporiadanie. Ukážeme si, že túto zložitosť možno znížiť pomocou tzv. haldy.



Obr. 1.11: Príklad haldy

Halda je binárny orientovaný strom, v ktorom ohodnotenia vrcholov sú v súlade so štruktúrou stromu ako na obr. 1.11. Presnejšie z jeho koreňa (zdroja) je každý vrchol dosiahnuteľný; každý vrchol s výnimkou koreňa má prichádzajúci stupeň 1; každý vrchol má odchádzajúci stupeň nanajvýš 2; teda každý **otec** má nanajvýš dvoch **synov**, jeden sa bude nazývať ľavý a druhý pravý (a takto ich budeme aj zakreslovať v rovine). Vrstva je množina vrcholov rovnako vzdialených od koreňa. Žiadame, aby boli vrcholy umiestnené čo najbližšie ku koreňu, t.j. vrstva so vzdialenosťou $i+1$ môže existovať iba vtedy, ak je vrstva so vzdialenosťou i kompletná, t.j. má 2^i vrcholov. Okrem toho budeme mať vrcholy v jednej vrstve usporiadané: vždy ľavý syn má prednosť pred pravým a to sa týka aj ich potomkov, teda nejaký vrchol môže mať syna iba vtedy, ak každý uprednostnený vrchol v tej istej vrstve má dvoch synov (je kompletný). Pokiaľ ide o ohodnotenia, tak vyžadujeme, aby syn v mal aspoň také ohodnotenie ako otec u, t.j. $h(v) \geq h(u)$. Počet vrcholov haldy s výškou k kolíše podľa zaplnenia k-tej vrstvy: $2^k \leq n \leq 2^{k+1} - 1$, a teda približne $k = \log_2 n$.

Pri konštrukcii a deštrukcii haldy sa budeme opierať o dve operácie: (i) pridanie prvku a (ii) odobranie koreňa. Po ich realizácii opäť získame haldu (pri deštrukcii jednovrcholovej haldy dostaneme prázdnu haldu).

(i) Prvok pridáme na prvú voľnú pozíciu v poslednej vrstve, ak nie je kompletná; inak týmto prvkom založíme novú vrstvu. Ak jeho otec má väčšie ohodnotenie, tak ich vzájomne vymeníme. Prípadnú poruchu medzi otcom otca a novým prvkom riešime vzájomnou výmenou, atď. V najhoršom prípade sa nový prvok premiestní až na pozíciu koreňa a porucha zmizne. Celkove na pridanie prvku vystačíme s $O(\log_2 n)$ operáciami.

(ii) Po vynechaní koreňa premiestníme na túto pozíciu posledný prvok poslednej vrstvy. Ak vznikne porucha, t.j. niektorý syn bude mať menšie ohodnotenie ako koreň, tak vzájomne vymeníme koreň s menším synom. Prípadnú novú poruchu prenesieme analogicky do ďalšej vrstvy, atď. Prinajhoršom sa takto dostaneme až do poslednej vrstvy, kde už porucha zmizne. Celkove odobranie koreňa sa dá zrealizovať v čase $O(\log_2 n)$.

USPORIADANIE ČÍSEL

Uvažujme n reálnych čísel $c_1, c_2, \ldots, c_n$, ktoré treba usporiadať do neklesajúcej postupnosti. Túto postupnosť môžeme uvažovať ako množinu prvkov $\{a_i\}$ s hodnotami $h(a_i) = c_i$. Najprv skonštruujeme haldu tak, že začneme z prázdnej haldy a postupne pridáme všetky prvky a_i. Na toto vystačíme s $O(n \log_2 n)$ operáciami. Je zrejmé, že koreň tejto haldy je najmenší, a po jeho odobraní nová halda bude mať za koreň druhé najmenšie číslo, atď. Vidíme, že takto postupne odoberáme čísla tvoriace neklesajúcu postupnosť. Týchto odoberaní je n, a teda vystačíme s $O(n \log_2 n)$ operáciami. Sumárne sme usporiadanie n čísiel zvládli v čase $O(n \log_2 n)$, čo je asymptoticky menej ako $O(n^2)$.

Cvičenie 1.28. Ukážte, že ohodnotenia prvkov haldy možno uložiť do vektora, v ktorom každý prvok bude mať presnú pozíciu (prvý je koreň, potom jeho synovia, potom synovia týchto synov, atď.) , a teda netreba ju reprezentovať ako všeobecný digraf.

Poznamenajme, že hoci stromovú reprezentáciu haldy vieme obísť, poslúžila ako vzor pre ďalšie dômyselnejšie haldy v tvare stromov.

Kapitola 2

PRIESKUM A DOSIAHNUTEĽNOSŤ

2.1 Prieskum grafov

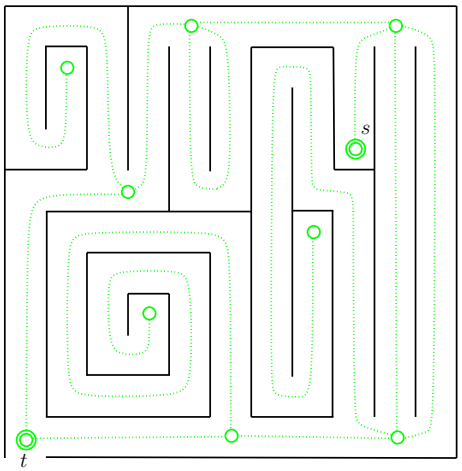
Pri riešení úloh robíme systematický prieskum grafu. Najznámejšie sú 2 postupy: do šírky a do hĺbky.

Postup do šírky je veľmi prirodzený a jednoduchý: Začínajúc v nejakom vrchole s najprv prezrieme susedov tohoto vrchola, potom susedov týchto susedov, atď. Týmto postupom vlastne rozložíme vrcholy grafu, resp. digrafu, do tried podľa vzdialenosti od s.

Cvičenie 2.1. Ukážte, že takto možno nájsť komponenty grafu v čase $O(m+n)$.

Neskôr tento postup použijeme pri hľadaní najkratších ciest.

Postup do hĺbky je netriválna metóda. Hoci sa objavila až okolo r. 1970, jej základom je omnoho starší labyrintový algoritmus. Preto začneme problémom o labyrinte. V tomto probléme treba nájsť sled začínajúci v bode s do bodu t, kde je východ z labyrintu. Príklad labyrintu je na obr. 2.1. Túto úlohu ešte rozširujeme tak, že žiadame nájsť sled začínajúci v s a obsahujúci všetky chodby labyrintu, ktoré sú z bodu s dostupné. Takéto zovšeobecnenie zahŕňa aj situáciu, keď sa k východu z labyrintu nedá dostať. K danému labyrintu môžeme priradiť (pseudo)graf, nasledovne. Rázcestiam a ďalším dôležitým bodom priradíme vrcholy a dva takéto vrcholy spojíme hranou, ak vedie priama chodba medzi príslušnými bodmi v labyrinte. Táto konštrukcia je urobená na obr. 2.1. Pre tento problém stačí skúmať graf (vsunutím nového vrcholu do násobnej hrany a vsunutím 2 vrcholov do slučky, získame zo pseudografu graf). Zdôraznime, že v probléme nemáme daný plán labyrintu, ale labyrint postupne spoznávame. Na rázcestiach vidíme vždy len začiatky incidentných chodieb kde si môžeme zaznamenať lokálnu informáciu. Uvedieme 2 algoritmy, ktoré boli objavené nezávisle koncom 19. storočia.



Obr. 2.1: Príklad labyrintu a korešpondujúci pseudograf (zelený)

TARRYHO ALGORITMUS V GRAFE

Tarry navrhol utvoriť tzv. **tarryovský sled**, čo je ľubovoľný nepredĺžiteľný sled začínajúci v s a postupne skonštruovaný pri dodržaní nasledujúcich 2 pravidiel:
(T1) Každou hranou môžeme v jednom smere prechádzať nanajvýš raz.
(T2) Objaviteľskou hranou (t.j. hranou, po ktorej sme prišli do daného vrcholu prvýkrát) sa môžeme vrátiť iba vtedy, ak niet inej možnosti.

Postup utvárania tarryovského sledu je ilustrovaný na obr. 2.2, kde je postupnosť hrán v slede číslovaná. Smer prechodu hranou ukazujú šípky, pričom objaviteľské šípky sú plné, ostatné otvorené. Keďže každou hranou môžeme prechádzať nanajvýš 2 razy (raz v každom smere) a graf je konečný, tak je zrejmé, že tarryovský sled existuje a celý sa nachádza v komponente obsahujúcom vrchol s.

VETA 2.1. *Tarryovský sled grafu je uzavretý a každú hranu komponentu obsahujúceho vrchol s prechádza oboma smermi.*

Dôkaz: Najprv ukážeme, že sled končí vo vrchole s. Pre spor predpokladajme, že skončil v $x \neq s$ a uvažujme situáciu v tom čase pri x. Nemôže tam existovať hrana neprejdená vôbec a ani hrana prejdená iba v smere do x (lebo by sme po takej hrane mohli odísť). Teda môžu tam byť iba hrany prejdené oboma smermi a hrany prejdené iba v smere od x. Potom však celkový počet odchodov z x je aspoň taký ako počet príchodov do x. Na druhej strane však počet príchodov musí byť o jednotku väčší ako počet odchodov (lebo niekedy sme do x prišli, potom odišli, ..., a nakoniec prišli), spor.

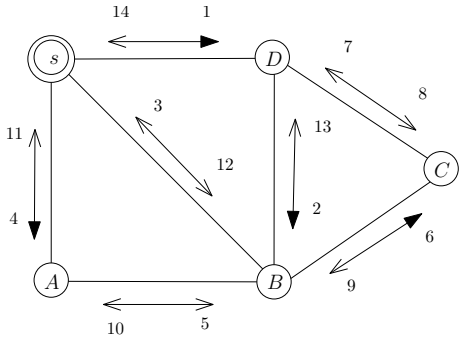
Pre druhú časť tvrdenia nech $v_0 = s, v_1, v_2, ..., v_q$ sú všetky vrcholy sledu v tom poradí ako boli prvýkrát objavené a nech tvoria množinu T. Najprv indukciou podľa tohoto poradia dokážeme, že pre každé $k = 0, 1, 2, ..., q$ po skončení algoritmu sú všetky hrany pri v_k prejdené oboma smermi. Najprv to dokážeme pre vrchol s. Pri s nemôže byť hrana neprejdená vôbec a ani hrana prejdená iba v smere do s. Pretože sme začali aj skončili v s, tak celkový počet odchodov a príchodov musí byť rovnaký. Potom však tam nemôže byť ani hrana prejdená iba v smere od s. Teda všetky hrany pri s sú prejdené oboma smermi.

Teraz nech $k > 0$. Potom vrchol v_k bol objavený z nejakého vrchola v_j kde $j < k$ cez objaviteľskú hranu (v_j, v_k). Uvažujme situáciu pri v_k bezprostredne po odchode proti objaviteľskej šípke do v_j (taký čas nastane, lebo podľa indukčného predpokladu na konci algoritmu je táto hrana prejdená oboma smermi). Pretože sme už mohli odísť proti objaviteľskej šípke, tak pri v_k neexistujú: hrany neprejdené ani raz a hrany iba s prichádzajúcimi šípkami. Pretože počet príchodov a odchodov musí byť rovnaký, tak tam neexistujú ani hrany iba s odchádzajúcimi šípkami. Teda pri vrchole v_k môžu byť iba hrany prejdené oboma smermi, čo sme chceli dokázať.

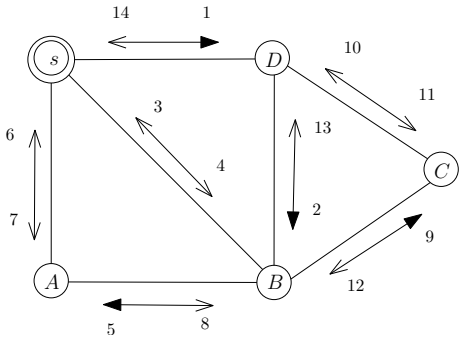
Zostáva dokázať, že sme navštívili každý vrchol w komponentu obsahujúceho štartujúci vrchol s. Z definície komponentu existuje s-w cesta. Podľa dokázaného všetky hrany pri prvom vrchole tejto cesty boli prejdené, a teda aj pri druhom, atď. To znamená, že sme navštívili aj posledný vrchol tejto cesty. ■

TRÉMAUXOV ALGORITMUS V GRAFE

Tento algoritmus je v skutočnosti spresnený Tarryho algoritmus, ku ktorého pravidlám (T1) a (T2) je pridané ešte jedno pravidlo:
(T3) Ak novou hranou (t.j. hranou prechádzanou prvýkrát) prídeme do starého vrchola (t.j. už prv objaveného), tak ihneď v nasledujúcom kroku sa touto hranou vraciame.



Obr. 2.2: Tarryho algoritmus v grafe



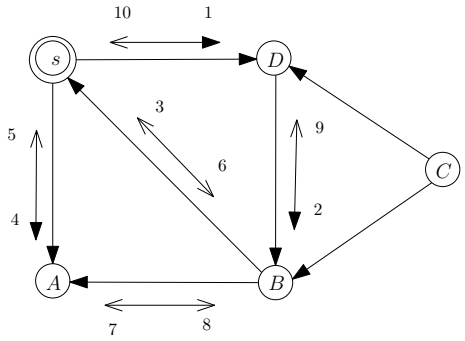
Obr. 2.3: Trémauxov algoritmus v grafe z obr. 2.2

Cvičenie 2.2. Predpokladajte, že graf je zadaný zoznamom susedných vrcholov. Navrhnite lineárny algoritmus (t.j. zložitosti $O(m + n)$) pre Tarryho prieskum a tiež pre Trémauxov prieskum.

2.2 Prieskum digrafov

TARRYHO A TRÉMAUXOV ALGORITMUS V DIGRAFE

Hoci orientované labyrinty sa nebudovali a preto ani neskúmali, môžeme si ich predstaviť. Napr. uvažujme sieť jednosmerných chodieb v nejakom zámku. Jednosmernosť je zabezpečená okrem šípok aj tým, že v každej chodbe sú dvere, ktoré možno otvoriť iba z jednej strany (iba z jednej strany majú kľučku); avšak po otvorení už zostávajú otvorené, a teda príslušná chodba je už potom priechodná aj v opačnom smere. To možno zobraziť (pseudo)digrafom, pričom pri prieskume vytvárame polosled, v ktorom môžeme ísť aj proti orientácii, ale iba vtedy, ak sme už príslušnou hranou šli v smere orientácie. Oba predložené algoritmy možno adaptovať pre digrafy tak, že k uvedeným pravidlám pridáme nasledovné pravidlo (ako prvé):
(TD) Každou hranou môžeme prvýkrát prechádzať iba v smere šípky.



Obr. 2.4: Tarryho, ale nie Trémauxov, algoritmus v digrafe

Cvičenie 2.3. Ukážte, že ak je digraf zadaný zoznamom susedných (odchádzajúcich) vrcholov, tak Tarryho a tiež Trémauxov algoritmus možno realizovať v lineárnom čase.

VETA 2.2. *Tarryovský polosled digrafu je uzavretý, obsahuje všetky vrcholy dosiahnuteľné z s a všetky hrany odchádzajúce z týchto vrcholov. Pritom každú prejdenú hranu prechádza oboma smermi.*

Dôkaz:

Cvičenie 2.4.

■

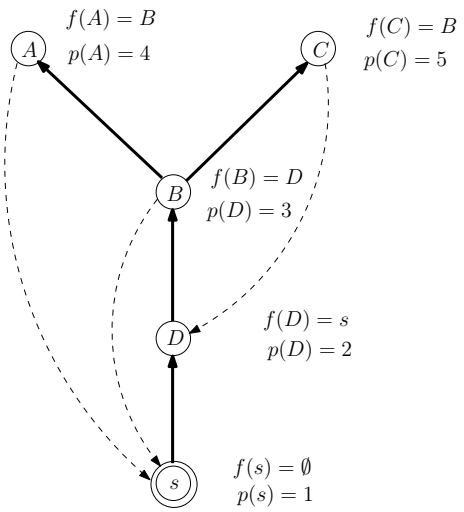
POSTUP DO HĹBKY

Ide o istý protipól k postupu do šírky. Pôvodne bol formulovaný ako rekurzívna procedúra na prehľadávanie grafov a digrafov. No ukázalo sa, že je to vlastne nový pohľad na Trémauxov algoritmus. Počas prieskumu postupne číslujeme objavované vrcholy a tiež si zaznamenávame z ktorého vrchola bol daný

vrchol objavený. Presnejšie, každý navštívený vrchol v dostane **poradové číslo** $p(v)$ (štartujúci vrchol dostane 1). Ďalej, ak bol vrchol v objavený cez objaviteľskú hranu začínajúcu vo vrchole u, tak dostane **otca** $f(v) = u$ (štartujúci vrchol s nemá otca, formálne kladieme $f(s) = \emptyset$). Okrem toho, k prieskumu potom priradíme tzv. **palmu**, čo je digraf na pôvodných vrcholoch s hranami rozdelenými na objaviteľské, tzv. **stromové** a ostatné, tzv. **spätné**, pričom ich šípky zodpovedajú smerom prvého prechodu hranou. Na obr. 2.5 je takáto palma pre prieskum z obr. 2.3.

Cvičenie 2.5. Dokážte, že v súvislom grafe tvoria stromové hrany naozaj strom

.



Obr. 2.5: Palma postupu do hĺbky zodpovedajúca prieskumu z obr. 2.3

Podobne ako v grafoch, aj v digrafoch Trémauxov prieskum dáva základ postupu do hĺbky, kde analogicky priraďujeme vrcholom otcov a poradové čísla. Voľne potom hovoríme o potomkoch a predkoch vrcholov. Teraz však hrany možno rozdeliť do 4 tried. Orientovaná hrana (šíp) (u, v) sa nazýva: (1) **stromová**, ak $f(v) = u$ (vrchol v je bezprostredný potomok vrchola u), (2) **spätná**, ak existuje v-u cesta pozostávajúca zo stromových hrán (vrchol u je potomok vrchola v), (3) **priama**, ak existuje u-v cesta pozostávajúca zo stromových hrán (vrchol v je potomok vrchola u)), (4) **krížová**, ak nie je stromová, spätná ani priama. Je zrejmé, že pre otázky dosiahnuteľnosti môžeme priame hrany aj vynechať.

Cvičenie 2.6. Nahliadnite, že pre krížovú hranu (u, v) platí $p(v) < p(u)$.

2.3 Hľadanie komponentov a silných komponentov

Ak máme graf zadaný zoznamom susedných vrcholov, tak je ľahké navrhnúť algoritmus pre nájdenie komponentov grafu v čase $O(m+n)$ použitím postupu do šírky, Tarryho algoritmu, alebo postupu do hĺbky.

Cvičenie 2.7. Urobte to.

Avšak navrhnúť lineárny algoritmus pre hľadanie silných komponentov digrafu je netriviálna úloha. Tarjan dal takýto algoritmus využitím postupu do hĺbky. Základom je pozorovanie, že ak v Trémauxovom prieskume prídeme do silného komponentu zodpovedajúcemu ústiu kondenzácie, tak tento komponent opustíme až po jeho úplnom preskúmaní, t.j. ak sú navštívené všetky vrcholy a každá hrana je prejdená oboma smermi. Pritom používame zásobník vrcholov Z, do ktorého postupne dávame objavované vrcholy (nájdené cez objaviteľskú hranu, nie znova navštívené). Prvý navštívený vrchol silného komponentu nazývame **koreň** tohoto komponentu (pri danom prieskume). Silný komponent zodpovedajúci ústiu kondenzácie nazývame **koncový**. Ak vrchol v bol koreň koncového silného komponentu , tak pri odchode z tohoto komponentu bude vrchol v posledným navštíveným vrcholom a vtedy posledné vrcholy v zásobníku Z až po vrchol v včítane, vyberieme zo Z a tieto vrcholy tvoria ten silný komponent. Problémom zostáva zistiť ten koreň.

Pre tento cieľ postup do hĺbky rozšírime o ďalšiu informáciu. Každý navštívený vrchol v dostane okrem otca $f(v)$ a poradového čísla $p(v)$ ešte tzv. **dolné číslo** $r(v)$ ukazujúce najmenšie poradové číslo $p(x)$ vrchola x, pre ktorý sme našli v budovanej palme v-x cestu. Formálne, pri objavení vrchola v položíme $r(v) = p(v)$ a potom vždy keď sa vrátime do v proti šípke nejakej hrany (v, w), kde vrchol $w \in Z$, kladieme $r(v) = \min\{r(v), r(w)\}$. Ak sme vo vrchole v pred návratom k otcovi (t.j. každá odchádzajúca hrana je už prejdená a všetci potomkovia vrchola v sú kompletne preskúmaní), tak testujeme, či $r(v) = p(v)$. V kladnom prípade robíme výstup silného komponentu. Vo zvyšnom digrafe sa potom môže nejaký silný komponent stať novým „ústim“. Celý postup je ilustrovaný na príklade z obr. 2.6.

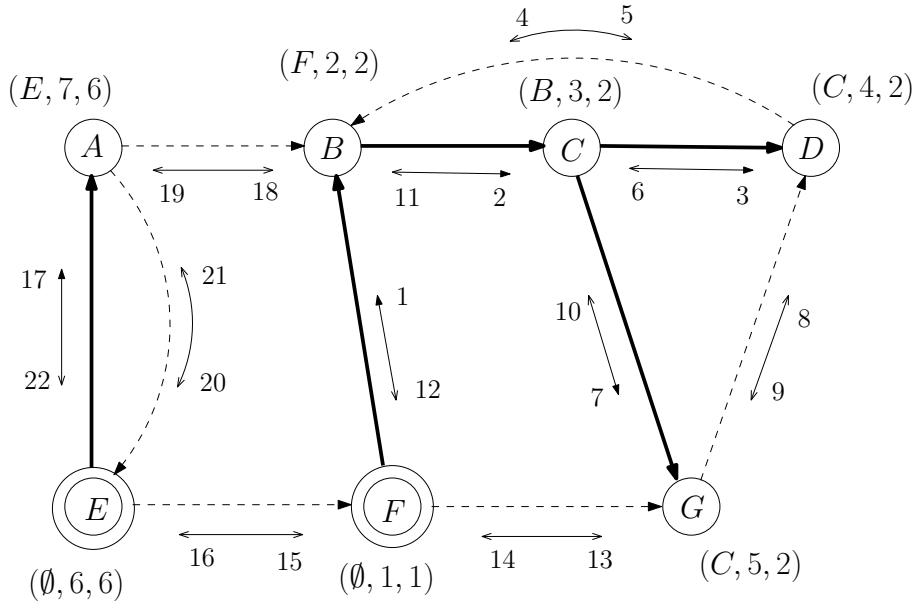
Na obr. 2.6 je (na konci výpočtov) v každom silnom komponente dolné číslo každého vrchola rovné poradovému číslu koreňa (prvého navštíveného vrchola) silného komponentu.

Cvičenie 2.8. Ukážte na príklade digrafu s 3 vrcholmi a 4 hranami, že vždy to tak nemusí byť.

Správnosť algoritmu vyplýva z nasledovného tvrdenia.

VETA 2.3. *Ak sme vo vrchole v pred návratom k otcovi, tak v je koreň koncového silného komponentu priebežného digrafu* $\Leftrightarrow p(v) = r(v)$.

Dôkaz: ($\Rightarrow$) Pri objavení vrchola v sme položili $r(v) = p(v)$, a preto stačí vylúčiť možnosť zmenšenia čísla $r(v)$. To je však zrejmé, lebo všetky vrcholy tohoto silného komponentu sú potomkovia koreňa v (a teda majú väčšie poradové



Obr. 2.6: Hľadanie silných komponentov. Prvý štart z vrchola F vypúšťa zo zásobníka (F, B, C, D, G) silný komponent s vrcholmi B, C, D, G a potom silný komponent na vrchole F; druhý štart z vrchola E dáva silný komponent s vrcholmi E, A

čísla) a do iných silných komponentov nedosiahneme, lebo náš silný komponent je koncový.

($\Leftarrow$) Predpokladajme, že vrchol v nie je koreňom. Potom v tomto silnom komponente máme koreň $u \neq v$ a jeho poradové číslo je menšie ako $p(v)$. Preto u nie je potomkom vrchola v. Ďalej v tomto silnom komponente existuje v-u cesta a nech w_2 je prvý vrchol na nej, ktorý nie je potomkom vrchola v. Preto vrchol w_2 bol už navštívený pred vrcholom v, a teda $r(w_2) \leq p(w_2) < p(v)$. Potom jeho predchodca w_1 na tejto ceste je potomkom a hrana (w_1, w_2) je alebo spätná alebo krížová a dolné číslo vrchola w_2 bolo skúmané na zníženie dolného čísla vrchola w_1. Toto dolné číslo sa potom (pri návratoch k otcom) použilo pri aktualizácii dolného čísla vrchola v a bolo by $r(v) < p(v)$, čo je spor. ■

Cvičenie 2.9. Ukážte, že ak je digraf zadaný zoznamom odchádzajúcich vrcholov, tak uvedený postup hľadania silných komponentov dáva lineármy algoritmus.

Cvičenie 2.10. Navrhnite lineárny algoritmus na zostrojenie kondenzácie digrafu.

2.4 Eulerovské ťahy

V tarryovskom slede sa každá hrana vyskytuje práve dvakrát. Možno sa pýtať, či existuje úspornejší prieskum kde každou hranou prejdeme práve raz, navštívime všetky vrcholy a vrátime sa na začiatok. To zodpovedá existencii uzavretého eulerovského ťahu a odpoveď dáva nasledujúce tvrdenie (Euler).

VETA 2.4. *Graf má uzavretý eulerovský ťah $\Leftrightarrow$ ak je súvislý a každý vrchol má párny stupeň.*

Dôkaz: ($\Rightarrow$) Taký ťah začína a končí v nejakom vrchole s a prechádza všetkými vrcholmi, a preto graf musí byť súvislý. Ak v rámci tohoto ťahu prídeme do nejakého vrchola v hranou e_1, tak z neho odídeme hranou $e_2 \neq e_1$ a tak tieto hrany tvoria dvojicu pri vrchole v. Okrem toho, prvá a posledná hrana ťahu tiež tvoria dvojicu pri vrchole s. Teda pri každom vrchole možno hrany rozložiť do dvojíc, a preto stupeň každého vrchola je párny.

($\Leftarrow$) Začnime v ľubovoľnom vrchole u a skonštruujme nejaký nepredĺžiteľný ťah T. Vzhľadom na párnosť stupňov vrcholov, tento ťah musí skončiť v začiatočnom vrchole. Ak uzavretý ťah T neobsahuje všetky hrany pri nejakom navštívenom vrchole u_1 tak v takom vrchole prerušíme ťah T a cestujúc po doteraz neprejdených hranách vytvoríme nepredĺžiteľný ťah T_1 končiaci vo vrchole u_1 (lebo pri každom vrchole je počet neprejdených hrán párny). Teraz môžeme vsunúť ťah T_1 do T a vytvoriť dlhší ťah. Takéto predlžovanie možno opakovať. Preto môžeme predpokladať, že ťah T už obsahuje všetky hrany pri každom navštívenom vrchole. Predpoklad súvislosti grafu potom implikuje, že všetky vrcholy sme už v rámci ťahu T navštívili a teda T je uzavretý eulerovský ťah. ■

Digraf sa nazýva rovnovážne orientovaný, ak pre každý vrchol v sa počet prichádzajúcich šípov rovná počtu odchádzajúcich šípov, t.j. $deg^-(v) = deg^+(v)$.

VETA 2.5. *Digraf má uzavretý eulerovský ťah $\Leftrightarrow$ ak je silne súvislý a rovnovážne orientovaný.*

Dôkaz:

Cvičenie 2.11.

■

Cvičenie 2.12. Na základe viet 2.4, 2.5 a ich dôkazov navrhnite efektívny algoritmus pre eulerovský ťah (a) v grafe, (b) v digrafe.

2.5 Orientácie grafov

Pripomeňme si, že pod orientáciou grafu rozumieme digraf, ktorý vznikne ľubovoľným zorientovaním všetkých jeho hrán. Dobrou motiváciou je potreba zorientovať ulice v meste, t.j. z obojsmerných urobiť jednosmerné ulice. Pravda, treba zachovať dosiahnuteľnosť.

SILNÁ ORIENTÁCIA GRAFU

Základná požiadavka v komunikačných sieťach je vzájomná dosiahnuteľnosť vrcholov. V grafe to zodpovedá súvislosti a v digrafe silnej súvislosti. Keďže orientovaním hrán nemôžeme vytvoriť novú dosiahnuteľnosť, ale skôr pokaziť starú, tak sa dostávame k nasledovnému tvrdeniu.

VETA 2.6. *Graf má silne súvislú orientáciu $\Leftrightarrow$ keď je súvislý a neobsahuje most.*

Dôkaz: ($\Rightarrow$) Zrejmé.

($\Leftarrow$) Podľa vety 1.4 náš graf G obsahuje nejaký cyklus. Zorientovaním všetkých hrán tohoto cyklu v smere obiehania získame silne súvislý digraf S, ktorý tvorí zárodok hľadaného digrafu. Vo všeobecnosti, nech S je netriviálny silne súvislý digraf, ktorý vznikol zorientovaním nejakého podgrafu pôvodného grafu. Ak S zahŕňa všetky vrcholy, tak je hľadanou orientáciou. Inak v G existuje vrchol mimo S a aj nejaká cesta spájajúca tento vrchol s S, lebo graf G je súvislý. Na tejto ceste existuje taká hrana $[u, v]$, že u patrí do S a v je mimo S. Táto hrana tiež leží v nejakom cykle, a preto cestujúc po tomto cykle z vrchola u, potom do v, atď. až po prvý vrchol w ležiaci v S a súčasne orientujúc hrany v smere cestovania, získame po ich pridaní do S väčší zárodok (lebo bude silne súvislý ako sa ľahko nahliadne). Opakovaním tohoto postupu dostaneme požadovanú silne súvislú orientáciu S obsahujúcu všetky vrcholy. Ľubovoľným zorientovaním hrán mimo S a pridaním týchto šípov k S už dostaneme silne súvislú orientáciu celého pôvodného grafu G. ■

Hneď je vidieť, že uvedený dôkaz platí aj pre multigrafy a pseudografy.

Častejší prípad v realite je potreba zjednosmerniť jednu konkrétnu ulicu v cestnej sieti kde už niektoré ulice sú jednosmerné. To je motivácia pre problém zorientovania hrany v migrafe riešený nasledujúcou vetou. Uveďme, že most v migrafe je definovaný ako most v dezorientácii migrafu. Pod silne súvislou orientáciou hrany migrafu rozumieme takú jej orientáciu, že z migrafu vznikne silne súvislý (multi)migraf.

VETA 2.7. *Neorientovanú hranu e silne súvislého migrafu možno silne súvisle zorientovať $\Leftrightarrow$ hrana e nie je mostom.*

Dôkaz:

Cvičenie 2.13 (Všimnite si silné komponenty a kondenzáciu po vynechaní hrany e.)**.**

■

Cvičenie 2.14. Navrhnite algoritmus pre nájdenie silne súvislej orientácie grafu na základe dôkazu vety 2.6.

Cvičenie 2.15. Navrhnite algoritmus pre nájdenie silne súvislej orientácie hrany migrafu bez mostov podľa vety 2.7.

ORIENTÁCIA S MINIMÁLNYM DIAMETROM

Hoci pri orientácii súvislého grafu G bolo hlavným cieľom získať silne súvislý digraf D, vzdialenosti medzi niektorými vrcholmi sa môžu veľmi predĺžiť. Preto vznikli zosilnené požiadavky na výsledný digraf. Jednou z nich je aby diameter $diam(D)$ bol minimalizovaný. Ukázalo sa, že táto úloha je NP-ťažká a teda nepoznáme pre ňu efektívny (polynomiálny) algoritmus. Pre špeciálne grafy optimálnu orientáciu niekedy dokážeme ľahko nájsť ako ilustrujú nasledujúce cvičenia.

Cvičenie 2.16. Pre každé $n \geq 3$ nájdite orientáciu s minimálnym diametrom pre kompletný graf K_n na n vrcholoch (ktorý má diameter 1).

Cvičenie 2.17. Pre každý graf z obr. 1.7 zistite diameter a tiež minimálny diameter orientácie.

Cvičenie 2.18. Nájdite graf na 7 vrcholoch, ktorý má diameter 2, ale minimálny diameter jeho orientácie je 6.

Kapitola 3

OPTIMÁLNE CESTY

V dnešnej dobe sa veľmi často cestuje. Nejde iba o prepravu osôb, ale najmä prepravu materiálov a výrobkov. Vo všeobecnosti máme danú „dopravnú sieť“, čo u nás bude znamenať najakú grafovú štruktúru s prípadnými ohodnoteniami hrán, začiatok a koniec cesty. Takých ciest môže byť veľmi mnoho a cieľom je vybrať takú trasu, ktorá je v istom zmysle optimálna. Najčastejšie sa stretávame s úlohou vybrať najkratšiu trasu v cestnej sieti keď chceme cestovať z jednej obce do druhej. Ale oboznámime sa aj s inými úlohami.

3.1 Najkratšie cesty

Tu budeme predpokladať, že každá hrana má nejaké reálne ohodnotenie, ktoré budeme volať dĺžka. Potom pod dĺžkou sledu sa rozumie súčet dĺžok všetkých hrán sledu (každú hranu započítame pri každom jej výskyte v slede). Obmedzíme sa na digrafy, lebo vo väčšine reálnych úloh môžeme neorientovanú hranu nahradiť dvojicou protišípov (tak ako je obojsmerný cestný úsek stredovou deliacou čiarou rozdelený na dva jednosmerné úseky). Z ekonomického hľadiska je vlastne dôležité nájsť najkratší sled, ale v prakticky zaujímavých situáciach sa to väčšinou redukuje na hľadanie najkratšej cesty. Všimneme si niekoľko špeciálnych úloh a algoritmov pre ne. Namiesto očakávanej úlohy pre dané vrcholy s a t nájsť najkratšiu s-t cestu, budeme často riešiť všeobecnejšiu úlohu: **z jedného vrchola do ostatných**, t.j. pre každý vrchol v nájsť dĺžku najkratšej s-v cesty a tiež samotnú cestu. Dokonca ešte všeobecnejšie: **z každého vrchola do každého**, t.j. pre každé dva vrcholy u a v nájsť dĺžku najkratšej u-v cesty, resp. samotnú cestu. Dôvod je ten, že príslušný algoritmus pri riešení základnej úlohy už vlastne vyrieši aj takúto všeobecnejšiu úlohu.

Cvičenie 3.1. Ak by sme pre dané dva vrcholy s a t našli všetky s-t cesty a potom ich dĺžky, tak zrejme by sme našli aj najkratšiu s-t cestu. Ukážte na príklade neohodnoteného digrafu, že takých ciest môže byť exponenciálne mnoho (vzhľadom k $n + m$), a teda uvedený nápad nedáva efektívny algoritmus.

MOOROV ALGORITMUS

Tu predpokladáme, že je daný neohodnotený digraf (t.j. dĺžka každej hrany je 1) a začiatočný vrchol s. Treba vyriešiť úlohu najkratšej cesty z vrchola s do ostatných.

Na začiatku sú všetky vrcholy neoznačkované a postupom do šírky všetky vrcholy dosiahnuteľné z s označkujemene. Značka nejakého vrchola v bude usporiadaná dvojica (d_v, p_v), kde d_v je vzdialenosť vrchola v od s a p_v je predchodca vrchola v, z ktorého sme vrchol v objavili. Algoritmus pozostáva z dvoch fáz. V prvej fáze nájdeme vzdialenosti od vrchola s a pripravíme značky na identifikáciu ciest. V druhej potom nájdeme požadovanú cestu (cesty). Pre rozhodujúcu prvú fázu realizujeme nasledovné kroky.

1. (Inicializácia): Položíme $k = 0$ a vrchol s dostane značku $(0, \emptyset)$.

2. (Iterácia): Pokiaľ existuje vrchol u so značkou $d_u = k$ tak robíme: pre každý šíp (u, v) s neoznačkovaným vrcholom v priradíme vrcholu v značku $(k + 1, u)$.

3. (Rozhodovanie): Ak bol v kroku 2 označkovaný aspoň jeden vrchol, tak položíme $k = k + 1$ a ideme na krok 2. Inak je koniec 1. fázy.

V druhej fáze potom pre každý predpísaný vrchol v nájdeme najkratšiu s-v cestu spätným postupom: predposledný vrchol cesty bude predchodca vrchola v, potom nájdeme predchodcu vrchola p_v, potom jeho predchodcu, atď. až sa dostaneme do vrcholu s.

Obr. 3.1 ilustruje Moorov algoritmus. Ak je digraf zadaný pomocou zoznamu susedných vrcholov, tak celý proces priraďovania vzdialeností a predchodcov zvládneme v čase $O(m+n)$. Identifikácia s-v cesty sa dá zrealizovať v čase $O(n)$ pre jeden vrchol v (lebo cesta má nanajvýš n vrcholov).

Cvičenie 3.2. Zdôvodnite Moorov algoritmus.

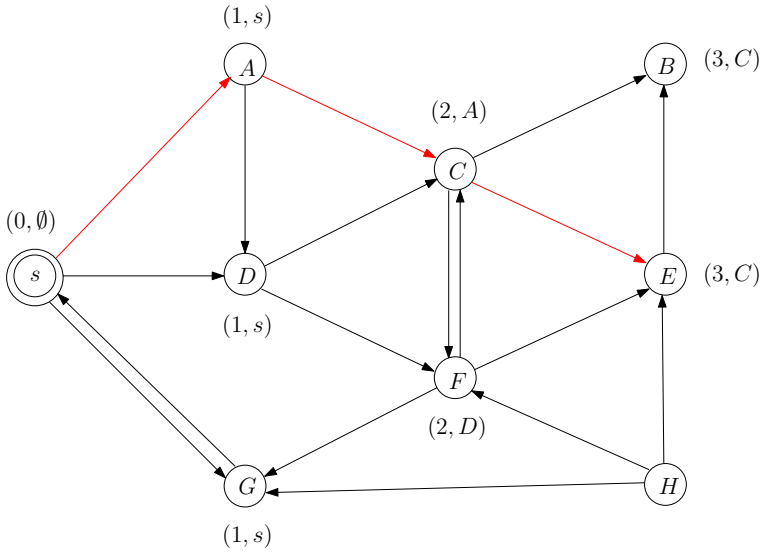
DIJKSTROV ALGORITMUS

Tu predpokladáme, že je daný digraf G s dĺžkami hrán $c_{ij} \geq 0$ a začiatočný vrchol s. Treba vyriešiť úlohu najkratšej cesty z vrchola s do ostatných.

V tomto algoritme tiež značkujeme vrcholy, ale značky sú dočasné a postupne sa zlepšujú. Každý vrchol v bude mať značku (d_v, p_v), kde d_v je dĺžka nejakej najkratšej priebežne známej s-v cesty a p_v je jej predposledný vrchol.

1. (Inicializácia): Vrchol s dostane značku $(0, \emptyset)$ a každý iný vrchol značku $(\infty, \emptyset)$. Položíme $S = \emptyset$ a $\bar{S} = V(G) \setminus S$.

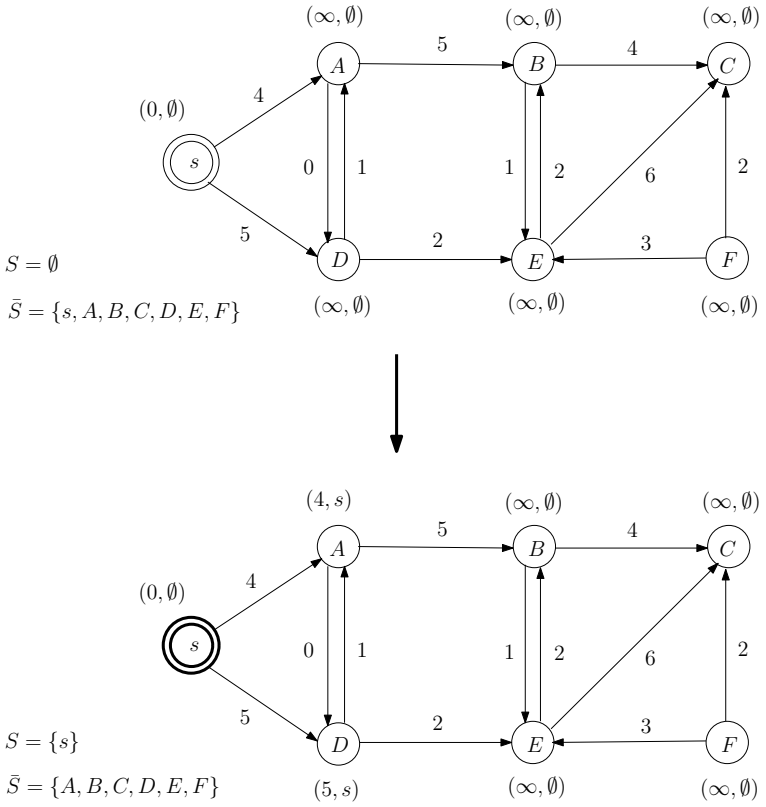
2. (Iterácia): Pokiaľ v množine $\bar{S}$ existujú vrcholy s konečnou prvou zložkou, tak nájdeme taký vrchol $k \in \bar{S}$, ktorý má minimálnu prvú zložku d_k a pre každý šíp (k, j), kde $j \in \bar{S}$ urobíme: ak $d_k + c_{kj} < d_j$, tak položíme $d_j = d_k + c_{kj}$ a $p_j = k$; položíme $S = S \cup \{k\}$, $\bar{S} = \bar{S} \setminus \{k\}$ a ideme na krok 2.



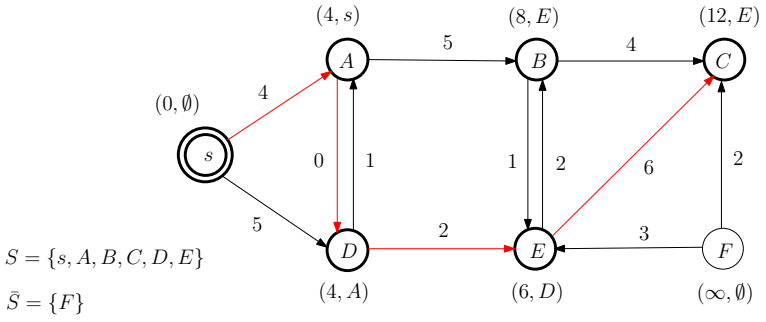
Obr. 3.1: Moorov algoritmus; nájdená najkratšia s-E cesta je červená

3. (Ukončenie): Inak (t.j. ak $\bar{S}$ neobsahuje vrcholy s konečnou prvou zložkou), tak je koniec 1. fázy.

Po skončení 1. fázy nastupuje nájdenie najkratšej s-v cesty spätným postupom na základe druhých zložiek značiek.



Obr. 3.2: Dijkstrov algoritmus - prvá iterácia



Obr. 3.3: Dijkstrov algoritmus - koniec; nájdená najkratšia s-C cesta je červená

Správnosť Dijkstrovho algoritmu vyplýva z nasledujúcej vety.

VETA 3.1. *(a) Ak v Dijkstrovom algoritme vrchol k vstupuje do množiny S,*

tak jeho značka je už trvalá a platí: d_k je dĺžka najkratšej s-k cesty a p_k je jej predposledný vrchol.

(b) Ak po skončení algoritmu zostal vrchol $w \in \bar{S}$, tak neexistuje s-w cesta.

Dôkaz: (a) Použijeme indukciu podľa poradia v akom vrcholy vstupujú do množiny S. Pre vrchol s je tvrdenie zrejme pravdivé. Z algoritmu vidíme, že ak je d_k konečné, tak sme objavili nejakú s-k cestu, ktorej predposledný vrchol je p_k. Zostáva dokázať, že neexistuje kratšia cesta. Nech P je najkratšia s-k cesta. Keďže $s \in S$ a $k \in \bar{S}$, tak na ceste P existuje taký šíp (u, v), že $u \in S$ a $v \in \bar{S}$. Týmto sa cesta P rozloží na 3 úseky: P_{s-u}, šíp (u, v) a P_{v-k}. Teda pre dĺžku cesty P platí: $L(P) = L(P_{s-u}) + c_{uv} + L(P_{v-k})$. Odhadneme dĺžky jednotlivých úsekov. Podľa indukčného predpokladu vrchol u už má trvalú značku. Preto každá najkratšia s-u cesta, a teda aj s-u úsek cesty P, má dĺžku d_u. V čase keď vrchol u vstupoval do množiny S bol skúmaný aj šíp (u, v), a preto $d_v \leq d_u + c_{uv}$. Potom sa d_v už mohlo len zmenšiť, a preto sa táto nerovnosť zachová. Keďže dĺžky všetkých šípov sú nezáporné, tak $L(P_{v-k}) \geq 0$. Sumarizujúc máme:

$$L(P) = d_u + c_{uv} + L(P_{v-k}) \geq d_v \geq d_k,$$

lebo podľa algoritmu je d_k minimálne v množine $\bar{S}$.

(b) Nech pre spor existuje nejaká s-w cesta Q. Jej prvý vrchol $s \in S$ a teda podľa algoritmu aj jej druhý vrchol dostane konečnú značku a teda tento vrchol sa niekedy dostane do množiny S. Analogicky sa dostanú do S všetky vrcholy cesty Q, čo je spor. ■

Odhadnime *zložitosť* Dijkstrovho algoritmu. V každej iterácii sa dostane jeden vrchol do množiny S, a teda celkove máme $O(n)$ iterácií. V každej iterácii na nájdenie vrchola k vystačí $O(n)$ operácií a na aktualizáciu značiek takisto $O(n)$ operácií (lebo odchádzajúci stupeň vrchola k je nanajvýš $n-1$). Sumárne na iteráciu stačí $O(n)$ operácií a teda celková zložitosť algoritmu je $O(n^2)$.

Cvičenie 3.3. Ukážte, že použitím haldy možno znížiť zložitosť Dijkstrovho algoritmu na $O((m+n)\log_2 n)$.

Cvičenie 3.4. Nájdite príklad acyklického digrafu so zápornými dĺžkami šípov kde Dijkstrov algoritmus nedáva správny výsledok.

ACYKLICKÝ ALGORITMUS

Tu predpokladáme, že je daný acyklický digraf G s ľubovoľnými reálnymi dĺžkami hrán c_{ij} a začiatočný vrchol s. Treba vyriešiť úlohu najkratšej cesty z vrchola s do ostatných.

V tomto algoritme tiež značkujeme vrcholy, ale značky sú usporiadané trojice, ktoré sú dočasné a postupne sa zlepšujú. Každý vrchol v bude mať značku (d_v, p_v, q_v), kde d_v je dĺžka nejakej najkratšej priebežne známej s-v cesty, p_v je jej predposledný vrchol a q_v je počet nepreskúmaných šípov prichádzajúcich

do v. Okrem toho budeme používať zásobník vrcholov Z pripravených na skúmanie. Pred aplikáciou vlastného algoritmu z digrafu vynecháme všetky šípy prichádzajúce do s, čím sa s stane prameňom. Ostatné pramene možno vynechať a takisto aj vzniknuté nové pramene. Dôvod je ten, že tieto prvky digrafu nemôžu ležať v žiadnej hľadanej ceste. Avšak algoritmus dokáže pracovať aj bez takéhoto očistenia od prameňov rôznych od s.

1. (Inicializácia): Vynecháme všetky šípy prichádzajúce do s. Vrchol s dostane značku $(0, \emptyset, 0)$ a každý iný vrchol v značku $(\infty, \emptyset, deg^-(v))$. Do zásobníka Z vložíme všetky pramene rôzne od s a nakoniec vrchol s. Všetky vrcholy a šípy sú nepreskúmané.

2. (Iterácia): Ak je zásobník Z neprázdny, tak vyberieme z neho posledný element u a preskúmame každý jeho odchádzajúci šíp (u, v) nasledovne. Položíme $q_v = q_v - 1$; ak $d_u + c_{uv} < d_v$, tak položíme $d_v = d_u + c_{uv}$ a $p_v = u$; ak $q_v = 0$, tak vrchol v dáme do zásobníka Z. Ideme na krok 2.

3. (Ukončenie): Inak (t.j. ak je zásobník Z prázdny), tak je koniec 1. fázy.

Po skončení 1. fázy nastupuje nájdenie najkratšej s-v cesty spätným postupom na základe druhých zložiek značiek.

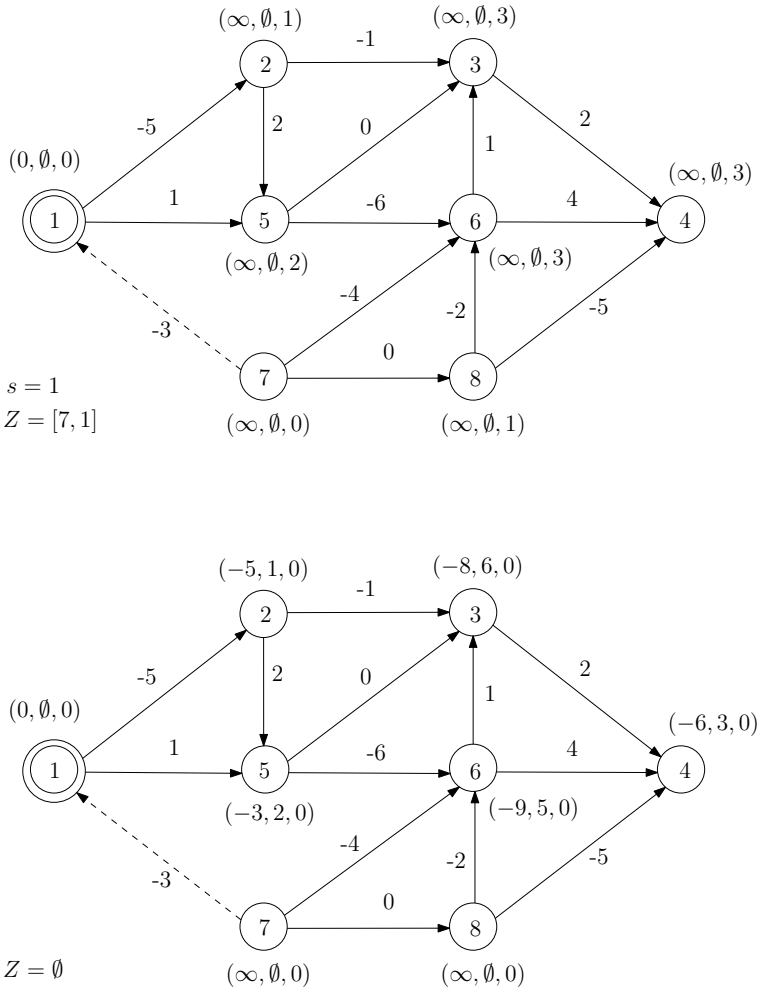
VETA 3.2. *V acyklickom algoritme sa každý vrchol v niekedy dostane do zásobníka a v tom čase je jeho značka už trvalá a platí: (a) Ak d_v je konečné, tak je to dĺžka najkratšej s-v cesty a p_v je jej predposledný vrchol.*
(b) Inak (t.j. ak $d_v = \infty$), tak neexistuje s-v cesta.

Dôkaz: Na začiatku dáme všetky pramene do zásobníka a po ich vybraní zo zásobníka preskúmame všetky odchádzajúce šípy z týchto vrcholov, čo si môžeme predstaviť ako odstránenie prameňov z pôvodného digrafu. Zvyšný digraf je acyklický, a teda má pramene, ktoré sa pri skúmaní pôvodných prameňov dostanú do zásobníka atď. Teda každý vrchol sa dostane do zásobníka.

(a) Dokazujeme indukciou podľa poradia v akom idú vrcholy do zásobníka. Pre začiatočné vrcholy v zásobníku je tvrdenie zrejmé. Nech P je najkratšia s-v cesta a nech (u, v) je jej posledný šíp. Keďže vrchol v ide do zásobníka, tak všetky prichádzajúce šípy do neho už boli preskúmané, a teda aj šíp (u, v), ktorý mohol byť skúmaný iba z vrchola u. Preto vrchol u už bol v zásobníku, a teda podľa indukčného predpokladu d_u je dĺžka najkratšej s-u cesty a preto aj dĺžka s-u úseku cesty P. Podľa algoritmu platí: $d_v \leq d_u + c_{uv}$, a preto d_v sa rovná dĺžke cesty P, lebo menšie ako dĺžka najkratšej cesty to nemôže byť.

(b) Pre spor nech existuje nejaká s-v cesta P . Potom podľa algoritmu sa postupne vrcholu v priradí konečná hodnota d_v (ak by sa použili iba hrany cesty P, tak by sa d_v rovnalo jej dĺžke, ale inak to môže byť ešte menej), čo je spor. ■

Cvičenie 3.5. Predpokladajte, že digraf (aj s ohodnoteniami šípov) je zadaný pomocou zoznamu odchádzajúcich vrcholov. Ukážte, že acyklický algoritmus má lineárnu zložitosť, t.j. $O(m + n)$.



Obr. 3.4: Príklad ilustrujúci acyklický algoritmus: pred prvou iteráciou a po poslednej iterácii

Cvičenie 3.6. Uveďte nejaký lineárny algoritmus pre zistenie toho, či daný digraf je acyklický.

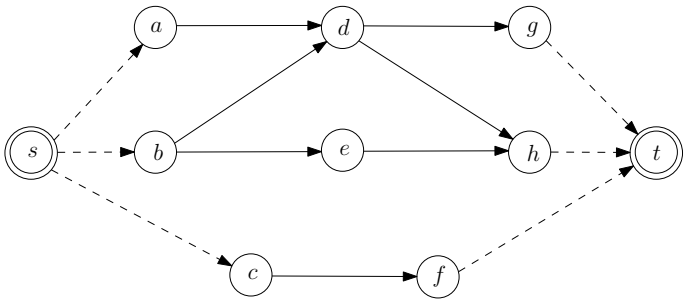
Cvičenie 3.7. Preskúmajte čo sa stane, ak by sme aplikovali acyklický algoritmus na digraf obsahujúci cyklus.

ČASOVÁ ANALÝZA PROJEKTU

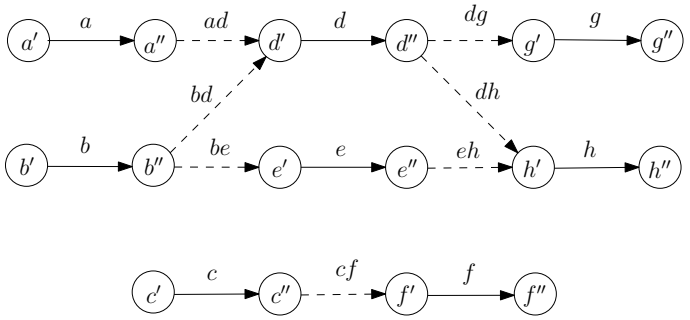
Pod projektom sa obvykle rozumie komplex dielčích činností spolu s informáciami o jednotlivých činnostiach a väzbách medzi nimi. Túto vágnu definíciu nahradíme presnejšou po objasnení na príklade. Uvažujme napr. výstavbu budovy. Tá zahŕňa množstvo dielčích činností (budovanie inžinierskych sietí, kopanie základov, betónovanie základov, výstavba múrov, konštrukcia strechy, omietanie múrov, vodoinštalácie, elektroinštalácie, zhotovenie podláh, maľovanie, atď. Niektoré činnosti možu prebiehať paralelne (napr. vodoinšalácie a elektroinštalácie), ale niektoré iné sa môžu realizovať iba v danom poradí. Napr. maľovať môžeme až po výstavbe múrov. Teda činnosti sú čiastočne usporiadané reláciou "predchádzať". Z úsporných dôvodov zadávame iba reláciu "bezprostredne predchádzať", lebo zrejme platí tranzitívnosť. (Napr. to, že maľovaniu predchádza výstavba múrov vyplýva z toho, že maľovaniu predchádza omietanie a omietaniu výstavba múrov.) Bezprostredné predchádzanie môžeme celkom prirodzene zobraziť digrafom činností tak, ako je to urobené pre príklad 3.8 na obr. 3.5.

Príklad 3.8.

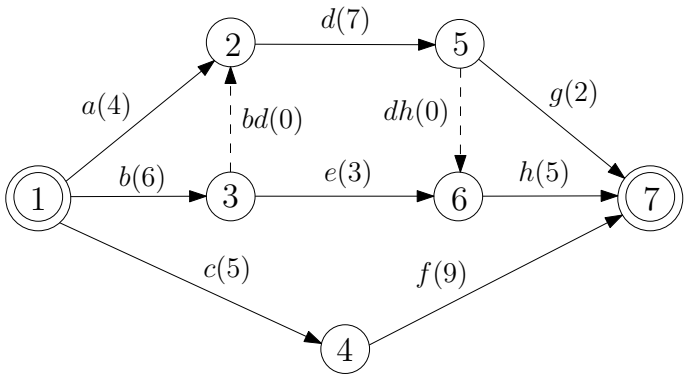
činnosť	trvanie	bezprostredný predchodca
a	4	
b	6	
c	5	
d	7	a, b
e	3	b
f	9	c
g	2	d
h	5	d, e



Obr. 3.5: Digraf s vrcholovými činnosťami pre príklad 3.8



Obr. 3.6: Medzikrok konštrukcie digrafu s hranovými činnosťami pre príklad 3.8



Obr. 3.7: Digraf projektu s hranovými činnosťami pre príklad 3.8

Tam činnostiam zodpovedajú vrcholy a ak činnosti y bezprostredne predchádza činnosť x, tak tam dáme šíp (x, y); naviac, pridáme 2 fiktívne činnosti s (začiatok projektu) a t (koniec projektu) pričom s bude predchádzať všetky činnosti (stačí spojiť do tých, ktoré nemali predchodcov; pozri čiarkované šípy) a t bude nasledovať po všetkých činnostiach (stačí keď budú prichádzať šípy z činností bez nasledovníkov; pozri čiarkované šípy). Tieto fiktívne činnosti budú mať nulové trvanie. Pre analýzu projektu budeme však používať model, v ktorom sú činnosti reprezentované šípmi a potom trvanie c_{ij} činnosti (i, j) budeme pokladať za dĺžku šípu. Z modelu s vrcholovými činnosťami sa ľahko získa model s hranovými činnosťami (pozri obr. 3.5, 3.6 a 3.7). Takýto model budeme nazývať **digraf činností**, resp. **digraf projektu**.

Ak by digraf projektu obsahoval cyklus, tak činnosti tohoto cyklu (a tým celý projekt) by sa nedali zrealizovať (lebo činnosť sa môže začať až po skončení predchádzajúcej činnosti v cykle). Teda budeme predpokladať, že **digraf projektu je acyklický**. Okrem toho z konštrukcie digrafu projektu vidíme, že každým šípom prechádza nejaká s-t cesta.

Základnou otázkou pri skúmaní digrafu projektu je **trvanie projektu**, čo je minimálny čas, za ktorý sa dá projekt zrealizovať. Hneď vidíme, že trvanie projektu je aspoň také ako je dĺžka ľubovoľnej s-t cesty. Takže dĺžka najdlhšej s-t cesty je dolnou medzou na trvanie projektu. Preto pri skúmaní projektu budeme hľadať najdlhšie cesty. Ak prenásobíme všetky trvania činností číslom -1, tak môžeme aplikovať acyklický algoritmus. Ale lepšie je adaptovať acyklický algoritmus pre hľadanie najdlhších ciest v pôvodnom digrafe projektu.

Cvičenie 3.9. Urobte to.

Takýto algoritmus budeme nazývať **acyklický algoritmus pre najdlhšie cesty**. Horeuvedené pozorovanie možno zosilniť:

VETA 3.3. *Trvanie projektu sa rovná dĺžke najdlhšej s-t cesty.*

Dôkaz: Nech T je trvanie projektu a L dĺžka najdlhšej s-t cesty. Už vieme, že $T \geq L$. Ukážeme obrátenú nerovnosť. Ak aplikujeme acyklický algoritmus pre najdlhšie cesty, tak pri každom vrchole v získame dĺžku d_v najdlhšej s-v cesty, čo môžeme interpretovať nasledovne. Začínajúc v čase $d_s = 0$ všetky činnosti prichádzajúce do vrchola v možno zrealizovať do času d_v. (To je vidieť indukciou podľa poradia, v ktorom vrcholy idú do zásobníka, lebo $d_v \geq d_u + c_{uv}$ pre každý šíp (u, v).) Teda $T \leq d_t = L$. ■

Významnou informáciou o činnosti (i, j) je tzv. **rezerva činnosti** r_{ij}, ktorá predstavuje maximálne predĺženie trvania c_{ij} bez toho, aby sa predĺžilo trvanie projektu (pri nezmenených trvaniach ostatných činností). Podľa vety 3.3 stačí vyšetrovať najdlhšie s-t cesty prechádzajúce šípom (i, j). Každá takáto cesta sa skladá z 3 úsekov: najdlhšej s-i cesty, šípa (i, j) a najdlhšej j-t cesty. Prvý úsek má dĺžku d_i a druhý $c_{ij} + r_{ij}$. Ak označíme dĺžku tretieho symbolom $\bar{d}_j$, tak dostávame vzťah

$$d_i + c_{ij} + r_{ij} + \bar{d}_j = d_t, \tag{3.1}$$

odkiaľ možno r_{ij} vypočítať. Totiž, čísla d_i a d_t získame použitím acyklického algoritmu pre najdlhšie cesty na digraf projektu a $\bar{d}_j$ možno získať tiež týmto algoritmom a tak rezervy všetkých činností možno vypočítať v čase $O(m + n)$.

Cvičenie 3.10. Dokážte.

Činnosti s nulovou rezervou sa v praxi nazývajú **kritické** (lebo predĺženie takejto činnosti má za následok predĺženie trvania celého projektu). Zrejme každá najdlhšia cesta v digrafe projektu pozostáva iba z kritických činností, a preto sa tiež nazýva **kritická cesta**.

Cvičenie 3.11. Dokážte, že činnosť je kritická práve vtedy, ak leží na kritickej ceste.

Cvičenie 3.12. V úlohe o celočíselnom batohu sú dané celé čísla $a_j, c_j > 0 \quad \forall j = 1, 2, ..., n$ a celé číslo $b > 0$; treba vyriešiť úlohu celočíselného lineárneho programovania: $\max\{\sum_{j=1}^{n} c_j x_j | \sum_{j=1}^{n} a_j x_j \leq b, x_j \geq 0$ a celé $\quad \forall j\}$. Prevaďte úlohu o celočíselnom batohu na úlohu nájsť najdlhšiu s-t cestu v acyklickom digrafe. [Návod: Digraf bude mať vrcholy 0, 1, ..., b; z vrchola $i = 0, 1, ..., b-1$ vedieme šíp do vrcholu $i + a_j \leq b$ a dáme mu dĺžku $c_j \quad \forall j = 1, 2, ..., n$; potom ešte pre každý taký vrchol, z ktorého nevedie šíp do b pridáme takýto šíp a dáme mu dĺžku 0; položíme $s = 0$ a $t = b$.] Doriešte detaily a prediskutujte náročnosť tejto transformácie.

Ukazuje sa, že predpoklad nezápornosti dĺžok v Dijkstrovom algoritme a predpoklad neexistencie cyklov v acyklickom algoritme možno spoločne oslabiť tak, že pripustíme ľubovoľné reálne dĺžky, ale zakážeme cykly zápornej dĺžky. Algoritmus typu z jedného vrchola do ostatných pri uvedenom predpoklade existuje a je známy ako Bellmanov alebo Fordov algoritmus a je založený na postupe do šírky. Tu okrem čísel d_v a p_v sa ešte uvažujú čísla a_v. Potom v k-tej iterácii ($k = 1, 2, ..., n-1$) sa dĺžka d_v najkratšej s-v cesty s počtom hrán $a_v \leq k-1$ porovnáva s dĺžkou každej s-v cesty s počtom hrán k a ak je dlhšia, tak sa ňou nahradí. Algoritmus má zložitosť $O(mn) \leq O(n^3)$. Tu uvedieme iný algoritmus, ktorý dokáže ešte viac.

FLOYDOV ALGORITMUS

Tu predpokladáme, že je daný digraf $G = (V, E)$ s ľubovoľnými reálnymi dĺžkami hrán c_{ij} neobsahujúci záporný cyklus. Treba vyriešiť úlohu najkratšej cesty z každého vrchola do každého.

V tomto algoritme budeme predpokladať, že vrcholy sú nejako usporiadané. Kvôli jednoduchosti nech vrcholová množina $V = \{1, 2, ..., n\}$. Hovoríme, že nejaký sled je typu W^k ak každý jeho vnútorný vrchol (t.j. vrchol rôzny od začiatku a konca sledu) patrí do množiny $\{1, 2, ..., k\}$. Algoritmus bude pracovať na maticiach typu (n, n). Pre každé $k = 0, 1, ..., n$ budeme mať v priebežnej matici $D^{(k)}$ s prvkami $d_{ij}^{(k)}$ dĺžky najkratších ciest a v inej matici $P^{(k)}$ s prvkami $p_{ij}^{(k)}$ predposledné vrcholy týchto ciest (pre identifikáciu ciest). Presnejšie, $d_{ij}^{(k)}$ je dĺžka najkratšej i-j cesty typu W^k a $p_{ij}^{(k)}$ je jej predposledný vrchol. Z matíc $D^{(k-1)}$ a $P^{(k-1)}$ zostrojíme matice $D^{(k)}$ a $P^{(k)}$ na základe nasledujúceho tvrdenia.

VETA 3.4. *Nech digraf neobsahuje záporný cyklus. Ak i-j sled, ktorý je zreťazením ľubovoľnej najkratšej i-k cesty typu W^{k-1} a ľubovoľnej najkratšej k-j cesty typu W^{k-1} má menšiu dĺžku ako najkratšia i-j cesta typu W^{k-1}, tak je to najkratšia i-j cesta typu W^k.*

Dôkaz: Pre ľubovoľnú najkratšiu i-j cestu typu W^k zrejme platí: buď je typu W^{k-1}, alebo je zreťazením nejakej najkratšej i-k cesty typu W^{k-1} a nejakej najkratšej k-j cesty typu W^{k-1}. Ak náš sled S nie je cestou, tak obsahuje cyklus prechádzajúci iba raz vrcholom k. Preto po vynechaní tohoto cyklu získame i-j sled S' typu W^{k-1}, takže po vynechaní ďalších cyklov (ak existujú) nakoniec získame i-j cestu P typu W^{k-1}. Všetky vynechané cykly mali nezápornú dĺžku

(lebo záporné neexistujú), a preto cesta P má dĺžku nanajvýš takú ako má sled S. Takto P má dĺžku menšiu ako najkratšia i-j cesta typu W^{k-1}, čo je nemožné. Preto S je i-j cesta typu W^k. Ak by existovala kratšia i-j cesta typu W^k, tak aspoň jeden z jej úsekov i-k a k-j, ktoré sú typu W^{k-1}, by bol kratší ako príslušný úsek cesty S. To však nie je možné, lebo do S sme vyberali najkratšie také úseky. ■

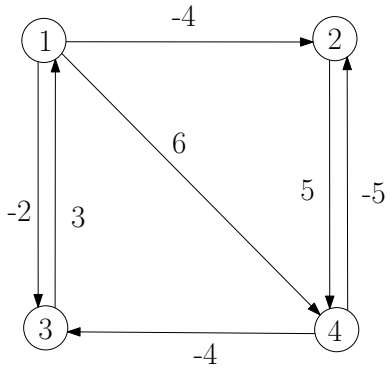
1. (Inicializácia): Položíme $k = 0$ a zostrojíme matice $D^{(0)}$ a $P^{(0)}$:

$$d_{ij}^{(0)} = \begin{cases} c_{ij}, & \text{ak} \quad (i,j) \in E \\ 0, & \text{ak} \quad i = j \\ \infty, & \text{inak} \end{cases} \qquad p_{ij}^{(0)} = \begin{cases} i, & \text{ak} \quad (i,j) \in E \\ \emptyset, & \text{inak} \end{cases}$$

2. (Iterácia): Pokiaľ $k < n$, tak položíme $k = k+1$ a pre všetky $i, j = 1, 2, ..., n$ urobíme: ak $d_{ik}^{(k-1)} + d_{kj}^{(k-1)} < d_{ij}^{(k-1)}$, tak položíme $d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$ a $p_{ij}^{(k)} = p_{kj}^{(k-1)}$; inak položíme $d_{ij}^{(k)} = d_{ij}^{(k-1)}$ a $p_{ij}^{(k)} = p_{ij}^{(k-1)}$. Ideme na krok 2.

3. (Ukončenie): Inak (t.j. ak $k = n$), tak je koniec 1. fázy.

Po skončení 1. fázy nastupuje nájdenie požadovaných najkratších i-j ciest spätným postupom na základe matice $P^{(n)}$.



Obr. 3.8: Digraf pre Floydov algoritmus

$$D^{(0)} = \begin{bmatrix} 0 & -4 & -2 & 6 \\ \infty & 0 & \infty & 5 \\ 3 & \infty & 0 & \infty \\ \infty & -5 & -4 & 0 \end{bmatrix} \qquad P^{(0)} = \begin{bmatrix} \emptyset & 1 & 1 & 1 \\ \emptyset & \emptyset & \emptyset & 2 \\ 3 & \emptyset & \emptyset & \emptyset \\ \emptyset & 4 & 4 & \emptyset \end{bmatrix}$$

$$D^{(1)} = \begin{bmatrix} 0 & -4 & -2 & 6 \\ \infty & 0 & \infty & 5 \\ 3 & -1 & 0 & 9 \\ \infty & -5 & -4 & 0 \end{bmatrix} \qquad P^{(1)} = \begin{bmatrix} \emptyset & 1 & 1 & 1 \\ \emptyset & \emptyset & \emptyset & 2 \\ 3 & \emptyset & \emptyset & 1 \\ \emptyset & 4 & 4 & \emptyset \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} 0 & -4 & -2 & 1 \\ \infty & 0 & \infty & 5 \\ 3 & -1 & 0 & 4 \\ \infty & -5 & -4 & 0 \end{bmatrix} \qquad P^{(2)} = \begin{bmatrix} \emptyset & 1 & 1 & 2 \\ \emptyset & \emptyset & \emptyset & 2 \\ 3 & 1 & \emptyset & 2 \\ \emptyset & 4 & 4 & \emptyset \end{bmatrix}$$

$$D^{(3)} = \begin{bmatrix} 0 & -4 & -2 & 1 \\ \infty & 0 & \infty & 5 \\ 3 & -1 & 0 & 4 \\ -1 & -5 & -4 & 0 \end{bmatrix} \qquad P^{(3)} = \begin{bmatrix} \emptyset & 1 & 1 & 2 \\ \emptyset & \emptyset & \emptyset & 2 \\ 3 & 1 & \emptyset & 2 \\ 3 & 4 & 4 & \emptyset \end{bmatrix}$$

$$D^{(4)} = \begin{bmatrix} 0 & -4 & -2 & 1 \\ 4 & 0 & 1 & 5 \\ 3 & -1 & 0 & 4 \\ -1 & -5 & -4 & 0 \end{bmatrix} \qquad P^{(4)} = \begin{bmatrix} \emptyset & 1 & 1 & 2 \\ 3 & \emptyset & 4 & 2 \\ 3 & 1 & \emptyset & 2 \\ 3 & 4 & 4 & \emptyset \end{bmatrix}$$

Cvičenie 3.13. Ukážte, že vo Floydovom algoritme stačí mať priestor pre jednu maticu $D^{(k-1)}$ a postupne ju meniť na maticu $D^{(k)}$ (uveďte, čo sa dá bez počítania prevziať a prečo). Podobne je to pre maticu $P^{(k-1)}$.

Cvičenie 3.14. Zistite výpočtovú zložitosť Floydovho algoritmu. Prediskutujte aj zložitosť 2. fázy.

Zistiť, či digraf obsahuje záporný cyklus je netriviálna úloha. Avšak Floydov algoritmus je v tomto smere sebestačný.

Cvičenie 3.15. Ukážte, že ak aplikujeme Floydov algoritmus na ľubovoľný digraf, tak platí: Záporný cyklus existuje $\Leftrightarrow$ niektorý element $d_{ii}^{(k)} < 0$.

Cvičenie 3.16. Ukážte, že ak vo Floydovom algoritme v začiatočnej matici $D^{(0)}$ dáme do hlavnej diagonály symboly ∞, tak algoritmus nám poskytne pre každý vrchol dĺžku najkratšieho cyklu prechádzajúceho týmto vrcholom. Uveďte podrobnosti a zdôvodnite.

Cvičenie 3.17. V algoritmoch pre najkratšie cesty sme okrem dĺžok najkratších ciest určili aj ich predposledné vrcholy pre efektívne zistenie ciest. Ukážte, že pomocou samotných dĺžok vieme najkratšie cesty tiež identifikovať v každom z uvedených algoritmov, avšak nie tak efektívne.

Predpoklad neexistencie záporného cyklu je v istom zmysle hraničný. Totiž bez neho je úloha NP-ťažká ako vidieť z nasledujúceho cvičenia.

Cvičenie 3.18. Nech U označuje úlohu pre dané dva vrcholy s a t nájsť najkratšiu s-t cestu v ľubovoľnom digrafe s reálnymi dĺžkami hrán bez akýchkoľvek obmedzení. Ukážte, že ak by sme mali polynomiálny algoritmus pre U, tak by sme vedeli vyriešiť v polynomiálnom čase každú z nasledujúcich ťažkých úloh v digrafe aj v grafe :

(i) Nájsť hamiltonovskú cestu s predpísaným začiatkom a koncom.

(ii) Nájsť hamiltonovskú cestu (s voľným začiatkom i koncom).

(iv) Nájsť hamiltonovský cyklus.

Cvičenie 3.19. Nech G je digraf s reálnymi dĺžkami c_{ij} šípov a nech sú dané dva vrcholy $s, t \in V(G)$. K úlohe nájsť najkratšiu s-t cestu priraďme nasledujúcu úlohu lineárneho programovania (LP), kde premenná x_i je dĺžka najkratšej s-i cesty, resp. jej podhodnotením.

$x_t \to \max$
$x_s = 0$
$x_j \leq x_i + c_{ij} \quad \forall (i,j) \in E(G)$

Dokážte nasledujúce tvrdenia:

(i) V úlohe LP je neprípustnosť $\Leftrightarrow$ digraf G obsahuje záporný cyklus.

(ii) V úlohe LP je neohraničenosť $\Leftrightarrow$ digraf G neobsahuje záporný cyklus a neexistuje s-t cesta.

(iii) V úlohe LP je optimalita $\Leftrightarrow$ digraf G neobsahuje záporný cyklus a existuje s-t cesta. Naviac, v prípade optimality sa optimálna hodnota účelovej funkcie rovná dĺžke najkratšej s-t cesty.

[Pomôcky: Digraf G bez záporných cyklov možno doplniť na kompletný šípmi veľkej dĺžky a použiť Floydov algoritmus. Aplikujte obmedzenia úlohy LP na šípy záporného cyklu a tiež na šípy s-t cesty v G.]

3.2 Iné optimálne cesty

Stručne uvedieme dve iné kritéria používané pri optimalizácii ciest.

NAJŠIRŠIA CESTA

V cestnej sieti niekedy potrebujeme prepraviť tzv. nadrozmerný náklad, ktorý sa do niektorej úzkej komunikácie nezmestí. Z tohoto dôvodu uvažujeme digraf, kde každý šíp (i,j) má tzv. šírku $b_{ij} \geq 0$. Potom šírka nejakej cesty je definovaná ako minimálna šírka hrany tejto cesty. Podobne ako pri najkratších cestách, aj tu môžeme rozlišovať úlohy nájsť najširšiu cestu z jedného vrcholu do druhého, z jedného do ostatných a z každého do každého vrchola. Aj tu sú tieto varianty vzájomne veľmi blízke a tieto rozdiely nie sú podstatné.

Jedna myšlienka ako obísť úlohu o najširšej ceste je zistiť šírku nákladu, resp. minimálnu potrebnú šírku hrany a všetky užšie hrany z digrafu vynechať. Potom na zvyšný digraf stačí aplikovať napr. Moorov algoritmus.

Cvičenie 3.20. Uveďte zložitosť takéhoto postupu.

Iná idea spočíva v usporiadaní hrán podľa šírky do monotónnej postupnosti a potom vynechať úzke hrany.

Cvičenie 3.21. Rozpracujte detaily a zistite výpočtovú zložitosť celého algoritmu.

Tiež možno vypracovať analogický algoritmus ako je Dijkstrov algoritmus.

Cvičenie 3.22. Urobte to.

NAJSPOĽAHLIVEJŠIA CESTA

V tomto probléme má každý šíp (i, j) tzv. spoľahlivosť (pravdepodobnosť správnej činnosti) $0 < q_{ij} \leq 1$. Pod spoľahlivosťou nejakej cesty rozumieme súčin spoľahlivostí všetkých jej hrán. Cieľom je nájsť cestu s maximálnou spoľahlivosťou.

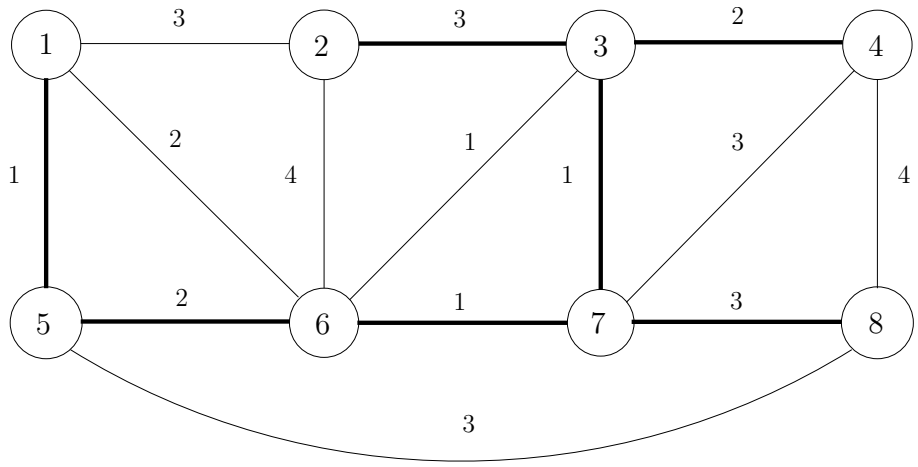
Cvičenie 3.23. Ukážte, že túto úlohu možno previesť na problém najkratšej cesty.

Kapitola 4

NAJLACNEJŠIE STROMY

4.1 Najlacnejšia kostra grafu

Kostra grafu je taký podgraf na všetkých vrcholoch, ktorý je stromom (čo je súvislý podgraf bez cyklov). Obrazne povedané, pokiaľ v grafe zachováme nejakú kostru, tak graf bude súvislý a súčasne táto súvislosť je zabezpečená ekonomicky (minimálnym počtom hrán). V praxi sa často stretávame s grafmi kde každá hrana má nejakú cenu a potom chceme súvislosť zabezpečiť za najmenšiu sumárnu cenu, t.j. nájsť taký súvislý podgraf na všetkých vrcholoch, ktorý má minimálnu sumárnu cenu, teda je najlacnejší. Ak sú všetky ceny grafu nezáporné, tak za takýto podgraf stačí zobrať najlacnejšiu kostru.

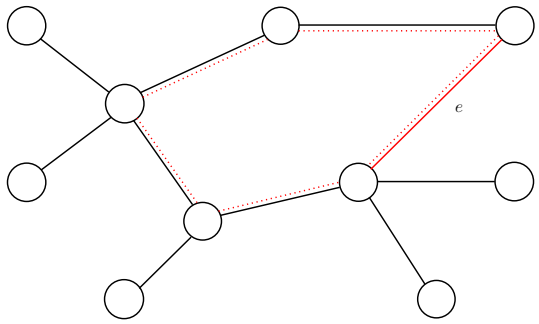


Obr. 4.1: Graf s vyznačenou najlacnejšou kostrou

Ako príklad si predstavme nejakú rozvojovú krajinu kde je naplánovaná výstavba cestnej siete spájajúcej jednotlivé obce, pričom je známa cena každého

úseku ako na obr. 4.1. Vzhľadom na nedostatok financií je rozumné najprv vybudovať iba podsieť zabezpečujúcu spojenie medzi ľubovoľnými dvoma obcami. Samozrejme, takáto podsieť by mala byť najlacnejšia. Na obr. 4.1 je jedna taká vyznačená hrubými čiarami; jej cena je 13. Zdá sa, že O. Borůvka (Brno) bol prvý, kto sa hľadaním najlacnejšej kostry grafu zaoberal. Bolo to v r. 1926 a motiváciou bola elektrifikácia južnej Moravy.

Nasledujúca lema ukazuje ako možno z nejakej kostry utvoriť novú kostru výmenou jednej hrany.



Obr. 4.2: Pridanie hrany e (červená) ku kostre vytvorí cyklus (bodkovaný)

Lema 4.1. *Nech K je kostra grafu G a e je hrana grafu G nepatriaca do K. Potom graf $K+e$ bude obsahovať práve jeden cyklus (ako podgraf) a po vynechaní ľubovoľnej jednej hrany tohoto cyklu dostaneme kostru grafu G.*

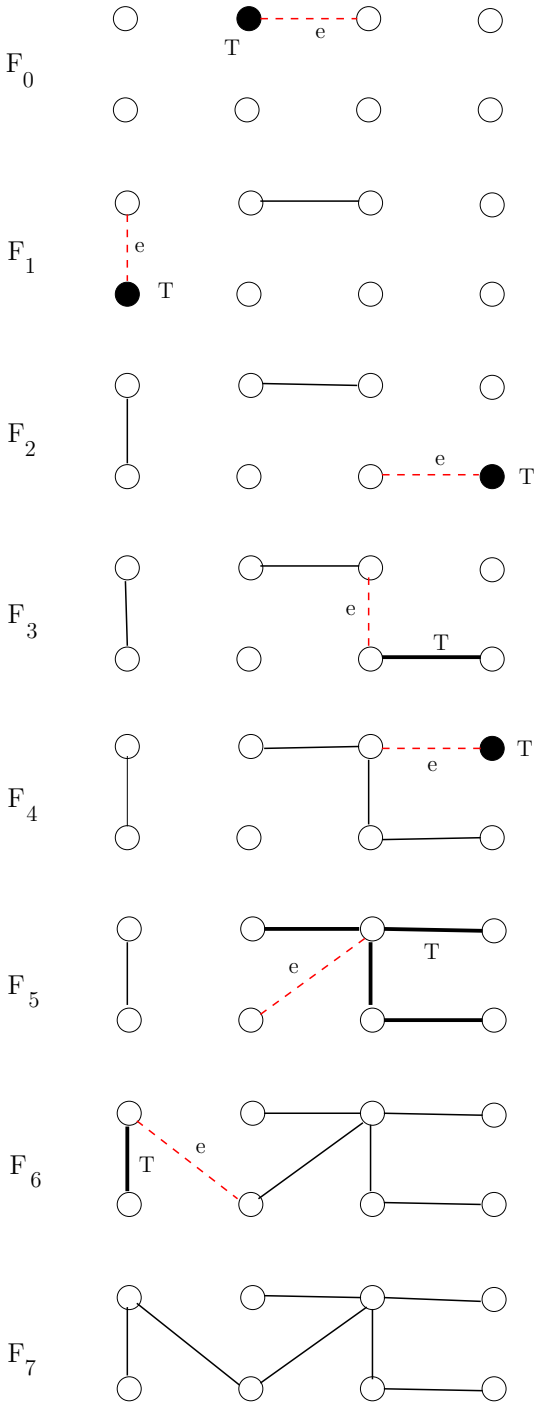
Dôkaz: Takáto situácia je zobrazená na obr. 4.2 kde kostra K je vyznačená čiernymi čiarami, hrana e je červená a cyklus je vyznačený bodkovanou červenou čiarou. Keďže konce hrany e sú v kostre spojené práve jednou cestou, tak po jej pridaní vznikne práve jeden cyklus (ako podgraf). Po vynechaní ľubovoľnej jednej hrany tohoto cyklu sa cyklus poruší, ale graf zostane súvislý, a preto bude strom a tým kostra. ■

Nasledujúci algoritmus, ktorého autorom je Tarjan, vychádza z myšlienok niektorých starších algoritmov. Tie sa potom ukážu ako špeciálne prípady.

SPÁJACÍ ALGORITMUS

Začínajúc z lesa F_0, podgrafu pozostávajúceho zo všetkých vrcholov a žiadnej hrany, postupne budujeme lesy $F_1, F_2, ..., F_{n-1}$ nasledovne: V lese F_i zvolíme ľubovoľný strom T a nájdeme jednu najlacnejšiu hranu e grafu G spájajúcu T s iným stromom lesa F_i. Túto hranu pridáme k lesu F_i a tým dostaneme les F_{i+1}. (Les F_i má presne i hrán, a teda F_{n-1} je strom (kostra) - výstup algoritmu.

Aplikujme algoritmus na úlohu z obr. 4.1. Postup je zobrazený na obr. 4.3. Výsledná kostra F_7 má cenu 13; je však iná ako tá na obr. 4.1.

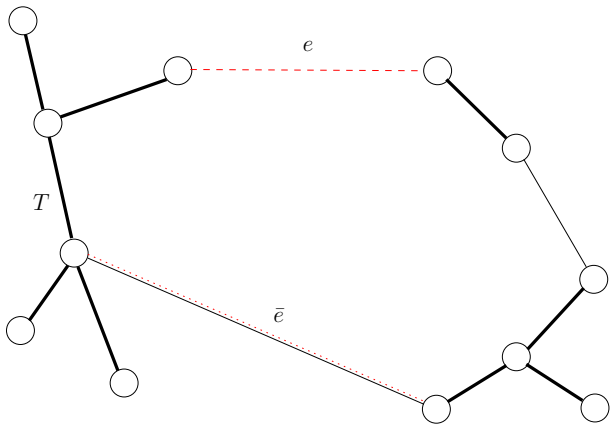


Obr. 4.3: Postup hľadania najlacnejšej kostry grafu z obr. 4.1

Správnosť algoritmu vyplýva z nasledovného tvrdenia.

VETA 4.2. *Pre každé $i = 0, 1, ..., n-1$ les F_i generovaný spájacim algoritmom leží v nejakej najlacnejšej kostre grafu G.*

Dôkaz: Dokazujeme indukciou podľa i. Pre $i = 0$ je tvrdenie zrejmé. Z predpokladu existencie najlacnejšej kostry S_i obsahujúcej F_i dokážeme, že existuje najlacnejšia kostra S_{i+1} obsahujúca F_{i+1}. Situáciu znázorňuje obr. 4.4, kde hrubé čiary predstavujú les F_i, kostra S_i je zobrazená plnými čiernymi čiarami (hrubými i tenkými) a najlacnejšia hrana e spájajúca strom T s iným stromom je červená čiarkovaná. Podľa algoritmu je $F_{i+1} = F_i + e$. Podľa lemy 4.1 po pridaní hrany e ku kostre S_i vznikne jediný cyklus. Tento cyklus obsahuje časť stromu T a existuje hrana tohoto cyklu $\bar{e} \neq e$ (na obr. 4.4 vyznačená bodkovanou čiarou), ktorá tiež spája strom T s iným stromom lesa F_i. Podľa voľby hrany e platí $c(e) \leq c(\bar{e})$. Preto nová kostra $S_{i+1} = S_i + e - \bar{e}$ má cenu $c(S_{i+1} = c(S_i) + c(e) - c(\bar{e} \leq c(S_i)$. Keďže lacnejšia kostra ako S_i nemôže existovať, tak tam platí rovnosť, a teda S_{i+1} je tiež najlacnejšia kostra a obsahuje les F_{i+1}, čo sme chceli dokázať. ■



Obr. 4.4: Ilustrácia k dôkazu vety 4.2

Zložitosť algoritmu. Máme $O(n)$ iterácií. V každej iterácii na nájdenie hrany e a vytvorenie nového lesa vystačíme s $O(m)$ operáciami. Teda celkove máme algoritmus zložitosti $O(mn) \leq O(n^3)$. Pre špeciálne verzie však možno túto zložitosť znížiť.

PRIMOV ALGORITMUS

(Vo svetovej literatúre je známy pod týmto menom, ale jeho autorom je Jarník a Prim ho znova objavil.) V každej iterácii volíme za strom T najväčší strom. (Na začiatku sú všetky stromy jednovrcholové, a teda T je ľubovoľný vrchol. Avšak v ďalších iteráciach máme jediný viacvrcholový strom a ostatné sú jednovrcholové, a preto je strom T jednoznačne určený.)

V takejto hrubej verzii je zložitosť Primovho algoritmu $O(mn) \leq O(n^3)$, ale možno ju zmenšiť. Predpokladajme, že v každej iterácii máme pri každom vrchole $j \in V(G) \setminus V(T)$ informáciu o najlacnejšej hrane spájajúcej strom T a vrchol j (teda jej cenu a druhý koniec ležiaci v T).

Cvičenie 4.1. Ukážte,že túto informáciu možno ľahko vyrobiť a tým získať celkovú zložitosť $O(n^2)$.

Pre riedke grafy možno ešte ušetriť. Napr. použitím binárnej haldy, v ktorej udržiavame vrcholy nepatriace do T), možno získať zložitosť $O(m+n)\log n = O(m \log n)$.

Cvičenie 4.2. Ukážte to.

Dômyselnejšie postupy to posunuli ešte ďalej ($O(m + n \log n)$).

KRUSKALOV ALGORITMUS

V tomto algoritme najprv usporiadame hrany podľa ceny do neklesajucej postupnosti:

$$c(e_1) \leq c(e_2) \leq ... \leq c(e_m).$$

Začínajúc s lesom F_0 bez hrán, postupne z uvedenej postupnosti skúšame pridať najlacnejšiu nepreskúmanú hranu (u, v): Ak u a v neležia v tom istom strome, tak ju pridáme, inak ju preskočíme.

Cvičenie 4.3. Ukážte, že tento postup tiež spadá pod spájací algoritmus

.

Úvodné triedenie (usporiadanie) možno urobiť v čase $O(m \log n)$. Keďže testovanie hrany a spájanie stromov sa dá zrealizovať v čase $O(m + n \log n)$, tak celý algoritmus má zložitosť $O(m \log n)$. Poznamenajme, že Kruskalov algoritmus zaraďujeme medzi tzv. **pažravé algoritmy**. Takýto algoritmus vyberá do riešenia vždy prvok, ktorý je momentálne najvýhodnejší.

Alternatívny postup navrhol **Borůvka**. Tu úvodné usporiadanie nie je, ale pre každý strom lesa nájdeme jednu najlacnejšiu hranu spájajúcu tento strom s iným stromom lesa a všetky takéto hrany pridáme k lesu naraz. Tým sa zredukuje počet stromov aspoň na polovicu, a teda počet iterácií neprekročí $O(\log_2 n)$. Aby sme sa vyhli utvoreniu cyklu, tak predpokladáme, že všetky ceny hrán sú vzájomne rôzne. To môžeme v praxi ľahko dodržať a v teórii to zariadime perturbáciou cien. Aj tento algoritmus možno implementovať na zložitosť $O(m \log n)$. Algoritmus sa ukázal výhodný pri paralelnom počítaní.

4.2 Najlacnejšia zdrojová kostra digrafu

Podobne ako sme pre grafy s cenami hrán postavili úlohu nájsť najlacnejší súvislý podgraf na všetkých vrcholoch, aj pre digrafy vzniká podobná úloha. Pravda, otázkou je, čo žiadať namiesto súvislosti. Slabá súvislosť nám nedáva nič nové. Silná súvislosť dáva pekný problém, ktorý je však NP-ťažký dokonca

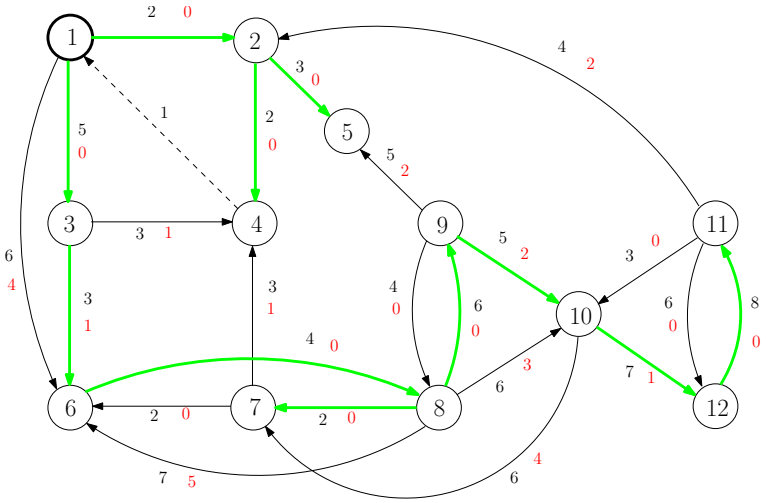
pre neohodnotené digrafy. Takým stredným problémom je úloha nájsť najlacnejšiu zdrojovú kostru digrafu: Daný je digraf G s reálnymi cenami hrán c_{ij}, v ktorom je predpísaný vrchol r . Treba nájsť najlacnejšiu kostru K digrafu G (podstrom na všetkých vrcholoch), v ktorej je vrchol r zdrojom (koreňom) (t.j. každý vrchol je z neho dosiahnuteľný). Samozrejme, taká kostra existuje práve vtedy, keď r je zdrojom v G.

Cvičenie 4.4. Nahliadnite, že tento problém je zovšeobecnením problému najlacejšej kostry.

Uvedený problém sa nazýva sa úlohou **nájsť najlacnejšiu zdrojovú kostru digrafu s predpísaným zdrojom** na rozdiel od príbuznej úlohy **nájsť najlacnejšiu zdrojovú kostru digrafu s voľným zdrojom** (kde zdroj nie je predpísaný). Tieto úlohy sú ekvivalentné v tom zmysle, že jednu možno previesť na druhú.

Cvičenie 4.5. Urobte to.

V ďalšom sa obmedzíme na prípad s predpísaným zdrojom. Ako aplikáciu možno uviesť (Edmonds) problém najlacnejšieho rozšírenia informácie (rozkazu) od generála ku každému členovi veliteľstva armády, kde je podávanie informácií individuálne a predstavuje isté náklady (z digrafu teoretických možností treba vybrať ekonomický „informačný strom“). Na obr. 4.5 je príklad kde zdroj je vyznačený hrubým krúžkom (vrchol 1) a ceny šípov sú čiernou farbou. Zelené šípy tvoria najlacnejšiu zdrojovú kostru; jej cena je 47.



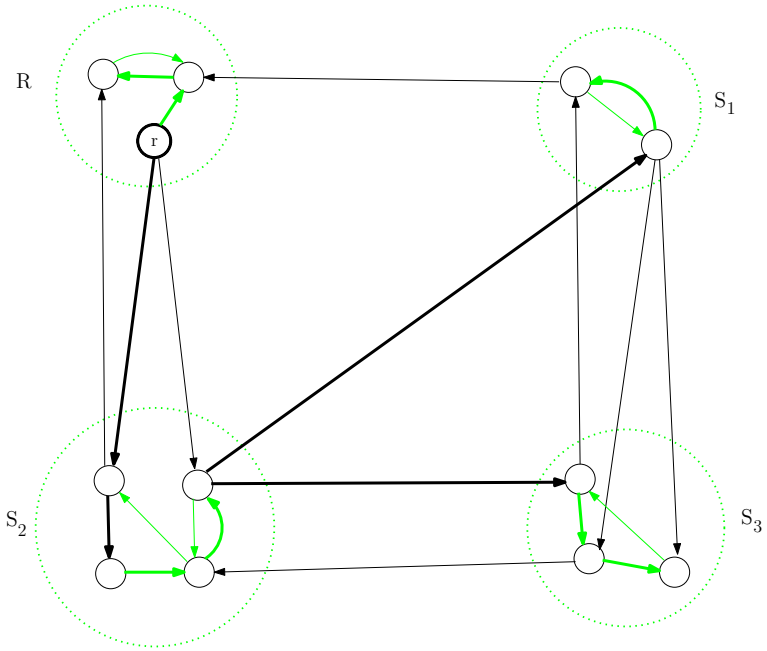
Obr. 4.5: Príklad digrafu pre najlacnejšiu zdrojovú kostru

Týmto problémom sa ako prví zaoberali čínski matematici Chu a Liu (1965) a nezávisle Edmonds. Pri vyvíjaní algoritmu využijeme isté zjednodušenia. Predovšetkým môžeme predpokladať, že všetky ceny $c_{ij} \geq 0$. To vyplýva z toho, že

ak ku každej cene hrany digrafu G pripočítame to isté číslo, tak sa optimálna kostra nezmení.

Cvičenie 4.6. Dokážte.

Ďalej môžeme predpokladať, že vrchol r má v G iba odchádzajúce šípy. Jednoducho všetky prichádzajúce šípy vynecháme, lebo žiaden z nich nemôže patriť do zdrojovej kostry so zdrojom r. Základným pozorovaním je nasledovná lema ilustrovaná na obr. 4.6.



Obr. 4.6: Ilustrácia k leme 4.3

Lema 4.3. *Nech G je digraf a K je jeho kostra so zdrojom r (čierne hrany na obr. 4.6). Nech vrcholova množina digrafu G je rozložená do indukovaných poddigrafov R, S_1,..., S_p (bodkované zelené obálky) tak, že v R sú všetky vrcholy dosiahnuteľné z r a ostatné poddigrafy S_i sú silne súvislé. Potom vynechaním niektorých hrán z K a pridaním niektorých hrán z poddigrafov R a S_i možno z K vyrobiť kostru K' so zdrojom r, ktorá v každom digrafe R, S_1,..., S_p dáva nejakú zdrojovú kostru (t.j. $K' \cap R$ a $K' \cap S_i$ sú zdrojové kostry).*

Dôkaz: Keďže v R je každý vrchol dosiahnuteľný z r, tak napr. postupom do šírky možno v R ľahko nájsť kostru K_0 so zdrojom r. Tým máme pokryté všetky vrcholy v R, a preto vynecháme všetky hrany kostry K prichádzajúce zvonka do R a všetky ostatné hrany vo vnútri. Pre každý digraf S_i ľubovoľne ponecháme práve jeden šíp kostry K prichádzajúci do S_i z R; nech prichádza do

vrchola v_i. Potom v S_i nájdeme kostru K_i so zdrojom v_i (ako pre R) a všetky ostatné šípy v S_i vynecháme. Tým máme pokryté aj digrafy S_i do vzdialenosti 1 od R. Analogicky vyšetríme digrafy so vzdialenosťou 2 od R. To sú tie, do ktorých vedie aspoň jeden šíp kostry K z prvej vrstvy (digrafy so vzdialenosťou 1 od R). Teda ponecháme z takých šípov práve jeden, v digrafe utvoríme vhodnu kostru a ostatné šípy digrafu S_i vynecháme. Po vyšetrení všetkých vrstiev získame požadovanú kostru K'. ■

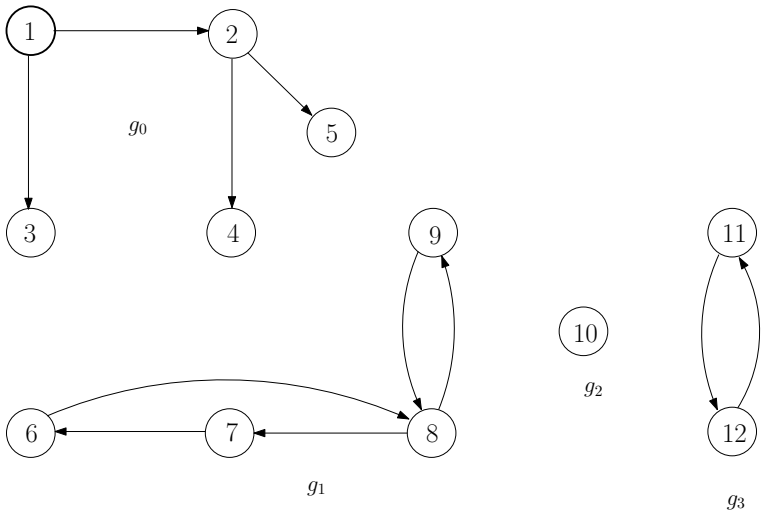
Po tejto príprave postupujeme nasledovne.

(1) Pre každý vrchol $i \neq r$ nájdeme najmenšiu prichádzajúcu cenu c_i^{min} a všetky prichádzajúce ceny o ňu zmenšíme, t.j. definujeme nové ceny $c'_{ji} = c_{ji} - c_i^{min}$. (Pozri červené ceny na obr. 4.5.) Optimálna kostra sa pri nových cenách nezmení.

Cvičenie 4.7. Zdôvodnite to na základe faktu, že v každej zdrojovej kostre so zdrojom r prichádza do i práve jeden šíp.

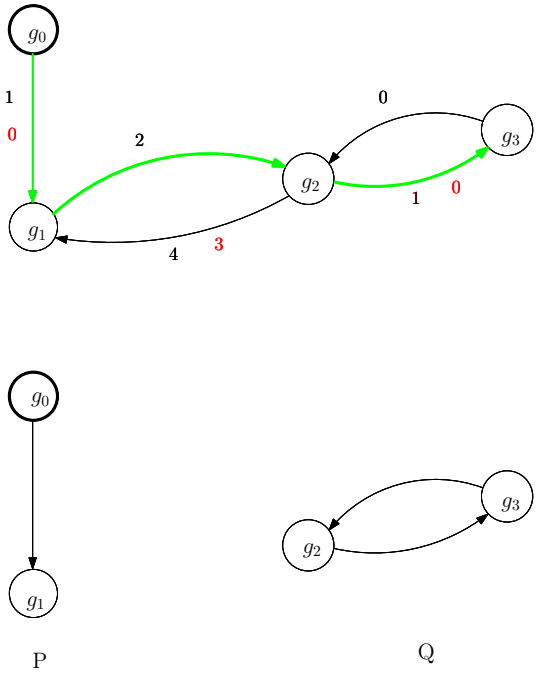
Takto v nových cenách budú všetky šípy znova nezáporné a pri každom vrchole $i \neq r$ získame aspoň jeden prichádzajúci šíp nulovej ceny. Nech Z je množina všetkých šípov nulovej ceny.

(2) V poddigrafe $F = (V, Z)$ nájdeme maximálnu (v zmysle inklúzie) podmnožinu vrcholov R dosiahnuteľných z vrcholu r (teda po nulových šípoch). Nech $T_r = (R, S)$ je príslušný strom zo zdrojom r ukazujúci dosiahnuteľnosť (nájdeme ho napr. postupom do šírky). (Pre digraf z obr. 4.5 pozri strom g_0 z obr. 4.7.) Sú dve možnosti:



Obr. 4.7: Zdrojový strom a silné komponenty pre príklad z obr. 4.5

(a) $R = V$. Vtedy T_r je hľadanou kostrou a skončíme.

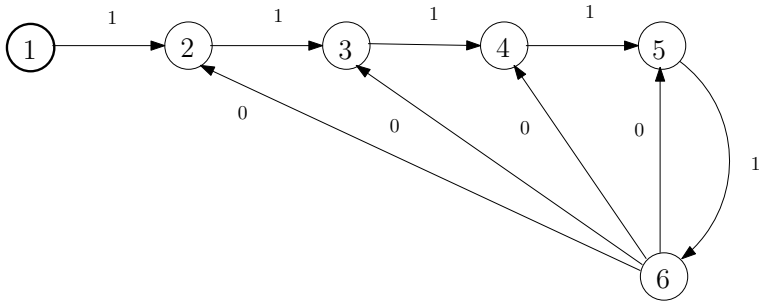


Obr. 4.8: Digraf získaný kontrakciou z digrafu na obr. 4.5 a príslušné komponenty

Obr. 4.9: Finálny digraf získaný z digrafu na obr. 4.8

(b) $R \neq V$. Nájdeme všetky silné komponenty v F disjunktné s R. K tomu stačí nájsť silné komponenty digrafu indukovanému v F množinou $V \setminus R$, lebo každý silný komponent digrafu F buď má všetky vrcholy v R, alebo je s R disjunktný. (Pre digraf z obr. 4.5 sú príslušné silné komponenty g_1, g_2 a g_3 na obr. 4.7. Keďže predpokladáme, že v pôvodnom digrafe G je vrchol r zdrojom (t.j. každý vrchol je z vrchola r dosiahnuteľný), tak aspoň jeden z týchto silných komponentov bude netriviálny (bude mať aspoň dva vrcholy). Každý takýto silný komponent a tiež množinu R stiahneme v G na jeden vrchol, utvorené slučky vynecháme a z paralelných šípov ponecháme vždy iba najlacnejší. Tým z G získame nový menší digraf $\hat{G}$ s nezápornými šípmi a s predpísaným zdrojom $\hat{r}$, ktorý vznikol kontrakciou množiny R. Po vynechaní všetkých šípov prichádzajúcich do $\hat{r}$ ideme s týmto digrafom na krok (1) ako by to bol G.

Ak nájdeme najlacnejšiu zdrojovú kostru $\hat{K}$ pre $(\hat{G}, \hat{r})$, tak ľahko sa z nej získa zdrojová kostra K pre (G, r) tej istej ceny.



Obr. 4.10: V každej iterácii sa tu digraf zmenší iba o jeden vrchol

Cvičenie 4.8. Ukážte ako.

Kostra K musí byť najlacnejšia, lebo inak by podľa lemy 4.3 lacnejšia kostra pre (G, r) produkovala aj lacnejšiu kostru ako $\hat{K}$ pre $(\hat{G}, \hat{r})$, čo by bol spor. Uvedený algoritmus je ilustrovaný na obr. 4.5 až 4.9. Ceny hrán sú vyznačené čiernou farbou a po redukcii (odčítaní najmenšej prichádzajúcej ceny do vrcholu) ich vyznačujeme červenou farbou. Zdrojová kostra získaná v poslednej iterácii je na obr. 4.9 vyznačená zelenou farbou; z nej získame zdrojovú kostru pre digraf z predošlej iterácie (obr. 4.8) a z tej výslednú kostru pre pôvodný digraf na obr. 4.5.

Keďže $\hat{G}$ má aspoň o jeden vrchol menej ako G, tak celkove v algoritme vystačíme s $O(n)$ iteráciami. Na každú z nich stačí $O(m+n) = O(m)$ operácií. Teda sumárne má predložený algoritmus zložitosť $O(mn)$. Poznamenajme, že odhad počtu iterácií nemožno vo všeobecnosti znížiť ako ukazuje príklad z obr. 4.10, ale podarilo sa vyvinúť aj efektívnejšie implementácie.

4.3 Steinerov problém v grafoch

Nasledujúci problém je zovšeobecnením problému najlacnejšej kostry. Daný je súvislý graf G s reálnymi ohodnoteniami hrán $c_{ij} > 0$ a množina $P \subset V(G)$ tzv. povinných vrcholov. Ostatné vrcholy sú tzv. voliteľné vrcholy. Podgraf T grafu G, ktorý je stromom a obsahuje všetky povinné vrcholy, sa nazýva steinerovský strom (pre P) (môže obsahovať aj niektoré voliteľné vrcholy). Jeho cenou $c(T)$ je súčet cien všetkých jeho hrán. Pod Steinerovym problémom rozumieme úlohu nájsť najlacnejší steinerovský strom. Tento problém zahŕňa problém najlacnejšej kostry (ak $P = V(G)$) a tiež problém nájsť najkratšiu cestu v takom grafe (ak $|P| = 2$). Na obr. 4.11 sú povinné vrcholy čierne a najlacnejší steinerovský strom je vyznačený hrubými zelenými čiarami.

Steinerov problém patrí medzi NP-ťažké úlohy. Hoci nepoznáme polynomiálny algoritmus na jeho vyriešenie, existujú polynomiálne algoritmy dávajúce steinerovský strom, ktorého cena sa približne rovná cene najlacnejšieho steinerovského stromu. Uvedieme jeden taký algoritmus. Postup je ilustrovaný na obr.

4.11.

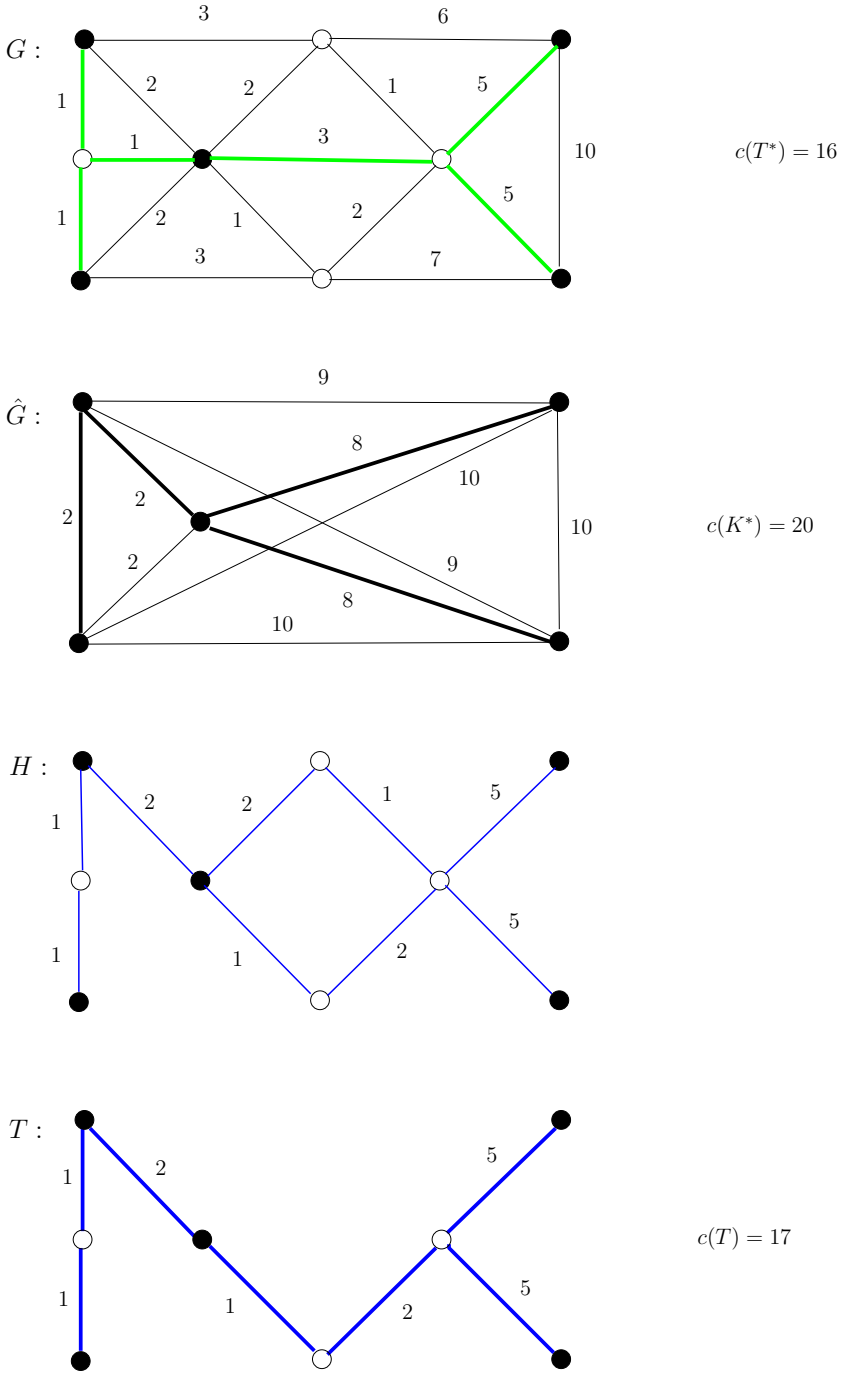
Najprv zostrojíme kompletný graf $\hat{G}$ na množine povinných vrcholov P, pričom jeho hrana (i, j) dostane cenu $\hat{c}_{ij}$, ktorá sa rovná dĺžke najkratšej i-j cesty v grafe G pri pôvodných cenách považovaných za dĺžky hrán. Potom nájdeme najlacnejšiu kostru K^* v $\hat{G}$. Nakoniec prenesieme túto kostru do pôvodného grafu G tak, že každú jej hranu nahradíme príslušnou jednou najkratšou cestou. Získaný graf H po očistení dáva nejaký steinerovský strom pre P. Očistenie možno urobiť vynechaním niektorých najdrahších hrán ležiacich v cykloch (napr. ponecháme iba hrany nejakej najlacnejšej kostry v H) a potom postupným vynechaním koncových voliteľných vrcholov (t.j. voliteľných vrcholov stupňa 1 v priebežnom grafe).

Tento algoritmus je zrejme polynomiálny.

Cvičenie 4.9. Odhadnite jeho zložitosť.

VETA 4.4. *Uvedený algoritmus je 2-aproximačný, t.j. dáva steinerovský strom T s cenou $c(T) \leq 2c(T^*)$, kde T^* je najlacnejší steinerovský strom.*

Dôkaz: Keďže v kostre K^* všetky povinné vrcholy súvisia, tak budú súvisieť aj v grafe H a tiež po očistení, t.j. vo výslednom strome T. Pre dôkaz kvality aproximácie sledujme obr. 4.12, kde máme v rovine zakreslený nejaký najlacnejší steinerovský strom T^*. Utvorme sled S okolo T^* a označme povinné



$c(T^*) = 16$

$c(K^*) = 20$

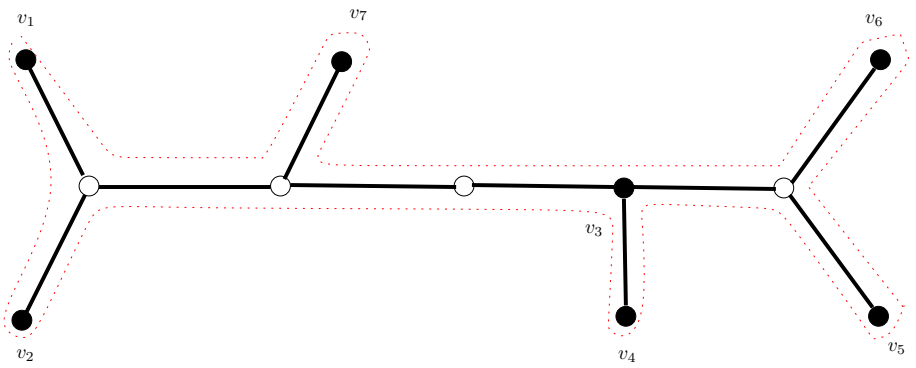
$c(T) = 17$

Obr. 4.11: Príklad Steinerovho problému a jeho riešenie aproximačným algoritmom

vrcholy takto: Začnime v ľuvoľnom povinnom vrchole a označme ho v_1. Potom sledujúc S prídeme do povinného vrchola, ktorý označíme v_2, potom do ďalšieho doteraz neoznačeného vrchola, ktorý dostane značku v_3, atď. Takto označíme všetky povinné vrcholy: $v_1, v_2, ..., v_p$, kde $p = |P|$ je počet povinných vrcholov a skončíme vo v_1. (Na obr. 4.12 je $p = 7$.) Takto sled S pozostáva z úsekov $S_{12}, S_{23}, ..., S_{p1}$. Keďže každú hranu v T^* prejdeme 2-krát, tak $c(S_{12}) + c(S_{23}) + \cdots + c(S_{p1}) = c(S) = 2c(T^*)$. Pre dĺžku $d_{i,i+1}$ najkratšej v_i-v_{i+1} cesty v G zrejme platí $d_{i,i+1} \leq c(S_{i,i+1})$, lebo v G môže byť aj iná cesta ako $S_{i,i+1}$. Ak K je kostra grafu $\hat{G}$ pozostávajúca z $p-1$ hrán $(v_1, v_2), (v_2, v_3), ..., (v_{p-1}, v_p)$, tak vidíme, že

$$\begin{aligned} c(T) \leq c(K^*) \leq c(K) &= d_{12} + d_{23} + \cdots + d_{p-1,p} \\ &\leq c(S_{12}) + c(S_{23}) + \cdots + c(S_{p-1,p}) < c(S) = 2c(T^*). \end{aligned}$$

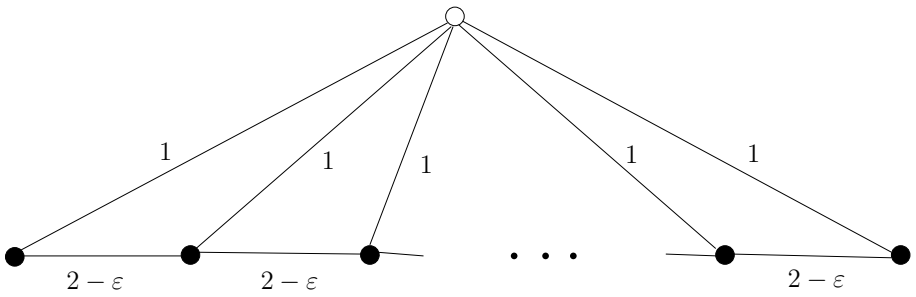
■



Obr. 4.12: Ilustrácia k dôkazu aproximačného algoritmu: sled S okolo T^*

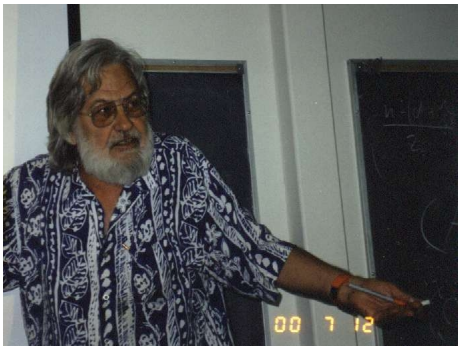
Cvičenie 4.10. Ukážte, že na obr. 4.11 možno vybrať K^* a H tak, že dostaneme strom T s cenou 20, ale aj tak, že získame strom T s cenou 16. Vo všeobecnosti však aproximačný pomer 2 vo vete 4.4 nemožno pre tento algoritmus nahradiť menším ako ukazuje obr. 4.13, kde $\varepsilon > 0$ je dostatočne malé

. Existujú však zložitejšie aproximačné algoritmy s lepšou aproximáciou.



Obr. 4.13: Zlý príklad pre aproximačný algoritmus

Obr. 4.14: Robert Endre Tarjan 1948-

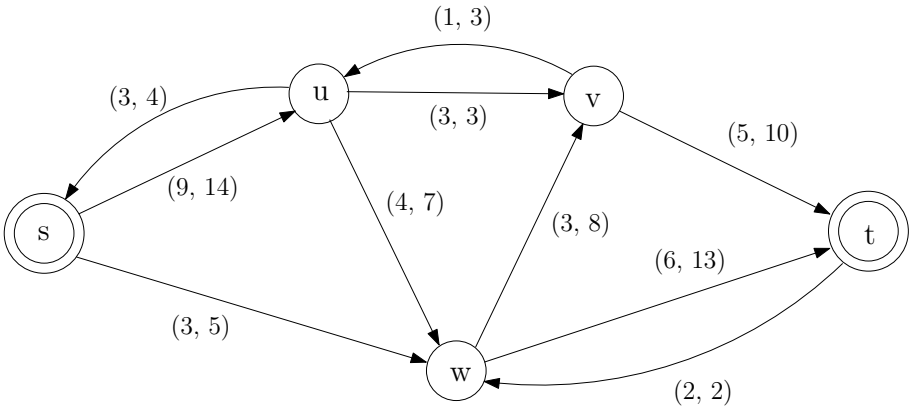


Obr. 4.15: Jack Edmonds 1934-

Kapitola 5

TOKY

Tu sa budeme zaoberať abstrakciou známych reálnych tokov vo vodovodných, cestných, elektrických a pod. sieťach. Pritom budeme predpokladať, že tok je tam ustálený, odteká z daného zdroja s a priteká do daného stoku t. Tiež na každom úseku bude daný smer toku a kapacita (priepustnosť, t.j. maximálny prúd, ktorý tam môže tiecť). Naviac budeme predpokladať, že prepravovaná hmota sa počas transportu nestráca (nikto ju nekradne) ani nenarastá. Toto budeme modelovať na digrafe s dvoma špeciálnymi vrcholmi s a t, pričom každý šíp (i, j) bude mať predpísanú kapacitu $b_{ij} \geq 0$. Na obr. 5.1 je príklad toku. V praxi nás zaujíma tzv. veľkosť toku, t.j. čistý odtok z s: sumárna odtekajúca hodnota z s mínus sumárna pritekajúca hodnota do s (ekvivalentne: sumárna pritekajúca hodnota do t mínus sumárna odtekajúca hodnota z t). Na obr. 5.1 máme tok veľkosti 9.



Obr. 5.1: Príklad s-t toku: na každom šípe (i, j) je dvojica (x_{ij}, b_{ij}) kde x_{ij} je hodnota toku a b_{ij} je kapacita

Formálne máme daný digraf $G = (V, E)$, špeciálne vrcholy $s, t \in V$ a ka-

pacitnú funkciu $b : E \mapsto R_+$. Potom tokom, presnejšie s-t tokom, nazývame funkciu $x : E \mapsto R_+$ spĺňajúcu podmienky:

$$0 \leq x_{ij} \leq b_{ij} \quad \forall (i,j) \in E \quad \text{(kapacitné obmedzenia)} \tag{5.1}$$

$$\sum_{i:(i,k)\in E} x_{ik} = \sum_{j:(k,j)\in E} x_{kj} \quad \forall k \in V \setminus \{s,t\} \quad \text{(podmienky kontinuity)} \tag{5.2}$$

Poznamenajme, že b a x sú vektory z $R^{|E|}$. Veľkosť s-t toku x je

$$f(x) = \sum_{j:(s,j)\in E} x_{sj} - \sum_{i:(i,s)\in E} x_{is} \tag{5.3}$$

Veľkosť s-t toku môže byť aj záporná. Napr. v digrafe na obr. 5.1 môžeme po ceste (t, w, v, u, s) pustiť 2 jednotky a na ostatných šípoch dať nulový tok. Tento s-t tok bude mať veľkosť -2. Pravda, vždy vieme zabezpečiť tok nulovej veľkosti. Jednoducho položíme $x = 0$ a obmedzenia (5.1) aj (5.2) budú splnené. Tok s nulovou veľkosťou sa nazýva **cirkulácia** (nemusí byť nutne nulový, ale koľko pritečie do t toľko aj odtečie). V praxi nás však zaujíma čo najväčší tok. To vedie k úlohe o maximálnom s-t toku:

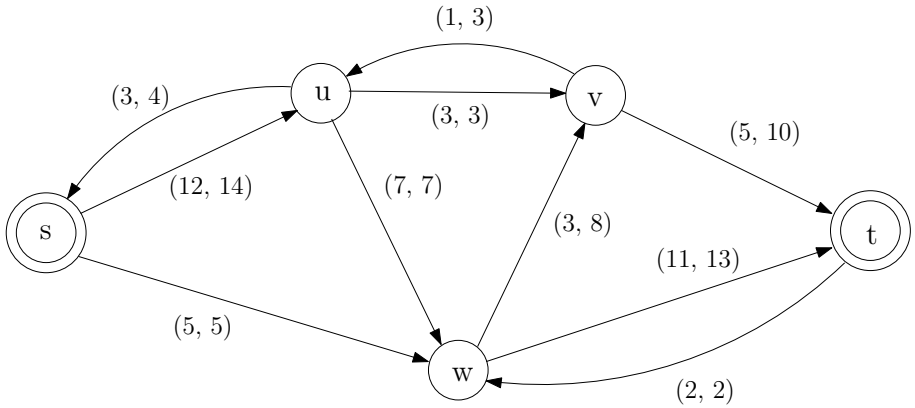
$$\text{maximalizovať} \quad f(x) \quad \text{za podmienok (5.1) a (5.2).} \tag{5.4}$$

Tok s maximálnou veľkosťou sa nazýva maximálny tok. Vidíme, že úloha o maximálnom toku (5.4) je úlohou lineárneho programovania (LP), príslušná množina prípustných riešení je neprázdna (existuje nulový tok) a ohraničená. Preto má vždy optimálne riešenie. Teda maximálny s-t tok vždy existuje a vyriešením uvedenej úlohy LP ho vieme nájsť. Pravda, táto úloha je veľmi špeciálna, a preto možno očakávať aj existenciu špeciálnej metódy, ktorá je jednoduchšia ako všeobecné metódy LP. S takou metódou sa oboznámime.

5.1 Algoritmus Forda a Fulkersona

Snáď prvá myšlienka pre hľadanie maximálneho toku, ktorá každého napadne, je nájsť ľubovoľný s-t tok (napr. nulový) a ten postupne zväčšovať. Pre zväčšenie toku nájdeme tzv. zväčšujúcu s-t cestu. Na takejto ceste má každá hrana kladnú rezervu, t.j. tok je menší ako kapacita. Napr. na obr. 5.1 je cesta (s, w, t) zväčšujúca a existujúci tok možno po nej zväčšiť až o 2 jednotky tak, že na každej jej hrane zväčšíme tok o 2. Ľahko overíme, že získame tok (podmienky (5.1) a (5.2) budú splnené) a jeho veľkosť bude podľa (5.3) o 2 jednotky väčšia, teda 11. Tento tok môžeme ďalej zväčšiť po ceste (s, u, w, t) o 3 jednotky. Výsledný tok veľkosti 14 je na obr. 5.2. Tento tok je už tzv. *blokujúci*, t.j. neexistuje preň zväčšujúca s-t cesta, ako sa ľahko presvedčíme. Pritom, ako uvidíme neskôr, nie je maximálny.

Cvičenie 5.1. Nájdite taký príklad kde veľkosť maximálneho toku je 10 a veľkosť blokujúceho toku je 1.



Obr. 5.2: Blokujúci tok získaný z toku na obr. 5.1 po zväčšujúcich cestách

Ford a Fulkerson navrhli silnejší prostriedok na zväčšovanie toku a totiž **zväčšujúcu** s-t **polocestu**, ktorá je zovšeobecnením zväčšujúcej s-t cesty. Tu žiadame, aby na súhlasných šípoch bol tok menší ako kapacita a na nesúhlasných väčší ako 0. Rezerva šípu polocesty sa definuje ako rozdiel kapacity a toku, ak je šíp súhlasny, a ako hodnota toku, ak je šíp nesúhlasný. Ak δ je minimálna rezerva na zväčšujúcej poloceste, tak po tejto poloceste možno tok zväčšiť až o δ, ako ukazuje obr. 5.3. Po zmene toku sa dodržia kapacitné obmedzenia (to vyplýva z voľby čísla δ) i podmienky kontinuity (to ľahko overíme pre vrcholy polocesty, lebo je len niekoľko možných situácií ako na obr. 5.3). Ešte ľahšie nahliadneme, že veľkosť nového toku vzrastie o δ. Pre tok z obr. 5.2 existuje zväčšujúca s-t polocesta: $(s, (s,u), u, (v,u), v, (v,t), t)$. Minimálna rezerva je 1 (realizuje sa na nesúhlasnej hrane (v,u)). Ak zmeníme tok po tejto poloceste o $\delta = 1$, tak získame tok uvedený na obr. 5.4, ktorý je už maximálny, ako uvidíme.

K otestovaniu optimality toku možno použiť tzv. s-t **rez**, čo je usporiadaná dvojica (S,T) podmnožín tvoriaca rozklad množiny vrcholov V taká, že $s \in S$ a $t \in T$ (porovnaj obr. 5.4). *Kapacitu rezu* definujeme takto:

$$b(S,T) = \sum_{i \in S, j \in T} b_{ij}. \tag{5.5}$$

Analogicky k (5.5) budeme stručne písať $x(S,T)$ a pod. Nasledujúce tvrdenie pripomína slabú vetu o dualite. V skutočnosti ho možno odtiaľ odvodiť, ale priamy prístup je jednoduchší.

Lema 5.1. *Veľkosť ľubovoľného s-t toku x nepresahuje kapacitu ľubovoľného s-t rezu (S,T), t.j.*

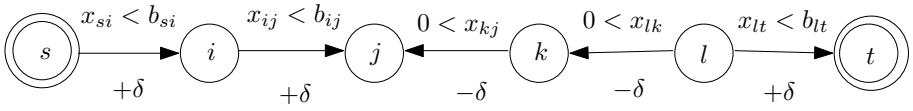
$$f(x) \leq b(S,T).$$

Dôkaz: Stačí si uvedomiť, že čistý odtok z s sa rovná čistému odtoku z S. Potom podľa kapacitných obmedzení a definície kapacity rezu máme $f(x) = x(S,T) - x(T,S) \leq b(S,T)$.

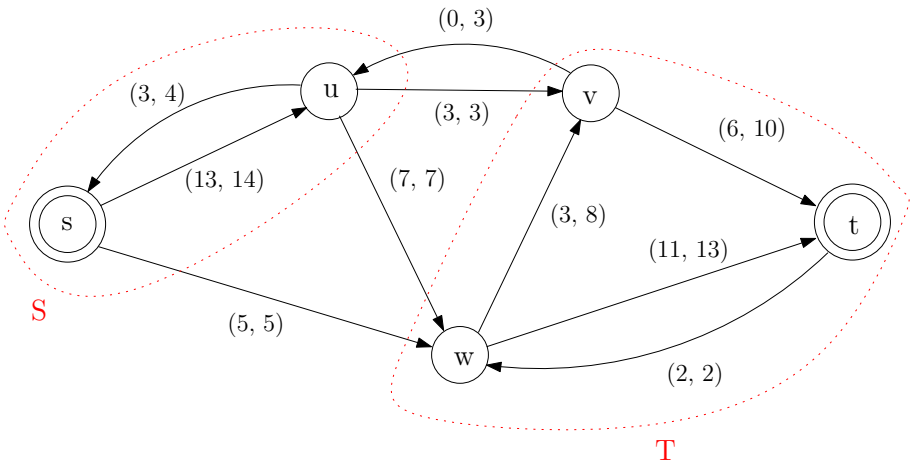
■

VETA 5.2. *Nasledujúce tvrdenia sú ekvivalentné:*
(i) s-t tok x je maximálny,
(ii) neexistuje zväčšujúca s-t polocesta pre x,
(iii) existuje taký s-t rez (S,T), že $f(x) = b(S,T)$.

Dôkaz: (i)⇒ (ii) Zrejmé.
(ii)⇒ (iii) Definujme $S = \{i \mid$ existuje zväčšujúca s-i polocesta pre $x\}$ a $T = V \setminus S$. Zrejme $s \in S$ a $t \in T$. Teda (S,T) je s-t rez a platí: (1) pre každý šíp (i,j), kde $i \in S, j \in T$, máme $x_{ij} = b_{ij}$ (inak zväčšujúcu s-i polocestu by sme mohli predĺžiť do j a bolo by $j \in S$, (2) pre každý šíp (k,i), kde $k \in T$ a $i \in S$, máme $x_{ki} = 0$ (inak zväčšujúcu s-i polocestu môžeme predĺžiť do k a bolo by $k \in S$). Preto $f(x) = x(S,T) - x(T,S) = b(S,T)$.
(iii)⇒ (i) Bezprostredne z lemy 5.1. ■



Obr. 5.3: Zmena toku po zväčšujúcej poloceste



Obr. 5.4: Maximálny s-t tok získaný z toku na obr. 5.2 po zväčšujúcej poloceste a minimálny s-t rez

Veta 5.2 je základom nasledujúceho algoritmu Forda a Fulkersona pre hľadanie maximálneho toku:

Začneme s nejakým (napr. nulovým) s-t tokom a ten postupne zväčšujeme po zväčšujúcich s-t polocestách, pričom na danej poloceste vždy zväčšíme tok

o maximálnu možnú hodnotu (t.j. minimálnu rezervu hrany). Ak pre priebežný tok žiadna zväčšujúca polocesta neexistuje, tak už je maximálny.

Problémom zostáva hľadanie zväčšujúcich polociest. Pre tento účel Ford a Fulkerson navrhli značkovaciu procedúru, kde pre daný tok x sú na začiatku všetky vrcholy neoznačkované a na konci budú niektoré označkované. Potom na základe tých značiek alebo nájdeme nejakú zväčšujúcu s-t polocestu, alebo usúdime, že žiadna neexistuje. Značka pri vrchole i má tvar (δ_i, p_i) kde δ_i je dodatočné množstvo toku, ktoré možno dopraviť popri existujúcom toku do vrcholu i po nejakej zväčšujúcej s-i poloceste a p_i je jej predposledný vrchol. (To je podobná informácia ako pri algoritmoch pre nakratšie cesty.) Postupujeme takto: Vrchol s dostane značku $(\infty, -)$ (druhá zložka je prázdna) a značky rozširujeme v súlade s obr. 5.5: Ak vrchol i má značku, tak nájdeme nejakú hranu (i,j), kde vrchol j nemá značku a $x_{ij} < b_{ij}$, potom vrcholu j priradíme značku $(\min\{\delta_i, b_{ij} - x_{ij}\}, i^+)$ (posledný šíp polocesty je súhlasný), alebo nájdeme nejakú hranu (k,i), kde vrchol k je neoznačkovaný a $x_{ki} > 0$, potom vrcholu k dáme značku $(\min\{\delta_i, x_{ki}\}, i^-)$ (posledný šíp polocesty je nesúhlasný). Snahou je označkovať vrchol t. Ak sa podarí označkovať t, tak proces značkovania skončíme a tok zväčšíme o δ_t po poloceste, ktorú postupne identifikujeme od konca podľa druhých zložiek značiek. V opačnom prípade máme označkovanú nejakú množinu vrcholov S a žiadny ďalší vrchol sa už nedá označkovať. Vtedy podľa lemy 5.1 dostávame minimálny s-t rez $(S,T$, kde $T = V \setminus S$, lebo každý šíp idúci z S do T je nasýtený a každý šíp idúci z T do S je nulový podľa pravidiel značkovania. Takže $f(x) = b(S,T)$ a podľa vety 5.2 je tok x maximálny.

Poznamenajme, že namiesto hľadania zväčšujúcej s-t polocesty možno hľadať s-t cestu v tzv. *rezervnom (multi)digrafe* $G'(x)$, ktorý zostrojíme k danému digrafu G a priebežnému toku x.

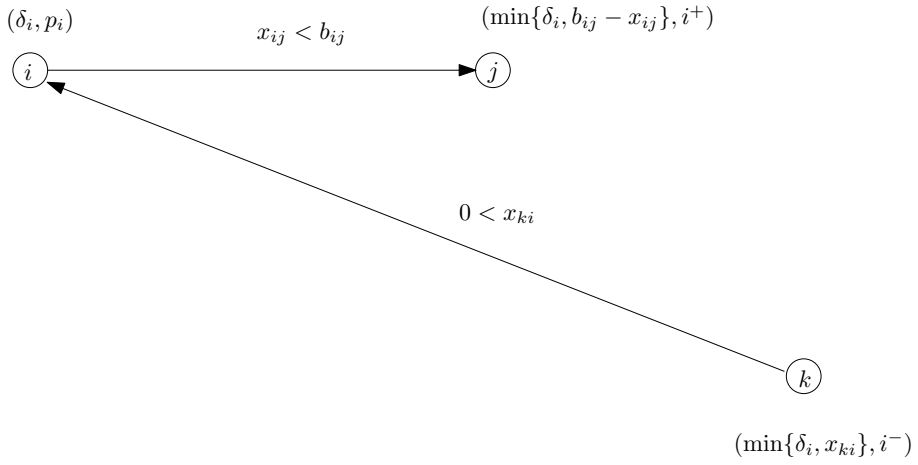
Cvičenie 5.2. Dokončite túto myšlienku uvedením podrobnej konštrukcie pre $G'(x)$.

Všimnime si, že pri celočíselných kapacitách uvedený algoritmus bude zachovávať celočíselnosť, t.j. ak je daný tok celočíselný (každá zložka je celočíselná, a teda aj jeho veľkosť je celočíselná), tak aj zväčšený tok bude celočíselný. (Porovnajte toky na obr. 5.2 a 5.4). Na tom je založené nasledovné tvrdenie.

VETA 5.3. *Ak sú všetky kapacity celočíselné, tak algoritmom Forda a Fulkersona možno nájsť maximálny s-t tok, ktorý je naviac celočíselný, pričom vystačíme s $O(mf^*)$ operáciami, kde f^* je veľkosť maximálneho s-t toku.*

Dôkaz: Ak začneme z nejakého celočíselného s-t toku s nezápornou veľkosťou (napr. nulového toku), tak stačí urobiť nanajvýš f^* zväčšovaní toku, lebo každý priebežný tok sa bude zväčšovať o celé kladné číslo. Na jedno značkovanie vystačíme s $O(m)$ operáciami a na následnú zmenu toku stačí $O(n) \leq O(m)$ operácií. ■

Cvičenie 5.3. Ukážte, že v prípade racionálnych kapacít tiež možno zaručiť konečnosť algoritmu Forda a Fulkersona.



Obr. 5.5: Značkovanie vrcholov pre hľadanie zväčšujúcej polocesty

V prípade iracionálnych kapacít nie je finitnosť zaručená. Na obr. 5.6 je príklad úlohy o maximálnom toku, kde algoritmus Forda a Fulkersona vykoná nekonečne mnoho zväčšovaní toku ak začneme z nulového toku a ďalšie zväčšovania robíme nasledovne. V tejto úlohe je r kladný koreň rovnice zlatého rezu

$$\frac{r}{1} = \frac{1-r}{r} \quad \text{t.j.} \quad r^2 = 1 - r$$

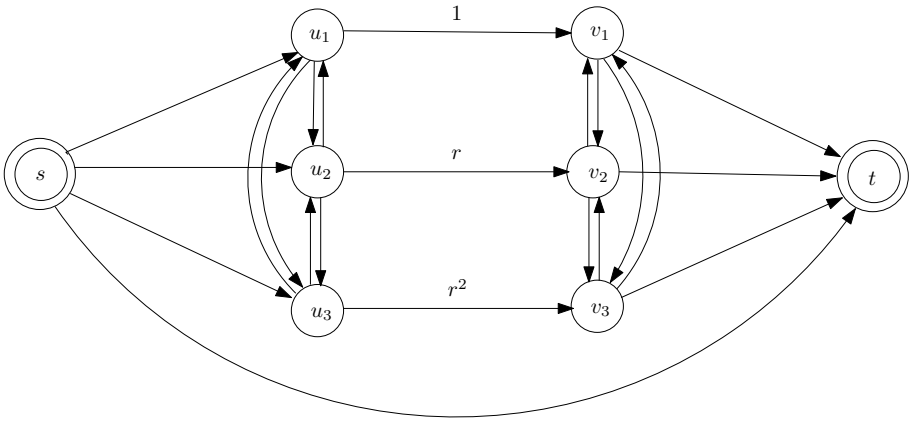
($r \approx 0.62$). Najprv zväčšíme tok o 1 po ceste (s, u_1, v_1, t) a budeme si všímať rozdiely kapacity a toku na trojici šípov $(u_1, v_1), (u_2, v_2)$ a (u_3, v_3) bez ohľadu na poradie. Momentálne máme: $0, r, r^2$. Ďalej zväčšíme tok o r^2 po poloceste $(s, u_3, v_3, v_1, u_1, u_2, v_2, t)$ a dostaneme trojicu: $0, r^2, r - r^2$. Z rovnice zlatého rezu vyplýva, že $r^{k+2} = r^k - r^{k+1}$ pre $k \geq 0$. Takže vlastne máme trojicu: 0, r^2, r^3. Vo všeobecnosti, ak máme tok x veľkosti $1 + r^2 + \cdots + r^k$, tak máme trojicu: $0, r^k, r^{k+1}$. Potom možno tok x zväčšiť o r^{k+1} a získať trojicu: $0, r^{k+1}, r^{k+2}$.

Cvičenie 5.4. Ukážte to.

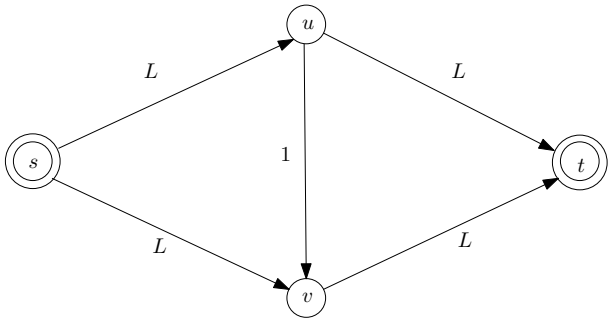
Takto v limite získame tok veľkosti $1 + r^2 + r^3 + \cdots = 1 + r^2/(1 - r) = 2$, ale maximálny tok má veľkosť $1 + r + r^2 + L = 2 + L$, čo môže byť veľmi veľké.

Žiaľ, ani pri celočíselných kapacitách nemôžeme byť s algoritmom Forda a Fulkersona spokojní, ako vidieť z príkladu na obr. 5.7. Tam je L veľké celé číslo. Veľkosť maximálneho s-t toku je zrejme $2L$ a pritom, začínajúc z nulového toku, algoritmus vykoná až $2L$ zväčšovaní toku vždy o jednotku. Skutočne, stačí striedavo aplikovať zväčšujúce polocesty (s, u, v, t) a (s, v, u, t). Keďže dĺžka úlohy je približne $4 \log_{10} L$, tak vidíme, že algoritmus Forda a Fulkersona nie je polynomiálny.

Uvedené zlé príklady naznačujú, že dômyselnejšia voľba zväčšujúcich polociest by mohla zrýchliť algoritmus Forda a Fulkersona. V oboch týchto prikla-



Obr. 5.6: Príklad s iracionálnymi kapacitami, kde $r = (\sqrt{5}-1)/2$ a všetky nevyznačené kapacity sa rovnajú $L \geq 3$



Obr. 5.7: Príklad s celočíselnými kapacitami ($L \gg 0$) kde algoritmus Forda a Fulkersona vykoná $2L$ zväčšovaní toku

doch by pomohlo, keby sme uprednostnili krátke zväčšujúce polocesty. Uvidíme, že to je dobrá myšlienka.

5.1.1 Metóda najkratších polociest

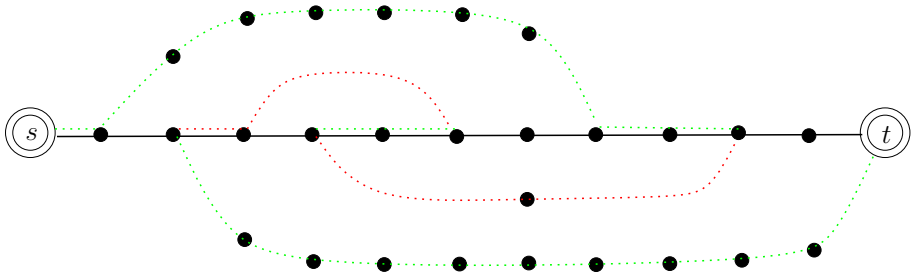
Túto metódu navrhol Dinic a nezávisle Edmonds a Karp. Vždy treba aplikovať najkratšiu zväčšujúcu s-t polocestu (t.j. polocestu s najmenším počtom hrán). Hrana digrafu, ktorá leží v aspoň jednej takej poloceste sa nazýva *pokrytá*.

Pre s-t tok x, ktorý nie je maximálny, nech $\mu(x)$ je dĺžka najkratšej zväčšujúcej s-t polocesty a nech $\sigma(x)$ je počet pokrytých hrán.

Lema 5.4. *Ak s-t tok x bol zväčšený po najkratšej zväčšujúcej s-t poloceste na tok x', ktorý ešte nie je maximálny, tak platí:*

(1) $\mu(x) \leq \mu(x')$,
(2) ak $\mu(x) = \mu(x')$, *tak* $\sigma(x) > \sigma(x')$.

Dôkaz: V súlade s obr. 5.8 uvažujme nakratšiu zväčšujúcu s-t polocestu P pre x (plná čiara) a najkratšiu zväčšujúcu s-t polocestu P' pre x' (bodkovaná čiara). Cestujúc po P', niektoré úseky prechádzame vzhľadom k P dopredu (vyznačené sú zelenou farbou) a iné dozadu (červené). Každý úsek má dĺžku aspoň 1, ale každý idúci dopredu má dĺžku aspoň takú ako premosťovaný úsek polocesty P, lebo inak by P nebola najkratšia. Preto, ak z je počet hrán polocesty P premostených úsekmi idúcimi dozadu (násobné premostenia sa započítajú násobne), tak dĺžka polocesty P' musí byť aspoň o z väčšia ako dĺžka polocesty P. Z toho vyplýva prvé tvrdenie lemy.



Obr. 5.8: Ilustrácia k dôkazu lemy 5.4

Pre druhé tvrdenie si stačí uvedomiť, že na poloceste P sa tok aspoň jednej hrany zmení tak, že takáto (kritická) hrana už nebude môcť byť prechádzaná na poloceste P' v tom istom smere ako na P. Ak by bola prechádzaná v opačnom smere, t.j. dozadu, tak by bolo $\mu(x) < \mu(x')$. Preto ak $\mu(x) = \mu(x')$, tak kritická hrana nemôže ležať na žiadnej najkratšej zväčšujúcej s-t poloceste P' pre x', a teda $\sigma(x) > \sigma(x')$. ■

VETA 5.5. *Metóda najkratších polociest dáva algoritmus zložitosti* $O(m^2n)$.

Dôkaz: Nech $x^0, x^1, x^2, ...x^q$ je postupnosť tokov vyrobená touto metódou. Keďže pre všetky i máme $\mu(x^i) \leq n-1$ a $\sigma(x^i) \leq m$, tak podľa lemy 5.4 postupnosť $\mu(x^0), \mu(x^1), \mu(x^2), ..., \mu(x^q)$ je neklesajúca a každý úsek stagnácie môže mať nanajvýš $m+1$ členov. Preto počet iterácií je $O(mn)$. Jedna iterácia (nájdenie zväčšujúcej polocesty a zmena toku) sa dá zrealizovať v čase $O(m+n) = O(m)$ aplikovaním Moorovho algoritmu (v netriviálnom prípade je $m \geq n-1$). ■

Poznamenajme, že uvedenú ideu sa podarilo zlepšiť na metódu, v ktorej sa uvažujú všetky najkratšie s-t cesty (v rezervnom digrafe) naraz, čo dalo algoritmus zložitosti $O(n^3)$. Pomerne úspešná je aj idea aplikovať najširšiu (t.j. s najväčšou rezervou) s-t polocestu. Hoci vo všeobecnosti nezaručuje konečnú procedúru, v praxi sa osvedčil a v prípade celočíselných kapacít dáva polynomiálny

algoritmus. Všimnite si, že tiež si ľahko poradí so zlými príkladmi z obr. 5.6 a 5.7.

Tok sa nazýva acyklický, ak digraf pozostávajúci z hrán kde tečie nenulový tok je acyklický. Zaujímavým pozorovaním je nasledujúce tvrdenie ilustrované na obr. 5.9.

VETA 5.6. *(a) Každý s-t tok možno rozložiť na acyklický a cirkuláciu.*
(b) Každý acyklický tok možno rozložiť na $O(m)$ s-t ciest.
(c) Každú cirkuláciu možno rozložiť na $O(m)$ cyklov.

Dôkaz:

Cvičenie 5.5. Dajte dôkaz.

■

Teda stačí nájsť maximálny acyklický *s-t* tok. Takýto tok možno získať z nulového postupným zväčšovaním po $O(m)$ zväčšujúcich *s-t* cestách (netreba polocesty).

Cvičenie 5.6. Ukážte to.

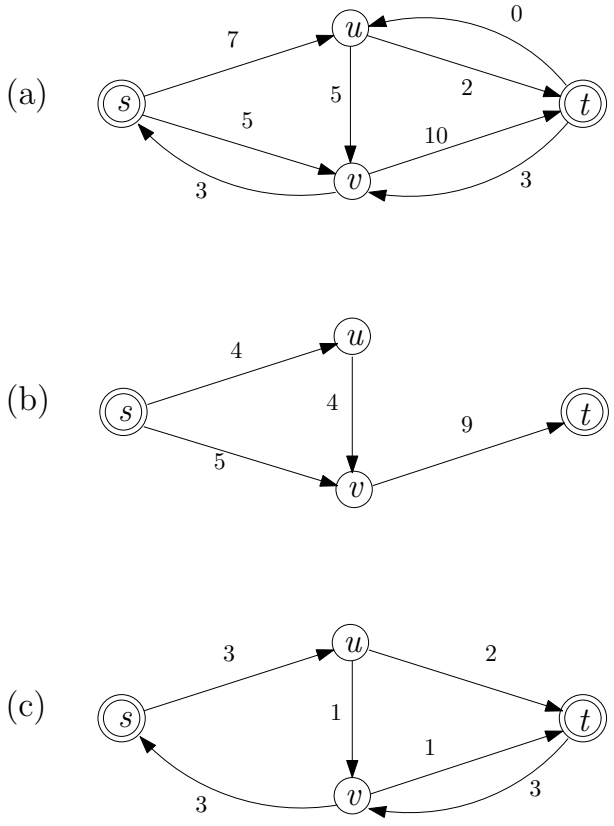
Žiaľ, algoritmicky to nevieme efektívne využiť.

5.2 Miery súvislosti grafov

Pripomeňme si, že graf môže byť súvislý, alebo nesúvislý. V tejto časti zavedieme meranie súvislosti a tým presnejšie rozlišovanie. Takže nejaký graf bude „viac súvislý“ ako iný. Ak si graf predstavíme ako model komunikačnej siete, tak je potrebné, aby bol súvislý. Ak sa však zamyslíme nad odolnosťou takejto siete proti poruchám, tak obr. 5.10 nám naznačuje, čo si budeme všímať v súvislom grafe. V jednom prípade pôjde o množinu hrán, ktorej vynechaním získame nesúvislý graf. Ak budeme naviac žiadať, aby to bola minimálna (v zmysle inklúzie) množina s touto vlastnosťou, tak sa nazýva *hranový rez*. Podobne definujeme *vrcholový rez*. Minimálny počet hrán, ktorých vynechaním získame z grafu G nesúvislý graf alebo triviálny (jednovrcholový), sa nazýva **hranová súvislosť** a označíme ju symbolom $\lambda(G)$ (pre netriviálny graf je to minimálny počet prvkov hranového rezu grafu G). Podobne definujeme **vrcholovú súvislosť** grafu a označujeme ju symbolom $\kappa(G)$ (pre nekompletný graf je to minimálny počet prvkov vrcholového rezu). Všimnime si, že z kompletného grafu sa vynechávaním vrcholov nedá získať nesúvislý graf. Na obr. 5.10 máme graf G s hodnotami $\lambda(G) = 3$ a $\kappa(G) = 2$, lebo menej početné rezy tam neexistujú. Našou hlavnou úlohou bude zisťovanie týchto dvoch parametrov pomocou tokov. Z tohoto dôvodu je vhodné začať od digrafov, kde tieto parametre definujeme formálne rovnako, len namiesto súvislosti uvažujeme silnú súvislosť.

HRANOVÁ SÚVISLOSŤ DIGRAFOV

Podľa definície je hranová súvislosť $\lambda(G)$ digrafu G minimálny počet hrán, ktorých vynechaním získame z G triviálny digraf, alebo digraf bez silnej súvislosti. Ak je digraf triviálny, tak $\lambda(G) = 0$, a preto sa stačí zaoberať problémom



Obr. 5.9: Rozklad toku z obr. (a) na acyklický tok a cirkuláciu (obr. (b) a (c))

pre netriviálne digrafy. K tomu zavedieme tzv. *lokálnu hranovú súvislosť* $\lambda(u,v)$ pre dané 2 rôzne vrcholy u a v (v tomto poradí), čo je minimálny počet hrán separujúcich u a v (t.j. minimálny počet hrán potrebný na porušenie všetkých u-v ciest). Potom

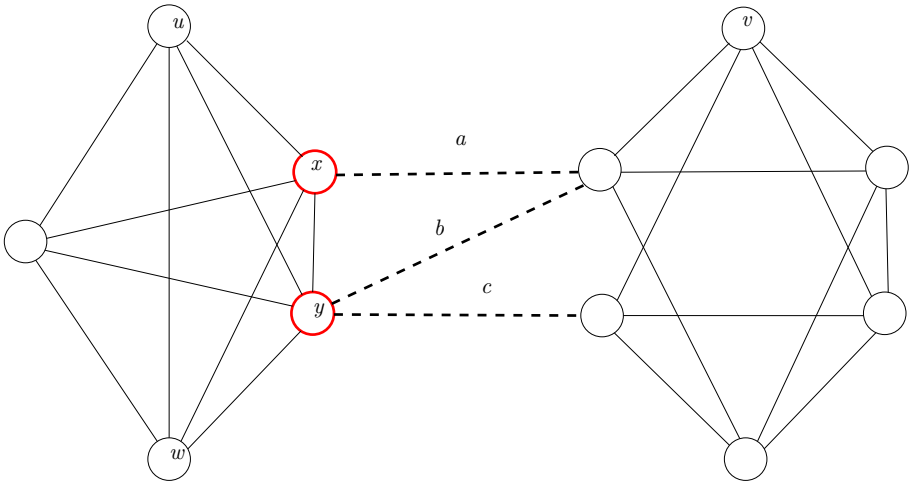
$$\lambda(G) = \begin{cases} 0, & \text{ak } n = 1 \\ \min\{\lambda(u,v) | u,v \in V(G), u \neq v\}, & \text{inak} \end{cases} \tag{5.6}$$

VETA 5.7. *(Menger) Nech G je graf alebo digraf a $u, v \in V(G)$ sú 2 rôzne vrcholy. Potom*

(i) minimálny počet $\lambda(u,v)$ hrán separujúcich u a v sa rovná maximálnemu počtu $\lambda'(u,v)$ hranovo-disjunktných u-v ciest,

(ii) ak neexistuje hrana (u,v), tak minimálny počet $\kappa(u,v)$ vrcholov separujúcich u a v sa rovná maximálnemu počtu $\kappa'(u,v)$ vnútorne-disjunktných u-v ciest (okrem vrcholov u a v nemajú spoločný vrchol).

Dôkaz: Obmedzíme sa na digrafy, lebo pre naše ciele stačí graf previesť na



Obr. 5.10: Vrcholový rez $\{x, y\}$ a hranový rez $\{a, b, c\}$ grafu

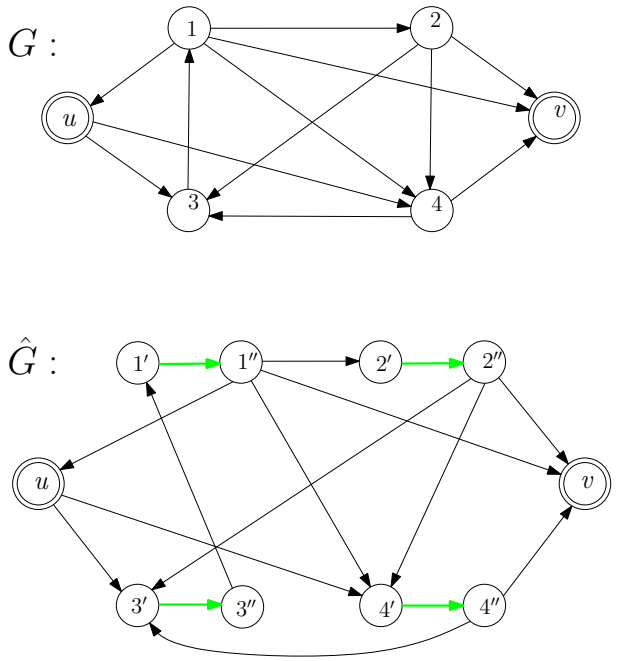
digraf nahradením každej hrany $\{u, v\}$ dvojicou protišípov (u, v) a (v, u).

(i) Uvažujme nejaký systém $\lambda'(u, v)$ hranovo-disjunktných u-v ciest. Ak chceme u a v separovať, tak každú cestu tohoto systému musíme porušiť, t.j. vynechať z nej aspoň jednu hranu. Teda $\lambda(u, v) \geq \lambda'(u, v)$.

Obrátenú nerovnosť je ťažšie dokázať. Využijeme teóriu tokov. Dajme každému šípu kapacitu 1 a uvažujme maximálny u-v tok x. Podľa vety 5.3 môžeme predpokladať, že je celočíselný a podľa vety 5.6, že je acyklický a rozložený do u-v ciest. Keďže máme jednotkové kapacity šípov, tak tieto cesty sú hranovo-disjunktné a ich počet sa rovná veľkosti f toku x. Teda $\lambda'(u, v) \geq f$. Podľa vety 5.2 existuje u-v rez kapacity (a teda mohutnosti) f, ktorého vynechaním prerušíme všetky u-v cesty, a preto $\lambda(u, v) \leq f \leq \lambda'(u, v)$.

(ii) Analogicky ako v dôkaze (i) vidíme, že $\kappa(u, v) \geq \kappa'(u, v)$. K digrafu G zostrojme nový digraf $\hat{G}$ tak, že každý vrchol $w \in V(G) \setminus \{u, v\}$ rozštiepime na dva vrcholy w' a w'' a pridáme šíp (w', w''). Potom každý šíp prichádzajúci v G do w bude prichádzať v $\hat{G}$ do w' a každý odchádzajúci z w bude odchádzať z w''. Príklad tejto konštrukcie je na obr. 5.11. Teraz šípom (w', w'') digrafu $\hat{G}$ dáme kapacity 1 a ostatným veľké celočíselné kapacity (napr. m). Maximálny u-v tok x nemôže na žiadnom šípe prekročiť 1, a preto môžeme predpokladať, že je celočíselný, acyklický a rozložený do u-v ciest. Tieto sú nielen hranovo-disjunktné, ale aj vnútorne-disjunktné a korešpondujúce u-v cesty v G sú tiež také. Rovnaké úvahy ako pre (i) nám dajú požadovanú nerovnosť, lebo z voľby kapacít vyplýva, že minimálny u-v rez zodpovedajúci toku x bude realizovaný v $\hat{G}$ iba šípmi typu (w', w'') (zelené šípy na obr. 5.11) a teda vrcholmi v originálnom digrafe G.

■



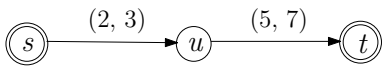
Obr. 5.11: Konštrukcia digrafu $\hat{G}$ pre náhradu vrcholového rezu hranovým

5.3 Toky s dolnými medzami

Ide o zovšeobecnenie klasického toku kde pre každý šíp (i, j) je okrem kapacity (hornej medze) b_{ij} na tok daná aj dolná medza $a_{ij} \geq 0$. Takže kapacitné obmedzenie (5.1) sa zmení na

$$a_{ij} \leq x_{ij} \leq b_{ij} \quad \forall (i,j) \in E \tag{5.7}$$

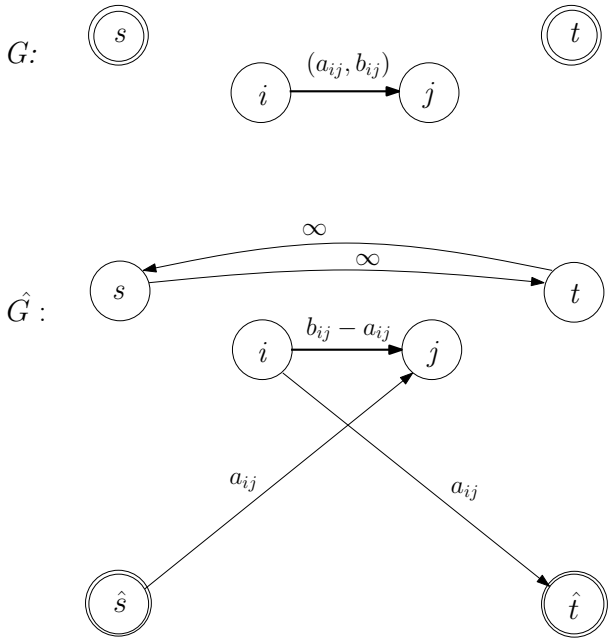
Podmienky kontinuity zostávajú rovnaké a tiež rovnako ako prv meriame veľkosť toku. Zatiaľčo klasický tok vždy existoval (napr. nulový tok), tok s dolnými medzami nemusí vždy existovať, ako ukazuje príklad z obr. 5.12. Takto



Obr. 5.12: Tok s dolnými medzami tu neexistuje

celkom prirodzene môžeme postaviť nasledujúce úlohy pre toky s dolnými medzami.
(1) Nájsť prípustný s-t tok.
(2) Nájsť maximálny s-t tok.
(3) Nájsť minimálny s-t tok.

Prípustný s-t tok v digrafe G s dolnými medzami možno nájsť pomocou klasického toku tak, ako ukazuje obr. 5.13. Na

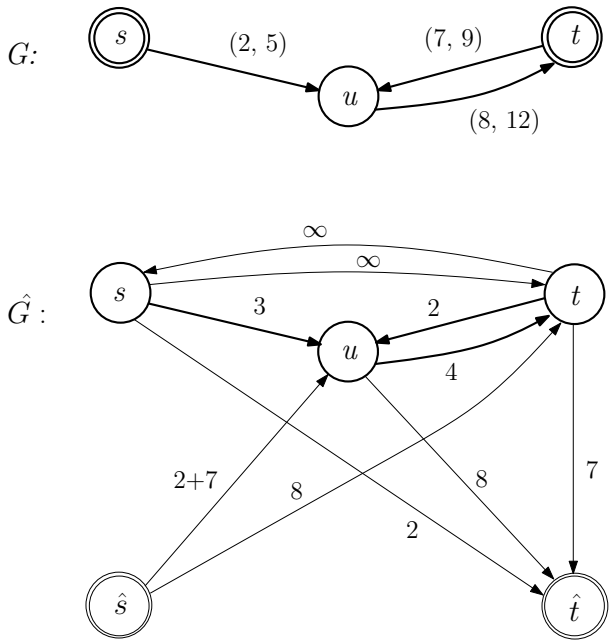


Obr. 5.13: Transformácia úlohy nájsť prípustný tok s dolnými medzami

každom šípe (i, j) s dolnou medzou a_{ij} a hornou medzou b_{ij} tieto medze zmenšíme o a_{ij}, čím získame kapacitu $b_{ij} - a_{ij}$. Aby to nenarušilo príslušný tok, tak na začiatku (vo vrchole i) odvedieme z neho množstvo a_{ij} do pomocného vrchola $\hat{t}$ a také isté množstvo zase privedieme na konci (vo vrchole j) z pomocného vrchola $\hat{s}$. Privádzajúcemu aj odvádzajúcemu šípu dáme kapacitu a_{ij}. Pôvodný zdroj s a pôvodný stok t zmeníme na tranzitné vrcholy pridaním dvoch šípov (t, s) a (s, t) kapacity ∞ (ak v G máme s-t tok veľkosti $f \geq 0$ tak po prvom z nich vedieme množstvo f a po druhom 0; ak je $f < 0$, tak po druhom vedieme množstvo $-f$ a po prvom 0). Aby sme sa vyhli násobným šípom, tak ich nahradíme jedným so sumárnou kapacitou. Tým získame nový digraf $\hat{G}$ so zdrojom $\hat{s}$ a stokom $\hat{t}$, v ktorom sú všetky dolné medze nulové. Príklad ilustrujúci túto konštrukciu je na obr. 5.14.

Vidíme, že ak v G existuje s-t tok x, tak v $\hat{G}$ existuje $\hat{s}$-$\hat{t}$ tok y, ktorý nasycuje všetky šípy pri $\hat{s}$ (a teda aj všetky šípy pri $\hat{t}$). Keďže platí aj obrátená implikácia a nasycujúci tok je maximálny, tak dostávame:

VETA 5.8. *V digrafe G s dolnými a hornými medzami existuje prípustný s-t tok $x \Leftrightarrow$ v digrafe $\hat{G}$ ľubovoľný maximálny $\hat{s}$-$\hat{t}$ tok y nasycuje všetky šípy pri vrcholoch $\hat{s}$ a $\hat{t}$.*



Obr. 5.14: Digraf $\hat{G}$ pre príklad úlohy s dolnými medzami

Dôkaz:

Cvičenie 5.7. Dajte dôkaz.

■

Ak už máme prípustný s-t tok, tak maximálny s-t tok možno nájsť analogicky ako v klasickom prípade pomocou zväčšujúcich polociest keďže uvedený postup a teóriu možno ľahko zovšeobecniť.

Cvičenie 5.8. Sformulujte a dokážte príslušné tvrdenia.

. Podobne možno pristupovať aj k úlohe nájsť minimálny s-t tok.

Cvičenie 5.9. Sformulujte a dokážte potrebné tvrdenia.

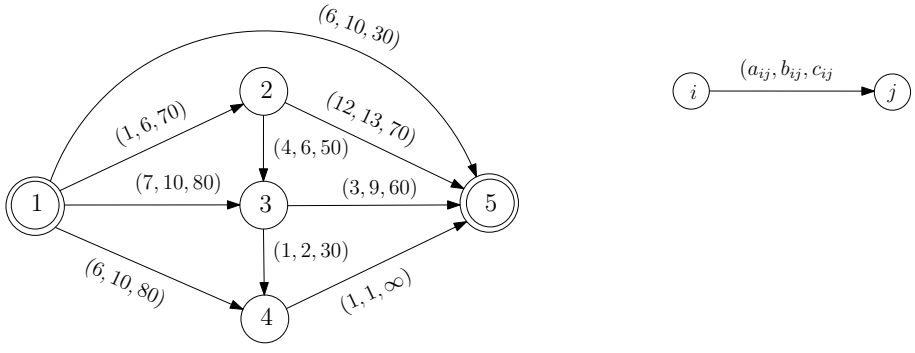
.

5.4 Nákladová analýza projektu

Uvažujme digraf projektu, kde činnosti sú reprezentované šípmi tak ako v časovej analýze projektu. Teraz predpokladáme, že pre každú činnosť (i, j) je dané normálne trvanie b_{ij}, minimálne trvanie a_{ij} a cena (náklady) c_{ij} za skrátenie

trvania činnosti o jednotku v intervale $[a_{ij}, b_{ij}]$. Teda pre realizačný čas y_{ij} tejto činnosti musí platiť $a_{ij} \leq y_{ij} \leq b_{ij}$ a náklady za skrátenie budú $c_{ij}(b_{ij} - y_{ij})$. Napríklad pri stavbe malého domu máme také činnosti ako kopanie základov, betónovanie základov, výstavba múrov, konštrukcia strechy, omietanie múrov, vodoinštalácie, plynoinštalácie, atď. Predpokladajúc kopanie základov bagrom odhadneme normálne trvanie napr. na 4 hodiny, ale minimálne trvanie bude tiež 4 hodiny, lebo 2 bagre by sa tam nezmestili. Na druhej strane, zhotovenie podlahy odhadneme napr. na 20 hodín ako normálne trvanie, ale minimálne trvanie môže byť iba 2 hodiny (ak to bude robiť 10-krát viac podlahárov; keby ich bolo ešte viac, tak by si vzájomne prekážali a práca by sa spomalila). Samozrejme, viac pracovníkom musíme aj viac zaplatiť. Avšak niekedy môžeme prácu zrýchliť iba použitím novej, ale drahšej technológie. Napr. postaviť klasické lešenie môže vyžadovať mnoho skrutkovania, zatiaľčo moderná stavebnica to nevyžaduje. Budeme predpokladať, že digraf projektu má n vrcholov $1, 2, \ldots, n$, pričom vrchol 1 zodpovedá začiatku projektu a vrchol n koncu. Tak je to aj na obr. 5.15. Činnosti, ktoré nemožno skrátiť majú normálne a minimálne trvanie rovnaké; v takom prípade na cene za skrátenie nezáleží (zvyčajne ju dávame 0), ale pre účely nášho algoritmu je to na obr. 5.15 zvýraznené nekonečnou cenou za skrátenie.

Na základe normálnych trvaní jednotlivých činností sme v časovej analýze (pozri acyklický algoritmus v časti 3.1) zistili normálne trvanie projektu (t.j. minimálny realizačný čas projektu) dané dĺžkou najdlhšej (tzv. kritickej) 1-n cesty. Úlohou *nákladovej analýzy* je zistiť, či toto trvanie možno skrátiť na základe skrátenia niektorých činností a koľko za to musíme zaplatiť. Presnejšie, treba určiť najmenšie možné trvanie projektu a pre každý čas z intervalu od najmenšieho po normálne trvanie určiť minimálne náklady za skrátenie. Graf tejto funkcie sa nazýva **nákladová krivka projektu**. Pre náš príklad je na obr. 5.21.



Obr. 5.15: Príklad digrafu projektu pre nákladovú analýzu

Nákladovú analýzu možno sformulovať ako úlohu parametrického lineárneho programovania nasledovne. Nech neznáma y_{ij} označuje skutočné trvanie čin-

nosti (i, j) a x_i čas, v ktorom sú už všetky činnosti prichádzajúce do vrcholu i zrealizované a môžu začať činnosti odchádzajúce z tohoto vrcholu. Ak λ bude parameter označujúci trvanie celého projektu, tak dostávame úlohu:

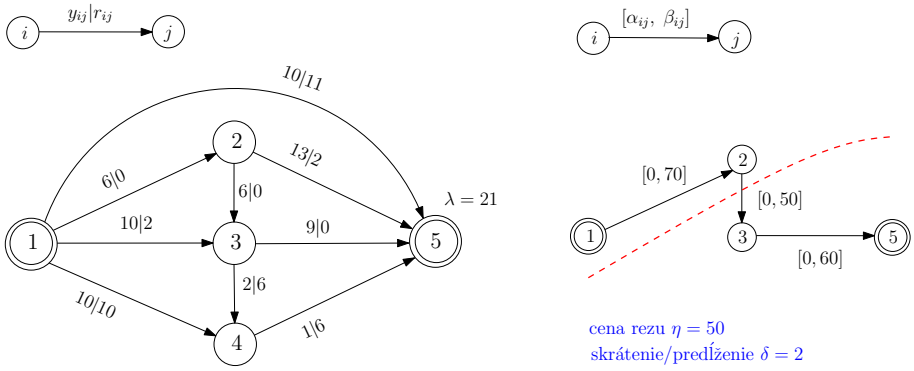
$$\sum_{(i,j)\in E(G)} c_{ij}(b_{ij} - y_{ij}) \to \min \tag{5.8}$$

$$x_j \geq x_i + y_{ij} \qquad \forall (i,j) \in E(G) \tag{5.9}$$

$$x_n = x_1 + \lambda \tag{5.10}$$

$$a_{ij} \leq y_{ij} \leq b_{ij} \qquad \forall (i,j) \in E(G) \tag{5.11}$$

Časť účelovej funkcie je konštantná a teda ju môžeme vynechať. Dostaneme tak minimalizačnú parametrickú úlohu lineárneho programovania s parametrom iba na pravej strane, ktorej každá zložka (v našej úlohe iba jedna) je afinne lineárnou funkciou parametra. O takejto úlohe vieme z lineárneho programovania, že množina tých parametrov kde existuje optimálne riešenie je interval a optimálna hodnota účelovej funkcie je tam konvexná a po častiach lineárna (ako na obr. 5.21). Takúto úlohu vieme riešiť štandardným postupom založeným na simplexovej metóde, ale pre jej častú rozsiahlosť je výhodnejšie použiť špecializované metódy. Takéto metódy boli vypracované už okolo r. 1960 na báze istého algoritmu pre najlacnejší tok. Neskôr bol predložený modifikovaný postup, ktorý má intuitívne jednoduchšie zdôvodnenie. Tento tzv. *rezový algoritmus* uvedieme bez formálneho dôkazu, ale načrtneme základné myšlienky, na ktorých je algoritmus postavený a ilustrujeme na príklade.

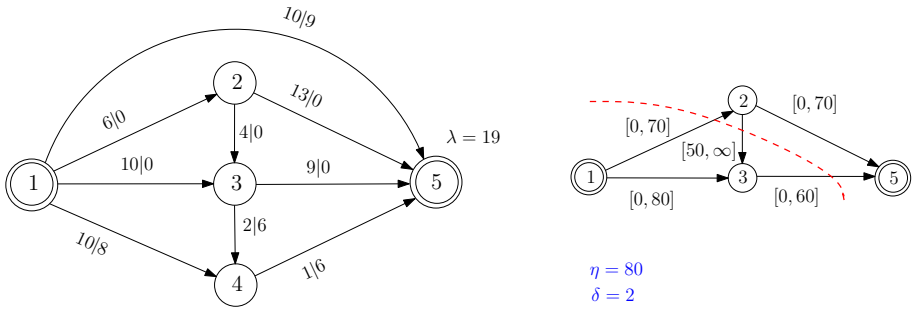


Obr. 5.16: 1. iterácia riešenia úlohy z obr. 5.15

Predpokladajme, že pri normálnych trvaniach má projekt trvanie λ dané dĺžkou kritickej 1-n cesty, pritom kritických ciest môže byť aj viac. Ak chceme projekt skrátiť, tak každú kritickú cestu musíme skrátiť. Ekonomicky sa to urobí tak, že zostrojíme tzv. kritický digraf, ktorý je zjednotením všetkých kritických ciest, t.j. pozostáva zo šípov, ktoré majú nulovú rezervu, a v ňom nájdeme

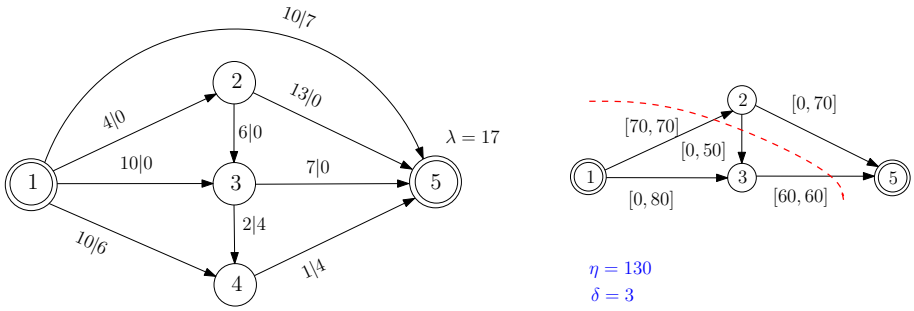
najlacnejší 1-n rez (hranový). Takýto rez možno nájsť pomocou algoritmu pre maximálny 1-n tok, ak za kapacity dáme ceny šípov (činností) za skrátenie. Pre náš malý príklad rez vždy nájdeme preskúmaním všetkých možností. Uvažujeme už uvedený príklad z obr. 5.15. Na obr. 5.16 je prvá iterácia algoritmu. Vľavo je základný digraf s priebežnými (na začiatku s normálnymi) trvaniami činností a vypočítanými časovými rezervami; priebežné trvanie projektu je $\lambda = 21$. Vpravo je kritický digraf s kapacitami $\beta_{ij} = c_{ij}$ a vyznačený minimálny rez; jeho cena je $\eta = 50$. Ďalej určíme maximálne číslo δ, o ktoré možno každý šíp (činnosť) rezu skrátiť. (Každý šíp rezu skracujeme o to isté číslo, čím sa každá kritická cesta rovnako skráti.) Tu musíme byť opatrný, lebo treba dodržať obmedzenie 5.11; to nám teraz dovoľuje skrátiť o 2 jednotky. Okrem toho, žiadna nekritická 1-n cesta sa nesmie stať dlhšou ako skrátené kritické cesty. Z tohoto dôvodu nebudeme skracovať o viac ako je minimálna kladná rezerva činnosti, čo je u nás tiež 2. Takže $\delta = 2$.

Prejdeme k 2. iterácii, ktorá už má všeobecný charakter. Vychádzame z nových „normálnych trvaní“, ktoré sme získali v predošlej iterácii. Pre zabezpečenie optimálneho skrátenia projektu musíme pripustiť, že niektoré prv skrátené šípy bude potrebné predĺžiť. Na obr. 5.17 vľavo sú už uvedené trvania činností po skrátení z 1. iterácie, čo dáva nové trvanie projektu $\lambda = 19$ a aj nové rezervy. Vpravo je príslušný kritický digraf, kde na každom šípe je dvojica $[\alpha_{ij}, \beta_{ij}]$. Číslo β_{ij} je cena za skrátenie a α_{ij} zisk za predĺženie činnosti o jednotku. Ak je už trvanie činnosti na dolnej hranici a_{ij}, tak sa nedá skrátiť (a ani skrátenie zaplatiť), a preto je tam $\beta_{ij} = \infty$. Teraz hľadáme najlacnejší obojsmerný 1-n rez, kde nápredné šípy sa počítajú s cenou β_{ij} a opačné šípy s cenou $-\alpha_{ij}$. Potom nápredné šípy rezu budeme skracovať a opačné šípy predlžovať o to isté číslo δ. Cena rezu je $\eta = 70 - 50 + 60 = 80$. Obmedzenie 5.11 nám dáva $\delta = 2$ (kladné rezervy sú väčšie).

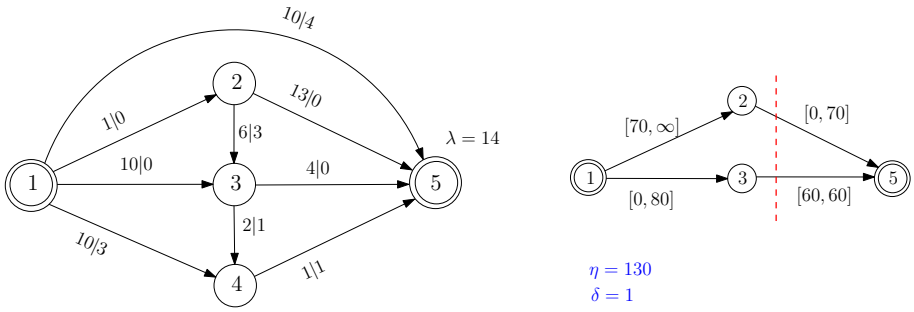


Obr. 5.17: 2. iterácia riešenia úlohy z obr. 5.15

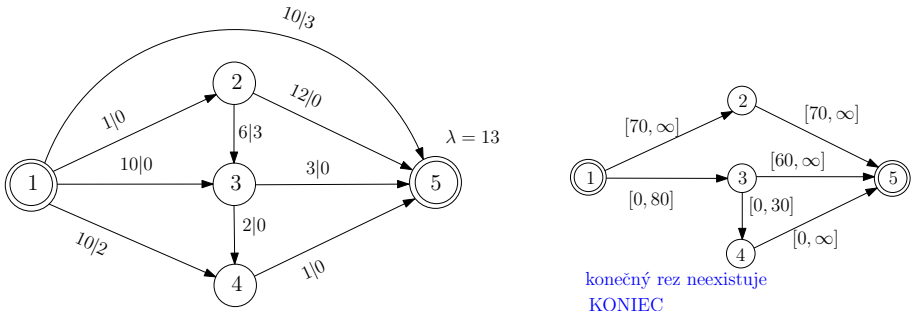
Ďalšie 2 iterácie sú na obr. 5.18 a 5.19 a prebiehajú rovnakým spôsobom. Finálna iterácia je na obr. 5.20 kde v kritickom digrafe už neexistuje 1-5 rez konečnej ceny, čo znamená, že projekt už nemožno viac skrátiť. Výsledky výpočtov dávajú nákladovú krivku z obr. 5.21.



Obr. 5.18: 3. iterácia riešenia úlohy z obr. 5.15



Obr. 5.19: 4. iterácia riešenia úlohy z obr. 5.15

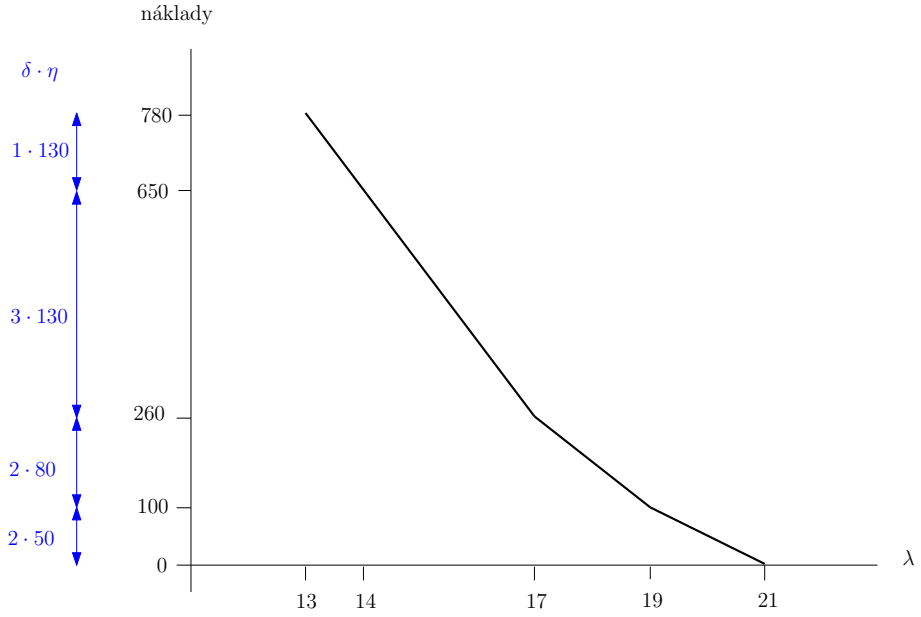


Obr. 5.20: 5. (posledná) iterácia riešenia úlohy z obr. 5.15

Zosumarizujeme pravidlá rezovej metódy pre jednu iteráciu.

(1) V základnom digrafe s priebežnými normálnymi trvaniami y_{ij} (na začiatku $y_{ij} = b_{ij}$) časovou analýzou vypočítame príslušné trvanie projektu a rezervy činností r_{ij}.

(2) Zostrojíme kritický digraf (pozostáva zo šípov s rezervou $r_{ij} = 0$) a



Obr. 5.21: Nákladová krivka pre úlohu z obr. 5.15

každému jeho šípu priradíme dvojicu $[\alpha_{ij}, \beta_{ij}]$, kde

$$[\alpha_{ij}, \beta_{ij}] = \begin{cases} [0, c_{ij}], & \text{ak} \quad a_{ij} < y_{ij} = b_{ij} \\ [c_{ij}, c_{ij}], & \text{ak} \quad a_{ij} < y_{ij} < b_{ij} \\ [c_{ij}, \infty], & \text{ak} \quad a_{ij} = y_{ij} < b_{ij} \\ [0, \infty], & \text{ak} \quad a_{ij} = y_{ij} = b_{ij} \end{cases} \tag{5.12}$$

(3) V kritickom digrafe nájdeme najlacnejší obojsmerný 1-n rez (S, T) (t.j. rozklad vrcholovej množiny kde $1 \in S, n \in T$ a $\beta(S,T) - \alpha(T,S)$ je konečné a minimálne). To možno urobiť napr. pomocou algoritmu pre nájdenie maximálneho 1-n toku s dolnými medzami α a kapacitami β, ktorý po nájdení maximálneho toku nám dáva aj minimálny rez). Ak neexistuje rez konečnej veľkosti, tak už projekt nemožno ďalej skrátiť (to zbadáme podľa toho, že nájdeme 1-n tok nekonečnej veľkosti).

(4) Nájdeme maximálne $\delta > 0$, o ktoré možno skrátiť dopredné šípy rezu (S, T) (šípy idúce z S do T), predĺžiť spätné šípy rezu (S, T) (šípy idúce z T do S), ak sú predĺžiteľné a pritom sa žiadna nekritická cesta nestala dlhšou ako modifikovaná kritická cesta. Takto máme:

$$\delta = \min \begin{cases} y_{ij} - a_{ij}, & \text{kritická činnosť } (i,j) \in (S,T) \\ b_{ij} - y_{ij}, & \text{kritická činnosť } (i,j) \in (T,S) \text{ kde } y_{ij} < b_{ij} \\ r_{ij}, & \text{nekritická činnosť } (i,j) \end{cases} \tag{5.13}$$

Potom dostaneme modifikované trvania činností:

$$y'_{ij} = \begin{cases} y_{ij} - \delta, & \text{kritická činnosť } (i,j) \in (S,T) \\ y_{ij} + \delta, & \text{kritická činnosť } (i,j) \in (T,S) \text{ a } y_{ij} < b_{ij} \\ y_{ij}, & \text{kritická činnosť } (i,j) \in (T,S) \text{ a } y_{ij} = b_{ij} \\ y_{ij}, & \text{nekritická činnosť } (i,j) \end{cases} \tag{5.14}$$

Cvičenie 5.10. Pri hľadaní maximálneho toku s dolnými medzami sme v časti 5.3 najprv hľadali prípustný tok. Ukážte, že v kritickom digrafe sú v 1. iterácii dolné medze nulové a v každej ďalšej iterácii možno využiť tok z predchádzajúcej iterácie.

Cvičenie 5.11. Urobte nákladovú analýzu pre nasledujúce digrafy projektu 5.15 až 5.19. Začiatok je vo vrchole 1 a koniec vo vrchole s maximálnym číslom n; v každom riadku je popísaná jedna činnosť (i,j): $[i \quad j \quad a_{ij} \quad b_{ij} \quad c_{ij}]$. [Vpravo je výsledok (súradnice zlomových bodov); v každom riadku jeden bod (trvanie, náklady). Pre cvičenie 5.18 uvádzame aj nákladovú krivku (obr. 5.22).]

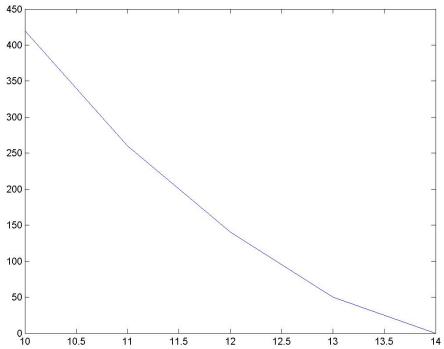
$$\begin{bmatrix} 1 & 2 & 2 & 5 & 20 \\ 1 & 3 & 5 & 9 & 50 \\ 2 & 3 & 3 & 4 & 30 \\ 2 & 4 & 6 & 10 & 60 \\ 3 & 4 & 2 & 6 & 40 \end{bmatrix} \qquad \begin{pmatrix} 8 & 610 \\ 11 & 280 \\ 12 & 180 \\ 15 & 0 \end{pmatrix} \tag{5.15}$$

$$\begin{bmatrix} 1 & 2 & 2 & 7 & 70 \\ 1 & 3 & 3 & 8 & 20 \\ 2 & 3 & 1 & 4 & 40 \\ 2 & 4 & 8 & 9 & 20 \\ 3 & 4 & 1 & 7 & 50 \end{bmatrix} \qquad \begin{pmatrix} 10 & 560 \\ 12 & 380 \\ 15 & 140 \\ 16 & 80 \\ 18 & 0 \end{pmatrix} \tag{5.16}$$

$$\begin{bmatrix} 1 & 2 & 1 & 3 & 3 \\ 1 & 3 & 2 & 4 & 4 \\ 2 & 3 & 0 & 0 & 0 \\ 2 & 4 & 2 & 4 & 1 \\ 2 & 5 & 5 & 5 & 0 \\ 3 & 4 & 1 & 6 & 3 \\ 4 & 5 & 4 & 4 & 0 \end{bmatrix} \qquad \begin{pmatrix} 7 & 31 \\ 9 & 17 \\ 11 & 9 \\ 14 & 0 \end{pmatrix} \tag{5.17}$$

$$\begin{bmatrix} 1 & 2 & 8 & 10 & 6 \\ 1 & 3 & 14 & 16 & 2 \\ 2 & 3 & 4 & 6 & 2 \\ 2 & 4 & 12 & 14 & 2 \\ 3 & 4 & 6 & 8 & 6 \end{bmatrix} \qquad \begin{pmatrix} 20 & 32 \\ 22 & 12 \\ 24 & 0 \end{pmatrix} \tag{5.18}$$

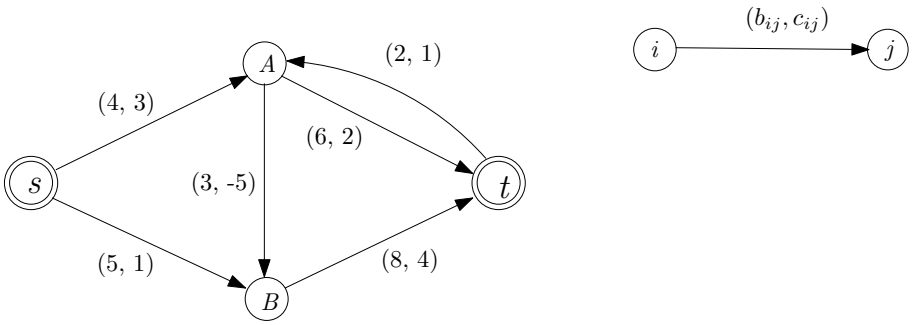
$$\begin{bmatrix} 1 & 2 & 3 & 4 & 60 \\ 1 & 3 & 5 & 8 & 70 \\ 2 & 3 & 3 & 5 & 50 \\ 2 & 4 & 7 & 9 & 40 \\ 3 & 4 & 3 & 5 & 80 \end{bmatrix} \qquad \begin{pmatrix} 10 & 420 \\ 11 & 260 \\ 12 & 140 \\ 13 & 50 \\ 14 & 0 \end{pmatrix} \tag{5.19}$$



Obr. 5.22: Nákladová krivka k cvičeniu 5.19

5.5 Najlacnejší maximálny tok

Vo všeobecnosti môže existovať viac maximálnych s-t tokov a pri daných cenách na šípoch má zmysel pýtať sa na taký maximálny tok, ktorý má minimálnu cenu. Presnejšie, na každom šípe (i, j) máme predpísanú okrem kapacity aj cenu (poplatok) c_{ij} za jednotku toku na tom šípe (pozri príklad na obr. 5.23).



Obr. 5.23: Príklad úlohy o najlacnejšom maximálnom s-t toku

Predpokladáme, že náklady sú lineárnou funkciou veľkosti toku, t.j. za tok veľkosti x_{ij} zaplatíme $c_{ij}x_{ij}$. Úlohou je spomedzi maximálnych s-t tokov vybrať najlacnejší, t.j. taký, ktorý má sumárnu cenu za toky na všetkých šípoch

minimálnu. Ak veľkosť maximálneho s-t toku je f^*, tak úlohu nájsť najlacnejší maximálny tok môžeme sformulovať ako úlohu lineárneho programovania:

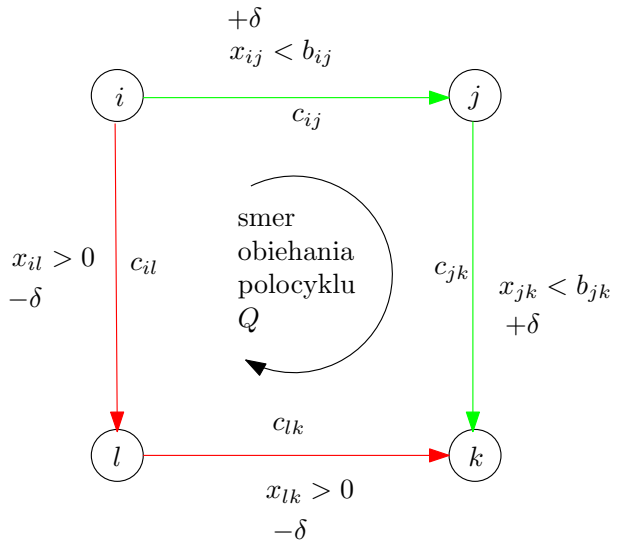
$$\sum_{(i,j)\in E} c_{ij}x_{ij} \to \min \tag{5.20}$$

$$0 \le x_{ij} \le b_{ij} \quad \forall (i,j) \in E \quad \text{(kapacitné obmedzenia)} \tag{5.21}$$

$$\sum_{i:(i,k)\in E} x_{ik} = \sum_{j:(k,j)\in E} x_{kj} \quad \forall k \in V \setminus \{s,t\} \quad \text{(podmienky kontinuity)} \tag{5.22}$$

$$\sum_{j:(s,j)\in E} x_{sj} - \sum_{i:(i,s)\in E} x_{is} = f^* \quad \text{(maximálnosť toku)} \tag{5.23}$$

Metódami lineárneho programovania vieme túto úlohu vyriešiť, ale tu uprednostníme grafový prístup a uvedieme Kleinov algoritmus, ktorý je ideove blízky k algoritmu Forda a Fulkersona. Klein navrhuje najprv nájsť nejaký maximálny s-t tok x a potom ho postupne zlacňovať po záporných rezervných polocykloch. Obr. 5.24 ilustruje tento pojem.



Obr. 5.24: Príklad rezervného polocyklu Q; súhlasné šípy sú zelené a nesúhlasné červené; $\delta = \min\{b_{ij} - x_{ij}, b_{jk} - x_{jk}, x_{lk}, x_{il}\}$; $c(Q) = c_{ij} + c_{jk} - c_{lk} - c_{il}$

Šípy polocyklu, ktoré sú orientované v smere obiehania polocyklu nazývame súhlasné a tie opačne orientované nesúhlasné. Rezervný polocyklus Q má na súhlasných šípoch tok menší ako kapacitu a na nesúhlasných je tok kladný. Potom je ľahko vidieť, že existuje také číslo $\delta > 0$, o ktoré možno tok x na súhlasných šípoch zväčšiť o δ a na nesúhlasných zmenšiť o δ a tým získať nový s-t tok x' s rovnakou veľkosťou ako mal x.

Cvičenie 5.12. Ukážte to.

Naviac, definujme cenu $c(Q)$ polocyklu Q ako súčet cien súhlasných šípov a súčet záporne vzatých cien na nesúhlasných šípoch. Potom

$$c(x') = c(x) + \delta c(Q).$$

Polocyklus Q so zápornou cenou sa nazýva záporný. Z posledného vzťahu vidíme, že ak je Q záporný rezervný polocyklus, tak nový tok x' bude lacnejší maximálny s-t tok ako x. Záporný rezervný polocyklus zodpovedá zápornému cyklu v rezervnom (multi)digrafe $G'(x)$ a tam ho hľadáme pomocou algoritmov pre najkratšie cesty (napr. Floydov algoritmus).

Ešte uveďme, že číslo δ volíme maximálne možné a teda je to minimálna rezerva na šípoch polocyklu, pričom rezerva šípu (i,j) sa rovná $b_{ij} - x_{ij}$ ak je súhlasný a x_{ij}, ak je nesúhlasný. (Pozri obr. 5.24.) Kleinov algoritmus aplikovaný na príklad z obr. 5.23 je predvedený na obr. 5.25.

VETA 5.9. *Maximálny s-t tok x je najlacnejší $\Leftrightarrow$ neexistuje záporný rezervný polocyklus pre x.*

Dôkaz: ($\Rightarrow$) Zrejmé.

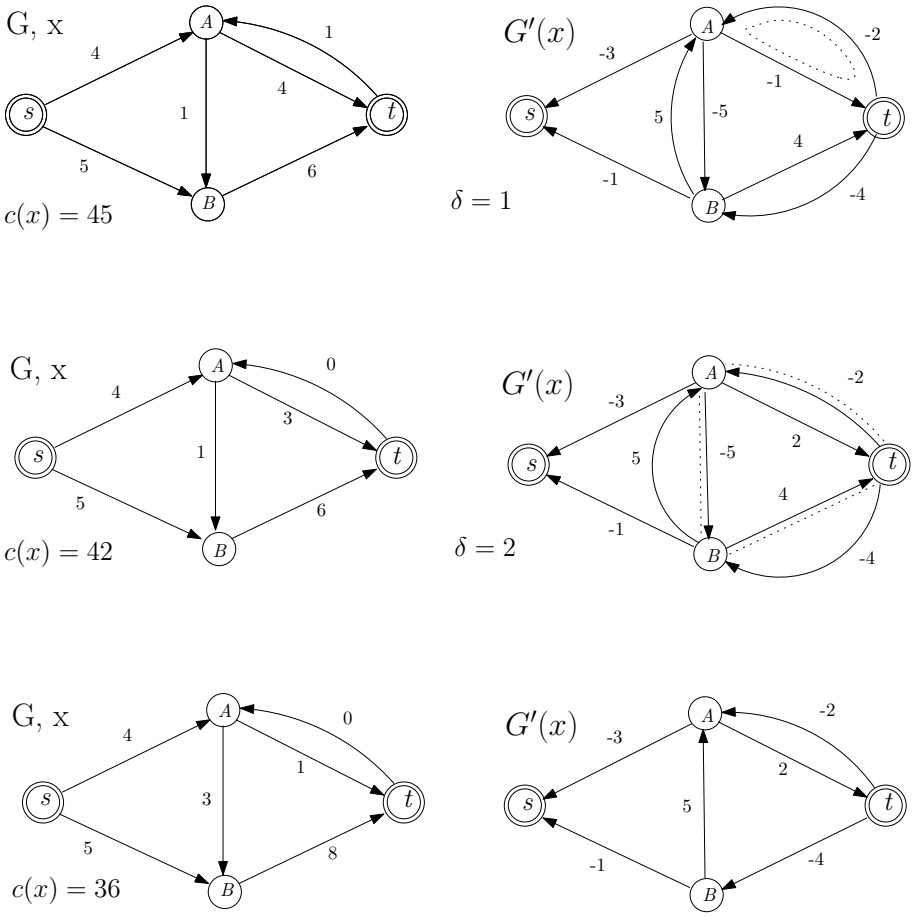
($\Leftarrow$) Pre spor predpokladajme, že tok x nie je najlacnejší a teda $c(x) > c(x^*)$, kde x^* je najlacnejší maximálny s-t tok. Potom vektor $y = x^* - x$ v uvažovanom digrafe G spĺňa podmienky kontinuity a má veľkosť 0, ale nie je vo všeobecnosti tokom (lebo tam kde $x^*_{ij} < x_{ij}$ tečie proti šípke). Avšak ak pre nenulové zložky vektora y definujeme nový vektor y' a nový vektor cien c' podľa obr. 5.26, tak tento nový vektor už bude v rezervnom (multi)digrafe $G'(x)$ tokom, (myslíme si dostatočne veľké kapacity). Samozrejme, jeho veľkosť zostane nulová, t.j. bude cirkuláciou a platí:

$$c'(y') = \sum_{(i,j)\in G'(x)} c'_{ij} y'_{ij} = \sum_{(i,j)\in G} c_{ij} y_{ij} = c(y) = c(x^*) - c(x) < 0$$

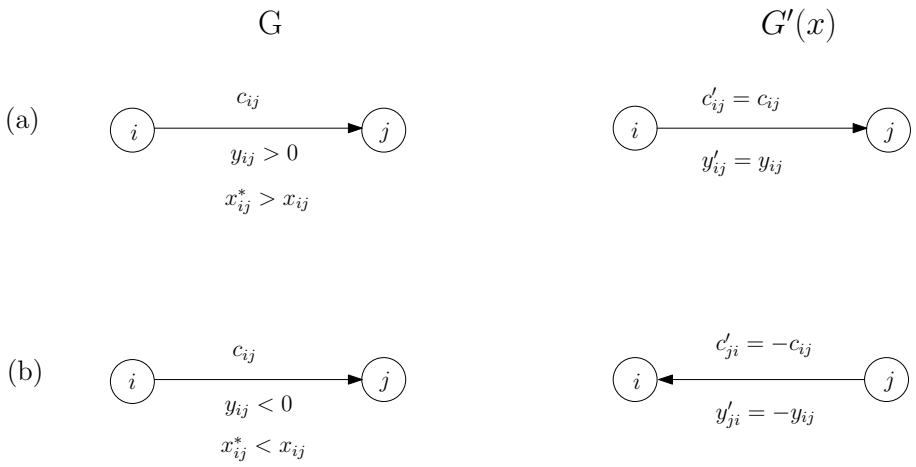
Keďže cirkuláciu y' možno rozložiť na cykly a súčet nových cien všetkých cyklov máme záporný, tak v $G'(x)$ existuje cyklus Q' s $c'(Q') < 0$. Tomu zodpovedá rezervný polocyklus Q pre x v G s cenou $c(Q) = c'(Q') < 0$, čo je spor s predpokladom. ■

Pri celočíselných kapacitách Kleinov algoritmus dáva v každej iterácii celočíselný tok a teda aj najlacnejší maximálny tok, ktorý získame, bude celočíselný.

Poznamenajme, že Kleinov algoritmus záporných polocyklov nie je polynomiálny, ale optimálnou voľbou polocyklov sa z neho taký algoritmus podarilo urobiť (vždy berieme taký polocyklus, ktorého cena delená počtom hrán polocyklu je minimálna, t.j. polocyklus s minimálnou priemernou cenou hrany). Pre nájdenie najlacnejšieho maximálneho toku sa podarilo vyvinúť aj silne polynomiálne algoritmy, ale ideove sú komplikovanejšie.



Obr. 5.25: Postup riešenia úlohy z obr. 5.23

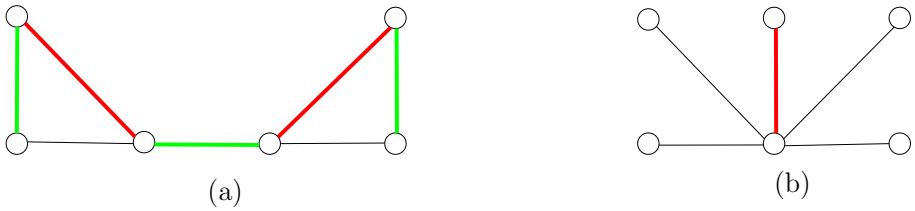


Obr. 5.26: Ilustrácia k dôkazu vety 5.9

Kapitola 6

PÁRENIA

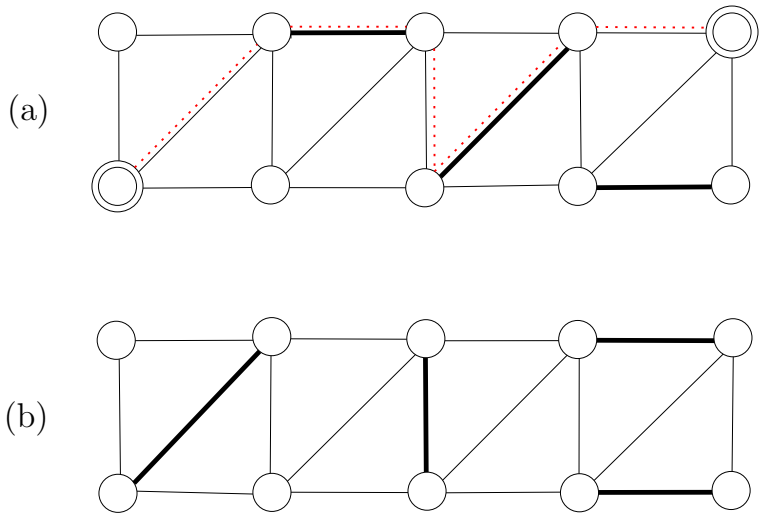
Každú hranu grafu možno chápať ako neusporiadanú dvojicu (pár) jej koncových vrcholov. **Párenie** v grafe je ľubovoľná podmnožina hrán grafu, z ktorých každé dve rôzne hrany sú nezávislé, t.j. bez spoločného vrcholu. Takto párenie predstavuje množinu párov vrcholov. V praxi často vytvárame dvojice (napr. manželské páry, alebo dvojice pracovníkov, dvojčlenné vojenské hliadky a pod.). Párenie sa nazýva **maximálne** (v zmysle inklúzie), ak ho nemožno rozšíriť pridaním ďalšej hrany grafu. **Najpočetnejšie** párenie má maximálny počet prvkov zo všetkých možných párení. **Perfektné** párenie pokrýva všetky vrcholy grafu, t.j. má presne $n/2$ hrán. Obr. 6.1 ilustruje tieto pojmy, kde aj vidíme, že maximálne párenie nemusí byť najpočetnejšie a najpočetnejšie nemusí byť perfektné. Zrejme však platí, že každé perfektné párenie je najpočetnejšie a každé najpočetnejšie je maximálne. Vzhľadom k danému pareniu M sa hrana grafu nazýva páriaca ak patrí do M a nepáriaca, ak nepatrí do M.



Obr. 6.1: Párenia v grafoch: (a) maximálne (červené) a perfektné (zelené), (b) najpočetnejšie (červené).

Jednou zo základných úloh o pareniach je úloha v danom grafe nájsť najpočetnejšie párenie. Pri riešení tejto úlohy sa ukázal dôležitý pojem alternujúcej cesty pre dané párenie M, t.j. cesty, v ktorej sa striedajú páriace a nepáriace hrany. Ak takáto cesta spája dva rôzne voľné vrcholy, t.j. vrcholy nepokryté párením M, tak sa nazýva **zväčšujúca alternujúca cesta pre** M. Totiž vtedy možno na tejto ceste vzájomne vymeniť páriace a nepáriace hrany a získať nové

párenie M', ktoré bude mať o 1 hranu viac ako má M. Obr. 6.2 ilustruje túto operáciu.

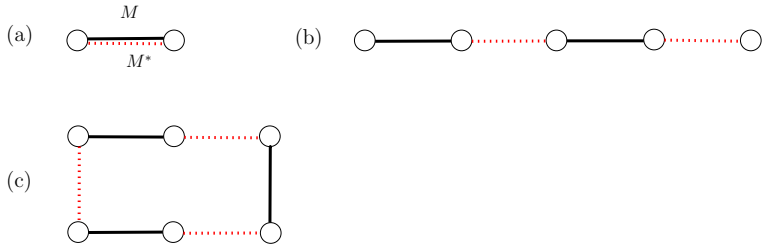


Obr. 6.2: Zväčšenie párenia pomocou zväčšujúcej alternujúcej cesty: (a) Párenie (hrubé čiary) a alternujúca cesta (červená bodkovaná). (b) Nové párenie (hrubé čiary).

VETA 6.1. *Párenie M je najpočetnejšie $\Leftrightarrow$ neexistuje zväčšujúca alternujúca cesta pre M.*

Dôkaz: ($\Rightarrow$) Zrejmé. ($\Leftarrow$) Nech M^* je najpočetnejšie párenie. Dokážeme, že M má taký istý počet prvkov ako M^*. Uvažujme podgraf H tvorený všetkými hranami patriacimi do $M \cup M^*$ a príslušnými incidentnými vrcholmi. Ľahko je vidieť, že každý komponent grafu H má jeden z tvarov na obr. 6.3 kde hrany patriace do M sú čierne a patriace do M^* sú červené bodkované: (a) hrana patriaca do M aj do M^*, (b) alternujúca cesta začínajúca hranou z M a končiaca hranou z M^* (alebo obrátene), (c) alternujúci cyklus párnej dĺžky. Totiž, alternujúca cesta začínajúca aj končiaca hranou z M by bola zväčšujúca pre M^* a alternujúca cesta začínajúca a končiaca hranou z M^* by bola zväčšujúca pre M. Teda v každom komponente majú obe párenia rovnaký počet prvkov. ■

Vzniká prirodzená otázka ako nájsť zväčšujúcu alternujúcu cestu. Hneď z definície sa ponúka postupovať tak, že začneme vo voľnom vrchole a cez nepáriace hrany postúpime do susedných vrcholov. Potom z týchto cez páriace hrany do ďalších vrcholov, z nich cez nepáriace hrany do ďalších, atď. Ak nájdeme voľný vrchol, tak spätne zistíme zväčšujúcu alternujúcu cestu. Tomuto postupu hovoríme, že budujeme alternujúci strom. Všobecnejšie môžeme budovať alternujúci les tak, že začneme z nejakej podmnožiny R voľných vrcholov, tzv. koreňov.

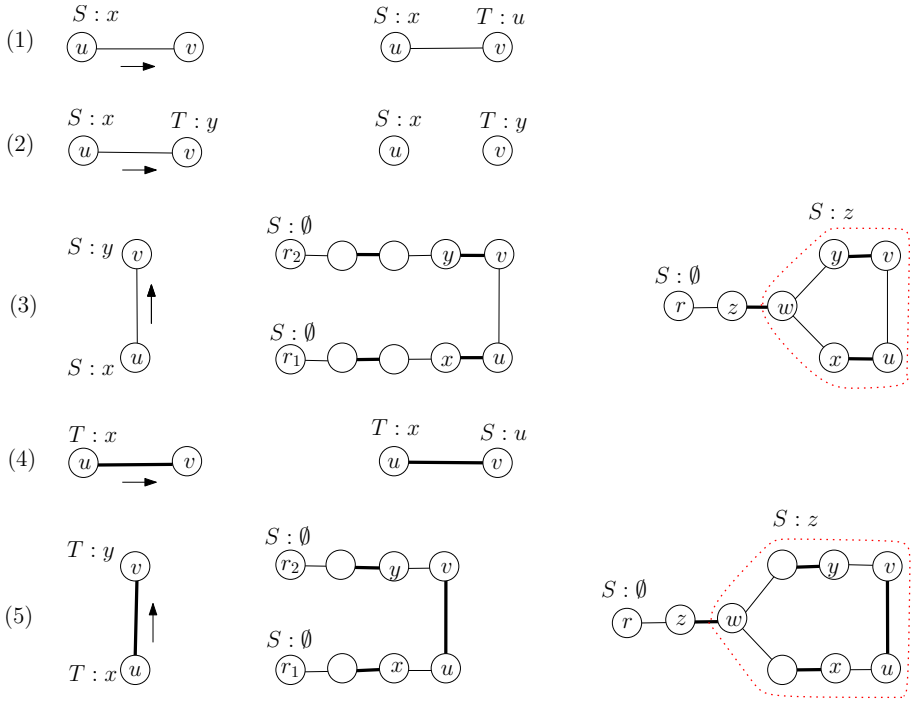


Obr. 6.3: Možné typy komponentov zjednotenia párení M a M^*.

Obvykle za R berieme buď jeden voľný vrchol, alebo všetky voľné vrcholy (prípadne v bipartitnom grafe všetky voľné vrcholy v jednej partícii). Podľa toho postupujeme a robíme závery. Tu predpokladáme, že za množinu koreňov berieme množinu všetkých voľných vcholov, avšak aj v iných prípadoch je postup podobný. Pre riadenie procesu a identifikáciu cesty používame značkovanie vrcholov. Pritom značky nemeníme. Je vhodné používať postup do šírky. Korene lesa (vrcholy množiny R) dostanú značku „$S : \emptyset$“. Potom vrcholom priraďujeme značky v súlade s obr. 6.4, kde šípka vedľa hrany označuje postup skúmania hrany: (1) Z S-vrchola u (stručné pomenovanie vrchola so značkou „$S : x$“) cez nepáriace hrany označkujeme susedné vrcholy; každý takýto neoznačkovaný vrchol v dostane značku „$T : u$“. To vyjadruje fakt, že existuje alternujúca cesta z koreňa do v, ktorej predposledný vrchol je u. (2) Ak susedný vrchol v už je T-vrchol, tak jeho značku nemeníme a hranu (u, v) do lesa nezoberieme. (3) Ak vrchol v už je S-vrchol, tak zistíme korene vrcholov u a v. Ak sú ich korene r_1 a r_2 rôzne, tak zreťazenie alternujúcej r_1-u cesty, hrany (u, v) a alternujúcej v-r_2 cesty dáva alternujúcu cestu medzi nimi, a teda zväčšujúcu alternujúcu cestu. Ak majú spoločný koreň r, takéto zreťazenie dáva podgraf pozostávajúci z alternujúcej r-b cesty, tzv. stopky kvetu (môže mať aj nulovú dĺžku) a nepárneho alternujúceho b-b cyklu, tzv. kvetu B (vždy má dĺžku aspoň 3). Tieto majú spoločný jediný vrchol b, ktorý sa nazýva báza kvetu. V tomto prípade stiahneme kvet na jediný vrchol (tzv. pseudovrchol) B' a dáme mu značku jeho bázy (teda „$S : z$“) a v novom grafe pokračujeme v značkovaní. (4) Z T-vrchola u cez páriacu hranu označkujeme jej druhý neoznačkovaný koniec v značkou „$S : u$“. (5) Ak v je už T-vrchol, tak podobne ako v prípade (3) dostaneme buď zväčšujúcu alternujúcu cestu, alebo kvet, ktorý stiahneme na jediný pseudovrchol a pokračujeme v značkovaní v novom grafe.

Na obr. 6.5 je príklad značkovania.

V každom kroku budovania alternujúceho lesa vyšetríme jednu hranu, ktorú buď zaradíme do lesa a vtedy označkujeme jej druhý koniec, alebo ju nezaradíme (a nič neoznačkujeme). V prípade nájdenia zväčšujúcej alternujúcej cesty, zväčšíme pôvodné párenie, všetky značky zotrieme a začneme budovať nový alternujúci les. Ešte zostáva možnosť, že značkovanie je maximálne, t.j. už ho nemožno zväčšiť (všetko, čo sa dalo, sme označkovali) a pritom sa zväčšujúca alternujúca cesta nenašla. V tomto prípade sa vybudovaný alternujúci les na-



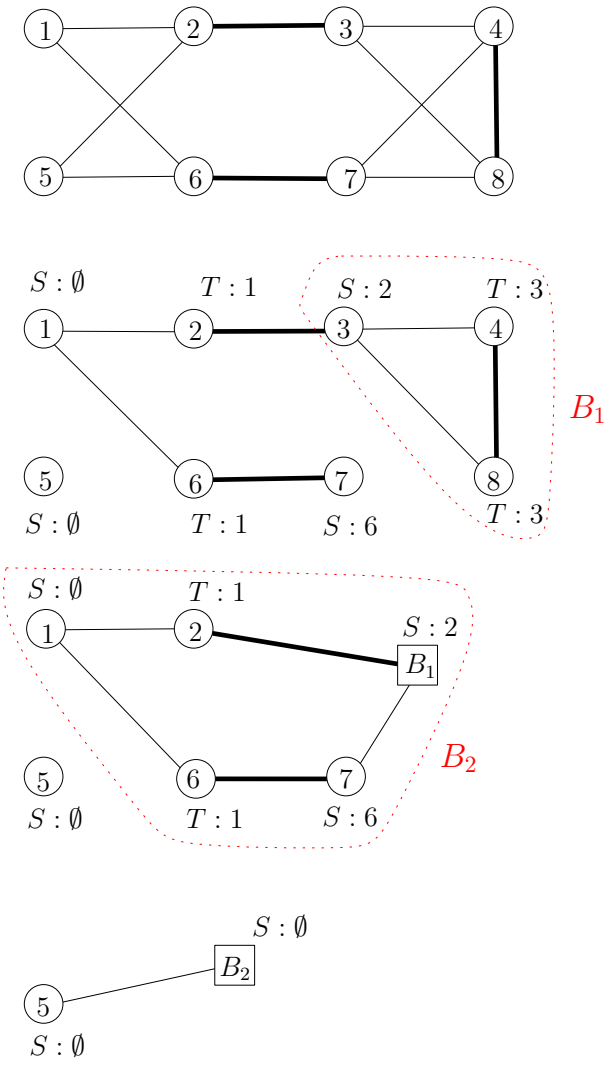
Obr. 6.4: Možné situácie pri budovaní alternujúceho lesa; vľavo je šípkou označená skúmaná hrana a vpravo je výsledok, resp. možnosti.

zýva **maďarský**. Z pravidiel značkovania vyplýva, že v prípade maďarského lesa ani žiadna zväčšujúca alternujúca cesta v priebežnom grafe neexistuje. Z nasledujúcej vety potom vyplýva, že ani v pôvodnom grafe taká cesta neexistuje.

VETA 6.2. *Nech v grafe G s párením M objavíme pri značkovaní kvet B a nech po kontrakcii kvetu B na pseudovrchol B' získame graf G' a párenie M'. Potom v G existuje zväčšujúca alternujúca cesta pre $M \Leftrightarrow$ v G' existuje zväčšujúca alternujúca cesta pre M'.*

Dôkaz: ($\Rightarrow$) Nech P je zväčšujúca alternujúca r_1-r_2 cesta v G. Ak neprechádza kvetom B, tak bude existovať aj v novom grafe G' a niet čo dokazovať. Inak dosť dlhá analýza prípadov (ako môže cesta P prechádzať kvetom B a jeho stopkou) dáva v novom grafe G' zväčšujúcu alternujúcu cestu P' pre M'. Podrobnosti vynechávame. Poznamenajme, že vo všeobecnosti v novom grafe G' nemusí existovať zväčšujúca alternujúca cesta s tými istými koncami (koreňmi) ako zväčšujúca alternujúca cesta v grafe G (obr. 6.6).

($\Leftarrow$) Nech v grafe G' je P' zväčšujúca alternujúca r_1-r_2 cesta pre M'. Stačí vyšetriť prípad keď obsahuje pseudovrchol B'. V G utvoríme požadovanú r_1-r_2 cestu tak, že budeme sledovať cestu P' až po kvet B, potom ideme vhodnou

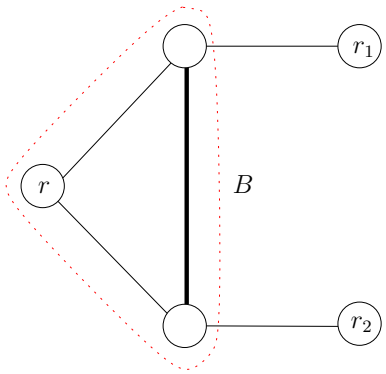


Obr. 6.5: Príklad značkovania.

časťou kvetu až do vrcholu, v ktorom kvet opustíme a úsekom cesty P' prídeme do vrcholu r_2. Z vlastností kvetu vidíme, že získaná r_1-r_2 cesta P v grafe G bude zväčšujúca alternujúca cesta pre M. ■

V uvedenom algoritme sa zväčšuje párenie $O(n)$-krát. Zväčšenie párenia a nájdenie zväčšujúcej alternujúcej cesty možno urobiť v čase $O(n^2)$ ak sa dômyselne manipuluje s kvetmi. To potom celkove dáva algoritmus zložitosti $O(n^3)$.

Poznamenajme, že v bipartitnom grafe sa algoritmus zjednoduší, lebo nevytvárame kvety.



Obr. 6.6: V grafe G existuje zväčšujúca alternujúca r_1-r_2 cesta, ale v G' nie.

Cvičenie 6.1. Ukážte, že v tomto prípade možno nájsť najpočetnejšie párenie aj pomocou maximálneho toku.

.

Vrcholové pokrytie grafu je taká podmnožina jeho vrcholov, ktorá pokrýva všetky hrany grafu. Vo všeobecnosti každé vrcholové pokrytie musí mať aspoň toľko prvkov ako je počet hrán v ľubovoľnom párení (lebo každý vrchol môže pokryť nanajvýš jednu páriacu hranu). Zaujímavejšie je to v prípade bipartitného grafu:

VETA 6.3. *(König) V bipartitnom grafe sa minimálny počet prvkov vrcholového pokrytia rovná maximálnemu počtu prvkov párenia.*

Cvičenie 6.2. Dokážte túto vetu a navrhnite algoritmus pre nájdenie vrcholového pokrytia minimálnej kardinality v bipartitnom grafe. [Návod: Pridajte ku grafu 2 vrcholy s a t, vhodné hrany a zorientujte smerom od s ku t. V tomto digrafe aplikujte vrcholovú verziu Mengerovej vety 5.7.]

Königova veta súvisí s nasledujúcim tvrdením, kde $N(A)$ označuje susednú množinu pre A, t.j. $N(A) = \{y|\exists(x,y)x \in A\}$.

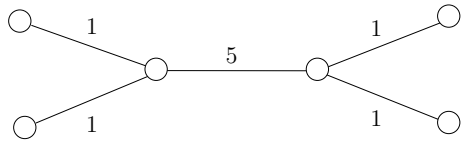
VETA 6.4. *(Hall) V bipartitnom grafe s partiami S a T možno popáriť S do $T \Leftrightarrow$ pre každú množinu $A \subseteq S$ platí $|A| \leq |N(A)|$.*

Cvičenie 6.3. Dajte dôkaz uvedenej vety. [Návod: Použite alternujúce cesty, resp. Königovu vetu 6.3.]

6.1 Najcennejšie párenie

V praktických aplikáciach sa často vyskytujú hrany s cenami (ľubovoľné reálne čísla) a namiesto najpočetnejšieho párenia treba nájsť najcennejšie párenie (t.j.

párenie s maximálnou sumárnou cenou). Je zrejmé,že ak má každá hrana rovnakú kladnú cenu, tak najcennejšie párenie je aj najpočetnejšie. Pri nerovnakých cenách to tak nemusí byť ako ukazuje príklad z obr. 6.7.



Obr. 6.7: Najcennejšie párenie nie je najpočetnejšie.

6.1.1 Rôzne úlohy

Často sa vyskytujú aj iné formulácie úlohy najcennejšieho párenia. Uvedieme niektoré z nich a ukážeme ich ekvivalenciu. Nech $\Re$ a $\Re_{++}$ príslušne označujú množiny všetkých reálnych a kladných reálnych čísel. Uvedenú úlohu najcennejšieho párenia môžeme sformulovať takto.

(P1) Daný je graf $G_1 = (V_1, E_1)$ spolu s cenovou funkciou $c_1 : E_1 \mapsto \Re$. Treba nájsť najcennejšie (nie nutne perfektné) párenie M_1 grafu G_1.

Ešte sformulujeme ďalšie 3 úlohy.

(P2) Daný je graf $G_2 = (V_2, E_2)$ spolu s cenovou funkciou $c_2 : E_2 \mapsto \Re$. Treba nájsť najlacnejšie (nie nutne perfektné) párenie M_2 grafu G_2.

(P3) Daný je kompletný graf $G_3 = (V_3, E_3)$, kde $|V_3|$ je párne, spolu s cenovou funkciou $c_3 : E_3 \mapsto \Re$. Treba nájsť najlacnejšie perfektné párenie M_3 grafu G_3.

(P4) Daný je kompletný graf $G_4 = (V_4, E_4)$, kde $|V_4|$ je párne, spolu s kladnou cenovou funkciou $c_4 : E_4 \mapsto \Re_{++}$ splňajúcou prísnu trojuholníkovú nerovnosť $(c_4(i,k) < c_4(i,j) + c_4(j,k))$. Treba nájsť najlacnejšie perfektné párenie M_4 grafu G_4.

VETA 6.5. *Uvedené 4 problémy sú silne polynomiálne ekvivalentné.*

Dôkaz: Uvedieme potrebné redukcie a prenechávame čitateľovi overiť, že sú korektné.

(P1) $\rightarrow$ (P2): Stačí položiť $c_2 = -c_1$.

(P2) $\rightarrow$ (P3): Ak graf G_2 má nepárny rád, tak najprv pridáme jeden izolovaný vrchol, aby sme mali párny počet vrcholov. Doplníme graf na kompletný graf G_3 novými hranami veľkej ceny L.

(P3) $\rightarrow$ (P4): Zdvihneme cenu každej hrany o jednotnú veľkú konštantu B.

(P4) $\rightarrow$ (P1): Pre každú hranu i položíme $c_1(i) = Q - c_4(i)$, kde Q je jednotná veľká konštanta.

■

6.2 Hľadanie najcennejšieho párenia

Zjednotením hrán dvoch párení dostávame podgraf, ktorého každý komponent je alebo cesta, alebo párny cyklus. To vedie k pojmom *zväčšujúca alternujúca cesta* a *zväčšujúci alternujúci cyklus* pre párenie s cenami hrán. Tiež platí podobné tvrdenie ako veta 6.1: Párenie M je najcennejšie $\Leftrightarrow$ neexistuje zväčšujúca alternujúca cesta alebo cyklus pre M.

Cvičenie 6.4. Dokážte to.

Keďže takýto prístup neviedol k efektívnym algoritmom, hľadali sa iné. Tie sú založené na lineárnom programovaní a nie sú jednoduché. Na ilustráciu tu uvedieme špeciálny prípad keď graf je bipartitný. Ten súvisí s tzv. priraďovacím problémom, kde treba nájsť najcennejšie párenie v kompletnom bipartitnom grafe s rovnako veľkými partíciami.

Cvičenie 6.5. Sformulujte a dokážte pre bipartitné grafy analógiu vety 6.5.

6.2.1 Klasický priraďovací problém

V grafovej reči je daný kompletný bipartitný graf s rovnakými partiami, ktorého každá hrana má cenu a úlohou je nájsť najcennejšie perfektné párenie, t.j. s maximálnym súčtom cien. Pre metódu, ktorú uvádzame ďalej je vhodnejšie uvažovať najlacnejšie párenie. Tradične sa kompletný bipartitný graf $K_{p,p}$ reprezentuje maticou typu (p,p) kde riadky zodpovedajú jednej partii a stĺpce druhej. Potom perfektné párenie zodpovedá permutačnej matici rovnakého rozmeru. Inými slovami, v zadanej matici cien treba vybrať p nezávislých elementov, t.j. tak, aby sa v každom riadku a tiež v každom stĺpci nachádzal práve jeden vybraný prvok. Typickou aplikáciou je priradiť p pracovníkov k p strojom, kde i-ty pracovník môže pracovať na j-tom stroji za mzdu c_{ij} a chceme minimalizovať sučet miezd. Tu načrtneme základnú myšlienku tzv. **maďarskej metódy**, ktorú vyvinul Kuhn a na počesť maďarských matematikov Königa a Egerváryho ju takto pomenoval. Königa sme už spomínali (veta 6.3). Egerváry sa zaoberal rozšírením Königovej vety pre kompletné bipartitné grafy s nezápornými cenami hrán (teda uvažoval priraďovací problém) a v skutočnosti pracoval s maticami, čo je ekvivalentný postup. V jeho dôkaze je implicitne obsiahnutý istý algoritmus, ktorý Kuhn zlepšil na efektívnu procedúru. V tejto časti uvedieme jeden populárny postup, ktorý ešte neobsahuje všetky Kuhnove zlepšenia (tie uvedieme až v ďalšej časti 6.2.2). Tento postup neobsahuje explicitne lineárne programovanie a možno ho nazvať **elementárna maďarská metóda**. Ilustrujeme ho na nasledujúcom príklade, ktorý je zadaný maticou cien $C = \|c_{ij}\|$ $(p = 5)$ umiestnenou v 1. rámčeku:

−1	0	−3	−2	4
6	10	4	7	7
−1	11	10	9	4
7	3	2	4	−2
5	6	1	7	4

2	3	0	1	7
2	6	0	3	3
0	12	11	10	5
9	5	4	6	0
4	5	0	6	3

			•		
•	2	0	0	0	7
	2	3	0	2	3
•	0	9	11	9	5
•	9	2	4	5	0
	4	2	0	5	3

Základná myšlienka metódy je veľmi jednoduchá a je vyjadrená nasledujúcou triviálnou lemou.

Lema 6.6. *Ak sú všetky $c_{ij} \geq 0$ a z nulových elementov možno vybrať perfektné párenie, tak toto párenie je najlacnejšie.*

Predpoklady tejto lemy (postupne) zabezpečíme aplikovaním nasledujúcej, tiež triviálnej, lemy.

Lema 6.7. *(modifikačná lema) Ak ku všetkým elementom nejakého riadku alebo stĺpca matice C pripočítame to isté reálne číslo, tak množina optimálnych riešení sa nezmení.*

Túto lemu sme aplikovali na 1. maticu tak, že sme našli minimálny prvok v riadku a ten sme odčítali od každého prvku tohoto riadku. Výsledkom je 2. matica, ktorá už má všetky prvky nezáporné a v každom riadku aspoň jednu nulu. Aby sme dosiahli nulu aj v každom stĺpci, tak aplikovaním stĺpcovej formy modifikačnej lemy na 2. maticu sme dostali 3. maticu, ktorá už má požadovanú vlastnosť. Hoci táto matica už má hodne núl, ešte stále z nich nemožno vybrať perfektné párenie.

Pre zlepšenie situácie sa ešte pokúsime zmeniť aj poslednú maticu. Na nulových prvkoch má najpočetnejšie párenie 4 elementy, a teda podľa Königovej vety 6.3 minimálny počet línií (riadkov a stĺpcov), ktoré pokrývajú všetky nuly matice je tiež 4 (3 riadkové a 1 stĺpcová línia) a sú vyznačené čiernymi krúžkami. Nech μ je minimálny (týmito líniami) nepokrytý element; u nás máme $\mu = 2$. Urobme nasledujúcu, tzv. *maďarskú, operáciu* na 3. matici: (1) každý nepokrytý prvok zmenšime o μ a (2) každý prvok pokrytý 2-ma líniami zväčšime o μ. (Prvky pokryté jednou líniou zostanú nezmenené.)

Cvičenie 6.6. Ukážte, že túto operáciu možno zrealizovať aj v zmysle modifikačnej lemy.

Takto získame 4. maticu, ktorá bude mať tie isté optimálne riešenia ako pôvodná matica, ale aspoň o jeden nulový prvok viac na nepokrytom mieste (hoci dvojmo pokrytý nulový prvok stratíme). Očakávame, že opakovaním tejto operácie nakoniec získame maticu s najpočetnejším párením kardinality p. U nás už 4. matica má túto vlastnosť: 5 nezávislých núl sme vyznačili krúžkami a minimálny počet línií pokrývajúcich všetky nuly je samozrejme tiež 5 (môžeme zobrať 5 riadkových línií, alebo 5 stĺpcových línií, alebo tie vyznačené čiernymi krúžkami). Spočítaním cien v 1. matici na zakrúžkovaných pozíciach dostaneme cenu najlacnejšieho perfektného párenia: $5{=}p$. Progres tejto operácie je obsahom nasledujúcej lemy.

Lema 6.8. *Nech $k < p$ je počet línií pokrývajúcich všetky nulové prvky (p, p) matice. Potom aplikovaním maďarskej operácie sa súčet všetkých prvkov matice zmenší o $\mu p(p - k)$.*

Dôkaz: Nech pokrývajúce línie pozostávajú z r riadkových a s stĺpcových línií (teda $k = r + s$). Takto pri operácii sa $(n - r)(n - s)$ prvkov zmenší a rs prvkov zväčší. Ostatné sa nezmenia. Sumárne sa súčet všetkých prvkov matice zmenší o $\mu[(p - r)(p - s) - rs] = \mu p[p - (r + s)] = \mu p(p - k)$.

■

		•			•
•	2	Ⓞ	2	0	7
•	0	1	0	Ⓞ	1
	Ⓞ	9	13	9	5
	9	2	6	5	Ⓞ
•	2	0	Ⓞ	3	1

Z lemy vidíme, že pri celočíselných, alebo aj racionálnych prvkoch cenovej matice, je zaručená konečnosť algoritmu elementárnej maďarskej metódy. Ďalej vidíme, že netreba hľadať minimálny počet pokrývajúcich línií, ale stačí aby bol menší ako rozmer matice, čo môže byť ľahšie. Pravda, minimálne vrcholové pokrytie sa zvyčajne nájde ako vedľajší produkt pri hľadaní najpočetnejšieho párenia, ako naznačuje Königova veta.

Cvičenie 6.7. Ukážte to podrobnejšie

.

Uvedenou metódou vieme vyriešiť aj všeobecnejšiu úlohu tak, že ju prevedieme na úlohu v kompletnom bipartitnom grafe s rovnakými partíciami. Avšak, riedke úlohy takýmto spôsobom zbytočne zahusťujeme, a preto ukážeme postup, ktorý rieši takúto všeobecnú úlohu priamo. Naviac, taký spôsob je efektívnejší (ukážeme, že je to silne polynomiálny algoritmus).

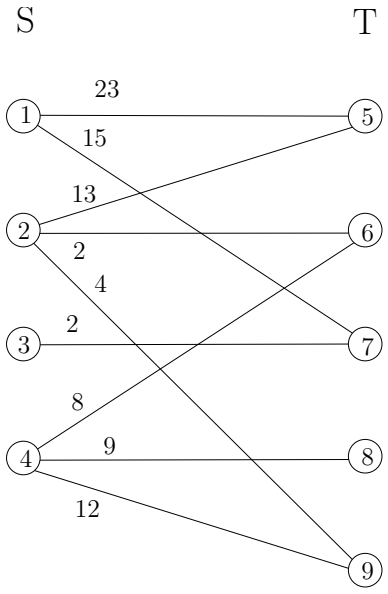
6.2.2 Maďarská metóda pre najcennejšie párenie v bipartitnom grafe

Tu uvádzame úplnú maďarskú metódu pre nasledujúcu všeobecnú úlohu. Daný je graf $G = (S, T, E)$ s partíciami S a T a reálnymi cenami hrán. Budeme predpokladať, že $|S| = p \leq q = |T|$. V skutočnosti môžeme všetky hrany s nekladnými cenami vynechať, a preto budeme predpokladať, že každá hrana (i, j) má cenu $c_{ij} > 0$. Metódu budeme ilustrovať na príklade z obr. 6.8, kde máme $S = \{1, 2, 3, 4\}, T = \{5, 6, 7, 8, 9\}$.

Najprv úlohu sformulujeme ako úlohu binárneho lineárneho programovania. Ku každej hrane $(i, j) \in E$ priradíme premennú $x_{ij} \in \{0, 1\}$, pričom chceme, aby $x_{ij} = 1 \Leftrightarrow$ hrana (i, j) leží v hľadanom párení. Symbol $N(i)$ vyjadruje množinu všetkých susedných vrcholov vrcholu i.

$$
\begin{array}{lrcll}
\text{(U)} & \sum\limits_{(i,j)\in E} c_{ij}x_{ij} & \rightarrow & \max & \\
& \sum\limits_{j\in N(i)} x_{ij} & \leq & 1 & \forall i \in S \\
& \sum\limits_{i\in N(j)} x_{ij} & \leq & 1 & \forall j \in T \\
& x_{ij} & \in & \{0,1\} & \forall (i,j) \in E
\end{array}
$$

K úlohe (U) priradíme úlohu lineárneho programovania (P) tak, že podmienku $x_{ij} \in \{0, 1\}$ nahradíme podmienkou $0 \leq x_{ij} \leq 1$ (v skutočnosti napíšeme iba prvú nerovnosť, lebo druhá je implicitne obsiahnutá v iných obmedzeniach).



Obr. 6.8: Príklad pre maďarskú metódu.

$$
\begin{array}{lrcll}
\text{(P)} & \sum\limits_{(i,j)\in E} c_{ij}x_{ij} & \to & \max & \\
& \sum\limits_{j\in N(i)} x_{ij} & \leq & 1 & \forall i \in S \\
& \sum\limits_{i\in N(j)} x_{ij} & \leq & 1 & \forall j \in T \\
& x_{ij} & \geq & 0 & \forall (i,j) \in E
\end{array}
$$

Množina prípustných riešení úlohy (P) je neprázdna (obsahuje nulový vektor) a ohraničená. Preto úloha má optimálne riešenie.

VETA 6.9. *Ak je nejaké optimálne riešenie úlohy (P) celočíselné, tak je optimálnym riešením úlohy (U).*

Dôkaz: Dôkaz je zrejmý. ■

Poznamenajme, že z lineárneho programovania (z vlastnosti dopravnej úlohy) vieme, že úloha (P) má aj také optimálne riešenie, ktoré je celočíselné. Prevod úlohy (P) na dopravnú poskytuje jeden spôsob riešenia úlohy (U).

Cvičenie 6.8. Uveďte podrobnosti.

Tu však pôjdeme inou cestou. K úlohe (P) zostrojme duálnu:

$$\begin{array}{lrcl} \text{(D)} & \sum_{i\in S} u_i + \sum_{j\in T} v_j & \to & \min \\ & u_i + v_j & \geq & c_{ij} \quad \forall (i,j) \in E \\ & u_i & \geq & 0 \quad \forall i \in S \\ & v_j & \geq & 0 \quad \forall j \in T \end{array}$$

Z lineárneho programovania vieme, že tzv. podmienky komplementarity sú nutné a postačujúce pre optimalitu prípustných riešení primárnej a duálnej úlohy. Pre naše úlohy (P) a (D) sú to nasledujúce podmienky.

$$\text{(K1)} \qquad (\sum_{j\in N(i)} x_{ij} - 1)u_i = 0 \quad \forall i \in S$$

$$\text{(K2)} \qquad (\sum_{i\in N(j)} x_{ij} - 1)v_j = 0 \quad \forall j \in T$$

$$\text{(K3)} \qquad (u_i + v_j - c_{ij})x_{ij} = 0 \quad \forall (i,j) \in E$$

Aby sme splnili predpoklad vety 6.9, tak nájdeme primárne prípustné $x_{ij} \in \{0,1\}$ a duálne prípustné u_i a v_j, ktoré budú spĺňať podmienky komplementarity. Hľadanie takých hodnôt je iteračný proces, v ktorom priebežné hodnoty budú spĺňať primárnu a duálnu prípustnosť, všetky podmienky (K2), (K3) a postupne viac podmienok (K1).

Na začiatku stačí zistiť maximálnu cenu $c_{\max}$ hrany v grafe a položiť:

$$\begin{array}{rcl} x_{ij} & = & 0 \quad \forall (i,j) \in E \\ u_i & = & c_{\max} \quad \forall i \in S \\ v_j & = & 0 \quad \forall j \in T \end{array} \tag{6.1}$$

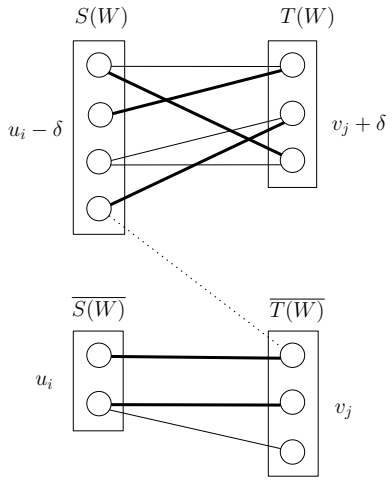
Teda začíname s prázdnym párením a žiadna podmienka (K1) nie je splnená. V iteráciach sa snažíme splniť viac podmienok (K1) zmenou párenia pomocou zväčšujúcej alternujúcej cesty v tzv. *rovnostnom grafe*, t.j. podgrafe grafu G pozostávajúcom zo všetkch vrcholov a tých hrán (i,j) kde platí rovnosť $u_i + v_j = c_{ij}$. Totiž, ak pre takúto hranu zmeníme x_{ij} z 0 na 1, tak sa (K3) dodrží. Taká alternujúca cesta začínajúca v S musí končiť v T (lebo sme v bipartitnom grafe). Ľahko sa overí primárna prípustnosť ako aj dodržanie (K2). Naviac, pre začiatok alternujúcej cesty sa splní aj (K1).

Hľadanie zväčšujúcej alternujúcej cesty robíme pomocou alternujúceho lesa, ktorého koreňovú množinu tvoria všetky voľné vrcholy množiny S. Ak narazíme na voľný vrchol množiny T, tak sme našli zväčšujúcu alternujúcu cestu, párenie zväčšíme, všetky značky zmažeme a ideme budovať nový alterujúci les. Ak alternujúci les W je maďarský (t.j. neoznačkujeme voľný vrchol množiny T, tak budeme robiť zmenu duálnych premnných nasledovne:

Nech $S(W)$ je množina všetkých S-vrcholov, t.j. vrcholov lesa W ležiacich v partícii S. Podobne nech $T(W)$ je množina všetkých T-vrcholov. Komplementárne množiny $\overline{S(W)} = S \setminus S(W)$ a $\overline{T(W)} = T \setminus T(W)$ sú príslušné množiny neoznačkovaných vrcholov. Vypočítame

$$\delta_1 = \min\{u_i | i \in S(W)\} \tag{6.2}$$
$$\delta_2 = min\{u_i + v_j - c_{ij} | i \in S(W), j \in \overline{T(W)}, (i,j) \in E\} \tag{6.3}$$
$$\delta = \min\{\delta_1, \delta_2\} \tag{6.4}$$

Neskôr uvidíme, že $\delta > 0$. Teraz zmeníme duálne premenné v súlade s obr. 6.9: $\forall i \in S(W)$ hodnoty u_i zmenšíme o δ a $\forall j \in T(W)$ hodnoty v_j zväčšíme o δ. Ostatné u_i a v_j nemeníme. Ak bolo $\delta = \delta_1$, tak (ako uvidíme) všetky u_i pri voľných S-vrcholoch budú mať hodnotu 0 a tým bude hlásená optimalita. Inak sa $\delta = \delta_2$ realizuje na nejakej hrane (i,j) kde $i \in S(W)$ a $j \in \overline{T(W)}$ (bodkovaná hrana na obr. 6.9). Táto hrana po zmene pribudne do rovnostného grafu a pokračujúc v značkovaní pribudne aj do lesa W. V značkovaní pokračujeme až pokiaľ nenájdeme zväčšujúcu alternujúcu cestu, alebo maďarský les.



Obr. 6.9: Maďarský les W a zmena duálnych premenných.

Lema 6.10. *Vždy platí: minimálna hodnota u_i sa realizuje na každom voľnom vrchole partície S a je nezáporná. Naviac, ak niektoré $u_i = 0$, tak už máme optimalitu.*

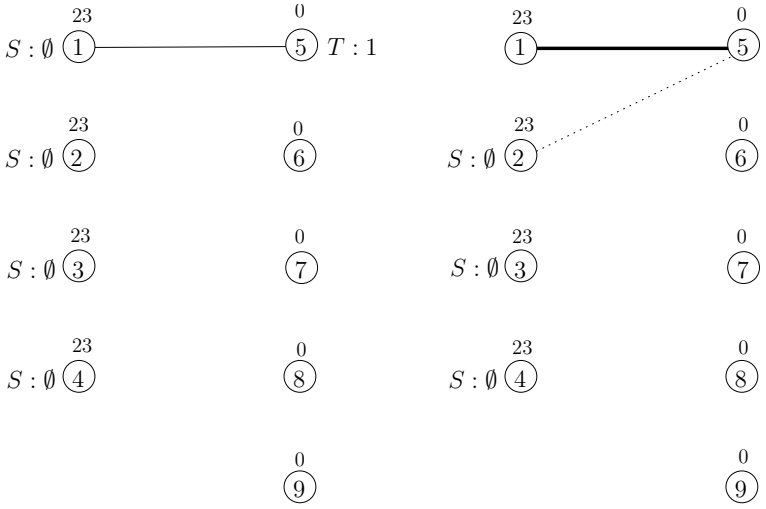
Dôkaz: Na začiatku všetky vrcholy v S boli voľné a mali rovnaké $u_i = c_{\max} > 0$, a teda to platilo. Pri zmene duálnych premnných sa u_i pri niektorých vrcholoch zmenšujú o to isté $\delta > 0$. Totiž pri optimalite už nezmenšujeme (preto $\delta_1 > 0$), hrana dávajúca δ_2 nie je rovnostná a platí duálna prípustnosť (preto $\delta_2 > 0$). Takto ak niektoré $u_i = 0$, tak aj všetky voľné vrcholy majú $u_i = 0$, a preto budú podmienky (K1) splnené. ■

Ľahko možno vidieť, že po zmene duálnych premenných budeme mať zachovanú primárnu a duálnu prípustnosť, doterajšie splnenie podmienok $(K1)$ a splnenie všetkých podmienok $(K2)$ a $(K3)$.

Cvičenie 6.9. Ukažte to.

Pokiaľ ide o rovnostný graf, niektoré hrany môžu po zmene z rovnostného grafu vypadnúť (hrany medzi $\overline{S(W)}$ a $T(W)$. To však nevadí. Dôležité je, že tam zostane každá hrana lesa W a tiež každá párіaca hrana. Pri značkovaní preto nezostrojujeme rovnostný graf dopredu, ale pri danom S-vrchole testujeme každú nepáriacu hranu grafu, či je rovnostná; pri T-vrchole hľadáme párіacu hranu.

Vyriešime príklad z obr. 6.8. Uvádzame len medzivýsledky: zväčšujúcu alternujúcu cestu a maďarský les. Nachádzajú sa na obrázkoch 6.10 až 6.14.

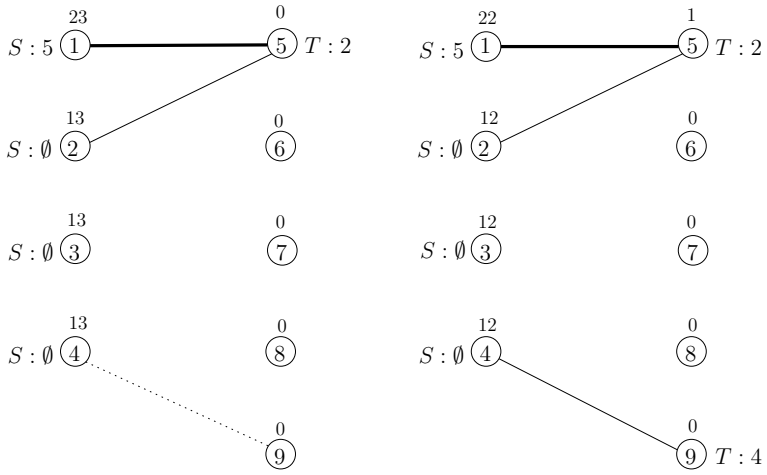


Obr. 6.10: Riešenie príkladu z obr. 6.8 maďarskou metódou. Prvé 2 medzivýsledky: zväčšujúca alternujúca cesta a maďarský les ($\delta = 10$).

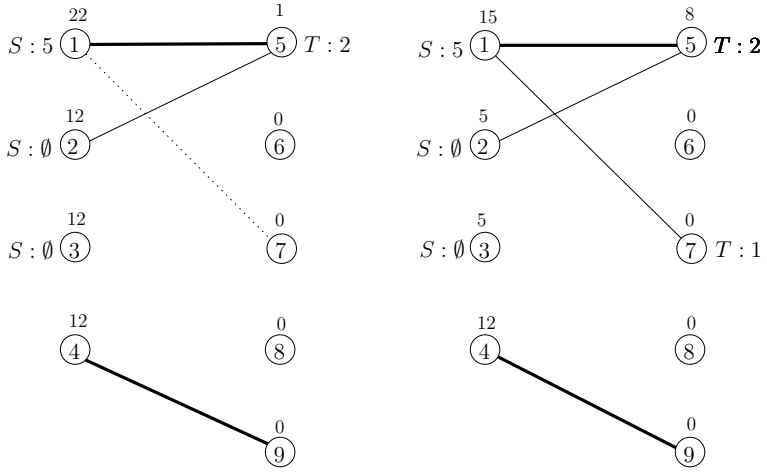
ZLOŽITOSŤ ALGORITMU

Predpokladali sme, že $p \leq q$, a preto párenie môže mať nanajvýš p hrán. Takto v celom algoritme urobíme nanajvýš p zväčšovaní párenia. Na každé zväčšovanie vybudujeme alternujúci les, ktorý má nanajvýš $p+q$ vrcholov a teda nanajvýš $p+q-1$ hrán. Preto na jeho vybudovanie vystačíme s $O(p+q)$ zmenami duálnych premenných, lebo pri každej zmene (okrem finálnej) sa $\delta = \delta_2$ a pridáme do lesa aspoň jednu hranu (na ktorej sa realizuje δ_2). Jedna zmena duálnych premenných sa dá urobiť v čase $O(pq)$ (dominuje hľadanie δ_2). Teda sumárne máme $O(p(p+q)pq) = O(p^2q^2) \leq O(n^4)$, kde $n = p+q$.

Túto zložitosť možno znížiť ak zavedieme pomocné údaje (podobne ako v Primovom algoritme pre najlacejšiu kostru), čím sa potom zjednoduší hľadanie δ_2. Každému vrcholu $j \in T$ priradíme dvojicu (d_j, i_j), kde d_j je minimálna hodnota výrazu $u_i + v_j - c_{ij}$ pre nepáriace hrany (i,j) a vrchol i_j je jeden z tých kde sa to realizuje. Na začiatku a tiež po každom zväčšení párenia položíme $d_j = \infty \quad \forall j \in T$. Pri skúmaní S-vrchola i robíme aktualizáciu. Pre každú nepáriacu hranu (i,j) urobíme: Ak $u_i + v_j - c_{ij} < d_j$, tak dáme $d_j = u_i + v_j - c_{ij}$



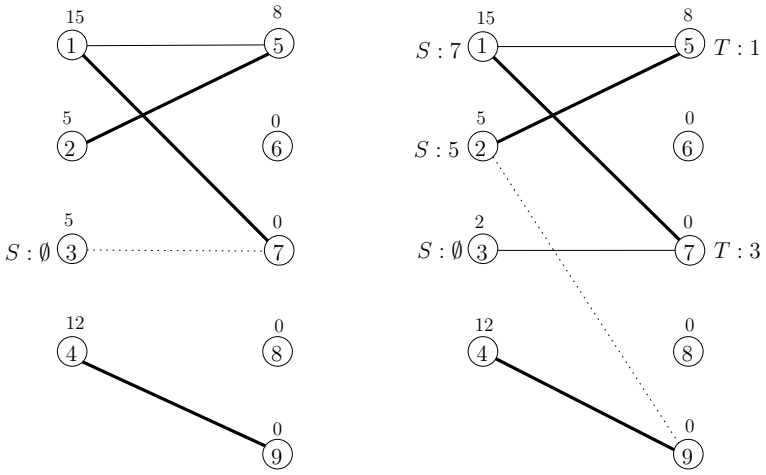
Obr. 6.11: Riešenie príkladu z obr. 6.8 maďarskou metódou. Tretí a štvrtý medzivýsledok: maďarský les ($\delta = 1$) a zväčšujúca alternujúca cesta.



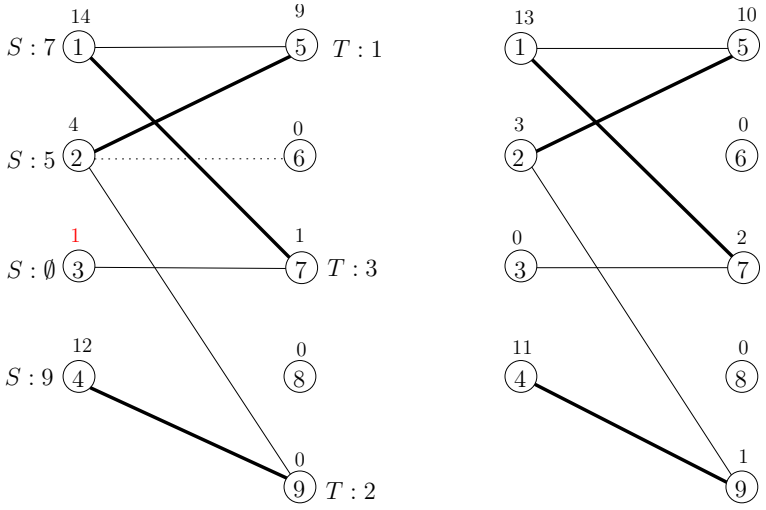
Obr. 6.12: Riešenie príkladu z obr. 6.8 maďarskou metódou. Piaty a šiesty medzivýsledok: maďarský les ($\delta = 7$) a zväčšujúca alternujúca cesta.

a $i_j = i$, inak zmenu nerobíme. Na aktualizácie počas budovania lesa vystačíme s $O(pq)$ operáciami. Potom pri hľadaní δ_2 stačí nájsť minimálne d_j spomedzi neoznačkovaných vrcholov v T a i_j nám určí príslušnú hranu. To potrebuje iba $O(q)$ operácií a tak dostaneme celkovú zložitosť $O(n^3)$.

Cvičenie 6.10. Vyriešte priraďovacie problémy pre nasledovné cenové matice. [V zátvorke je optimálna hodnota.]



Obr. 6.13: Riešenie príkladu z obr. 6.8 maďarskou metódou. Siedmy a ôsmy medzivýsledok: maďarský les ($\delta = 3$) a maďarský les ($\delta = 1$).



Obr. 6.14: Riešenie príkladu z obr. 6.8 maďarskou metódou. Deviaty medzivýsledok: maďarský les ($\delta = \delta_1 = 1$) a finálny výsledok: $\sum c_{ij}x_{ij} = 40 = \sum u_i + \sum v_j$.

(a) $\begin{bmatrix} 7 & 6 & 8 & 8 \\ 5 & 2 & 7 & 8 \\ 6 & 1 & 4 & 9 \\ 5 & 6 & 5 & 9 \end{bmatrix}$ $(z^* = 29)$

(b) $\begin{bmatrix} 6 & 5 & 9 & 7 & 3 & 2 \\ 5 & 3 & 5 & 4 & 6 & 3 \\ 4 & 2 & 2 & 1 & 7 & 1 \\ 9 & 1 & 1 & 5 & 3 & 5 \\ 3 & 5 & 6 & 2 & 5 & 0 \\ 2 & 4 & 4 & 9 & 3 & 2 \end{bmatrix}$ $(z^* = 42)$

(c) $\begin{bmatrix} 6 & 4 & 5 & 7 & 1 & 8 \\ 1 & 6 & 3 & 3 & 5 & 6 \\ 5 & 5 & 1 & 1 & 6 & 4 \\ 6 & 1 & 2 & 5 & 0 & 2 \\ 2 & 2 & 0 & 7 & 6 & 4 \\ 0 & 2 & 7 & 2 & 4 & 1 \end{bmatrix}$ $(z^* = 40)$

Kapitola 7

POCHÔDZKY

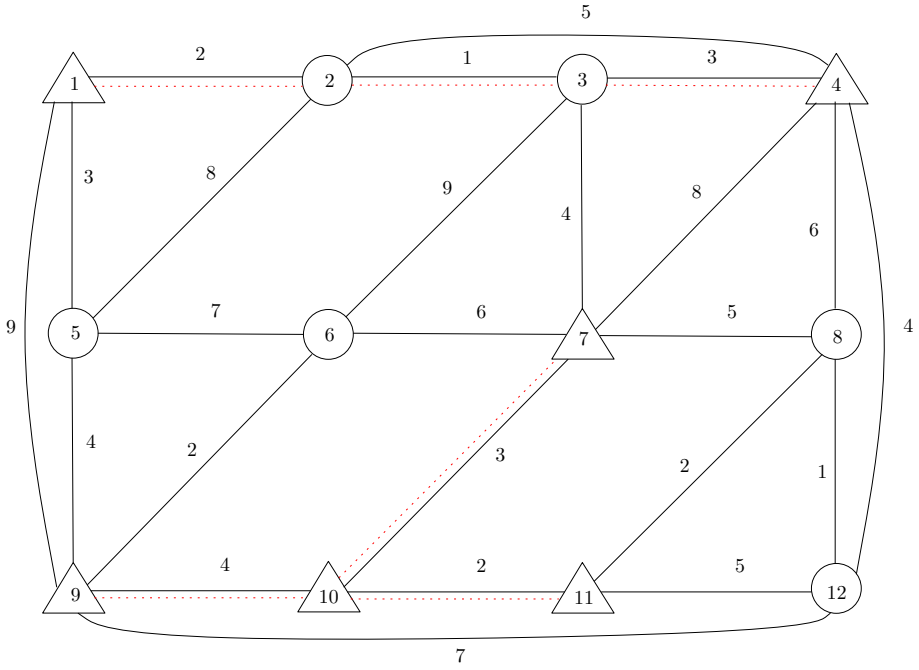
7.1 Úloha čínskeho poštára v grafoch

Daný je súvislý graf G, kde každá hrana (i, j) má reálnu dĺžku $c_{ij} > 0$). Treba nájsť najkratší uzavretý sled obsahujúci všetky hrany, t.j. najkratší uzavretý eulerovský sled. Takýto sled zodpovedá nakratšej pochôdzke poštára, ktorý začne na pošte, prejde všetky ulice svojho rajónu a skončí na pošte. Prvýkrát sa touto úlohou zaoberal čínsky matematik Kwan, čo vysvetľuje názov úlohy.

Keďže k dĺžke sledu prispieva každá hrana svojou dĺžkou toľkokrát koľkokrát ju sled obsahuje, tak každý takýto sled bude mať dĺžku aspoň takú ako je súčet dĺžok všetkých hrán grafu. Pritom tu nastáva rovnosť práve vtedy, keď sled obsahuje každú hranu práve raz, čo sa dá zrealizovať v prípade eulerovského grafu, t.j. ak má všetky stupne vrcholov párne. V opačnom prípade musíme niektorými hranami prechádzať viackrát. Takúto situáciu máme na obr. 7.1, kde každý opakovaný prechod hranou je vyznačený bodkovanou červenou čiarou (smer cestovania nie je vyznačený). Tieto bodkované hrany budeme nazývať fiktívne a multigraf G', ktorý tvoria sa nazýva fiktívny. Zjednotenie pôvodného grafu G a fiktívneho multigrafu G' dáva tzv. rozšírený multigraf $\hat{G}$, na ktorom je realizovaný eulerovský sled, a preto má všetky stupne vrcholov párne. Je vhodné dĺžky hrán považovať za ceny a potom sumárnu dĺžku hrán nazývať cenou multigrafu. Takto môžeme stručne písať, že $c(\hat{G}) = c(G) + c(G')$. Teda najkratšej pštárovej pochôdzke (jej dĺžka je $c(\hat{G})$) zodpovedá najlacnejší fiktívny multigraf. Nech $N(G)$ označuje množinu všetkých nepárnych vrcholov grafu G (na obr. 7.1 sú nepárne vrcholy vyznačené trojuholníkmi). Jednoduché pozorovania sú zhrnuté v nasledujúcej vete.

VETA 7.1. *Pre ľubovoľný fiktívny multigraf G' korešpondujúci grafu G platí:*
(i) $N(G) = N(G')$ (t.j. G a G' majú tie isté nepárne vrcholy).
(ii) Ak G' je najlacnejší, tak je acyklický (t.j. les) a možno ho hranovo rozložiť na $N(G)/2$ ciest a množinu $N(G)$ do $N(G)/2$ dvojíc vrcholov tak, že každá cesta má za konce vrcholy práve jednej takejto dvojice.

Dôkaz: (i) Rozšírený multigraf $\hat{G}$ má všetky vrcholy párne. (ii) Ak by



Obr. 7.1: Príklad úlohy čínskeho poštára

obsahoval cyklus, tak vynechaním hrán cyklu by sa nezmenila parita stupňov vrcholov a cena by sa zmenšila. Pre druhé tvrdenie stačí vedieť rozložiť každý komponent, teda strom. To možno urobiť tak, že v strome zvolíme ľubovoľné 2 vrcholy nepárneho stupňa a zoberieme (jedinú) cestu ktorá ich spája. Keďže začiatok aj koniec tejto cesty majú v G' nepárny stupeň, tak v G sú to vrcholy nepárneho stupňa. Vynechaním hrán tejto cesty získame menší les, ktorý bude mať o 2 nepárne vrcholy menej, lebo parita ostatných sa zachová. Zvyšok rozkladáme takto ďalej až získame požadovaný rozklad. ■

Teda úloha čínskeho poštára sa transformovala na úlohu nájsť najlacnejší fiktívny graf (už vieme, že najlacnejší fiktívny multigraf nemá paralelné hrany, a teda je to graf). A táto úloha sa previedla na úlohu popáriť nepárne vrcholy grafu G najlacnejšou sústavou hranovo-disjunktných ciest, čo je netriviálna úloha. Edmonds navrhol nasledovný postup na jej riešenie:

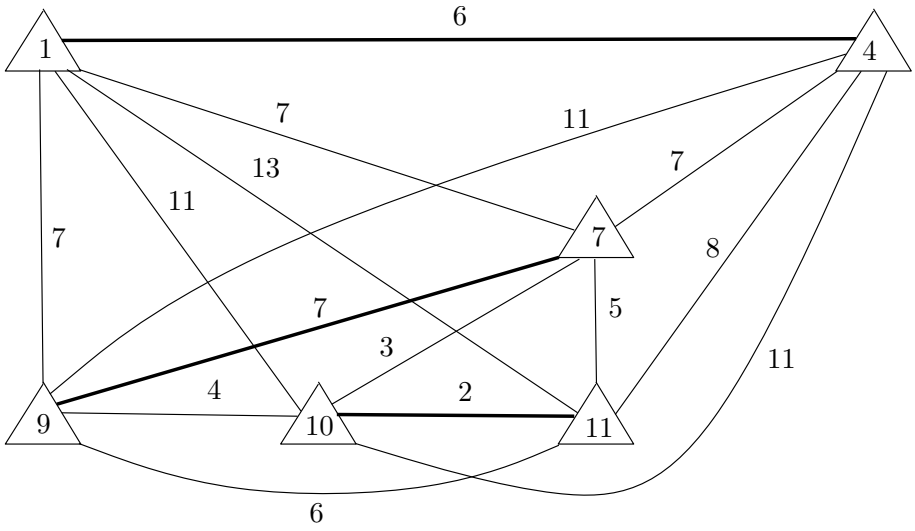
(1) Zostrojme kompletný graf K na množine nepárnych vrcholov $N(G)$, pričom hrane (i, j) priraďme cenu d_{ij}, ktorá sa rovná dĺžke najkratšej i-j cesty v G.

(2) V grafe K nájdime najlacnejšie perfektné párenie M.

(3) Každú hranu $(i, j) \in M$ nahraďme ľubovoľnou najkratšou i-j cestou v G a tým získame nejakú sústavu S ciest v G páriacu všetky nepárne vrcholy.

Tento postup pre príklad z obr. 7.1 je ilustrovaný na obr. 7.2. K uvede-

nému pároniu M zodpovedá sústava ciest S v grafe G (na obr. 7.1 vyznačená bodkovane).

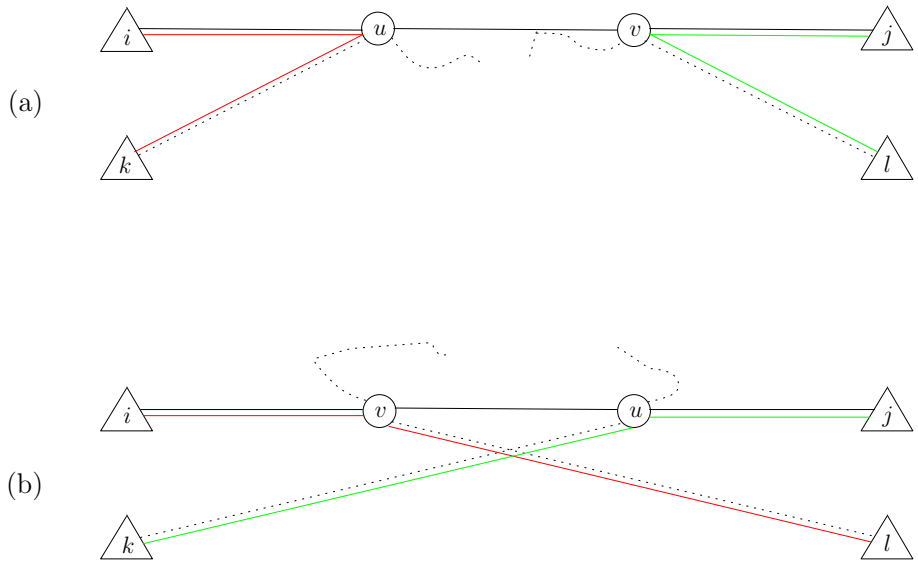


Obr. 7.2: Kompletný hranovo-ohodnotený graf K k príkladu z obr. 7.1 s hrubo-vyznačeným najlacnejším perfektným párením M

VETA 7.2. *Sústava ciest S je najlacnejšia a po dvoch hranovo-disjunktná.*

Dôkaz: To, že je najlacnejšia, vyplýva z definovania cien grafu K. Pre spor dpokladajme, že v tejto sústave nejaké 2 cesty P a Q a majú spoločnú hranu. Rozlíšime 2 situácie podľa obr. 7.3, kde P je i-j cesta vyznačená plnou čiernou čiarou a Q je k-l cesta zobrazená bodkovanou čiernou čiarou. Cestujúc po k-l ceste Q, nech u je prvý spoločný vrchol s i-j cestou P a v je posledný spoločný vrchol. Pripomíname, že i, j, k, l sú 4 rôzne nepárne vrcholy. Nie je dôležité ako vyzerá úsek cesty Q medzi vrcholmi u a v (má však spoločnú hranu s cestou P). V oboch prípadoch vieme popáriť uvedené vrcholy pomocou nových 2 ciest (vyznačené sú červenou a zelenou čiarou), ktoré nebudú mať spoločnú hranu (ak $u = v$, tak budú mať spoločný tento vrchol). Celkove sme nejaké hrany zo zjednotenia ciest P a Q vynechali a žiadne nové nepribrali. Preto cena novej sústavy ciest je znížená, čo je spor. ■

Cvičenie 7.1. V úlohe čínskeho poštára sme predpokladali, že cena každej hrany je kladná. Prediskutujte horeuvedený postup pre taký prípad, keď cena každej hrany je nezáporná.



Obr. 7.3: Dve možnosti pre vzájomný vzťah dvoch ciest páriacich nepárne vrcholy

7.2 Čínsky poštár v digrafoch

Formálne rovnako ako v grafoch, aj tu máme daný silne súvislý digraf G s kladnými reálnymi dĺžkami (cenami) hrán c_{ij} a úlohou je nájsť najkratší uzavretý eulerovský sled. Ak je digraf rovnovážne orientovaný, tak má uzavretý eulerovský ťah a ten je riešením úlohy. Inak v slede musíme ísť niektorými hranami viackrát. Opakovaný prechod hranou si môžeme predstaviť ako pridanie paralelnej hrany. Tieto nové hrany budeme nazývať fiktívne a utvoria tzv. fiktívny multidigraf G'. V príklade na obr. 7.1 je vyznačený bodkovanými čiarami. Zjednotenie pôvodného digrafu G a multidigrafu G' je tzv. rozšírený multidigraf $\hat{G}$, ktorý je rovnovážne orientovaný, lebo sled sa v ňom realizuje ako ťah. Tak ako pre grafy, aj tu máme $c(\hat{G}) = c(G)+c(G')$. Teda najkratšej pštárovej pochôdzke (jej dĺžka je $c(\hat{G})$) zodpovedá najlacnejší fiktívny multidigraf. Pre každý vrchol v grafu G definujeme zisk $z(v) = deg^-(v) - deg^+(v)$ (rozdiel medzi prichádzajúcim a odchádzajúcim stupňom). Vrchol s kladným ziskom sa nazýva kladný a vrchol so záporným ziskom je záporný. Nech $V^+(G)$ $[V^-(G)]$ označuje množinu všetkých kladných [záporných] vrcholov. (Na obr. 7.4 sú kladné vrcholy vyznačené päťuholníkmi a záporné štvorčekmi.) Jednoduché pozorovania sú zhrnuté v nasledujúcej vete.

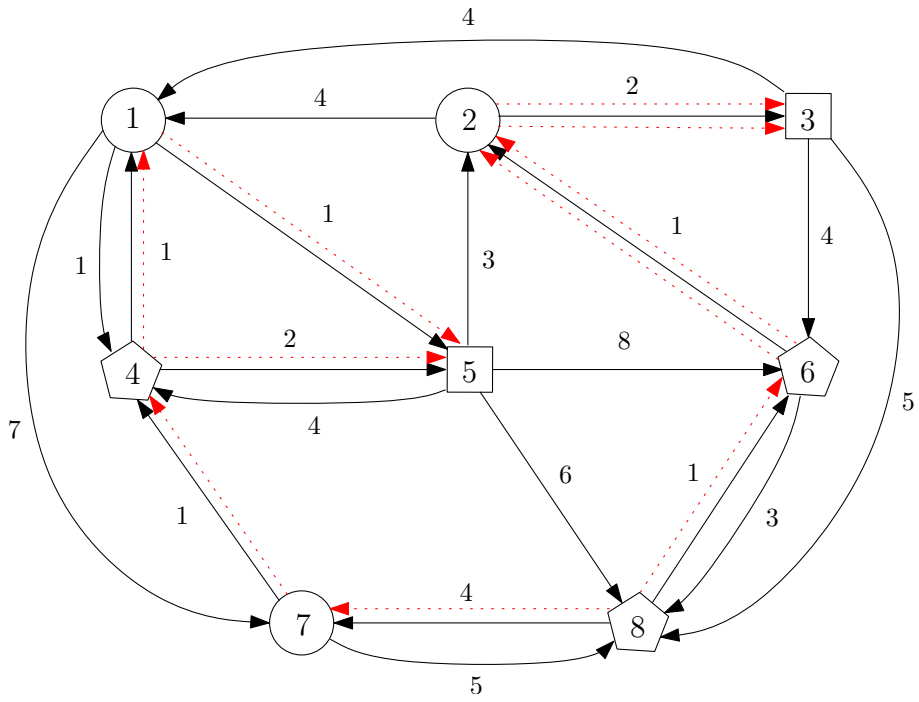
VETA 7.3. *Pre ľubovoľný digraf G a ľubovoľný korešpondujúci fiktívny multidigraf G' platí:*

(i) $V^+(G') = V^-(G)$, $V^-(G') = V^+(G)$, $z(V^+(G) = -z(V^-(G)$.

(ii) Ak G' je najlacnejší fiktívny multidigraf pre G, tak je acyklický a možno ho rozložiť na vzájomne hranovo-disjunktné cesty, ktorých je $\sum_{u \in V^+(G)} z(u)$, pričom v každom vrchole $u \in V^+(G)$ začína práve $z(u)$ ciest a v každom vrchole $v \in V^-(G)$ končí práve $-z(v)$ ciest.

Dôkaz: (i) To vyplýva z faktu, že $\hat{G}$ je rovnovážne orientovaný a z definície zisku.

(ii) Ak by G' obsahoval nejaký cyklus, tak vynechaním jeho hrán by sme získali lacnejší fiktívny multidigraf, lebo zisky vrcholov sa zachovajú. Ak začneme v G' cestovať v ľubovoľnom vrchole u s kladným ziskom v G (a teda podľa (i) záporným v G') stále po nových hranách, tak niekedy dosiahneme vrchol v, ktorý bude mať záporný zisk v G (a teda kladný v G', lebo G' je acyklický a má ústie). Vynechaním hrán tejto cesty z G' sa zmení zisk začiatku a konca tejto cesty presne o jednotku a postup môžeme opakovať pokiaľ bude zvyšok obsahovať hranu. ■



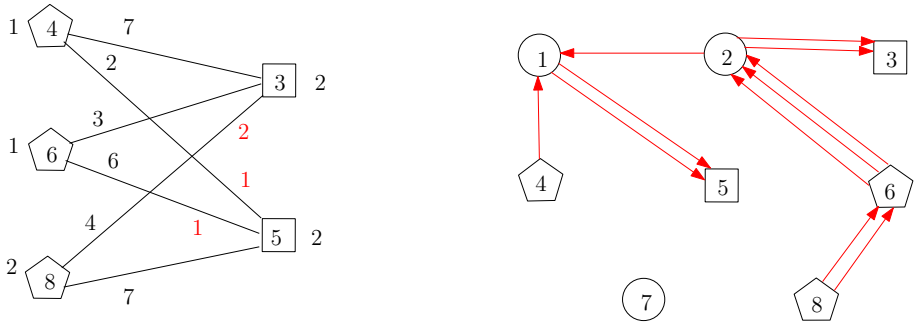
Obr. 7.4: Príklad úlohy čínskeho poštára v digrafe (plné čiary) a fiktívny multidigraf (červené bodkované čiary)

Úloha čínskeho poštára sa transformovala na úlohu nájsť najlacnejší fiktívny mutidigraf. A táto úloha sa previedla na úlohu nájsť najlacnejšiu sústavu ciest S v digrafe G (nemusia byť vzájomne hranovo-disjunktné) takú, že v každom

vrchole $u \in V^+(G)$ začína práve $z(u)$ ciest a v každom vrchole $v \in V^-(G)$ končí práve $-z(v)$ ciest.

Túto úlohu možno sformulovať ako celočíselnú dopravnú (vybilancovanú) úlohu, kde každý vrchol $u \in V^+(G)$ predstavuje dodavateľa s požiadavkou $z(u)$ a každý vrchol $v \in V^-(G)$ odberateľa s požiadavkou $-z(v)$. Pritom jednotkové dopravné náklady z u do v sa rovnajú dĺžke d_{uv} najkratšej u-v cesty v digrafe G. Z lineárneho programovania je známe, že medzi optimálnymi riešeniami bude aj celočíselné riešenie a také tu potrebujeme: najkratšiu u-v cestu zoberieme do hľadanej sústavy x_{uv}-krát, kde x_{uv} je príslušná zložka optimálneho riešenia (prepravované množstvo z u do v). Pre náš digraf G z obr. 7.4 je $V^+(G) = \{4, 6, 8\}$ a $V^-(G) = \{3, 5\}$; príslušné zisky sú (1, 1, 2) a (-2, -2). Potrebné vzdialenosti d_{uv} a zodpovedajúce najkratšie cesty sú tieto: $d_{43} = 7$, (4, 5, 2, 3); $d_{45} = 2$, (4, 1, 5); $d_{63} = 3$, (6, 2, 3); $d_{65} = 6$, (6, 2, 1, 5); $d_{83} = 4$, (8, 6, 2, 3); $d_{85} = 7$, (8, 7, 4, 5). Korešpondujúca dopravná úloha je zobrazená na obr. 7.5. Jedno z optimálnych riešení má nenulové zložky: $x_{45} = 1$, $x_{65} = 1$ a $x_{83} = 2$. Tieto dali najlacnejší fiktívny multidigraf ceny 16 uvedený na tom istom obrázku (je iný ako ten na obr. 7.4).

Cvičenie 7.2. Nájdite iné optimálne riešenie tejto dopravnej úlohy a zakreslite príslušný fiktívny multidigraf.



Obr. 7.5: Dopravná úloha k príkladu z obr. 7.4. Nenulové zložky optimálneho riešenia sú červené; vpravo je korešpondujúci fiktívny multidigraf

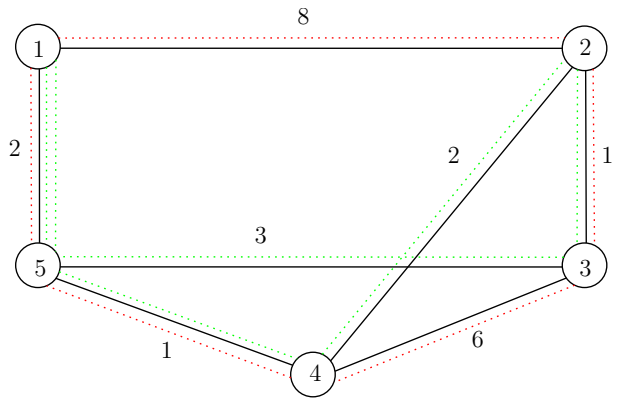
Alternatívny prístup k úlohe čínskeho poštára v digrafoch je (namiesto riešenia dopravnej úlohy) hľadanie najlacnejšej celočíselnej cirkulácie s dolnou medzou 1.

Cvičenie 7.3. Sformulujte príslušnú tokovú úlohu a zdôvodnite.

Poznamenajme, že úloha čínskeho poštára v migrafoch je NP-ťažká. Taká je aj tzv. úloha vidieckeho poštára, kde treba prejsť predpísané hrany, ale využívať môžeme všetky hrany daného grafu, resp. digrafu.

7.3 Úloha obchodného cestujúceho

Tu je úlohou pochodiť všetky vrcholy (nie hrany). Tradične sa uvažuje úloha: Daný je súvislý graf s nezápornými reálnymi dĺžkami (cenami) hrán a treba nájsť najkratší (najlacnejší) hamiltonovský cyklus. To zodpovedá tomu, že obchodný cestujúci má prejsť najkratšiu okružnú trasu, v rámci ktorej má navštíviť každé mesto rajónu práve raz. Táto posledná podmienka je dosť umelá – ide o idealizáciu. Vidíme, že pre mnoho grafov nebude existovať ani prípustné riešenie (lebo nemusia mať hamiltonovský cyklus), a preto sa často študuje aj úloha podobná úlohe čínskeho poštára, ktorá lepšie zodpovedá realite: Nájsť najkratší (najlacnejší) hamiltonovský sled (t.j. najkratší uzavretý sled obsahujúci všetky vrcholy). Na obr. 7.6 je príklad grafu, kde nakratší hamiltonovský cyklus $(1, 2, 3, 4, 5, 1)$ má dĺžku 18 a najkratší hamiltonovský sled $(1, 5, 3, 2, 4, 5, 1)$ má dĺžku 11.



Obr. 7.6: Najkratší hamiltonovský cyklus (červený) a najkratší hamiltonovský sled (zelený)

Napriek tomu sú oba problémy v istom zmysle rovnocenné.

VETA 7.4. *Úloha nájsť najkratší hamiltonovský cyklus (NHC) a úloha nájsť najkratší hamiltonovský sled (NHS) sú silne polynomiálne ekvivalentné.*

Dôkaz: Najprv urobíme transformáciu z NHC na NHS. Nech M je veľmi veľké číslo. Pridajme ku každej cene c_{ij} číslo M a tým dostaneme nové ceny c'_{ij}. Ak pri týchto nových cenách nájdeme NHS, tak bude cyklom práve vtedy, keď bude mať dĺžku $D' < (n+1)M$ (lebo uzavrety sled obsahujúci všetky vrcholy je cyklom práve vtedy, keď má presne n hrán). Tento cyklus bude riešením úlohy nájsť NHC.

Obrátenú transformáciu možno urobiť tak, že k danému grafu G priradíme kompletný graf $\hat{G}$ na tých istých vrcholoch, pričom cenu $\hat{c}_{ij}$ hrany $(i, j) \in E(\hat{G})$ definujeme rovnú dĺžke najkratšej i-j cesty v G. Potom NH sledu v G zodpovedá NH cyklus v $\hat{G}$ tej istej dĺžky a obrátene. ■

7.3.1 Aproximácia

Vo všeobecnosti je úloha obchodného cestujúceho NP-ťažká aj keď žiadame nájsť iba hamiltonovský cyklus, ktorého cena nepresahuje ρ-násobok optimálnej ceny pre $\rho > 1$.

Cvičenie 7.4. Dokážte to. [Návod: V danom grafe definujte všetky ceny hrán jednotkové a doplňte graf na kompletný hranami veľkej ceny. Potom pôvodný graf má hamiltonovský cyklus práve vtedy, keď ρ-aproximačný algoritmus pre úlohu obchodného cestujúceho v zostrojenom kompletnom grafe dá cyklus ceny n.)

V prípade, že v úlohe obchodného cestujúceho je graf G kompletný (označíme ho symbolom K), ceny sú nezáporné a spĺňajú trojuholníkovú nerovnosť, tak hovoríme o metrickej úlohe obchodného cestujúceho. Vtedy vieme istú aproximáciu zaručiť. Uvedieme 2 aproximačné algoritmy.

METÓDA ZDVOJENIA KOSTRY.

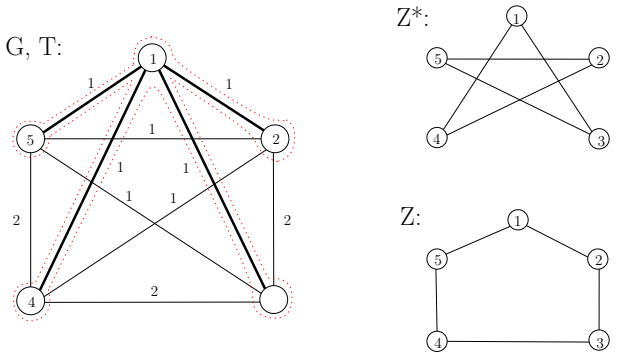
(i) Nájdeme najlacnejšiu kostru T daného kompletného grafu K.

(ii) Pre T nájdeme uzavretý sled S obsahujúci každú hranu práve 2-krát (napr. Tarryho algoritmom).

(iii) Cestujúc po S utvoríme z neho cyklus Z tak, že opakujúci sa vrchol (vrcholy) preskočíme použitím novej hrany.

Tento algoritmus je ilustrovaný na obr. 7.7, kde kostra T je vyznačená hrubými čiarami a sled $S = (1, 2, 1, 3, 1, 4, 1, 5, 1)$ bodkovanou čiarou. Vpravo je najlacnejší hamiltonovský cyklus $Z^* = (1, 3, 5, 2, 4, 1)$ a tiež cyklus $Z = (1, 2, 3, 4, 5, 1)$ utvorený podľa kroku (iii). Vidíme, že $c(Z^*) = 5$, $c(Z) = 8$ a teda $c(Z) = 1,6\ c(Z^*)$.

Poznamename, že niekedy sa krok (ii) uvádza takto: (ii') Nahradíme každú hranu kostry T dvojicou paralelných hrán (odtiaľ je názov metódy) a v získanom eulerovskom multigrafe T' nájdeme uzavretý eulerovský ťah S.



Obr. 7.7: Príklad k metóde zdvojenia kostry

VETA 7.5. *Pre metrickú úlohu obchodného cestujúceho metóda zdvojenia kostry dáva hamiltonovský cyklus Z ceny $c(Z) \leq 2c(Z^*)$, kde Z^* je najlacnejší hamiltonovský cyklus.*

Dôkaz: Z trojuholníkovej nerovnosti máme $c(Z) \leq c(S) = 2c(T)$. Na druhej strane, vynechaním ľubovoľnej hrany e zo Z^* získame kostru grafu K. Preto $c(T) \leq c(Z^* - e) \leq c(Z^*)$. ■

Cvičenie 7.5. Odhadnite výpočtovú zložitosť metódy zdvojenia kostry.

Cvičenie 7.6. Zistite či vo vete 7.4 môžeme dať ostrú nerovnosť.

Cvičenie 7.7. Zovšeobecnite príklad z obr. 7.7 a na ňom ukážte, že číslo 2 vo vete 7.5 nemožno nahradiť menším.

CHRISTOFIDESOVA METÓDA KOSTRY A PÁRENIA

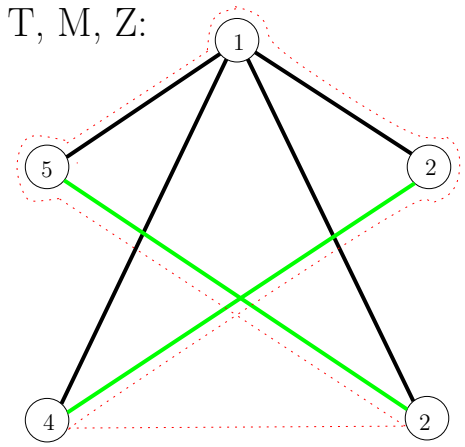
Christofides dosiahol úsporu v konštrukcii eulerovského grafu tým, že namiesto zdvojenia kostry pridal ku kostre párenie na nepárnych vrcholoch.

(i) Nájdeme najlacnejšiu kostru T daného kompletného grafu K.

(ii) Na množine všetkých nepárnych vrcholov kostry T nájdeme v K najlacnejšie perfektné párenie M.

(iii) V multigrafe $T + M$ (ktorý je eulerovský), nájdeme uzavretý eulerovský ťah S.

(iv) Cestujúc po ťahu S utvoríme z neho cyklus Z tak, že opakujúci sa vrchol (vrcholy) preskočíme použitím novej hrany.



Obr. 7.8: Ilustrácia ku Christofidesovej metóde aplikovanej na príklad z obr. 7.7

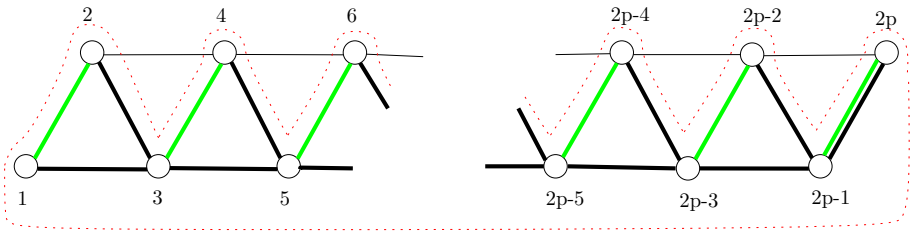
Na obr. 7.8 je táto metóda aplikovaná na príklad z obr. 7.6. Dostávame hamiltonovský cyklus Z ceny $c(Z) = 6 = 1,2\,c(Z^*)$, čo je lepšie ako pri predchádzajúcej metóde. Aj teoretický odhad je lepší.

VETA 7.6. *Pre metrickú úlohu obchodného cestujúceho Christofidesova metóda dáva hamiltonovský cyklus Z ceny $c(Z) \leq 1,5\ c(Z^*)$, kde Z^* je najlacnejší hamiltonovský cyklus.*

Dôkaz: Keďže pre ľubovoľnú hranu e cyklu Z^* je $Z^* - e$ kostrou grafu K, tak $c(T) \leq c(Z^* - e) \leq c(Z^*)$. Nech N je množina všetkých nepárnych vrcholov kostry T a $K(N)$ je indukovaný podgraf grafu K na N. K cyklu Z^* vyrobme cyklus Z_N^* v $K(N)$ tak, že úseky nepatriace do N nahradíme hranou v $K(N)$. Z trojuholníkovej nerovnosti máme $c(Z_N^*) \leq c(Z^*)$. Cyklus Z_N^* má párnu dĺžku, a teda ho možno rozložiť na 2 párenia grafu $K(N)$. Každé z nich má cenu aspoň $c(M)$, a preto $c(M) \leq \frac{1}{2}c(Z_N^*)$. Kombináciou predlošlých vzťahov dostávame:

$$c(Z) \leq c(S) = c(T) + c(M) \leq c(Z^*) + \frac{1}{2}c(Z_N^* \leq \frac{3}{2}c(Z^*)).$$

■



Obr. 7.9: Príklad ukazujúci, že koeficient 1,5 pre Christofidesovu metódu vo vete 7.6 nemožno zmenšiť

Cvičenie 7.8. Uvažujte príklad kompletného grafu na 2p vrcholoch, ktorého podstatná časť je na obr. 7.9 kde každá nakreslená hrana plnou čiarou má dĺžku 1 a ostatné hrany majú dĺžky rovné vzdialenostiam ich koncových vrcholov v nakrelenom podgrafe. Overte, že hrubo vyznačené čierne hrany tvoria najlacnejšiu kostru a zelené hrany najlacnejšie párenie. Ukážte, že Christofidesova metóda z toho dá cyklus $Z = (1, 2, 3, \ldots, 2p-1, 2p, 1)$ dĺžky $3p - 1$ (červená bodkovaná čiara) a pritom najkratší hamiltonovský cyklus má dĺžku $2p$.

Uvedený príklad v podstate funguje aj pre body v rovine s euklidovskou metrikou. Pre takúto triedu úloh obchodného cestujúceho však boli vyvinuté geometrické algoritmy dávajúce lepšiu aproximáciu.