Kapitola 6

Nejkratší cesty z jednoho uzlu

Při zavedení vzdálenosti na neorientovaných grafech jsme se nezabývali problémem, jak pro daný graf G a jeho nějakou dvojici uzlů u, v určíme vzdálenost $d(u, v)$. Odvodili jsme pouze některé jednoduché vlastnosti vzdálenosti a zavedli několik dalších grafových pojmů, které se vzdáleností souvisejí.

V této a následující kapitole se budeme důkladně věnovat právě otázkám určování vzdálenosti na grafech: ukážeme různé varianty tohoto problému, algoritmy jejich řešení a odvodíme asymptotické časové složitosti těchto algoritmů. S ohledem na potřebu co největší obecnosti prezentovaných výsledků začneme zobecněním samotného pojmu vzdálenosti, díky kterému vzdálenost ztratí některé ze svých dřívějších vlastností. Vzdálenost však bude vždy definována prostřednictvím optimální (nejkratší) cesty mezi dvěma uzly, takže problematika vzdáleností se kryje s určováním nejkratších cest.

6.1 Vzdálenost na orientovaných a ohodnocených grafech

V závěru odst. 3.1 jsme zavedli vzdálenost $d(u, v)$ z uzlu u do uzlu v na orientovaném grafu jako délku (tj. počet hran) nejkratší **orientované cesty** z uzlu u do uzlu v. Taková vzdálenost už není symetrická, ale další vlastnosti neorientované vzdálenosti pro ni zůstávají zachovány. Výrazněji se situace změní až tehdy, když při zavedení vzdálenosti uvažujeme možnost ohodnocení hran grafu reálnými čísly, jako jsme to udělali při formulaci problému minimální kostry.

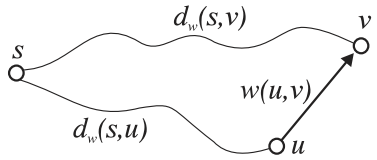
Definice 6.1: Nechť $S = \langle h_1, h_2, \ldots, h_k \rangle$ je spojení orientovaného grafu $G = \langle H, U \rangle$ s reálným ohodnocením hran $w : H \mapsto \mathbf{R}$. **W-délkou** spojení S pak nazýváme reálné číslo $d_w(S)$ definované vztahem

$$d_w(S) \;=\; \sum_{i=1}^{k} w(h_i). \tag{6.1}$$

W-vzdáleností $d_w(u, v)$ z uzlu u do uzlu v grafu G rozumíme nejmenší w-délku spojení z u do v, pokud takové spojení v grafu G existuje. Je-li uzel v z uzlu u nedostupný, pokládáme $d_w(u, v) = \infty$.

Takto pojaté chápání vzdálenosti nám dává větší aplikační možnosti. Ohodnocení hran w může mít v konkrétních aplikacích grafů různé interpretace: kromě skutečné délky nějaké křivky to může být např. časový údaj, cena, ztráta nebo jakákoliv jiná kvantita, jejíž úhrnné množství se získá součtem ohodnocení jednotlivých hran. Ve většině případů budou hodnoty $w(h)$ nezáporné, à priori však nechceme vyloučit ani možnost záporného ohodnocení některých (nebo všech) hran grafu.

V takovém případě ale vzniká otázka, zda má předchozí definice vůbec smysl. Pokud má nějaká orientovaná cesta C z u do v neprázdný průnik s cyklem grafu G, který má zápornou w-délku, pak se s každým opakovaným zařazením tohoto cyklu do spojení s cestou C stále snižuje výsledná w-délka, takže minimální spojení neexistuje. V takových případech budeme



Obrázek 6.1: Vliv w-délky na vzdálenost

pokládat $d_w(u,v) = -\infty$. Jestliže žádná orientovaná cesta z u do v neprochází cyklem grafu G se zápornou nebo nulovou w-délkou (pokud např. graf G žádný takový cyklus nemá), pak bude každé minimální spojení určující w-vzdálenost $d_w(u,v)$ orientovanou cestou. Při počítání se vzdálenostmi budeme používat konvenci vyjádřenou následujícími vztahy:

$$\begin{aligned} a + (-\infty) &= (-\infty) + a = -\infty \quad \text{pro libovolné reálné } a \neq \infty \\ a + \infty &= \infty + a = \infty \quad \text{pro libovolné reálné } a \neq -\infty \end{aligned}$$

Všimněme si po tomto doplnění původních vlastností vzdálenosti a zkusme, v jakých případech lze pro w-vzdálenost formulovat alespoň analogické vztahy, pokud by stejné vztahy neplatily.

(0) *vzdálenost $d(u,v)$ je celé nezáporné číslo*
Tuto vlastnost lze zachovat jen při nezáporném celočíselném ohodnocení všech hran.

(1) $d(u,v) \geq 0$*; přitom* $d(u,v) = 0$*, právě když* $u = v$
Tuto vlastnost lze zachovat jen při kladném ohodnocení všech hran.

(2) $d(u,v) = d(v,u)$
Se symetrií je třeba se u orientovaných grafů definitivně rozloučit.

(3) $d(u,v) \leq d(u,z) + d(z,v)$
Trojúhelníková nerovnost platí beze změny i pro w-vzdálenosti, neboť je dána podmínkou minimality a součtovým charakterem určení w-délky cesty.

(4) *je-li* $d(u,v) > 1$*, pak existuje uzel* $z \in U$ *tak, že* $u \neq z \neq v$ *a platí* $d(u,v) = d(u,z) + d(z,v)$
Pro w-vzdálenost je možné vyslovit analogické tvrzení, je jen třeba upravit jeho předpoklad. Podmínka $d_w(u,v) > 1$ totiž nevyjadřuje to, co je pro důkaz zapotřebí, tedy existenci nějakého vnitřního uzlu na minimální cestě. Následující lemma je zobecněním vlastnosti (4).

Lemma 6.2: Nechť $G = \langle H, U \rangle$ je orientovaný graf s ohodnocením hran $w : H \mapsto \mathbf{R}$ a buď $L = \langle u_1, u_2, \ldots, u_k \rangle$ cesta určující vzdálenost $d_w(u_1, u_k)$ z uzlu u_1 do uzlu u_k. Pro libovolná $i, j : 1 \leq i \leq j \leq k$ potom platí

$$d_w(u_1, u_k) = d_w(u_1, u_i) + d_w(u_i, u_j) + d_w(u_j, u_k). \tag{6.2}$$

Důkaz: Dokazovaný vztah vyjadřuje, že se celková w-délka cesty počítá jako součet w-délek jednotlivých hran, tedy i jako součet w-délek dílčích cest. Tyto cesty musí být současně cestami minimálními a určovat w-vzdálenost mezi příslušnými uzly, jinak by ani celková cesta nemohla být minimální. △

Hledání minimálních cest splňuje tedy podmínku optimálnosti podstruktury – libovolná část minimální cesty je opět minimální. Je-li speciálně (u,v) poslední hranou minimání cesty z uzlu s do uzlu v, bude platit

$$d_w(s,v) = d_w(s,u) + w(u,v) \tag{6.3}$$

Pro libovolnou hranu (u,v) pak lze formulovat následující analogii lemmatu 4.14.

Lemma 6.3: Nechť $G = \langle H, U \rangle$ je orientovaný graf s ohodnocením hran $w : H \mapsto R$ a buď s jeho zvolený uzel. Potom pro každou hranu $(u,v) \in H$ platí

$$d_w(s,v) \leq d_w(s,u) + w(u,v). \tag{6.4}$$

Důkaz: Situaci ilustruje obrázek 6.1. Je-li v dostupný z s po cestě, která prochází cyklem záporné w-délky, pak platí ve vztahu (6.4) ostrá nerovnost nebo rovnost podle toho, zda i uzel u je takto dostupný. Protože nemůže nastat situace, že by takto byl dostupný pouze uzel u a nikoliv uzel v, nemusíme nadále v důkazu existenci záporných cyklů uvažovat.
Pokud je uzel v dostupný z uzlu s a hrana (u, v) leží na nějaké minimální cestě z s do v, potom podle (6.3) platí ve vztahu (6.4) rovnost. Je-li v dostupný z s, ale hrana (u, v) na žádné minimální cestě neleží (uzel u může být i nedostupný z s), pak platí ve vztahu (6.4) ostrá nerovnost. Jsou-li oba uzly u i v z uzlu s nedostupné, pak je podle naší konvence $d_w(s, v) = \infty = d_w(s, u) + w(u, v)$. △

Základní úlohou při určování vzdáleností v orientovaném grafu $G = \langle H, U \rangle$ je **nalezení nejkratších cest z jednoho zadaného uzlu** $s \in U$ do všech ostatních uzlů $v \in U$ grafu G. Algoritmus řešení této základní úlohy lze použít i v dalších variantách hledání nejkratších cest:

- ◇ **Problém nejkratších cest do jediného cílového uzlu** spočívá v nalezení nejkratších cest ze všech uzlů $u \in U$ do pevně zadaného cílového uzlu t. Stačí zřejmě řešit základní úlohu v opačně orientovaném grafu G^-.
- ◇ **Problém jediné nejkratší cesty** spočívá v určení nejkratší cesty pro jedinou zadanou (uspořádanou) dvojici uzlů $u, v \in U$. Pokud řešíme základní úlohu s výchozím uzlem u, vyřešíme tím současně i problém jediné nejkratší cesty. I když se může zdát, že se jedná o problém jednodušší než je základní úloha, není znám žádný algoritmus, který by měl asymptotickou složitost v nejhorším případě lepší než algoritmus pro základní úlohu.
- ◇ **Problém všech nejkratších cest** znamená nalezení nejkratší cesty pro všechny (uspořádané) dvojice uzlů $u, v \in U$. Také tento problém lze řešit s použitím algoritmu pro základní úlohu tak, že jej použijeme postupně pro každý uzel grafu v roli výchozího uzlu. Existuje však rychlejší postup, kterému se věnujeme podrobněji v následující kapitole.

Mezi další varianty problému nejkratších cest patří např. určit nejkratší cestu procházející zadanými vnitřními uzly nebo nalézt prvních k nejkratších cest mezi dvěma uzly, atd. Někdy mohou být časové a paměťové nároky hledání nejkratší cesty vlivem složitosti grafu tak vysoké, že se spokojíme s mírně suboptimální cestou, která se nalezne rychle a s rozumnými nároky na paměť (viz např. [30], [20]). Mezi postupy používané pro řešení takto formulovaných úloh patří algoritmy tzv. **heuristického hledání**, které tvoří součást metod umělé inteligence, jimž se věnujeme v samostatné kapitole 9.

Relaxace

Algoritmy použitelné pro řešení základní úlohy (hledání cest z jednoho uzlu) mají mnoho společného s prohledáváním grafu do šířky. Jedná se především o to, že i zde se prostřednictvím odkazů na předchůdce zachycuje struktura stromu nejkratších cest. Pro každý uzel u je informace o předchůdci obsahem složky $p[u]$ v jeho reprezentaci. Dále má každý uzel přiřazenu složku $d[u]$, která je aproximací jeho vzdálenosti od výchozího uzlu s. Inicializační část můžeme formulovat analogicky jako při prohledávání do šířky s tím rozdílem, že zatím nebudeme předpokládat rozlišení uzlů podle jejich stavu.

Algoritmus 6.4 Inicializace pro hledání cest
INIT-PATHS(G, s, w)

```
1  for každý uzel u ∈ U                 Všechny uzly grafu
2      do d[u] := ∞                     mají od s nekonečnou vzdálenost
3         p[u] := NIL                   a nemají předchůdce,
4  d[s] := 0                            jen s je výjimka.
```

Velmi podstatným rozdílem je však nutnost hodnoty $d[u]$ průběžně upravovat, neboť první nalezení uzlu ještě nazaručuje, že se k němu dospělo nejkratší cestou. Každé následující „znovuobjevení“ stejného uzlu u může způsobit snížení dosavadní hodnoty $d[u]$, neboť při určení w-vzdálenosti nehraje roli počet hran minimální cesty, ale součet jejich ohodnocení.

Snižování hodnoty $d[u]$ přitom vychází z následující základní úvahy: je-li $d[u]$ horní mezí skutečné vzdálenosti $d_w(s,u)$, potom podle vztahu (6.4) platí

$$d[u] + w(u,v) \geq d_w(s,u) + w(u,v) \geq d_w(s,v),$$

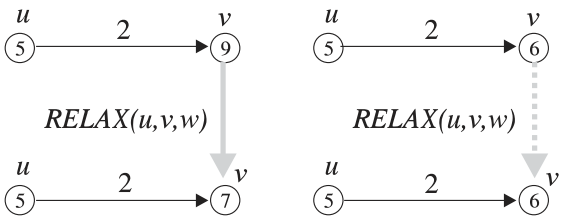
takže také hodnota $d[u]+w(u,v)$ pro libovolnou hranu (u,v) je horní mezí vzdálenosti $d_w(s,v)$. Protože hodnotu horní meze je možné zpřesňovat pouze tak, že ji snižujeme, můžeme novou hodnotu $d[v]$ určit jako $d[u] + w(u,v)$, jakmile bude nová hodnota menší, než byla hodnota předchozí. Tím jsme se dostali k elementární operaci označované jako **relaxace hrany** (u,v).

Algoritmus 6.5 Relaxace hrany

RELAX(u,v,w)

```
1   if d[v] > d[u] + w(u,v)
2       then d[v] := d[u] + w(u,v)
3            p[v] := u
```

Lze-li $d[v]$ zmenšit, pak to provedeme a také upravíme předchůdce.



Obrázek 6.2: Relaxace hrany

Na obr. 6.2 ukazujeme účinek relaxace ve dvou rozdílných situacích. V prvním případě je podmínka na řádce 1 operace RELAX splněna, takže po relaxaci je nová hodnota $d[v]$ menší. Ve druhém případě se relaxací hodnota $d[v]$ nezměnila. Jak je bezprostředně vidět, po provedení relaxace hrany (u,v) bude hodnota $d[v]$ zaručeně splňovat nerovnost $d[v] \leq d[u] + w(u,v)$. Mohla ji splňovat již před relaxací, a pak se nic nezměnilo, nebo platila nerovnost opačná, a pak se při relaxaci dosadila do $d[v]$ hodnota nová. Ukážeme, že intuitivní zdůvodnění relaxace bylo oprávněné.

Lemma 6.6: Nechť $G = \langle H, U\rangle$ je orientovaný graf s ohodnocením hran $w : H \mapsto R$ a $s \in U$ jeho zvolený uzel. Předpokládejme, že počáteční nastavení hodnot $d[u]$ proběhlo pomocí procedury INIT-PATHS(G,s,w). Potom platí $d[u] \geq d_w(s,u)$ pro všechny uzly $u \in U$ a tato nerovnost zůstane v platnosti po libovolném počtu relaxací hran grafu G. Navíc, jakmile $d[u]$ dosáhne své dolní meze $d_w(s,u)$, již se dále nemění.

Důkaz: (indukcí podle počtu n operací RELAX měnících hodnotu $d[v]$ – ostatní případy žádnou hodnotu nemění, a není proto třeba je uvažovat)
1. Pro $n = 0$ máme situaci po inicializaci, a tam platí $d[u] = \infty$ pro každý uzel $u \in U$ kromě s, pro který je $d[s] = 0$. Přitom $d_w(s,s)$ může být nejvýš rovna nule (nebo $-\infty$, pokud s leží na nějakém cyklu záporné w-délky), takže nerovnost platí.

2. Nechť nerovnost platí po provedení libovolných $n(\geq 0)$ relaxací hrany a uvažujme nějakou posloupnost $(n+1)$ relaxací, přičemž poslední z nich se provedla pro hranu (u,v). Potom při poslední relaxaci s použitím indukčního předpokladu na hodnotu $d[u]$ a vztahu (6.4) dostáváme

$$d[v] \;=\; d[u] + w(u,v) \;\geq\; d_w(s,u) + w(u,v) \;\geq\; d_w(s,v).$$

Protože $d[v]$ byla jediná měněná hodnota a pro všechny zbývající už požadovaná nerovnost platila podle indukčního předpokladu, je vztah dokázán. Poslední část tvrzení plyne z toho, že nastane-li někdy $d[u] = d_w(s,u)$, potom se díky platnosti $d[u] \geq d_w(s,u)$ nemůže hodnota $d[u]$ dále snižovat, a zůstává tedy stejná. △

Důsledek 1: Pro uzel v nedosažitelný z uzlu s se po provedení počátečního nastavení procedurou INIT-PATHS(G,s,w) a po libovolném počtu provedení operace RELAX hodnota $d[v] = \infty$ nezmění.

Důsledek 2: Pro uzel v dosažitelný z uzlu s nechť $\langle s, u_1, \ldots, u_k, u, v\rangle$ je nějaká minimální cesta z s do v. Předpokládejme dále, že po počátečním nastavení procedurou INIT-PATHS(G, s, w) se provedl nějaký počet relaxací včetně RELAX(u, v, w). Jestliže platilo $d[u] = d_w(s, u)$ před tímto provedením operace RELAX, tak po jejím provedení a nadále bude platit $d[v] = d_w(s, v)$.

Strom nejkratších cest

Až dosud jsme si podrobně všímali pouze hodnot složky $d[u]$ v průběhu relaxací hran grafu. Současně se změnou této složky dochází však při relaxaci i ke změně odkazu na předchůdce $p[u]$, takže je důležité zjistit, jaký bude konečný efekt těchto změn. Intuitivně očekáváme, že v okamžiku, kdy se prostřednictvím posloupnosti relaxací snížily hodnoty všech $d[u]$ na vzdálenosti $d_w(s, u)$, bude p-podles vyjádřený odkazy $p[u]$ představovat strom minimálních cest z uzlu s.

Věta 6.7: Nechť $G = \langle H, U\rangle$ je orientovaný graf s ohodnocením hran $w : H \mapsto \mathbf{R}$, buď $s \in U$ jeho zvolený uzel. Nechť graf G neobsahuje cykly se zápornou w-délkou dosažitelné z uzlu s. Potom po provedení počátečního nastavení procedurou INIT-PATHS(G, s, w) a provedení libovolného počtu relaxací představuje p-podgraf G_p kořenový strom s kořenem s. Navíc, jakmile se pomocí relaxací dosáhne hodnot $d[u] = d_w(s, u)$ pro všechny uzly $u \in U$, je p-podgraf G_p stromem minimálních cest v grafu G do všech uzlů dosažitelných z uzlu s.

Důkaz: Připomeňme nejprve, že p-podgraf $G_p = \langle H_p, U_p\rangle$ je indukován množinou hran $H_p = \{(p[v], v) \in H : p[v] \neq \text{ NIL}\}$. Jeho množina uzlů U_p obsahuje v průběhu relaxací pouze uzly dostupné z uzlu s (ne nutně všechny), neboť pouze pro ně se podle důsledku 1 lemmatu 6.6 může změnit hodnota $p[v]$, nastavená na počátku na NIL. Dokážeme nejprve sporem, že G_p nemůže obsahovat žádný cyklus.

Nechť tedy $C = \langle u_0, u_1, \ldots, u_k\rangle, u_0 = u_k$ je nějaký cyklus v G_p, tedy $p[u_i] = u_{i-1}$ pro $i = 1, 2, \ldots, k$. Můžeme předpokládat, že to byla právě relaxace hrany (u_{k-1}, u_k), která způsobila uzavření tohoto cyklu. Představme si nyní situaci na cyklu C těsně před provedením RELAX(u_{k-1}, u_k, w) – chceme dokázat, že C musí mít zápornou w-délku, a tak dosáhnout sporu.

Vzhledem k nastavení předchůdců $p[u_i] = u_{i-1}$ pro $i = 1, 2, \ldots, k-1$ musely být příslušnými relaxacemi nastaveny i složky $d[u_i]$ na hodnoty $d[u_{i-1}]+w(u_{i-1}, u_i)$ pro $i = 1, 2, \ldots, k-1$. Pokud se $d[u_{i-1}]$ od té doby změnilo, pak se mohlo jen snížit. Před provedením RELAX(u_{k-1}, u_k, w) tedy platí

$$d[u_i] \;\geq\; d[u_{i-1}] + w(u_{i-1}, u_i) \quad \text{pro všechna } i = 1, 2, \ldots, k-1. \tag{6.5}$$

Jelikož se provedením RELAX(u_{k-1}, u_k, w) mění hodnota $p[u_k]$, musí platit ostrá nerovnost

$$d[u_k] \;>\; d[u_{k-1}] + w(u_{k-1}, u_k). \tag{6.6}$$

Sečtením všech nerovností v (6.5) a (6.6) dostaneme následující bilanci pro hodnoty $d[u]$ uzlů cyklu C:

$$\sum_{i=1}^{k} d[u_i] \;>\; \sum_{i=1}^{k} (d[u_{i-1}] + w(u_{i-1}, u_i)) \;=\; \sum_{i=1}^{k} d[u_{i-1}] + \sum_{i=1}^{k} w(u_{i-1}, u_i)$$

Součty hodnot $d[u_i]$ a $d[u_{i-1}]$ na obou stranách jsou stejné, neboť se sčítá podél celého cyklu C, takže je můžeme od nerovnosti odečíst a dostáváme

$$0 > \sum_{i=1}^{k} w(u_{i-1}, u_i),$$

což je požadovaný spor.

Nyní bychom měli dokázat, že uzly v G_p jsou dostupné z uzlu s. Každý uzel $u \in U_p$ kromě s má ale jediného předchůdce $p[u]$, takže hodnotami $p[u]$ je určena jediná cesta z s do u (detailně by se to dokázalo indukcí). Zbývá dokázat, že po dosažení w-vzdáleností $d_w(s,u)$ ve složkách $d[u]$ obsahuje G_p právě všechny uzly dostupné z s a je pro ně stromem minimálních cest.

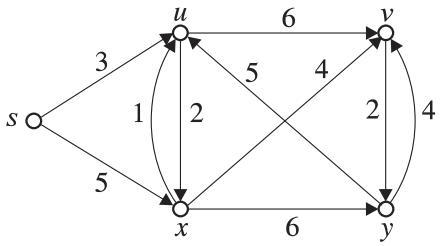
Uzlům, jimž se při relaxaci změnila hodnota $p[u]$, byla současně přiřazena konečná horní mez vzdálenosti $d[u]$, takže jsou to podle důsledku 1 lemmatu 6.6 právě uzly dostupné z uzlu s. Nechť $L = \langle u_0, u_1, \ldots, u_k \rangle, u_0 = s, u_k = u$ je jediná cesta z s do u v p-podgrafu G_p. Potom pro všechna $i = 1, 2, \ldots, k$ platí současně $d[u_i] = d_w(s,u_i)$ a $d[u_i] \geq d[u_{i-1}] + w(u_{i-1}, u_i)$, neboť už nelze relaxací změnit žádnou hodnotu $d[u_i]$. Potom ale platí $w(u_{i-1}, u_i) \leq d_w(s,u_i) - d_w(s,u_{i-1})$ a sumací podél cesty L dostaneme

$$\begin{aligned} w(L) &= \sum_{i=1}^{k} w(u_{i-1}, u_i) \leq \sum_{i=1}^{k} (d_w(s,u_i) - d_w(s,u_{i-1})) = \\ &= d_w(s,u_k) - d_w(s,u_0) = d_w(s,u_k). \end{aligned}$$

Délka žádné cesty v grafu G však nemůže být menší než vzdálenost jejích krajních uzlů, takže L je skutečně minimální cestou z s do u v grafu G. △

Přestože jsme o použití relaxace a jejích výsledcích vyslovili řadu tvrzení, stále nám chybí jednoduchý návod, jak hrany pro relaxaci vybírat, abychom dostali skutečně efektivní algoritmus hledání cest. Návrhu a analýze složitosti takových algoritmů se věnujeme v následujících dvou odstavcích.

Cvičení



Obrázek 6.3: Určení stromu minimálních cest

6.1-1. Určete alespoň dva různé stromy minimálních cest z uzlu s grafu na obr. 6.3.

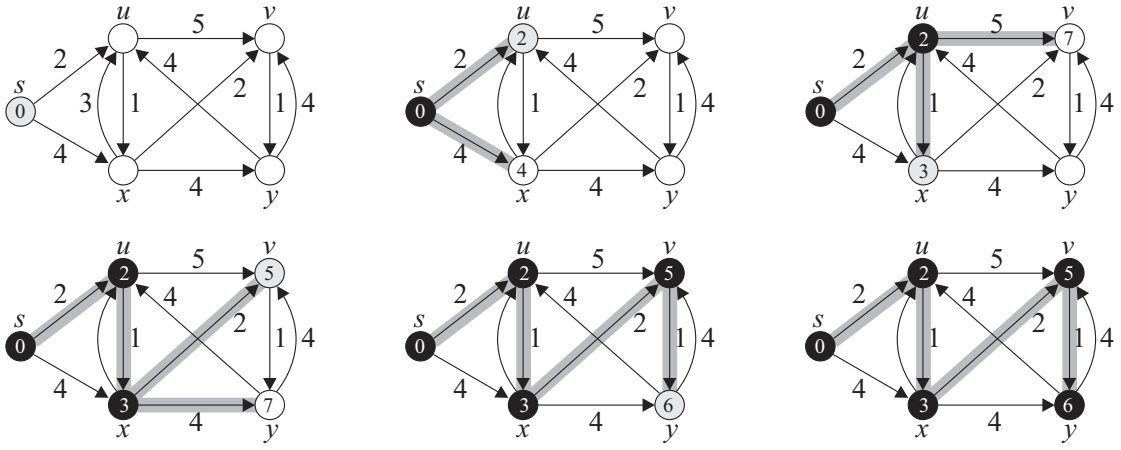
6.1-2. Nechť $G = \langle H, U \rangle$ je orientovaný graf s ohodnocením hran $w : H \mapsto \mathbf{R}$, buď $s \in U$ jeho zvolený uzel. Předpokládejme, že se provedlo počáteční nastavení procedurou INIT-PATHS(G, s, w) a libovolný počet relaxací. Dokažte, že pokud je hodnota odkazu $p[s]$ různá od NIL, potom graf G obsahuje cyklus se zápornou w-délkou.

6.1-3. Nechť $G = \langle H, U \rangle$ je orientovaný graf s ohodnocením hran $w : H \mapsto \mathbf{R}$, buď $s \in U$ jeho zvolený uzel. Nechť graf G neobsahuje cykly se zápornou w-délkou dosažitelné z uzlu s. Dokažte, že po počátečním nastavení procedurou INIT-PATHS(G, s, w) a provedení libovolného počtu relaxací představuje každá konečná hodnota $d[u]$ w-délku cesty z s do u určenou pomocí odkazů na předchůdce $p[u]$.

6.1-4. Nechť $G = \langle H, U \rangle$ je orientovaný graf s nezáporným ohodnocením hran $w : H \mapsto \mathbf{R}^+$, buď $s \in U$ jeho zvolený uzel. Nechť $p[u]$ pro každý dostupný uzel u odkazuje na jeho předchůdce na nějakém minimálním spojení z s do u. Uveďte příklad grafu a hodnot $p[u]$ takových, že přiřazené hodnoty $p[u]$ vytvářejí cyklus. (Podle věty 6.7 ovšem tato situace nemohla vzniknout žádnou posloupností relaxací.)

6.2 Dijkstrův algoritmus

Dijkstrův algoritmus je možné chápat jako určité zobecnění postupu prohledávání grafu do šířky. Také při něm se počínaje výchozím uzlem s šíří ve směru k dosud vůbec nebo jen „oklikou“ objeveným částem grafu systematicky posouvaná vlna. Tato vlna pohlcuje jen takové uzly, k nimž byla již nalezena nejkratší cesta.



Obrázek 6.4: Provedení Dijkstrova algoritmu

Důležitým předpokladem použitelnosti Dijkstrova algoritmu je, aby ohodnocení hran w mělo pouze nezáporné hodnoty. V tomto celém odstavci se tedy zabýváme řešením základní úlohy o vzdálenostech z jednoho uzlu do všech ostatních na orientovaném grafu $G = \langle H, U \rangle$ s nezáporným ohodnocením hran $w : H \mapsto \mathbf{R}^+$.

Základní kroky Dijkstrova algoritmu a údaje uchovávané o každém uzlu se kryjí s tím, co jsme podrobně popsali v předchozím odstavci. Jeho specifičnost spočívá pouze ve způsobu výběru hran pro relaxaci. Algoritmus postupně rozšiřuje množinu uzlů S, jejichž aproximovaná vzdálenost $d[u]$ se již shoduje se skutečnou w-vzdáleností $d_w(s, u)$. V analogii s prohledáváním do šířky bychom tyto uzly mohli označit jako uzavřené.

Zbývající uzly grafu G, tedy $U - S$, jsou umístěny v prioritní frontě Q seřazené vzestupně podle hodnot $d[u]$. Přestože se formálně nerozlišují otevřené a nové uzly, na začátku této fronty jsou vždy uzly otevřené, neboť jen ony mají přiřazenu konečnou hodnotu $d[u]$. Následující vyjádření Dijkstrova algoritmu předpokládá reprezentaci grafu pomocí seznamu následníků.

Algoritmus 6.8 Dijkstrův algoritmus hledání cest

DIJKSTRA(G, s, w)

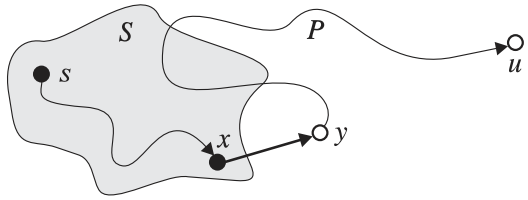
1	INIT-PATHS(G, s)	Počáteční nastavení $d[u], p[u]$
2	$S := \emptyset$	a zatím žádný zavřený uzel.
3	INIT-QUEUE(Q)	Všechny uzly grafu
4	**for** každý uzel $u \in U$	se uloží do fronty
5	**do** ENQUEUE(Q, u)	seřazené podle $d[u]$
6	**while** not EMPTY(Q)	Dokud něco zbývá,
7	**do** $u :=$EXTRACT-MIN(Q)	vyber uzel nejbližší k s,
8	$S := S \cup \{u\}$	přeřaď jej mezi uzavřené
9	**for** každý uzel $v \in Adj[u]$	a všem jeho následníkům
10	**do** RELAX(u,v,w)	uprav vzdálenost od s.

Použití Dijkstrova algoritmu ilustrujeme na obr. 6.4. Nejkratší cestu k dosud objeveným uzlům tvoří silně vytažené hrany, hodnoty $d[u]$ jsou uvedeny uvnitř uzlů. Uzly množiny S vyznačujeme tmavě, jednotlivé dílčí obrázky kromě prvního zachycují situaci po provedení všech relaxací pro vybraný uzel u. Je vidět, že při prvním průchodu cyklem **while** se jako nejbližší vybere právě uzel s, který má jako jediný přiřazenu konečnou (nulovou) hodnotu $d[u]$. Při relaxaci hrany (u, v) se v případě úpravy hodnoty $d[v]$ předpokládá zařazení uzlu v na odpovídající místo ve frontě Q. Do této fronty se však žádný uzel neukládá opakovaně, takže z ní bude vybrán právě jednou a hlavní cyklus na řádcích 6 – 10 se bude opakovat $|U|$-krát.

Srovnáme-li Dijkstrův algoritmus s algoritmem Jarníkovým-Primovým, neunikne nám jejich téměř shodná struktura. Jediný podstatnější rozdíl spočívá v tom, že při určení minimální kostry grafu se uzly řadí do fronty Q podle jejich vzdálenosti k dosud vytvořené dílčí kostře. Tato vzdálenost je vždy určena nejmenším ohodnocením hrany přecházející z nějakého uzlu dílčí kostry k uzlu ležícímu vně této kostry. U Dijkstrova algoritmu představuje kritérium řazení $d[u]$ (horní) odhad w-vzdálenosti $d_w(s,u)$, tedy délky nějaké minimální cesty, a jeho výsledkem je strom minimálních cest z uzlu s.

Shodná struktura uvedených dvou algoritmů prozrazuje, že i Dijkstrův algoritmus je jednou z variant hladového algoritmu. I v případě hledání nejkratších cest je splněna jak podmínka optimálnosti podstruktury (viz lemma 6.2), tak i podmínka hladového výběru (lokálně optimální volba vede ke globálně optimálnímu řešení). Ověření posledně uvedené podmínky je cílem následující věty, která je potvrzením správnosti Dijkstrova algoritmu.

Věta 6.9: (Správnost Dijkstrova algoritmu) Nechť se na orientovaném grafu $G = \langle H, U\rangle$ s nezáporným ohodnocením hran w provádí Dijkstrův algoritmus vycházející ze zvoleného uzlu s. Potom v okamžiku výběru každého uzlu u na řádce 7 algoritmu platí $d[u] = d_w(s,u)$ a tato hodnota se již nezmění.



Obrázek 6.5: Minimálnost $d[u]$

Důkaz: (indukcí podle počtu uzavřených uzlů, tedy uzlů vybraných z fronty Q)
1. První výběr z fronty Q se týká uzlu s, pro který je $d[s] = d_w(s,s) = 0$, tvrzení věty je tedy splněno.

2. Nechť tvrzení platí pro každý z prvních $n(\geq 1)$ uzavřených uzlů a uvažujme $(n+1)$-ní uzel u vybraný z fronty Q. Platnost tvrzení dokážeme sporem.

Nechť při výběru uzlu u platí $d[u] > d_w(s,u)$, pak je hodnota $d[u]$ konečná a v G existuje nějaká orientovaná cesta z s do u. Existuje tedy i nejkratší cesta P z s do u, označme jako y první uzel cesty P, který dosud nebyl uzavřen (může být i $u = y$), a jako x předchůdce uzlu y na cestě P (viz obr. 6.5).

Protože $x \in S$, všechny relaxace hran vystupujících z x již byly provedeny při uzavření uzlu x, takže podle důsledku 2 lemmatu 6.6 platí $d[y] = d_w(s,y)$ a tato hodnota se už nemůže změnit. Vzhledem k nezápornému ohodnocení hran (a tedy i každé w-délce nějaké cesty v G) nyní dostáváme

$$d[y] = d_w(s,y) \leq d_w(s,u) < d[u],$$

ale to je spor s podmínkou výběru uzlu u jako uzlu s minimální hodnotou $d[u]$ z dosud neuzavřených uzlů. △

Dijkstrův algoritmus tedy správně určí w-vzdálenosti od uzlu s, zbývá ještě stanovit jeho výpočetní složitost. Počáteční inicializace a cyklus na řádcích 4 – 5 budou trvat $O(|U|)$. Hlavní cyklus se provede $|U|$-krát a v rámci každého průchodu proběhne jednou výběr uzlu a zpracování všech jeho následníků. Předpokládáme-li realizaci prioritní fronty binární haldou, lze výběr uzlu provést v čase $O(\lg|U|)$, celkově tedy budou operace výběru trvat $O(|U|\lg|U|)$. Relaxace hrany se provádí celkem $|H|$-krát a při změně hodnoty $d[u]$ je třeba zařadit uzel u na odpovídající místo v prioritní frontě, což trvá $O(\lg|U|)$. Dohromady tedy dostáváme asymptotickou časovou složitost $O(|U|\lg|U| + |H|\lg|U|) = O((|U| + |H|)\lg|U|) = O(|H|\lg|U|)$.

Tento výsledek ukazuje, že binární haldu bude výhodné použít především pro řídké grafy, kdy je počet hran lineárně omezen počtem uzlů. Při použití Fibonacciho haldy je dokonce možné dále snížit celkovou složitost pro operaci výběru uzlů na $O(|U|\lg|U|)$ a na $O(|H|)$ pro úpravy pořadí při relaxacích, neboť jedna operace zařazení uzlu se sníženou hodnotou

$d[u]$ se pak provádí v konstantním čase. Celkově bychom tedy v tomto případě měli složitost $O(|U| \lg |U| + |H|)$.

Pro husté grafy (tj. $|H| = O(|U|^2)$) je ovšem možné zvážit i sekvenční implementaci fronty Q v poli. Výběr jednoho uzlu pak trvá $O(|U|)$, celkově tedy $O(|U|^2)$, naproti tomu snížení hodnoty $d[u]$ lze zajistit v konstantním čase. Celková asymptotická časová složitost Dijkstrova algoritmu je v tomto případě $O(|U|^2 + |H|) = O(|U|^2)$.

Cvičení

6.2-1. Simulujte provedení Dijkstrova algoritmu pro graf na obr. 6.3 s použitím alespoň dvou různých uzlů v roli výchozího uzlu. Průběh výpočtu s okamžitými hodnotami $d[u]$ a $p[u]$ na konci hlavního **while** cyklu vyjádřete podobně jako na obr. 6.4.

6.2-2. V důkazu věty 6.9 nalezněte důvod, proč Dijkstrův algoritmus nepřipouští hrany se zápornou délkou. Uveďte příklad grafu s některými záporně ohodnocenými hranami, pro který bude výsledek získaný Dijkstrovým algoritmem nesprávný.

6.2-3. Na závěr Dijkstrova algoritmu obsahuje prioritní fronta Q pouze jeden uzel, ale algoritmus končí až potom, co byl i tento uzel z fronty vybrán. Rozhodněte, zda je možné bez porušení správnosti algoritmu změnit podmínku cyklu na řádce 6 na tvar

6 **while** $|Q| > 1$

6.2-4. Nechť je v grafu $G = \langle H, U \rangle$ zadáno ohodnocení, které každé hraně (u, v) přiřazuje reálné číslo $w(u, v)$ z intervalu $\langle 0, 1 \rangle$. Toto ohodnocení vyjadřuje spolehlivost komunikační linky (u, v), tj. pravděpodobnost bezchybného přenosu po této hraně. Za předpokladu vzájemné nezávislosti těchto pravděpodobností navrhněte efektivní algoritmus určení nejspolehlivější komunikační cesty mezi dvěma uzly.

6.2-5. Nechť je v grafu $G = \langle H, U \rangle$ zadáno ohodnocení hran $w : H \mapsto \{0, 1, \ldots, K - 1\}$ pro nějaké nezáporné celé číslo K. Upravte Dijkstrův algoritmus tak, aby nalezl nejkratší cesty z jednoho do všech uzlů v tomto grafu v čase $O(K|U| + |H|)$.

6.2-6. Upravte řešení předchozího cvičení tak, aby jeho asymptotická časová složitost byla $O((|U| + |H|) \lg K)$.
(*Návod:* Kolik různých hodnot $d[u]$ může obsahovat fronta Q v libovolném okamžiku?)

6.3 Bellmanův-Fordův algoritmus

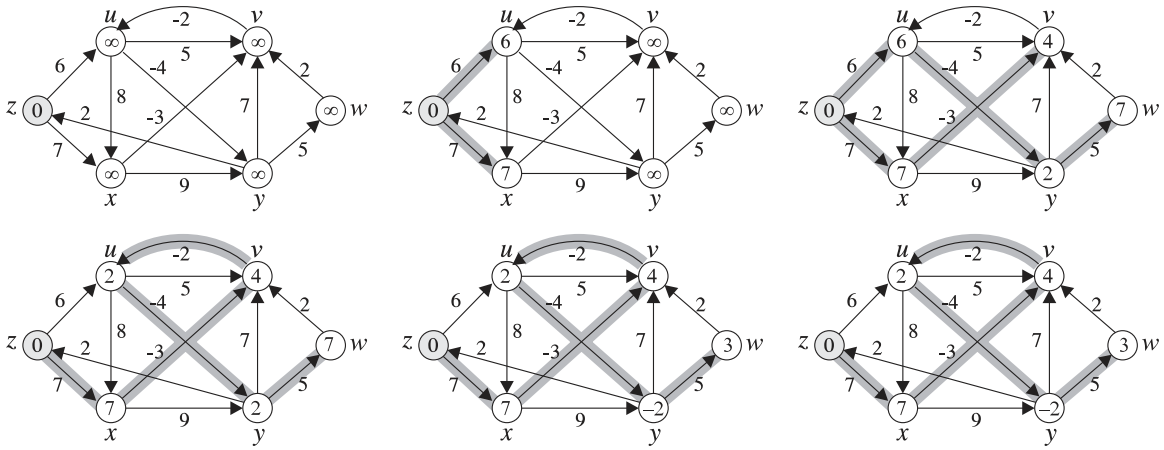
V případě grafů se záporně ohodnocenými hranami není Dijkstrův algoritmus použitelný, přitom w-vzdálenost z daného uzlu s zůstává korektně definována pro všechny uzly, k nimž vede z s orientovaná cesta neprocházející žádným cyklem se zápornou w-délkou. Pro takové případy zajišťuje Bellmanův-Fordův algoritmus nalezení nejkratších cest a současně dokáže zjistit, zda je splněna podmínka týkající se záporných cyklů.

I při formulaci Bellmanova-Fordova algoritmu použijeme techniku relaxace hran popsanou v odst. 6.7 systematicky snižující horní meze $d[u]$ vzdáleností $d_w(s, u)$ tak dlouho, až dosáhnou svého minima – tedy w-délky nejkratší cesty z s do u. Jediný rozdíl je ve výběru hran pro relaxaci: zatímco Dijkstrův algoritmus relaxoval vždy jen hrany vycházející z vybraného uzlu, v tomto případě se opakovaně relaxují systematicky všechny hrany. Následující procedura je současně koncipována jako test neexistence cyklů se zápornou w-délkou dostupných z uzlu s.

Algoritmus 6.10 Bellmanův-Fordův algoritmus hledání cest

BELLMAN-FORD(G, s, w)

1	INIT-PATHS(G, s)	Počáteční nastavení $d[u], p[u]$.
2	**for** $i := 1$ **to** $\|U\| - 1$ **do**	Opakovaně relaxuj



Obrázek 6.6: Provedení Bellmanova-Fordova algoritmu

```
3       for každou hranu (u, v) ∈ H          všechny hrany grafu.
4           do RELAX(u, v, w)
5   for každou hranu (u, v) ∈ H              Zkus nalézt hranu,
6       do if d[v] > d[u] + w(u, v)          která ještě zmenšuje d[v],
7             then return FALSE              pak je zde záporný cyklus.
8   return TRUE                              Záporný cyklus nebyl nalezen.
```

Použití Bellmanova-Fordova algoritmu ilustrujeme na obr. 6.6. Strukturu p-podgrafu ukazují silně vytažené hrany, hodnoty $d[u]$ jsou uvedeny uvnitř uzlů. Hrany se relaxují v abecedním pořadí, jednotlivé dílčí obrázky kromě prvního zachycují situaci po provedení jednoho průchodu hlavním cyklem. U Bellmanova-Fordova algoritmu nelze postupně vytvářet množinu uzlů, které již mají přesně určenu vzdálenost, neboť i při posledním průchodu hlavním cyklem se mohou změnit hodnoty $d[u]$ všech uzlů.

Asymptotická časová složitost Bellmanova-Fordova algoritmu je $O(|U|\cdot|H|)$, neboť relaxace hrany má konstantní složitost a provádí se $|U|\cdot|H|$-krát. Zaměříme se tedy na ověření správnosti tohoto algoritmu.

Věta 6.11: (Správnost Bellmanova-Fordova algoritmu) Nechť se na orientovaném grafu $G = \langle H, U\rangle$ s ohodnocením hran w provádí Bellmanův-Fordův algoritmus vycházející ze zvoleného uzlu s. Potom platí:

(a) Pokud G neobsahuje cyklus záporné w-délky dosažitelný z uzlu s, algoritmus skončí s výslednou hodnotou TRUE, pro všechny uzly $u \in U$ platí $d[u] = d_w(s,u)$ a prostřednictvím odkazů $p[u]$ je určen odpovídající strom nejkratších cest z uzlu s.

(b) Pokud G obsahuje cyklus záporné w-délky dosažitelný z uzlu s, pak algoritmus vrátí hodnotu FALSE.

Důkaz:
(a) Pro uzly nedosažitelné z uzlu s se podle důsledku 1 lemmatu 6.6 hodnota $d[u] = d_w(s,u)$ nikdy nezmění, uvažujme tedy pouze dostupné uzly. Indukcí podle i lze dokázat, že po i-tém průchodu hlavním cyklem budou již algoritmem správně spočteny vzdálenosti $d[u] = d_w(s,u)$ těch uzlů, k nimž vede minimální cesta tvořená nejvýše i hranami (viz cvič. 6.3-1). Jelikož ke každému dostupnému uzlu musí existovat nejkratší cesta tvořená nejvýše $|U|-1$ hranami, bude určitě po $|U|-1$ průchodech hlavním cyklem nalezena.
Odkazy $p[u]$ určují strom nejkratších cest podle věty 6.7, takže stačí dokázat, že algoritmus vrací výslednou hodnotu TRUE. Pro všechny hrany $(u,v) \in H$ ale platí (viz lemma 6.3)

$$d[v] \;=\; d_w(s,v) \;\leq\; d_w(s,u) + w(u,v) \;=\; d[u] + w(u,v),$$

takže podmínka v závěrečném cyklu algoritmu nebude pro žádnou hranu splněna a výsledná hodnota je TRUE.

(b) Nechť graf G obsahuje cyklus $C = \langle u_0, u_1, \ldots, u_k \rangle, u_0 = u_k$ záporné w-délky dosažitelný z uzlu s. Tvrzení dokážeme sporem. Budeme předpokládat, že algoritmus vrací výslednou hodnotu TRUE, takže i pro uzly na cyklu C musí platit $d[u_i] \leq d[u_{i-1}] + w(u_{i-1}, u_i)$ pro $i = 1, 2, \ldots, k$. Sečtením pro všechny uzly na cyklu dostaneme

$$\sum_{i=1}^{k} d[u_i] \;\leq\; \sum_{i=1}^{k} d[u_{i-1}] + \sum_{i=1}^{k} w(u_{i-1}, u_i).$$

Všechny hodnoty $d[u_i]$ jsou ale konečné (cyklus C je dosažitelný z uzlu s) a jejich součty na levé i pravé straně nerovnosti jsou stejné, neboť se oba týkají množiny uzlů cyklu C. Po odečtení těchto součtů dostáváme

$$0 \;\leq\; \sum_{i=1}^{k} w(u_{i-1}, u_i) \;=\; w(C),$$

což je ale ve sporu s tím, že C má zápornou w-délku. Výsledná hodnota tedy musela být v tomto případě FALSE. △

V každém průchodu hlavním cyklem Bellmanova-Fordova algoritmu se relaxují všechny hrany grafu. Je přitom zřejmé, že naději na splnění relaxační podmínky $d[v] > d[u] + w(u, v)$ mají jen ty hrany (u, v), pro jejichž počáteční uzel u se některou předchozí relaxací snížila hodnota $d[u]$ (za takové snížení považujeme i počáteční nastavení $d[s] = 0$ v rámci INIT-PATHS). Takové uzly můžeme evidovat např. tak, že je ukládáme do fronty. Vybírání z této fronty není tentokrát vázáno na hodnotu $d[u]$, takže obě operace s frontou budou mít konstantní složitost. Dostáváme tak následující modifikaci Bellmanova-Fordova algoritmu.

Algoritmus 6.12 Upravený Bellmanův-Fordův algoritmus hledání cest

BELLMAN-FORD-Q(G, s, w)

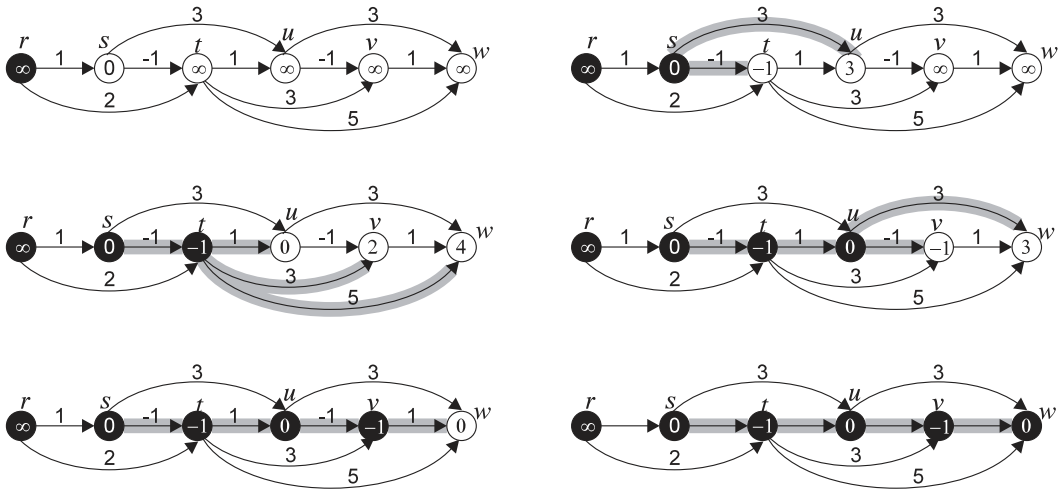
```
1   INIT-PATHS(G, s)                    Počáteční nastavení d[u], p[u]
2   INIT-QUEUE(Q); ENQUEUE(Q, s)        a uložení s do fronty Q.
3   while not EMPTY(Q)                  Dokud je něco ve frontě,
4       do u :=QUEUE-FIRST(Q)           vyber další uzel
5          for každý uzel v ∈ Adj[u]    a všechny vycházející hrany
6              do RELAX-Q(u, v, w)      relaxuj.
```

RELAX-Q(u, v, w)

```
1   if d[v] > d[u] + w(u, v)            Lze-li d[v] zmenšit,
2      then d[v] := d[u] + w(u, v)      pak to provedeme,
3           p[v] := u                   upravíme uzlu v předchůdce
4           ENQUEUE(Q, v)               a uložíme jej do fronty.
```

Na rozdíl od původní verze, kde byl počet průchodů hlavním cyklem explicite stanoven, je nyní ukončení algoritmu vázáno na vyprázdnění fronty Q. V případě existence cyklů se zápornou w-délkou dostupných z uzlu s se však fronta nikdy nevyprázdní, a výpočet by tedy neskončil. Tento nedostatek lze odstranit (viz cvič. 6.3-7), a tak získáme variantu se stejnými vlastnostmi ohledně záporných cyklů, jako měl původní algoritmus. Asymptotická výpočetní složitost této varianty je rovněž $O(|U| \cdot |H|)$, neboť v nejhorším případě se do fronty Q budou opakovaně dostávat všechny uzly grafu.

Starosti se zápornými cykly ovšem zcela odpadají u acyklických grafů, pro které lze navrhnout výrazně zjednodušenou variantu se složitostí $\Theta(|U| + |H|)$. V acyklickém grafu využijeme topologického uspořádání uzlů: pokud existuje orientovaná cesta z uzlu u do uzlu v, pak se



Obrázek 6.7: Hledání nejkratších cest na acyklickém grafu

uzel u nachází před uzlem v v topologickém uspořádání. To znamená, že stačí procházet uzly v tomto pořadí a pro každý uzel relaxovat všechny z něj vycházející hrany.

Algoritmus 6.13 Nejkratší cesty pro acyklický graf

DAG-PATHS(G, s, w)
1 Topologické uspořádání uzlů grafu G
2 INIT-PATHS(G, s)
3 **for** každý uzel u v pořadí jeho topologického uspořádání
4 **do for** každé $v \in Adj[u]$
5 **do** RELAX(u, v, w)

Průběh hledání nekratších cest v acyklickém grafu ukazujeme na obr. 6.7, topologické uspořádání uzlů je dáno jejich umístěním zleva doprava. Každý dílčí obrázek znázorňuje situaci po uzavření jednoho uzlu (t.j. po relaxaci hran z něho vycházejících), tučně vytažené hrany vyjadřují hodnoty $p[u]$.

Zdůvodnění časové složitosti této varianty vychází z toho, že topologické uspořádání na řádce 1 lze provést v čase $\Theta(|U| + |H|)$. Hlavní cyklus na řádcích 3 – 5 se provádí pro každý uzel právě jednou a výběr uzlu u trvá $O(1)$. Ve vnitřním cyklu se pak relaxuje každá hrana grafu právě jednou, a to opět v čase $O(1)$, neboť na rozdíl od Dijkstrova algoritmu se již uzly nemusí řadit do prioritní fronty.

Cvičení

6.3-1. Dokažte indukcí podle i, že po i-tém průchodu hlavním cyklem Bellmanova-Fordova algoritmu budou již správně spočteny vzdálenosti $d[u] = d_w(s, u)$ těch uzlů, k nimž vede minimální cesta tvořená nejvýše i hranami.

6.3-2. Nalezněte příklad orientovaného grafu s ohodnocením hran, pro který při určitém pořadí procházení hran algoritmem Bellmana-Forda dojde v posledním průchodu hlavním cyklem ke změně hodnot $d[u]$ všech uzlů. Může taková situace nastat u grafu neobsahujícího cyklus záporné w-délky?

6.3-3. Proveďte výpočet Bellmanova-Fordova algoritmu pro graf na obr. 6.6 s tím, že za výchozí uvažujete postupně každý uzel grafu.

6.3-4. Nechť je pro orientovaný graf $G = \langle H, U \rangle$ s ohodnocením hran w známo, že neobsahuje cykly se zápornou w-délkou a pro libovolnou dvojicí uzlů $u, v \in U$ lze nalézt nejkratší (ve

smyslu w-délek) cestu mezi nimi tvořenou nejvýše k hranami. Navrhněte úpravu Bellmanova-Fordova algoritmu, která umožní získat w-vzdálenosti ze zadaného výchozího uzlu po $(k+1)$ průchodech hlavním cyklem.

6.3-5. Upravte Bellmanův-Fordův algoritmus tak, aby nastavil hodnoty $d[u] = -\infty$ všem uzlům u, pro které existuje cesta z s do u procházející nějakým cyklem se zápornou w-délkou.

6.3-6. Nechť je pro orientovaný graf $G = \langle H, U \rangle$ s ohodnocením hran w známo, že obsahuje cyklus se zápornou w-délkou dostupný z uzlu $s \in U$. Navrhněte efektivní algoritmus, který určí všechny uzly ležící na nějakém takovém cyklu.

6.3-7. Dokažte, že graf G obsahuje cyklus se zápornou w-délkou dostupný z uzlu s právě tehdy, pokud se algoritmem BELLMAN-FORD-Q uloží a opět vybere do fronty Q nějaký uzel více než $|U|$-krát. Použijte tuto vlastnost k doplnění algoritmu BELLMAN-FORD-Q tak, aby i v tomto případě skončil a signalizoval existenci záporného cyklu.

6.3-8. Pomocí vhodné úpravy algoritmu hledání nejkratších cest v acyklickém grafu určete vzdálenosti uzlu w od všech ostatních uzlů v grafu na obr. 6.7.

6.3-9. Ukažte, že algoritmus hledání nejkratších cest v acyklickém grafu bude správně pracovat, i když cyklus na řádce 3 ukončíme po uzavření prvních $|U| - 1$ uzlů.

6.3-10. Navrhněte efektivní algoritmus, který zjistí celkový počet cest v orientovaném acyklickém grafu.