Kapitola 12

Ohodnocené grafy

V praktických aplikacích teorie grafů zpravidla graf slouží jako nástroj k popisu nějaké struktury. Jednotlivé prvky této struktury mají často přiřazeny nějaké hodnoty (může jít např. o parametry součástek v elektrickém obvodu, délky železničních tratí, ceny za přepravu jednotky zboží nebo propustnosti datových spojů). Zadáním pak je realizovat nějaký cíl (třeba přepravit zboží) optimálním způsobem. Úlohy tohoto typu se nazývají *optimalizační úlohy* a my se v této kapitole s několika z nich seznámíme. Popíšeme přitom několik důležitých algoritmů, zejména Dijkstrův algoritmus pro hledání minimální cesty a hladový algoritmus pro hledání minimální kostry.

12.1 Definice ohodnocených grafů

Uvažme graf, jehož vrcholy jsou evropská města a hrany jsou železniční tratě, které je spojují. Chceme-li najít nejkratší cestu vlakem dejme tomu z Budapešti do Paříže, není ani tak důležité, kolik hran našeho grafu bude tato cesta obsahovat — zajímá nás spíše, kolik kilometrů bude měřit. Bude tedy nutné přidat do grafu dodatečnou informaci o 'délce' jednotlivých hran. Dostaneme tzv. ohodnocený graf.

Definice 12.1 *Ohodnocený orientovaný graf* (G, w) je orientovaný graf G spolu s reálnou funkcí $w : E(G) \to (0, \infty)$. Je-li e hrana grafu G, číslo $w(e)$ se nazývá její *ohodnocení* nebo *váha*.

Podobně je definována neorientovaná verze ohodnoceného grafu. I v tomto oddílu budeme hovořit především o orientovaných grafech, s tím, že k neorientovanému případu lze vždy přejít přes symetrickou orientaci. Ohodnocení hran symetrické orientace grafu (G, w) je přirozeně odvozeno z původního ohodnocení (obě protichůdné hrany vzniklé z neorientované hrany e dostanou ohodnocení $w(e)$).

Často je vhodné uvažovat grafy bez ohodnocení jako speciální případy ohodnocených grafů, v nichž je váha každé hrany rovna jedné. V rámci této představy můžeme definovat následující zobecnění matice sousednosti.

Definice 12.2 *Vážená matice sousednosti* ohodnoceného orientovaného grafu (G, w) s vrcholy $v_1, \ldots, v_n$ je matice $W(G) = (w_{ij})$, kde

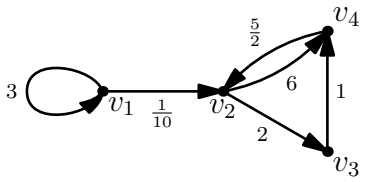
$$w_{ij} = \begin{cases} w(v_i v_j) & \text{pokud } v_i v_j \in E(G), \\ 0 & \text{jinak,} \end{cases}$$

pro $i, j = 1, \ldots, n$.

Všimněme si jedné důležité věci: nezáporné čtvercové matice jednoznačně odpovídají ohodnoceným orientovaným grafům. Například matici

$$W = \begin{bmatrix} 3 & \frac{1}{10} & 0 & 0 \\ 0 & 0 & 2 & 6 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{5}{2} & 0 & 0 \end{bmatrix}$$

odpovídá graf na obr. 12.1.



Obrázek 12.1: Ohodnocený orientovaný graf G popsaný maticí W.

Přirozeným zobecněním délky cesty v neohodnoceném grafu je její váha.

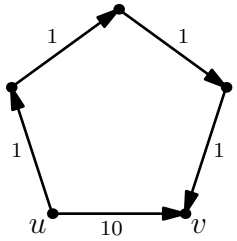
Definice 12.3 Nechť M je množina hran ohodnoceného orientovaného grafu (G, w). *Váha* $w(M)$ množiny M je součet vah jednotlivých hran $e \in M$. Pro stručnost definujeme *váhu* $w(P)$ *cesty* P jako váhu její množiny hran.

Jsou-li u, v vrcholy grafu G, pak *minimální cesta* z u do v je každá cesta, jejíž váha je minimální (tj. žádná cesta z u do v nemá menší váhu). *Vážená vzdálenost* $d^w(u, v)$ vrcholů u, v je váha minimální cesty z u do v.

Poznamenejme, že podle této definice je speciálně $d^w(u, u) = 0$. Dále si všimněme, že minimální cesta z u do v zdaleka nemusí být nejkratší, pokud jde o počet hran. Příkladem je graf na obr. 12.2.

Cvičení

► **12.1** Které matice odpovídají ohodnoceným neorientovaným grafům?



Obrázek 12.2: Minimální cesta z u do v nemusí být nejkratší.

12.2 Dijkstrův algoritmus

Jak lze najít váženou vzdálenost vrcholů v ohodnoceném grafu? Asi nejznámějším postupem je *Dijkstrův*[1] *algoritmus*, který si ukážeme v tomto oddílu. Vstupem algoritmu je ohodnocený orientovaný graf (G, w) a vrchol $v \in V(G)$. Jeho výstupem je pro každý vrchol x:

(1) vážená vzdálenost $d^w(v, x)$ vrcholů v a x,

(2) (nějaká) minimální cesta P_x z v do x.

'Počítá' tedy mnohem víc než jen váženou vzdálenost dané dvojice vrcholů.

Algoritmus proběhne v nejvýše n krocích. V celém jeho průběhu bude definován strom T, jehož vrcholy tvoří podmnožinu $V(G)$ (na počátku to bude jediný vrchol v, postupně se T rozšíří až na kostru grafu G). Pro každý vrchol $x \in V(G)$ bude dále určen (horní) *odhad vzdálenosti* $c(x)$. I toto číslo se typicky bude měnit a po dokončení algoritmu bude rovno vážené vzdálenosti $d^w(v, x)$. Pro všechny vrcholy z s konečnou nenulovou hodnotou $c(z)$ bude definován jejich *předchůdce* $\bar{z}$.

I. Příprava: T budiž strom na jediném vrcholu v. Položíme $c(v) := 0$. Pro sousedy z vrcholu v položíme

$$c(z) = w(vz) \text{ a } \bar{z} = v.$$

Pro všechny ostatní vrcholy $x \in V(G)$ definujeme $c(x) := \infty$. Předchůdce není určen pro žádný z těchto vrcholů x ani pro vrchol v.

II. Jeden krok algoritmu: Pokud T je kostra grafu G nebo každý vrchol $z \notin V(T)$ má $c(z) = \infty$, přejdeme na bod III. Jinak mezi vrcholy $z \notin V(T)$ vybereme vrchol x, který má minimální hodnotu $c(x)$. Je-li jich víc, zvolíme mezi nimi libovolně. Přidáme do stromu T vrchol x a hranu $\bar{x}x$ (předchůdce vrcholu x je jistě definován, protože $c(x)$ je konečné).

Dále pro všechny vrcholy $z \notin V(T)$, pro něž je xz hranou grafu G, přepočítáme odhad vzdálenosti: je-li $c(x) + w(xz) < c(z)$, položíme $c(z) := c(x) + w(xz)$ a rovněž $\bar{z} = x$. Opakujeme bod II.

[1] Edsger W. Dijkstra (1930–2002).

III. Konec: Pro každý vrchol $x \in V(T)$ je hledanou minimální cestou P_x cesta z v do x ve stromu T (která je jednoznačně určena). Její váha určuje váženou vzdálenost $d^w(v,x)$. Pro ostatní vrcholy x žádná cesta z v do x neexistuje a $d^w(v,x) = \infty$.

Odhadněme nyní pro Dijkstrův algoritmus jeho časovou složitost, jak jsme ji definovali v oddílu 6.7. Přípravná fáze bude pro graf na n vrcholech vyžadovat čas $O(n)$ (pro každý z n vrcholů potřebujeme nastavit odhad vzdálenosti a předchůdce). Krok II. bude proveden maximálně n-krát, protože pro každý vrchol se provádí nejvýše jednou. Jakou časovou složitost má jedno jeho provedení?

Nejprve vybíráme vrchol $x \in V(G) - V(T)$ s minimálním konečným odhadem vzdálenosti. K tomu stačí n operací. Dalších $O(n)$ operací nám zabere přepočítání odhadů vzdálenosti a úprava předchůdců u sousedů vrcholu x. Na jeden krok II. tak stačí $O(n)$ operací. Zjišťujeme tedy, že časová náročnost Dijkstrova algoritmu je $O(n^2)$.

Dijkstrův algoritmus lze použít i pro testování souvislosti neohodnoceného grafu G. Stačí každou hranu symetrické orientace ohodnotit vahou 1 a libovolně zvolit vrchol v. Strom T, nalezený Dijkstrovým algoritmem, je kostrou grafu G, právě když G je souvislý graf. (Platí dokonce, že vrcholy stromu T tvoří komponentu obsahující vrchol v.) Podobným postupem lze testovat i silnou souvislost.

Cvičení

▶ **12.2** Najděte Dijkstrovým algoritmem minimální cestu z vrcholu u do vrcholu v v grafech na obr. 12.3.

▶ **12.3** Najděte Dijkstrovým algoritmem minimální cestu z vrcholu u do vrcholu v v grafech na obr. 12.4.

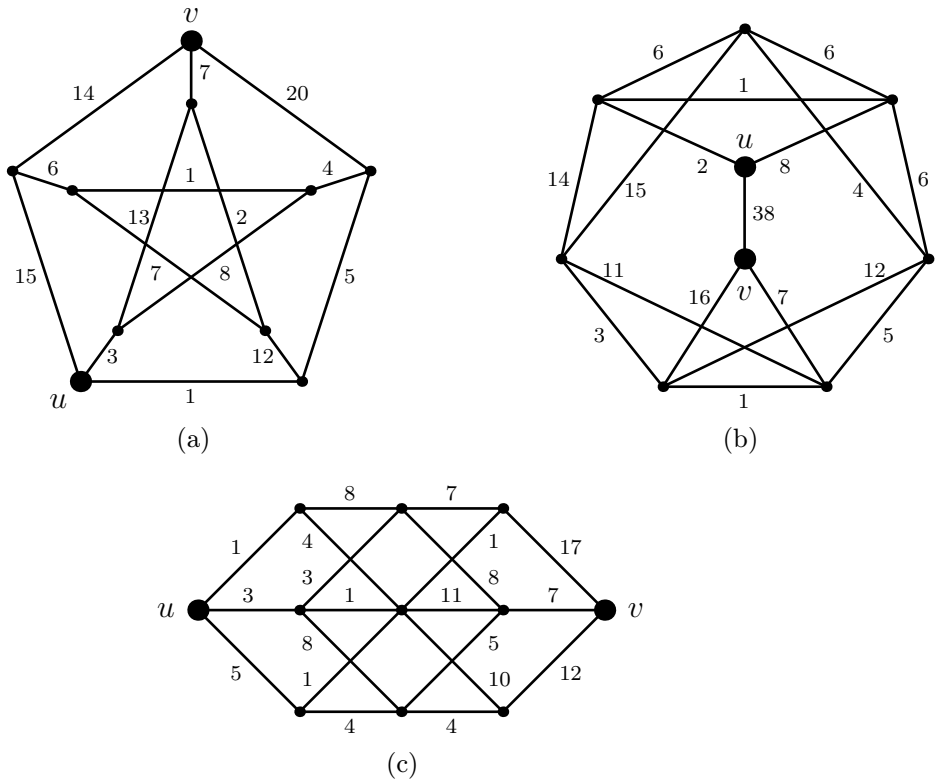
12.3 Matice vážených vzdáleností

K zachycení vážených vzdáleností všech dvojic vrcholů v ohodnoceném orientovaném grafu slouží následující matice.

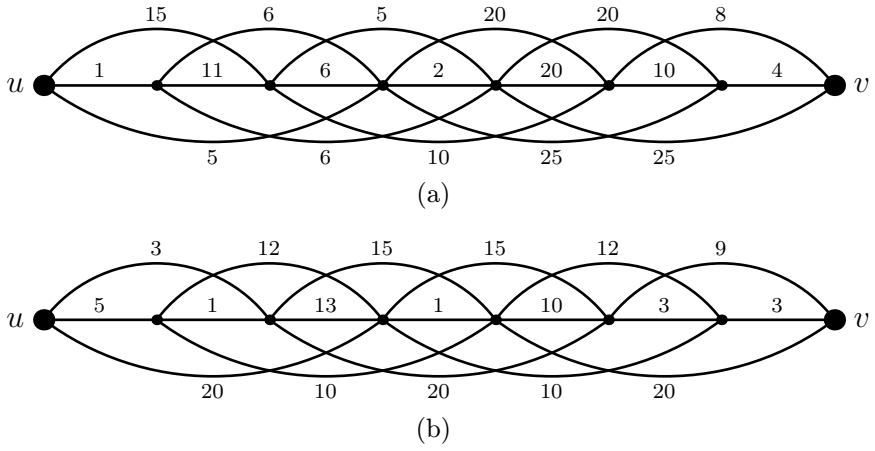
Definice 12.4 *Matice vážených vzdáleností* (též *w-distanční matice*) ohodnoceného orientovaného grafu (G, w) s vrcholy $v_1, \ldots, v_n$ je matice $D^w(G)$ o rozměrech $n \times n$, kde

$$D^w(G) = (d^w(v_i, v_j))_{i,j=1}^n.$$

Jednou možností, jak ji sestrojit, je použít Dijkstrův algoritmus. Ten nám libovolný její řádek najde v čase $O(n^2)$, takže celkový čas výpočtu je $O(n^3)$.



Obrázek 12.3: Najděte minimální cestu z u do v.



Obrázek 12.4: Najděte minimální cestu z u do v.

Ukážeme si jinou jednoduchou metodu, která pracuje v o něco horším čase $O(n^4)$. Nejprve pomocná definice. Nechť je dáno $k \geq 1$. Cesta z v_i do v_j je *k-minimální*, pokud její délka je nejvýše k a pokud žádná jiná cesta z v_i do v_j o délce nejvýše k nemá menší váhu.

Označme jako D_k matici, jejíž prvek d_{ij}^k na pozici (i,j) je roven váze k-minimální cesty z v_i do v_j (případně má hodnotu ∞, pokud taková cesta neexistuje). Vzhledem k tomu, že každá cesta má délku nejvýše $n-1$, musí být matice D_{n-1} rovna hledané w-distanční matici $D^w(G)$. Otázkou tedy je, jak matice D_k sestrojit. Pro $k=1$ je to snadné — pro prvky d_{ij}^1 zjevně platí:

$$d_{ij}^1 = \begin{cases} 0 & \text{pro } i=j, \\ w(v_iv_j) & \text{pro } ij \in E(G), \\ \infty & \text{jinak.} \end{cases}$$

Matice D_1 tedy bude téměř shodná s váhovou maticí $W(G)$, až na to, že nuly *mimo hlavní diagonálu* jsou v matici D_1 nahrazeny hodnotami ∞.

Dejme tomu, že již známe matici D_k a chceme sestrojit matici D_{k+1}. Začněme pozorováním. Nechť P je k-minimální cesta z v_i do v_j. Odebráním posledního vrcholu vznikne cesta P' z v_i řekněme do v_p. Tato cesta musí být $(k-1)$-minimální, jinak bychom dostali spor s k-minimalitou cesty P. Odtud vidíme, že pro délku d_{ij}^k k-minimální cesty z v_i do v_j platí

$$d_{ij}^k = d_{ip}^{k-1} + w(v_pv_j)$$

pro nějaký vrchol v_p s vlastností $v_pv_j \in E(G)$. Přestože předem nevíme, o který vrchol v_p se jedná, můžeme psát

$$d_{ij}^k = \min_{\ell=1}^{n} \left(d_{i\ell}^{k-1} + d_{\ell j}^1 \right), \tag{12.1}$$

přičemž využíváme fakt, že pokud $v_\ell v_j \notin E(G)$, pak $d_{\ell j}^1 = \infty$, takže v minimu bude daný součet zanedbán.

Všimněme si, že vzorec (12.1) nápadně připomíná definici násobení matic — pouze obsahuje minimum na místě sumy a součet na místě součinu. Skutečně, pokud na reálných číslech 'předefinujeme' sčítání a násobení předpisem

$$x \boxplus y = \min(x,y),$$
$$x \boxdot y = x + y,$$

pak platí, že matice D_{k+1} je součinem matic D_k a D_1 vzhledem k operacím $\boxplus$ a $\boxdot$. Indukcí snadno dostaneme následující větu.

Věta 12.5 *Matice D_k je k-tou mocninou matice D_1 vzhledem k operacím $\boxplus$ a $\boxdot$.* □

Vidíme, že ke spočítání matice $D^w(G)$ stačí provést $n-2$ 'vynásobení' maticí D_1. Následující tvrzení říká, že tento proces lze navíc přerušit již v okamžiku, kdy se 'vynásobením' matice poprvé nezměnila.

Tvrzení 12.6 *Pokud pro nějaké $q \geq 1$ platí $D_{q+1} = D_q$, pak D_q je hledanou maticí $D^w(G)$.*

Důkaz. Dokážeme indukcí, že za daného předpokladu pro všechna $k > q$ platí

$$D_k = D_q. \qquad (*)$$

Pro $k = q+1$ je tvrzení pravdivé. Dejme tomu, že je chceme dokázat pro dané $k \geq q+1$ za předpokladu, že pro $k-1$ tvrzení (*) platí.

Podle indukčního předpokladu můžeme člen $d_{i\ell}^{k-1}$ ve vzorci (12.1) nahradit hodnotou $d_{i\ell}^{q}$. Dostáváme tak rovnost $d_{ij}^{k} = d_{ij}^{q+1}$. Proto $D_k = D_{q+1}$ a tím pádem $D_k = D_q$. Tvrzení (*) je tak dokázáno pro všechna $k > q$.

Důkaz je u konce, jakmile si všimneme, že pro každé $k \geq n-1$ je $D_k = D^w(G)$. □

Cvičení

▶ **12.4** Najděte matici vážených vzdáleností $D^w(\vec{G})$ ohodnoceného orientovaného grafu $\vec{G}$, který je dán následující maticí sousednosti:

(a)
$$W(\vec{G}) = \begin{bmatrix} 0 & 3 & 1 & 0 \\ 1 & 0 & 0 & 3 \\ 8 & 2 & 0 & 2 \\ 0 & 7 & 6 & 0 \end{bmatrix},$$

(b)
$$W(\vec{G}) = \begin{bmatrix} 0 & 0 & 1 & 9 & 7 \\ 9 & 0 & 10 & 3 & 3 \\ 4 & 7 & 0 & 4 & 11 \\ 0 & 5 & 2 & 0 & 0 \\ 6 & 6 & 0 & 1 & 0 \end{bmatrix},$$

(c)
$$W(\vec{G}) = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 4 & 0 & 1 & 2 & 3 \\ 3 & 4 & 0 & 1 & 2 \\ 2 & 3 & 4 & 0 & 1 \\ 1 & 2 & 3 & 4 & 0 \end{bmatrix}.$$

(d)

$$W(\vec{G}) = \begin{bmatrix} 0 & 2 & 4 & 0 & 6 \\ 6 & 0 & 1 & 6 & 14 \\ 9 & 2 & 0 & 3 & 6 \\ 0 & 3 & 4 & 0 & 1 \\ 8 & 5 & 8 & 2 & 0 \end{bmatrix}.$$

12.4 Minimální kostra

Téma tohoto oddílu úzce souvisí s pojmem minimální cesta. Omezíme se na ohodnocené *neorientované* grafy.

Dejme tomu, že potřebujeme propojit n měst rozvodem elektrické energie. Pro každá dvě sídla známe náklady na přímé spojení a chceme, aby celková cena byla co nejnižší. Situaci lze modelovat grafem G, jehož vrcholy odpovídají městům a váhy hran cenám spojení. Uvažované rozvody energie odpovídají souvislým faktorům grafu G (musí totiž být také 'souvislé' a pokrývat všechna města). Faktor, který obsahuje nějakou kružnici, nemůže určovat nejlevnější řešení, neboť odstraněním libovolné hrany této kružnice snížíme cenu a faktor zůstane souvislý. Hledaný faktor F je tedy kostrou grafu G, a to kostrou, která má nejmenší možnou váhu.

Definice 12.7 Kostra T ohodnoceného neorientovaného grafu G je *minimální*, pokud má mezi všemi kostrami minimální váhu $w(T) = w(E(T))$ (viz definice 12.3).

Uvedeme tzv. *hladový algoritmus* pro hledání minimální kostry souvislého grafu G, který jako první formuloval český matematik Otakar Borůvka (1899–1995). Zaveďme následující označení: je-li H graf a e hrana s koncovými vrcholy z $V(H)$, pak $H + e$ je graf, který vznikne přidáním hrany e ke grafu H.

Algoritmus pracuje v $n - 1$ krocích. V i-tém kroku je definován faktor L_i, který neobsahuje kružnice. (Protože graf bez kružnic je disjunktním sjednocením stromů, označujeme ho jako *les*.) Výsledný faktor L_{n-1} je, jak uvidíme, minimální kostrou.

Výchozí les L_0 sestává z izolovaných vrcholů, tj. $E(L_0) = \emptyset$. Uvažme i-tý krok algoritmu. Naší snahou je rozšířit faktor L_{i-1} přidáním jedné hrany na faktor L_i. Některé hrany ale přidat nemůžeme, protože bychom vytvořili kružnici. Z hran, které přidat lze, vyberme hranu e_i s nejmenší vahou[2] a položme

$$L_i = L_{i-1} + e_i.$$

Musíme ještě ukázat, že hrana e_i s požadovanou vlastností vůbec existuje. To je snadné: graf L_{i-1} má n vrcholů a méně než $n - 1$ hran. Podle věty 6.11 je

[2] Algoritmus bere v každém kroku 'nejlehčí' hranu, která přichází v úvahu — odtud název *hladový*.

tedy nesouvislý. Graf G je ovšem souvislý, takže mezi některými dvěma komponentami grafu L_{i-1} vede alespoň jedna hrana. Přidáním této hrany kružnici jistě nevytvoříme.

Popsali jsme způsob nalezení kostry L_{n-1}. Dokažme nyní, že tato kostra je minimální.

Věta 12.8 *Nechť G je ohodnocený neorientovaný graf. Kostra L_{n-1}, nalezená hladovým algoritmem, je minimální kostrou grafu G.*

Důkaz. Nechť $L = L_{n-1}$ je kostra grafu G, získaná hladovým algoritmem. Pro důkaz sporem předpokládejme, že není minimální kostrou, a ze všech minimálních koster zvolme takovou kostru M, která má s kostrou L společný největší počet hran.

Vezměme nejmenší i, pro které platí, že $e_i \notin E(M)$ (připomeňme, že e_i je hrana přidávaná v i-tém kroku algoritmu). Pak jistě $E(L_{i-1}) \subset E(M)$. Protože koncové vrcholy x, y hrany e_i jsou ve stromu M spojeny právě jednou cestou (věta 7.3), graf $M + e_i$ obsahuje právě jednu kružnici C.

Nechť f je libovolná hrana kružnice C, která není obsažena ve stromu L. Graf $L_{i-1} + f$ je podgrafem kostry M, a neobsahuje tedy kružnice. Přidání hrany f tak přichází v úvahu v i-tém kroku hladového algoritmu. Protože byla místo ní přidána hrana e_i, musí být

$$w(e_i) \leq w(f). \tag{12.2}$$

Odstraněním hrany f z grafu $M + e_i$ zanikne jediná jeho kružnice. Výsledný graf M' je tedy stromem, a tím pádem kostrou grafu G. Díky nerovnosti (12.2) není jeho váha větší než váha minimální kostry M. Přitom platí

$$|E(M') \cap E(L)| > |E(M) \cap E(L)|,$$

což je ve sporu s výběrem kostry M. Tento spor ukazuje, že kostra L je minimální. □

Jaká je časová složitost hladového algoritmu? Ukážeme, že při vhodné implementaci jím lze minimální kostru najít v čase $O(mn)$, kde m je počet hran a n počet vrcholů vstupního grafu G. (Složitost je v tomto případě vyjádřena jako funkce dvou proměnných, ne jenom počtu vrcholů n.)

Počet kroků algoritmu je $n-1$, protože v každém kroku přidáme jednu z hran kostry L. V každém kroku musíme pro každou z $O(m)$ zbývajících hran zjistit, zda jejím přidáním vznikne kružnice. To by při nešikovné implementaci vyžadovalo čas $O(n)$, takže celková složitost této implementace by byla $O(mn^2)$. Lze však ušetřit pomocí následujícího vylepšení, známého jako *Kruskalův algoritmus*.

Seřadíme vrcholy do posloupnosti $v_1, \ldots, v_n$. Pro každý vrchol v_j budeme udržovat číslo $m(v_j)$, které udává minimum z indexů všech vrcholů, které leží ve stejné komponentě jako v_j. V rámci přípravné fáze algoritmu pro každé j položíme $m(v_j) := j$. Po každém provedeném kroku algoritmu přepočítáme hodnoty $m(v_j)$

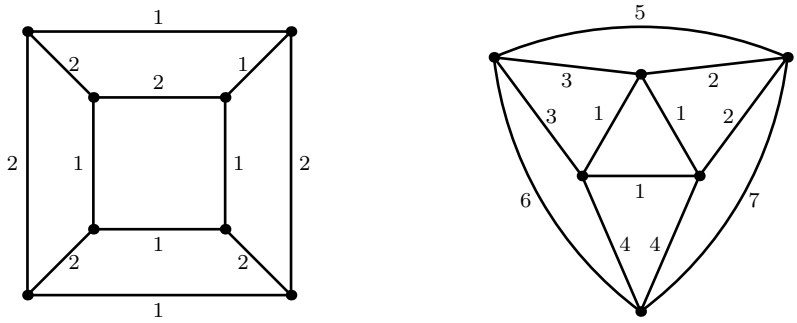
(přidáním hrany se spojí dvě komponenty faktoru L_i, takže v jedné z nich je třeba hodnoty upravit). Ke zjištění, zda přidáním hrany v_jv_k vznikne kružnice, nyní stačí porovnat hodnoty $m(v_j)$ a $m(v_k)$: kružnice vznikne právě tehdy, když jsou shodné.

Analyzujme složitost vylepšené verze algoritmu. Přípravná fáze zabere $O(n)$ operací na inicializaci hodnot $m(v_j)$. Následuje $O(n)$ kroků algoritmu. V rámci každého z nich je třeba pro každou z $O(m)$ hran provést konstantní počet[3] operací (porovnání hodnot $m(v_j)$). Z hran, které lze přidat, vybereme tu s minimální vahou, a *poté* v čase $O(n)$ přepočítáme hodnoty $m(v_j)$. Dostáváme tedy celkový odhad $O(mn)$.

Cvičení

▶ **12.5** Je pravda, že je-li T minimální kostra ohodnoceného neorientovaného grafu G a $u, v \in V(G)$, pak cesta z u do v v kostře T je minimální cestou z u do v v G? Dokažte nebo najděte protipříklad.

▶ **12.6** Pomocí hladového algoritmu najděte minimální kostry grafů na obrázku 12.5.



Obrázek 12.5: Najděte minimální kostry.

12.5 Problém obchodního cestujícího

Všechny úlohy, které jsme zatím v kapitole 12 zkoumali, byly řešitelné efektivními algoritmy. Nyní stručně popíšeme problém, u kterého se situace zdá být odlišná. Jedná se o tzv. *úlohu obchodního cestujícího*.

[3]Konstantní zde znamená 'nezávislý na m a n'. Konstantní veličinu lze v rámci používaného formalismu značit symbolem $O(1)$.

Obchodní cestující má za úkol objet všechna města v daném kraji, a to po nejkratší možné trase (měřeno počtem ujetých kilometrů). Každé město musí navštívit právě jednou a má skončit ve výchozím bodě.

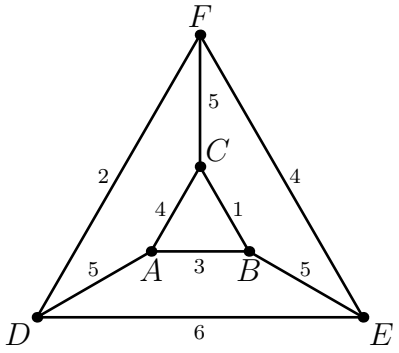
Reformulace problému v jazyce teorie grafů je nasnadě: zadáním je ohodnocený neorientovaný graf, jehož vrcholy odpovídají městům, hrany dvojicím měst s přímým silničním spojením, a váha každé hrany je vzdálenost příslušné dvojice měst. Hledaným řešením je hamiltonovská kružnice s nejmenší možnou vahou (*optimální* hamiltonovská kružnice).

Pro grafy malé velikosti lze řešení poměrně rychle najít 'hrubou silou' jako v následujícím příkladu.

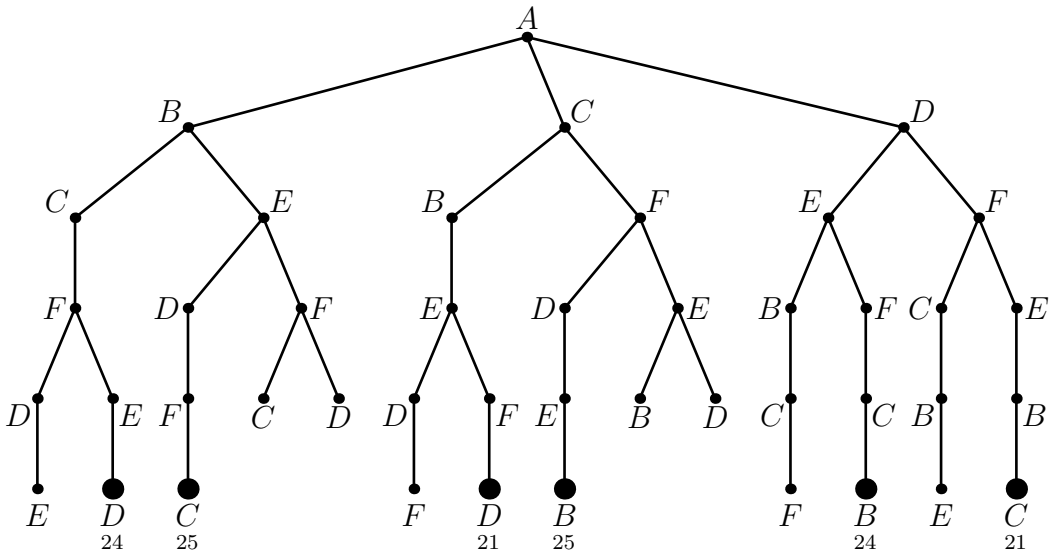
Nechť G je ohodnocený graf na obr. 12.6. Budeme procházet všechny hamiltonovské kružnice v grafu G a hledat mezi nimi optimální řešení. Začneme-li ve vrcholu A, můžeme pokračovat do libovolného z jeho 3 sousedů. Z každého souseda pak lze přejít do dvou dosud nenavštívených vrcholů atd. Nemá-li již aktuální vrchol x žádného nenavštíveného souseda, jsou dvě možnosti. Buďto je možné přejít z x do A a uzavřít tím hamiltonovskou kružnici (pak je nutné porovnat její váhu se stávajícím minimem a případně jej aktualizovat), nebo to možné není a daná větev prohledávání skončila neúspěchem. V každém případě se vrátíme k poslední učiněné volbě a zvolíme další z variant. Jsou-li všechny možnosti probrány, algoritmus končí.

Postup lze přehledně znázornit kořenovým stromem na obr. 12.7 (tzv. *rozhodovací strom*). Kořenu je přiřazen výchozí vrchol A, potomci každého vrcholu odpovídají variantám dalšího prohledávání. Zvýrazněné listy tohoto stromu představují hamiltonovské kružnice (čísla u listů jsou váhy těchto kružnic), ostatní listy představují cesty, které se na hamiltonovské kružnice nedají rozšířit. Úloha má dvě řešení o váze 21.

I na tomto malém příkladu je vidět, že se rozhodovací strom našeho algoritmu může rychle rozrůst. Časová složitost je zhruba dána funkcí $n!$ (roste tedy ještě mnohem rychleji než 2^n). Např. u úplného grafu totiž probíráme všech $(n-1)!$ hamiltonovských kružnic (viz cvičení 12.7).



Obrázek 12.6: Zadání úlohy obchodního cestujícího.



Obrázek 12.7: Rozhodovací strom pro úlohu obchodního cestujícího.

Pro úlohu obchodního cestujícího není znám žádný efektivní algoritmus. Stejně jako problém rozpoznávání hamiltonovských grafů, o kterém jsme hovořili v oddílu 6.6, patří i tato úloha k tzv. NP-úplným problémům.

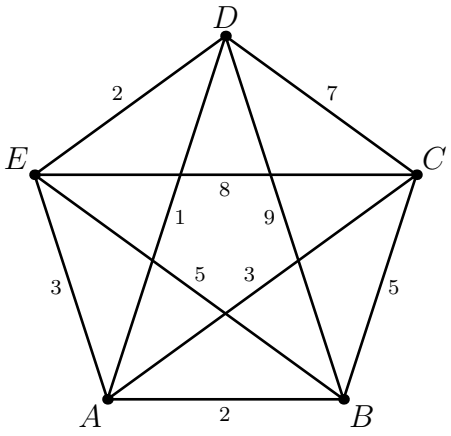
Cvičení

► **12.7** Ukažte, že úplný graf na n vrcholech má $(n-1)!$ hamiltonovských kružnic.

► **12.8** Pomocí rozhodovacího stromu najděte řešení úlohy obchodního cestujícího pro ohodnocený graf na obr. 12.8.

12.6 Toky v sítích

V této kapitole jsme uvedli jen několik aplikací ohodnocených grafů. Oblastí velmi bohatou na tyto aplikace je také teorie toků v sítích, která se zabývá následující situací. *Síť* je orientovaný graf obsahující dva význačné vrcholy z (zdroj) a s (stok). Každá hrana má určenou *propustnost* (budeme-li si hrany představovat jako potrubí, jde o množství kapaliny, které může hranou za jednotkový čas protéct). *Tok* v této síti je ohodnocení hran s vlastností, že součet ohodnocení hran vstupujících do libovolného vrcholu $x \notin \{z, s\}$ je shodný se součtem ohodnocení hran, které z x vystupují (z x tedy odtéká tolik kapaliny, kolik do něj přiteklo). Cílem je najít *maximální tok*, tj. tok, pro který ze zdroje vytéká maximální množství kapaliny. Existuje věta (tzv. věta o maximálním toku), která toto množství



Obrázek 12.8: Graf ke cvičení 12.8.

jistým elegantním způsobem charakterizuje. Pro nalezení maximálního toku jsou k dispozici efektivní algoritmy. Podrobnější informace k tomuto tématu lze získat mj. v přednáškách [9].