Grafové algoritmy

Matematické algoritmy (11MAG)

Jan Přikryl

5. přednáška 11MAG
pondělí 4. listopadu 2013

verze: 2013-11-05 16:26

Obsah

1 Úvod do teorie grafů **1**

1.1 Základní definice 2
1.2 Podgraf 3
1.3 Isomorfismus 4
1.4 Zvláštní typy grafů 4
1.5 Orientované grafy 5

2 Implementace grafů **6**

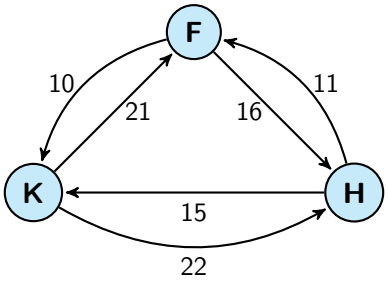
2.1 Eulerovské grafy 6

3 Základní grafové úlohy **7**

3.1 Procházení grafem 7
3.2 Hledání nejkratší cesty 8
3.2.1 Dijkstrův algoritmus 8
3.2.2 Algoritmus A* 8
3.3 Kostry grafů 10
3.3.1 Kruskalův algoritmus 10
3.3.2 Prime-Jarníkův algoritmus 11
3.3.3 Sollinův-Borůvkův algoritmus 12

1 Úvod do teorie grafů

Teorie grafů je matematická a informatická disciplína zabývající se studiem matematických struktur popisujících párové reakce mezi objekty – obecných grafů.Grafy jsou jednou z klíčových



Obrázek 1: Graf popisující možné cesty MHD mezi jednotlivými budovami ČVUT FD: **F** označuje budovu Na Florenci, **H** Horskou a **K** Konvikt. Ohodnocení hran je nejrychlejší cesta v kombinaci *pěšky* a *MHD* v minutách.

oblastí, kterou studuje diskrétní matematika. K oblibě grafů v inženýrských disciplínách přispívá široká oblast možných aplikací: Obecné grafy lze použít k modelování mnoha typů vztahů a dynamických jevů ve fyzice, biologii, sociálních vědách či informačních systémech.

Některé aplikace grafů jsou zcela zjevné [1]: daný problém „ze života“ nejprve přeformulujeme na **grafovou úlohu**, tedy tak, aby celý problém popisoval nějaký typ obecného grafu (viz například Obrázek 1) a nějakou základní operaci na tomto grafu – o tom, jaké známe základní typy obecných grafů a jaké úlohy na nich typicky řešíme, si povíme později. Příslušný graf, modelující konkrétní situaci, potom nakreslíme nebo zadáme vhodným způsobem do počítače, grafovou úlohu nějakým známým postupem (jenž v tomto kontextu budeme nazývat **grafovým algoritmem**) vyřešíme a získané řešení opět z řeči grafů přeložíme zpět do běžné mluvy. Často jsou však aplikace grafů skryté: nikde se žádný graf explicitně nepoužije, graf zůstává skrytý v pozadí, ale použitý postup řešení má svou paralelu ve světě grafů a lze jej pomocí grafů zdůvodnit. ■ **Nějaký příklad?** ■

Umění aplikovat grafy zahrnuje dvě dílčí dovednosti [1]:

1. Musíme být schopni zkonstruovat graf, jenž modeluje danou situaci (tedy takový graf, na němž jsou zachyceny vztahy, o něž se zajímáme). Někdy to je zcela triviální (například při formulaci rozhodovací úlohy, jakým způsobem se nejrychleji přemístit z mé pražské kanceláře na výuku v Děčíně), jindy to vyžaduje značné úsilí. ■ **Příklad?** ■
2. Musíme být schopni řešit alespoň základní grafové úlohy a musíme vědět, které z nich jsou snadno a rychle řešitelné a které jsou naopak komplikované a výpočetně složité. S úspěchem můžeme využít i schopnost převést nějakou neznámou grafovou úlohu na jinou, kterou již dovedeme vyřešit (toto umění převádět úlohy má mnoho společného s uměním najít vhodný grafový model).

Následující text je stručným výtahem z publikací pana Hliněného [3], pana Demela [1] a pánů Matouška a Nešetřila [4]. První text je spíše prakticky zaměřen, poslední dva texty považuji základní českou literaturu o grafech a doporučuji je ke studiu v případě, že se o grafech a grafových algoritmech potřebujete dozvědět více.

1.1 Základní definice

[

Základní poučka na úvod] Základní poučka na úvod zní: *Nepleťte si obecný graf s grafem funkce!*

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

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

Orientovaný graf má všechny hrany orientované, **neorientovaný graf** má všechny hrany neorientované. Teoreticky existují i grafy smíšené, v nichž se vyskytují oba druhy hran, těmi se ale nebudeme zabývat.

Zkusme to nyní popsat formálněji.

Definice 1 (Neorientovaný graf)**.** Neorientovaný graf je dvojice $G = (\mathcal{V}, \mathcal{E})$ tvořená neprázdnou konečnou množinou $\mathcal{V}$, jejíž prvky nazýváme vrcholy a konečnou množinou $\mathcal{E}$, jejíž prvky nazýváme orientovanými hranami. Každá hrana množiny $\mathcal{E}$ je přitom dvouprvkovou podmnožinou množiny $\mathcal{V}$, $\forall e_{ij} \in \mathcal{E} : e_{ij} = \{v_i, v_j\}, v_i, v_j \in \mathcal{V}$.

Hranu mezi vrcholy i a j bývá zvykem značit $\{i, j\}$ nebo krátce e_{ij}. Množinu vrcholů grafu G označujeme $V(G)$, analogicky $E(G)$ označuje množinu hran tohoto grafu.

O hraně e_{xy} říkáme, že vede z vrcholu x do vrcholu y a také, že spojuje vrcholy x a y. O vrcholech x a y pak říkáme, že jsou incidentní (nebo že incidují) s hranou e_{xy} a také naopak můžeme říci, že hrana e_{xy} je incidentní s vrcholy x a y. Oba vrcholy x a y také souhrnně nazýváme krajními vrcholy hrany e_{xy}. Vrchol, který není incidentní s žádnou hranou, nazýváme izolovaným vrcholem.

Počet hran, incidentních s nějakým vrcholem $v \in V(G)$ nazýváme **stupněm vrcholu**. Izolovaný vrchol má stupeň 0.

Jestliže hrana spojuje vrchol se sebou samým (tedy například hrana e_{xx}), nazýváme ji (orientovanou) smyčkou.

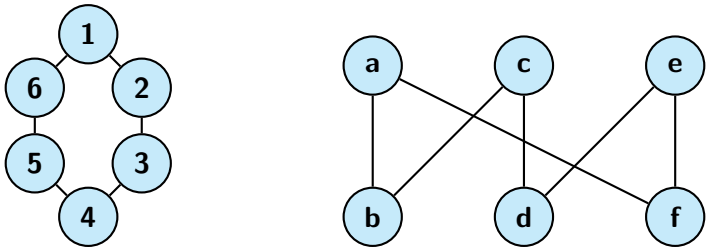
Obecně je možné, aby několik hran mělo stejné počáteční a koncové vrcholy, tedy aby pro různé hrany (pozor, musíme změnit značení hran) $e_l \neq e_2$ platilo $e_1 = \{x, y\}$ a zároveň $e_2 = \{x, y\}$. O takových hranách říkáme, že jsou rovnoběžné nebo též násobné. Množina hran grafu může být i prázdná.

1.2 Podgraf

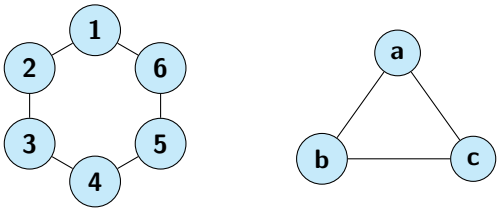
Každý graf (až na ten zcela triviální s jedním vrcholem a prázdnou množinou hran) lze rozdělit na podgrafy. **Podgraf** je, jednoduše řečeno, část grafu. Avšak tato slova musíme definovat poněkud přesněji, abychom předešli situacím, kdy by nám zbyly pouze hrany bez odpovídajících vrcholů.

Definice 2 (Podgraf)**.** Podgrafem grafu G rozumíme libovolný graf H na podmnožině vrcholů $V(H) \subseteq V(G)$, jenž má za hrany libovolnou podmnožinu hran grafu G majících oba vrcholy ve $V(H)$. Píšeme $H \subseteq G$, tj. stejně jako množinová inkluze (ale význam je trochu jiný).

Definice 3 (Indukovaný podgraf)**.** Indukovaným podgrafem grafu G rozumíme takový podgraf $H \subseteq G$, kde podmnožina hran $E(H)$ zahrnuje všechny původní hrany mezi vrcholy z $V(H)$. Někdy se mu také říká plný podgraf.



Obrázek 2: Oba grafy na obrázku popisují to samé – jsou izomorfní.



Obrázek 3: Kružnice a trojúhelník

1.3 Isomorfismus

Co když vezmeme nějaký graf (třeba ten na Obrázku 2) a nakreslíme jej jednou tak, podruhé zase jinak — jedná se o tentýž graf nebo ne? Přísně formálně řečeno, každé nakreslení jistého grafu, třeba toho na Obrázku 2, je jiným grafem. ale přitom bychom rádi řekli, že různá nakreslení téhož grafu jsou ekvivalentní – už jen proto, že grafy mají modelovat vztahy mezi dvojicemi objektů, ale tyto vztahy přece vůbec nezávisí na tom, jak si grafy nakreslíme. Pro tuto stejnost grafů se vžil pojem **isomorfní grafy**.

Pro správné pochopení a využití teorie grafů je tedy třeba nejprve dobře chápat pojem isomorfismu grafů.

Definice 4 (Izomorfismus)**.** Izomorfismus grafů G a H je bijektivní (vzájemně jednoznačné) zobrazení $f : V(G) \to V(H)$, pro které platí, že každá dvojice vrcholů $(x, y) \in E(G)$ je spojená hranou v G právě tehdy, když je dvojice $f(u), f(v) \in V(H)$ spojená hranou v H.

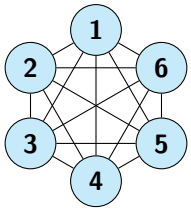
Grafy G a H jsou izomorfní, pokud mezi nimi existuje izomorfismus. Značíme $G \simeq H$. Izomorfní grafy mají stejný počet vrcholů i hran. Vrcholu stupně k lze izomorfismem přiřadit pouze vrchol stejného stupně k. Dvojici sousedních vrcholů může být izomorfismem přiřazena opět jen dvojice sousedních vrcholů.

1.4 Zvláštní typy grafů

Některé často se vyskytující typy grafů je zvykem nazývat specifickými názvy. Nejdůležitější jsou patrně ty následující:

Definice 5 (Kružnice)**.** Kružnice C_n délky n je graf, jenž má n vrcholů spojených do jednoho cyklu n hranami, $n \geq 3$. Podgrafu $H \subseteq G$, který je isomorfní nějaké kružnici, říkáme kružnice v G.

Kružnici délky 3 říkáme trojúhelník.



Obrázek 4: Úplný graf, jenž vznikl doplněním hrad do kružnice na Obrázku 3

Definice 6 (Cesta)**.** Cesta P_n délky n má $n+1$ vrcholů spojených za sebou n hranami. Podgrafu $H \subseteq G$, který je izomorfní nějaké cestě, říkáme cesta v G.

Definice 7 (Úplný graf)**.** Úplný graf K_n má $n \geq 2$ vrcholů, všechny navzájem pospojované.[1] Grafy P_1 (cesta délky 1) a C_3 (trojúhelník) jsou zároveň úplnými grafy.

Definice 8 (Klika)**.** Podgrafu $H \subseteq G$, který je isomorfní nějakému úplnému grafu, říkáme klika v G.

Někdy se za kliku považuje pouze takový úplný podgraf, který je maximální vzhledem k inkluzi.

1.5 Orientované grafy

V některých případech (například u toků v sítích) potřebujeme u každé hrany vyjádřit její směr. To vede na definici orientovaného grafu, ve kterém hrany jsou uspořádané dvojice vrcholů. V obrázcích kreslíme orientované hrany se šipkami.

Definice 9 (Orientovaný graf)**.** Orientovaný graf je uspořádaná dvojice $D = (\mathcal{V}, \mathcal{A})$, kde $\mathcal{A} \subset \mathcal{V} \times \mathcal{V}$ je množina orientovaných hran mezi vrcholy grafu.

Orientované grafy odpovídají relacím, které nemusí být symetrické: Hrana $\{x, y\}$ v orientovaném grafu D začíná ve vrcholu x a končí ve vrcholu v. Opačná hrana $\{y, x\}$ není totožná s hranou $\{x, y\}$.

Nejprve bychom si měli přesně ujasnit, jak se pohybujeme grafem, tedy co je vlastně procházkou v grafu. Tento pojem by měl postihnout základní věc, že v grafu procházíme hranami vždy z vrcholu do sousedního vrcholu, a přitom ponechat dostatek volnosti pro vracení se a zacyklení procházek.

Definice 10 (Sled)**.** Sledem délky n v grafu G rozumíme posloupnost vrcholů a hran $v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$, ve které vždy hrana e_i má koncové vrcholy v_{i-1}, v_i, tedy $e_i = \{v_{i-1}, v_i\}$.

Sled je vlastně procházka po hranách grafu z vrcholu v_x do vrcholu v_y. Příkladem sledu může být průchod IP paketu internetem (včetně cyklení).

Věta 11. *Pokud mezi dvěma vrcholy grafu G existuje sled, pak mezi nimi existuje cesta.*

Důkaz. Nechť $u = v_0, e_1, v_1, \ldots, e_n, v_n = v$ je sled délky n mezi vrcholy u a v grafu G. Začneme budovat nový sled W z vrcholu $w_0 = u$, který už bude cestou: Předpokládejme, že nový sled W už obsahuje nějakou sekvenci vrcholů a hran, $w_0, e_1, w1, \ldots, w_i$ (na začátku pro $i = 0$ jde pouze

[1] Takový graf má celkem $\binom{n}{2}$ hran.

o vrchol w_0 bez hran), kde $w_i = v_j$ pro některé $j \in \{0, 1, \ldots, n\}$. Najdeme největší index $k \geq j$ takový, že $v_k = v_j = w_i$, a sled W pokračujeme krokem ...,$w_i = v_j = v_k, e_{k+1}, w_{i+1} = v_{k+1}$. Zbývá dokázat, že nový vrchol $w_{i+1} = v_{k+1}$ se ve sledu W neopakuje. Pokud by tomu ale tak bylo $w_l = w_{i+1}$, $l \leq i$, pak bychom na se na vrchol w_{i+1} dostali už dříve z vrcholu w_l, což je v rozporu z naším předpokladem, že jde o nový vrchol. Budování sledu ukončíme v okamžiku, kdy $w_i = v$. □

Definice 12 (Souvislý graf)**.** Graf G je souvislý pokud je G tvořený nejvýše jednou komponentou souvislosti, tedy pokud každé dva vrcholy G jsou spojené cestou.

2 Implementace grafů

Mějme jednoduchý graf G na n vrcholech a značme vrcholy jednoduše čísly $V(G) = \{0, 1, \ldots, n-1\}$. Pro počítačovou implementaci grafu G pomocí *statických datových struktur* se nabízejí dva základní způsoby:

- **Maticí sousednosti $\mathbf{G}_{n\times n}$** ve které $\mathbf{G}_{ij} = 1$ znamená hranu mezi vrcholy i a j. Matici sousednosti implementujeme jako dvourozměrné pole binárních (nebo celočíselných) hodnot.
- **Výčtem sousedů $\mathbf{S}_{n\times m}$**, kde počet sloupců m je dán nejvypšším počtem sousedících vrcholů grafu C. Prvky $\mathbf{S}_{ij}$ v tomto pojetí udávají j-tý sousední vrchol vrcholu i.

Obrázek

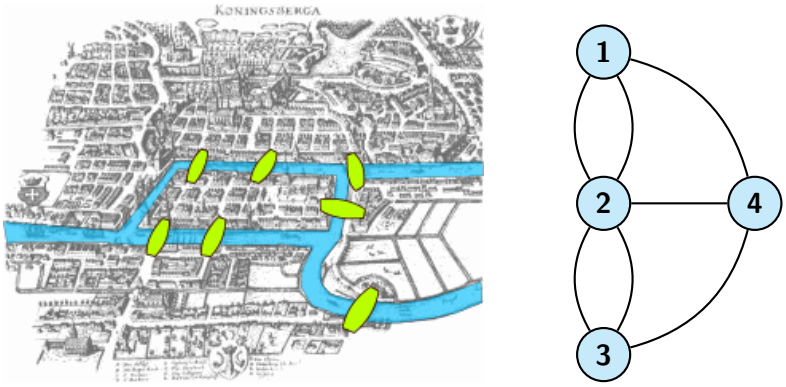
$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{S} = \begin{pmatrix} 0 & -1 & -1 \\ 0 & -1 & -1 \\ 0 & -1 & -1 \\ 0 & -1 & -1 \\ 0 & -1 & -1 \end{pmatrix}$$

2.1 Eulerovské grafy

Snad nejstarší výsledek vůbec v oblasti teorie grafů pochází od Leonharda Eulera, jenbž tak de facto položil základ celému oboru. Jedná se o slavný *problém sedmi mostů v Královci* (původně Königsbergu, dnešním ruském Kaliningradě). Toto pruské město leží na řece Pregole, která vytváří dva ostrovy. Ostrovy byly s ostatním městem spojeny sedmi mosty, vyznačenými na Obrázku 5.

O jaký problém se tehdy jednalo? Městští radní chtěli vědět, zda mohou suchou nohou přejít po každém ze sedmi mostů právě jednou. Euler dokázal, že tomu tak není, a aby důkaz mohl formulovat, celý problém převedl do abstraktní roviny – z mostů vytvořil hrany a z ostrovů a pevniny uzly tehdy neznámé matematické struktury, dnes nazývané graf. Eulerovo pozorování,



Obrázek 5: Sedm mostů v Královci. Upravená dobová rytina. Převzato z Wikipedia Commons [5] (vlevo). Odpovídající graf (vpravo)

že základem problému je počet mostů a umístění jejich koncových bodů bez toho, abychom se museli zajímat o přesnou geometrickou polohu, je předzvěstí nástupu dalšího nového matematického oboru – topologie.

Rozbor problému mostů v Královci vedl Eulera k následující definici a odpovědi.

Definice 13 (Otevřený a uzavřený tah)**.** Tah je sled v grafu bez opakování hran. Uzavřený tah je tahem, který končí ve vrcholu, ve kterém začal. Otevřený tah je tahem, který končí v jiném vrcholu, než ve kterém začal.

Nejstarší výsledek teorie grafů od Leonarda Eulera poté zní:

Věta 14 (Eulerovský tah)**.** *Graf G lze nakreslit jedním uzavřeným tahem právě tehdy, když G je souvislý a všechny vrcholy v G jsou sudého stupně.*

Důsledkem této věty je také zjištění, že graf G lze nakreslit jedním otevřeným tahem právě tehdy, když G je souvislý a všechny vrcholy v G až na dva jsou sudého stupně.

3 Základní grafové úlohy

3.1 Procházení grafem

Zcela základní grafovou úlohou je úloha, kdy chceme postupně navštívit všechny uzly grafu a nějakým blíže nespecifikovaným způsobem aktualizovat či ověřit ohodnocení hran či uzlů. Tuto úlohu nazýváme úlohou **procházení grafem**.

Při procházení grafem z vrcholu v_i si máme dvě možnosti, jak postupovat:

- **procházení do hloubky** -– před zpracováním sousedních uzlů v_j navštívíme všechny jejich nenavštívené potomky, projdeme tedy nejprve co nejhlouběji to v daném grafu jde a pak se postupně vracíme zpět,
- **procházení do šířky** – zpracujeme nejdříve všechny sousedních uzly v_j a až poté pokračujeme s jejich potomky, procházíme tedy nejprve co nejširší množinu bezprostředních sousedů.

3.2 Hledání nejkratší cesty

Problém 15 (Nejkratší cesta v grafu)**.** *Najděte takovou cestu C z vrcholu u do v v grafu $G(V, E, w)$, jejíž ohodnocení je minimální možné.*

Pro nalezení nejkratší (vážené) cesty mezi dvěma vrcholy kladně váženého grafu se používá tradiční Dijkstrův algoritmus či jeho vhodná vylepšení (A*). Takové algoritmy se například používají při vyhledávání vlakových spojení. Pravděpodobně se i vy někdy dostanete do situace, kdy budete nejkratší cestu hledat, proto si popsaný algoritmus včetně jeho vylepšení A* zapamatujte.

3.2.1 Dijkstrův algoritmus

Dijkstrův algoritmus, popsaný v Algoritmu 1, je variantou na procházení grafu do šířky (jak uvidíme, nejde ovšem o čisté procházení do šířky), kdy u každému vrcholu grafu ještě přiřadíme proměnnou, udávající vzdálenost od výchozího bodu cesty (tedy délku nejkratšího sledu, kterým jsme se do tohoto vrcholu zatím dostali). V dalším kroku algoritmu procházíme úschovnu nalezených nových vrcholů a z ní vždy vybíráme vrchol s nejmenší vzdáleností k nějakému z již nalezených vrcholů (do takového vrcholu se žádnou kratší cestou už dostat nelze, všechny ostatní cesty budou delší). Na konci zpracování tyto proměnné vzdálenosti udávají správně nejkratší vzdálenosti z počátečního vrcholu do všech ostatních vrcholů.

Poznamenejme, že pokud necháme tento algoritmus proběhnout až do zpracování všech vrcholů, získáme ve vzdal[i] nejkratší vzdálenosti z počátečního vrcholu do všech ostatních vrcholů. Všimněme si dále, že Algoritmus 1 počítá stejně dobře nejkratší cestu i v orientovaném grafu.

Celkový počet kroků Dijkstrova algoritmu nutný k nalezení nejkratší cesty z uzlu a do uzlu b je přibližně $\mathcal{O}(|V(G)|^2)$, počet kroků tedy roste kvadraticky s počtem vrcholů grafu. Při efektivnější implementaci úschovny nezpracovaných vrcholů (můžeme například použít haldu indexovanou hodnotou vzdálenosti) lze na řídkých grafech, s nimiž často pracujeme, dosáhnout i mnohem rychlejšího běhu.[2]

3.2.2 Algoritmus A*

Algoritmus A* je rozšíření původního Dijkstrova algoritmu, postavené na heuristickém výběru následujícího vrcholu. Při *vhodně* zvolené heuristice je A* algoritmus výrazně rychlejší, než Dijkstrův algoritmus.

Nechť heuristika $h(x)$ udává libovolný dolní odhad vzdálenosti z vrcholu x do cíle v. Každá hrana $e_{xy} = \{x, y\}$ grafu $G(V, E, w)$ dostane nové délkové ohodnocení $w'(e_{xy}) = w(e_{xy}) + h(y) - h(x)$. Přípustná je taková heuristika, kde všechna upravená ohodnocení jsou nezáporná, neboli $w(e_{xy}) \geq h(x) - h(y)$. Všimněte si, že vzhledem k tomu, že se hodnoty $h(x)$ a $h(y)$ v obecném případě liší,A* algoritmus každý graf implicitně převádí na graf orientovaný – $w'(e_{xy}) \neq w'(e_{yx})$.

Pro použití při navigaci v mapě[3] může heuristika $h(x)$ udávat například Euklidovskou vzdálenost bodů x a v. Taková heuristika je podle trojúhelníkové nerovnosti vždy přípustná.

Použijeme-li na graf $G(V, E, w')$ s takto upraveným ohodnocením hran původní Dijkstrův algoritms, bude tento algoritmus silně preferovat hrany vedoucí ve směru k cíli v – ohodnocení

[2] Udává se čas téměř úměrný počtu hran prohledávaného grafu.
[3] Velmi častá aplikace A* algoritmu

Algoritmus 1 Dijkstrův pro nejkratší cestu v grafu. Tento algoritmus nalezne nejkratší cestu mezi vrcholy x a y kladně váženého grafu G, daného seznamem sousedních vrcholů.

Require: $G(V, E, w)$ na n vrcholech popsaný seznamem sousedů $s_{i,j}$ a délek hran $d_{i,j}$
Require: Počáteční uzel $x \in V(G)$, koncový uzel $y \in V(G)$
Ensure: Nejkratší cesta z x do y

```
for i = 1 to n do
    c_i ← ∞ {zatím nalezená délka nejkratší cesty do tohoto uzlu}
    z_i ← false {vrchol nebyl zatím zpracován}
end for
c_x ← 0 {výchozí uzel bude označen jako zpracovaný v prvním cyklu}
while z_y ≠ true do
    j ← n
    for i = 1 to n − 1 do
        if z_i = false and c_i < c_j then
            j ← i
        end if
    end for{zde jsme našli nejbližší nezpracovaný vrchol j, ten teď zpracujeme}
    if c_j is ∞ then
        return Mezi vrcholy x a y není cesta.
    end if
    z_j ← true {označíme jako zpracovaný}
    for k = 1 to ‖s_j‖ do {projdeme všechny sousedy k uzlu j}
        if c_j + d_{j,k} < c_{s_{j,k}} then {a pokud je cesta z x přes j kratší}
            a_{s_{j,k}} ←= j {zapamatujeme si to}
            c_{s_{j,k}} ← c_j + d_{j,k} {a uložíme vzdálenost od x}
        end if
    end for{zpracovali jsme všechny sousedy uzlu j}
end while
return Cesta délky c_y, uložená v poli a.
```

takových hran bude totiž velmi nízké, zatímco ohodnocení hran vedoucích od cílového vrcholu směrem k počátečnímu vrcholu se téměř zdvojnásobí. Výsledkem bude výrazně menší počet prohledávaných vrcholů grafu (do nalezení cesty do cíle v) a také menší potřebná velikost úschovny vrcholů.

3.3 Kostry grafů

Kromě stromů samotných se zabýváme i stromy, které jsou obsaženy jako podgrafy ve větších grafech.

Definice 16 (Kostra grafu)**.** Kostrou souvislého grafu G je podgraf v G, který je sám stromem a obsahuje všechny vrcholy grafu G.

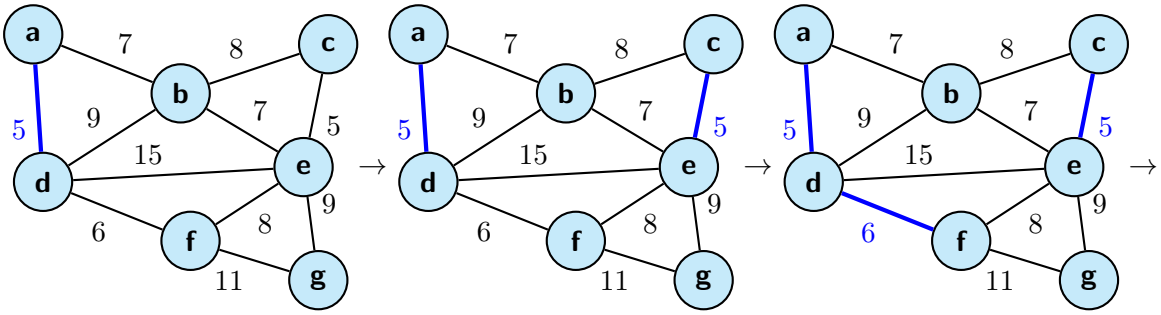
Problém 17 (Problém minimální kostry grafu)**.** *Je dán (souvislý) ohodnocený graf $G = (V, E, w)$ s nezáporným ohodnocením hran w. Otázkou je najít takovou kostru T v G, jež má nejmenší možné celkové ohodnocení.*

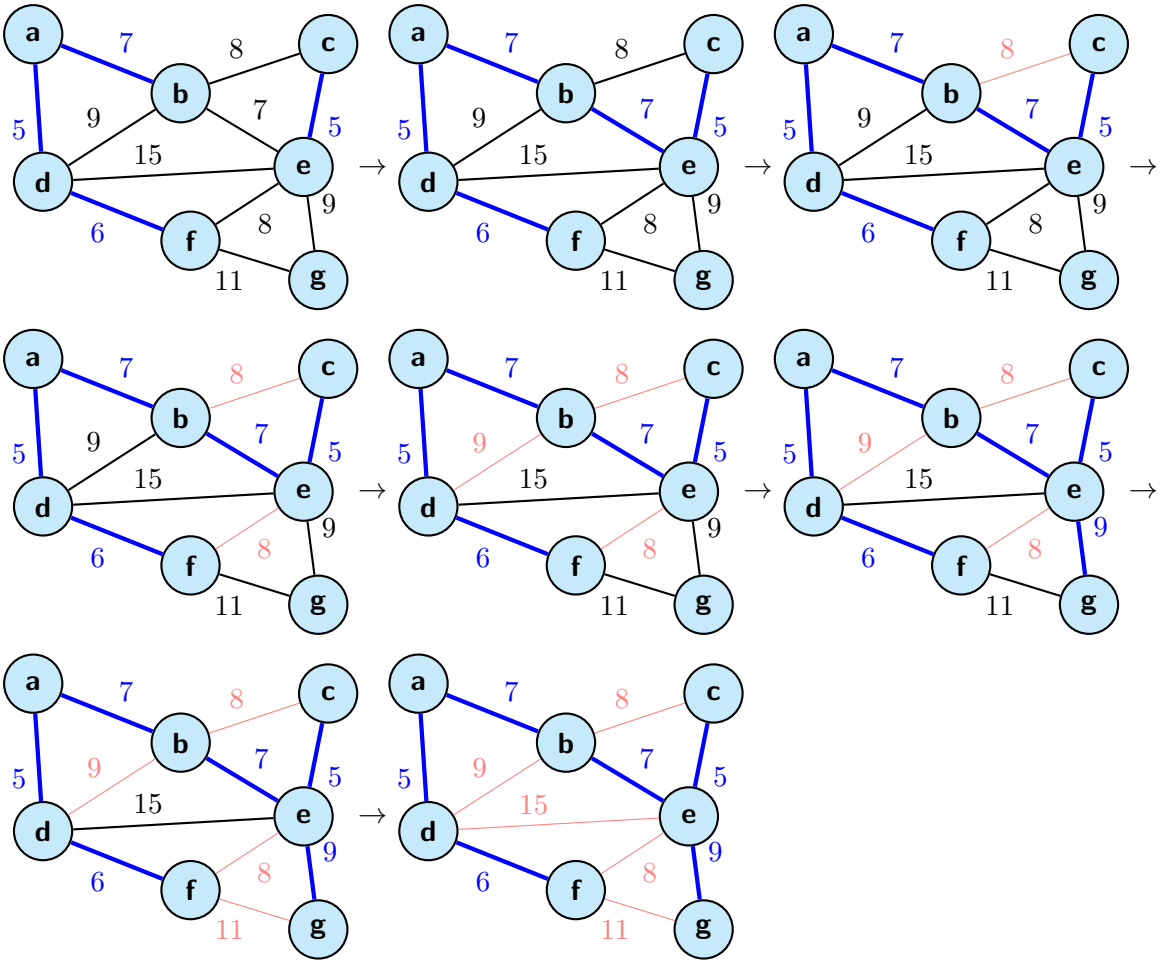
3.3.1 Kruskalův algoritmus

Algoritmus 2 Kruskalův „hladový“ algoritmus hledání minimální kostry grafu

Require: $G(V, E, w)$ s n hranami, souvislý, $\forall e_i : w(e_i) \geq 0$
Ensure: minimální kostra grafu T
  seřadíme hrany grafu G vzestupně podle jejich ohodnocení {bude $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_n)$}
  $T = \{\}$ {kostra je prázdná}
  for $i = 1$ **to** n **do**
    if $T \cup \{e_i\}$ nevytváří kružnici **then**
      $T \leftarrow T \cup \{e_i\}$
    end if
  end for
  množina T obsahuje hrany minimální kostry

Ukážeme si nyní, jak tento algoritmus funguje:

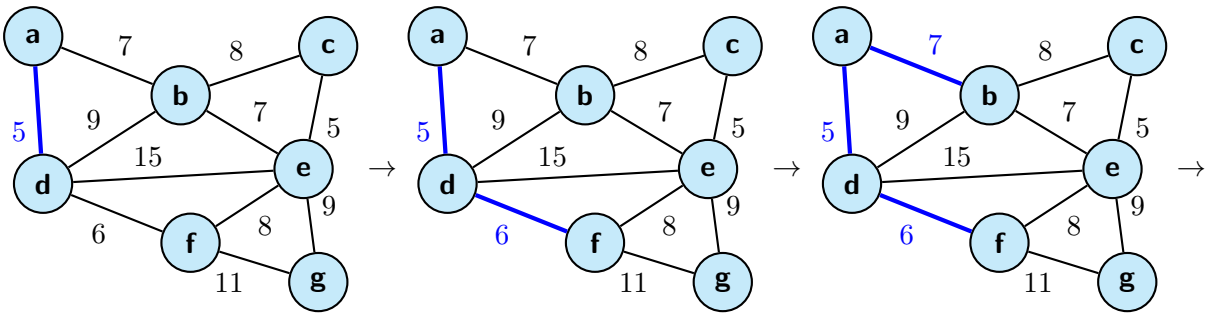




3.3.2 Prime-Jarníkův algoritmus

Algoritmus je velmi podobný Kruskalovu, hrany ale na začátku neseřazujeme, kostru začneme vytvářet z jednoho náhodně vybraného vrcholu a v každém kroku přidáme nejmenší z hran, které vedou z již vytvořeného podstromu do zbytku grafu.

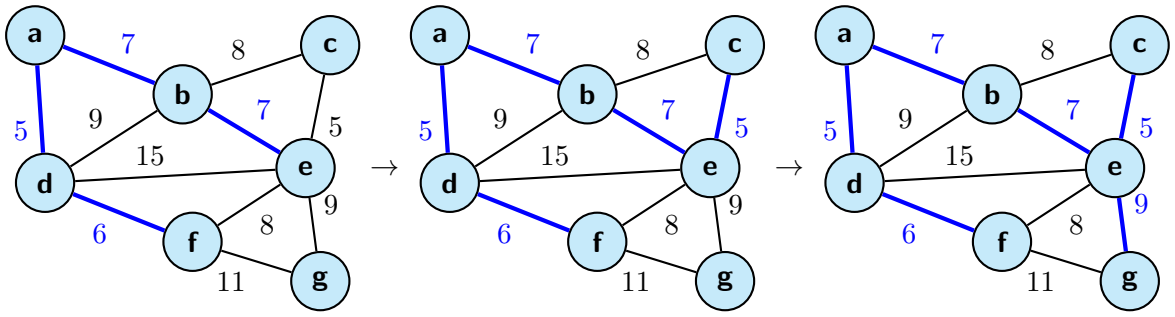
Tento algoritmus je velmi vhodný pro praktické výpočty a je dodnes široce používaný. Málokdo ve světě však donedávna věděl, že pochází od známého českého matematika Vojtěcha Jarníka – původní Jarníkova práce byla psána česky a ve světové literatuře se tak algoritmus obvykle připisuje Američanu Primovi, jenž jej nezávisle objevil až skoro 30 let po Jarníkovi.



Algoritmus 3 Prime-Jarníkův algoritmus hledání minimální kostry grafu

Require: $G(V, E, w)$ na n vrcholech popsaný seznamem sousedů $s_{i,j}$ a délek hran $d_{i,j}$
Ensure: Minimální kostra H

```
W ← {x}, kde x je libovolný prvek z V
F ← {} je prázdná množina hran kostry
while W ≠ V do
  d_min ← ∞
  for all x ∈ W do
    for all y ∈ (V \ W) do
      if d_{x,y} < d_min then
        d_min ← d_{x,y}
        z ← y
        φ ← {x, y} {zapamatujeme si nejbližší prvek ke komponentě}
      end if
    end for
  end for
  W ← W ∪ {z}
  F ← F ∪ {φ}
end while
return Minimální kostra H(W, F, ω).
```



3.3.3 Sollinův-Borůvkův algoritmus

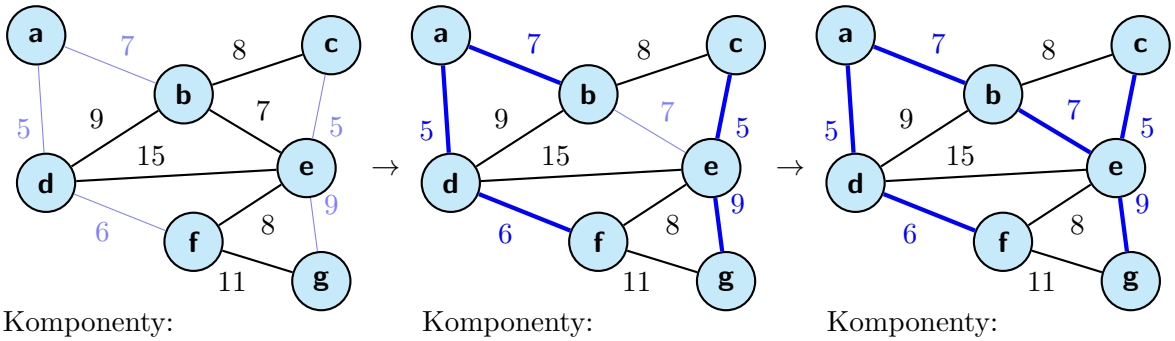
Historicky vůbec první algoritmus pro problém minimální kostry (z roku 1928) byl nalezen jiným českým (brněnským) matematikem a potkal jej snad ještě kurióznější osud, než algoritmus Jarníkův, byl totiž objeven několikrát (v Polsku, ve Francii), ale až pan Sollin jej popsal anglicky a proto většinou nese jeho jméno.

Jedná se o poněkud složitější algoritmus, chová se jako Jarníkův algoritmus spuštěný zároveň ze všech vrcholů grafu najednou. Detaily lze nalézt v literatuře [4, Oddíl 4.5.3].

Algoritmus 4 Borůvkův algoritmus hledání minimální kostry grafu

Require: $G(V, E, w)$ na n vrcholech popsaný seznamem sousedů $s_{i,j}$ a délek hran $d_{i,j}$
Ensure: Minimální kostra H

```
W ← {x}, kde x je libovolný prvek z V
F ← {} je prázdná množina hran kostry
while W ≠ V do
  d_min ← ∞
  for all x ∈ W do
    for all y ∈ (V \ W) do
      if d_{x,y} < d_min then
        d_min ← d_{x,y}
        z ← y
        φ ← {x, y} {zapamatujeme si nejbližší prvek ke komponentě}
      end if
    end for
  end for
  W ← W ∪ {z}
  F ← F ∪ {φ}
end while
return Minimální kostra H(W, F, ω).
```



Reference

[1] DEMEL, Jiří. *Grafy a jejich aplikace*. Vyd. 1. Praha: Academia, 2002, 257 s. ISBN 80-200-0990-6.

[2] JIROVSKÝ, Lukáš. *Teorie grafů* [online]. Praha, 2008 [cit. 2013-10-27]. Dostupné z: http://teorie-grafu.cz/

[3] HLINĚNÝ, Petr. *Základy teorie grafů* [online]. Vyd. 1. Brno: Masarykova univerzita, 2010 [cit. 2013-10-27]. Elportál. Dostupné z: http://is.muni.cz/elportal/?id=878389. ISSN 1802-128X.

[4] MATOUŠEK, Jiří a Jaroslav NEŠETŘIL. *Kapitoly z diskrétní matematiky*. 4., upr. a dopl. vyd. Praha: Karolinum, 2009, 442 s. ISBN 978-80-246-1740-4.

[5] Seven Bridges of Königsberg. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2013 [cit. 2013-10-27]. Dostupné z: http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg.