PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA ALGEBRY A GEOMETRIE

BAKALÁŘSKÁ PRÁCE

Základní grafové algoritmy

Autor: Martin Pospíšil
Vedoucí práce: Doc. RNDr. Petr Emanovský Ph.D.
Studijní obor: Diskrétní matematika
Forma studia: Prezenční
Rok: 2014

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. RNDr. Petr Emanovský Ph.D., a že jsem uvedl veškerou použitou literaturu.

V Olomouci dne 30.3.2014 .....

Zde bych velice rád poděkoval Doc. RNDr. Petru Emanovskému Ph.D., vedoucímu mé bakalářské práce, za konzultace a rady během vypracovávání a všem, kteří mi jakkoliv pomohli při její tvorbě. Dále bych rád poděkoval mé přítelkyni a rodině, za trpělivost a podporu během studia.

Obsah

Úvod **6**

1 Základní pojmy **7**
1.1 Definice grafu 7
1.2 Stupeň vrcholu 8
1.3 Izomorfismus grafu 9
1.4 Sled, tah a cesta 10
1.5 Cyklus (kružnice) v grafu 11

2 Reprezentace grafů pro počítač **12**
2.1 Výčet hran 12
2.2 Matice sousednosti 13
2.3 Matice incidence 14
2.4 Laplaceova matice 15

3 Prohledávání grafů **16**
3.1 Prohledávání grafu do hloubky 16
3.2 Prohledávání grafu do šířky 21

4 Hledání nejkratší cesty **27**
4.1 Základní algoritmus 29
4.2 Zlepšený algoritmus 35
4.3 Dijkstrův algoritmus 40
4.4 Bellman-Fordův algoritmus 47
4.5 Zlepšený algoritmus pro nalezení nejkratší orientované cesty . 51
4.6 Hledání nejkratší cesty v acyklických grafech 57
4.7 Upravené násobení matic 63
4.8 Floydův algoritmus 65
4.9 Hledání nejdelší cesty v grafech 69

5 Souvislost grafu a hledání komponent souvislosti **70**
5.1 Zjištění souvislosti grafu 71
5.2 Určení komponent souvislosti 71

6 Toky v sítích **77**
6.1 Ford-Fulkersonův algoritmus 79

Závěr **88**

Seznam užitých symbolů

$V(G)$	množina vrcholů grafu G
v_i	i-tý vrchol
$E(G)$	množina hran grafu G
e_i	i-tá hrana
$G = (V, E)$	graf G s množinou vrcholů V a množinou hran E
c_i	ohodnocení hrany (délka hrany) e_i
$deg(v_i)$	stupeň vrcholu v_i
deg_+	vstupní stupeň vrcholu v_i
deg_-	výstupní stupeň vrcholu v_i
D_G	matice vzdáleností grafu G
H_G	matice délek grafu G
I_G	matice incidence grafu G
L_G	Laplaceova matice grafu G
S_G	matice sousednosti grafu G
T_G	součtová matice grafu G
A^T	matice transponovaná k matici A
$a_i(j, k)$	prvek a_i matice A na pozici j, k
$d(u, v)$	vzdálenost vrcholů u a v
$PV(e_i)$	počáteční vrchol hrany e_i
$KV(e_i)$	koncový vrchol hrany e_i
P_{v_i}	hodnota vrcholu v_i v poli P
z	zdroj
s	stok
$\|f\|$	velikost toku
$E^-(v)$	odtok z vrcholu v
$E^+(v)$	přítok do vrcholu v
$\|E'\|$	velikost řezu
R^+	množina kladných reálných čísel
F	tok (v síti)
C	kapacita
$w(e_i)$	kapacita hrany e_i
$f(e_i)$	tok hrany e_i
$r(e_i)$	rezerva hrany e_i
$(a_1, \ldots, a_n)$	uspořádaná n-tice prvků (pole)
$\{a_1, \ldots, a_n\}$	množina n prvků

Úvod

S teorií grafů a nejrůznějšími grafovými algoritmy se setkáváme nejenom při studiu některých oborů na vysoké škole, ale také v našem každodenním životě. Už jako malé děti se nejspíše každý z nás snažil dostat „toho panáčka“ na konec bludiště v našem oblíbeném dětském časopise. Později už jsme nejspíše všichni řešili, jak se co nejrychleji nebo co nejkratší cestou dostaneme do práce, školy, ke kamarádovi, nebo vůbec odněkud někam a hodně z nás tento problém řeší každý den. K vyřešení těchto a mnoha dalších problémů můžeme využít již zmiňované grafové algoritmy, které nám námi hledané řešení mohou pomoci rychle nalézt.

Cílem této práce je představení několika vybraných základních grafových algoritmů, popis jejich struktury a principu na kterém fungují a jejich názorné užití na příkladech. Na začátku si definujeme několik základních pojmů a ukážeme si, jak můžeme grafy reprezentovat, pokud je potřebujeme zpracovávat v počítači. Ve zbytku práce se budeme zabývat hlavním tématem, tedy jednotlivými algoritmy. Popíšeme si principy jak dané algoritmy funguji a poté si ukážeme jejich užití na příkladech. Každý příklad je rozepsán po jednotlivých krocích u kterých je i názorná ilustrace, pro lepší představu o fungování daného algoritmu.

Pro základní jednoduché kreslení grafů v počítači doporučuji čtenáři použití programu Geogebra [11], pro pokročilejší vykreslování a práci s grafy pak program wxMaxima [12], případně Wolfram Mathamatica [13], ve kterých lze na vytvořené grafy aplikovat zde uvedené a jiné algoritmy. Odkazy na uvedené programy je možné nalézt v referencích.

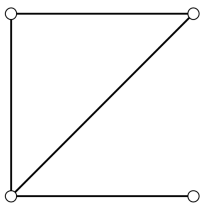
1 Základní pojmy

1.1 Definice grafu

Řekneme-li graf, nemusíme vždy chápat význam tohoto slova správně, jelikož tento pojem se vyskytuje v různých významech. Zde budeme pod pojmem graf chápat grafický způsob reprezentace vztahů mezi určitými objekty. Grafy obvykle znázorňujeme kreslením do roviny, kde vrcholy (objekty) jsou reprezentovány body (uzly) a vztahy hranami, které jsou spojením příslušných bodů. Při kreslení grafů se snažíme, aby se jednotlivé hrany s ostatními křížily co nejméně.

Definice 1.1. *Obecný graf* budeme chápat jako dvojici $G = (V, E)$, kde V je množina vrcholů $\{v_1, \ldots, v_n\}$ a E je množina hran $\{e_1, \ldots, e_m\}$ mezi nimi, kdy každá hrana e_k vždy spojuje nějaké dva vrcholy v_i, v_j. Vrcholy v_i, v_j, které daná hrana spojuje, nazýváme *incidentní* s hranou e_k.

Příklad 1.1. Příklad obecného grafu:

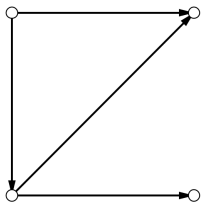


Budeme-li se později odkazovat například na vrcholy nebo hrany grafu $G = (V, E)$, pak budeme množinu vrcholů grafu G formálně zapisovat $V(G)$ a množinu hran $E(G)$.

Grafy obvykle rozlišujeme podle toho, jaký mají typ hran.

Definice 1.2. Graf $G = (V, E)$ nazveme *orientovaný graf*, jestliže každá jeho hrana ve tvaru $e_k = (v_i, v_j)$ má v_i jako svůj počáteční vrchol, píšeme $v_i = PV(e_k)$, a v_j jako svůj koncový vrchol, píšeme $v_j = KV(e_k)$. Hranu pak kreslíme jako šipku z vrcholu v_i do vrcholu v_j.

Příklad 1.2. Příklad orientovaného grafu:



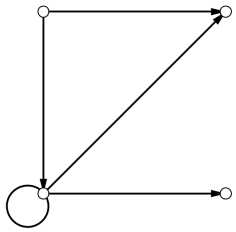
Definice 1.3. Graf $G = (V, E)$ nazveme *neorientovaný graf*, jestliže u každé hrany ve tvaru $e_k = (v_i, v_j)$ nezáleží na tom, který vrchol je počáteční a který koncový.

Definice 1.4. Vrchol v_i nazveme *sousedním* vrcholem vrcholu v_j, pokud v neorientovaném grafu mezi nimi existuje hrana, nebo mezi nimi v orientovaném grafu existuje orientovaná hrana, kdy vrchol v_i je její počáteční vrchol a vrchol v_j koncový.

Definice 1.5. *Smyčkou* grafu $G = (V, E)$ rozumíme takovou hranu $e_k = (v_i, v_j)$, kde $i = j$. Nebo-li je to taková hrana, která začíná a končí v jednom vrcholu.

Poznámka 1.1. U smyček nerozlišujeme orientaci, jelikož je jedno, jestli smyčka směřuje z vrcholu v_i do vrcholu v_i jedním nebo druhým směrem.

Příklad 1.3. Příklad grafu se smyčkou:



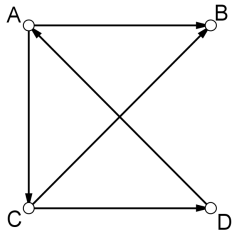
Definice 1.6. Graf $G_1 = (V_1, E_1)$ nazveme *podgrafem* grafu $G_2 = (V_2, E_2)$, jestliže $V(G_1)$ je podmnožinou $V(G_2)$ a $E(G_1)$ je podmnožinou $E(G_2)$.

1.2 Stupeň vrcholu

Definice 1.7. *Stupněm vrcholu* nazveme číslo $deg(v_i)$, které označuje počet hran, které jsou s daným vrcholem incidentní. Stupeň vrcholu také někdy nazýváme *index*.

U orientovaných grafů můžeme také rozlišit vstupní a výstupní stupeň každého vrcholu.

Definice 1.8. *Vstupním,* resp. *výstupním stupněm* vrcholu v_i orientovaného grafu G nazveme číslo $deg_{G^+}(v_i)$, resp. $deg_{G^-}(v_i)$, které označuje počet hran, které do vrcholu v_i směřují, resp. z vrcholu v_i vycházejí.



Obr 1.1

Příklad 1.4. Zjistěte z grafu na obrázku 1.1 vstupní a výstupní stupně jednotlivých vrcholů.

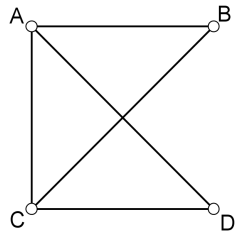
Řešení:

$$deg_-(A) = 2; deg_+(A) = 1; deg_-(B) = 0; deg_+(B) = 2;$$
$$deg_-(C) = 2; deg_+(C) = 1; deg_-(D) = 1; deg_+(D) = 1;$$

Definice 1.9. *Symetrizací* grafu $G = (V, E)$ nazveme operaci, kterou nahradíme orientované hrany za neorientované.

Symetrizace se dá také chápat jako zanedbání orientace hran.

Příklad 1.5. Příklad symetrizace grafu z obrázku 1.1 (Všechny orientované hrany nahradíme neorientovanými):



1.3 Izomorfismus grafu

Definice 1.10. Grafy $G_1 = (V_1, E_1)$ a $G_2 = (V_2, E_2)$ nazveme *izomorfní*, jestliže existuje vzájemně jednoznačné zobrazení $f : V_1 \to V_2$ takové, že platí

$$e_{k_1} = (v_i, v_j) \in E_1 \Leftrightarrow e_{k_2} = (f(v_i), f(v_j)) \in E_2;$$

neboli, pro každou hranu $e_{k_1} = (v_i, v_j)$ v grafu G_1 existuje hrana $e_{k_2} = (f(v_i), f(v_j))$ v grafu G_2 a naopak.

1.4 Sled, tah a cesta

Definice 1.11. *Sledem* nazveme libovolnou posloupnost vrcholů grafu $G = (V, E)$ takovou, že mezi každými dvěma po sobě jdoucími vrcholy je hrana.

Definice 1.12. Sled nazveme *orientovaný*, jestliže v libovolné posloupnosti vrcholů grafu $G = (V, E)$ mezi každými dvěma po sobě jdoucími vrcholy v_i a v_j, kdy $i < j$, je orientovaná hrana s počátečním vrcholem v_i a koncovým v_j.

Definice 1.13. *Tahem* nebo také *neorientovaným tahem* nazveme sled, ve kterém se neopakují žádné hrany. Tah, který začíná a končí ve stejném vrcholu nazýváme *uzavřený*.

Definice 1.14. Tah nazveme *orientovaný*, pokud mezi každými dvěma po sobě jdoucími vrcholy v_i a v_j $(i < j)$, je orientovaná hrana s počátečním vrcholem v_i a koncovým v_j.

Definice 1.15. *Cestou* nazveme soubor tahů, ve kterém se každý vrchol vyskytuje právě jednou.

Definice 1.16. Cestu nazveme *orientovanou*, resp. *neorientovanou*, pokud je souborem orientovaných, resp. neorientovaných tahů, ve kterém se neopakují žádné vrcholy.

Definice 1.17. Neorientovaný graf $G = (V, E)$ nazveme *souvislý*, jestliže mezi jeho každými dvěma vrcholy $v_i, v_j \in V$ existuje cesta z vrcholu v_i do vrcholu v_j. V opačném případě nazveme graf *nesouvislý*.

Definice 1.18. Orientovaný graf $G = (V, E)$ nazveme:

1. *Silně souvislý*, jestliže mezi každými jeho dvěma vrcholy $v_i, v_j \in V$ existuje orientovaná cesta z vrcholu v_i do vrcholu v_j i z vrcholu v_j do vrcholu v_i.

2. *Souvislý* nebo také *slabě souvislý*, jestliže jeho symetrizace je souvislý graf.

Jinak nazveme orientovaný graf *nesouvislý*.

Věta 1.1. *Každý souvislý graf s n vrcholy obsahuje nejdelší cestu maximálně o* $n - 1$ *hranách.*

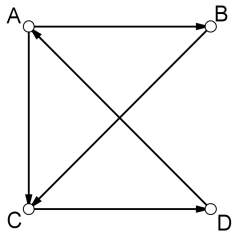
Důkaz této věty je možný nalézt například ve skriptech [7] na straně 14.

1.5 Cyklus (kružnice) v grafu

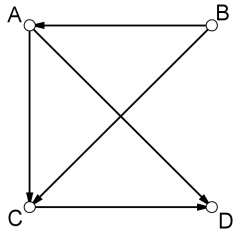
Definice 1.19. Cestu, která začíná ve vrcholu v_i a končí ve vrcholu v_i, tedy začíná a končí v jediném vrcholu, nazýváme *uzavřenou cestou*, neboli *cyklem*. Někdy se také používá označení *kružnice*.

Definice 1.20. Graf obsahující cyklus jako svůj podgraf nazýváme graf *cyklický*. V opačném případě *acyklický*.

Příklad 1.6. Příklad cyklického grafu, který je tvořen jedním cyklem:



Příklad acyklického grafu:



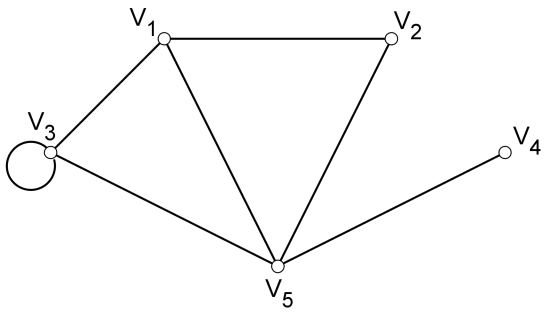
2 Reprezentace grafů pro počítač

Jak jsme si již řekli, pro člověka je nejsrozumitelnější reprezentací grafu jeho obrázek. Tato reprezentace není výhodná v okamžiku, kdy je graf příliš rozsáhlý, případně když jej chceme zpracovávat v počítači. Proto si zde uvedeme několik dalších možností reprezentace grafů.

Možnou reprezentací grafů je použití seznamu vrcholů a hran s případně dalšími atributy jako stupni vrcholů nebo kapacitou hran. Zde budeme uvažovat jen orientované, případně neorientované grafy s kladným ohodnocením hran, pokud není uvedeno jinak.

2.1 Výčet hran

Poznámka 2.1. U neorientovaného grafu jsou jednotlivé hrany určeny svými krajními vrcholy. U orientovaného grafu je každá hrana určena svým počátečním vrcholem (ze kterého hrana vychází) a koncovým vrcholem (kam hrana „směřuje“).



Obr. 2.1

Příklad 2.1. Příklad výčtu vrcholů a hran pro neorientovaný graf uvedený na obrázku 2.1 výše:

$$(v_1, v_2); (v_1, v_3); (v_1, v_5); (v_2, v_5); (v_3, v_3); (v_3, v_5); (v_4, v_5).$$

Tento zápis je vhodný například pro malé grafy (jako zde uvedený), nikoli však pro rozsáhlejší, jejichž zapsání by nám trvalo velice dlouho. Proto se pro zápis větších grafů používají mnohem efektivnější způsoby.

2.2 Matice sousednosti

Matematicky elegantní a velice užívanou reprezentací je *matice sousednosti* (adjency matrix), která nám určuje daný graf až na izomorfismus.

Definice 2.1. Nechť $G = (V, E)$ je graf. Uspořádáme-li libovolně, ale pevně, jeho množinu vrcholů $v_1, \ldots, v_n$, pak *matici sousednosti* S_G pro neorientovaný graf definujeme takto:

$$S_G = ||a_{ij}||, \text{ kde } a_{ij} = \begin{cases} 1 & \text{pokud } v_i \text{ je sousední vrchol } v_j; \\ 0 & \text{v ostatních případech.} \end{cases}$$

a v případě orientovaného grafu takto:

$$a_{ij} = \begin{cases} 1 & \text{pokud } v_i \text{ je sousední vrchol } v_j; \\ 0 & \text{v ostatních případech.} \end{cases}$$

Prvek a_{ii} je roven 1, pokud v daném vrcholu je smyčka.

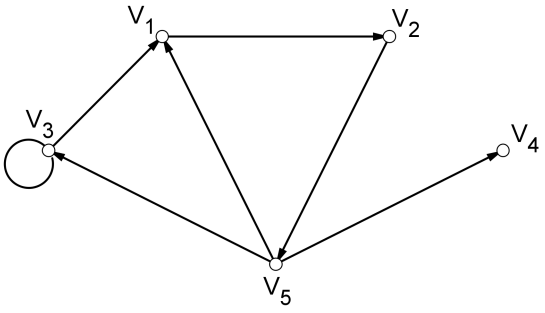
Poznámka 2.2. Matice sousednosti neorientovaného grafu je vždy symetrická, tj. čtvercová matice, která se po provedení operace transponování nezmění. Neboli:

$$S_G^T = S_G.$$

Příklad 2.2. Matice sousednosti pro neorientovaný graf z obrázku 2.1:

$$S_G = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Poznámka 2.3. Mají-li dva orientované/neorientované grafy stejnou matici sousednosti, pak jsou tyto grafy navzájem izomorfní. Naopak toto tvrzení neplatí, jelikož dva izomorfní grafy mohou mít různě uspořádané vrcholy.



Obr. 2.2

Příklad 2.3. Vytvořte matici sousednosti pro graf na obrázku 2.2.

2.3 Matice incidence

Dalším typem matice, kterým můžeme graf reprezentovat, je *matice incidence* (incidence matrix).

Definice 2.2. Nechť $G = (V, E)$ je graf. Uspořádáme-li libovolně, ale pevně, jeho množinu vrcholů $v_1, \ldots, v_n$ a hran $e_1, \ldots, e_m$, pak neorientovanému grafu můžeme přiřadit *matici incidence* I_G takto:

$$I_G = ||b_{ij}||, \text{ kde } b_{ij} = \begin{cases} 1 & \text{pokud } v_i \text{ je incidentní s } e_j; \\ 0 & \text{v ostatních případech.} \end{cases}$$

a v případě orientovaného grafu takto:

$$b_{ij} = \begin{cases} 1 & \text{pokud } v_i \text{ je počáteční vrchol hrany } e_j; \\ -1 & \text{pokud } v_i \text{ je koncový vrchol hrany } e_j; \\ 0 & \text{v ostatních případech.} \end{cases}$$

Poznámka 2.4. Orientované grafy, které obsahují alespoň jednu smyčku, nelze maticí incidence korektně reprezentovat, jelikož smyčka začíná a končí v jediném vrcholu v_i. Proto nevíme, zda $b_{ij} = 1$ nebo $b_{ij} = -1$. Neorientované grafy se smyčkou jsme schopni pomocí této matice reprezentovat, jelikož rozlišujeme jen jestli vrchol v_i je incidentní s hranou e_j nebo není.

Příklad 2.4. Vytvořte matici incidence pro dříve uvedený graf na obrázku 2.1.

Řešení:

Nejprve si označíme všechny hrany. K tomu můžeme využít například výsledků z příkladu 2.1.

$$e_1 = (v_1, v_2); e_2 = (v_1, v_3); e_3 = (v_1, v_5); e_4 = (v_2, v_5); e_5 = (v_3, v_3);$$
$$e_6 = (v_3, v_5); e_7 = (v_4, v_5).$$

Nyní již můžeme vytvořit matici incidence:

$$I_G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

2.4 Laplaceova matice

Další zajímavou maticí je *Laplaceova matice* (Laplace matrix). Tuto matici lze však použít jen pro neorientované grafy bez smyček.

Definice 2.3. Nechť $G = (V, E)$ je graf. Uspořádáme-li libovolně, ale pevně, množinu vrcholů $v_1, \ldots, v_n$, pak *Laplaceovu matici* L_G definujeme takto:

$$L_G = ||c_{ij}||, \text{ kde } c_{ij} = \begin{cases} deg(v_i) & \text{pokud } v_i = v_j; \\ -1 & \text{pokud } v_i \text{ je soused } v_j; \\ 0 & \text{v ostatních případech.} \end{cases}$$

Příklad 2.5. Laplaceova matice na začátku uvedeného grafu (Obr. 2.1), kterému odebereme smyčky:

$$L_G = \begin{pmatrix} 3 & -1 & -1 & 0 & -1 \\ -1 & 2 & 0 & 0 & -1 \\ -1 & 0 & 2 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{pmatrix}$$

Poznámka 2.5. Laplaceova matice je vždy symetrická.

3 Prohledávání grafů

Díváme-li se na libovolný graf, vnímáme jej jako obrázek. Pokud ale chceme s grafem pracovat v počítači, tento nadhled nemáme k dispozici, proto musíme graf postupně prohledávat. K tomuto účelu používáme následující algoritmy:

1. Prohledávání grafu do hloubky;
2. Prohledávání grafu do šířky.

Základní verze těchto algoritmů nám slouží k tomu, abychom například zjistili, zda jsou všechny vrcholy v našem grafu z námi vybraného vrcholu dosažitelné.

Definice 3.1. Vrchol v_j neorientovaného grafu $G = (V, E)$ nazveme *dosažitelným*, pokud v grafu G existuje cesta z libovolného vrcholu $v_i \neq v_j$ do vrcholu v_j. V opačném případě vrchol nazveme *nedosažitelným*.

Definice 3.2. Vrchol v_j orientovaného grafu $G = (V, E)$ nazveme *dosažitelným*, pokud v grafu G existuje orientovaná cesta z libovolného vrcholu $v_i \neq v_j$ do vrcholu v_j. V opačném případě vrchol nazveme *nedosažitelným*.

3.1 Prohledávání grafu do hloubky

Tento první algoritmus si lze představit jako průchod bludištěm, kdy procházíme z místnosti do místnosti po jednotlivých chodbách. V dosažené místnosti si vybíráme libovolnou chodbu, po které jsme ještě nešli. Pokud taková chodba neexistuje, musíme se vrátit chodbou kterou jsme do místnosti přišli.

Pomocná struktura:

1. V ... množina iniciovaných vrcholů grafu;
2. N ... množina nalezených vrcholů grafu;
3. Z ... množina zpracovaných vrcholů grafu;
4. Stavy vrcholů:
   (a) Iniciovaný - označení, které každý vrchol dostane na začátku;
   (b) Aktuální - vrchol, který se v danou chvíli zpracovává;
   (c) Nalezený - označení, které dostane vrchol poté, co jsme jej nalezli;
   (d) Zpracovaný - označení, které dostane vrchol poté, co jsme jej zpracovali.

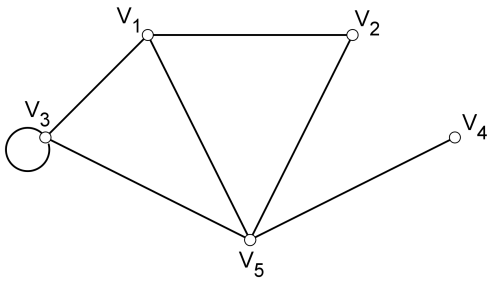
Algoritmus:

1. Provedeme základní nastavení:
   (a) Všechny vrcholy grafu označíme jako iniciované, tím dostaneme množinu V.
   (b) Vyprázdníme množinu nalezených vrcholů N a množinu zpracovaných vrcholů Z.
   (c) Vybereme si libovolný počáteční vrchol z množiny V, který nastavíme jako aktuální.
2. Hledáme sousední vrcholy aktuálního vrcholu. Mohou nastat tyto možnosti:
   (a) Nemá žádné sousední iniciované vrcholy:
      i. Nastavíme aktuální vrchol jako zpracovaný a přiřadíme do množiny Z.
      ii. Pokračujeme krokem 3.
   (b) Má jeden sousední iniciovaný vrchol:
      i. Aktuální vrchol nastavíme jako zpracovaný a přiřadíme do množiny Z.
      ii. Nalezený iniciační vrchol nastavíme jako aktuální a pokračujeme krokem 2 od začátku.
   (c) Má více sousedních iniciovaných vrcholů:
      i. Aktuální vrchol nastavíme jako zpracovaný a přiřadíme do množiny Z.
      ii. Jeden z nalezených iniciovaných vrcholů si vybereme a nastavíme jako aktuální.
      iii. Ostatní nalezené iniciované vrcholy nastavíme jako nalezené a přiřadíme do množiny N.
      iv. Pokračujeme krokem 2 od začátku.
3. Vyhodnocení a ukončení algoritmu:
   (a) Množina N je prázdná.
      i. Konec algoritmu.

(b) Množina N není prázdná.

i. Vezmeme naposledy přidaný nalezený vrchol z množiny N, nastavíme ho jako aktuální a odebereme jej z množiny N.

ii. Pokračujeme krokem 2.



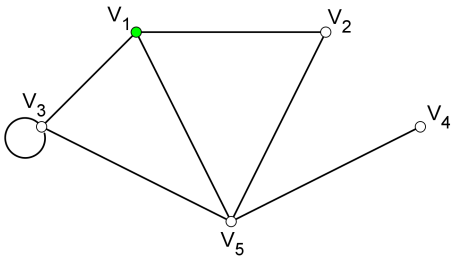
Obr. 3.1

Příklad 3.1. Zjistěte pomocí algoritmu průchodu grafu do hloubky, zda má graf na obrázku 3.1 všechny vrcholy dosažitelné.

Řešení:

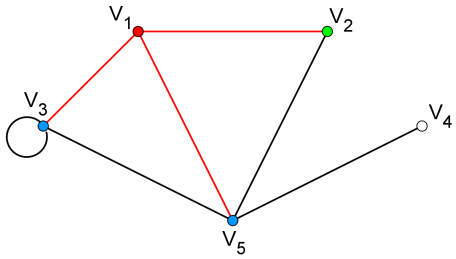
1. Všechny vrcholy grafu označíme jako iniciované, vyprázdníme obě pomocné množiny a nastavíme aktuální vrchol:

$$V = \{v_1, \ldots, v_5\};$$
$$N = \emptyset;$$
$$Z = \emptyset;$$
v_1 - aktuální.



2. Hledáme sousední vrcholy aktuálního vrcholu v_1. Nalezli jsme iniciované vrcholy: v_2, v_3, v_5. Nastavíme:

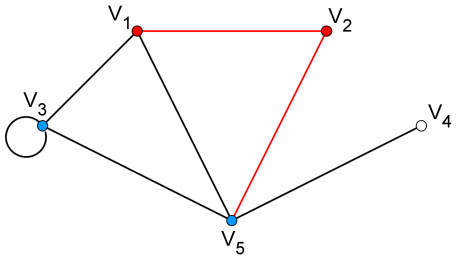
v_1 nastavíme na zpracovaný a přiřadíme do Z;
v_2 nastavíme jako aktuální;
v_3, v_5 přiřadíme jako nalezené do N.



$$Z = \{v_1\}; N = \{v_3, v_5\}.$$

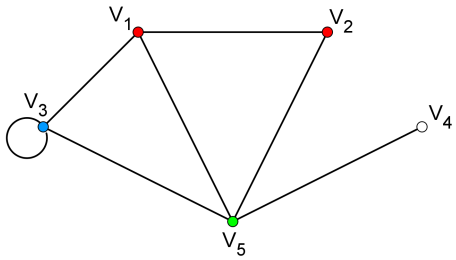
3. Hledáme sousední vrcholy aktuálního vrcholu v_2:

v_1 - zpracovaný;
v_5 - nalezený;
$v_2 \rightarrow$ zpracovaný $\rightarrow Z$.



$$Z = \{v_1, v_2\}; N = \{v_3, v_5\}.$$

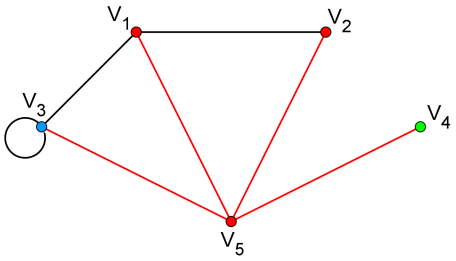
4. $N \neq \emptyset \Rightarrow v_5 \rightarrow$ aktuální.



Pokračujeme stejným způsobem dále.

5. Hledáme sousední vrcholy aktuálního vrcholu v_5:

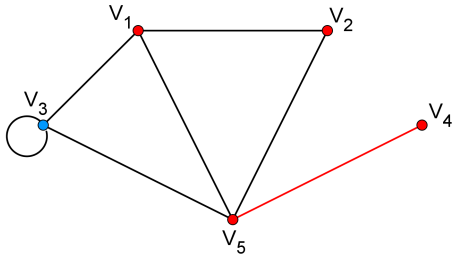
v_1, v_2 - zpracovaný;
v_3 - nalezený;
v_4 - iniciovaný;
$v_5 \rightarrow$ zpracovaný $\rightarrow Z$;
$v_4 \rightarrow$ aktuální.



$Z = \{v_1, v_2, v_5\}; N = \{v_3\}.$

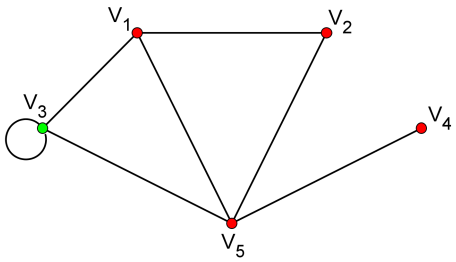
6. Hledáme sousední vrcholy aktuálního vrcholu v_4:

v_5 - zpracovaný;
$v_4 \rightarrow$ zpracovaný $\rightarrow Z$;



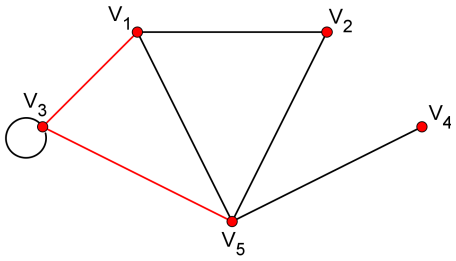
$Z = \{v_1, v_2, v_4, v_5\}; N = \{v_3\};$

7. $N \neq \emptyset \Rightarrow v_3 \rightarrow$ aktuální.



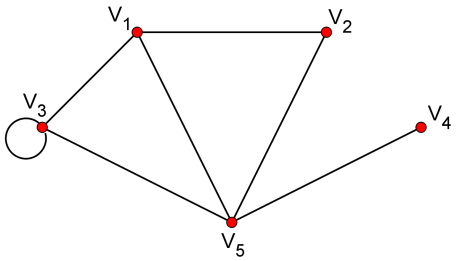
8. Hledáme sousední vrcholy aktuálního vrcholu v_3:

v_1, v_5 - zpracovaný;
$v_3 \rightarrow$ zpracovaný $\rightarrow Z$.



$Z = \{v_1, v_2, v_3, v_4, v_5\}; N = \emptyset$.

9. $N = \emptyset \Rightarrow$ konec algoritmu. Všechny vrcholy jsou dosažitelné.



Po aplikování algoritmu pro prohledávání grafu do hloubky jsme zjistili, že graf má všechny vrcholy dosažitelné, jelikož množina nedosažitelných vrcholů je prázdná.

3.2 Prohledávání grafu do šířky

Rozdíl v prohledávání grafu do šířky, oproti prohledávání grafu do hloubky, spočívá v tom, že se z místnosti, ve které jsme, rozhlédneme do všech ostatních chodeb kam můžeme jít.

Pomocná struktura:

Pomocná struktura je stejná jako u předchozího algoritmu.

Algoritmus:

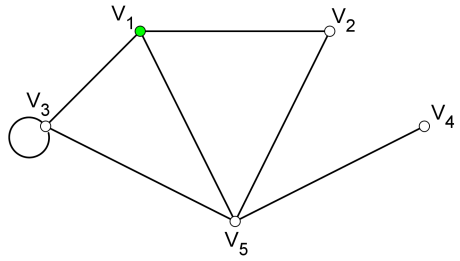
1. Provedeme základní nastavení:
   (a) Všechny vrcholy grafu označíme jako iniciované, tím dostaneme množinu V.
   (b) Vyprázdníme množinu nalezených vrcholů N a množinu zpracovaných vrcholů Z.
   (c) Vybereme si libovolný počáteční vrchol z množiny V, který nastavíme jako aktuální.
2. Hledáme sousední vrcholy aktuálního vrcholu. Mohou nastat tyto možnosti:
   (a) Nemá žádné sousední iniciované vrcholy.
      i. Aktuální vrchol nastavíme jako zpracovaný a přiřadíme do množiny Z.
      ii. Pokračujeme krokem 3.
   (b) Má jeden nebo více iniciovaných vrcholů.
      i. Aktuální vrchol nastavíme jako zpracovaný a přiřadíme do množiny Z.
      ii. Nalezené iniciované vrcholy nastavíme jako nalezené a přidáme do množiny N.
      iii. Pokračujeme krokem 3.
3. Vyhodnocení a ukončení algoritmu:
   (a) Množina N je prázdná.
      i. Konec algoritmu.
   (b) Množina N není prázdná.
      i. Vezmeme nejdříve přidaný prvek do množiny N, nastavíme jej jako aktuální a odebereme z množiny N.
      ii. Pokračujeme krokem 2.

Příklad 3.2. Zjistěte pomocí algoritmu průchodu grafem do šířky, zda má graf na obrázku 3.1 všechny vrcholy dosažitelné.

Řešení:

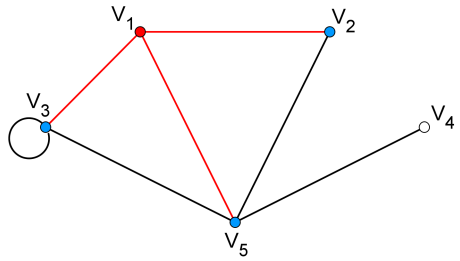
1. Všechny vrcholy grafu označíme jako iniciované, vyprázdníme obě pomocné množiny a nastavíme aktuální vrchol:

$$V = \{v_1, \ldots, v_5\};$$
$$N = \emptyset;$$
$$Z = \emptyset;$$
$$v_1 \text{ - aktuální.}$$



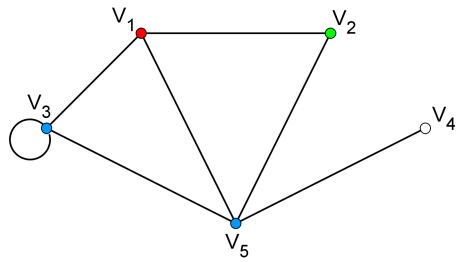
2. Hledáme sousední vrcholy aktuálního vrcholu v_1. Nalezli jsme iniciované vrcholy: v_2, v_3, v_5. Nastavíme:

v_1 nastavíme na zpracovaný a přiřadíme do Z;
v_2, v_3, v_5 nastavíme jako nalezené a přiřadíme do N.



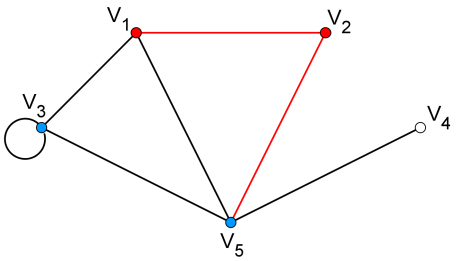
$$Z = \{v_1\}; N = \{v_2, v_3, v_5\}.$$

3. Množina $N \neq \emptyset$, proto z ní vybereme nejdříve přidaný vrchol, tedy v_2 a nastavíme jako aktuální.



4. Hledáme sousední vrcholy aktuálního vrcholu v_2. Nalezli jsme vrcholy v_1 a v_5:

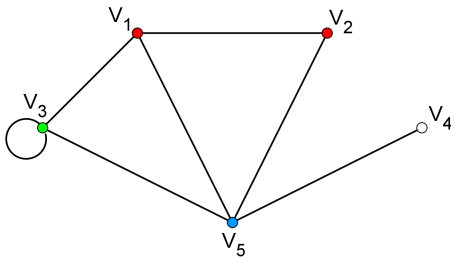
v_1 je zpracovaný, tedy nic s ním neprovádíme.
v_5 je nalezený, tedy nic s ním neprovádíme.
v_2 nastavíme na zpracovaný a přiřadíme do množiny Z.



$$Z = \{v_1, v_2\}; N = \{v_3, v_5\}.$$

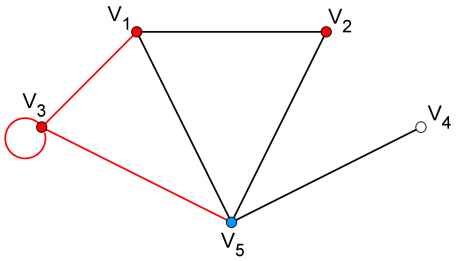
Stejným způsobem pokračujeme dále.

5. $N \neq \emptyset \Rightarrow v_3 \rightarrow$ aktuální.



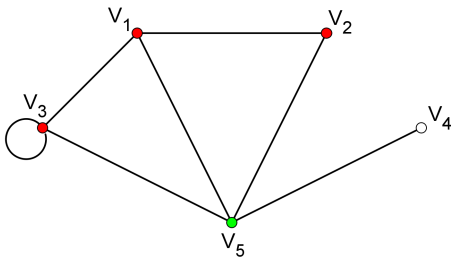
6. Hledáme sousední vrcholy aktuálního vrcholu v_3:

v_1 - zpracované;
v_5 - nalezené;
$v_3 \rightarrow$ zpracované $\rightarrow Z$.



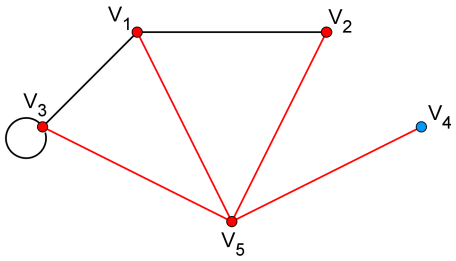
$$Z = \{v_1, v_2, v_3\}; N = \{v_5\}.$$

7. $N \neq \emptyset \Rightarrow v_5 \rightarrow$ aktuální.



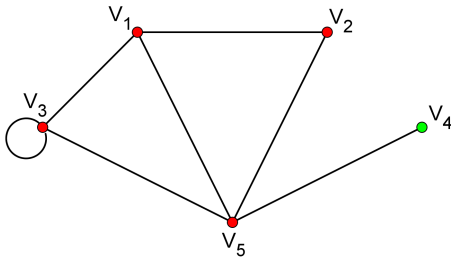
8. Hledáme sousední vrcholy aktuálního vrcholu v_5:

v_1, v_2, v_3 - zpracované;
v_4 - iniciované;
$v_5 \rightarrow$ zpracované $\rightarrow Z$;
$v_4 \rightarrow$ nalezené $\rightarrow N$.



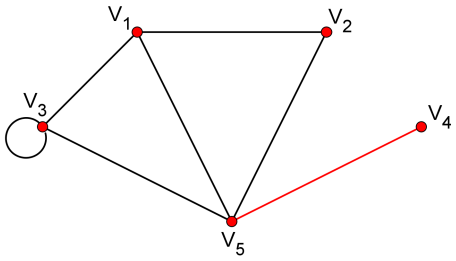
$$Z = \{v_1, v_2, v_3, v_5\}; N = \{v_4\}.$$

9. $N \neq \emptyset \Rightarrow v_4$ - aktuální.



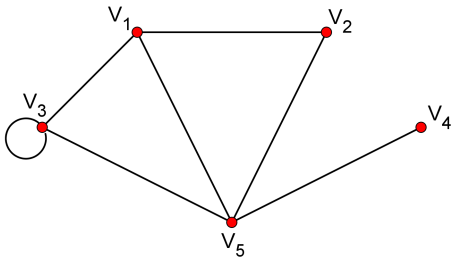
10. Hledáme sousední vrcholy aktuálního vrcholu v_4:

v_5 - zpracovaný;
$v_4 \rightarrow$ zpracovaný $\rightarrow Z$.



$$Z = \{v_1, v_2, v_3, v_4, v_5\}; N = \emptyset.$$

11. $N = \emptyset \Rightarrow$ konec algoritmu. Všechny vrcholy jsou dosažitelné.



Po aplikovaní algoritmu pro prohledávání grafu do šířky jsme zjistili, že graf má všechny vrcholy dosažitelné, jelikož množina nedosažitelných vrcholů je prázdná.

Poznámka 3.1. Oba zde uvedené algoritmy jsou jen lehce modifikovanou verzí toho samého algoritmu. K oběma lze také dále vytvořit několik různých modifikací [1].

Zde uvedené algoritmy byly upraveny kvůli odlišné interpretaci. Původní algoritmy, ze kterých tyto algoritmy vychází, je možné nalézt ve skriptech [1] v kapitole 3 na straně 24.

[1] např. pro zjištění zda je graf souvislý.

4 Hledání nejkratší cesty

V této kapitole se budeme zabývat hledáním nejkratších cest v grafu. K tomu, abychom takovou cestu mohli nalézt, musíme znát délku jednotlivých hran.

Definice 4.1. *Ohodnocením hrany* rozumíme číslo $c_i > 0$, které přiřadíme hraně $e_i \in E$ v grafu $G = (V, E)$. Toto ohodnocení se také nazývá *délka hrany.*

Definice 4.2. Graf, ve kterém mají všechny hrany svoji délku nazýváme *(hranově) ohodnocený graf.*

Poznámka 4.1. *Délkou cesty* rozumíme:

1. V hranově neohodnoceném grafu počet hran dané cesty z námi zvoleného počátečního vrcholu do námi zvoleného koncového vrcholu.
2. V hranově ohodnoceném grafu součet délek jednotlivých hran dané cesty z námi zvoleného počátečního vrcholu do námi zvoleného koncového vrcholu.

Vzdálenost

Definice 4.3. Vzdáleností $d(u, v)$ dvou vrcholů u a v v souvislém grafu $G = (V, E)$ nazveme délku nejkratší cesty mezi těmito vrcholy.

Věta 4.1. *Jsou-li délky hran v grafu kladná čísla, pak vzdálenost v grafu splňuje axiomy metriky. Tedy pro libovolné tři vrcholy u, v a w platí:*

1. $d(u, v) \geq 0$, *přičemž* $d(u, v) = 0 \Leftrightarrow u = v$;
2. $d(u, v) = d(v, u)$;
3. $d(u, v) \leq d(u, w) + d(w, v)$.

Důkaz této věty můžete nalézt například ve skriptech [7] na straně 57.

Algoritmy, pro nalezení nejkratší cesty v grafu, vychází z algoritmů na prohledávání grafu, které jsme si uvedli v předchozí kapitole. Tyto algoritmy jsou ale rozšířeny o důležité kriterium, kterým je právě délka cesty. Pokud při hledání cesty v grafu nalezneme jakoukoliv další cestu z počátečního vrcholu do jiného vrcholu, zkoušíme, zda to není kratší cesta než námi dosud nalezená. Toto ověřujeme pomocí *trojúhelníkové nerovnosti*:

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

kde

1. $c_j \ldots$ délka nejkratší dosud nalezené cesty do vrcholu v_j.
2. $c_i \ldots$ délka nejkratší dosud nalezené cesty do aktuálního vrcholu v_i, ze kterého právě hledáme novou cestu.
3. $c_{ij} \ldots$ délka aktuální hrany e_k mezi vrcholy v_i a v_j.

Poznámka 4.2. V hranově ohodnoceném grafu za nejkratší cestu považujeme tu cestu, která má nejmenší délku. Pokud je graf hranově neohodnocený, pak za nejkratší cestu považujeme tu, která obsahuje nejmenší počet hran.

Poznámka 4.3. V následujících algoritmech se budeme setkávat se symbolem ∞ v délkách cest do daných vrcholů. V praktickém programování nahradíme symbol ∞ nějakým velkým číslem, u kterého máme zaručeno, že je větší než délka nejdelší cesty v grafu G.

Poznámka 4.4. Pod pojmem *pole* o velikosti n budeme chápat uspořádanou n-tici prvků.

Matice délek

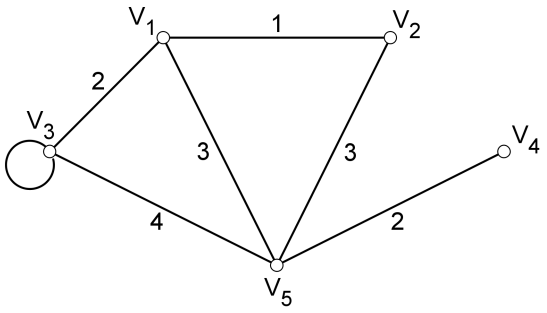
Pro hranově neohodnocené grafy můžeme využívat již zmiňovanou matici sousednosti. Pro hranově ohodnocené grafy, ale můžeme využít *matici délek* (lenghts matrix). Tato matice popisuje délky hran mezi jednotlivými vrcholy.

Definice 4.4. Nechť $G = (V, E)$ je graf. Uspořádáme-li libovolně, ale pevně, jeho množinu vrcholů $v_1, \ldots, v_n$, hran $e_1, \ldots, e_m$ a položíme-li c_{ij} rovno délce hrany mezi vrcholy v_i a v_j, pak hranově ohodnocenému neorientovanému grafu můžeme přiřadit *matici délek* H_G takto:

$$H_G = ||h_{ij}||, \text{ kde } h_{ij} = \begin{cases} c_{ij} & \text{pokud mezi vrcholy } v_i, v_j \ (i \neq j) \text{ je hrana;} \\ \infty & \text{pokud mezi vrcholy } v_i, v_j \ (i \neq j) \text{ není hrana;} \\ 0 & \text{pokud } i = j. \end{cases}$$

a v případě hranově ohodnoceného orientovaného grafu takto:

$$h_{ij} = \begin{cases} c_{ij} & \text{pokud existuje orientovaná hrana } e_k \in E(G) \text{ tak,} \\ & \text{že } v_i = PV(e_k) \text{ a } v_j = KV(e_k), \text{ kde } i \neq j; \\ \infty & \text{pokud neexistuje orientovaná hrana } e_k \in E(G) \text{ tak,} \\ & \text{že } v_i = PV(e_k) \text{ a } v_j = KV(e_k), \text{ kde } i \neq j; \\ 0 & \text{pokud } i = j. \end{cases}$$



obr. 4.1

Příklad 4.1. Matice délek pro neorientovaný graf na obrázku 4.1:

$$H_G = \begin{pmatrix} 0 & 1 & 2 & \infty & 3 \\ 1 & 0 & \infty & \infty & 3 \\ 1 & \infty & 0 & \infty & 4 \\ \infty & \infty & \infty & 0 & 2 \\ 3 & 3 & 4 & 2 & 0 \end{pmatrix}$$

Všechny následující algoritmy jsou vhodné pro orientované i neorientované grafy s kladným hodnocením hran, pokud není uvedeno jinak.

4.1 Základní algoritmus

Základní algoritmus funguje na principu procházení hran grafu, kdy zároveň zjišťujeme, které hrany splňují trojúhelníkovou nerovnost: $c_j > c_i + c_{ij}$. Pokud je tato nerovnost splněna, přepíšeme délku nejkratší cesty ve vrcholu v_j na $c_i + c_{ij}$. Všechny hrany procházíme tak dlouho, dokud nepřestane docházet ke změnám v délek cest. Tento algoritmus je velice podobný již dříve uvedenému algoritmu pro procházení grafu do hloubky.

Pomocná struktura:

1. V ... množina všech vrcholů grafu.
2. E ... množina všech hran grafu.
3. Každá hrana $e_1, \ldots, e_n \in E$ je ve tvaru $e_k = (v_i, v_j, c_{ij})$, kdy v neorientovaném grafu jsou v_i a v_j krajní vrcholy hrany e_k a c_{ij} délka dané hrany k. V orientovaném grafu je v_i počáteční a v_j koncový vrchol hrany e_k.
4. P ... pomocné pole pro uchování „předchozích“ vrcholů.

5. $C \ldots$ pomocné pole pro uchování nejkratších cest do daných vrcholů.

6. Každý vrchol je ve tvaru $v_k = (p_k, c_k)$, kdy $p_k \in P$ je předchozí vrchol, ze kterého jsme nalezli dosud nejkratší cestu do vrcholu v_k a $c_k \in C$ je nejkratší dosud nalezená cesta do vrcholu j.

Algoritmus:

1. Provedeme základní nastavení:
   (a) Vybereme libovolný počáteční vrchol a ten nastavíme na hodnotu $v_i = (0, 0)$.
   (b) Všechny ostatní vrcholy nastavíme na hodnoty $v_k = (0, \infty)$.
   (c) Pokud nemáme hrany pevně dány, pak všechny libovolně, ale pevně, uspořádáme.

2. Postupně procházíme všechny hrany a měníme hodnoty polí P a C podle následujících kriterií:
   (a) Pokud c_j splňuje nerovnost $c_j > c_i + c_{ij}$, pak:
      i. Hodnotu c_j u vrcholu v_j změníme na hodnotu $c_i + c_{ij}$ v poli P a hodnotu p_j změníme na hodnotu v_i v poli C.
      ii. Pokračujeme krokem 2.
   (b) Pokud c_j nesplňuje nerovnost $c_j > c_i + c_{ij}$, pak:
      i. Hodnoty v polích P a C u daného vrcholu v_j neměníme.
      ii. Pokračujeme krokem 3.

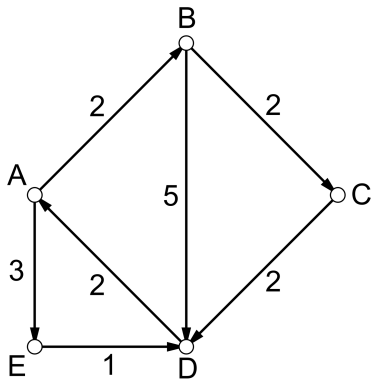
3. Pokud došlo při procházení všech hran ke změně alespoň jedné hodnoty v poli C, pokračujeme krokem 2, v opačném případě algoritmus končí.

Příklad 4.2. V hranově ohodnoceném orientovaném grafu $G = (V, E)$, který je zadán následujícím výčtem hran:

$$E(G) = \{(A, B, 2); (A, E, 3); (B, C, 2); (B, D, 5); (C, D, 2); (D, A, 2); (E, D, 1)\};$$

určete nejkratší cestu z vrcholu A do vrcholu D pomocí základního algoritmu.

Řešení:



1. Nastavíme počáteční vrchol:

$$A = (0, 0)$$

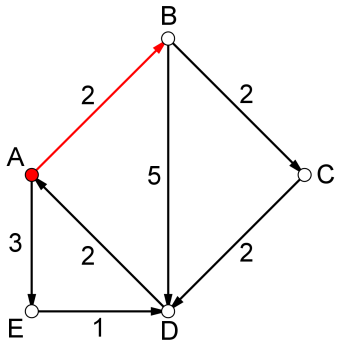
Nastavíme ostatní vrcholy:

$$B = (0, \infty); C = (0, \infty); D = (0, \infty); E = (0, \infty)$$

Všechny hrany pevně uspořádáme:

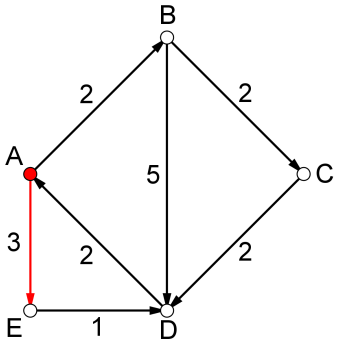
$$e_1 = (A, B, 2), e_2 = (A, E, 3), \ldots, e_7 = (E, D, 1)$$

2. Ověříme hranu e_1:



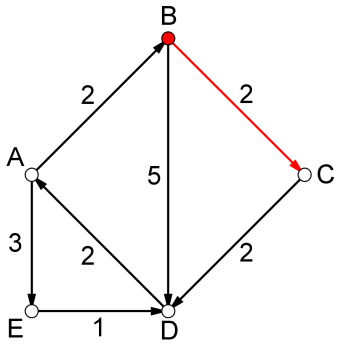
Zjistíme zda $c_B > c_A + c_{AB}$, tzn.: $\infty > 0 + 2 \Rightarrow c_B$ nerovnost splňuje, tedy $0 \to A; \infty \to 2 \Rightarrow B = (A, 2)$ a pokračujeme další hranou.

3. e_2:



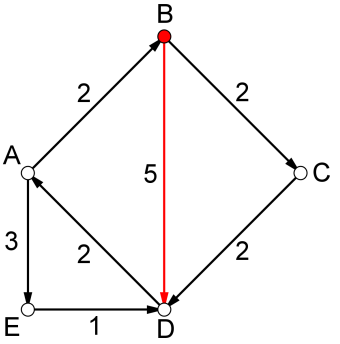
$c_E > c_A + c_{AE}$, tzn.: $\infty > 0 + 3 \Rightarrow c_E$ nerovnost splňuje,
tedy $0 \to A; \infty \to 3 \Rightarrow E = (A, 3)$ a pokračujeme hranou e_3.

4. e_3:



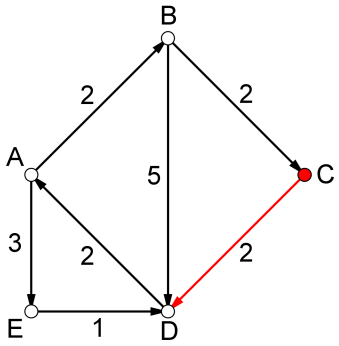
$c_C > c_B + c_{BC}$, tzn.: $\infty > 2 + 2 \Rightarrow c_C$ nerovnost splňuje
$\Rightarrow 0 \to B; \infty \to 4 \Rightarrow C = (B, 4)$ a pokračujeme hranou e_4.

5. e_4:



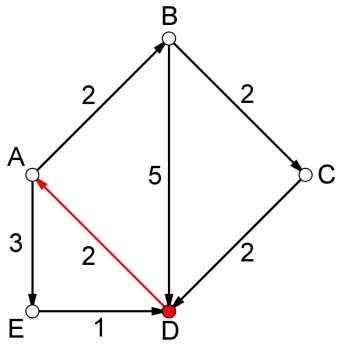
$c_D > c_B + c_{BD}$, tzn.: $\infty > 2 + 5 \Rightarrow c_D$ nerovnost splňuje
$\Rightarrow 0 \to B; \infty \to 7 \Rightarrow D = (B, 7)$ a pokračujeme hranou e_5.

6. e_5:



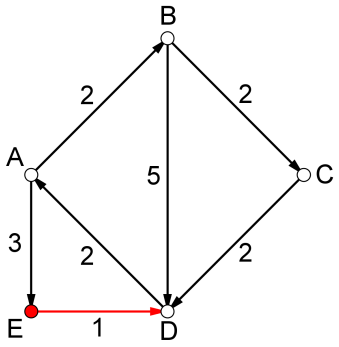
$c_D > c_C + c_{CD}$, tzn.: $7 > 4 + 2 \Rightarrow c_D$ nerovnost splňuje
$\Rightarrow B \to C; 7 \to 6 \Rightarrow D = (C, 6)$ a pokračujeme hranou e_6.

7. e_6:



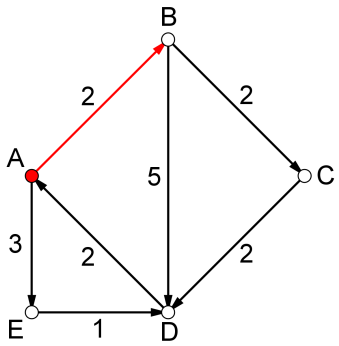
$c_A > c_D + c_{AD}$, tzn.: $0 > 6 + 2 \Rightarrow c_A$ nerovnost nesplňuje
$\Rightarrow A$ zůstává; a pokračujeme hranou e_7.

8. e_7:



$c_D > c_E + c_{DE}$, tzn.: $6 > 3 + 1 \Rightarrow c_D$ nerovnost splňuje
$\Rightarrow C \to E; 6 \to 4 \Rightarrow D = (E, 4)$ a pokračujeme znovu hranou e_1.

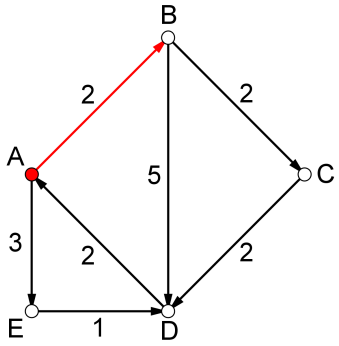
9. e_1:



$c_B > c_A + c_{AB}$, tzn.: $2 > 0 + 2 \Rightarrow c_B$ nerovnost nesplňuje
$\Rightarrow B$ zůstává a pokračujeme hranou e_2.
(Podobně jako u e_1 se v tomto případě nic na výsledku nezmění ani po znovu projití všech ostatních hran. Proto přejdeme rovnou k předposlednímu kroku.)

⋮

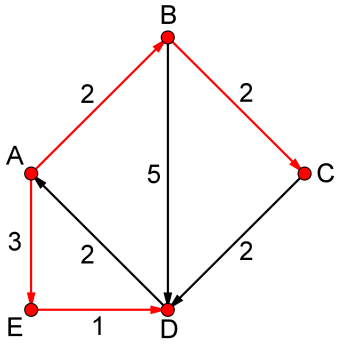
16. e_1:



$c_B > c_A + c_{AB}$. tzn.: $2 > 0 + 2 \Rightarrow c_B$ nerovnost nesplňuje
$\Rightarrow B$ zůstává a pokračujeme krokem 3.

17. Konec algoritmu. Výsledné hodnoty pro jednotlivé vrcholy jsou:

$$A = (0, 0); B = (A, 2); C = (B, 4); D = (E, 4); E = (A, 3)$$



Po projití grafu základním algoritmem pro hledání nejkratší cesty jsme zjistili, že nejkratší cesta z počátečního vrcholu A do vrcholu D vede přes vrchol E a má velikost 4.

Jak si můžeme všimnout, algoritmus nezjistil jen námi požadovanou cestu z vrcholu A do vrcholu D, ale také nejkratší cesty z vrcholu A do všech ostatních vrcholů.

Poznámka 4.5. Pokud nepotřebujeme znát z jakého vrcholu jsme do daného vrcholu přišli, pak pole P nemusíme používat.

4.2 Zlepšený algoritmus

U tohoto algoritmu vycházíme z již pro nás známého algoritmu pro prohledávání grafu do šířky. Budeme si tedy pamatovat již dosažené vrcholy a délky jednotlivých cest z výchozího (počátečního) vrcholu do již dosažených vrcholů. V každém kroku si z dosažených vrcholů vybereme vrchol, do kterého vede hrana s nejmenší délkou. Z tohoto vrcholu se pak budeme "rozhlížet"po dostupných sousedních vrcholech.

Pomocná struktura:

Pomocná struktura je stejná jako u základního algoritmu. Navíc ještě použijeme novou množinu M, ve které budou ty vrcholy, do kterých jsme v posledním kroku nalezli nejkratší cestu.

Algoritmus:

1. Provedeme základní nastavení:
   (a) Vyprázdníme množinu M.

(b) Vybereme libovolný počáteční vrchol, ten nastavíme na hodnotu $v_i = (0, 0)$ a přidáme do množiny M.

(c) Všechny ostatní vrcholy nastavíme na hodnoty $v_k = (0, \infty)$.

(d) Pokud nemáme hrany pevně dány, pak všechny libovolně, ale pevně uspořádáme.

2. Ověříme množinu M:

   (a) Jestliže $M = \emptyset$, pokračujeme krokem 4.

   (b) Jestliže $M \neq \emptyset$, vybereme z množiny M takový vrchol v_i, do kterého se dostaneme nejkratší hranou, tzn. pro který platí.

   $$v_i = min\{c_k; v_k \in M\};$$

   a pokračujeme krokem 3.

3. Procházíme všechny hrany vedoucí z vrcholu v_i, tzn. všechny sousedy v_i, a provedeme:

   (a) Odebereme v_i z množiny M.

   (b) Pokud c_j splňuje nerovnost $c_j > c_i + c_{ij}$, pak:

      i. Hodnotu c_j u vrcholu v_j změníme na hodnotu $c_i + c_{ij}$ v poli P a hodnotu p_j změníme na hodnotu v_i v poli C.

      ii. Pokud $v_j \notin M$ pak $v_j \to M$.

      iii. Pokračujeme krokem 2.

   (c) Pokud c_j nesplňuje nerovnost $c_j > c_i + c_{ij}$, pak:

      i. Hodnoty v polích P a C u daného vrcholu v_j neměníme.

      ii. Pokračujeme krokem 2.

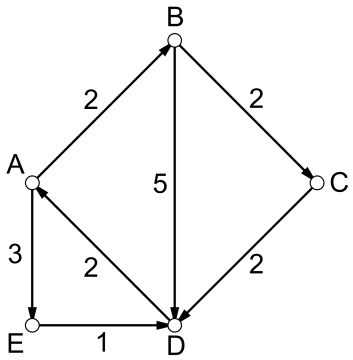
4. Konec algoritmu.

Příklad 4.3. V hranově ohodnoceném orientovaném grafu $G = (V, E)$, který je zadán následujícím výčtem hran:

$$E(G) =$$
$$\{(A, B, 2); (A, E, 3); (B, C, 2); (B, D, 5); (C, D, 2); (D, A, 2); (E, D, 1)\};$$

určete nejkratší cestu z vrcholu A do vrcholu D pomocí zlepšeného algoritmu.

Řešení:



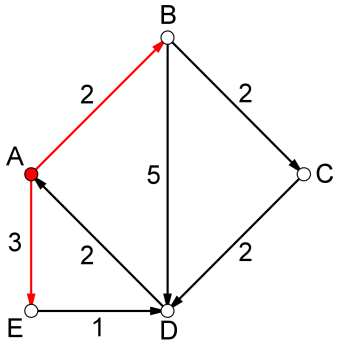
1. Nastavíme počáteční vrchol: $A = (0, 0)$ a přiřadíme do množiny M;
   Nastavíme ostatní vrcholy:

   $$B = (0, \infty); C = (0, \infty); D = (0, \infty); E = (0, \infty);$$

   Všechny hrany pevně uspořádáme:

   $$e_1 = (A, B, 2), e_2 = (A, E, 3), \ldots, e_7 = (E, D, 1);$$

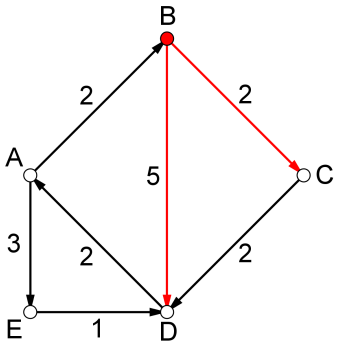
2. $M = \{A\} \neq \emptyset$, vybereme vrchol A.



3. Odebereme vrchol A z množiny M.
   $c_B > c_A + c_{AB}$, tzn.: $\infty > 0 + 2 \Rightarrow c_B$ nerovnost splňuje,
   tedy $0 \to A; \infty \to 2 \Rightarrow B = (A, 2); B \to M$.

4. $c_E > c_A + c_{AE}$, tzn.: $\infty > 0 + 3 \Rightarrow c_B$ nerovnost splňuje,
   tedy $0 \to A; \infty \to 3 \Rightarrow E = (A, 3); E \to M$.

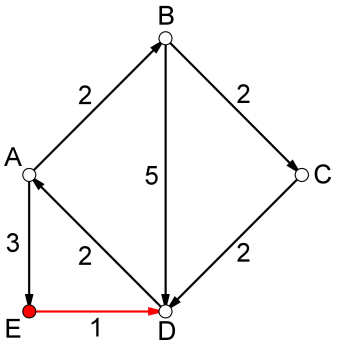
5. $M = \{B, E\} \neq \emptyset$, vybereme vrchol B, jelikož $c_B = 2$ zatímco $c_E = 3$.



6. Odebereme vrchol B z množiny M.
   $c_C > c_B + c_{BC}$, tzn.: $\infty > 2 + 2 \Rightarrow c_C$ nerovnost splňuje,
   tedy $0 \to B; \infty \to 4 \Rightarrow C = (B, 4); C \to M$.

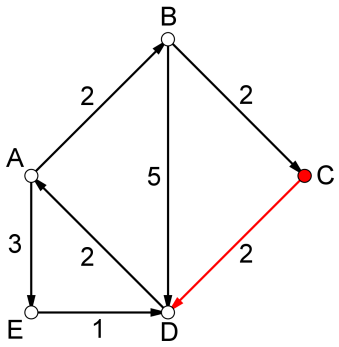
7. $c_D > c_B + c_{BD}$, tzn.: $\infty > 2 + 5 \Rightarrow c_D$ nerovnost splňuje,
   tedy $0 \to B; \infty \to 7 \Rightarrow D = (B, 7); D \to M$.

8. $M = \{C, D, E\} \neq \emptyset$, vybereme vrchol E.



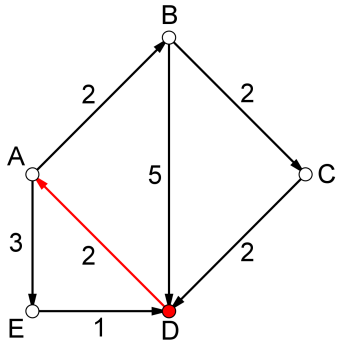
9. Odebereme vrchol E z množiny M;
   $c_D > c_E + c_{DE}$, tzn.: $7 > 3 + 1 \Rightarrow c_D$ nerovnost splňuje,
   tedy $B \to E; 7 \to 4 \Rightarrow D = (E, 4); D$ je v M již obsaženo.

10. $M = \{C, D\} \neq \emptyset$; jelikož $c_C = c_D$ můžeme si zvolit. Vybereme např. C.



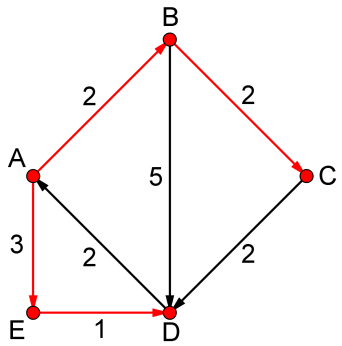
11. Odebereme vrchol C z množiny M;
$c_D > c_C + c_{CD}$, tzn.: $4 > 4 + 2 \Rightarrow c_C$ nerovnost nesplňuje $\Rightarrow D$ zůstane nezměněno.

12. $M = \{D\} \neq \emptyset$, vybereme vrchol D.



13. Odebereme vrchol D z množiny M;
$c_A > c_D + c_{AD}$, tzn.: $0 > 4 + 2 \Rightarrow c_A$ nerovnost nesplňuje $\Rightarrow A$ zůstane nezměněno.

14. $M = \emptyset \Rightarrow$ konec algoritmu. Výsledné hodnoty pro jednotlivé vrcholy jsou:



$A = (0, 0); B = (A, 2); C = (B, 4); D = (E, 4); E = (A, 3)$

Po aplikování zlepšeného algoritmu pro hledání nejkratších cest v grafu jsme nalezli nejkratší cestu z vrcholu A do vrcholu D. Nalezená nejkratší cesta vede přes vrchol E a má délku 4.

Můžeme si povšimnout, že zlepšený algoritmus našel námi hledanou nejkratší cestu mezi vrcholy A a D rychleji než předchozí základní algoritmus a stejně jako on nám také nalezl i všechny ostatní nejkratší cesty z vrcholu A do všech ostatních vrcholů.

4.3 Dijkstrův algoritmus

Jedna z nejjednodušších a nejdůležitějších metod pro hledání nejkratších cest v grafu je tzv. Dijkstrův algoritmus (čti „Dajkstrův“ - jelikož je to holandské jméno). Tento algoritmus pracuje s hranami ohodnocenými kladnými reálnými čísly.

Pomocná struktura:

1. V . . . množina všech vrcholů grafu.
2. E . . . množina všech hran grafu.
3. Každá hrana $e_1, \ldots, e_n \in E$ je ve tvaru $e_k = (v_i, v_j, c_{ij})$, kdy v neorientovaném grafu jsou v_i a v_j krajní vrcholy hrany e_k a c_{ij} délka dané hrany k. V orientovaném grafu je v_i počáteční a v_j koncový vrchol hrany e_k.
4. P . . . pomocné pole pro uchování "předchozích"vrcholů.
5. C . . . pomocné pole pro uchování nejkratších cest do daných vrcholů .
6. Každý vrchol je ve tvaru $v_k = (p_k, c_k)$, kdy $p_k \in P$ je předchozí vrchol, ze kterého jsme nalezli dosud nejkratší cestu do vrcholu v_k a $c_k \in C$ je nejkratší dosud nalezená cesta do vrcholu j.

Oproti základnímu algoritmu pro hledání nejkratších cest použijeme ještě dvě pomocné množiny:

7. $N \ldots$ množina nalezených vrcholů.
8. $I \ldots$ množina iniciovaných vrcholů.

Algoritmus:

1. Provedeme základní nastavení:
   (a) Množinu N vyprázdníme.
   (b) Vybereme libovolný počáteční vrchol, ten nastavíme na hodnotu $v_i = (0, 0)$ a přiřadíme do množiny nalezených vrcholů N.
   (c) Všechny ostatní vrcholy nastavíme na hodnoty $v_k = (0, \infty)$.
   (d) Všechny vrcholy přiřadíme do množiny iniciovaných vrcholů I.
   (e) Pokud nemáme hrany pevně dány, pak všechny libovolně, ale pevně uspořádáme.
2. Nalezneme potřebný vrchol:
   (a) Pokud množina $I = \emptyset$, pokračujeme krokem 4.
   (b) Pokud množina $I \neq \emptyset$, pokračujeme.
   (c) Z množiny iniciovaných vrcholů I vybereme takový vrchol v_i, který splňuje:
   $$c_i = min\{c_k; v_k \in N\}$$
   (d) Pokud pro tento vrchol platí, že $c_i = \infty$, pokračujeme krokem 4.
3. Procházíme dosažitelné (sousední) vrcholy v_j a testujeme, zda splňují danou nerovnost:
   (a) Pokud c_j splňuje nerovnost $c_j > c_i + c_{ij}$, pak:
      i. Hodnotu c_j u vrcholu v_j změníme na hodnotu $c_i + c_{ij}$ v poli P a hodnotu p_j změníme na hodnotu v_i v poli C.
      ii. Pokud v_j není v množině N, tak ho do ní přidáme.
      iii. Pokračujeme krokem 2.
   (b) Pokud c_j nesplňuje nerovnost $c_j > c_i + c_{ij}$, pak:
      i. Hodnoty v polích P a C u daného vrcholu v_j neměníme.
      ii. Pokud v_j není v množině N, tak ho do ní přidáme.

iii. Pokračujeme krokem 2.

(c) Vrchol v_i odebereme z množiny iniciovaných vrcholů I.

4. Konec algoritmu.

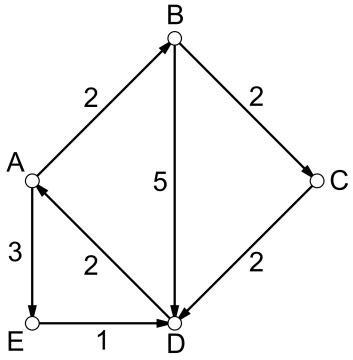
Poznámka 4.6. Důkaz správnosti tohoto algoritmu můžeme nalézt například ve skriptech [3] na straně 110.

Příklad 4.4. V hranově ohodnoceném orientovaném grafu $G = (V, E)$, který je zadán následujícím výčtem hran:

$$E(G) = \{(A, B, 2); (A, E, 3); (B, C, 2); (B, D, 5); (C, D, 2); (D, A, 2); (E, D, 1)\};$$

určete nejkratší cestu z vrcholu A do vrcholu D pomocí Dijkstrova algoritmu.

Řešení:



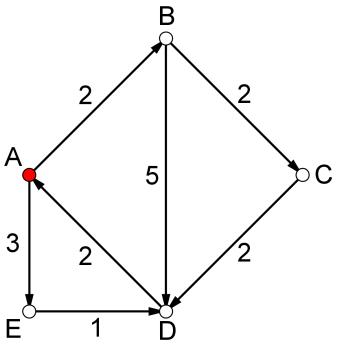
1. Množinu N vyprázdníme $\Rightarrow N = \emptyset$.
   Vybereme a nastavíme počáteční vrchol $A = (0, 0)$ a přiřadíme do množiny $N \Rightarrow N = \{A\}$.
   Nastavíme ostatní vrcholy:

   $$B = (0, \infty); C = (0, \infty); D = (0, \infty); E = (0, \infty);$$

   Všechny vrcholy označíme jako iniciované a přiřadíme do množiny:

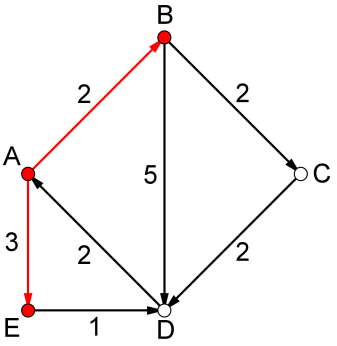
   $$I \Rightarrow I = \{A, B, C, D, E\};$$

2. Ověříme množinu I: $I = \{A, B, C, D, E\} \neq \emptyset$.
   Podle kriteria $c_i = min\{c_k; v_k \in N\}$ vybereme vrchol v_i: A.



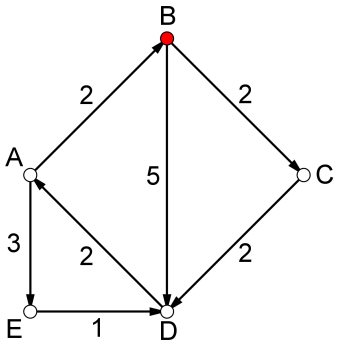
   Ověříme vrchol A: $c_A \neq \infty \Rightarrow$ pokračujeme krokem 3.

3. $c_B > c_A + c_{AB}$, tzn.: $\infty > 0 + 2 \Rightarrow c_B$ splňuje nerovnost
   tedy $\Rightarrow 0 \rightarrow A; \infty \rightarrow 2 \Rightarrow B = (A, 2)$.
   B přiřadíme do množiny nalezených vrcholů N (tzn. $B \rightarrow N$).
   $c_E > c_A + c_{AE}$, tzn.: $\infty > 0 + 3 \Rightarrow c_E$ splňuje nerovnost,
   tedy $\Rightarrow 0 \rightarrow A; \infty \rightarrow 3 \Rightarrow E = (A, 3)$.
   E přiřadíme do množiny nalezených vrcholů N (tzn. $E \rightarrow N$).

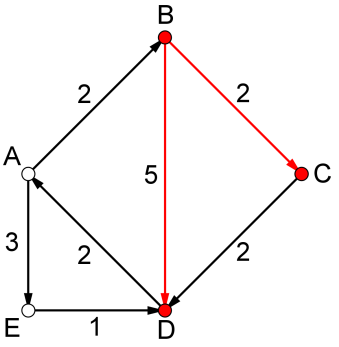


   $N = \{A, B, E\}; A$ odebereme z $I \Rightarrow I = \{B, C, D, E\}$.

4. Pokračujeme stejným stylem dále:
   $I = \{B, C, D, E\} \neq \emptyset$. Vybereme vrchol B. $c_B \neq \infty \Rightarrow$ pokračujeme.

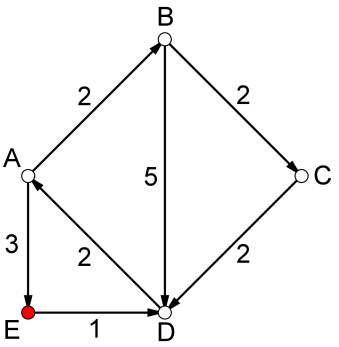


5. $c_C > c_B + c_{BC}$, tzn.: $\infty > 2 + 2 \Rightarrow c_C$ splňuje nerovnost,
   tedy $\Rightarrow C = (B, 4); C \rightarrow N$.
   $c_D > c_B + c_{BD}$, tzn.: $\infty > 2 + 5 \Rightarrow c_D$ splňuje nerovnost,
   tedy $\Rightarrow D = (B, 7); D \rightarrow N$.

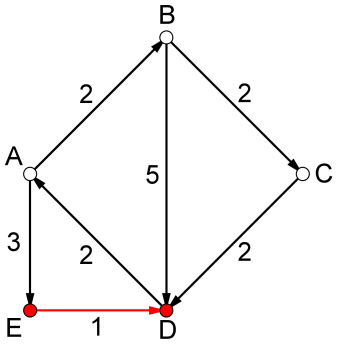


   $N = \{A, B, C, D, E\}; B$ odebereme z $I \Rightarrow I = \{C, D, E\}$.

6. $I = \{C, D, E\} \neq \emptyset$. Vybereme vrchol E. $c_E \neq \infty \Rightarrow$ pokračujeme.

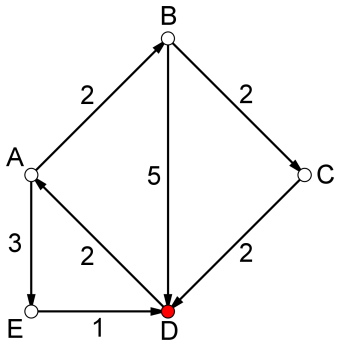


7. $c_D > c_E + c_{DE}$, tzn.: $7 > 3 + 1 \Rightarrow c_D$ splňuje nerovnost, tedy $\Rightarrow D = (E, 4)$; D již v množině N je.

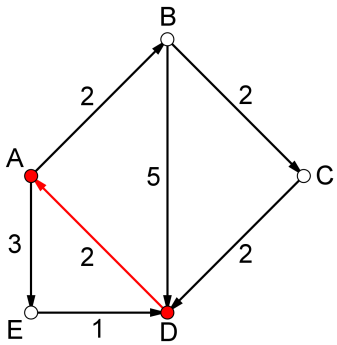


$N = \{A, B, C, D, E\}$; E odebereme z $I \Rightarrow I = \{C, D\}$.

8. $I = \{C, D\} \neq \emptyset$. Vybereme například vrchol D, jelikož $c_D = c_C$. $c_D \neq \infty \Rightarrow$ pokračujeme.

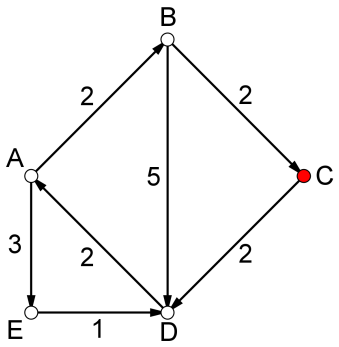


9. $c_A > c_D + c_{AD}$, tzn.: $0 > 4 + 2 \Rightarrow c_A$ nerovnost nesplňuje, tedy A zůstává; A již v množině N je.

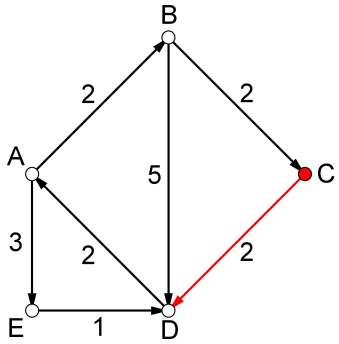


$N = \{A, B, C, D, E\}$; D odebereme z $I \Rightarrow I = \{C\}$.

10. $I = \{C\} \neq \emptyset$. Vybereme vrchol C; $c_C \neq \infty \Rightarrow$ pokračujeme.

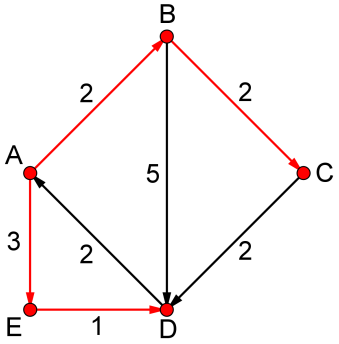


11. $c_D > c_C + c_{CD}$, tzn.: $4 > 4 + 2 \Rightarrow c_D$ nerovnost nesplňuje, tedy D zůstává; D již v množině N je.



$N = \{A, B, C, D, E\}$; C odebereme z $I \Rightarrow I = \emptyset$.

12. $I = \emptyset \Rightarrow$ konec algoritmu. Výsledné hodnoty pro jednotlivé vrcholy jsou:



$A = (0, 0); B = (A, 2); C = (B, 4); D = (E, 4); E = (A, 3)$

Pomocí Dijkstrova algoritmu jsme nalezli nejkratší cestu z vrcholu A do vrcholu D vedoucí přes vrchol E o délce 4.

Tento algoritmus je efektivnější a méně pracný na počítání než zlepšený algoritmus a už podstatně efektivnější než základní algoritmus pro hledání nejkratších cest.

4.4 Bellman-Fordův algoritmus

Tento algoritmus je velice podobný dvěma předchozím algoritmům. Oproti nim navíc pro vrcholy obsahuje pole H, ve kterém je u každého vrcholu uvedeno, z kolika hran se skládá nejkratší cesta z počátečního do tohoto vrcholu. Tento algoritmus preferuje nejkratší cestu s nejmenším počtem hran. Jelikož víme, že graf o n vrcholech obsahuje cestu délky nejvýše $(n-1)$ hran, můžeme tento algoritmus použít také pro zjištění, zda daný graf neobsahuje cyklus se záporným ohodnocením hran (pokud algoritmus neskončí po $(n-1)$ krocích).

Pomocná struktura:

1. V ... množina všech vrcholů grafu.
2. E ... množina všech hran grafu.
3. Každá hrana $e_1, \ldots, e_n \in E$ je ve tvaru $e_k = (v_i, v_j, c_{ij})$, kdy v neorientovaném grafu jsou v_i a v_j krajní vrcholy hrany e_k a c_{ij} délka dané hrany k. V orientovaném grafu je v_i počáteční a v_j koncový vrchol hrany e_k.
4. P ... pomocné pole pro uchování „předchozích“ vrcholů.
5. C ... pomocné pole pro uchování nejkratších cest do daných vrcholů.
6. H ... pomocné pole pro uchování informace o počtu hran, kterými jsme se do daného vrcholu dostali.
7. k ... pomocná proměnná pro uchování informace o aktuální délce cesty.
8. n ... pomocná proměnná pro uchování informace o počtu hran grafu.
9. Každý vrchol je ve tvaru $v_k = (h_m, p_n, c_k)$, kdy $h_m \in H$ je počet hran cesty, pomocí které jsme se do daného vrcholu dostali, $p_n \in P$ je předchozí vrchol, ze kterého jsme nalezli dosud nejkratší cestu do vrcholu v_k a $c_k \in C$ je nejkratší dosud nalezená cesta do vrcholu j.

Algoritmus:

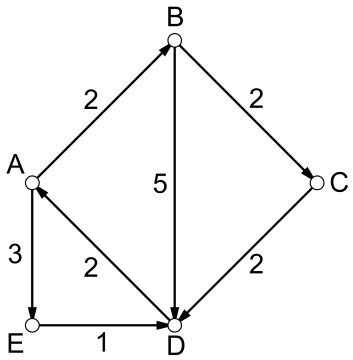
1. Provedeme základní nastavení:
   (a) Vybereme libovolný počáteční vrchol, ten nastavíme na hodnotu $v_i = (0, 0, 0)$.
   (b) Všechny ostatní vrcholy nastavíme na hodnoty $v_k = (-1, 0, \infty)$.
   (c) Pomocnou proměnou k nastavíme na $k = 0$.
   (d) Pokud nemáme hrany pevně dány, pak všechny libovolně, ale pevně uspořádáme.
   (e) Počet hran označíme n.
2. Vybereme vrcholy, pro které $h_i = k$ a pro každý takhle nalezený vrchol v_i postupně procházíme všechny vrcholy, do kterých se z daného vrcholu dostaneme a zjišťujeme, zda splňují nerovnost $c_j > c_i + c_{ij}$.
   (a) Pokud nenalezneme ani jeden vrchol pro který $h_j = k$, algoritmus končí.
   (b) Hodnotu k zvětšíme o 1.
   (c) Pokud $k = n - 1$, pak algoritmus končí jinak pokračujeme dále.
   (d) Pokud c_j splňuje nerovnost $c_j > c_i + c_{ij}$, pak:
      i. Hodnotu c_j u vrcholu v_j změníme na hodnotu $c_i + c_{ij}$ v poli P, hodnotu p_j změníme na hodnotu v_i v poli C a hodnotu h_j změníme na hodnotu k v poli H.
      ii. Pokračujeme dalším nalezeným vrcholem splňujícím rovnost $h_i = k$.
   (e) Pokud c_j nesplňuje nerovnost $c_j > c_i + c_{ij}$, pak:
      i. Hodnoty v polích H, P a C u daného vrcholu v_j neměníme.
      ii. Pokračujeme dalším nalezeným vrcholem splňujícím rovnost $h_i = k$.
   (f) Pokud jsme prošli všechny vrcholy, které splňují rovnost $h_i = k$, pokračujeme krokem 2 znovu od začátku.

Příklad 4.5. V hranově ohodnoceném orientovaném grafu $G = (V, E)$, který je zadán následujícím výčtem hran:

$$E(G) = \{(A, B, 2); (A, E, 3); (B, C, 2); (B, D, 5); (C, D, 2); (D, A, 2); (E, D, 1)\};$$

určete nejkratší cestu z vrcholu A do vrcholu D pomocí Bellman-Fordova algoritmu.

Řešení:



1. Nastavíme počáteční vrchol:

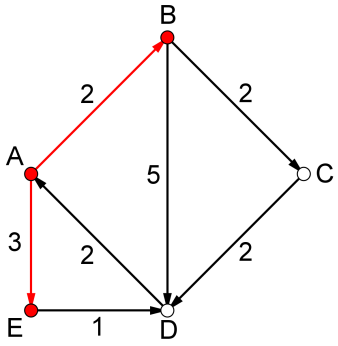
$$A = (0, 0, 0);$$

Nastavíme ostatní vrcholy:

$$B = (-1, 0, \infty); C = (-1, 0, \infty); D = (-1, 0, \infty); E = (-1, 0, \infty);$$

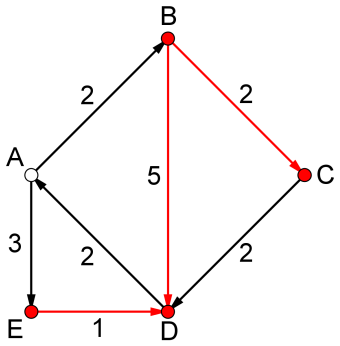
Nastavíme pomocnou proměnou k: $k = 0$.
Spočítáme a nastavíme počet hran: $n = 7$.

2. Vybereme vrcholy které splňují $h_i = k = 0$. Toto splňuje vrchol A.
k zvětšíme o 1 a ověříme zda $k = n - 1$.
$k = 1 \neq 6 \Rightarrow$ algoritmus pokračuje.
$c_B > c_A + c_{AB}$, tzn.: $\infty > 0 + 2 \Rightarrow c_B$ nerovnost splňuje, proto změníme daná pole: $-1 \to 1 = k; 0 \to A; \infty \to 2 \Rightarrow B = (1, A, 2)$.
Pokračujeme další hranou:
$c_E > c_A + c_{AE}$, tzn.: $\infty > 0 + 3 \Rightarrow c_E$ nerovnost splňuje, proto změníme daná pole: $-1 \to 1 = k; 0 \to A; \infty \to 3 \Rightarrow E = (1, A, 3)$.

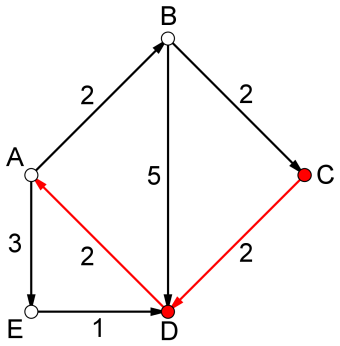


Pokračujeme znovu od začátku kroku 2.

3. $h_i = k = 1 \Rightarrow$ splňuje B, E; $k \to 2; k \neq 6$;
   $c_C > c_B + c_{BC}$, tzn.: $\infty > 2 + 2 \Rightarrow C = (2, B, 4)$;
   $c_D > c_B + c_{BD}$, tzn.: $\infty > 2 + 5 \Rightarrow D = (2, B, 7)$;
   $c_D > c_E + c_{DE}$, tzn.: $7 > 3 + 1 \Rightarrow D = (2, E, 4)$;

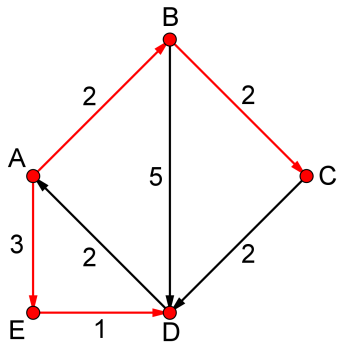


4. $h_i = k = 2 \Rightarrow$ splňuje C, D; $k \to 3; k \neq 6$;
   $c_D > c_C + c_{CD}$, tzn.: $4 > 4 + 2 \Rightarrow D$ zůstává nezměněno;
   $c_A > c_D + c_{AD}$, tzn.: $0 > 4 + 2 \Rightarrow A$ zůstává nezměněno.



5. $h_i = k = 3 \Rightarrow$ nesplňuje žádný vrchol $\Rightarrow$ algoritmus končí.
   Výsledné hodnoty pro jednotlivé vrcholy jsou:

$$A = (0, 0, 0); B = (1, A, 2); C = (2, B, 4); D = (2, E, 4); E = (1, A, 3)$$



Po užití Bllman-Fordova algoritmu pro nalezení nejkratší cesty jsme zjistili, že nejkratší cesta z vrcholu A do vrcholu D vede přes 2 hrany, přes vrchol E a je délky 4.

4.5 Zlepšený algoritmus pro nalezení nejkratší orientované cesty

Topologické uspořádání vrcholů a hran je další možnost, která nám může pomoci při hledání nejkratších cest v grafu mezi jednotlivými vrcholy.

Definice 4.5. *Topologické uspořádání vrcholů* grafu $G = (V, E)$ je taková posloupnost vrcholů pro kterou platí:

$$\forall e \in E(G), v_i = PV(e) \wedge v_j = KV(e) : i < j$$

Nebo-li, je to takové seřazení vrcholů, že počáteční vrchol v_i každé hrany e grafu G leží před jejím koncovým vrcholem v_j.

Definice 4.6. *Topologické uspořádání hran* grafu $G = (V, E)$ je taková posloupnost vrcholů pro kterou platí:

$$\forall v \in V(G), v = KV(e_i) \wedge v = PV(e_j) : i < j$$

Nebo-li, je to takové seřazení hran, že hrany e_i které do vrcholu v vstupují leží před hranami e_j, které z vrcholu v vystupují.

Topologické uspořádání vrcholů a hran můžeme využít při hledání nejkratších cest. Díky tomuto uspořádání víme, že z vrcholu s vyšším stupněm (indexem) neexistuje cesta do vrcholu s nižším stupněm (indexem). Tím můžeme snížit počet trojúhelníkových nerovností, které musíme v algoritmu provést, případně můžeme ihned rozhodnout o nedostupnosti daného vrcholu. Zde uvedený algoritmus je určený pro hranově neohodnocené orientované i neorientované grafy.

Pomocná struktura:

1. $V \ldots$ množina všech vrcholů grafu.
2. $E \ldots$ množina všech hran grafu.
3. Každá hrana $e_1, \ldots, e_n \in E$ je ve tvaru $e_k = (v_i, v_j)$, kdy v neorientovaném grafu jsou v_i a v_j krajní vrcholy hrany e_k. V orientovaném grafu je v_i počáteční a v_j koncový vrchol hrany e_k.
4. $V_s \ldots$ pole vstupních stupňů jednotlivých vrcholů.
5. $P_v \ldots$ pole pro posloupnosti vrcholů.
6. $P_e \ldots$ pole pro posloupnosti hran.
7. $M \ldots$ množina pro vrcholy s nulovým vstupním stupněm (tzn.: $V_{s_i} = 0$).

Algoritmus:

1. Provedeme výpočet vstupních stupňů jednotlivých vrcholů:
   (a) Vyprázdníme pole V_s, P_v a P_e a množinu M.
   (b) Stupně všech vrcholů nastavíme na hodnotu 0 a zvolíme si počáteční vrchol.
   (c) Projdeme jednou všechny hrany ve tvaru $e_k = (v_i, v_j)$ a pokud v_j je koncový vrchol hrany e_k, pak vstupní stupeň V_{s_j} vrcholu v_j zvětšíme o 1. U námi zvoleného počátečního vrcholu necháme stupeň nastaven na hodnotě 0.
   (d) Všechny vrcholy, které po projití všech hran mají vstupní stupeň roven 0, přiřadíme do množiny M.
2. Ověříme množinu M:
   (a) Pokud $M = \emptyset$, pak algoritmus končí.
   (b) Pokud $M \neq \emptyset$:
      i. Vybereme libovolný vrchol v_m z množiny M a zařadíme na konec pole posloupnosti vrcholů P_v.
      ii. Pokračujeme krokem 3.

3. Pro každou hranu $e_k = (v_m, v_n)$, kdy v_m je počáteční vrchol hrany e_k, provedeme (jinak pokračujeme krokem 2):

   (a) Hranu e_k zařadíme na konec pole posloupnosti hran P_e.

   (b) Stupeň vrcholu v_n snížíme o 1 (tzn.: $V_{s_n} = V_{s_n} - 1$) a je-li nyní stupeň $V_{s_n} = 0$, vrchol v_n zařadíme do množiny M.

   (c) Pokud je stupeň vrcholu v_n záporný, odebereme poslední přidanou hranu do pole P_e.

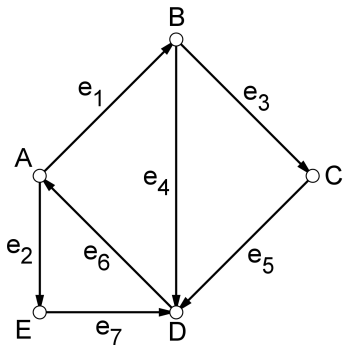
   (d) Pokračujeme krokem 2.

Příklad 4.6. Topologicky uspořádejte hrany a vrcholy v hranově neohodnoceném orientovaném grafu $G = (V, E)$, který je zadán následujícím výčtem hran:

$$E(G) = \{(A, B); (A, E); (B, C); (B, D); (C, D); (D, A); (E, D)\};$$

pomocí příslušného algoritmu.

Řešení:



1. Stupně všech vrcholů změníme na hodnotu 0:

   $$deg(A) = 0; deg(B) = 0; deg(C) = 0; deg(D) = 0; deg(E) = 0;$$

   Vybereme si počáteční vrchol, např. vrchol A. Projdeme jednou všechny hrany a nastavíme příslušné stupně jednotlivých vrcholů (nesmíme zapomenou, že u námi vybraného počátečního vrcholu stupeň zůstává na hodnotě 0):

   $$deg(A) = 0; deg(B) = 1; deg(C) = 1; deg(D) = 3; deg(E) = 1;$$

tedy

$$V_s = (0, 1, 1, 3, 1)$$

Vrcholy, které mají stupeň roven 0 přiřadíme do množiny M:
$deg(A) = 0 \Rightarrow A \rightarrow$ množiny M;

2. $M = \{A\} \neq \emptyset$, proto vybereme vrchol A a zařadíme ho na konec pole posloupnosti vrcholů P_v, tedy:

$$P_v = (0, 0, 0, 0, A)$$

Pokračujeme krokem 3;

3. Nalezneme všechny hrany, pro které je vrchol A počátečním vrcholem:
e_1 zařadíme na konec pole posloupnosti hran P_e, tedy:

$$P_e = (0, 0, 0, 0, 0, 0, e_1)$$

Snížíme stupeň příslušného konečného vrcholu hrany e_1:

$$deg(B) = 1 \rightarrow deg(B) = 0$$

a přiřadíme do množiny M, jelikož $deg(B) = 0$, a pokračujeme další nalezenou hranou;
e_2 zařadíme na konec pole posloupnosti hran P_e, tedy:

$$P_e = (0, 0, 0, 0, 0, e_1, e_2)$$

Snížíme stupeň příslušného konečného vrcholu hrany e_2:

$$deg(E) = 1 \rightarrow deg(E) = 0$$

a přiřadíme do množiny M, jelikož $deg(E) = 0$. Jelikož jsme prošli všechny hrany začínající ve vrcholu A, pokračujeme krokem 2.

4. $M = \{B, E\} \neq \emptyset \Rightarrow$ vybereme například vrchol E a zařadíme na konec P_v, tedy:

$$P_v = (0, 0, 0, A, E)$$

a pokračujeme krokem 3.

5. e_7 zařadíme na konec pole posloupnosti hran P_e, tedy:

$$P_e = (0, 0, 0, 0, e_1, e_2, e_7)$$

Snížíme stupeň příslušného konečného vrcholu hrany e_7:

$$deg(D) = 3 \rightarrow deg(D) = 2$$

a jelikož $deg(D) \neq 0$, nepřiřazujeme vrchol do množiny M a pokračujeme krokem 2.

6. $M = \{B\} \neq \emptyset \Rightarrow$ vybereme vrchol B a zařadíme na konec P_v, tedy:

$$P_v = (0, 0, A, E, B)$$

a pokračujeme krokem 3.

7. e_3 zařadíme na konec pole posloupnosti hran P_e, tedy:

$$P_e = (0, 0, 0, e_1, e_2, e_7, e_3)$$

Snížíme stupeň příslušného konečného vrcholu hrany e_3:

$$deg(C) = 1 \rightarrow deg(C) = 0$$

a přiřadíme do množiny M, jelikož $deg(C) = 0$, a pokračujeme další nalezenou hranou;
e_4 zařadíme na konec pole posloupnosti hran P_e, tedy:

$$P_e = (0, 0, e_1, e_2, e_7, e_3, e_4)$$

Snížíme stupeň příslušného konečného vrcholu hrany e_4:

$$deg(D) = 2 \rightarrow deg(D) = 1$$

vrchol D nepřiřadíme do množiny M, jelikož $deg(D) = 1$. Prošli jsme všechny hrany začínající ve vrcholu A, proto pokračujeme krokem 2.

8. $M = \{C\} \neq \emptyset \Rightarrow$ vybereme vrchol C a zařadíme na konec P_v, tedy:

$$P_v = (0, A, E, B, C)$$

a pokračujeme krokem 3;

9. e_5 zařadíme na konec pole posloupnosti hran P_e, tedy:

$$P_e = (0, e_1, e_2, e_7, e_3, e_4, e_5)$$

Snížíme stupeň příslušného konečného vrcholu hrany e_5:

$$deg(D) = 1 \rightarrow deg(D) = 0$$

a vrchol D přiřadíme do množiny M, jelikož $deg(D) = 0$. Prošli jsme všechny hrany začínající ve vrcholu A, proto pokračujeme krokem 2.

10. $M = \{D\} \neq \emptyset \Rightarrow$ vybereme vrchol D a zařadíme na konec P_v, tedy:

$$P_v = (A, E, B, C, D);$$

pokračujeme krokem 3.

11. e_6 zařadíme na konec pole posloupnosti hran P_e, tedy:

$$P_e = (e_1, e_2, e_7, e_3, e_4, e_5, e_6)$$

Snížíme stupeň příslušného konečného vrcholu hrany e_6:

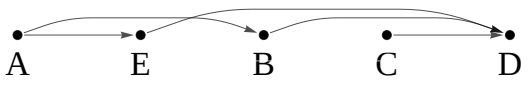
$$deg(A) = 0 \rightarrow deg(A) = -1$$

Jelikož vrchol A má záporný stupeň, nepřiřadíme ho do množiny M a odstraníme naposledy přidanou hranu do pole P_e, tedy:

$$P_e = (0, e_1, e_2, e_7, e_3, e_4, e_5)$$

pokračujeme krokem 2.

12. $M = \emptyset \Rightarrow$ konec algoritmu. Nyní můžeme ze získaných polí sestavit Topologický graf.



Poznámka 4.7. Tento algoritmus můžeme použít také na zjištění cykličnosti/acykličnosti grafu. Pokud je graf cyklický, pak algoritmus skončí dříve, než do pole P_v uloží všechny vrcholy z množiny vrcholů V.

4.6 Hledání nejkratší cesty v acyklických grafech

Nyní se budeme zabývat určitou modifikací dříve uvedených algoritmů pro topologické uspořádání vrcholů a hran a nalezení nejkratší cesty. V acyklických grafech totiž můžeme snížit počet hran, které testujeme pomocí trojúhelníkové nerovnosti. Podmínkou ovšem je, že musíme znát topologické uspořádání vrcholů a hran grafu. Neznáme-li topologické uspořádání vrcholů, ale víme-li, že daný graf je acyklický, pak upraveným algoritmem pro topologické uspořádání grafu, můžeme získat délky nejkratších cest z počátečního (výchozího) vrcholu do všech dostupných vrcholů.

Pomocná struktura:

1. $V \ldots$ množina všech vrcholů grafu.
2. Každý vrchol je ve tvaru $v_k = (V_{s_k}, p_n, c_k)$, kdy V_{s_k} je vstupní stupeň vrcholu v_k, $p_n \in P$ je předchozí vrchol, ze kterého jsme nalezli dosud nejkratší cestu do vrcholu v_k a $c_k \in C$ je nejkratší dosud nalezená cesta do vrcholu v_k.
3. $E \ldots$ množina všech hran grafu.
4. Každá hrana $e_1, \ldots, e_n \in E$ je ve tvaru $e_k = (v_i, v_j, c_{ij})$, kdy v_i je počáteční vrchol hrany e_k, v_j koncový vrchol hrany e_k a c_{ij} je délka hrany mezi vrcholy v_i a v_j tedy délka hrany e_k.
5. $V_s \ldots$ pole vstupních stupňů jednotlivých vrcholů.
6. $P_v \ldots$ pole pro posloupnosti vrcholů.
7. $P_e \ldots$ pole pro posloupnosti hran.
8. $M \ldots$ množina pro vrcholy s nulovým vstupním stupněm (tzn.: $V_{s_i} = 0$).
9. $P \ldots$ pomocné pole pro uchování „předchozích“ vrcholů.
10. $C \ldots$ pomocné pole pro uchování nejkratších cest do daných vrcholů.

Algoritmus:

1. Provedeme základní nastavení:
   (a) Vyprázdníme pole V_s, P_v a P_e a množinu M.
   (b) Stupně všech vrcholů nastavíme na hodnotu 0.

(c) Projdeme jednou všechny hrany ve tvaru $e_k = (v_i, v_j, c_{ij})$ a pokud v_j je koncový vrchol (KV) hrany e_k, pak vstupní stupeň V_{s_j}. vrcholu v_j zvětšíme o 1.

(d) Zvolíme si libovolný počáteční vrchol a nastavíme na $v_i = (V_{s_i}, 0, 0)$.

(e) Všechny ostatní vrcholy nastavíme na $v_k = (V_{s_k}, 0, \infty)$.

(f) Vrcholy, u kterých nám zůstal vstupní stupeň 0 zařadíme do množiny M.

2. Ověříme množinu M

   (a) Pokud $M = \emptyset$, pak algoritmus končí.

   (b) Pokud $M \neq \emptyset$, pak:

      i. Vybereme libovolný vrchol v_m z množiny M a zařadíme na konec pole posloupnosti vrcholů P_v.

      ii. Pokračujeme krokem 3.

3. Pro každou hranu $e_k = (v_m, v_n, c_{mn})$, kdy v_m je počáteční vrchol hrany e_k a c_{mn} je délka hrany mezi vrcholy v_m a v_n, provedeme (jinak pokračujeme krokem 2):

   (a) Hranu e_k zařadíme na konec pole posloupnosti hran P_e.

   (b) Stupeň vrcholu v_n snížíme o 1 (tzn.: $V_{s_n} = V_{s_n} - 1$) a je-li nyní stupeň $V_{s_n} = 0$, vrchol v_n zařadíme do množiny M.

   (c) Pokud vrchol v_m je v topologickém uspořádání a tedy i v poli P_v dříve, než náš vybraný počáteční vrchol, neprovádíme další krok (d), ale pokračujeme krokem (e).

   (d) Vezmeme vrchol v_n a zjistíme, zda c_n splňuje nerovnost

   $$c_n > c_m + c_{mn}$$

   mohou nastat tyto možnosti:

      i. Pokud c_n splňuje nerovnost, pak hodnotu c_n u vrcholu v_n změníme na hodnotu $c_m + c_{mn}$ v poli P a hodnotu p_n změníme na hodnotu v_m v poli C.

      ii. Pokud c_n nesplňuje nerovnost, pak hodnoty v polích P a C u vrcholu v_n neměníme.

   (e) Pokračujeme další hranou. Pokud jsme již prošli všechny hrany vycházející z vrcholu v_m, pokračujeme krokem 2.

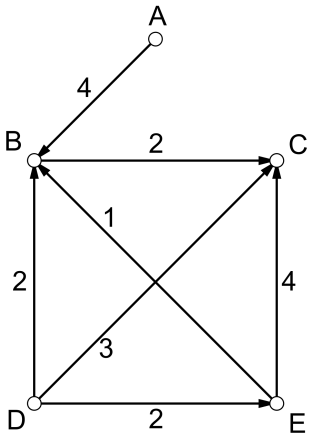
Poznámka 4.8. Algoritmus se zjednoduší tím, že trojúhelníkovou nerovností netestujeme vrcholy, které leží před námi zvoleným počátečním (výchozím) vrcholem.

Příklad 4.7. Najděte nejkratší cestu v orientovaném hranově ohodnoceném grafu $G = (V, E)$, který je zadán následujícím výčtem hran:

$$E(G) = \{(A, B, 4); (B, C, 2); (D, B, 2); (D, C, 3); (D, E, 2); (E, B, 1); (E, C, 4)\};$$

z vrcholu E do vrcholu C pomocí algoritmu pro hledání nejkratší cesty v acyklických grafech.

Řešení:



1. Vyprázdníme pole V_s, P_v a P_e a množinu M, stupně vrcholů nastavíme na hodnotu 0 a projdeme jednou všechny hrany, jejichž vstupní stupně vrcholů nastavíme:

   $$deg(A) = 0; deg(B) = 3; deg(C) = 3; deg(D) = 0; deg(E) = 1;$$

   Zvolíme si počáteční vrchol, např.: E a nastavíme na:

   $$E = (1, 0, 0);$$

   Nastavíme ostatní vrcholy:

   $$A = (0, 0, \infty); B = (3, 0, \infty); C = (3, 0, \infty); D = (0, 0, \infty);$$

2. $M = \{A, D\} \neq \emptyset \Rightarrow$ vybereme například vrchol A a zařadíme ho na konec pole posloupnosti vrcholů P_v:

$$P_v = (0, 0, 0, 0, A)$$

3. Vybereme hranu e_1 a přiřadíme ji do pole posloupnosti hran P_e:

$$P_e = (0, 0, 0, 0, 0, 0, e_1)$$

Snížíme stupeň konečného vrcholu hrany e_1:

$$deg(B) = 3 \rightarrow deg(B) = 2$$

Stupeň vrcholu B není roven 0, proto ho nezařadíme do množiny M a jelikož vrchol A je před naším zvoleným počátečním vrcholem E, pokračujeme až bodem 2;

4. $M = \{D\} \neq \emptyset \Rightarrow$ vybereme vrchol D a zařadíme na konec P_v:

$$P_v = (0, 0, 0, A, D)$$

5. Vybereme hranu e_3 a zařadíme do P_e:

$$P_e = (0, 0, 0, 0, 0, e_1, e_3)$$

Snížíme stupeň konečného vrcholu hrany e_2:

$$deg(B) = 2 \rightarrow deg(B) = 1$$

$deg(B) \neq 0 \Rightarrow$ nezařadíme do množiny M a jelikož i vrchol D je před naším zvoleným počátečním vrcholem, pokračujeme další hranou e_4:

$$P_e = (0, 0, 0, 0, e_1, e_3, e_4);$$
$$deg(C) = 3 \rightarrow deg(C) = 2$$

$deg(C) \neq 0 \Rightarrow$ nezařadíme do množiny M a pokračujeme další hranou e_5:

$$P_e = (0, 0, 0, e_1, e_3, e_4, e_5);$$
$$deg(E) = 1 \rightarrow deg(E) = 0$$

$deg(E) = 0 \Rightarrow$ vrchol E přidáme do množiny M a pokračujeme krokem 2.

6. $M = \{E\} \neq \emptyset \Rightarrow$ vybereme vrchol E a zařadíme:

$$P_v = (0, 0, A, D, E)$$

7. Vybereme hranu e_6:

$$P_e = (0, 0, e_1, e_3, e_4, e_5, e_6);$$
$$deg(B) = 1 \rightarrow deg(B) = 0$$

$deg(B) = 0 \Rightarrow$ vrchol B přidáme do množiny M. Nyní vezmeme vrchol B a ověříme, zda c_B splňuje nerovnost:

$$c_B > c_E + c_{EB}, \text{ tzn.:} \infty > 0 + 1 \Rightarrow B = (0, E, 1)$$

Pokračujeme další hranou e_7:

$$P_e = (0, e_1, e_3, e_4, e_5, e_6, e_7);$$
$$deg(C) = 2 \rightarrow deg(C) = 1;$$

Ověříme, zda c_C splňuje nerovnost:

$$c_C > c_E + c_{EC}, \text{ tzn.:} \infty > 0 + 4 \Rightarrow C = (1, E, 4)$$

Prošli jsme všechny hrany jdoucí z vrcholu E, pokračujeme dalším krokem.

8. $M = \{B\} \neq \emptyset \Rightarrow$ vybereme vrchol B a zařadíme:

$$P_v = (0, A, D, E, B)$$

9. Vybereme hranu e_2:

$$P_e = (e_1, e_3, e_4, e_5, e_6, e_7, e_2);$$
$$deg(C) = 1 \rightarrow deg(C) = 0$$

$deg(C) = 0 \Rightarrow$ vrchol C přidáme do M a ověříme nerovnost c_C:

$$c_C > c_B + c_{BC}, \text{ tzn.:} 4 > 1 + 2 \Rightarrow C = (0, B, 3);$$

Prošli jsme všechny hrany z vrcholu B, pokračujeme dalším krokem.

10. $M = \{C\} \neq \emptyset \Rightarrow$ vybereme C.

11. Vrchol C nemá žádné z něj vedoucí hrany, proto pokračujeme znovu krokem 2.

12. $M = \emptyset \Rightarrow$ konec algoritmu.

Pomocí algoritmu pro hledání nejkratších cest v acyklických grafech jsme zjistili, že $C = (0, B, 3)$, tedy, že nejkratší cesta z vrcholu E do vrcholu C vede přes vrchol B a má délku 3.

Matice vzdáleností

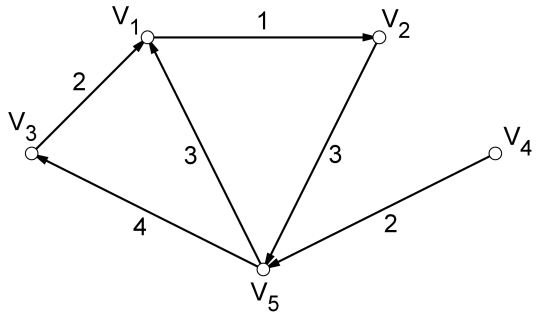
Matice vzdáleností (distance matrix) nám udává délku jednotlivých cest mezi všemi vrcholy. Definujeme ji takto:

Definice 4.7. Nechť $G = (V, E)$ je graf. Uspořádáme-li libovolně, ale pevně jeho množinu vrcholů $v_1, \ldots, v_n$, hran $e_1, \ldots, e_m$ a položíme-li c_{ij} rovno vzdálenosti (nejkratší cestě) mezi vrcholy v_i a v_j, pak hranově ohodnocenému neorientovanému grafu můžeme přiřadit *matici vzdáleností* D_G takto:

$$D_G = ||d_{ij}||, \text{ kde } d_{ij} = \begin{cases} c_{ij} & \text{pokud mezi vrcholy } v_i, v_j \ (i \neq j) \text{ je cesta;} \\ \infty & \text{pokud mezi vrcholy } v_i, v_j \ (i \neq j) \text{ není cesta;} \\ 0 & \text{pokud } i = j. \end{cases}$$

a v případě hranově ohodnoceného orientovaného grafu takto:

$$d_{ij} = \begin{cases} c_{ij} & \text{pokud existuje orientovaná cesta mezi vrcholy } v_i \text{ a } v_j \text{ tak,} \\ & \text{že } v_i \text{ je její počáteční a } v_j \text{ koncový vrchol, kde } i \neq j; \\ \infty & \text{pokud neexistuje orientovaná cesta mezi vrcholy } v_i \text{ a } v_j \text{ tak,} \\ & \text{že } v_i \text{ je její počáteční a } v_j \text{ koncový vrchol, kde } i \neq j; \\ 0 & \text{pokud } i = j. \end{cases}$$



Obr. 4.2

Příklad 4.8. Matice vzdáleností pro orientovaný graf na obrázku 4.2:

$$D_G = \begin{pmatrix} 0 & 1 & 8 & \infty & 4 \\ 6 & 0 & 7 & \infty & 3 \\ 2 & 3 & 0 & \infty & 6 \\ 5 & 6 & 6 & 0 & 2 \\ 3 & 4 & 4 & \infty & 0 \end{pmatrix}.$$

K nalezení matice vzdáleností můžeme využít 2 algoritmů, které si zde uvedeme.

4.7 Upravené násobení matic

Upravené násobení matic, pomocí kterého jsme schopni získat matici vzdáleností, vychází z již zmiňované matice délek, kterou označíme U_1. Tuto matici upravíme pomocí příslušného algoritmu. Tím získáme další matici U_2, kterou zase upravíme pomocí stejného algoritmu. Postupně takto dostaneme posloupnost matic $U_1, \ldots, U_{n-1}$, kde n je číslo, které nám udává počet hran přes kolik „můžeme jít“. Jelikož víme, že žádná cesta na n vrcholech nemůže obsahovat více než $n-1$ hran, je $U_{n-1} = U$ výsledná matice vzdáleností.

Pomocná struktura:

1. $V \ldots$ množina všech vrcholů grafu.
2. $E \ldots$ množina všech hran grafu.
3. Každá hrana $e_1, \ldots, e_n \in E$ je ve tvaru $e_k = (v_i, v_j, c_k)$, kdy v neorientovaném grafu jsou v_i a v_j krajní vrcholy hrany e_k a c_k délka dané hrany k. V orientovaném grafu je v_i počáteční a v_j koncový vrchol hrany e_k.
4. $H_G \ldots$ matice délek grafu.
5. $U_1, \ldots, U_n \ldots$ posloupnost matic, kdy každý prvek $u_p(i,j)$ matice U_p, který má vlastnost takovou, že $u_p(i,j)$ je délka nejkratší cesty z vrcholu v_i do vrcholu v_j, která má nejvýše p hran.
6. $n \ldots$ proměnná, která nám udává počet vrcholů daného grafu.

Algoritmus:

1. Vycházíme z matice délek grafu H_G, kterou si označíme jako matici U_1 a počet vrcholů si označíme n.
2. Počítáme matice U_p.

(a) Známe-li matici U_{p-1}, pak vypočítáme matici U_p takto:

$$u_p(i,j) = min\{u_{p-1}(i,k) + d_G(k,j)\};$$

(b) Matice U_p počítáme dokud neplatí, že $U_p = U_{n-1} = U$, což je výsledná matice vzdáleností.

Příklad 4.9. Vypočítejte pomocí upraveného násobení matic matici vzdáleností pro graf na obrázku 4.2.

Řešení:

1. Matici délek H_G nastavíme jako U_1:

$$U_1 = H_G = \begin{pmatrix} 0 & 1 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & 3 \\ 2 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 0 & 2 \\ 3 & \infty & 4 & \infty & 0 \end{pmatrix}.$$

2. Provedeme výpočty pro dané prvky podle výše uvedeného návodu:

$$\begin{aligned} u_1(1,2) + h_G(2,5) &= 1 + 3 = 4 = u_2(1,5); \\ u_1(2,5) + h_G(5,1) &= 3 + 3 = 6 = u_2(2,1); \\ u_1(2,5) + h_G(5,3) &= 3 + 4 = 7 = u_2(2,3); \\ u_1(3,1) + h_G(1,2) &= 2 + 1 = 3 = u_2(3,2); \\ u_1(4,5) + h_G(5,1) &= 2 + 3 = 5 = u_2(4,1); \\ u_1(4,5) + h_G(5,3) &= 2 + 4 = 6 = u_2(4,3); \\ u_1(5,1) + h_G(1,2) &= 3 + 1 = 4 = u_2(5,2). \end{aligned}$$

   Matice U_2 tedy je:

$$U_2 = \begin{pmatrix} 0 & 1 & \infty & \infty & 4 \\ 6 & 0 & 7 & \infty & 3 \\ 2 & 3 & 0 & \infty & \infty \\ 5 & \infty & 6 & 0 & 2 \\ 3 & 4 & 4 & \infty & 0 \end{pmatrix}.$$

3. Provedeme výpočty pro získanou matici U_2 (vypsány jsou pouze případy, kdy proběhla nějaká změna):

$$\begin{aligned} u_2(1,2) + h_G(2,3) &= 1 + 7 = 8 = u_3(1,3); \\ u_2(3,1) + h_G(1,5) &= 2 + 4 = 6 = u_3(3,5); \\ u_2(4,1) + h_G(1,2) &= 5 + 1 = 6 = u_3(4,2). \end{aligned}$$

Matice U_3 tedy je:

$$U_3 = \begin{pmatrix} 0 & 1 & 8 & \infty & 4 \\ 6 & 0 & 7 & \infty & 3 \\ 2 & 3 & 0 & \infty & 6 \\ 5 & 6 & 6 & 0 & 2 \\ 3 & 4 & 4 & \infty & 0 \end{pmatrix}.$$

4. Výpočty provedené na matici U_3, díky kterým získáme matici U_4, nám už výslednou matici nezmění, zjistíme tak, že $U_3 = U_4$, což se rovná výsledné matici vzdálenosti U:

$$U = U_4 = U_3 = \begin{pmatrix} 0 & 1 & 8 & \infty & 4 \\ 6 & 0 & 7 & \infty & 3 \\ 2 & 3 & 0 & \infty & 6 \\ 5 & 6 & 6 & 0 & 2 \\ 3 & 4 & 4 & \infty & 0 \end{pmatrix}.$$

Tento způsob výpočtu matice vzdálenosti je sice lehký na pochopení, ale zbytečně pracný. Výpočet lze zrychlit tím, že místo celé posloupnosti matic budeme počítat jen ty, kde p je mocninou dvojky dokud $p \geq n-1$. Rychlejší a jednodušší způsob výpočtu matice vzdálenosti je Floydův algoritmus.

4.8 Floydův algoritmus

Floydův algoritmus převádí, stejně jako předchozí algoritmus, matici délek na matici vzdálenosti. Výsledkem tohoto algoritmu tedy je matice, ve které jsou uvedeny délky nejkratších cest z libovolného vrcholu do všech ostatních vrcholů. Stejně jako u upraveného násobení matic i u tohoto algoritmu postupně počítáme posloupnost matic $U_0, \ldots, U_n$, kde n je počet vrcholů.

Pomocná struktura:

1. $V \ldots$ množina všech vrcholů grafu.
2. $E \ldots$ množina všech hran grafu.
3. Každá hrana $e_1, \ldots, e_n \in E$ je ve tvaru $e_k = (v_i, v_j, c_k)$, kdy v neorientovaném grafu jsou v_i a v_j krajní vrcholy hrany e_k a c_k délka dané hrany k. V orientovaném grafu je v_i počáteční a v_j koncový vrchol hrany e_k.
4. $H_G \ldots$ matice délek grafu.

5. $U_0, \ldots, U_n \ldots$ posloupnost matic, kdy každý prvek $u_p(i,j)$ matice U_p, který má vlastnost takovou, že $u_p(i,j)$ je délka nejkratší cesty z vrcholu v_i do vrcholu v_j, která kromě vrcholů v_i a v_j dále prochází jen přes vrcholy z množiny $\{1, 2, \ldots, k\}$.

6. $n \ldots$ proměnná, která nám udává počet vrcholů daného grafu.

Algoritmus:

1. Vycházíme z matice délek grafu H_G, kterou si označíme jako matici U_0 a počet vrcholů si označíme n.

2. Vypočítáme matici U_k: Známe-li matici U_{k-1}, pak vypočítáme matici U_k takto:

$$u_k(i,j) = min\{u_{k-1}(i,k) + u_{k-1}(k,j); u_{k-1}(i,j)\}$$

   Vybíráme a počítáme jen prvky z k-tého sloupce a k-tého řádku.

3. Krok 2 opakujeme, dokud neprojdeme všechny řádky s sloupce. Poslední dosažená matice je námi hledaná matice vzdáleností.

Poznámka 4.9. Prvky $u_k(i,i)$ nemusíme počítat, jelikož ty se v hranově nezáporně ohodnoceném grafu nemění. Stejně tak nemusíme počítat prvky $u_k(i,k)$ a $u_k(k,j)$, pokud alespoň jeden z nich je nekonečno.

Poznámka 4.10. Při výpočtu je výhodné si každou matici napsat zvlášť. Na počítači je to ale zbytečné, protože výpočet lze provést v jediné matici, kterou jen měníme. Při vytváření matice U_k se k-tý řádek a k-tý sloupec nemění.

Příklad 4.10. Vypočítejte pomocí Floydova algoritmu matici vzdáleností pro graf na obrázku 4.2.

Řešení:

Poznámka: V maticích jsou tučně zvýrazněny sloupce a řádky se kterými vždy v následujícím kroku pracujeme.

1. Matici délek H_G nastavíme jako matici U_0:

$$U_0 = H_G = \begin{pmatrix} \mathbf{0} & \mathbf{1} & \boldsymbol{\infty} & \boldsymbol{\infty} & \boldsymbol{\infty} \\ \boldsymbol{\infty} & 0 & \infty & \infty & 3 \\ \mathbf{2} & \infty & 0 & \infty & \infty \\ \boldsymbol{\infty} & \infty & \infty & 0 & 2 \\ \mathbf{3} & \infty & 4 & \infty & 0 \end{pmatrix}.$$

2. Vypočítáme následující matici U_1 podle návodu v kroku 2. Procházíme jednotlivé prvky v prvním řádku a prvním sloupci a podle výše uvedeného kriteria při nalezení nového minima změníme na daném místě prvek v matici U_0. Vybereme tedy první prvek z 1. řádku, který má smysl počítat (viz. poznámka 4.8) a provedeme:

$$u_1(3,2) = min\{u_0(3,1) + u_0(1,2); u_0(3,2)\};$$

tedy

$$u_1(3,2) = min\{2+1; \infty\} \Rightarrow u_1(3,2) = 3;$$

pokračujeme dalším prvkem:

$$u_1(5,2) = min\{u_0(5,1) + u_0(1,2); u_0(5,2)\};$$

tedy

$$u_1(5,2) = min\{3+1; \infty\} \Rightarrow u_1(5,2) = 4.$$

To jsou všechny výpočty pro matici U_0, které má smysl počítat. Matice U_1 tedy je:

$$U_1 = \begin{pmatrix} 0 & \mathbf{1} & \infty & \infty & \infty \\ \boldsymbol{\infty} & \mathbf{0} & \boldsymbol{\infty} & \boldsymbol{\infty} & \mathbf{3} \\ 2 & \mathbf{3} & 0 & \infty & \infty \\ \infty & \boldsymbol{\infty} & \infty & 0 & 2 \\ 3 & \mathbf{4} & 4 & \infty & 0 \end{pmatrix}.$$

3. Pokračujeme výpočtem matice U_2 z matice U_1. Procházíme jednotlivé prvky ve druhém řádku a druhém sloupci. Tedy:

$$u_2(1,5) = min\{u_1(1,2) + u_1(2,5); u_1(1,5)\};$$
$$u_2(3,5) = min\{u_1(3,2) + u_1(2,5); u_1(3,5)\}.$$

Odtud

$$u_2(1,5) = min\{1+3; \infty\} \Rightarrow u_2(1,5) = 4;$$
$$u_2(3,5) = min\{3+3; \infty\} \Rightarrow u_2(3,5) = 6.$$

Matice U_2 tedy je:

$$U_2 = \begin{pmatrix} 0 & 1 & \boldsymbol{\infty} & \infty & 4 \\ \infty & 0 & \boldsymbol{\infty} & \infty & 3 \\ \mathbf{2} & \mathbf{3} & \mathbf{0} & \boldsymbol{\infty} & \mathbf{6} \\ \infty & \infty & \boldsymbol{\infty} & 0 & 2 \\ 3 & 4 & \mathbf{4} & \infty & 0 \end{pmatrix}.$$

4. Pokračujeme výpočtem matice U_3 z matice U_2. Procházíme jednotlivé prvky ve třetím řádku a třetím sloupci. Při procházení jednotlivých prvků nám v matici nedošlo k žádné změně, proto matice U_3 je stejná jako matice U_2 tedy:

$$U_3 = U_2 = \begin{pmatrix} 0 & 1 & \infty & \boldsymbol{\infty} & 4 \\ \infty & 0 & \infty & \boldsymbol{\infty} & 3 \\ 2 & 3 & 0 & \boldsymbol{\infty} & 6 \\ \boldsymbol{\infty} & \boldsymbol{\infty} & \boldsymbol{\infty} & \mathbf{0} & \mathbf{2} \\ 3 & 4 & 4 & \boldsymbol{\infty} & 0 \end{pmatrix}.$$

5. Pokračujeme výpočtem matice U_4 z matice U_3. Procházíme jednotlivé prvky ve čtvrtém řádku a čtvrtém sloupci. Při procházení jednotlivých prvků nám znovu v matici nedošlo k žádné změně, proto matice U_4 je stejná jako matice U_3 a U_2 tedy:

$$U_4 = U_3 = U_2 = \begin{pmatrix} 0 & 1 & \infty & \infty & \mathbf{4} \\ \infty & 0 & \infty & \infty & \mathbf{3} \\ 2 & 3 & 0 & \infty & \mathbf{6} \\ \infty & \infty & \infty & 0 & \mathbf{2} \\ \mathbf{3} & \mathbf{4} & \mathbf{4} & \boldsymbol{\infty} & \mathbf{0} \end{pmatrix}.$$

6. Pokračujeme výpočtem poslední matice U_5. Procházíme tedy prvky v pátém řádku a pátém sloupci:

$$\begin{aligned} u_5(1,3) &= min\{u_4(1,5) + u_4(5,3); u_4(1,3)\}; \\ u_5(2,1) &= min\{u_4(2,5) + u_4(5,1); u_4(2,1)\}; \\ u_5(2,3) &= min\{u_4(2,5) + u_4(5,3); u_4(2,3)\}; \\ u_5(4,1) &= min\{u_4(4,5) + u_4(5,1); u_4(4,1)\}; \\ u_5(4,2) &= min\{u_4(4,5) + u_4(5,2); u_4(4,2)\}; \\ u_5(4,3) &= min\{u_4(4,5) + u_4(5,3); u_4(4,3)\}; \end{aligned}$$

tedy

$$
\begin{aligned}
u_5(1,3) &= min\{4+4;\infty\} \Rightarrow u_5(1,3) = 8;\\
u_5(2,1) &= min\{3+3;\infty\} \Rightarrow u_5(2,1) = 6;\\
u_5(2,3) &= min\{3+4;\infty\} \Rightarrow u_5(2,3) = 7;\\
u_5(4,1) &= min\{2+3;\infty\} \Rightarrow u_5(4,1) = 5;\\
u_5(4,2) &= min\{2+4;\infty\} \Rightarrow u_5(4,2) = 6;\\
u_5(4,3) &= min\{2+4;\infty\} \Rightarrow u_5(4,3) = 6.
\end{aligned}
$$

Matice U_5 a tedy výsledná matice vzdáleností U je:

$$
U = U_5 = \begin{pmatrix} 0 & 1 & 8 & \infty & 4 \\ 6 & 0 & 7 & \infty & 3 \\ 2 & 3 & 0 & \infty & 6 \\ 5 & 6 & 6 & 0 & 2 \\ 3 & 4 & 4 & \infty & 0 \end{pmatrix}.
$$

4.9 Hledání nejdelší cesty v grafech

Všechny zde dříve uvedené algoritmy hledaly v grafech nejkratší cestu. Někdy ale potřebujeme v grafu najít naopak cestu nejdelší. Kvůli tomuto nemusíme vymýšlet žádné nové algoritmy, stačí použít již zde uvedené, jen s mírnými modifikacemi. Nejjednodušší modifikace jsou:

1. Můžeme změnit znaménka u všech délek hran na opačná a použít již uvedené algoritmy pro hledání nejkratších cest. Díky této zaměně budou nyní nejdelší cesty nejkratšími a opačně.

2. Můžeme obrátit nerovnost v trojúhelníkové nerovnosti $c_j > c_i + c_{ij}$ na $c_j < c_i + c_{ij}$. Při této změně nesmíme ale zapomenout změnit také hodnotu ∞ na $-\infty$ při inicializaci vrcholů.

Poznámka 4.11. Zde uvedené algoritmy byly upraveny kvůli odlišné interpretaci. Původní algoritmy, ze kterých tyto algoritmy vychází, je možné nalézt ve skriptech [1] v kapitole 4 na straně 31.

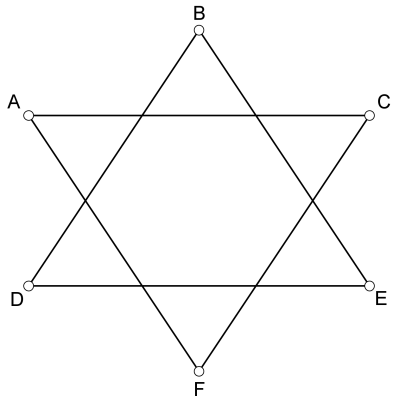
5 Souvislost grafu a hledání komponent souvislosti

V předchozích kapitolách jsme si uvedli algoritmy pro prohledávání grafů a zjišťování dostupnosti jednotlivých vrcholů a také algoritmy pro hledání nejkratších cest v hranově neohodnocených i ohodnocených grafech. Nyní si ukážeme, jak se dají tyto algoritmy využít pro zjištění souvislosti grafů a také si ukážeme jeden nový algoritmus, který nám pomůže určit jednotlivé komponenty souvislosti. Co znamená, že je graf souvislý, jsme si již řekli. Nyní si potřebujeme říci, co to jsou jeho komponenty souvislosti.

Definice 5.1. *Komponentou souvislosti* grafu $G = (V, E)$ nazveme každý maximální souvislý podgraf, který není obsažen ve větším souvislém podgrafu. Nebo-li je to maximální souvislá část grafu G.

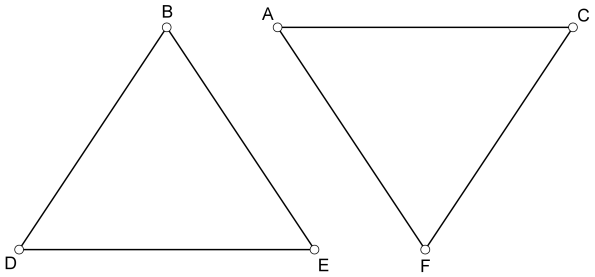
Definice 5.2. Komponentu souvislosti grafu $G = (V, E)$ nazveme *silná komponenta souvislosti*, jestliže je to takový podgraf grafy G, ve kterém pro každé jeho dva vrcholy $v_i, v_j \in V$ existuje cesta z v_i do v_j a zároveň z v_j do v_i. V opačném případě tuto komponentu nazveme *slabá komponenta souvislosti.*

Příklad 5.1. Z kolika komponent souvislosti se skládá graf na obrázku 5.1?



Obr. 5.1

Řešení:



Při bližším zkoumání zjistíme, že graf není souvislý, ale skládá se ze dvou souvislých silných komponent.

5.1 Zjištění souvislosti grafu

Na zjištění toho jestli je graf souvislý nebo ne, můžeme využít následujícího:

1. Algoritmy pro prohledávání grafů do hloubky a do šířky. Pokud porovnáme množinu zpracovaných vrcholů Z s množinou všech vrcholů grafu V a zjistíme, že jsou tyto množiny stejné, pak víme, že daný neorientovaný graf je souvislý, případně orientovaný graf je slabě souvislý.

2. Matice vzdáleností, kterou získáme např. Floydovým algoritmem nebo upraveným násobením matic z matice délek. Pokud výsledná matice vzdáleností nebude obsahovat žádné ∞, pak víme, že neorientovaný graf je souvislý, případně orientovaný graf silně souvislý.

Definice 5.3. *Součtová matice* T_G nám udává počet všech sledů grafu $G = (V, E)$ délky n, kde

$$n = min\{|V(G) - 1|; |E(G)|\};$$

kterými se dostaneme z vrcholu v_i do vrcholu v_j. Součtovou matici T_G můžeme získat ze součtu mocnin matice sousednosti S_G, nebo-li:

$$T_G = S_G + S_G^2 + \ldots + S_G^n$$

5.2 Určení komponent souvislosti

Pomocí tohoto algoritmu budeme moci určit jednotlivé komponenty souvislosti a podle toho určit souvislost grafu. Tento algoritmus vychází z matice sousednosti, případně ho lze lehce upravit pro součtovou matici.

Pomocná struktura:

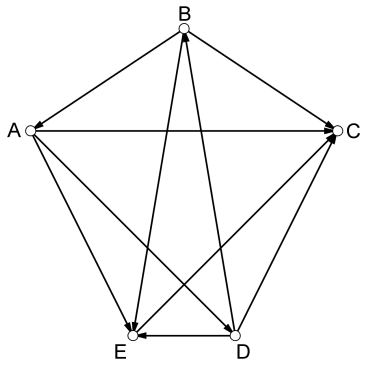
1. $V \ldots$ množina vrcholů grafu.
2. $E \ldots$ množina hran grafu.
3. $K_v \ldots$ pomocné pole pro vrcholy, které jsou v komponentě souvislosti s vrcholem v.
4. $K_{v_i} \ldots$ hodnota vrcholu v_i v poli K.
5. $n \ldots$ proměnná udávající počet vrcholů v grafu.

Algoritmus:

1. Základní nastavení:
   (a) Vyprázdníme pole K_v a nastavíme všechny hodnoty na 0.
   (b) Proměnné n přiřadíme počet vrcholů v grafu.
   (c) Upravíme matici sousednosti na součtovou matici.
2. Zjistíme, jestli pro alespoň jeden vrchol v_i v poli K_v platí: $K_{v_i} = 0$.
   (a) Pokud ano, pak vybereme libovolný vrchol, pro který tato rovnost platí a provedeme:
   $$K_{v_i} = i;$$
   (b) Pokud neexistuje vrchol, pro který tato rovnost platí, pokračujeme krokem 4.
3. Zkoušíme, jestli se z jeho dostupných sousedních vrcholů v_j, pro které platí:
   $$K_{v_j} = 0;$$
   dostaneme do vrcholu v_i.
   (a) Pokud ano, pak nastavíme vrchol $K_{v_j} = i$.
   (b) Pokud ne, nic neměníme a pokračujeme znovu krokem 2.
   (c) Pokud neexistuje žádný vrchol, který bychom mohli vyzkoušet, pokračujeme krokem 2.
4. Algoritmus končí. Vrcholy v poli K_v, které jsou v tomto poli reprezentovány stejným vrcholem, jsou spolu v jedné komponentě souvislosti.

Poznámka 5.1. Pro hledání dostupných vrcholů můžeme využít jakéhokoliv zde již dříve uvedeného algoritmu.

Příklad 5.2. Určete komponenty souvislosti orientovaného grafu na obrázku 5.2 pomocí algoritmu pro hledání komponent souvislosti.



Obr. 5.2

Řešení:

1. Vyprázdníme pole K_v a všechny hodnoty nastavíme na 0, tedy:

$$K_v = (0, 0, 0, 0, 0)$$

Nastavíme počet vrcholů:

$$n = 5$$

A upravíme matici sousednosti S na součtovou matici T_G podle vzorce $T_G = S_G + S_G^2 + S_G^3 + \ldots + S_G^n$, tedy:

$$S_G = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Vynásobíme-li matici S sebe samou, dostaneme matici S^2, tedy:

$$S_G^2 = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Matici S^3 dostaneme vynásobením matic S^2 a S a matici S^4 získáme např. vynásobením matice S^2 sebe samou. Tím dostaneme matice:

$$S_G^3 = \begin{pmatrix} 1 & 0 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad S_G^4 = \begin{pmatrix} 0 & 0 & 2 & 1 & 1 \\ 1 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Z těchto matic již pomocí zmiňovaného vzorce snadno získáme výslednou součtovou matici:

$$T_G = \begin{pmatrix} 1 & 1 & 6 & 2 & 4 \\ 2 & 1 & 6 & 1 & 4 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 5 & 1 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

2. Ověříme, zda alespoň pro jeden vrchol v_i platí: $K_{v_i} = 0$. Jelikož pole $K_v = (0, 0, 0, 0, 0)$, pak požadavek splňují všechny vrcholy. Takže si vybereme například vrchol A a provedeme:

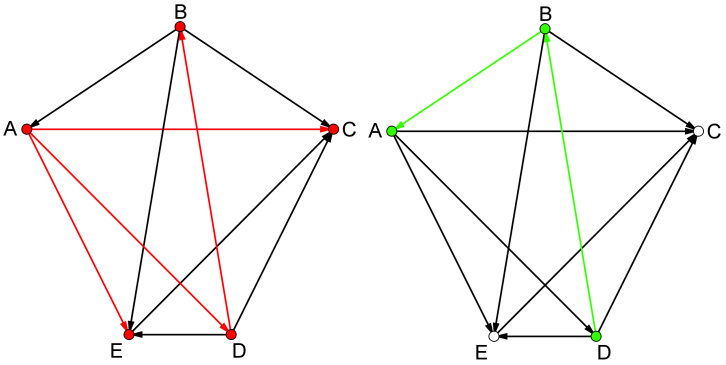
$$K_{v_A} = A$$

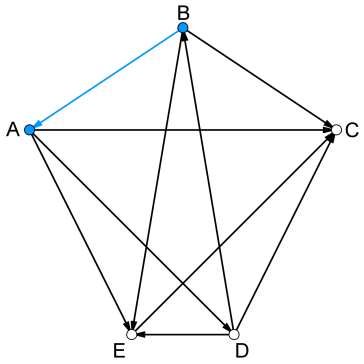
3. Nyní ověříme dostupnost vrcholu A pro jeho dostupné sousedy, pro které platí:

$$K_{v_j} = 0$$

Z vrcholu A jsou všechny ostatní vrcholy dosažitelné (červená), ale vrchol A je dosažitelný pouze pro vrcholy B (modrá) a D (zelená). Pro tyto vrcholy provedeme:

$$K_{v_B} = A; K_{v_D} = A$$





Nyní pokračujeme krokem 2.

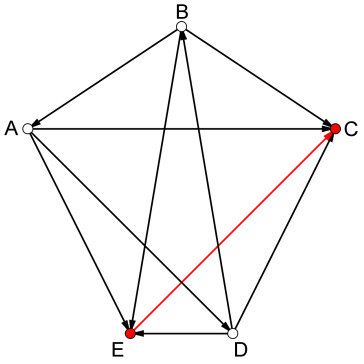
4. Pole $K_v = (A, A, 0, A, 0)$, požadavek $K_{v_i} = 0$ splňují vrcholy C a E. Vybereme si například vrchol E a provedeme:

$$K_{v_E} = E$$

5. Nyní ověříme dostupnost vrcholu E pro jeho dostupné sousedy (červená), pro které platí:

$$K_{v_j} = 0$$

z vrcholu E je dosažitelný pouze vrchol C, ale opačně toto neplatí. Proto pokračujeme krokem 2.

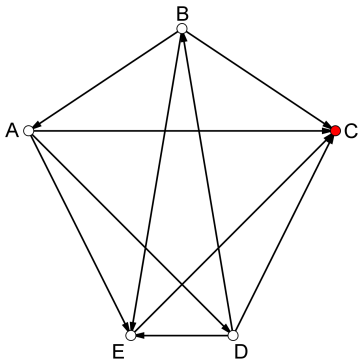


6. Pole $K_v = (A, A, 0, A, E)$, požadavek $K_{v_i} = 0$ splňuje jen vrchol C. Proto si ho vybereme a provedeme:

$$K_{v_C} = C$$

7. Nyní ověříme dostupnost vrcholu C pro jeho dostupné sousedy, pro které platí:

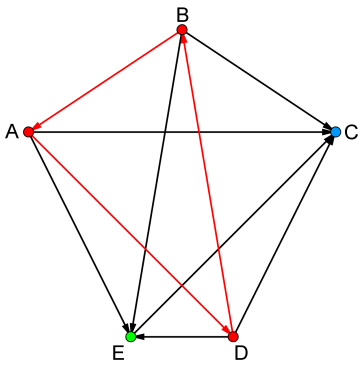
$$K_{v_j} = 0$$



Neexistuje ale žádný vrchol, který by tento požadavek splňoval, proto pokračujeme krokem 2.

8. Pole $K_v = (A, A, C, A, E)$, požadavek $K_{v_i} = 0$ nesplňuje žádný vrchol. Pokračujeme tedy krokem 4.

9. Konec algoritmu. Komponenty souvislosti tohoto orientovaného grafu tedy jsou:



$$K_1 = \{A, B, D\}; K_2 = \{C\}; K_3 = \{E\};$$

Tyto komponenty souvislosti jsou silné komponenty souvislosti.

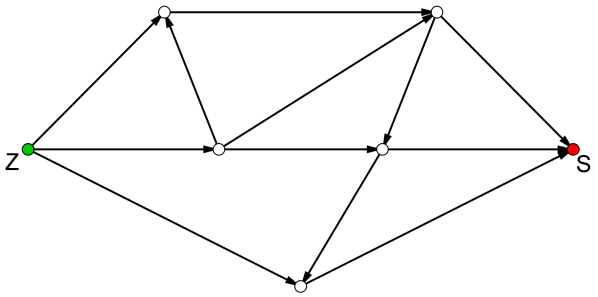
6 Toky v sítích

V poslední kapitole se budeme zabývat toky v sítích a algoritmy, které nám pomohou najít maximální tok v této síti. Síť a toky v síti si můžete představit například jako potrubí, kterým protéká voda a nebo třeba jako síť silnic, kterou jezdí automobily. Problém, který budeme chtít vyřešit je, jak přenést co nejvíce toho něčeho (např. vody v potrubí), z počátečního místa (zdroje) do cílového místa (stoku), za určitých omezení, kterými jsou kapacity jednotlivých cest.

Definice 6.1. *Sítí* nazveme čtveřici $S = (G, z, s, w)$, kde G je orientovaný graf, vrcholy $z, s \in V(G)$ jsou *zdroj* a *stok (spotřebič)* grafu G a zobrazení $w : E(G) \to R^+$ je kladné ohodnocení hran nazývané *kapacita*.

K reprezentaci těchto sítí budeme tedy výhradně používat orientované grafy s kladným ohodnocením hran.

Příklad 6.1. Příklad sítě se zdrojem a stokem:



Definice 6.2. *Tokem* v síti $S = (G, z, s, w)$ nazveme funkci $f : E(G) \to R_0^+$, které splňuje:

1. Pro každou hranu $e \in E(G)$ platí:

$$0 \leq f(e) \leq w(e)$$

   nebo-li tok v síti je vždy kladný a není větší než kapacita příslušné hrany.

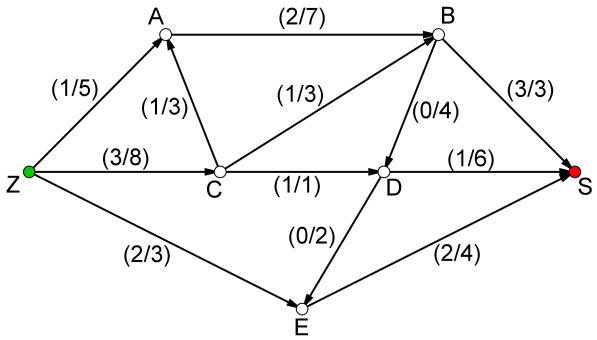
2. Pro každý vrchol $v \in V(G)$, který není zdroj ani stok (tzn. $v \neq z, s$), platí:

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

Tedy, že odtok každého vrcholu musí být roven přítoku do tohoto vrcholu. Tento vztah se nazývá Kirchhoffův zákon.

Poznámka 6.1. Tok F a kapacitu hran C budeme u každé hrany jednoduše zapisovat jako F/C.

Příklad 6.2. Příklad sítě se zdrojem, stokem, kapacitou hran a tokem:



Definice 6.3. *Velikost toku* značíme $|f|$ a platí:

$$|f| = \sum_{e \in E^+} f(z) - \sum_{e \in E^-} f(z);$$

nebo-li je to součet toho co nám do stoku přitéká, bez toho co nám ze stoku odtéká.

Definice 6.4. *Maximálním tokem* nazveme

$$max|f|;$$

tedy největší tok, který jsme schopni poslat ze zdroje do stoku.

Definice 6.5. *Řezem grafu* nazveme podmnožinu hran $E' \subseteq E$ mezi zdrojem z a stokem s tak, že v podgrafu $G' = (V, E \setminus E')$ neexistuje žádná orientovaná cesta ze zdroje z do stoku s.

Definice 6.6. *Velikostí řezu* nazveme součet kapacit hran řezu E', nebo-li:

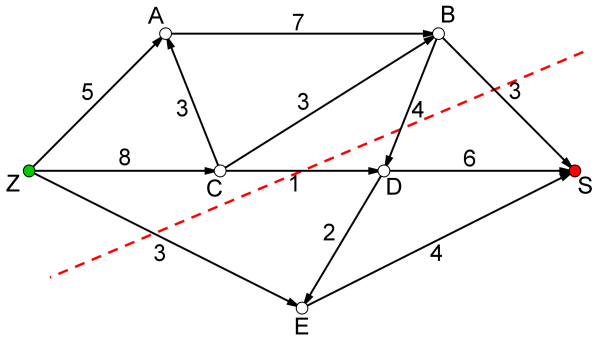
$$|E'| = \sum_{e \in E'} w(e)$$

Věta 6.1. *Maximální velikost toku ze zdroje z do stoku s v síti S je rovna minimální velikosti řezu mezi zdrojem a stokem.*

Důkaz této věty můžeme nalézt například ve skriptech [4] na straně 95.

Tato věta je také známa jako *Ford-Fulkersonova věta o maximálním toku a minimálním řezu.*

Příklad 6.3. Příklad minimálního řezu grafu:



Definice 6.7. *Nenasycenou cestou* nazveme neorientovanou cestu v grafu $G = (V, E)$ ze zdroje $z \in V(G)$ do stoku $s \in V(G)$, tzn. posloupnost hran $e_1, \ldots, e_n \in E(G)$, kde:

$$f(e_i) < w(e_i), \forall e_i \in E(G) \text{ ve směru ze zdroje } z \text{ do stoku } s.$$

Definice 6.8. *Rezervou* nazveme hodnotu:

$$r(e_i) = w(e_i) - f(e_i), \forall e_i \in E(G) \text{ ve směru ze zdroje } z \text{ do stoku } s.$$

Definice 6.9. *Zlepšující cestou* nazveme každou orientovanou cestu ze zdroje z do stoku s, jejíž každá hrana má nenulovou rezervu.

6.1 Ford-Fulkersonův algoritmus

Tento algoritmus se zakládá na velmi jednoduché myšlence. Pokud existuje nějaká zlepšující cesta ze zdroje z do stoku s, tedy, že každá hrana na této cestě má ještě rezervu, pak na všech hranách této cesty navýšíme hodnotu toku o maximální hodnotu o jakou lze tok zvýšit. Tedy o nejmenší rezervu některé hrany na této cestě. Celý postup opakujeme, dokud již nebude existovat žádná zlepšující cesta. Algoritmus je určen pouze pro orientované grafy (sítě).

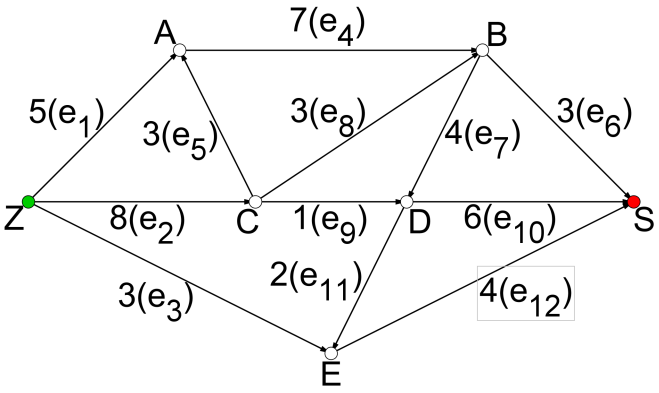
Pomocná struktura:

1. V ... množina vrcholů grafu G.
2. E ... množina hran grafu G.
3. Každá hrana $e_1, \ldots, e_n \in E$ je orientovaná a ve tvaru $e_k = (v_i, v_j, w_k)$, kde v_i je počáteční vrchol, v_j je koncový vrchol hrany e_k a w_k je kapacita hrany mezi vrcholy v_i a v_j.

4. $F_e \ldots$ pomocné pole pro uchování hodnoty toku jednotlivých hran.
5. $R_e \ldots$ pomocné pole pro uchování hodnoty rezervy jednotlivých hran.
6. $r \ldots$ proměnná pro uchování hodnoty minimální rezervy.

Algoritmus:

1. Základní nastavení:
   (a) Vytvoříme nulový tok tak, že všem hranám přiřadíme tok:
   $$f(e_i) = 0;$$
   (b) Proměnnou r nastavíme na hodnotu 0.
2. Nalezení zlepšující cesty:
   (a) Pro každou hranu e_i označíme r_i její rezervu. Platí, že:
   $$r_i > 0;$$
   a nastavíme v pomocném poli R_e na:
   $$R_{e_i} = w(e_i) - f(e_i);$$
   kde $f(e_i)$ je aktuální tok hranou e_i.
   (b) Hledáme nenasycenou cestu ze zdroje z do stoku s. K tomuto účelu můžeme využít například již nám známé algoritmy pro prohledávání grafu.
      i. Pokud žádná zlepšující cesta nebyla nalezena, algoritmus končí, nalezli jsme maximální tok v síti.
      ii. Pokud jsme nalezli zlepšující cestu, pak vybereme nejmenší rezervu r na naší nalezené nenasycené cestě, tedy:
      $$r = min\{r_i\};$$
3. Nasycení toku
   (a) Tok f na nalezené cestě nasytíme o minimální rezervu, tedy tok každé hrany na naší nenasycené cestě zvýšíme o minimální rezervu a také rezervu každé hrany snížíme o minimální rezervu. Tedy:
   $$F_{e_i} = f(e_i) + r$$
   (b) Pokračujeme krokem 2.



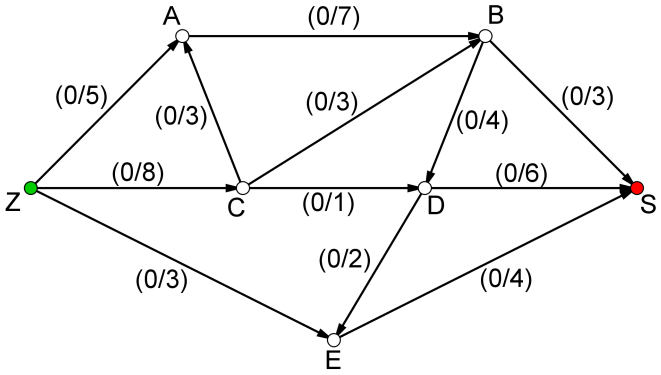
Obr. 6.1

Příklad 6.4. Najděte maximální tok v síti na obrázku 6.1:

Řešení:

1. Vyprázdníme proměnnou r a vytvoříme nulový tok, tedy:

$$r = 0; \forall e_i : f(e_i) = 0;$$

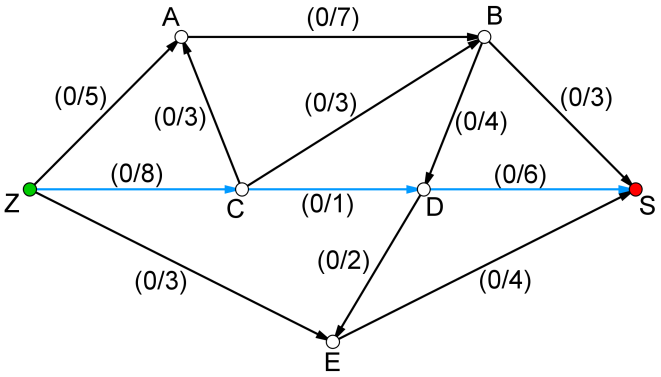


2. Pro každou hranu označíme její rezervu a nastavíme pomocné pole R_e:

$$R_e = (5, 8, 3, 7, 3, 3, 4, 3, 1, 6, 2, 4)$$

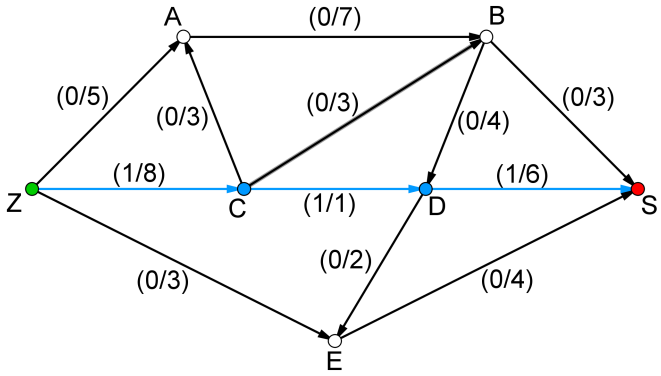
Hledáme nenasycenou cestu ze zdroje z do stoku s, nalezli jsme například nenasycenou cestu jdoucí hranami: e_2, e_9, e_{10} a vybereme nejmenší rezervu r na naší cestě, tedy:

$$r = min\{e_2, e_9, e_{10}\}, \text{ tzn.:} min\{8, 1, 6\} = 1$$



3. Nasytíme tok na nalezené cestě o minimální rezervu:

$$F_e = (0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0)$$



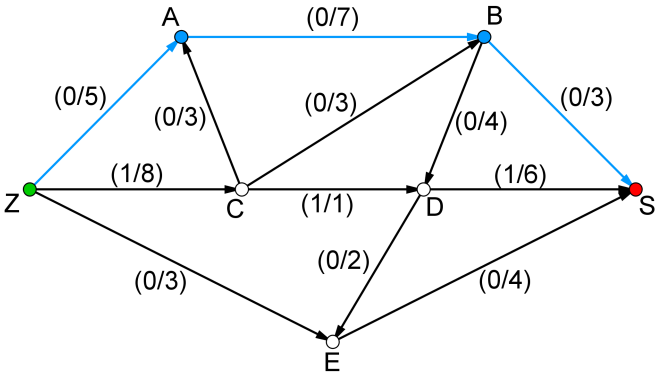
a pokračujeme znovu krokem 2.

4. Označíme a nastavíme rezervy jednotlivých hran:

$$R_e = (5, 7, 3, 7, 3, 3, 4, 3, 0, 5, 2, 4)$$

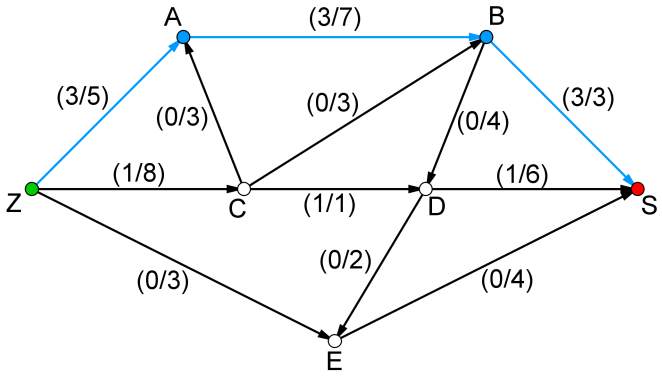
Hledáme nenasycenou cestu ze zdroje z do stoku s, nalezli jsme například nenasycenou cestu jdoucí hranami: e_1, e_4, e_6 a vybereme nejmenší rezervu r na naší cestě, tedy:

$$r = min\{e_1, e_4, e_6\}, \text{ tzn.:} min\{5, 7, 3\} = 3$$



5. Nasytíme tok na nalezené cestě o minimální rezervu:

$$F_e = (3, 1, 0, 3, 0, 3, 0, 0, 1, 1, 0, 0)$$



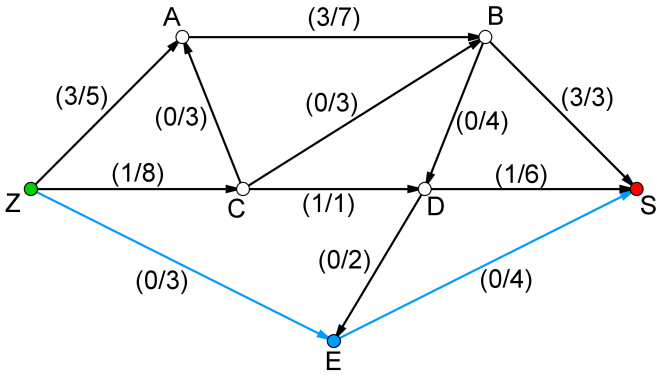
   a pokračujeme znovu krokem 2.

6. Označíme a nastavíme rezervy jednotlivých hran:

$$R_e = (2, 7, 3, 4, 3, 0, 4, 3, 0, 5, 2, 4)$$

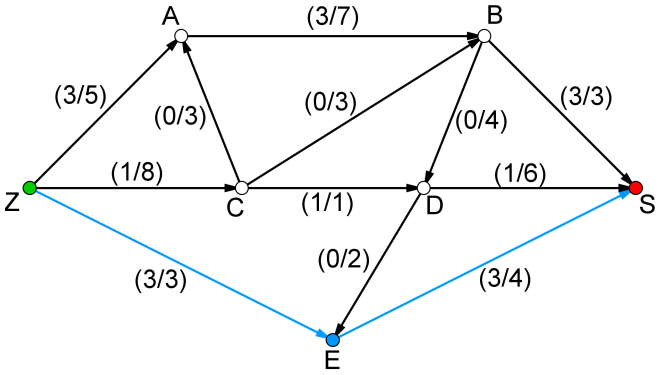
Hledáme nenasycenou cestu ze zdroje z do stoku s, nalezli jsme například nenasycenou cestu jdoucí hranami: e_3, e_{12} a vybereme nejmenší rezervu r na naší cestě, tedy:

$$r = min\{e_3, e_{12}\}, \text{ tzn.:} min\{3, 4\} = 3$$



7. Nasytíme tok na nalezené cestě o minimální rezervu:

$$F_e = (3, 1, 3, 3, 0, 3, 0, 0, 1, 1, 0, 3)$$



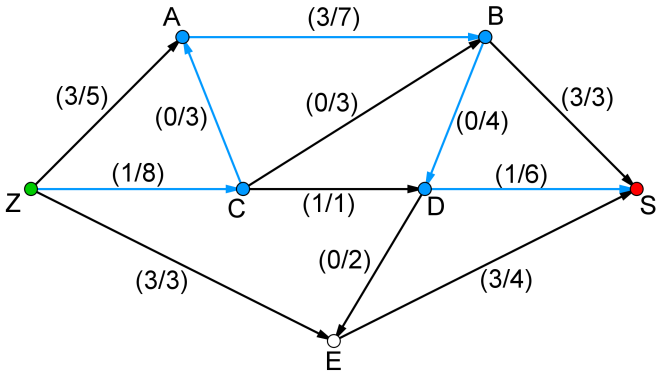
a pokračujeme znovu krokem 2.

8. Označíme a nastavíme rezervy jednotlivých hran:

$$R_e = (2, 7, 0, 4, 3, 0, 4, 3, 0, 5, 2, 1)$$

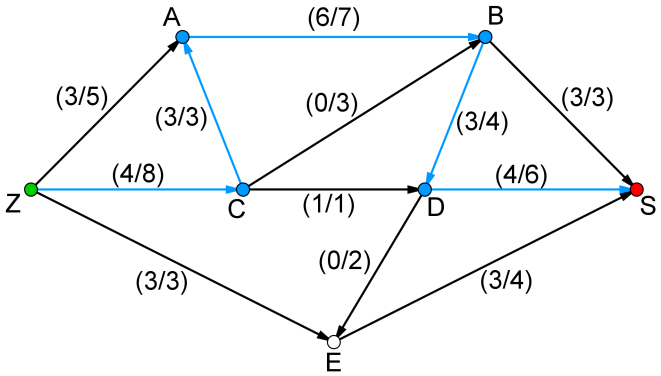
Nalezli jsme cestu: $e_2, e_5, e_4, e_7, e_{10}$.

$$r = min\{e_2, e_5, e_4, e_7, e_{10}\}, \text{ tzn.:} min\{7, 3, 4, 4, 5\} = 3;$$



9. Nasytíme tok na nalezené cestě o minimální rezervu:

$$F_e = (3, 4, 3, 6, 3, 3, 3, 0, 1, 4, 0, 3)$$



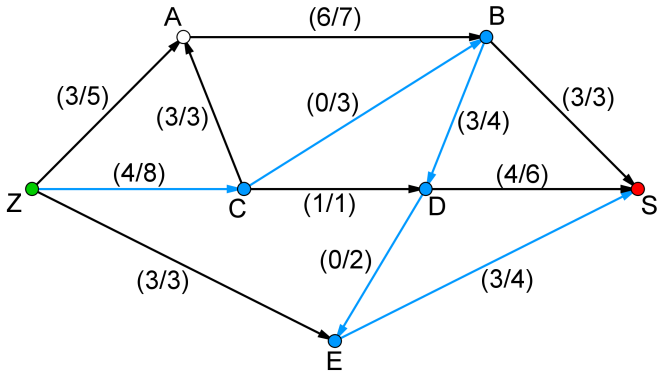
a pokračujeme znovu krokem 2.

10. Označíme a nastavíme rezervy jednotlivých hran:

$$R_e = (2, 4, 0, 1, 0, 0, 1, 3, 0, 2, 2, 1)$$

Nalezli jsme cestu: $e_2, e_8, e_7, e_{11}, e_{12}$;

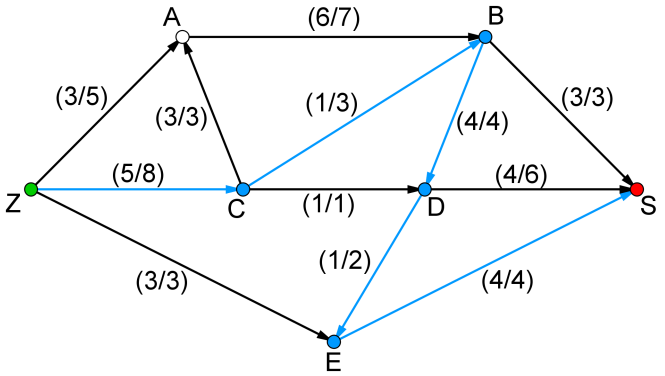
$$r = min\{e_2, e_8, e_7, e_{11}, e_{12}\}, \text{ tzn.:} min\{4, 3, 1, 2, 1\} = 1$$



11. Nasytíme tok na nalezené cestě o minimální rezervu:

$$F_e = (3, 5, 3, 6, 3, 3, 4, 1, 1, 4, 1, 4)$$

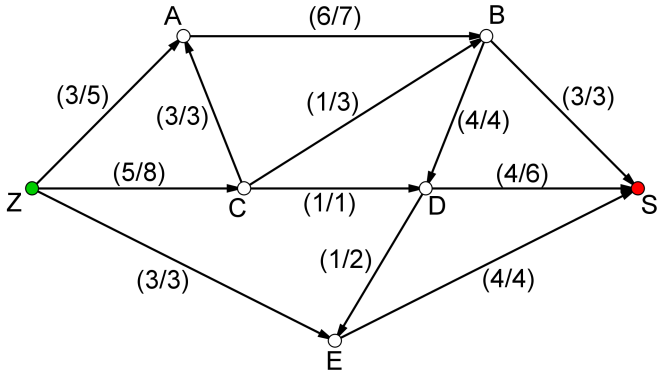
a pokračujeme krokem 2.



12. Označíme a nastavíme rezervy jednotlivých hran:

$$R_e = (2, 3, 0, 1, 0, 0, 0, 2, 0, 2, 1, 0)$$

Nenalezli jsme žádnou zlepšující cestu ⇒ algoritmus končí. Nalezli jsme maximální tok v síti, který je velikosti 11.



Poznámka 6.2. Způsobů, jak k maximálnímu toku v síti dojdeme, může být několik, ale hodnota maximálního toku je vždy stejná.

Závěr

Cílem této práce bylo názorně a jednoduše čtenáři představit a popsat princip fungování několika základních grafových algoritmů. U každého algoritmu definovat vše potřebné, popsat jeho strukturu, vlastní algoritmus a ukázat po jednotlivých krocích s názornou ilustrací jeho fungování na příkladu. Domnívám se, že tento cíl byl úspěšně naplněn. Do této práce bylo zahrnuto jen několik vybraných algoritmů pro prohledávání grafů, hledání nejkratších cest a toků v sítí. Ovšem algoritmů pro řešení zde uvedených, případně dalších grafových problémů, jako např. algoritmy pro barvení grafu, hledání Eulerovských tahů nebo Hamiltonovské kružnice je velké množství a výsledná práce i s výše uvedenými algoritmy by rozsahem mnohonásobně překročila daný rozsah. S dalšími algoritmy zde neuvedenými, které řeší výše zmíněnou problematiku se můžete seznámit v literatuře uvedené v referencích na konci této práce.

Podle mého názoru přínosem této práce je, že podle popisu a obrázků u jednotlivých příkladů je velice snadné pochopit fungování jednotlivých grafových algoritmů. Toho by se dalo využít například při výuce teorie grafů a jiných předmětů, které s teorií grafů a grafovými algoritmy souvisí.

V případě, že by se čtenář chtěl dále zabývat studiem grafů v počítači, doporučuji použití již zmíněných programů v úvodu. Pro základní jednoduché kreslení grafů, bez znalosti příslušného jazyka, program Geogebra [11] a pro pokročilejší práci s grafy, program wxMaxima [12]. Velkou výhodou těchto programů je, že si je čtenář může volně stáhnout a užívat na svém počítači, jelikož jsou volně šiřitelné (odkazy uvedeny v referencích). Nevýhodou programu wxMaxima je nutnost znát příslušný jazyk pro komunikaci s tímto programem, proto doporučuji při jeho používání užívat manuál. Dalším vhodným programem je Wolfram Mathematica [13]. Jedná se o obdobný program jako wxMaxima, ale podle mého názoru má lépe a přehledněji vytvořený manuál. Jeho velkou nevýhodou ovšem je, že se jedná o placený software. Po registraci ho lze sice využívat 30 dní, ale nelze z něj exportovat.

Reference

[1] OCHODKOVÁ E. : *Grafové algoritmy*, FEI VŠB - Technická univerzita Ostrava, 2003

[2] NEŠETŘIL J. : *Teorie grafů* 1. vydání, Praha: Nakladatelství SNTL, 1979

[3] MATOUŠEK J., NEŠETŘIL J. : *Kapitoly z diskrétní matematiky*, UK v Praze, Praha: Nakladatelství Karolinum, 2002, ISBN 80-246-0084-6

[4] HLINĚNÝ P. : *Diskrétní matematika* verze 1.02, FEI VŠB - Technická univerzita Ostrava, 2006

[5] HLINĚNÝ P. : *Teorie Grafů* verze 1.00, FEI VŠB - Technická univerzita Ostrava, 2008

[6] VOLEK J. : *Operační výzkum I*, Univerzita Pardubice, Dopravní fakulta Jana Pernera, 2001, ISBN 80-7194-410-6

[7] VEČERKA A. : *Grafy a grafové algoritmy*, PřF UP Olomouc, 2007

[8] WIKIPEDIE : *Graf (teorie grafů)* [Online] http://cs.wikipedie.org/wiki/Graf_(teorie_grafů)

[9] WIKIPEDIE : *Kružnice (graf)* [Online] http://cs.wikipedia.org/wiki/Kružnice_(graf)

[10] WIKIPEDIE : *Orientovaný graf* [Online] http://cs.wikipedia.org/wiki/Orientovaný_graf

[11] GEOGEBRA : *O programu* [Online] http://www.geogebra.org/cms/cs/

[12] WXMAXIMA : *Home* [Online] http://andrejv.github.io/wxmaxima/index.html

[13] WOLFRAM MATHEMATICA : *Overwiev* [Online] http://www.wolfram.com/mathematica/