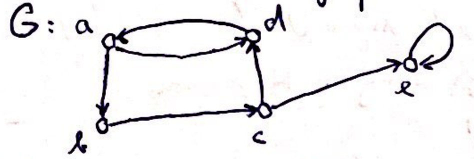
10. ORIENTOVANÉ GRAFY

Orientovaný graf

- dvojice $G=(U,H)$ kde U je konečná množina uzlů a H je konečná množina orientovaných hran, $H = \{(u,v) \mid u,v \in U\}$
- v obyčejném grafu je H množina (konečná) dvouprvkových množin

X

- v orientovaném grafu je H konečná množina dvojic
- existuje zobrazení $\varepsilon: H \to \{(u,v) \mid u,v \in U\}$ – zobrazení dává každé hraně z H dvojici uzlů, kde ta hrana vede z prvního do druhého
- oproti obyčejnému grafu se v orientovaném grafu může u rovnat v a tedy tam mohou být smyčky



$G = (\{a,b,c,d,e\}, \{(a,b),(a,d),(d,a),(b,c),(c,d),(c,e),(e,e)\})$

Výstupní a vstupní stupeň uzlu

- uzel $u \in U$ kde $G=(U,H)$
- $deg_-(u)$ je výstupní stupeň uzlu u – tedy počet hran které z u odcházejí
- $deg_+(u)$ je vstupní stupeň uzlu u – tedy počet hran které do u přicházejí
- $deg_-(u) = |N|$ – mohutnost množiny N kde $N = \{h \in H \mid \exists v \in U: h=(u,v)\}$
- $deg_+(u) = |M|$ – mohutnost množiny M kde $M = \{h \in H \mid \exists v \in U: h=(v,u)\}$

předchozí G:

	$deg_+(u)$	$deg_-(u)$
a	1	2
b	1	1
c	1	2
d	2	1
e	2	1

- součet všech vstupních stupňů grafu se rovná součtu všech výstupních stupňů grafu
- smyčka zvyšuje odchozí i příchozí stupeň

- uzel u kde $deg_+(u)=0$ je počáteční uzel
- uzel u kde $deg_-(u)=0$ je koncový uzel

Sled, tah, cesta, kružnice

- posloupnosti uzlů a hran
- stejné jako u obyčejného (neorientovaného) grafu a tedy sled – libovolně se opakující hrany a uzly akorát mezi dvěma po sobě jdoucími uzly musí vést v tomto případě orientovaná hrana tímto směrem, tah je sled kde každá hrana maximálně 1x, cesta je tah kde každý uzel maximálně 1x a kružnice je cesta kde počáteční uzel se rovná koncovému

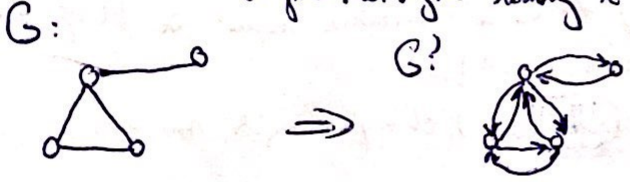
→ všude zde musí vést orientované cesty ve směru kterým jde sled $u_0,h_1,u_1,h_2,\dots,u_{n-1},h_n,u_n$ každá hrana h_i vede z u_{i-1} do u_i a je orientovaná

Symetrická orientace grafu

– máme graf $G=(U,H)$ který je obyčejný (neorientovaný) a jeho symetrickou orientací je graf $G'=(U',H')$ kde $U'=U$ a kde:

$$\forall \{u,v\} \in H \text{ kde } u,v \in U : (u,v) \in H' \wedge (v,u) \in H'$$

a jiné než byla hrany v H' nejsou



– každou neorientovanou hranu nahradíme dvěma orientovanými

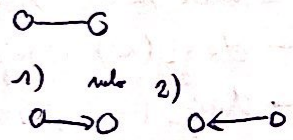
Orientace grafu

– máme graf $G=(U,H)$ který je obyčejný (neorientovaný) a jeho orientací je graf $G'=(U',H')$ kde $U'=U$ a kde platí:

$$\forall \{u,v\} \in H \text{ kde } u,v \in U : (u,v) \in H' \vee (v,u) \in H'$$

a jiné než byla hrany v H' nejsou

– každou neorientovanou hranu nahradíme jednou orientovanou v jednom nebo dvěma směrech



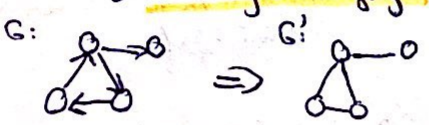
– symetrická orientace je vždy jen jedna
– orientací může být i víc

Symetrizace grafu

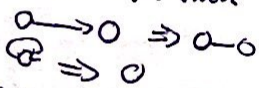
– máme orientovaný graf $G=(U,H)$ a jeho symetrizací je graf neorientovaný (obyčejný) $G'=(U',H')$ kde platí:

$$H' = \{\{u,v\} \mid u,v \in U, u \neq v, (u,v) \in H \vee (v,u) \in H\}$$

a tedy vede nejvýš jedna hrana mezi



– každou orientovanou hranu nahradíme obyčejnou neorientovanou hranou



Souvislost

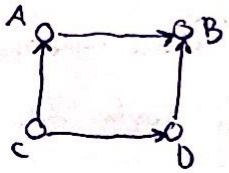
– graf orientovaný G je souvislý jestliže i jeho symetrizace je souvislá

Silná souvislost

– graf orientovaný G je silně souvislý jestliže pro každé dva uzly $u,v \in U$ platí že z u do v vede cesta a z v do u vede cesta

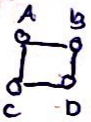
pro každé 2 uzly platí, že se dá dostat z jednoho do druhého a z druhého do prvního – vždy touž cestou

~~orientovaná hrana jedním nebo dvěma směry tedy $(u,v) \in H$ nebo $(v,u) \in H$~~



je souvislý

protože



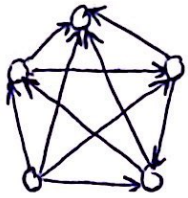
ale není silně souvislý protože např. není z B nevede cesta do A ani do C ani do B

– že se z jednoho do druhého a naopak

Turnaj

- orientovaný graf $G=(V,H)$ ve kterém platí, že mezi libovolnými dvěma vrcholy vede orientovaná hrana jedním či druhým směrem a nejsou zde smyčky

$$\forall u,v \in V: (u,v) \in H \vee (v,u) \in H \text{ a } (u,u) \notin H$$



Turnaj vs. Silně souvislý graf

- v turnaji je mezi libovolnými dvěma vrcholy hrana jedním či druhým směrem
- v silně souvislém grafu je mezi libovolnými dvěma vrcholy cesta jedním i druhým směrem

Eulerovský graf

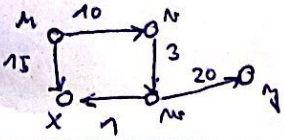
- je graf $G=(V,H)$, který obsahuje orientovaný tah (každá hrana max. 1x) který obsahuje všechny hrany orientovaného grafu
- všechny hrany se dají nakreslit jedním tahem a přes žádnou se nejede 2x
- platí pro každý uzel $u \in V$ platí, že $\deg_-(u) = \deg_+(u)$ pak graf G je eulerovský

Hamiltonovský graf

- hamiltonovská cesta – je cesta v orientovaném grafu (bez každá hrana max. 1 x a každý uzel max. 1x) která obsahuje všechny vrcholy v grafu
  - v každém turnaji je orientovaná hamiltonovská cesta
- hamiltonovská kružnice – je kružnice (cesta kde první a poslední uzel se rovnají) která obsahuje všechny body grafu
- hamiltonský graf – obsahuje hamiltonovskou kružnici

Délka hrany a cesty

- G je orientovaný graf bez smyček
- každé hraně h je přiřazeno kladné reálné číslo $l(h)$ nazývající se její délka
- délka p má délku $l(p)$ kde l délka cesty je dána délkou všech hran tohoto cesty (součtem) / cesta o jednom vrcholu má délku $l(p)=0$
- pokud mezi $u,v \in V$ vede cesta minimální délky pak její délka je $d(u,v)$ a pokud žádná cesta není tak $d(u,v)=\infty$



$d(u,x)=14$ $d(u,y)=33$ $d(y,x)=\infty$

– jsou tu i cesty 15 ale 14 je kratší

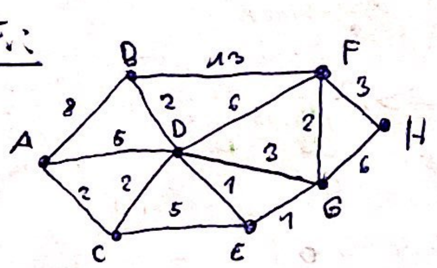
Dijkstrův algoritmus

- hledá nejkratší cestu v grafu z bodu u do bodu v (uzlu)
- v orientovaných grafech
- používá se v routovacích protokolech
- je použitelný pouze pro nezáporné ohodnocení hran – není schopen zajistit, že při zpracování uzlu □ již byla nalezena jeho nejkratší cesta
  - → zajišťuje nám Bellman-Fordův algoritmus
- prohledávání grafu do šířky – nešíří se dle počtu hran od zdroje ale vzdálenosti
- složitost závisí na prioritní frontě a vyhledávání nejkratší cesty do prvků ve frontě
  - nehrnuté vyhledávání $O(|U|^2)$ tedy kvadratická složitost
  - použití haldy (heap) $O(|H| \log_2 |U|)$ tedy logaritmická složitost

Postup algoritmu:

1) je vytvořena množina S všech uzlů pro které již byla algoritmem stanovena cesta nejkratší délky od počátečního uzlu
2) do této množiny vlož počáteční uzel s hodnotou 0 a tím jej označ jako zpracovaný
3) dokud nejsou v množině zpracovaných uzlů všechny uzly dělej následující:
4) všechny následníky tohoto uzlu přidej do prioritní fronty (priorita ve frontě značí vzdálenost do uzlu) – pokud již je jeden následník v množině zpracovaných tak jej přeskoč
   - pokud ve frontě již jsou tak zkontroluj zda vzdálenost do toho předchůdce kterého jsme označili před chvílí jako zpracovaného + vzdálenost z předchůdce do tohoto aktuálně zkoumaného uzlu není menší než ta co je u něj uvedena ve frontě → pokud ano tak mu ji ve frontě změň a zaznamenej že se šlo přes toho předchůdce
   - pokud ve frontě nejsou tak je tam přidej a vzdálenost nastav do toho předchůdce + z předchůdce do tohoto uzlu a označ že se do tohoto uzlu šlo z toho předchůdce
   - pokud nevede hrana a ve frontě nejsou tak dej ∞
5) vyber z prioritní fronty uzel s nejmenší vzdáleností a jdi do kroku 3)

Př.:



	A	B	C	D	E	F	G	H
A	[0_A]	8	2_A	5	∞	∞	∞	∞
			[2_A]					

↳ vybral jsem uzel C protože z té prioritní fronty má nejvyšší prioritu – označím jej jako zpracovaný a zkoumám vzdálenosti z C do B, D, E, F, G, H → do A už nás nezajímá protože to už je zpracované
– ty co jsou ve frontě a mají varianta pro ně není kratší tam nechávám a jen opíšu

	A	B	C	D	E	F	G	H
A	[0_A]	8	2	5	∞	∞	∞	∞
C		8	[2_A]	4	7	∞	∞	∞
D		6		[4_C]	5	10	7	∞
E		6			[5_D]	10	6	∞
B		[6_D]				10	6	∞
G						8	[6_E]	12
F						[8_B]		[11_F]

Nejkratší cesty z A:

A → A = 0
A → B (ACDB) = 6
A → C = 2
A → D (ACD) = 4
A → E (ACDE) = 5
A → F (ACDBF) = 8
A → G (ACDEG) = 6
A → H (ACDBFH) = 11 → přes co to jde

Floyd-Warshallův algoritmus

- umí najít nejkratší cestu i v grafu se záporně ohodnocenými hranami
- nalezne nejkratší cestu mezi každými dvěma uzly
- Dijkstrův algoritmus najde nejkratší cestu z jednoho uzlu do všech ostatních
- časová složitost má $O(N^3)$ tedy kubickou a paměťová $O(N^2)$ tedy kvadratickou
- umí nalézt v grafu kružnici záporné délky a ukončit tak výpočet nejkratších cest

→ ta kružnice by nám zkracovala vzdálenost do všech uzlů čím víckrát bychom jí prošli

Postup algoritmu:

— pokud se v A^n někde na diagonále objeví něco jiného než 0 tak je tam kružnice záporné délky a konec

1) sestaví se matice A_0 s hodnotami cest mezi uzly a matice P_0 předchůdců která se inicializuje že každý prvek sloupce má číslo sloupce
2) prochází se v každém kroku matice tedy se vždy vybere jeden řádek a sloupec jako referenční — tedy 1. řádek a 1. sloupec, potom 2. řádek a 2. sloupec atd...
   - nakonec n-tý řádek a n-tý sloupec a pak se skončí
3) pro všechny ostatní prvky matice které nejsou v právě vybraném sloupci a řádku se zjistí zda jejich hodnota je větší než součet průmětů do vybraného sloupce a řádku

1	3	2
2	8	4
5	6	1

→ jestli $8 > 2+3$

- a pokud ano tak se do toho pole zapíše ten výsledek součtu a do matice P se na totéž místo dá index iterace při které došlo ke změně v matici — pro vybraný 1. sloupec a 1. řádek se při změně dá 1 do P na to místo atd... — j-tá iterace — počítá se 0
- pokud to je ≤ tak se to nechá a jde se na další hodnotu

$$\text{if}\,(a_{ik}^{j-1} > a_{ij}^{j-1} + a_{jk}^{j-1} : a_{ik}^{j} = a_{ij}^{j-1} + a_{jk}^{j-1};\ p_{ik}^{j} = j)$$

$$\text{if}\,(a_{ik}^{j-1} \leq a_{ij}^{j-1} + a_{jk}^{j-1} : a_{ik}^{j} = a_{ik}^{j-1};\ p_{ik}^{j} = p_{ik}^{j-1})$$ — nechá se původní

j - vždy značí iteraci

i, k - značí daný zkoumaný prvek — pro j=1 jsou to i, k které nabývají hodnot z {2,3} a můžou se to kombinovat ((2,2), (3,2), (2,3), (3,3))

$$A^0 = \begin{bmatrix} 0 & 4 & -3 & \infty \\ -3 & 0 & -7 & \infty \\ \infty & 10 & 0 & 3 \\ 5 & 6 & 6 & 0 \end{bmatrix} \quad P^0 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix} \Rightarrow A^1 = \begin{bmatrix} 0 & 4 & -3 & \infty \\ -3 & 0 & -7 & \infty \\ \infty & 10 & 0 & 3 \\ 5 & 6 & 2 & 0 \end{bmatrix} \quad P^1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 1 & 4 \end{bmatrix}$$

→ $6 > (5-3)$; → j=1 tedy

$$\Rightarrow A^2 = \begin{bmatrix} 0 & 4 & -3 & \infty \\ -3 & 0 & -7 & \infty \\ 7 & 10 & 0 & 3 \\ 3 & 6 & -1 & 0 \end{bmatrix} \quad P^2 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 2 & 2 & 3 & 4 \\ 2 & 2 & 2 & 4 \end{bmatrix}$$

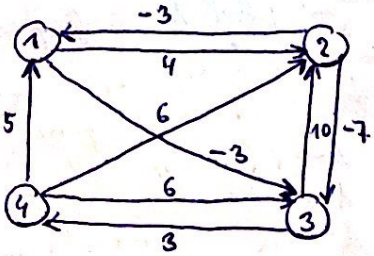
$\infty > (-3+10)$; → $5 > (-3+6)$; → $2 > (-7+6)$

$$\Rightarrow A^3 = \begin{bmatrix} 0 & 4 & -3 & 0 \\ -3 & 0 & -7 & -4 \\ 7 & 10 & 0 & 3 \\ 3 & 6 & -1 & 0 \end{bmatrix} \quad P^3 = \begin{bmatrix} 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 3 \\ 2 & 2 & 3 & 4 \\ 2 & 2 & 2 & 4 \end{bmatrix}$$

→ $\infty > (-3+3)$; → $\infty > (3-7)$

$$\Rightarrow A^4 = \begin{bmatrix} 0 & 4 & -3 & 0 \\ -3 & 0 & -7 & -4 \\ 6 & 9 & 0 & 3 \\ 3 & 6 & -1 & 0 \end{bmatrix} \quad P^4 = \begin{bmatrix} 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 3 \\ 4 & 4 & 3 & 4 \\ 2 & 2 & 2 & 4 \end{bmatrix}$$ HOTOVO

G:



- vždy jsem si v A^{j-1} obarvoval ten kříž už pro zkoumání pro další iteraci
- vidíme třeba že nejkr. cesta z 1 do 4 je 0 nebo z 2 do 3 je −7 atd...