

NEJKRATŠÍ CESTY MEZI VŠEMI UZLY

Nejkratší cesty mezi všemi uzly

Seznámíme se s následujícími pojmy:

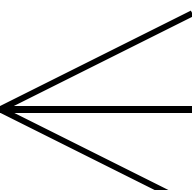
- **matice w-délek, matice w-vzdáleností, matice předchůdců**
- **Floydův-Warshallův algoritmus**
- **Johnsonův algoritmus, přehodnocení hran**

Skripta kap. 7, str. 123 – 138

Jak zjistíme nejkratší cesty mezi všemi páry uzlů ?

Označme $|U| = n$

Pro nezáporné ohodnocení hran lze uvažovat

$n \times$ Dijkstra ... 

- $O(n^2 \cdot \lg n + n \cdot |H|)$ Fibonacci-ho halda
- $O(|H| \cdot n \cdot \lg n)$ binární halda
- $O(n^3)$ fronta jako sekvenční seznam

A pro záporné ohodnocení hran

$n \times$ Bellman-Ford ... $O(n^2 \cdot |H|) \sim O(n^4) !?$

? Jde to lépe ?

ANO !

Graf G reprezentujeme **maticí w-délek W** (vychází z matice sousednosti V a zahrnuje současně délky hran w):

$$w_{ij} = \begin{cases} 0 & \text{pro } i = j \\ w(u_i, u_j), & i \neq j, (u_i, u_j) \in H \\ +\infty & \text{jinak} \end{cases}$$

Výsledky získáme ve formě

matice w-vzdáleností $D = [d_{ij}] : d_{ij} = d_w(u_i, u_j)$

Doplňující informace o samotných cestách ukládáme do matice předchůdců $P = [p_{ij}]$

$$p_{ij} = \begin{cases} \text{null} & \dots i=j \text{ nebo neexistuje cesta } u_i \rightarrow u_j, \\ u_k & \dots u_k \text{ je předchůdce } u_j \text{ na cestě } u_i \rightarrow u_j \end{cases}$$

Algoritmus Floyd-Warshalla

Základní myšlenka:

postupně rozšiřujeme povolené vnitřní uzly minimálních cest:

$\emptyset, \{u_1\}, \{u_1, u_2\}, \dots, \{u_1, u_2, \dots, u_k\}, \dots, \{u_1, u_2, \dots, u_n\}$

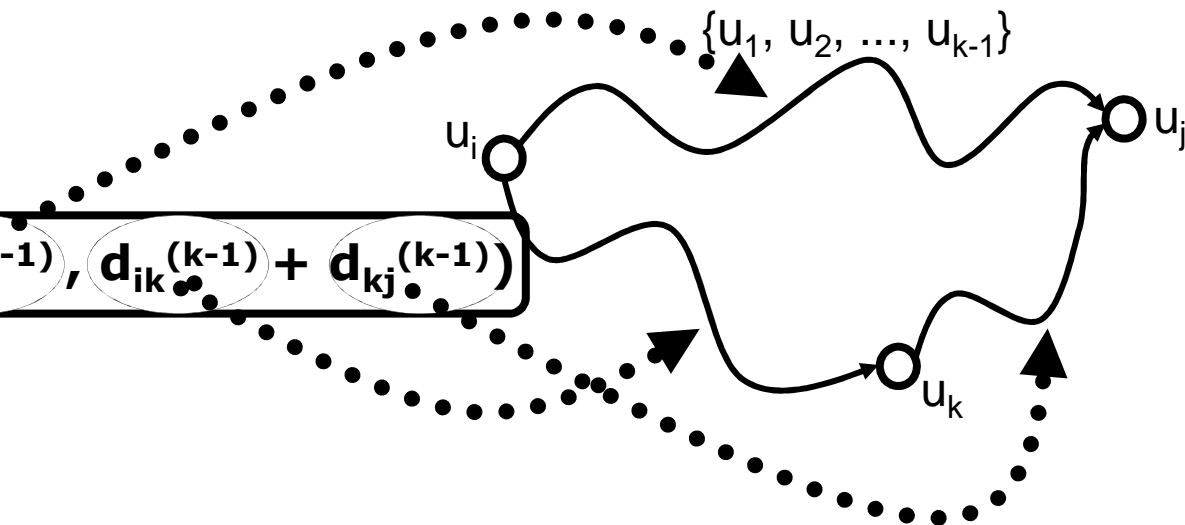
cesta $P : u_i \rightarrow u_j$ s vnitřními uzly z $\{u_1, u_2, \dots, u_k\}$

$d_{ij}^{(k)}$ - délka minimální cesty P

$$d_{ij}^{(0)} = w_{ij}$$

$$d_{ij}^{(k)} = \min (d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$

$$d_{ij}^{(n)} = d_{ij}$$



Algoritmus Floyd-Warshalla

```
1 D(0) = W;  
2 for (k=1; k<=n; k++) {  
3   for (i=1; i<=n; i++) {  
4     for (j=1; j<=n; j++) {  
5       dij(k) = min (dij(k-1), dik(k-1) + dkj(k-1));  
6     } } }  
7 return D(n);
```

!!! Časová složitost : $O(n^3)$!!!

? A co paměťová složitost ?

?A co, když chceme znát cesty, ne jen vzdálenosti ?

Výpočet matice předchůdců:

$$p_{ij}^{(0)} = \begin{cases} \text{null} & \text{pro } i=j \text{ nebo } w_{ij} = \infty \\ i & \text{jinak (tedy pro } (u_i, u_j) \in H) \end{cases}$$

$$p_{ij}^{(k)} = \begin{cases} p_{ij}^{(k-1)} & \text{pro } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ p_{kj}^{(k-1)} & \text{pro } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

Reflexivně-tranzitivní uzávěr grafu (relace)

$$W = V \vee E \quad (\text{Booleovská výchozí matice})$$

$$d_{ij}^{(k)} = d_{ij}^{(k-1)} \vee (d_{ik}^{(k-1)} \wedge d_{kj}^{(k-1)}) \quad - \text{dostupnost}$$

Kontrolní otázky

- 8.1** Podobně jako je při provádění Floyd-Warshallova algoritmu možné počítat matici P předchůdců uzlů na nejkratších cestách, je možné také počítat matici Q následníků uzlů na nejkratších cestách. Určete pravidlo, podle něhož se nastaví počáteční hodnoty prvků $q_{ij}^{(0)}$ této matice, a pravidlo pro přechod od $(k-1)$ -ní ke k -té iteraci hodnot q_{ij} .
- 8.2** Dalším rozšířením Floyd-Warshallova algoritmu zajistěte, aby po ukončení výpočtu byl znám počet hran na nejkratších cestách mezi všemi uzly (opět ve formě matice označené např. R). (Návod: Inspirujte se řešením obdobného problému pro algoritmus Dikstrův a Bellman-Fordův.)
- 8.3** Zdůvodněte, proč je provádění Floyd-Warshallova algoritmu možné všechny iterace matice $D^{(k)}$ uchovávat v jediném poli. (Návod: Ověřte, že vnitřní dva cykly nemění hodnotu prvků v k -tém řádku a k -tém sloupci, na nichž závisí hodnoty prvků v nové iteraci.)
- 8.4** Jak se při použití Floyd-Warshallova algoritmu zjistí případná existence záporných cyklů v grafu?

Johnsonův algoritmus

Úvaha: (označíme $|U|=n$)

Pro **řídke grafy** ($|H| \ll O(n^2)$) je $O(|H|.n)$ lepší než $O(n^3)$
takže také $O(n.(|H| + n.\lg n))$ je lepší než $O(n^3)$

Idea: n x Dijkstra (+ Fib. heap), **ale co s $w(h) < 0$??**

Přehodnocení $w \rightarrow w'$ takové, že

- pro každé u, v je cesta $u \rightarrow v$ minimální podle w' minimální také podle w
- $w'(u, v) \geq 0$

? Jak najít přehodnocení w' , které bude zachovávat minimálnost cest ?

V: Nechť je pro graf $G = \langle H, U \rangle$ s ohodnocením hran $w: H \rightarrow R$ dána funkce $h: U \rightarrow R$. Nechť je ohodnocení w' pro každou hranu (u,v) dané vztahem

$$w'(u,v) = w(u,v) + h(u) - h(v)$$

Potom pro cestu $L: u \rightarrow v$ platí, že L je minimální podle w , **právě když** je minimální podle w' .

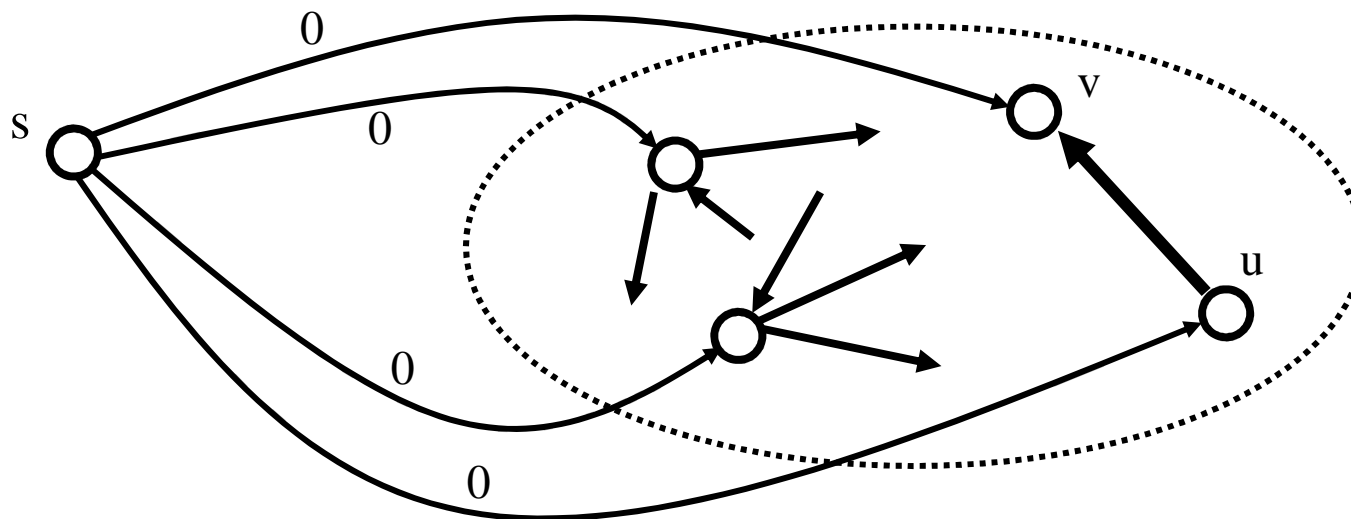
D: platí $w'(L) = w(L) + h(u) - h(v)$

\Rightarrow cesta minimální pro w je minimální i pro w' a naopak

Zbývá nalézt vhodné h , aby bylo $w'(u,v) \geq 0$!!!

Úprava G na G':

- přidáme uzel s : $U' = U \cup \{s\}$
a hrany $H = H \cup \{(s,u): u \in U\}$, $w(x) = 0$ pro nové hrany x



- položíme $h(u)=d(s,u)$... platí $h(v) \leq h(u) + w(u,v)$, tedy
 $w'(u,v) = w(u,v) + h(u) - h(v) \geq 0$!!!

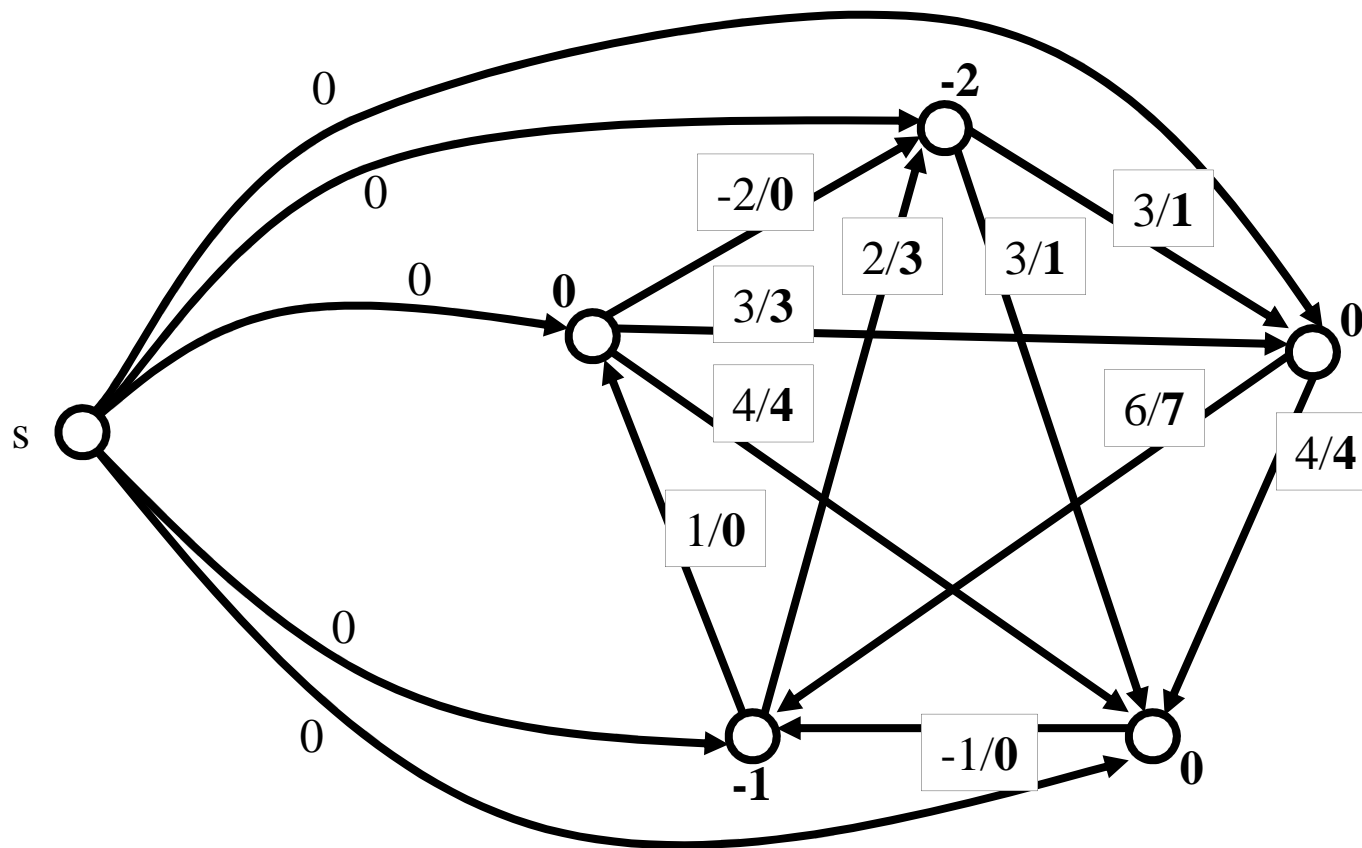
Johnsonův algoritmus

```
1  "G doplníme přidáním uzlu s na G' "  
2  if (!Bellman-Ford(G',w,s)) return false;  
3  for ( Node u in U(G) ) h(u) = d(s,u).  
4  for ( Edge (u,v) in H(G) )  
5      w'(u,v) = w(u,v) + h(u) - h(v).  
6  for ( Node u in U(G) ) {  
7      Dijkstra(G, w', u);  
8      for ( Node v in U(G) )  
9          d(u,v) = d'(u,v) - h(u) + h(v);  
10 }
```

$O(n \cdot |H|)$

$O(n^2 \cdot \lg n + n \cdot |H|)$
(pro Fib. heap)

$O(n \cdot |H| \cdot \lg n)$ pro binární heap



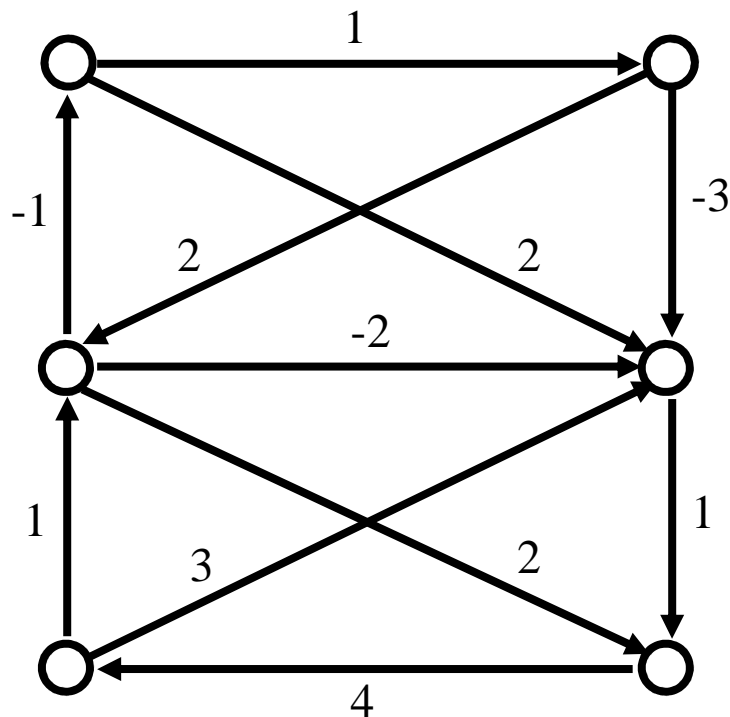
$$h(u) = \min (0, \text{délka nejkratší cesty } s \rightarrow u)$$

$$w'(u,v) = w(u,v) + h(u) - h(v)$$

Kontrolní otázky

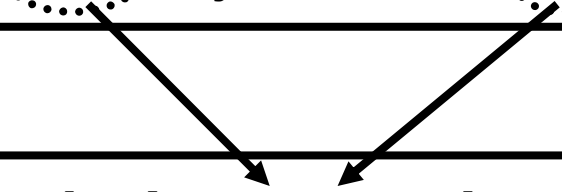
8.5 Jaký vztah platí mezi ohodnoceními $w(u,v)$ a $w'(u,v)$, pokud jsou hodnoty $w(u,v) \geq 0$ pro všechny hrany (u,v) ?

8.6 Pomocí Johnsonova algoritmu určete matici vzdáleností pro následující orientovaný graf :



Algebraické souvislosti

Floyd-Warshall:

$$d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$


dvojoperace min, +
označíme \oplus, \otimes

\otimes efekt složení dvou spojení $u \rightarrow v \otimes v \rightarrow x$

\oplus efekt kombinování alternativních spojení $u \rightarrow v \oplus v \rightarrow x$

?Jaké vlastnosti musí tyto operace mít, aby to fungovalo?

Polookruh

Ted' přijde trochu matematiky, podrobněji viz skripta ...

$P = \langle P, \oplus, \otimes, 0, 1 \rangle :$

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c$$

$$a \oplus 0 = 0 \oplus a = a$$

$$a \oplus b = b \oplus a$$

$$a \oplus a = a \text{ idempotence}$$

$$a \otimes (b \otimes c) = (a \otimes b) \otimes c$$

$$a \otimes 1 = 1 \otimes a = a$$

$$a \otimes 0 = 0 \otimes a = 0 \text{ s nulovým prvkem}$$

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

$$(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$$

$\langle P, \oplus, 0 \rangle$

je komutativní
monoid

$\langle P, \otimes, 1 \rangle$

je monoid

distributivnost

zleva a zprava

Uzavřený polookruh:

$a_1 \oplus a_2 \oplus a_3 \oplus \dots$ je definováno pro lib. a_1, a_2, a_3, \dots a je

- asociativní, komutativní, idempotentní a navíc
- distributivní vůči \otimes

Kdy máme ∞ mnoho spojení? Když máme cykly!

Uzávěr $c^* = 1 \oplus c \oplus (c \otimes c) \oplus (c \otimes c \otimes c) \dots$

$$0^* = 1, \quad c \otimes c^* = c^* \otimes c, \quad c^* = 1 \oplus (c^* \otimes c), \dots$$

Příklady polookruhů

$$\langle \mathbb{R}^+ \cup \{\infty\}, \min, +, \infty, 0 \rangle \quad a^* = \min(0, a, a+a, a+a+a, \dots) = 0$$

$$\langle \mathbb{R} \cup \{\infty, -\infty\}, \min, +, \infty, 0 \rangle \quad a^* = 0 \text{ nebo } -\infty \text{ (pro } a < 0)$$

Řešení obecné úlohy o spojeních

Máme jednoduchý OG $G = \langle H, U \rangle$ a ohodnocení

$\omega: H \rightarrow P$, $\omega(h) \neq \mathbf{0}$, kde $\mathbf{P} = \langle P, \oplus, \otimes, \mathbf{0}, \mathbf{1} \rangle$ je polookruh.

Doplníme ohodnocení o

$\omega(u, v) = \mathbf{0}$ pro $(u, v) \notin H$.

Spojení $L = \langle u_1, u_2, \dots, u_k \rangle$ ohodnotíme pomocí jeho hran

$$\omega(L) = \omega(u_1, u_2) \otimes \omega(u_2, u_3) \otimes \dots \otimes \omega(u_{k-1}, u_k)$$

takže platí $\omega(L_1 \circ L_2) = \omega(L_1) \otimes \omega(L_2)$

$$\omega(L_1 \circ L_2 \circ \dots \circ L_r) = \omega(L_1) \otimes \omega(L_2) \otimes \dots \otimes \omega(L_r)$$

Rozšíříme ω i na množiny spojení $\{L_i\}_{i \in I}$ mezi stejnými uzly

$$\omega(\{L_i\}_{i \in I}) = \bigoplus_{i \in I} \omega(L_i)$$

Ta trocha matematiky pokračuje ...

F-W algoritmus zobecníme tak, aby pro všechna $u, v \in U$ počítal

$$s_{uv} = \bigoplus_{L : u \rightarrow v} \omega(L)$$

Předpokládáme $U = \{1, 2, \dots, |U|\}$, $L(i, j, k)$ - všechna spojení z i do j s vnitřními uzly pouze z množiny $\{1, 2, \dots, k\}$.

Postupně se počítají

$$s_{ij}^{(k)} = \bigoplus_{L \in L(i, j, k)} \omega(L)$$

pomocí rekurence

$$s_{ij}^{(k)} = s_{ij}^{(k-1)} \oplus (s_{ik}^{(k-1)} \otimes (s_{kk}^{(k-1)})^* \otimes s_{kj}^{(k-1)})$$

a počátečního nastavení

$$s_{ij}^{(0)} = \omega(i, j) \text{ pro } i \neq j, \quad s_{ij}^{(0)} = 1 \oplus \omega(i, j) \text{ pro } i = j$$

Zobecněný F-W algoritmus

Floyd-Warshall-G(G, ω)

1 **for** ($i=1; i \leq n; i++$) {

2 **for** ($j=1; j \leq n; j++$) $s_{ij}^{(0)} = \omega(i,j);$

3 $s_{ii}^{(0)} = \mathbf{1} \oplus \omega(i,i);$

4 }

5 **for** ($k=1; k \leq n; k++$)

6 **for** ($i=1; i \leq n; i++$)

7 **for** ($j=1; j \leq n; j++$)

8 $s_{ij}^{(k)} = s_{ij}^{(k-1)} \oplus (s_{ik}^{(k-1)} \otimes (s_{kk}^{(k-1)})^* \otimes s_{kj}^{(k-1)});$

9 **return** $\mathbf{S}^{(n)};$

•••za co je tam ta 1 ???•••

A tohle je několik výsledků ...

Pro polookruh

$\langle \mathbf{R}^+ \cup \{\infty\}, \min, +, \infty, \mathbf{0} \rangle$ $a^* = \min(0, a, a+a, a+a+a, \dots) = 0$
 \Rightarrow uzávěrový činitel $(\mathbf{s}_{kk}^{(k-1)})^*$ odpadá

$$s_{ij}^{(k)} = s_{ij}^{(k-1)} \oplus (s_{ik}^{(k-1)} \otimes s_{kj}^{(k-1)})$$

Pro polokruh

$\langle \mathbf{R} \cup \{\infty, -\infty\}, \min, +, \infty, \mathbf{0} \rangle$ $a^* = 0$ nebo $-\infty$ (pro $a < 0$)
 \Rightarrow hodnota $(\mathbf{s}_{kk}^{(k-1)})^*$ je $-\infty$ pokud spojení z u do v prochází
 cyklem < 0

$\mathbf{P} = \langle \langle \mathbf{0}, \mathbf{1} \rangle, \max, ., \mathbf{0}, \mathbf{1} \rangle$ $a^* = 1$ - nejspolehlivější spojení

$\mathbf{P} = \langle \mathbf{R}_k, \max, \min, r_{\min}, r_{\max} \rangle$ - spojení s maximální propustností

\mathbf{R}_k je konečná množina reálných čísel

Kontrolní otázky

8.7 Jak je třeba definovat operace \oplus a \otimes a nosič (tj. množinu P) v odpovídajícím polookruhu, aby zobecněný Floyd-Warshallův algoritmus určil počet různých spojení mezi jednotlivými dvojicemi uzlů?