

# C2142 Návrh algoritmů pro přírodovědce

## 8. Nejkratší vzdálenosti.

Tomáš Raček

Jaro 2017

# Nejkratší vzdálenosti v grafu

---

**Problém.** Určete nejkratší vzdálenost mezi vrcholy  $u$  a  $v$  v grafu.

**Pozorování.** V tuto chvíli můžeme posuzovat vzdálenost pouze jako počet hran na cestě mezi  $u$  a  $v$ .

Takový výpočet realizuje **prohledávání do šířky (BFS)** v lineárním čase vůči velikosti grafu.

**Rozšíření.** Uvažme případ, kdy bychom chtěli přiřadit hranám na cestě různou váhu.

Graf  $G = (V, E, w_e)$  nazveme **hranově ohodnocený**,  $w_e$  je funkce, která každé hraně přiřazuje její ohodnocení – reálné číslo.

**Nejkratší vzdálenost mezi dvěma vrcholy** v grafu je minimální součet ohodnocení hran některé cesty mezi těmito vrcholy.

# Matice vzdáleností

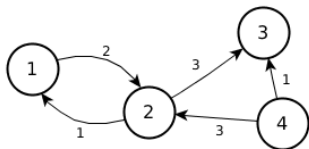
**Poznámka.** Na hranově neohodnocený graf se lze dívat jako na speciální případ ohodnoceného, kde  $\forall (u, v) \in E : w_e(u, v) = 1$ .

**Matice vzdáleností  $W$**  je rozšíření matice sousednosti:

$$W_{i,j} = \begin{cases} 0 & \text{pro } i = j \\ w_e(i, j) & \text{pro } (i, j) \in E \\ \infty & \text{pro } (i, j) \notin E \end{cases}$$

## Příklad

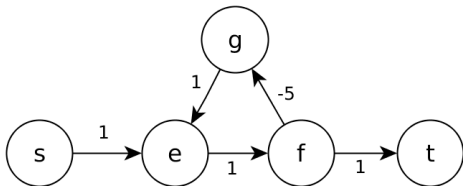
$$W = \begin{pmatrix} 0 & 2 & \infty & \infty \\ 1 & 0 & 3 & \infty \\ \infty & \infty & 0 & \infty \\ \infty & 3 & 1 & 0 \end{pmatrix}$$



# Záporný cyklus

---

**Příklad.** Určete nejkratší vzdálenost mezi vrcholy  $s$  a  $t$  v grafu:



**Pozorování.** Graf obsahuje cyklus záporné délky (vrcholy  $e, f, g$ ), nejkratší vzdálenost mezi  $s$  a  $t$  není definována.

**Poznámka.** Uvažme graf bez cyklů záporné délky. Každá nejkratší cesta mezi dvěma vrcholy obsahuje libovolný vrchol nejvýše jednou.

# Bellman-Fordův algoritmus

---

**Pozorování.** Z předchozí poznámky vyplývá, že nejkratší cesta mezi dvěma vrcholy v grafu obsahuje nejvýše  $|V| - 1$  hran.

**Relaxace hrany.** Necht'  $(u, v)$  je hrana v grafu  $G$  s ohodnocením  $w_e(u, v)$  a hodnoty  $u.d$  a  $v.d$  jsou v daném okamžiku nejkratší nalezené vzdálenosti do  $u$ , resp. do  $v$  z výchozího vrcholu. Zjevně pak platí, že:

$$v.d = \min(v.d, u.d + w_e(u, v))$$

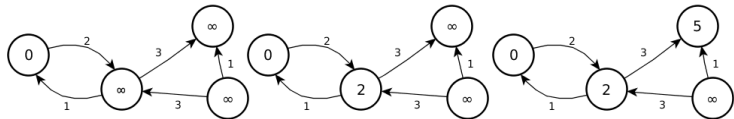
Tato (případná) změna ohodnocení  $v.d$  se nazývá relaxací.

**Bellman-Fordův algoritmus** je založen na těchto dvou principech.

- počítá nejkratší vzdálenosti z výchozího vrcholu do všech ostatních (1:N)
- $(|V| - 1)$ krát relaxuje všechny hrany

# Bellman-Fordův algoritmus – poznámky

**Složitost** Bellman-Fordova algoritmu je  $O(|V| \cdot |E|)$ , relaxace hrany má totiž zřejmě konstantní složitost.



## Pozorování

- Pokud při některé z iterací nedojde ke změně hodnoty  $v.d$  pro žádný vrchol  $v$ , pak je možné výpočet ukončit.
- Provedením jedné iterace výpočtu navíc lze v grafu **detekovat záporné cykly**.
- Pokud dojde k relaxaci hrany, lze u koncového vrcholu nastavit ukazatel na jeho předchůdce  $\rightarrow$  rekonstrukce cesty.

# Bellman-Fordův algoritmus – pseudokód

---

```
1: function Bellman-Ford( $G = (V, E, w_e)$ ,  $s$ ) is
2:    $\forall v \in V : v.d \leftarrow \infty; s.d \leftarrow 0$ 
3:   for  $i \leftarrow 1$  to  $|V| - 1$  do
4:     for all  $(u, v) \in E$  do
5:       if  $v.d > u.d + w_e(u, v)$  then
6:          $v.d \leftarrow u.d + w_e(u, v)$ 
7:       fi
8:     done
9:   done
10:  for all  $(u, v) \in E$  do
11:    if  $v.d > u.d + w_e(u, v)$  then
12:      Error : Negative cycle detected
13:    fi
14:  done
15: end
```

# Dijkstrův algoritmus

---

**Dijkstrův algoritmus** představuje odlišný způsob řešení problému nejkratších vzdáleností v grafu.

- Opět řeší problém typu 1:N.
- Vyžaduje graf s nezáporným ohodnocením všech hran.

**Myšlenka.** Pokud je posloupnost  $u, u_1, \dots, u_n, v$  nejkratší cesta z  $u$  do  $v$ , pak posloupnost  $u, u_1, \dots, u_k$ , kde  $k \leq n$  je nejkratší cesta z  $u$  do  $u_k$ .

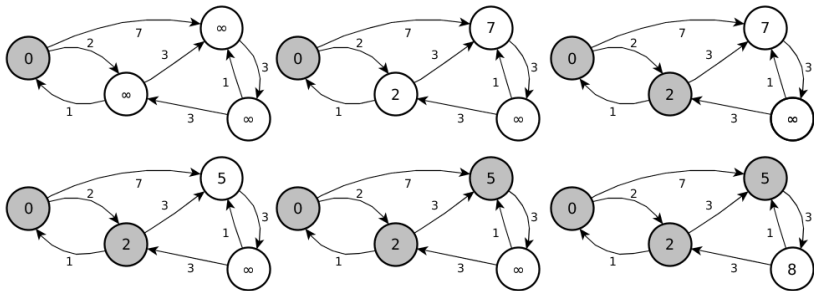
**Výpočet algoritmu.** V každé iteraci rozšiřujeme množinu vrcholů, do kterých již známe nejkratší vzdálenost z výchozího.



# Dijkstrův algoritmus

## Příklad

1. Pokud je  $Q$  množina vrcholů s dosud neurčenou nejkratší vzdáleností, tak z ní nejprve vybereme vrchol  $u$ , jehož ohodnocení  $u.d$  je nejnižší.
2. Poté přepočítáme všechny vzdálenosti do vrcholů z  $Q$ , do kterých vede z  $u$  hrana.



# Dijkstrův algoritmus – pseudokód

---

```
1: function Dijkstra( $G = (V, E, w_e)$ ,  $s$ ) is
2:    $\forall v \in V : v.d \leftarrow \infty$ 
3:    $s.d \leftarrow 0$ 
4:    $Q \leftarrow V$ 
5:   while  $Q$  není prázdná do
6:      $u \leftarrow t \in Q$  s minimální  $t.d$ 
7:     Odstraň  $u$  z  $Q$ 
8:     for all  $v : (u, v) \in E$  do
9:       if  $v.d > u.d + w_e(u, v)$  then
10:         $v.d \leftarrow u.d + w_e(u, v)$ 
11:      fi
12:    done
13:  done
14: end
```

# Dijkstra – volba datové struktury

---

**Složitost** Dijkstrova algoritmu je dána volbou datové struktury pro  $Q$ . Zajímají nás dvě operace:

- $f_{ext}$  – extrakce prvku s minimálním klíčem ( $|V|$  operací)
- $f_{dec}$  – snížení klíče  $v.d$  (nejvýše  $|E|$  operací)

## Seznam vrcholů

- snížení klíče –  $\mathcal{O}(1)$
- extrakce minimálního prvku –  $\mathcal{O}(|V|)$
- celkem  $\mathcal{O}(|E| + |V|^2) = \mathcal{O}(|V|^2)$

## Binární halda

- snížení klíče i extrakce minima –  $\mathcal{O}(\log |V|)$
- celkem  $\mathcal{O}(\log |V| \cdot (|V| + |E|))$

# Nejkratší vzdálenosti mezi všemi dvojicemi vrcholů

---

**Pozorování.** Určit nejkratší vzdálenost mezi dvěma vrcholy (1:1) má stejnou složitost jako určit nejkratší vzdálenost z jednoho vrcholu do všech ostatních (1:N).

**Nejkratší mezi všemi dvojicemi vrcholů** lze spočítat např. využitím  $|V|$  volání Dijkstrova nebo Bellman-Fordova algoritmu, kdy v každém výpočtu volíme jiný výchozí vrchol. Jde to i lépe?

**Floyd-Warshallův algoritmus** počítá nejkratší vzdálenosti mezi všemi dvojicemi vrcholů v čase  $O(|V|^3)$ . Navíc oproti Dijkstrovu algoritmu umí pracovat se zápornými hranami.

# Floyd-Warshallův algoritmus

---

**Myšlenka.** Označme  $d_{i,j}^{(k)}$  délku nejkratší cesty mezi vrcholy  $i$  a  $j$ , kde na této cestě jsou vrcholy pouze z množiny  $\{1, \dots, k\}$ . Zjevně  $d_{i,j}^{(0)} = w_e(i, j)$  a požadovaný výsledek odpovídá  $d_{i,j}^{(|V|)}$ .

Mohou nastat dvě možnosti pro  $d_{i,j}^{(k)}$ :

1.  $k$  není součástí nejkratší cesty  $\rightarrow d_{i,j}^{(k)} = d_{i,j}^{(k-1)}$
2.  $k$  je součástí nejkratší cesty  $\rightarrow d_{i,j}^{(k)} = d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}$

Odtud tedy:

$$d_{i,j}^{(k)} = \min \left\{ d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \right\}$$

**Poznámka.** Pokud  $d_{i,i}^{(|V|)} < 0$  pro nějaké  $i$ , pak graf obsahuje cyklus záporné délky.

# Floyd-Warshallův algoritmus – pseudokód

---

```
1: function Floyd-Warshall( $G = (V, E, w_e)$ ) is
2:    $\forall (u, v) \in \{1, \dots, |V|\} \times \{1, \dots, |V|\} : dist(u, v) \leftarrow \infty$ 
3:    $\forall v \in V : dist(v, v) \leftarrow 0$ 
4:    $\forall (u, v) \in E : dist(u, v) \leftarrow w_e(u, v)$ 
5:   for  $k \leftarrow 1$  to  $|V|$  do
6:     for  $i \leftarrow 1$  to  $|V|$  do
7:       for  $j \leftarrow 1$  to  $|V|$  do
8:         if  $dist(i, j) > dist(i, k) + dist(k, j)$  then
9:            $dist(i, j) \leftarrow dist(i, k) + dist(k, j)$ 
10:        fi
11:     done
12:   done
13: done
14: end
```