

# NEJKRATŠÍ CESTY

# Nejkratší cesty z jednoho uzlu

**Seznámíme se s následujícími pojmy:**

- **w-vzdálenost (vzdálenost na ohodnoceném grafu), relaxace, strom nejkratších cest**
- **Dijkstrův algoritmus**
- **Bellman-Fordův algoritmus**

**Skripta kap. 6, str. 110 – 122**

## Několik obecných úvah

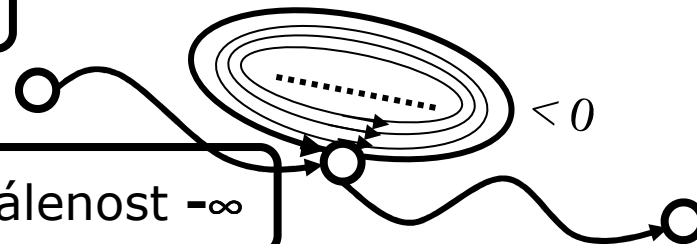
Uvažujeme nejobecnější případ - ohodnocené OG:

**w-délka** spojení  $\langle h_1, h_2, \dots, h_k \rangle = \sum w(h_i)$

**w-vzdálenost**  $d_w(u, v) = w\text{-délka nejkratšího spojení nebo } \infty$

- nedostupné uzly - vzdálenost  $+\infty$

- spojení se záporným cyklem - vzdálenost  $-\infty$



- počítání s nekonečny:

$$a + (-\infty) = (-\infty) + a = -\infty \text{ pro } a \neq \infty$$

$$a + \infty = \infty + a = \infty \text{ pro } a \neq -\infty$$

**? Které z vlastností 0 až 4 má takováto vzdálenost ?**

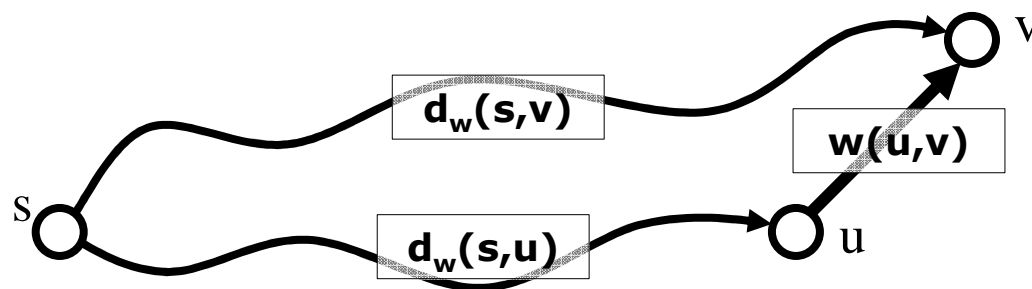
## Varianty úlohy hledání nejkratších cest:



## Důležité zjištění:

Pro libovolnou hranu  $(u,v) \in H$  a uzel  $s \in U$  platí

$$d_w(s,v) \leq d_w(s,u) + w(u,v)$$



**! Platí pro konečné i nekonečné hodnoty  $w$ -vzdáleností !**

### **Datové struktury:**

**d[u]** ... délka (dosud nalezené) minimální cesty

**p[u]** ... předchůdce na (dosud nalezené) minimální cestě

**Q** ... (prioritní) fronta otevřených uzlů (halda?)

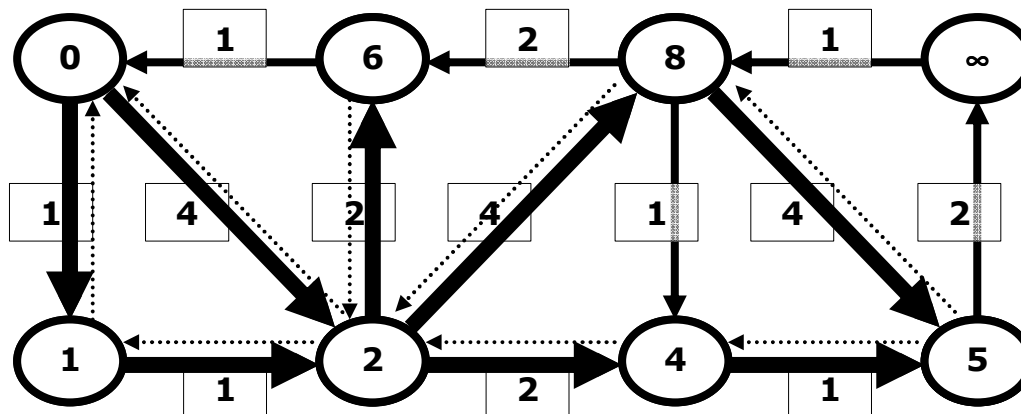
### **Společné operace pro základní varianty algoritmů:**

```
void InitPaths(Graph G, Node s) { // inicializace
1   for (Node u in U(G))
2       { d[u] =  $+\infty$  ; p[u] = null; }
3   d[s] = 0;
4 }
```

```
void Relax (Node u, Node v, Weights w) {
1   if (d[v] > d[u]+w(u,v)) // příp.úprava délky cesty
2       { d[v] = d[u]+w(u,v); p[v] = u; }
3 }
```

**V:** Předpokládejme, že pro nějaký graf provedeme operaci **InitPaths** a pak libovolný počet operací **Relax**. Potom

- platí  $d[u] \geq d_w(s, u)$
- jakmile  $d[u]$  dosáhne hodnoty  $d_w(s, u)$ , už se nemění
- jakmile se žádné  $d[u]$  nemění, máme strom nejkratších cest do všech dosažitelných uzlů z uzlu  $s$



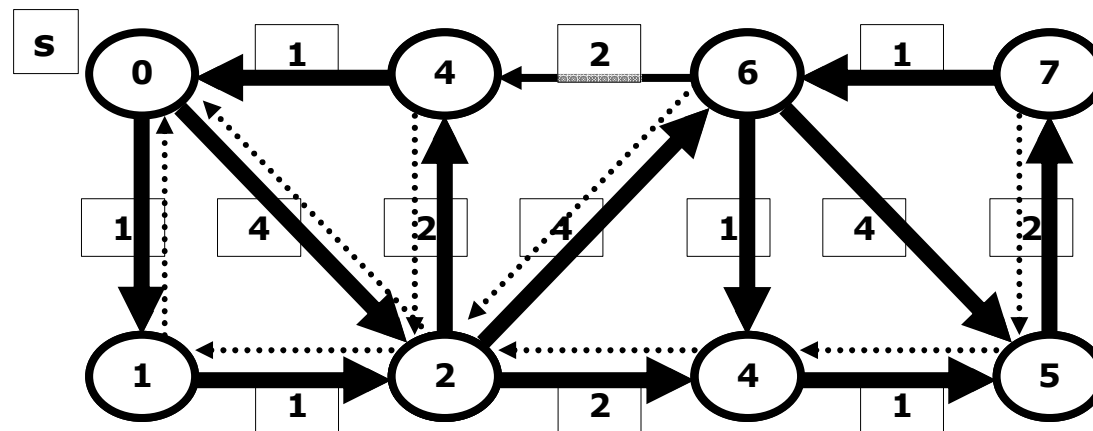
**? V jakém pořadí hran máme provádět relaxaci ?**

**? Jak dlouho máme provádět relaxaci ?**

# Dijkstrův algoritmus

**Základní předpoklad**  $w : H \rightarrow \mathbb{R}^+$  (nezáporné délky hran)

**Jedná se o upravený algoritmus prohledávání do šířky – otevřené uzly se řadí do prioritní fronty (a vybírají) podle hodnoty  $d[u]$  (tedy aproximace  $d_w(s,u)$ )**



```

void Dijkstra(Graph G, Node s, Weights w) {
1   InitPaths(G,s);
2   S = ∅; Queue.Init();
3   for (Node u in U(G)) Queue.Push(u);           O(|U|)
4   while (!Queue.Empty())...{
5       u = Queue.ExtMin(); S = S ∪ {u};           O(|U| · lg |U|)
6       for (Node v in Adj[u])...{
7           Relax(u,v,w);                         O(|H| · lg |U|)
8   } } }

```

Možné ještě  **$O(|U| \cdot \lg |U| + |H|)$**  nebo  **$O(|U|^{**2})$**   
 (podle způsobu implementace prioritní fronty – Fibonacci/seznam)

**Co se stane, když existují záporně ohodnocené hrany?**

Relax se musí doplnit o **vracení uzlů do prioritní fronty**  
 (?? ukončení algoritmu, časová složitost ??)

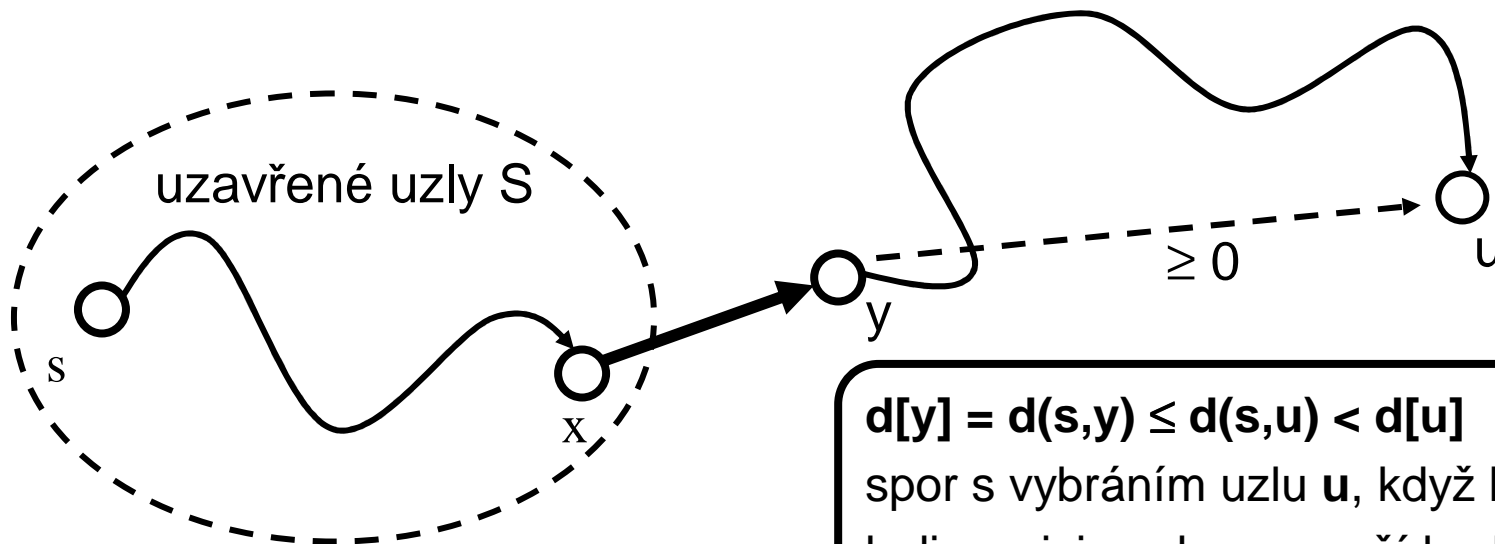


## Důkaz správnosti Dijkstrova algoritmu

**Tvrzení:** Při uzavření uzlu  $u$  (řádka 5 ...  $S = S \cup \{u\}$ ) platí

$$d[u] = d(s, u) \quad (\text{index } w \text{ už nepíšeme})$$

**D:** sporem - nechť je  $d[u] > d(s, u)$  pro nějaký uzavřený uzel, mějme nejkratší cestu  $s \rightarrow u$ ,  $x$  je poslední uzavřený uzel



$$d[y] = d(s, y) \leq d(s, u) < d[u]$$

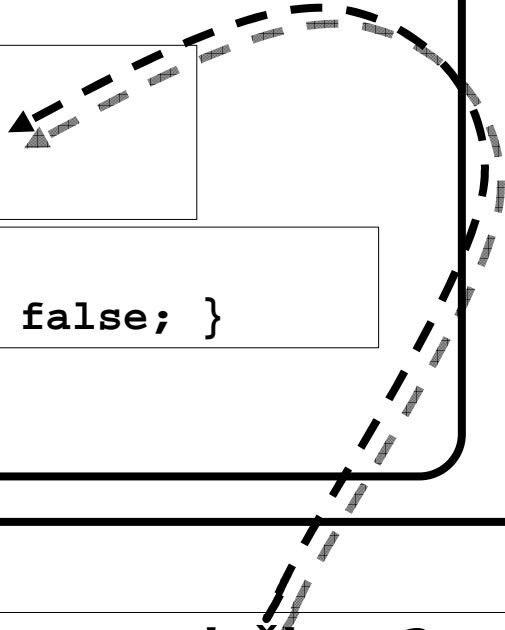
spor s vybráním uzlu  $u$ , když byl k dispozici uzel  $y$  s menší hodnotou

# Bellmanův-Fordův algoritmus

**? Co dělat v případě záporně ohodnocených hran ?**

**Relaxovat, relaxovat, relaxovat ...**

```
boolean Bellman-Ford (Graph G, Node s, Weights w) {  
1   InitPaths(G,s);  
2   for (int i=1; i<|U|; i++)  
3       for (Edge (u,v) in H)  
4           Relax(u,v,w);  
5   for (Edge (u,v) in H) {  
6       if (d[v] > d[u] + w(u,v)) return false; }  
7   return true;  
8 }
```



Složitost  **$O(|U| \cdot |H|)$**

**?Proč má nyní Relax konstantní časovou složitost?**

## ? Nelze B-F algoritmus nějak upravit / zrychlit ?

Co když **zavedeme frontu uzlů s úspěšným Relax** a bereme jen hrany vycházející z těchto uzlů? (a máme Dijkstru!)

- **ukončení** - při vyprázdnění fronty
- **problém** - co když se fronta nevyprázdní?
- v nejhorším případě zase  $O(|U| \cdot |H|)$

### DAG-Paths - nejkratší cesty pro acyklické grafy

```
1  "Topologicky uspořádáme uzly grafu G"
2  InitPaths(G,s);
3  for (Node u in U(G) v pořadí top. uspořádání) {
4      for (Node v in Adj[u]) Relax(u,v,w);
5  }
```

**?? Složitost ??  $O(|H|+|U|)$  !!**

## Kontrolní otázky

- 7.1** Která část Dijkstrova algoritmu je podstatně závislá na předpokladu nezáporného ohodnocení hran? Ukažte na jednom příkladu, že pro záporně ohodnocené hrany může Dijkstrův algoritmus dát špatný výsledek, a na jiném příkladu, že může dát správný výsledek.
- 7.2** Je možné prohlásit, že Dijkstrův algoritmus bude fungovat správně i při záporném ohodnocení hran, pokud bude zadaný graf acyklický?
- 7.3** Je možné prohlásit, že Dijkstrův algoritmus bude fungovat správně i při záporném ohodnocení hran, pokud bude použit k určení vzdáleností z kořene do ostatních uzlů kořenového stromu?
- 7.4** Navrhněte časově efektivní algoritmus pro určení celkového počtu různých orientovaných cest v acyklickém grafu.  
(Návod: Inspirujte se algoritmem DAG-Paths a za hodnotu  $d[u]$  berte počet cest končících v uzlu  $u$ .)
- 7.5** Navrhněte algoritmus, který určí vzdálenost ze všech uzlů do uzlu  $s$  v acyklickém orientovaném grafu. Určete potřebné datové struktury a časovou složitost navrženého algoritmu.
- 7.6** Navrhněte algoritmus lineární složitosti pro hledání nejdelších cest z daného uzlu do všech ostatních uzlů v acyklickém grafu.
- 7.7** Doplňte Dijkstrův a Bellman-Fordův algoritmus o výpočet hodnoty  $r[u]$ , která představuje počet hran nejkratší cesty z uzlu  $s$  do uzlu  $u$ .  
(Návod: Stačí vhodně upravit operace InitPaths a Relax.)

## Kontrolní otázky

**7.8 Navrhněte algoritmus, který v orientovaném grafu s nezáporným ohodnocením hran  $w$  nalezne druhou nejkratší orientovanou cestu z uzlu  $u$  do uzlu  $v$ . Určete asymptotickou časovou složitost tohoto algoritmu.**