Pokročilá algoritmizace

Daniel Průša, Ondřej Drbohlav

2025

Join at slido.com
#3298216

The Slido app must be installed on every computer you're presenting from

Co na to říkáte?

The Slido app must be installed on every computer you’re presenting from

Audience Q&A

The Slido app must be installed on every computer you're presenting from

Asymptotické odhady

- horní asymptotický odhad (velké omikron odhad):

$$f(n) \in \mathrm{O}(g(n))$$

- význam:

  f je shora asymptoticky ohraničená funkcí g (až na multiplikativní konstantu)

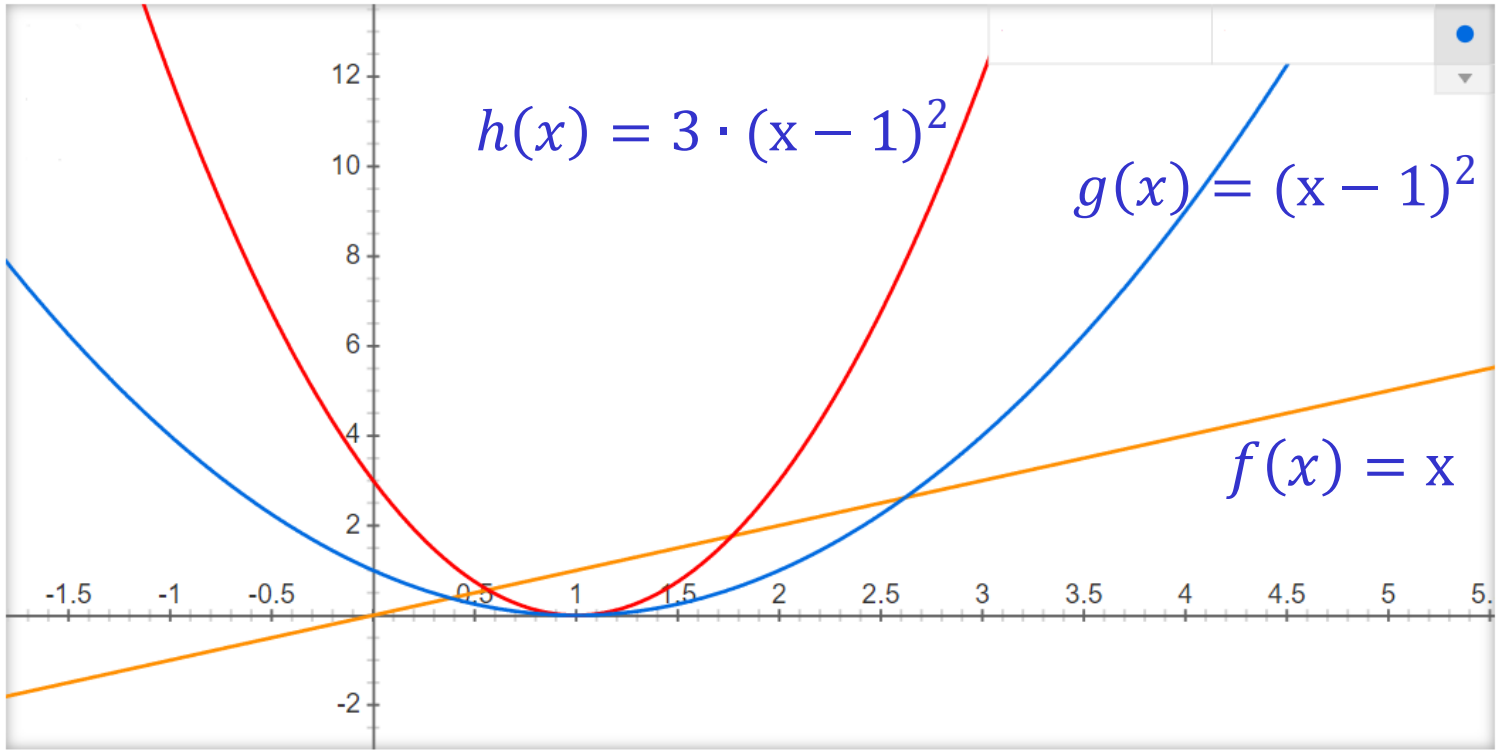
- definice:

$$(\exists c > 0)(\exists n_0)(\forall n > n_0) : \; f(n) \leq c \cdot g(n)$$

kde $c \in \mathbb{R}^{>0} \quad n_0, n \in \mathbb{N} \quad f, g \in \mathbb{N} \to \mathbb{R}^{\geq 0}$

Asymptotické odhady

- příklad $f(x) \in O(g(n)), \quad h(x) \in O(g(n))$



$$(\exists c > 0)(\exists n_0)(\forall n > n_0) : \ f(n) \leq c \cdot g(n)$$

Asymptotické odhady

- horní asymptotický odhad pro více proměnných:

$$f(n_1, \cdots, n_k) \in \mathrm{O}(g(n_1, \cdots, n_k))$$

- definice:

$$(\exists c > 0)(\exists n_0)(\forall n_1 > n_0) \cdots (\forall n_k > n_0) :$$

$$f(n_1, \cdots, n_k) \leq c \cdot g(n_1, \cdots, n_k)$$

kde $c \in \mathbb{R}^{>0} \quad n_0, n_1, \cdots, n_k \in \mathbb{N} \quad f, g \in \mathbb{N} \to \mathbb{R}^{\geq 0}$

Asymptotické odhady

- dolní asymptotický odhad (velké omega odhad):

$$f(n) \in \Omega(g(n))$$

- význam:

  f je zdola asymptoticky ohraničená funkcí g (až na konstantu)

- definice:

$$(\exists c > 0)(\exists n_0)(\forall n > n_0) : \; c \cdot g(n) \leq f(n)$$

kde $c \in \mathbb{R}^{>0} \quad n_0, n \in \mathbb{N} \quad f, g \in \mathbb{N} \to \mathbb{R}^{\geq 0}$

Asymptotické odhady

- optimální asymptotický odhad (velké théta odhad):

$$f(n) \in \Theta(g(n))$$

- význam:

  f je asymptoticky ohraničená funkcí g z obou stran (až na konstantu)

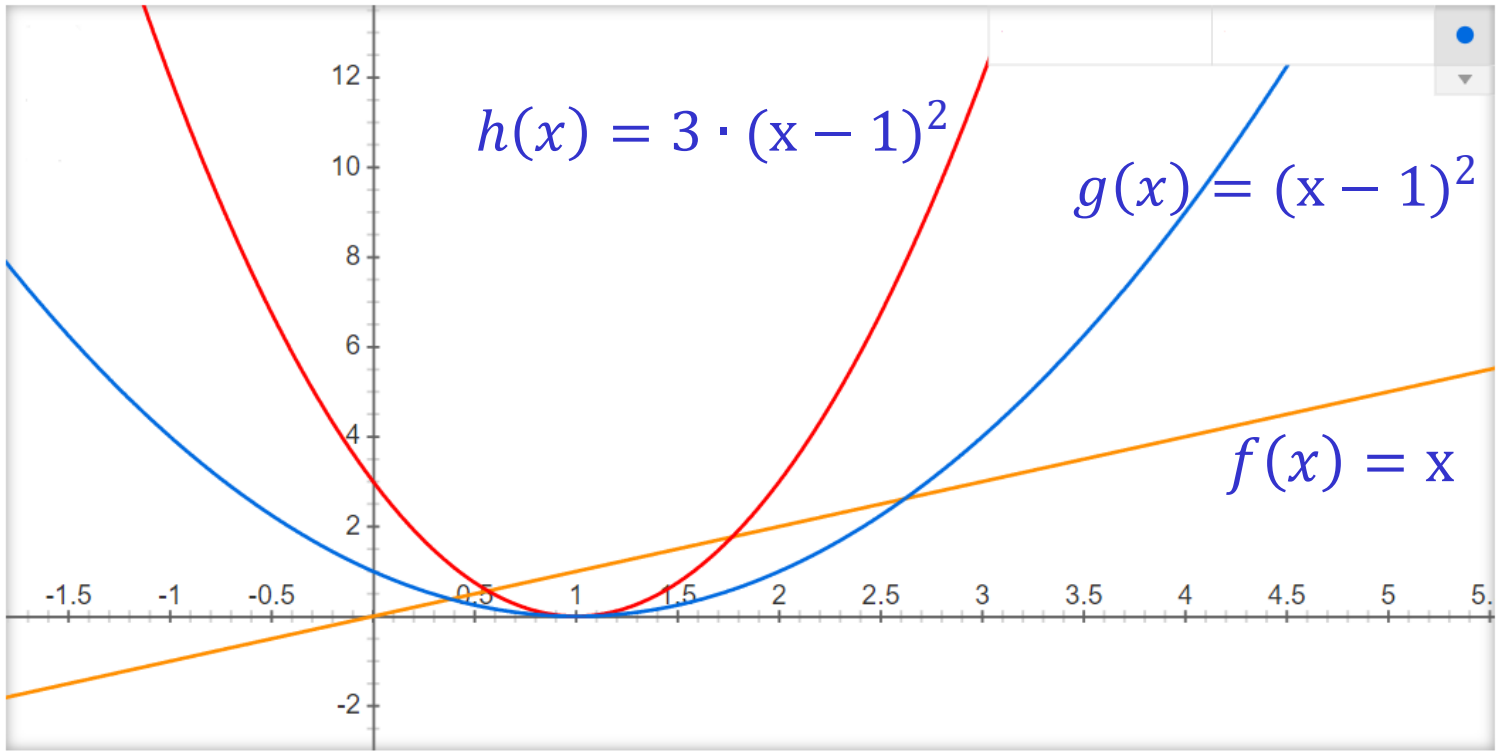
- definice: $\Theta(g(n)) \stackrel{\text{def}}{=} O(g(n)) \cap \Omega(g(n))$
- nebo alternativně:

$$(\exists c_1, c_2 > 0)(\exists n_0)(\forall n > n_0): c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

kde $c_1, c_2 \in \mathbb{R}^{>0}$ $n_0, n \in \mathbb{N}$ $f, g \in \mathbb{N} \to \mathbb{R}^{\geq 0}$

Asymptotické odhady

- příklad $g(n) \in \Theta(h(n)), \quad g(n) \notin \Theta(f(n))$



Asymptotické odhady

- Platí $\log_a(n) \in \Theta(\log_b(n))$

$$\log_a n = \frac{\log_b n}{\log_b a} = \frac{1}{\log_b a} \cdot \log_b n$$

vzoreček konstanta

Vlastnosti asymptotických odhadů

- Věta: Jsou-li funkce $f(n)$, $g(n)$ vždy kladné, pak pro limitu v ∞ platí
  - $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, pak $f(n) \in O(g(n))$, ale **neplatí** $f(n) \in \Theta(g(n))$
  - $\lim_{n\to\infty} \frac{f(n)}{g(n)} = a$, kde $0 < a < \infty$, pak $f(n) \in \Theta(g(n))$
  - $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, pak $g(n) \in O(f(n))$, ale **neplatí** $g(n) \in \Theta(f(n))$
- Důsledek: Mějme pevně zvolené číslo $k \in \mathbb{N}$, pak platí

$$(\log(n))^k \in O(n)$$

- Důkaz lze provést pomocí L'Hopitalova pravidla.

Audience Q&A

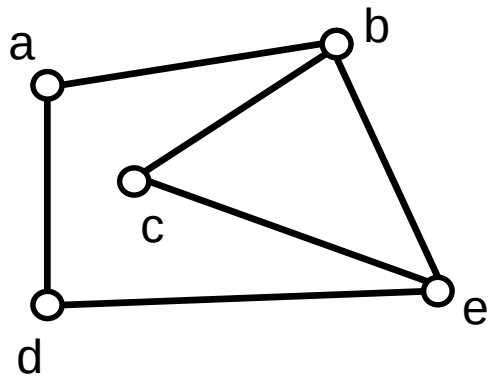
The Slido app must be installed on every computer you're presenting from

(Ne)orientovaný graf

$G_1 = (V, E_1)$

$V = \{a, b, c, d, \mathrm{e}\}$

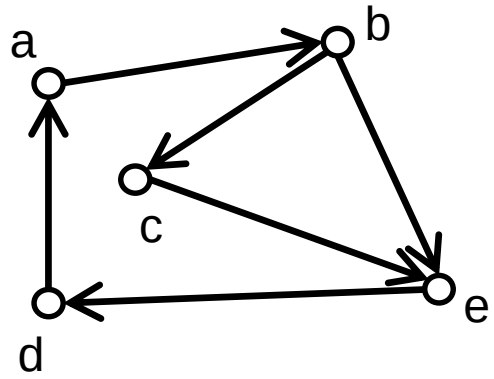
$E_1 = \begin{Bmatrix} \{a,b\}, \{b,c\}, \{b,e\}, \\ \{e,c\}, \{d,e\}, \{a,d\} \end{Bmatrix}$



$G_2 = (V, E_2)$

$V = \{a, b, c, d, \mathrm{e}\}$

$E_2 = \begin{Bmatrix} (a,b), (b,c), (b,e), \\ (c,e), (e,d), (d,a) \end{Bmatrix}$



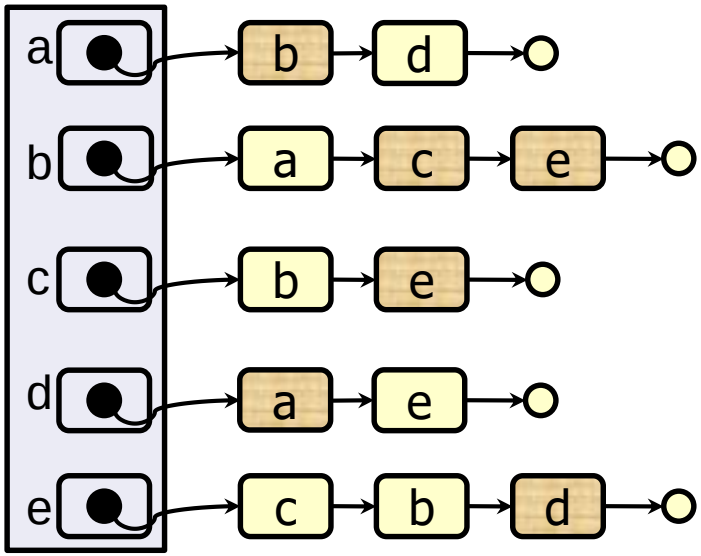
Znáte způsoby reprezentace grafu v paměti a průchody grafem?

Reprezentace grafu v paměti

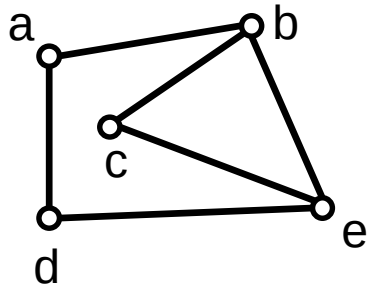
Matice sousednosti:

	a	b	c	d	e
a	0	**1**	0	**1**	0
b	**1**	0	**1**	0	**1**
c	0	**1**	0	0	**1**
d	**1**	0	0	0	**1**
e	0	**1**	**1**	**1**	0

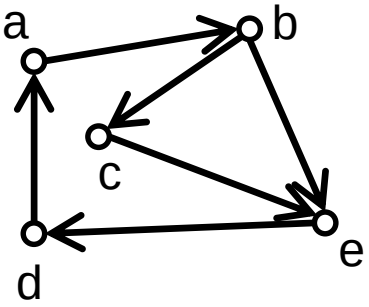
Seznam sousedů:



$G_1 = (V, E_1)$



$G_2 = (V, E_2)$



Reprezentace grafu G_2 obsahuje pouze prvky s pozadím 

Prostorová složitost

$\Theta(|V|^2)$

test existence hrany v konstantním čase

$\Theta(|V| + |E|)$

výhodnější pro řídké grafy

Audience Q&A

The Slido app must be installed on every computer you're presenting from

DFS - algoritmus

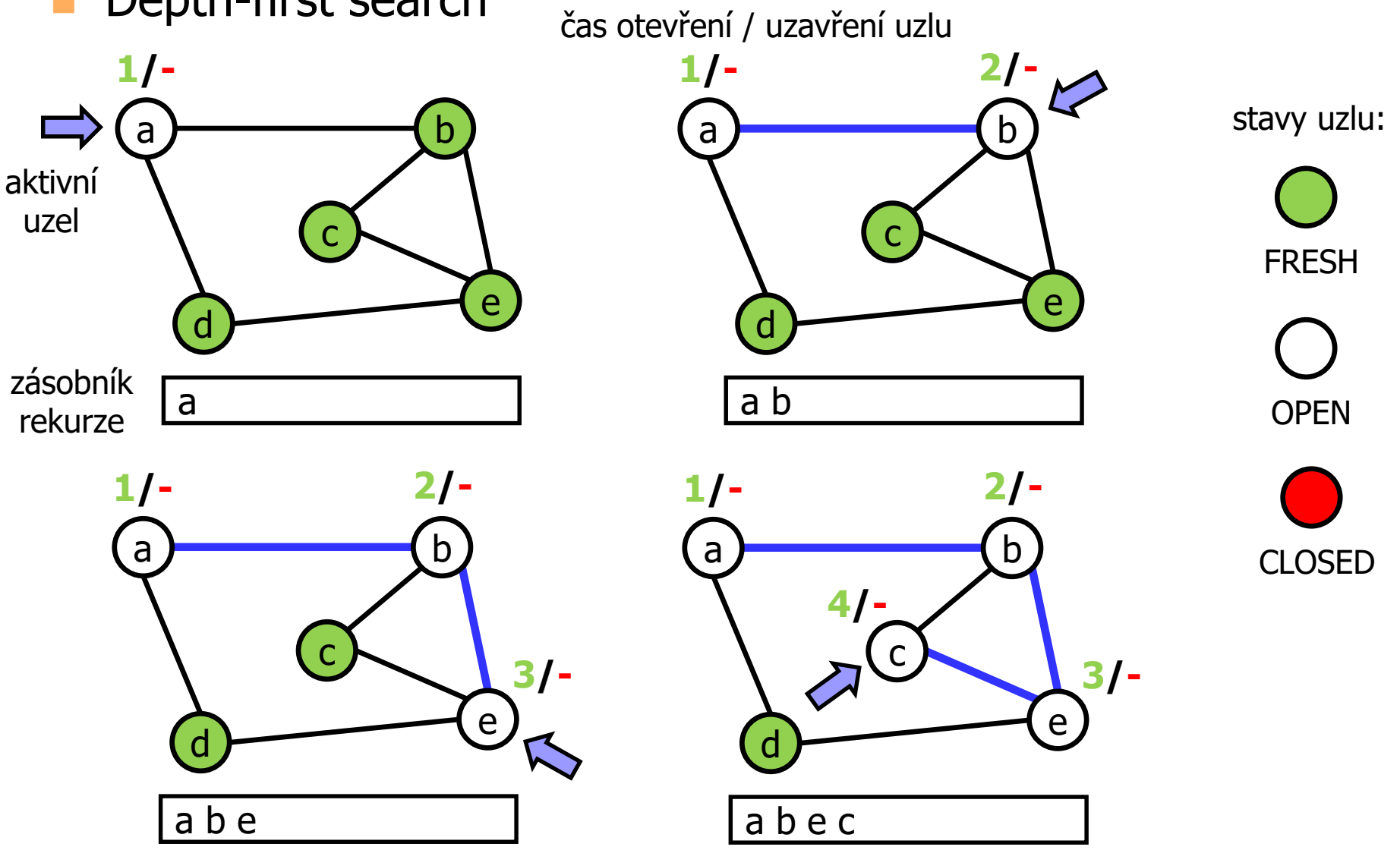
```
procedure dfs(start_vertex : Vertex)
var     to_visit : Stack = empty;
        visited : Vertices = empty;
{
        to_visit.push(start_vertex);
        while (size(to_visit) != 0) {
                v = to_visit.pop();
                if v not in visited then {
                        visited.add(v);
                        for all x in neighbors of v {
                                to_visit.push(x);
                        }
                }
        }
}
```

BFS - algoritmus

```
procedure bfs(start_vertex : Vertex)
var     to_visit :Queue= empty;
        visited : Vertices = empty;
{
        to_visit.push(start_vertex);
        while (size(to_visit) != 0) {
                v = to_visit.pop();
                if v not in visited then {
                        visited.add(v);
                        for all x in neighbors of v {
                                to_visit.push(x);
                        }
                }
        }
}
```

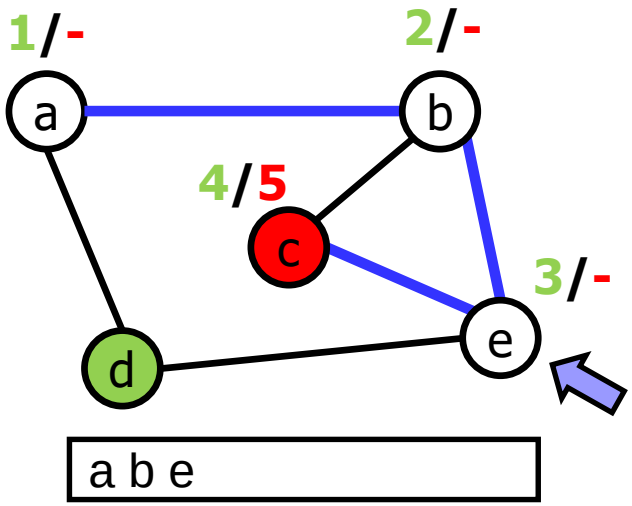
Průchod grafem do hloubky (DFS)

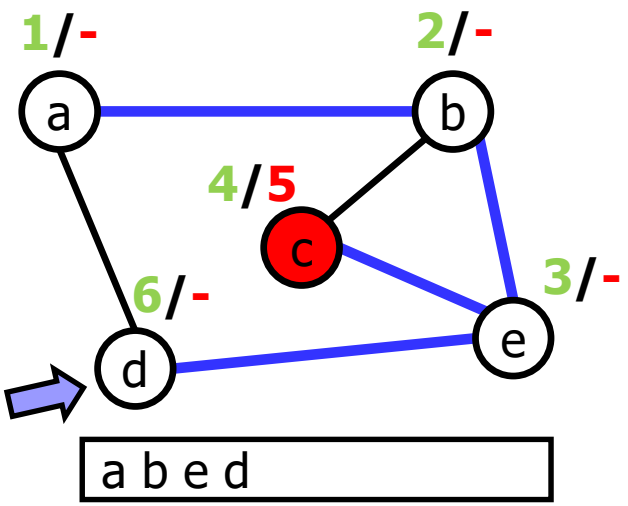
- Depth-first search

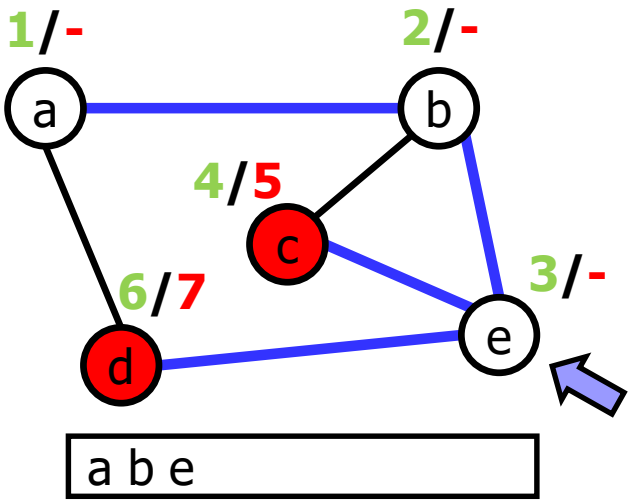


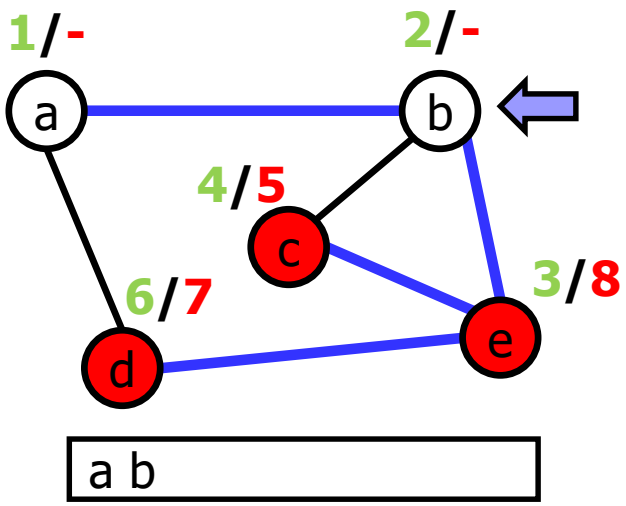
Průchod grafem do hloubky (DFS)

- Depth-first search









stavy uzlu:

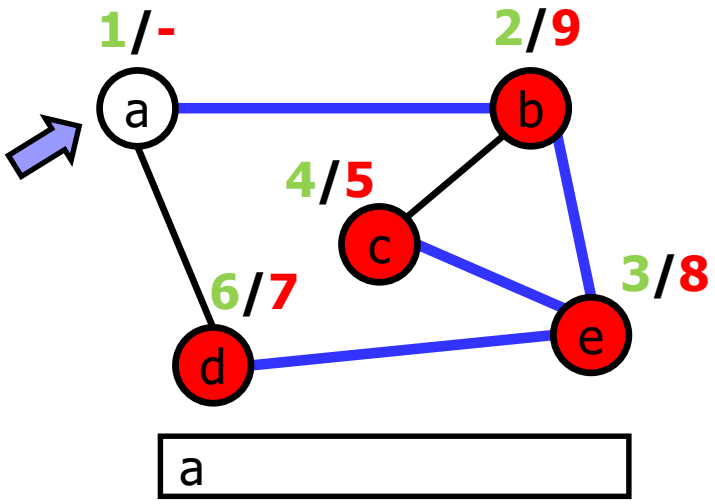
FRESH

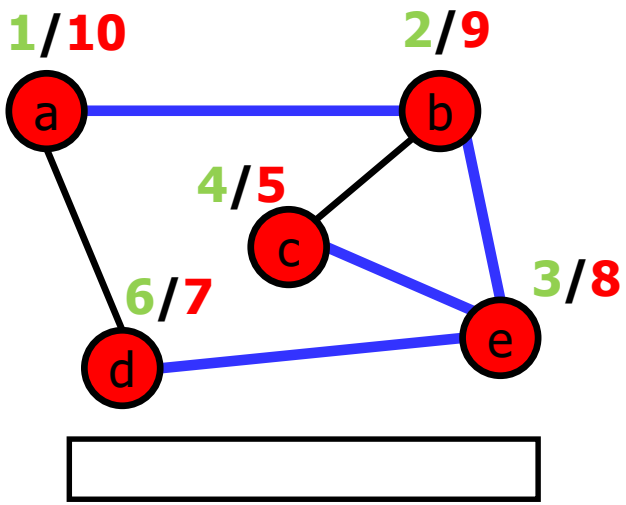
OPEN

CLOSED

Průchod grafem do hloubky (DFS)

- Depth-first search





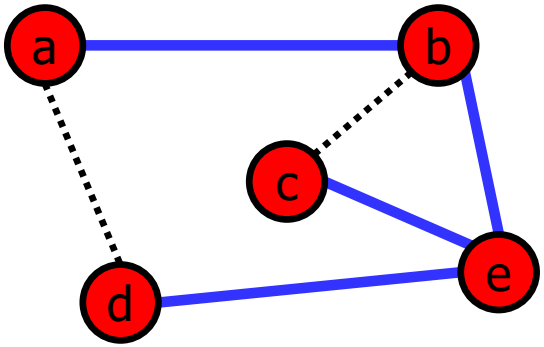
stavy uzlu:

FRESH

OPEN

CLOSED

DFS-strom:

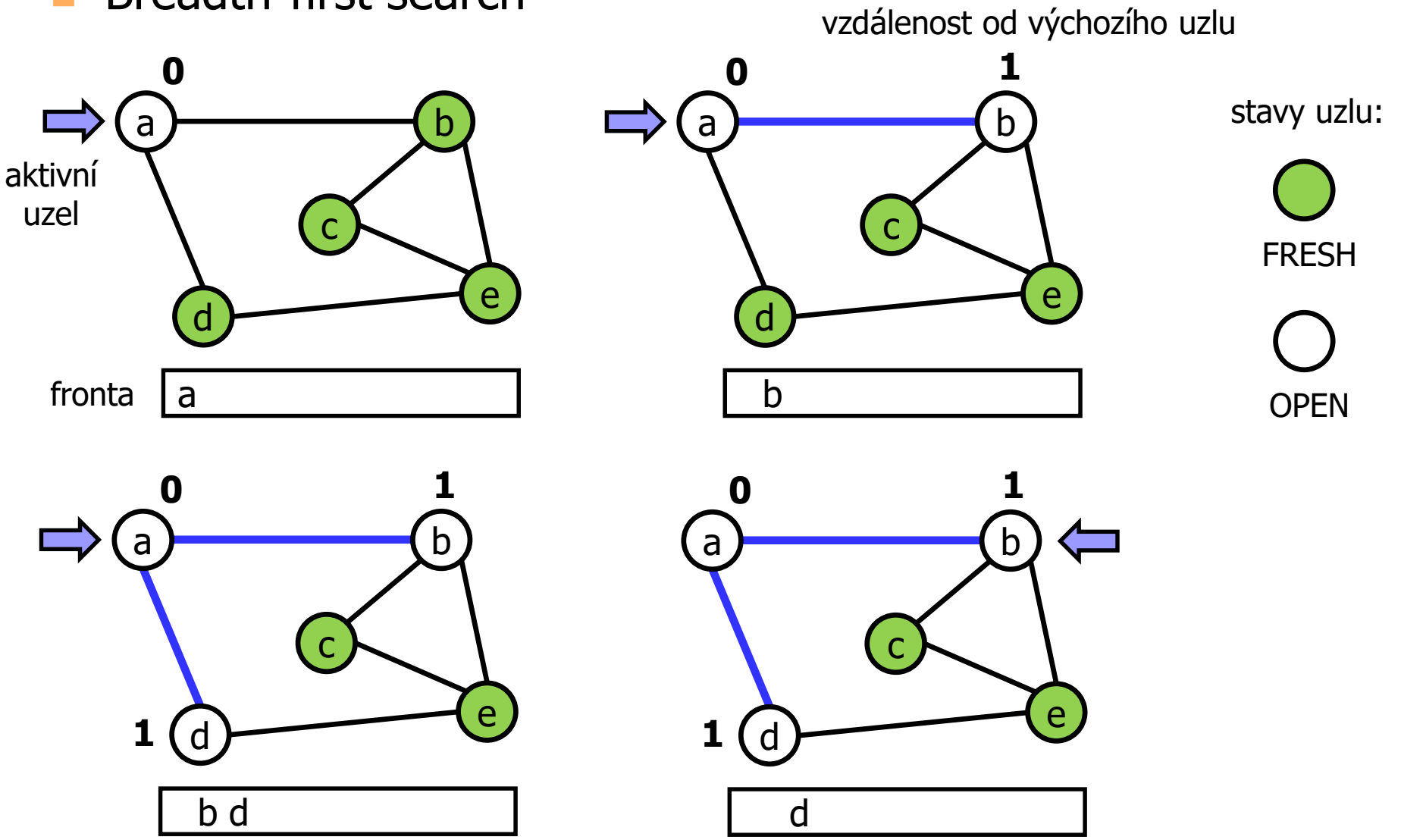


Audience Q&A

The Slido app must be installed on every computer you're presenting from

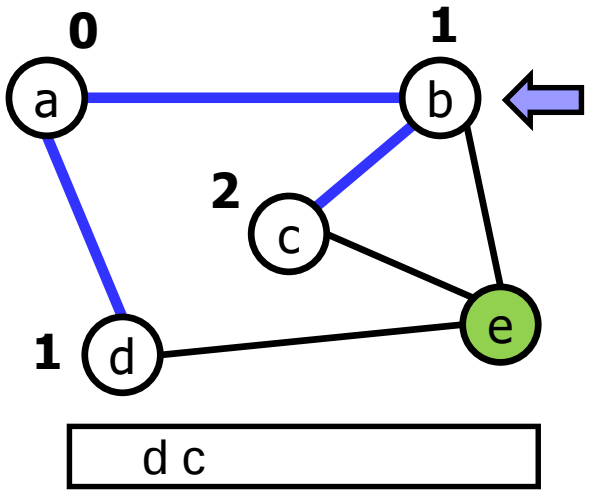
Průchod grafem do šířky (BFS)

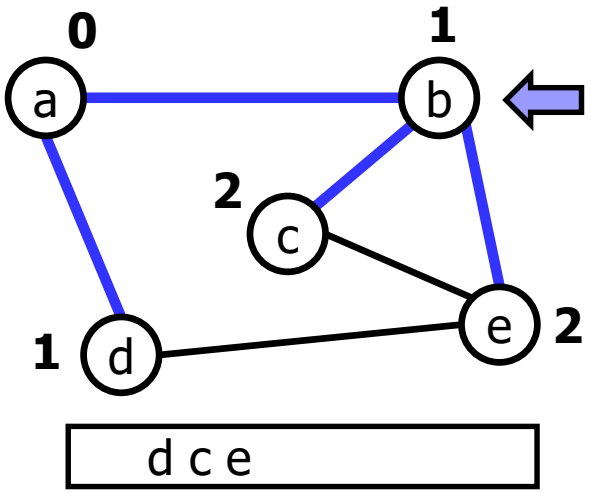
- Breadth-first search



Průchod grafem do šířky (BFS)

- Breadth-first search

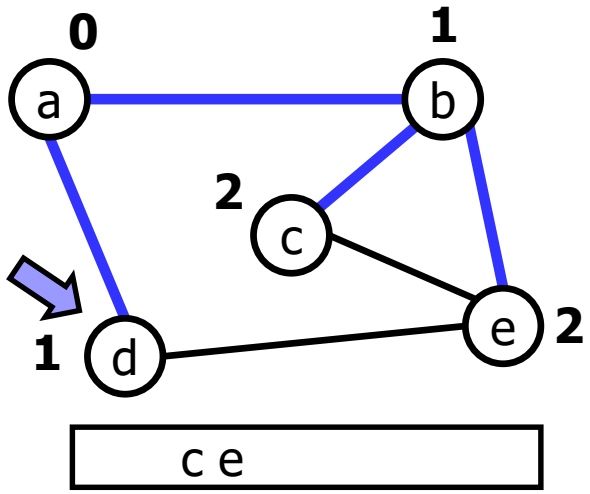


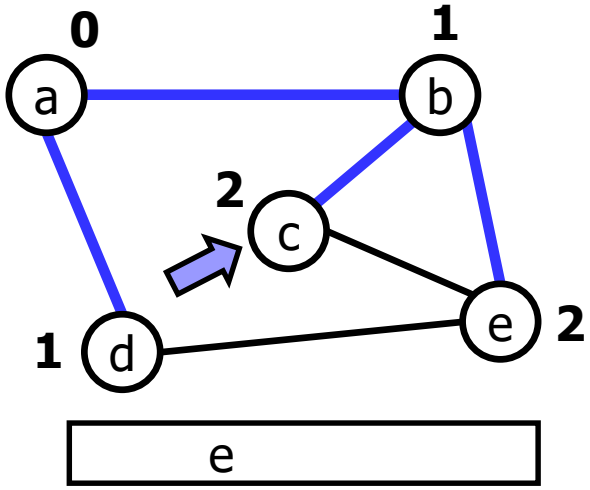


stavy uzlu:

FRESH

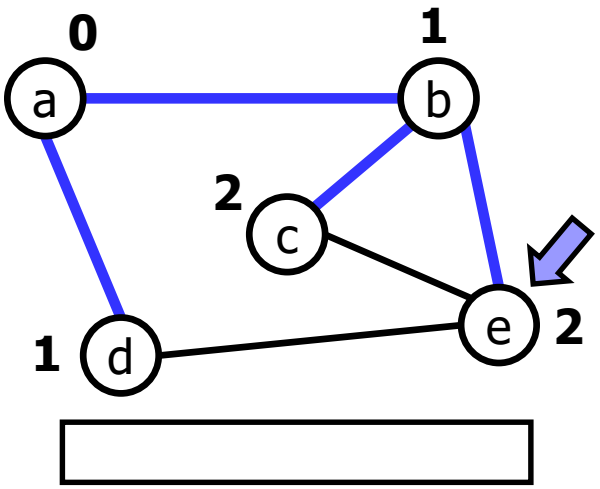
OPEN





Průchod grafem do šířky (BFS)

- Breadth-first search

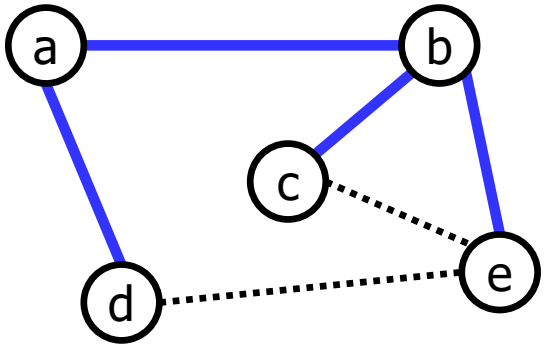


stavy uzlu:

FRESH

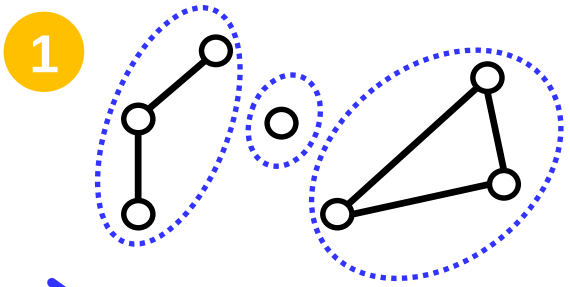
OPEN

BFS-strom:

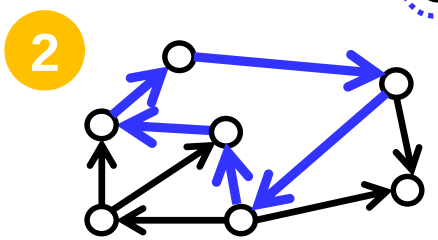


Aplikace průchodu grafem

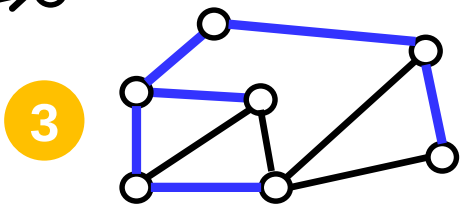
1. Detekce komponent souvislosti



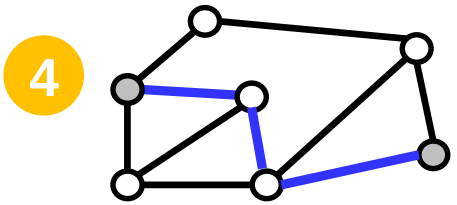
2. Detekce cyklu (jen DFS)
   (pokud při DFS objevíme uzel ve stavu OPEN => cyklus)



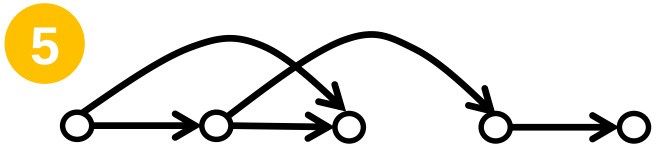
3. Nalezení kostry



4. Hranově nejkratší cesta (jen BFS)

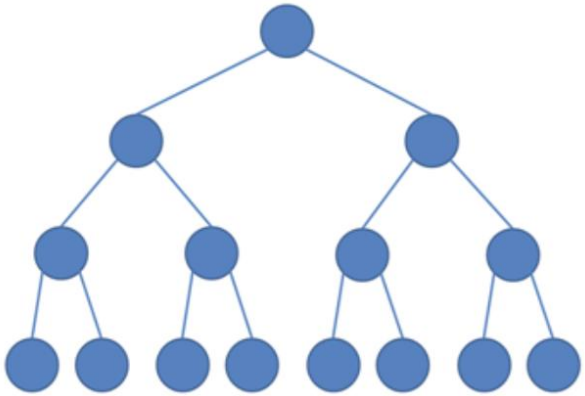


5. Topologické uspořádání (jen DFS)
   (uzly uspořádáme sestupně podle časů jejich uzavření)



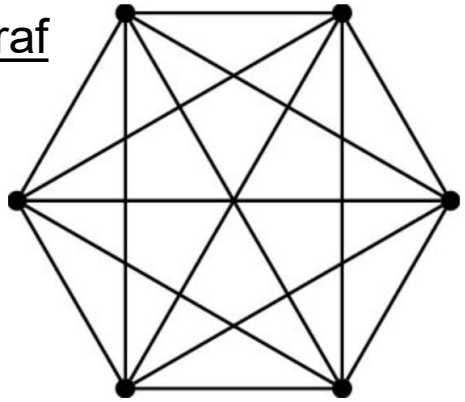
DFS vs. BFS z hlediska paměti

vyvážený strom



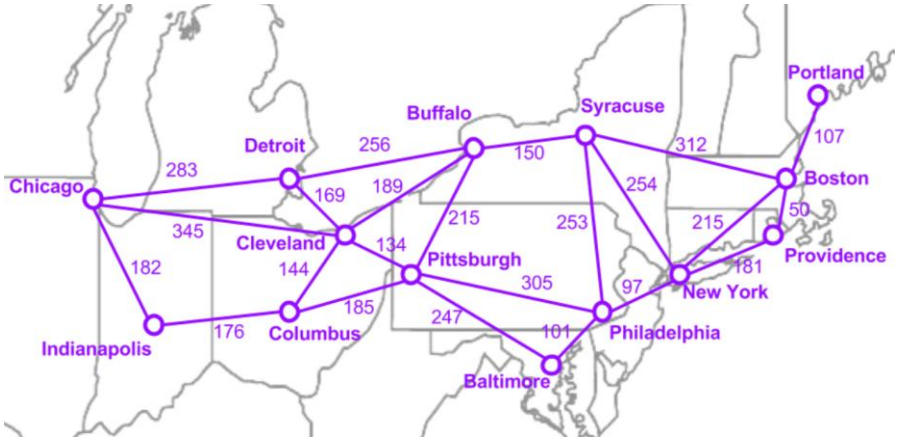
BFS: $\Theta(n)$ DFS: $\Theta(\log n)$

úplný graf



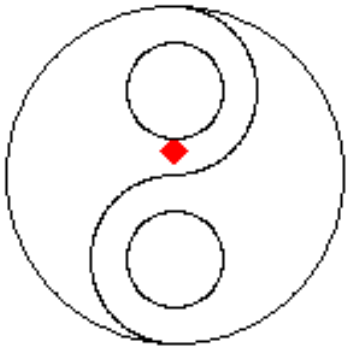
BFS: $\Theta(n)$ DFS: $\Theta(n)$

(řídký) rovinný graf

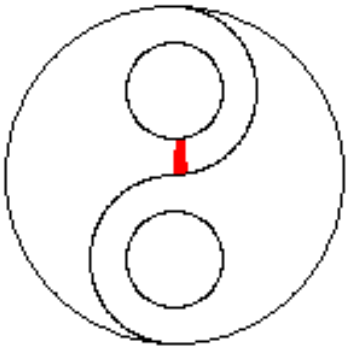


Flood fill algoritmus

BFS



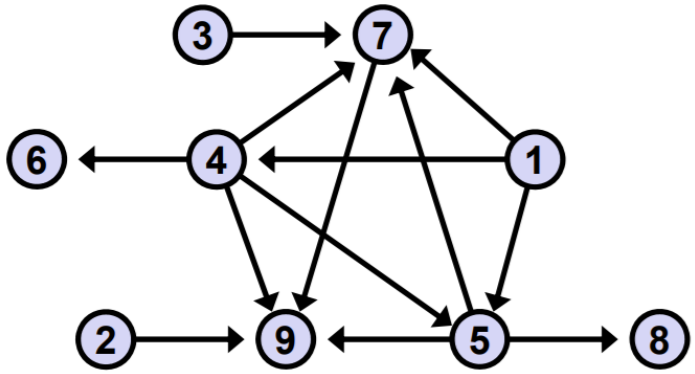
DFS

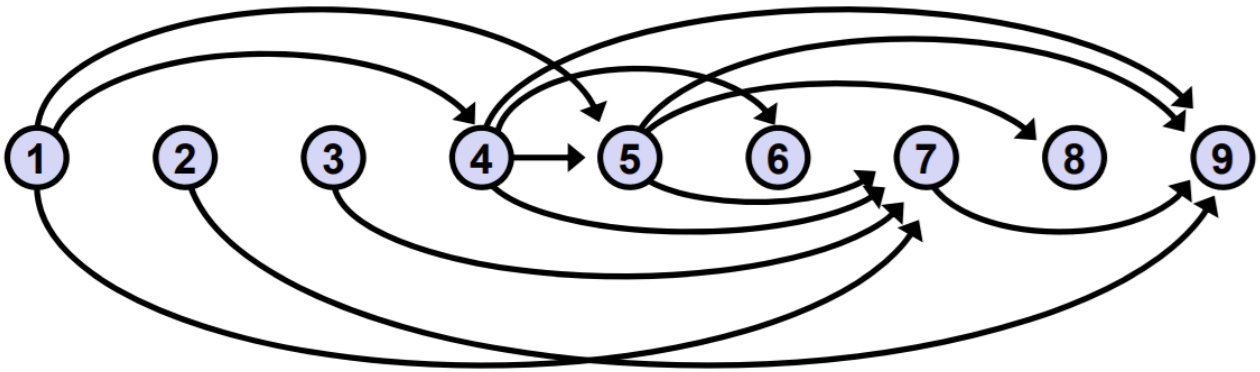


Audience Q&A

The Slido app must be installed on every computer you're presenting from

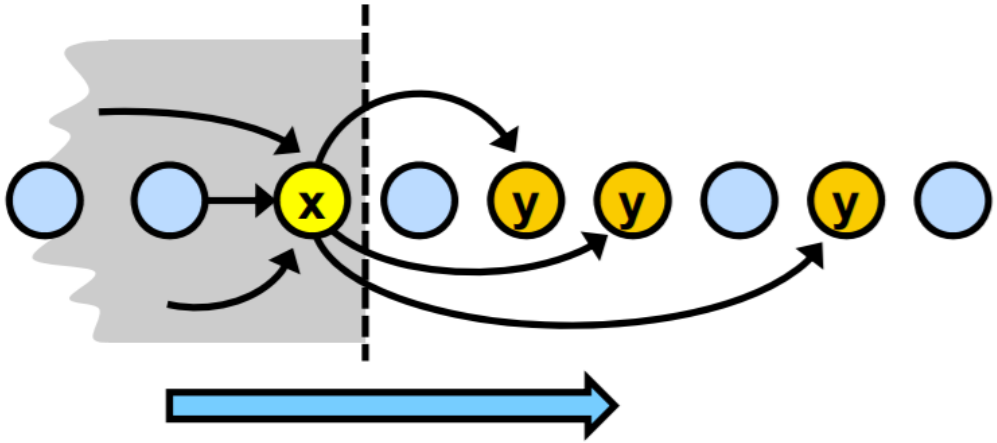
DAG, topologické uspořádání





Nejdelší cesta v DAG

```
for each node (x) in V(G) {
  for each edge e = (x, y) in E(G)
    if (y. dist < x.dist + e.weight) {
      y. dist = x.dist + e.weight
      y.pred = x;
    }
}
```



Audience Q&A

The Slido app must be installed on every computer you're presenting from