DELTA TopGun

(13) Hledání nejkratší cesty

Luboš Zápotočný, Tomáš Faltejsek, Michal Havelka

2023

Obsah

BFS - průchod grafu do šířky - opakování

Algoritmus prochází graf ze specifického vrcholu a označuje vrcholy, do kterých algoritmus vstoupí v následujících krocích

BFS - průchod grafu do šířky - opakování

Algoritmus prochází graf ze specifického vrcholu a označuje vrcholy, do kterých algoritmus vstoupí v následujících krocích

BFS používá datovou strukturu frontu, do které ukládá postupně sousedy (kteří ještě nejsou zafrontováni) aktuálního vrcholu

BFS - průchod grafu do šířky - opakování

Algoritmus prochází graf ze specifického vrcholu a označuje vrcholy, do kterých algoritmus vstoupí v následujících krocích

BFS používá datovou strukturu frontu, do které ukládá postupně sousedy (kteří ještě nejsou zafrontováni) aktuálního vrcholu

Po dokončení přidání sousedů se algoritmus vydá do vrcholu, který je uložen na začátku fronty

BFS - průchod grafu do šířky - opakování

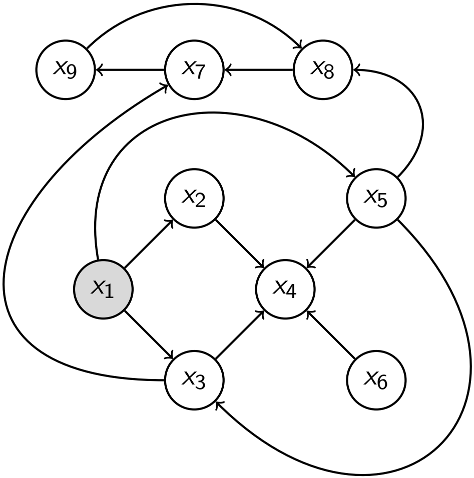
Algoritmus prochází graf ze specifického vrcholu a označuje vrcholy, do kterých algoritmus vstoupí v následujících krocích

BFS používá datovou strukturu frontu, do které ukládá postupně sousedy (kteří ještě nejsou zafrontováni) aktuálního vrcholu

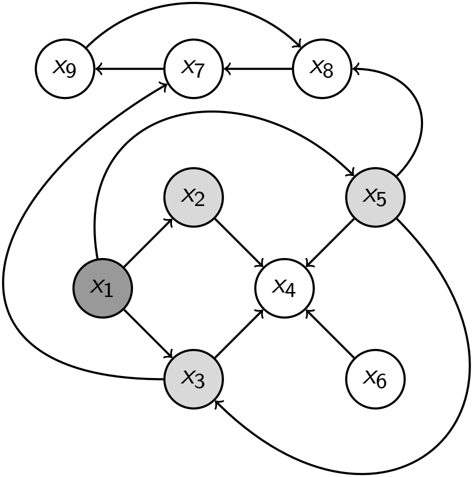
Po dokončení přidání sousedů se algoritmus vydá do vrcholu, který je uložen na začátku fronty

Výběr souseda, ve kterém algoritmus bude pokračovat, je jasně daný pořadím ve frontě

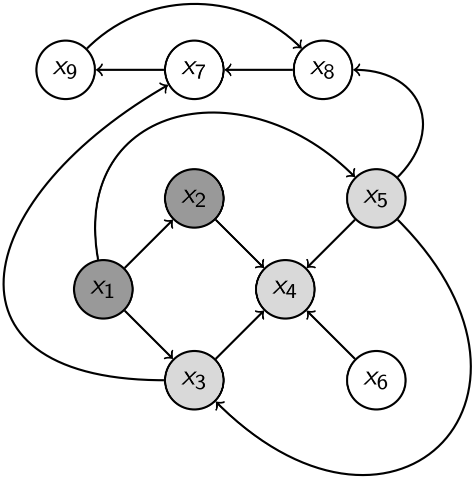
BFS - ukázka na grafu



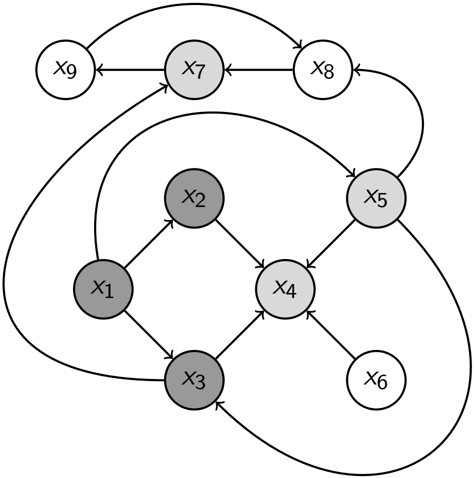
BFS - ukázka na grafu



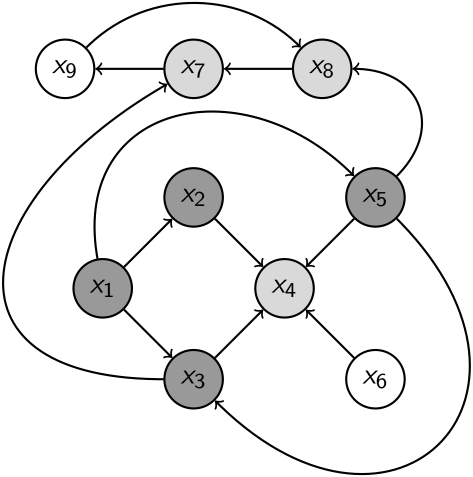
BFS - ukázka na grafu



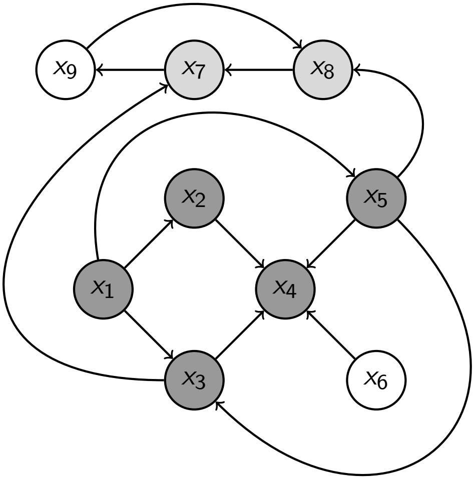
BFS - ukázka na grafu



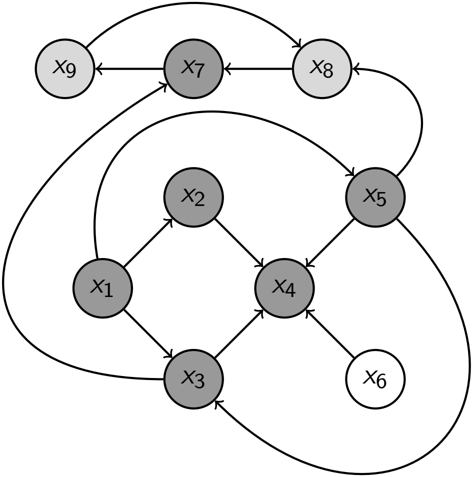
BFS - ukázka na grafu



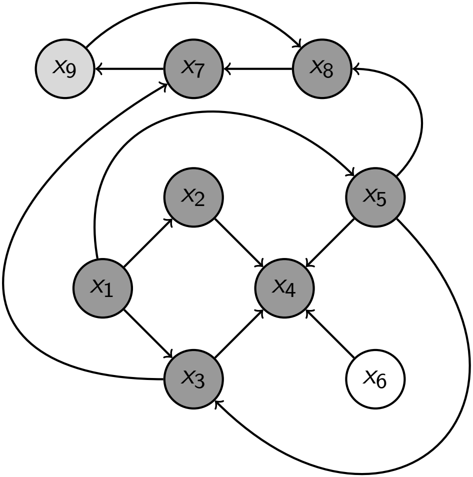
BFS - ukázka na grafu



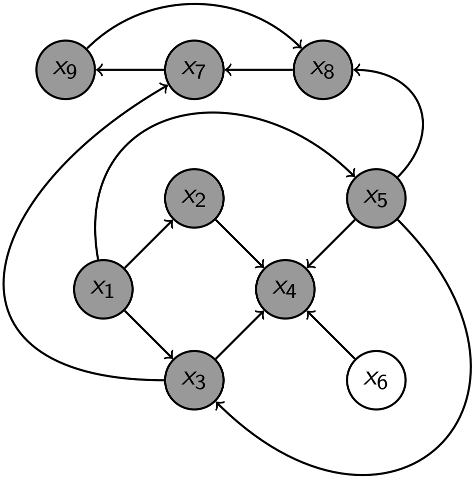
BFS - ukázka na grafu



BFS - ukázka na grafu



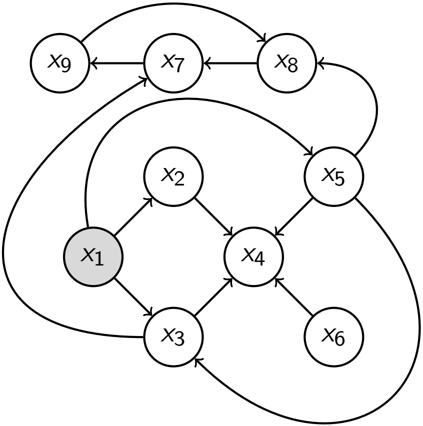
BFS - ukázka na grafu



BFS - pole předchůdců

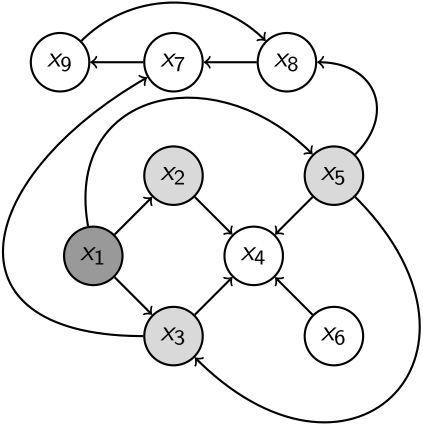
Při průchodu BFS si udržujeme záznam o předchůdci

BFS - pole předchůdců



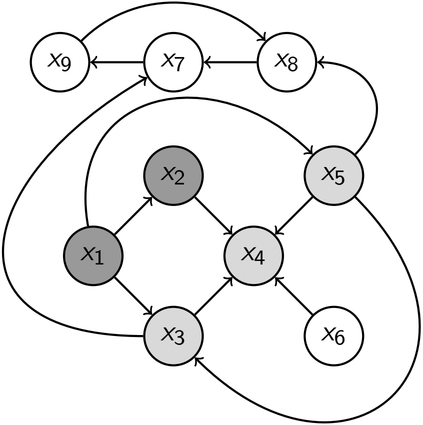
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

BFS - pole předchůdců



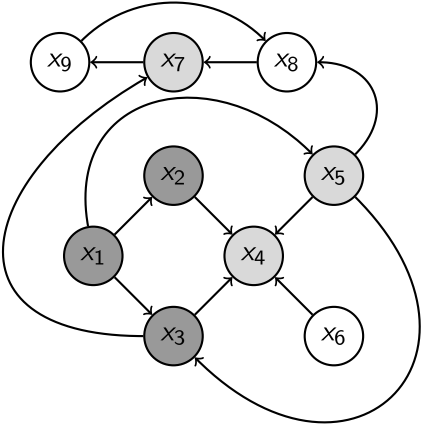
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\emptyset$	x_1	x_1	$\emptyset$	x_1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

BFS - pole předchůdců



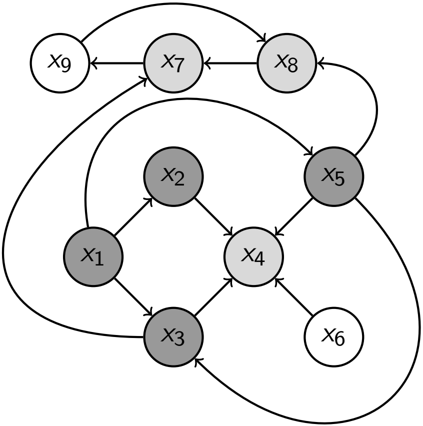
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\emptyset$	x_1	x_1	x_2	x_1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

BFS - pole předchůdců



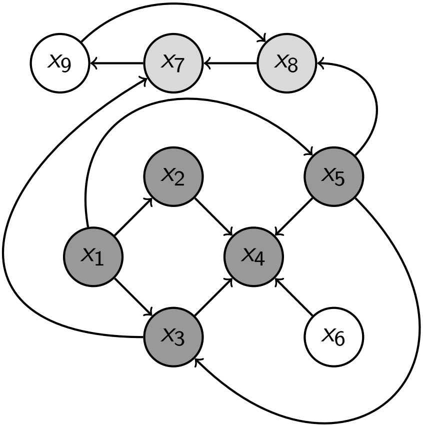
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\emptyset$	x_1	x_1	x_2	x_1	$\emptyset$	x_3	$\emptyset$	$\emptyset$

BFS - pole předchůdců



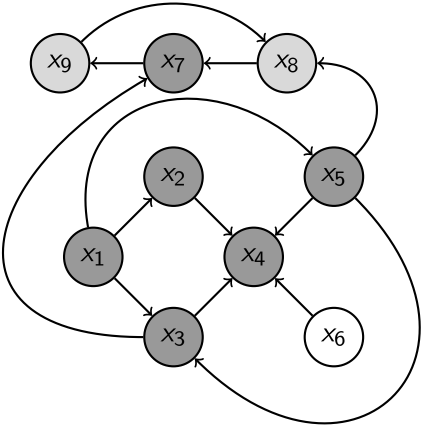
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\emptyset$	x_1	x_1	x_2	x_1	$\emptyset$	x_3	x_5	$\emptyset$

BFS - pole předchůdců



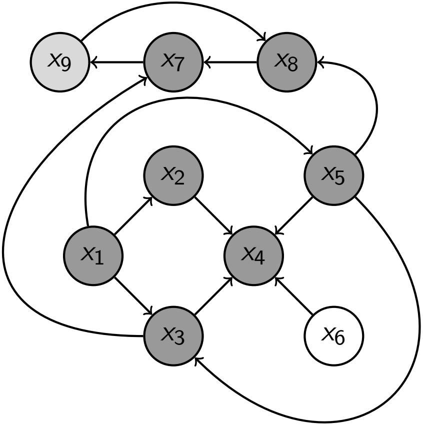
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\emptyset$	x_1	x_1	x_2	x_1	$\emptyset$	x_3	x_5	$\emptyset$

BFS - pole předchůdců



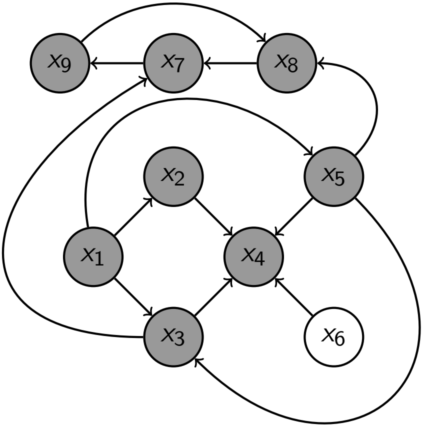
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\emptyset$	x_1	x_1	x_2	x_1	$\emptyset$	x_3	x_5	x_7

BFS - pole předchůdců



x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\emptyset$	x_1	x_1	x_2	x_1	$\emptyset$	x_3	x_5	x_7

BFS - pole předchůdců



x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\emptyset$	x_1	x_1	x_2	x_1	$\emptyset$	x_3	x_5	x_7

BFS - pole předchůdců

Ukázka na tabuli

Přerozdělení ohodnocené hrany

Mějme hranu v grafu, která je ohodnocená přirozeným číslem $\mathbb{N} = \{1, 2, 3, \ldots\}$

$$x_1 \xrightarrow{5} x_2$$

Přerozdělení ohodnocené hrany

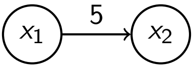
Mějme hranu v grafu, která je ohodnocená přirozeným číslem $\mathbb{N} = \{1, 2, 3, \ldots\}$

$$x_1 \xrightarrow{5} x_2$$

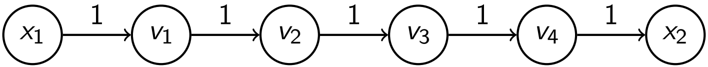
Hranu ohodnocenou váhou w můžeme přerozdělit pomocí $w - 1$ virtuálních vrcholů s hranami, které mají jednotkovou délku

Přerozdělení ohodnocené hrany

Mějme hranu v grafu, která je ohodnocená přirozeným číslem $\mathbb{N} = \{1, 2, 3, \ldots\}$



Hranu ohodnocenou váhou w můžeme přerozdělit pomocí $w - 1$ virtuálních vrcholů s hranami, které mají jednotkovou délku



Přerozdělení ohodnocené hrany

Mějme hranu v grafu, která je ohodnocená přirozeným číslem $\mathbb{N} = \{1, 2, 3, \ldots\}$

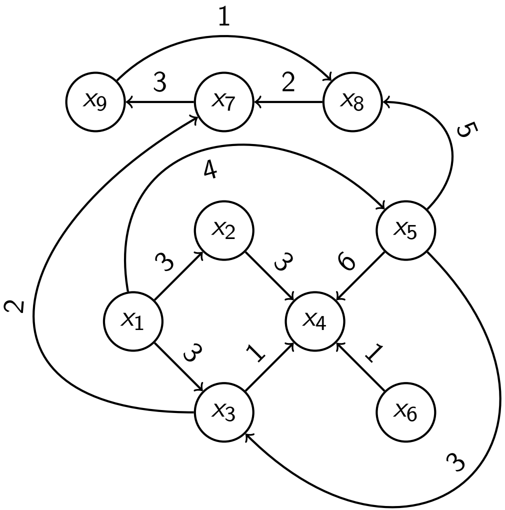
$$x_1 \xrightarrow{5} x_2$$

Hranu ohodnocenou váhou w můžeme přerozdělit pomocí $w - 1$ virtuálních vrcholů s hranami, které mají jednotkovou délku

$$x_1 \xrightarrow{1} v_1 \xrightarrow{1} v_2 \xrightarrow{1} v_3 \xrightarrow{1} v_4 \xrightarrow{1} x_2$$

V takto modifikovaném grafu je mezi vrcholem x_1 a x_2 stejná vzdálenost (váha) jako v grafu původním

Hledání nejkratší cesty



Hledání nejkratší cesty

Najděte pomocí předchozích algoritmů nejkratší cestu z x_1 do x_4

Hledání nejkratší cesty

Najděte pomocí předchozích algoritmů nejkratší cestu z x_5 do x_4

Hledání nejkratší cesty

Najděte pomocí předchozích algoritmů nejkratší cestu z x_1 do x_7

Hledání nejkratší cesty

Najděte pomocí předchozích algoritmů nejkratší cestu z x_1 do x_8

Časová složitost hledání nejkratší cesty pomocí BFS

Časová složitost BFS

Časová složitost hledání nejkratší cesty pomocí BFS

Časová složitost BFS

Záleží na metodě uložení grafu

Časová složitost hledání nejkratší cesty pomocí BFS

Časová složitost BFS

Záleží na metodě uložení grafu Při použití matice sousednosti je časová složitost BFS $\mathcal{O}(V^2)$

Časová složitost hledání nejkratší cesty pomocí BFS

Časová složitost BFS

Záleží na metodě uložení grafu Při použití matice sousednosti je časová složitost BFS $\mathcal{O}(V^2)$ Při použití seznamu sousedů je časová složitost BFS $\mathcal{O}(V + E)$

Časová složitost hledání nejkratší cesty pomocí BFS

Časová složitost BFS

Záleží na metodě uložení grafu Při použití matice sousednosti je časová složitost BFS $\mathcal{O}(V^2)$ Při použití seznamu sousedů je časová složitost BFS $\mathcal{O}(V + E)$

Největší váha hrany

Označme W největší váhu ze všech hran grafu G = (V, E)

Časová složitost hledání nejkratší cesty pomocí BFS

Časová složitost BFS

Záleží na metodě uložení grafu Při použití matice sousednosti je časová složitost BFS $\mathcal{O}(V^2)$ Při použití seznamu sousedů je časová složitost BFS $\mathcal{O}(V + E)$

Největší váha hrany

Označme W největší váhu ze všech hran grafu G = (V, E)
$W = max\{w(e)|e \in E\}$

Časová složitost hledání nejkratší cesty pomocí BFS

Časová složitost BFS

Záleží na metodě uložení grafu Při použití matice sousednosti je časová složitost BFS $\mathcal{O}(V^2)$ Při použití seznamu sousedů je časová složitost BFS $\mathcal{O}(V + E)$

Největší váha hrany

Označme W největší váhu ze všech hran grafu G = (V, E)
$W = max\{w(e)|e \in E\}$

Časová složitost hledání nejkratší cesty pomocí BFS

Časová složitost hledání nejkratší cesty pomocí BFS

Časová složitost BFS

Záleží na metodě uložení grafu Při použití matice sousednosti je časová složitost BFS $\mathcal{O}(V^2)$ Při použití seznamu sousedů je časová složitost BFS $\mathcal{O}(V + E)$

Největší váha hrany

Označme W největší váhu ze všech hran grafu G = (V, E)
$W = max\{w(e)|e \in E\}$

Časová složitost hledání nejkratší cesty pomocí BFS

Musíme nejdřív přerozdělit všechny hrany v grafu
Tato transformace vytvoří graf s $\mathcal{O}(W * V)$ vrcholy

Časová složitost hledání nejkratší cesty pomocí BFS

Časová složitost BFS

Záleží na metodě uložení grafu Při použití matice sousednosti je časová složitost BFS $\mathcal{O}(V^2)$ Při použití seznamu sousedů je časová složitost BFS $\mathcal{O}(V+E)$

Největší váha hrany

Označme W největší váhu ze všech hran grafu G = (V, E)
$W = max\{w(e)|e \in E\}$

Časová složitost hledání nejkratší cesty pomocí BFS

Musíme nejdřív přerozdělit všechny hrany v grafu
Tato transformace vytvoří graf s $\mathcal{O}(W * V)$ vrcholy
Celková časová složitost tohoto vyhledávání je tedy
$\mathcal{O}(W * V + E)$

Problémy s hledáním nejkratší cesty pomocí BFS

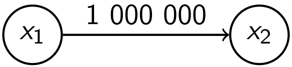
Problémy s hledáním nejkratší cesty pomocí BFS

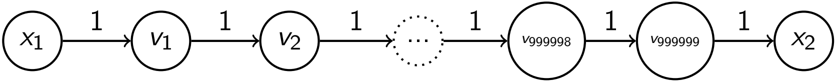
Co když jsou váhy v milionech?

$$x_1 \xrightarrow{1\ 000\ 000} x_2$$

Problémy s hledáním nejkratší cesty pomocí BFS

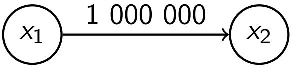
Co když jsou váhy v milionech?

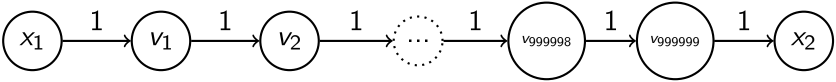




Problémy s hledáním nejkratší cesty pomocí BFS

Co když jsou váhy v milionech?

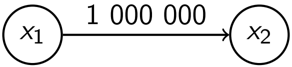


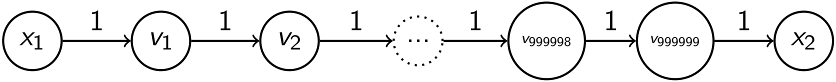


Co když váhy nejsou celočíselné hodnoty? (2.44, 1.75, 100.1)

Problémy s hledáním nejkratší cesty pomocí BFS

Co když jsou váhy v milionech?





Co když váhy nejsou celočíselné hodnoty? (2.44, 1.75, 100.1)

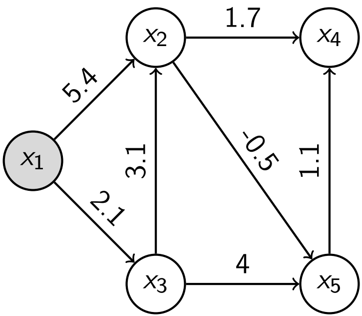
Co když povolíme záporné váhy? (-1, -44, -105.66)

Dijkstrův algoritmus

```
function Dijkstra(Graph, source):

    for each vertex v in Graph.Vertices:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]
        remove u from Q

        for each neighbor v of u still in Q:
            alt ← dist[u] + Graph.Edges(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Pseudocode

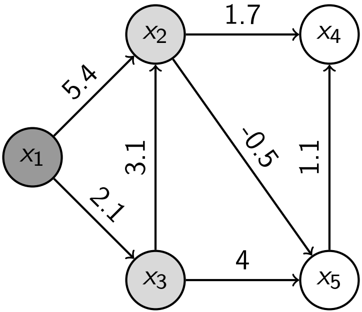
Dijkstrův algoritmus



V	x_1	x_2	x_3	x_4	x_5
P	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
D	0	∞	∞	∞	∞

Q	$\mathbf{x_1}$	x_2	x_3	x_4	x_5

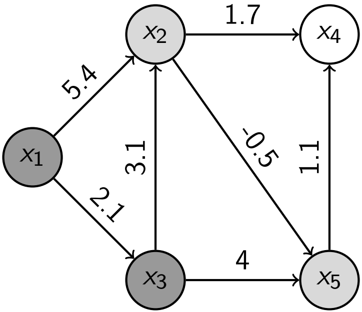
Dijkstrův algoritmus



V	x_1	x_2	x_3	x_4	x_5
P	$\emptyset$	$\mathbf{x_1}$	$\mathbf{x_1}$	$\emptyset$	$\emptyset$
D	0	**5.4**	**2.1**	∞	∞

Q	x_2	$\mathbf{x_3}$	x_4	x_5

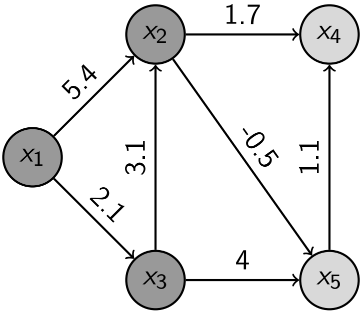
Dijkstrův algoritmus



V	x_1	x_2	x_3	x_4	x_5
P	$\emptyset$	$\mathbf{x_3}$	x_1	$\emptyset$	$\mathbf{x_3}$
D	0	**5.2**	2.1	∞	**6.1**

Q	$\mathbf{x_2}$	x_4	x_5

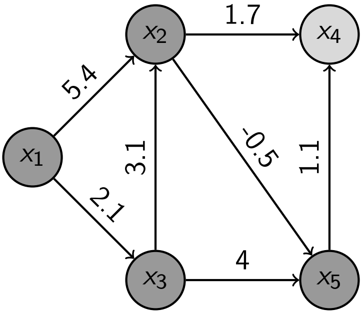
Dijkstrův algoritmus



V	x_1	x_2	x_3	x_4	x_5
P	$\emptyset$	x_3	x_1	$\mathbf{x_2}$	$\mathbf{x_2}$
D	0	5.2	2.1	**6.9**	**4.7**

Q	x_4	$\mathbf{x_5}$

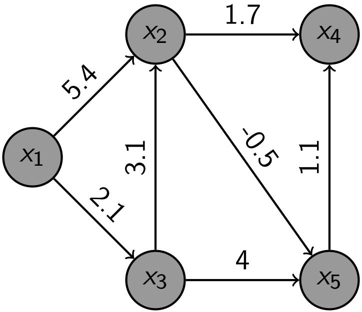
Dijkstrův algoritmus



V	x_1	x_2	x_3	x_4	x_5
P	$\emptyset$	x_3	x_1	$\mathbf{x_5}$	x_2
D	0	5.2	2.1	**5.8**	4.7

Q	$\mathbf{x_4}$

Dijkstrův algoritmus



V	x_1	x_2	x_3	x_4	x_5
P	$\emptyset$	x_3	x_1	x_5	x_2
D	0	5.2	2.1	5.8	4.7

Q

Časová složitost

Algoritmus vyžaduje přítomnost prioritní fronty

Časová složitost

Algoritmus vyžaduje přítomnost prioritní fronty

Pokud budeme procházel lineárně pole reprezentující frontu časová složitost Dijkstrova algoritmu bude

Časová složitost

Algoritmus vyžaduje přítomnost prioritní fronty

Pokud budeme procházel lineárně pole reprezentující frontu časová složitost Dijkstrova algoritmu bude $\mathcal{O}(|V|^2 + |E|)$

Časová složitost

Algoritmus vyžaduje přítomnost prioritní fronty

Pokud budeme procházel lineárně pole reprezentující frontu časová složitost Dijkstrova algoritmu bude $\mathcal{O}(|V|^2 + |E|)$

Pokud využijeme binární haldu (minulá přednáška), časová složitost operace vyhledání minima není lineární, ale pouze logaritmická Časová složitost se tedy zlepší na

Časová složitost

Algoritmus vyžaduje přítomnost prioritní fronty

Pokud budeme procházel lineárně pole reprezentující frontu časová složitost Dijkstrova algoritmu bude $\mathcal{O}(|V|^2 + |E|)$

Pokud využijeme binární haldu (minulá přednáška), časová složitost operace vyhledání minima není lineární, ale pouze logaritmická
Časová složitost se tedy zlepší na $\mathcal{O}((|V| + |E|)\log(|V|))$

Časová složitost

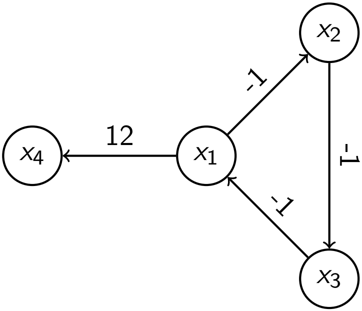
Algoritmus vyžaduje přítomnost prioritní fronty

Pokud budeme procházel lineárně pole reprezentující frontu časová složitost Dijkstrova algoritmu bude $\mathcal{O}(|V|^2 + |E|)$

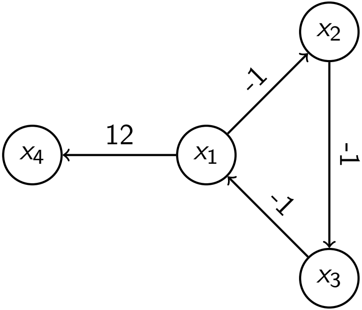
Pokud využijeme binární haldu (minulá přednáška), časová složitost operace vyhledání minima není lineární, ale pouze logaritmická Časová složitost se tedy zlepší na $\mathcal{O}((|V| + |E|)\log(|V|))$

Použitím Fibonacciho haldy je možné dosáhnout časové složitosti $\mathcal{O}(|E| + |V|\log(|V|))$

Negativní cykly

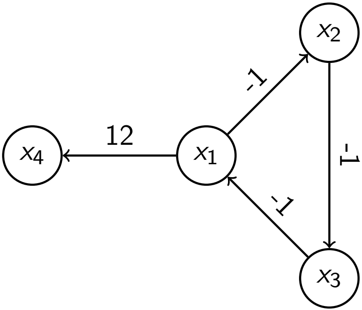


Negativní cykly



Jak dlouhá by byla nejkratší cesta z x_1 do x_4?

Negativní cykly



Jak dlouhá by byla nejkratší cesta z x_1 do x_4?
Na toto není Dijkstrův algoritmus stavěný
Relaxuje totiž stavy s lepší (menší) vzdáleností a může se takto zacyklit

Bellman-Ford

```
bellman-ford(vrcholy, hrany, zdroj)

// krok 1: inicializace grafu
for each v in vrcholy
  if v=zdroj then v.vzdálenost := 0
             else v.vzdálenost := nekonečno
  v.předchůdce := null

// krok 2: opakovaně relaxovat hrany
for i from 1 to size(vrcholy)-1
  for each h in hrany    // h je hrana z u do v
    u := h.počátek
    v := h.konec
    if u.vzdálenost + h.délka < v.vzdálenost
      v.vzdálenost := u.vzdálenost + h.délka
      v.předchůdce := u

// krok 3: kontrola záporných cyklů
for each h in hrany
  u := h.počátek
  v := h.konec
  if u.vzdálenost + h.délka < v.vzdálenost
    error "Graf obsahuje záporný cyklus."
```

Wikipedia

Bellman-Ford

Bellman-Fordův algoritmus detekuje negativní cykly a je tedy konečný i na grafech s negativním cyklem

Bellman-Ford

Bellman-Fordův algoritmus detekuje negativní cykly a je tedy konečný i na grafech s negativním cyklem

Tento algoritmus ale běží v čase $\mathcal{O}(|V| * |E|)$