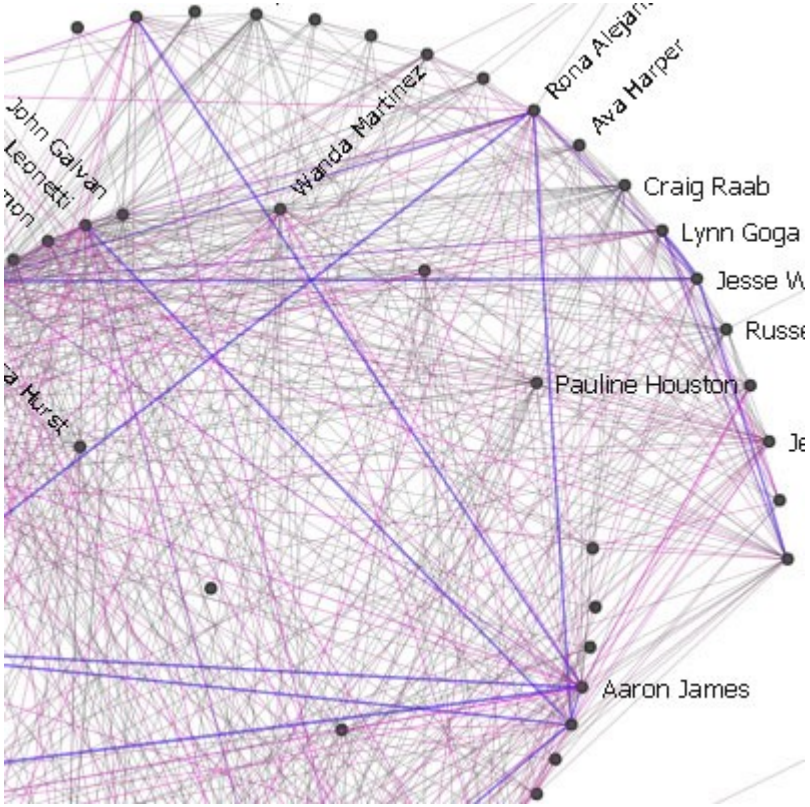
STU FEI

Programovacie techniky

10. Hľadanie do hĺbky, hľadanie do šírky, najkratšia cesta v grafe, Dijkstrov algoritmus

Facebook ako graf



Uzol = používateľ
Hrana = priateľstvo medzi používateľmi

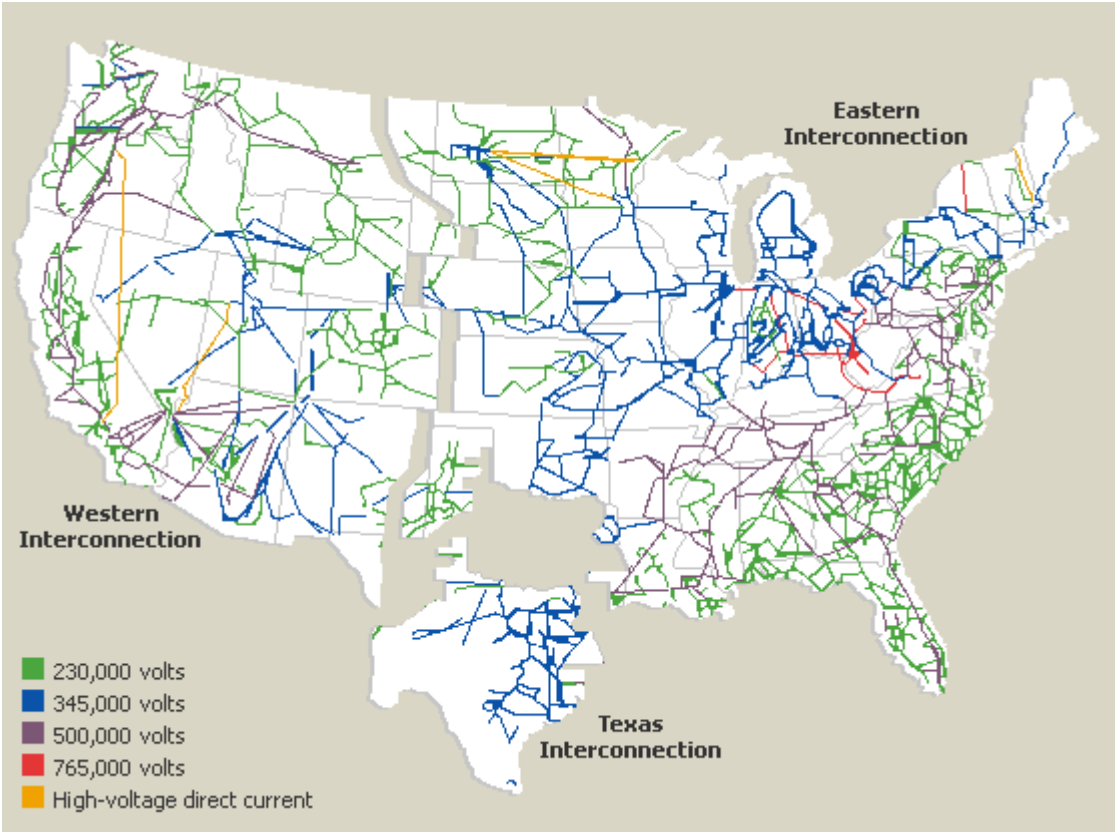
Hľadanie do šírky?
Priemerný stupeň uzla?
Najkratšia cesta?
Priemer grafu?
- six degrees of separation

Six degrees of separation

"We found that six degrees actually overstates the number of links between typical pairs of users: while 99.6 percent of all pairs of users are connected by paths with 5 degrees, 92 percent are connected by only four degrees."

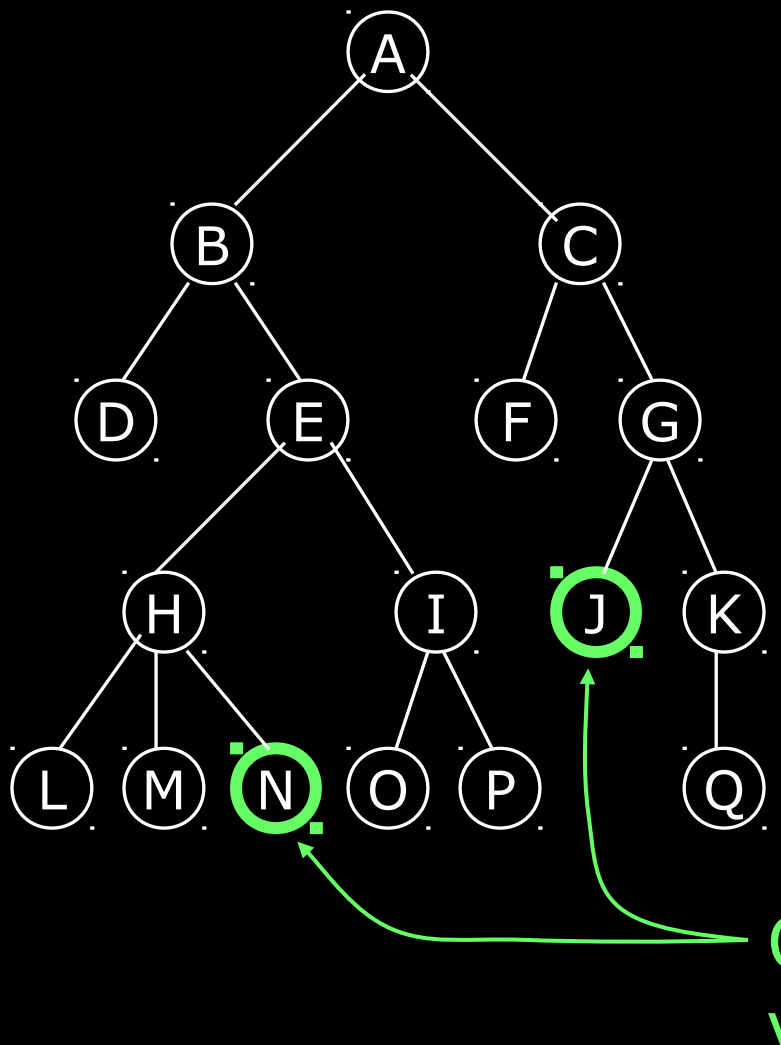
Source: https://www.facebook.com/notes/facebook-data-team/anatomy-of-facebook/10150388519243859

Elektrizačná prenosová sieť



Aký je priemerný stupeň uzla?
V porovnaní s Facebookom?

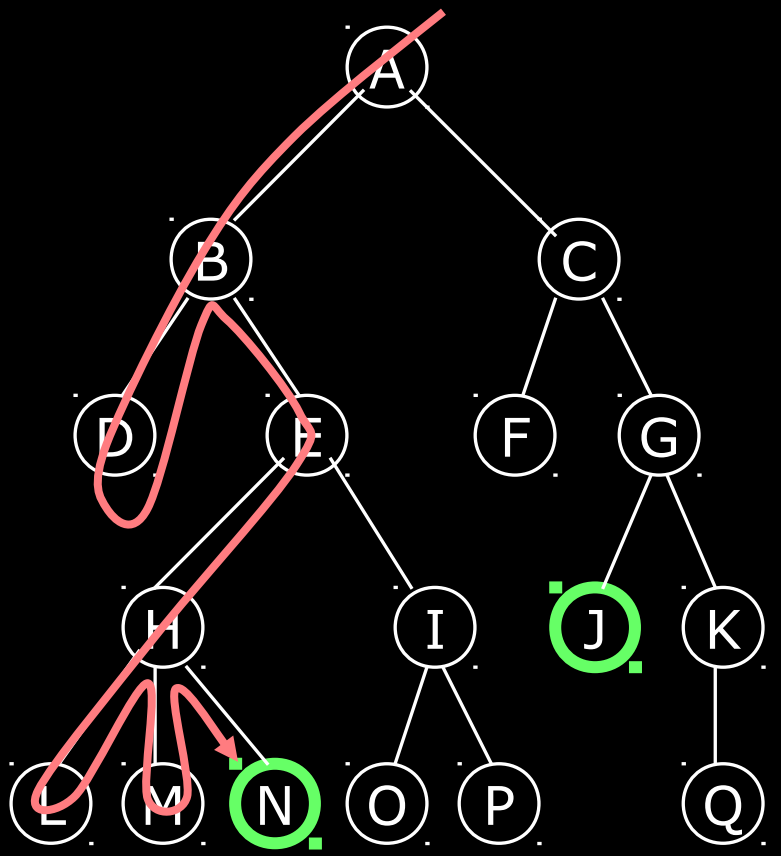
Hľadanie v strome



N a J sú vrcholy, ktoré hľadáme

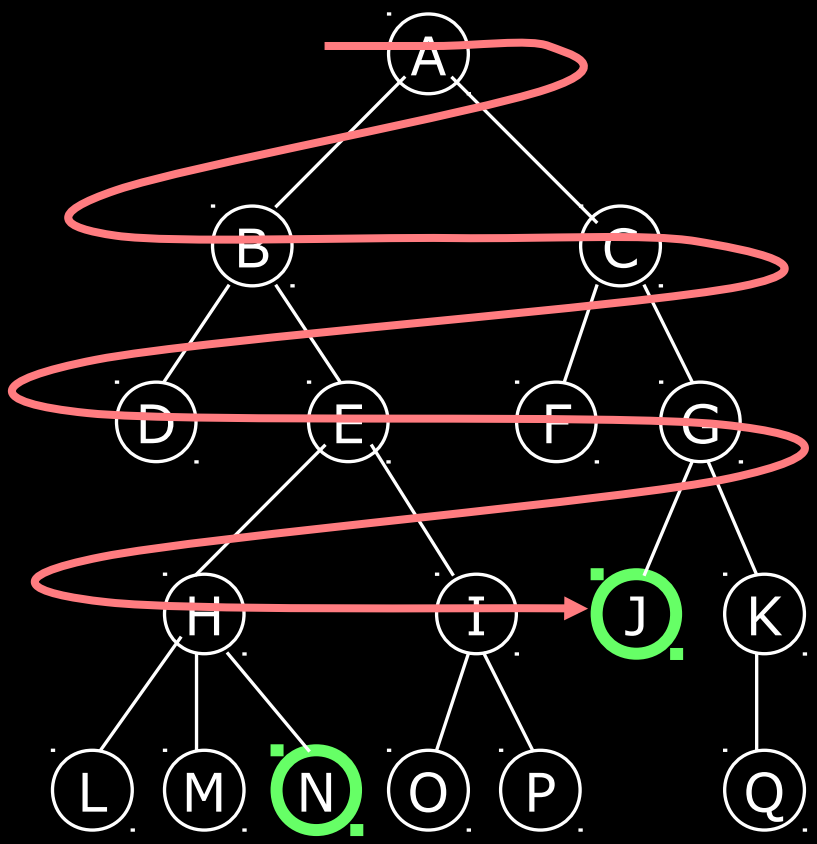
Nachádza sa N alebo J v strome?

Hľadanie do hĺbky



Vrcholy sú navštívené v poradí: A B D E H L M N I O P C F G J K Q

Hľadanie do šírky



Vrcholy sú navštívené v poradí: A B C D E F G H I J K L M N O P Q

J je nájdené pred N

Hľadanie do hĺbky

```
put the root node on a STACK;

while (stack is not empty) {
    remove a node from the stack;

    if (node is a goal node) return success;

    put all children of node onto the stack;
}

return failure;
```

Hľadanie do šírky

```
put the root node on a QUEUE;

while (queue is not empty) {
    remove a node from the queue;

    if (node is a goal node) return success;

    put all children of node onto the queue;
}

return failure;
```

Poznámky

Pre veľký strom, hľadanie do šírky môže mať veľké nároky na pamäť

Pre veľký strom, hľadanie do hĺbky môže trvať dlho pre cieľový uzol blízko koreňa

Zložitosť DFS, BFS: $O(|E|)$

$|E|$ je kardinalita (mohutnosť) množiny E, t.j. počet hrán

DFS, BFS

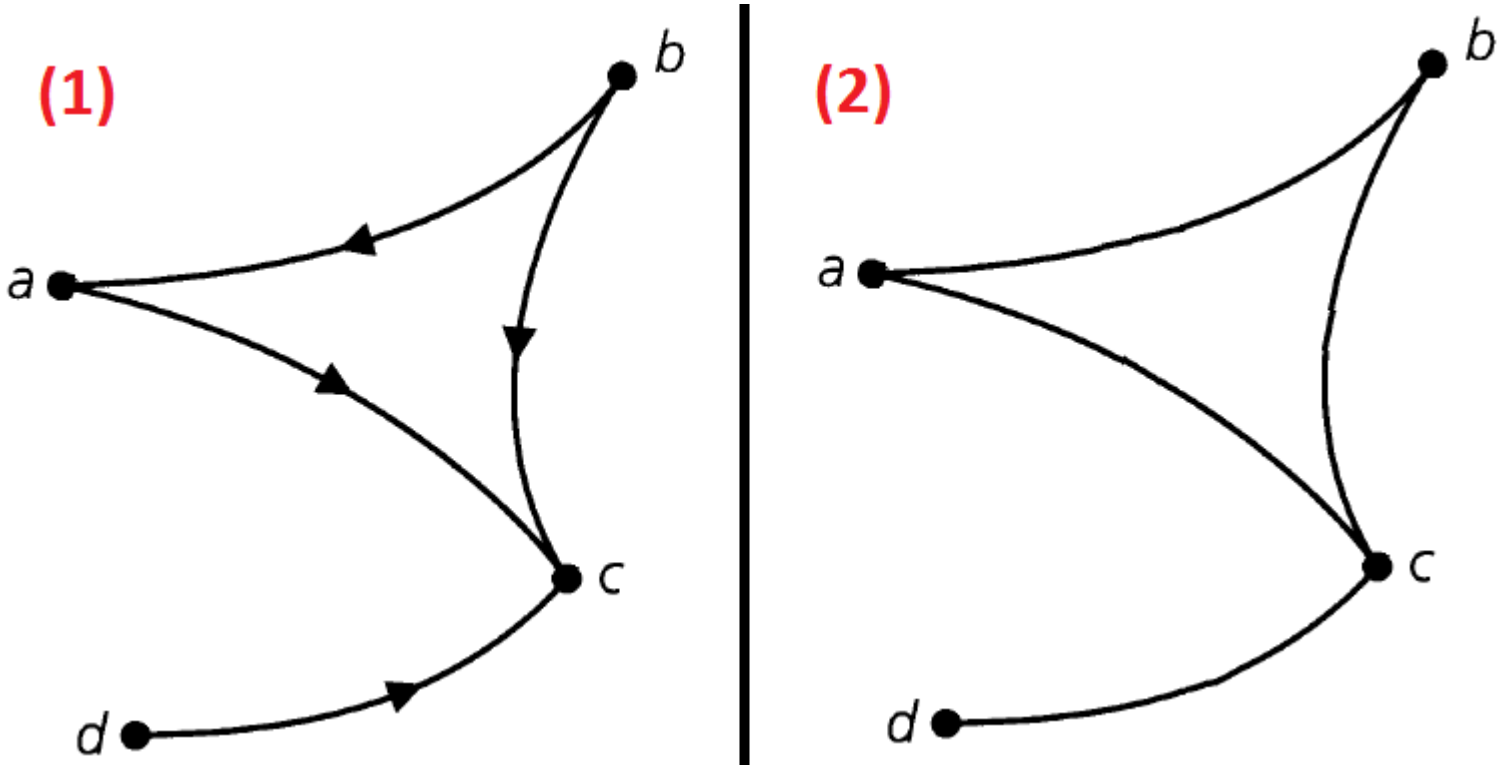
BFS = breadth first search

DFS = depth first search

Úloha

Implementuj DFS, ktorý hľadá len do hĺbky depth_limit

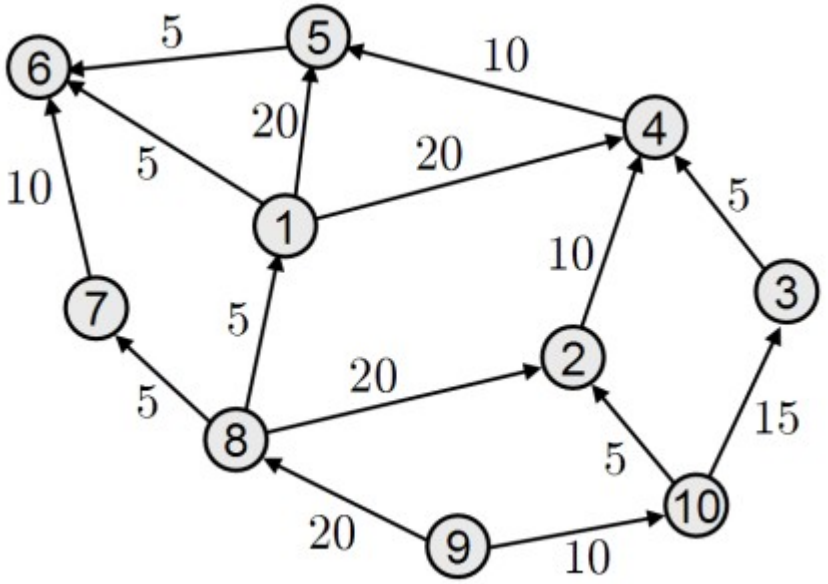
Orientovaný graf



Directed graph
Orientovaný graf

Undirected graph
Neorientovaný graf

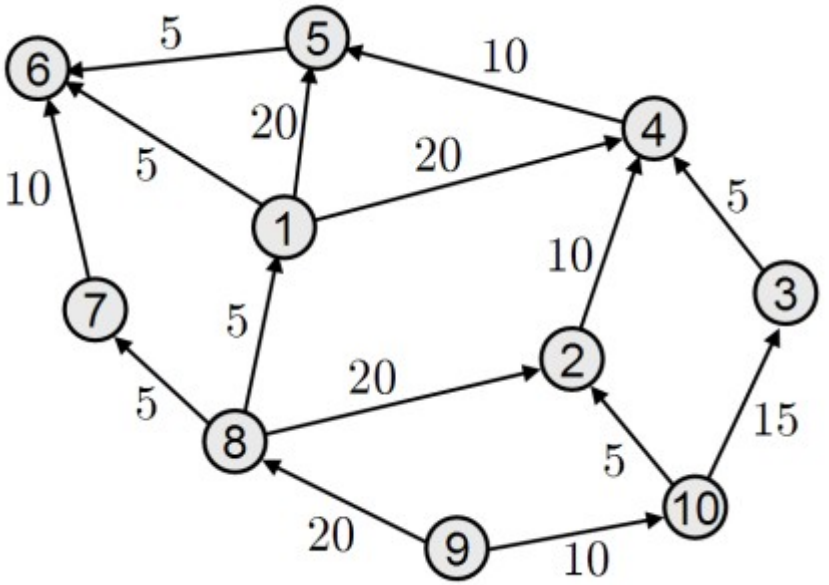
Hranovo-ohodnotený graf



Orientovaný hranovo-ohodnotený graf

Aká je najkratšia cesta z vrcholu 10 do vrcholu 6?

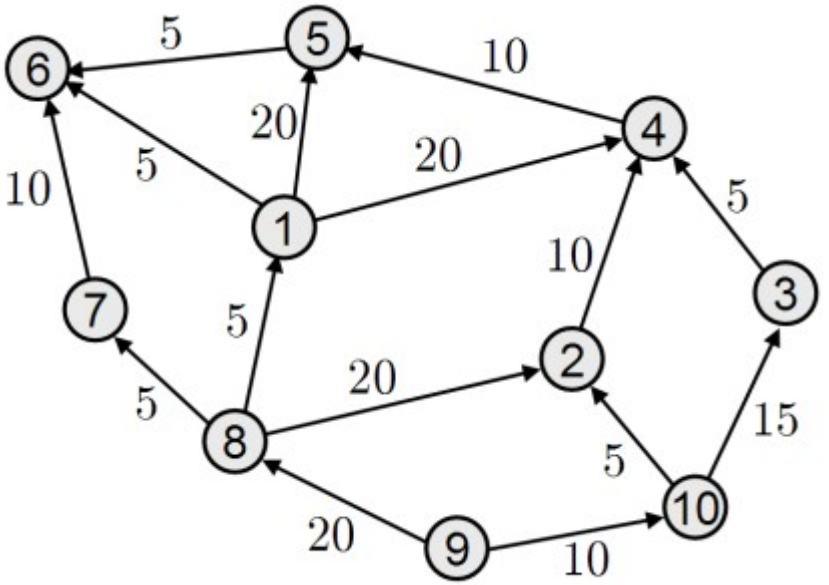
Hranovo-ohodnotený graf



G = (V, E, w) alebo G = (V, E)

w je funkcia E—>R, kde R je množina reálnych čísel
V je množina vrcholov
E je množina hrán: V x V

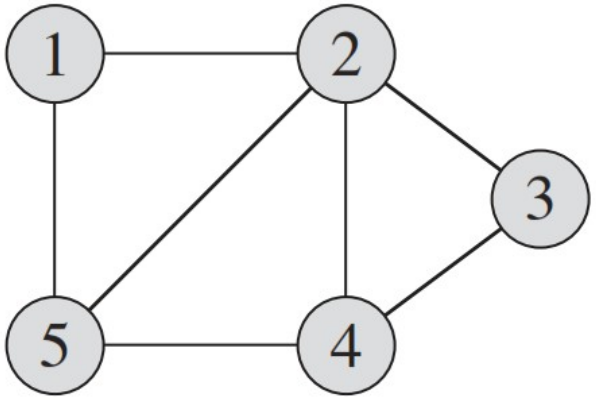
Hranovo-ohodnotený graf

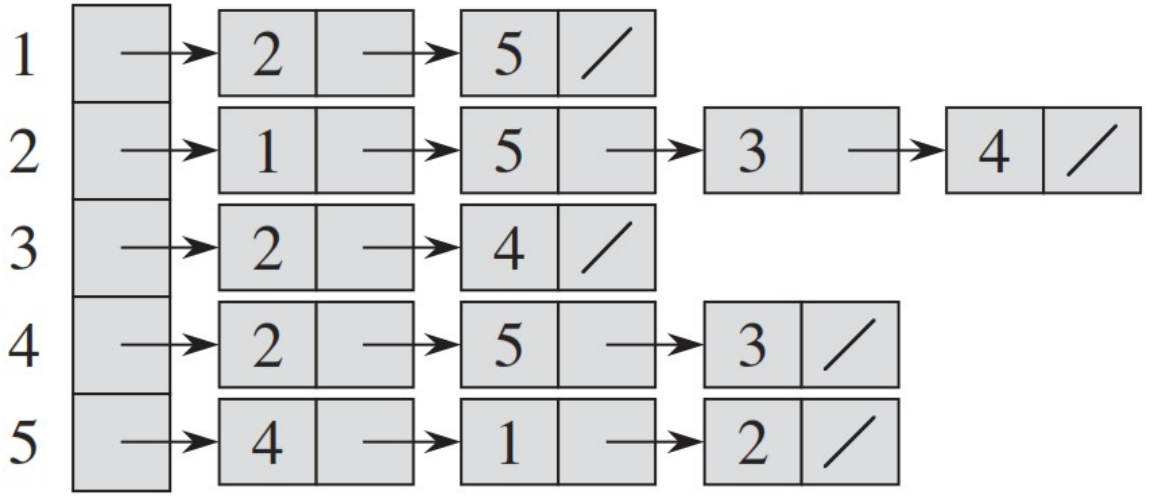


G = (V, E, w) alebo G = (V, E)

V = vertices
E = edges

Reprezetácia grafu



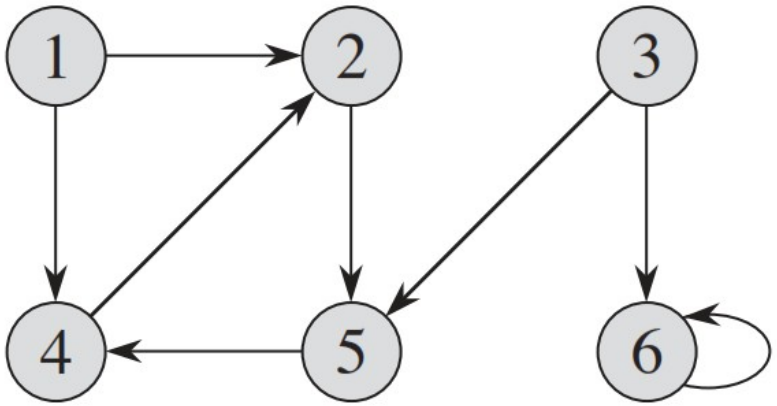


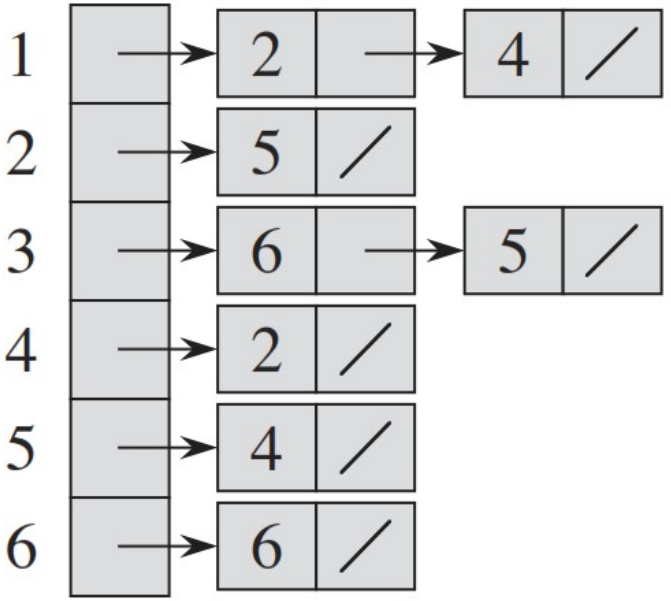
Zoznam susedov
Adjacency list

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Matica susedov
Adjacency matrix

Orientovaný graf





Zoznam susedov
Adjacency list

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Matica susedov
Adjacency matrix

BFS: pseudocode

```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```

Farba = {white, gray, black}

white nenájdený

gray nájdený

black všetci susedia nájdení

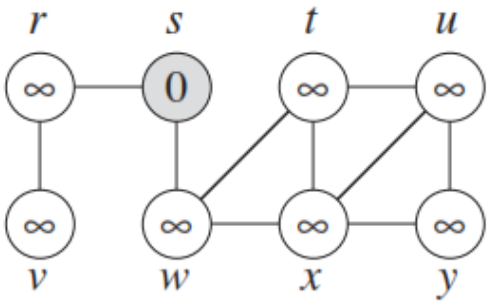
u.d je vzdialenosť z s do u

u.π je predchodca u (predecessor)

Q je fronta (queue)

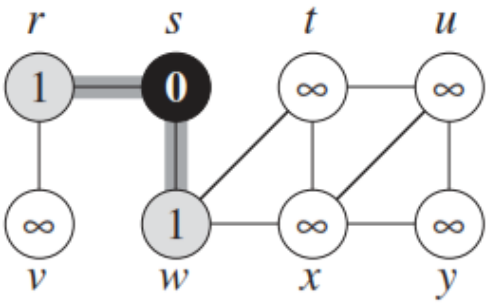
BFS: príklad

(a)



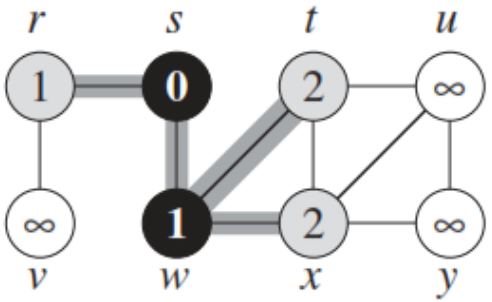
Q | s |
0

(b)



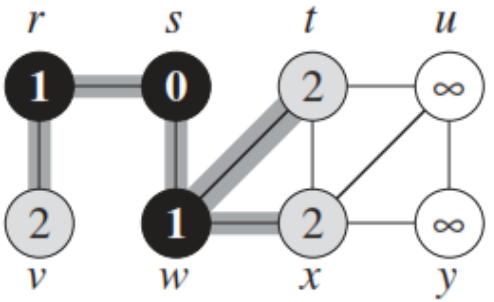
Q | w | r |
1 1

(c)



Q | r | t | x |
1 2 2

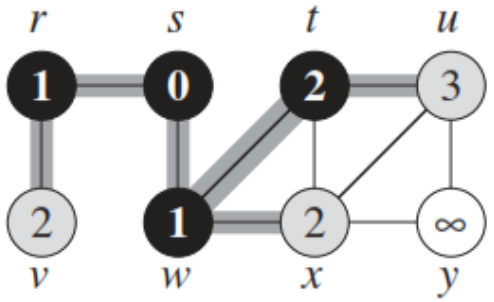
(d)



Q | t | x | v |
2 2 2

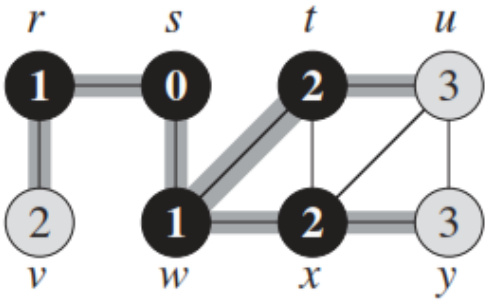
BFS: príklad

(e)



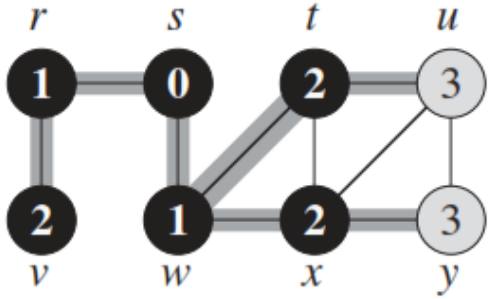
Q: x v u — 2 2 3

(f)



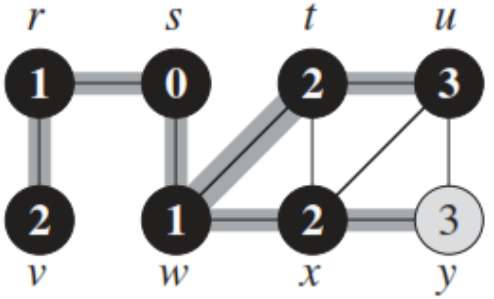
Q: v u y — 2 3 3

(g)



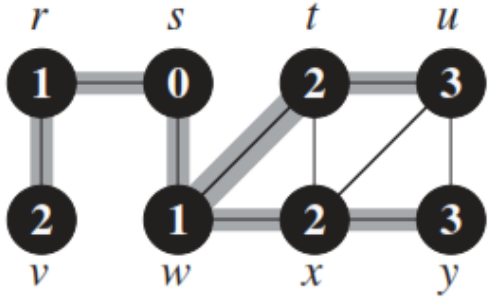
Q: u y — 3 3

(h)



Q: y — 3

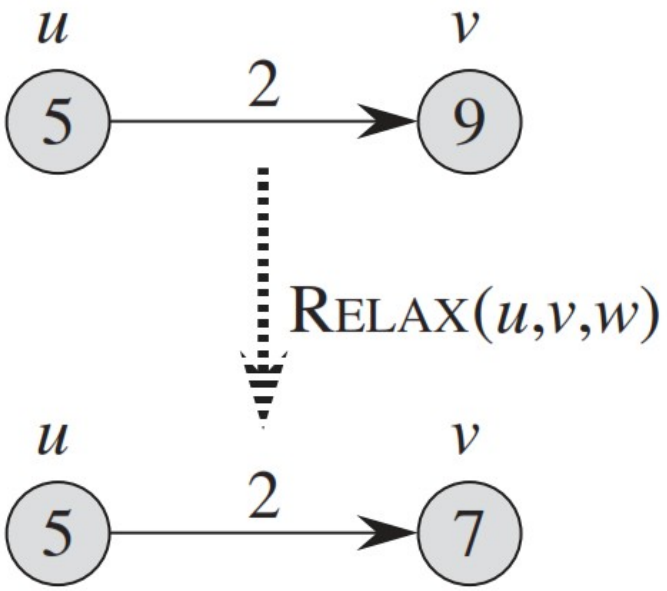
(i)

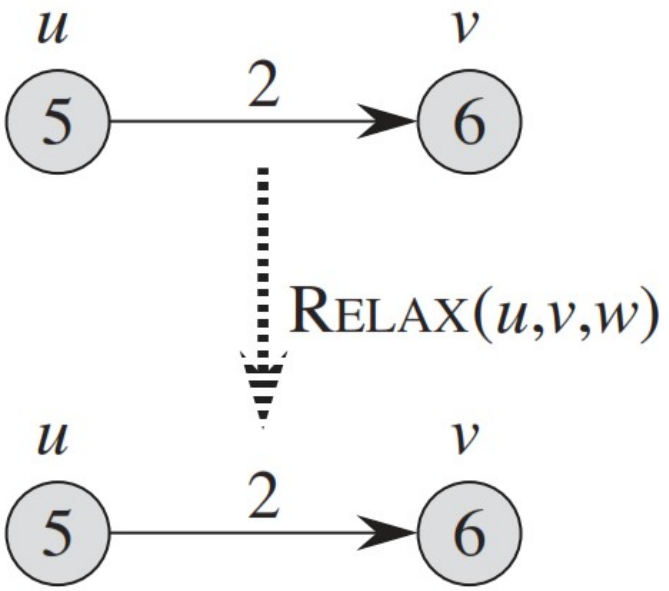


Q: $\emptyset$

Najkratšia cesta: relaxácia

```
RELAX(u, v, w)
1  if v.d > u.d + w(u, v)
2        v.d = u.d + w(u, v)
3        v.π = u
```





Najkratšia cesta: inicializácia

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2      v.d = ∞
3      v.π = NIL
4  s.d = 0
```

Dijkstrov algoritmus

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

S je množina vrcholov, ktorých vzdialenosť od s je už známa

Q je min priority queue

w je nezáporné ohodnotenie hrany

Zložitosť

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

G je orientovaný graf

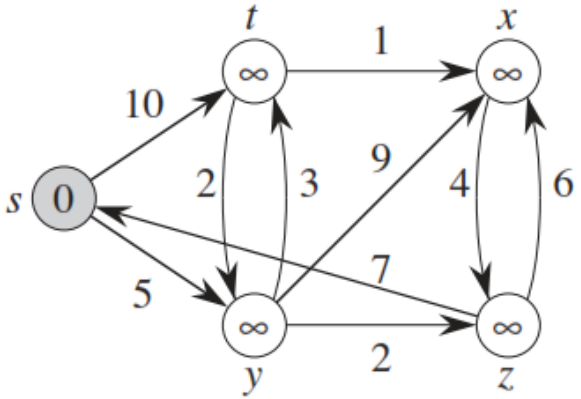
O(|V|)
minimum: O(log |V|)

O(|E|), celkovo
decrease key: zmena hodnoty vzdialenosti, O(log |V|)

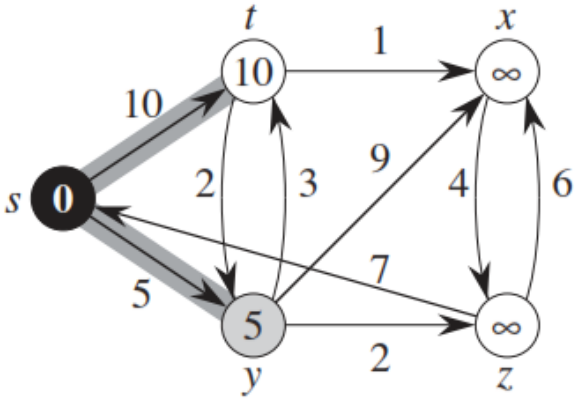
Heap: O((|E| + |V|) log |V|)

Fibonacci heap: O(|E| + |V| log |V|)

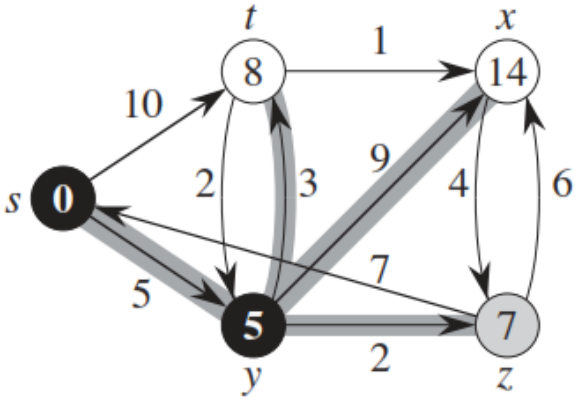
Dijkstrov algoritmus



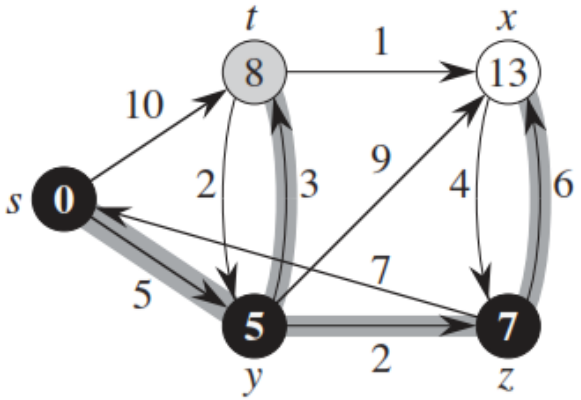
(a)



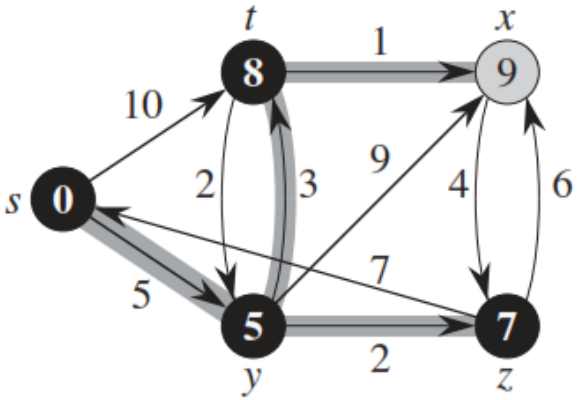
(b)



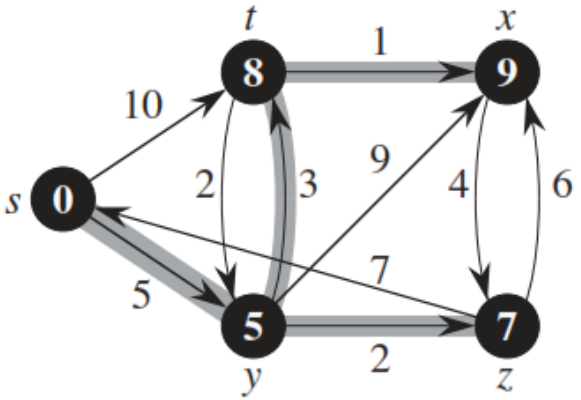
(c)



(d)

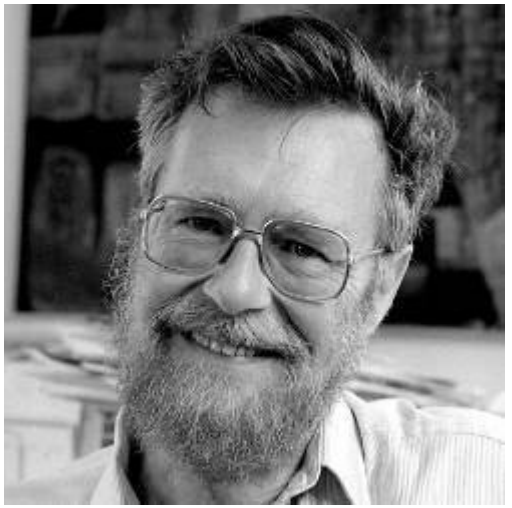


(e)



(f)

Edsger Wybe Dijkstra



Najkratšia cesta: 1959, Holandsko

Úlohy

Príklad, keď Dijkstra nenájde najkratšiu cestu v grafe so zápornými hodnotami hrán?

Aký je výstupný stupeň každého vrcholu? Výstupný stupeň v orientovanom grafe je počet hrán vychádzajúcich z vrcholu.

Vypočítaj priemer (diameter) grafu. Priemer je najdlhšia najkratšia cesta v grafe.