

PRŮCHOD GRAFEM DO ŠÍŘKY A DO HLOUBKY

Aneb, jak se neztratit v bludišti

Prohledávání do šířky

Breadth - First Search

BFS

BFS - Popis

- Algoritmus začne v libovolném počátečním vrcholu. Algoritmus nejprve projde všechny sousedy startovního vrcholu, poté sousedy sousedů atd. až projde všechny dostupné vrcholy.
- Algoritmus používá pro vrcholy v grafu následující stavy:
 - FRESH (Ještě nebyl objeven)
 - OPEN (Právě objeven)
 - CLOSE (Už byl prozkoumán)
- Vrcholy se při průchodu grafem ukládají na FIFO frontu a z ní jsou posléze odebírány.

BFS - Pseudokód

```
void BFS (Graph G, Node s){  
    for (Node u in U(G)-s){  
        state[u] = FRESH;  
    }
```

```
    state[s] = OPEN;  
    Queue.Add(s);
```

```
    while(!Queue.Empty()){  
        u = Queue.Remove();
```

```
        for(v in Adj[u]){  
            if(state[v] == FRESH){  
                state[v] = OPEN;  
                Queue.Add(v);  
            }  
        }
```

```
        state[u]=CLOSED;
```

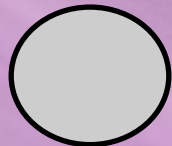
```
    }
```

```
}
```

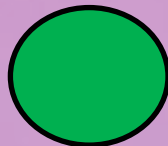
1. Všechny uzly, kromě počátečního označ jako FRESH.
2. Počáteční uzel označ jako OPEN a ulož ho do fronty.
3. Dokud není fronta prázdná, vyzvedni z ní uzel u.
4. Najdi všechny jeho sousedy a u těch, které mají stav FRESH , ho změň na OPEN a ulož je do fronty.
5. Uzlu u změň stav na CLOSED.

BFS - Ukázka

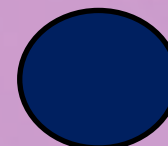
- Následuje ukázka průchodu naším bludištěm pomocí tohoto algoritmu.
- Všímejte si prosím dvou věcí:
 - Za prvé graf, kde se bude animováno postupné prozkoumávání grafu.
 - Za druhé frontu uzlů, kde bude animováno postupné přidávání a odebírání uzlů.
- Pro větší přehlednost budeme dodržovat pravidlo, že pokud budeme moci uložit více uzlů současně uložíme nejprve ten s nižším číslem.



FRESH

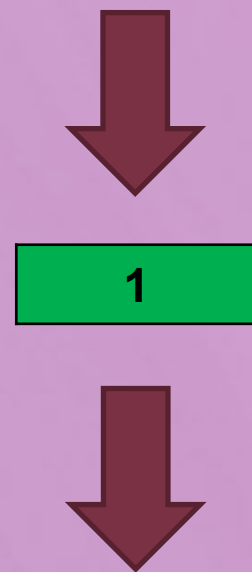
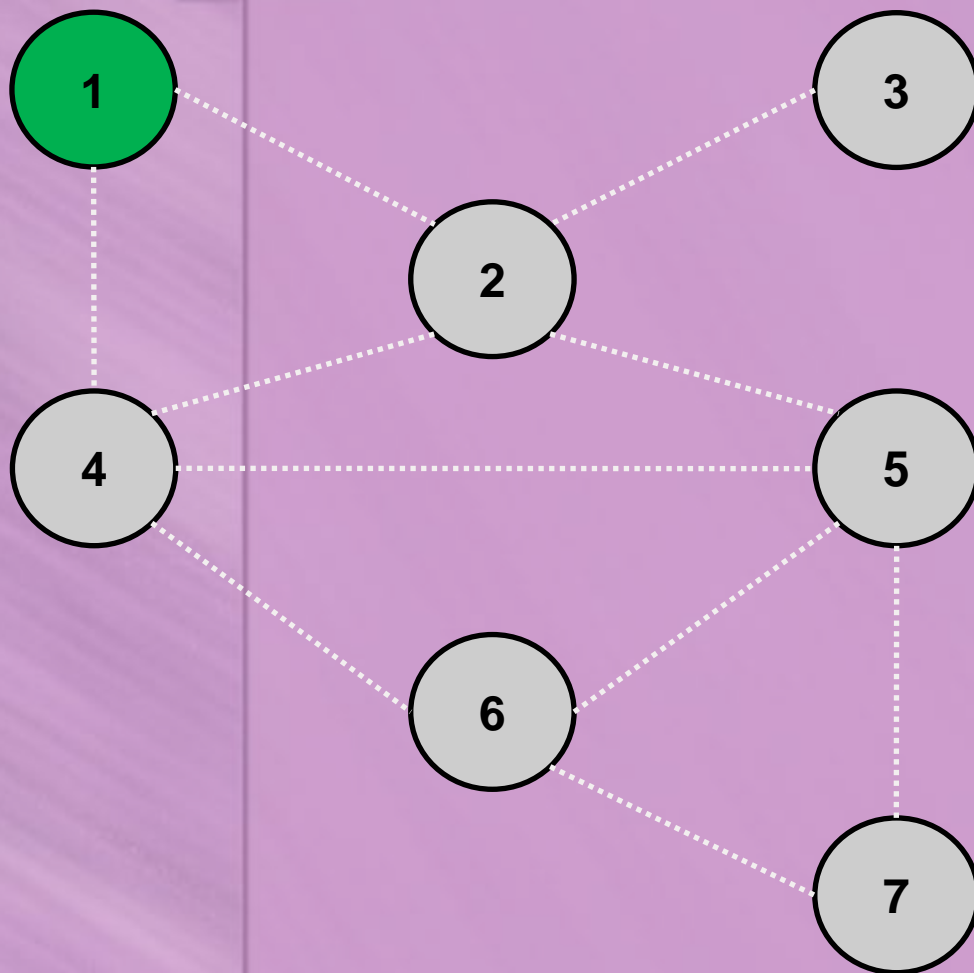


OPEN



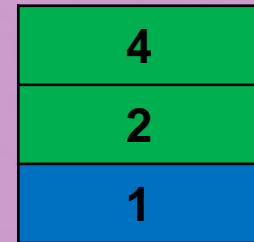
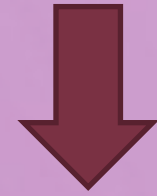
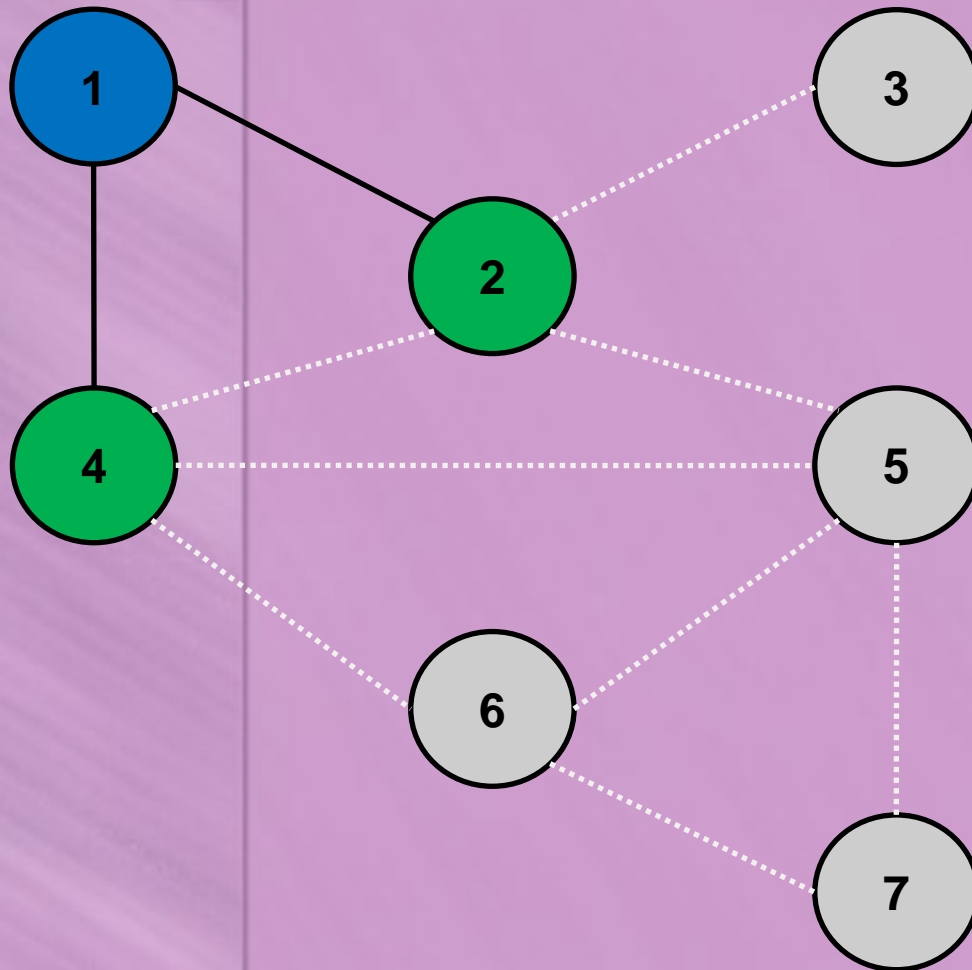
CLOSED

BFS - Ukázka



Ve frontě je OPEN uzel 1

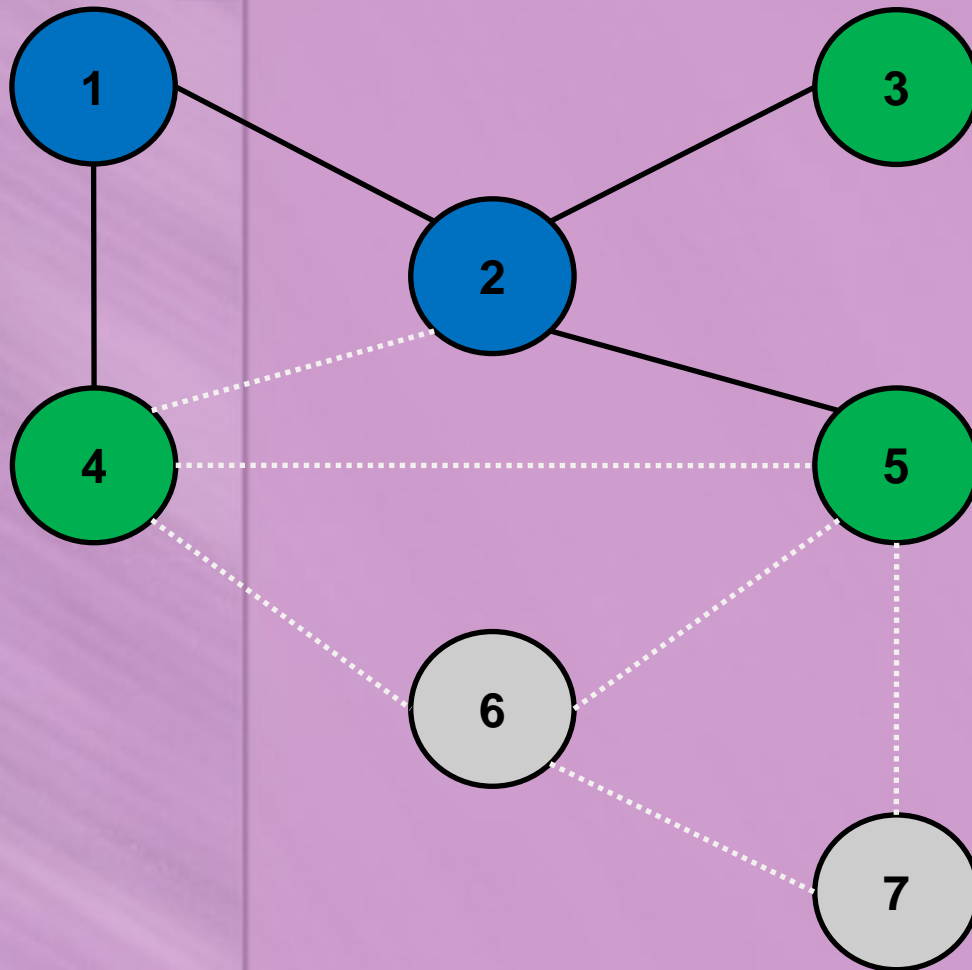
BFS - Ukázka



Odebíráme uzel 1

Přidáváme uzly 2 a 4

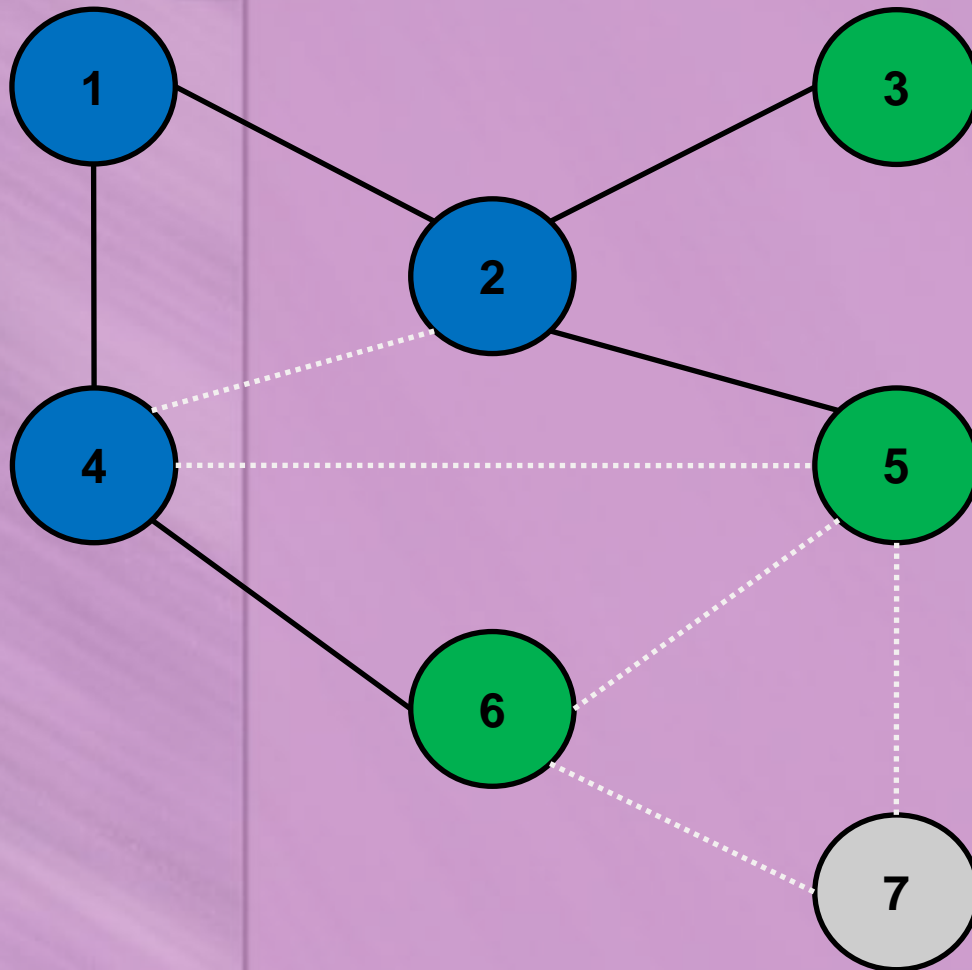
BFS - Ukázka



Odebíráme uzel 2

Přidáváme uzly 3 a 5

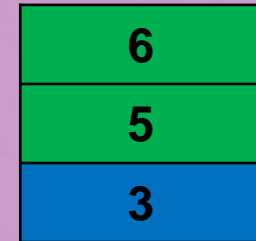
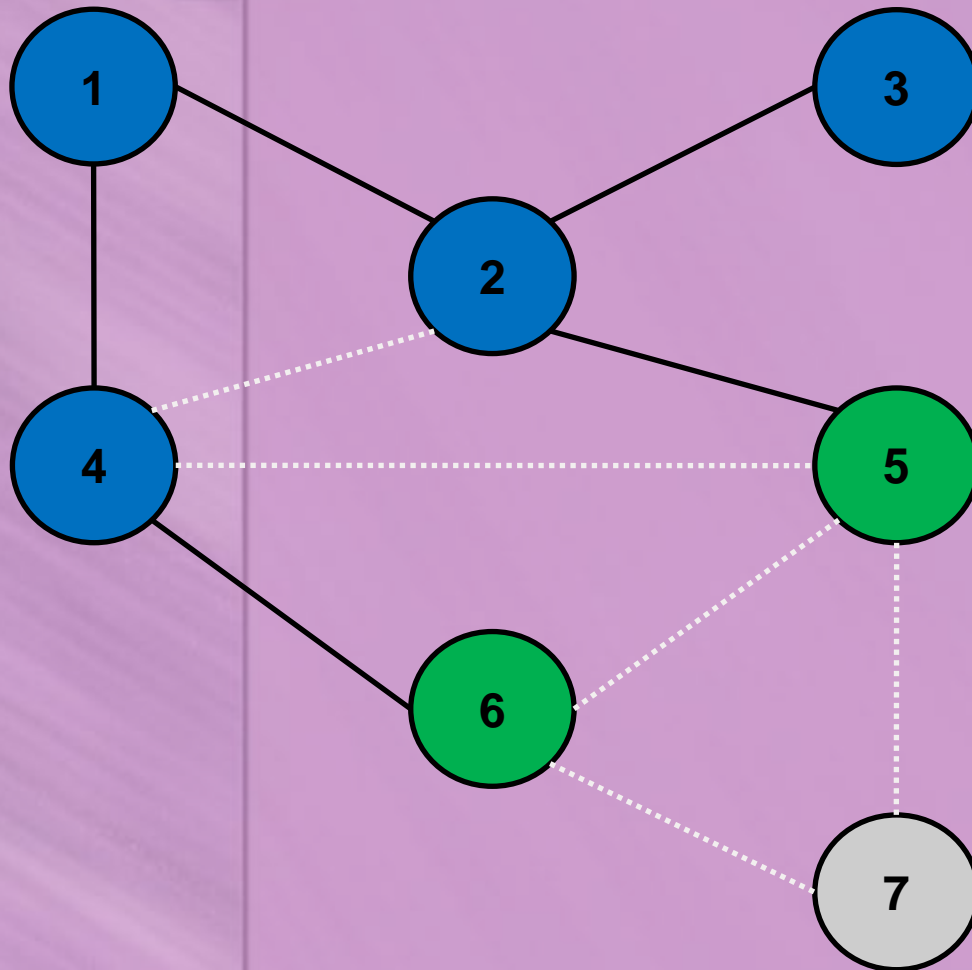
BFS - Ukázka



Odebíráme uzel 4

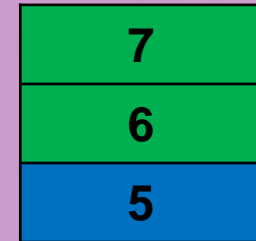
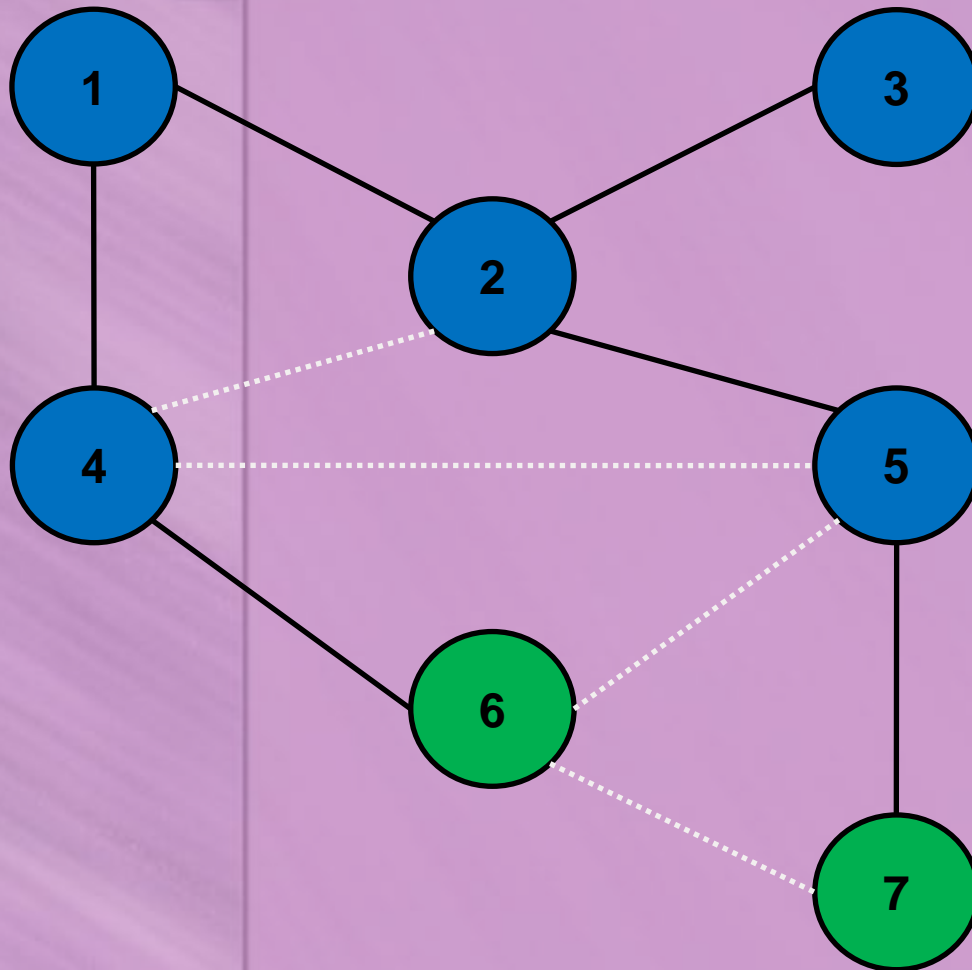
Přidáváme uzel 6

BFS - Ukázka



Odebíráme uzel 3

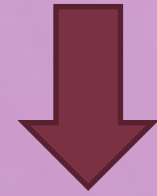
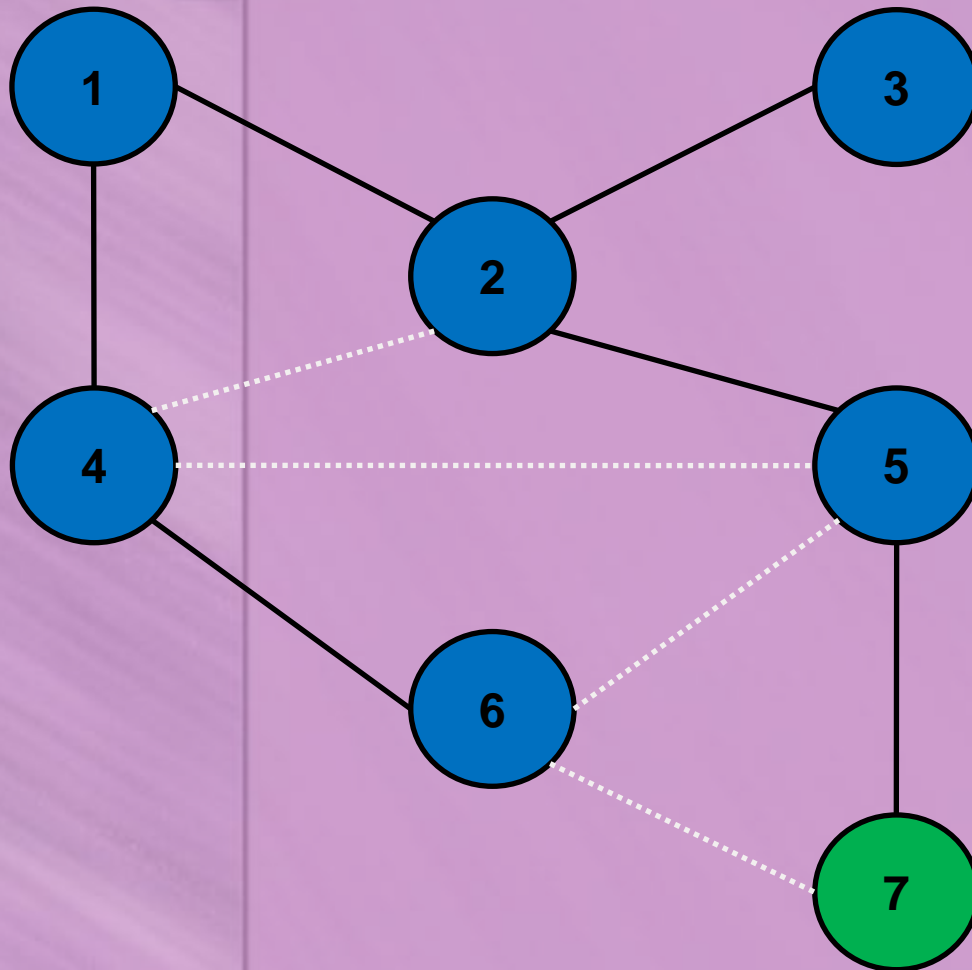
BFS - Ukázka



Odebíráme uzel 5

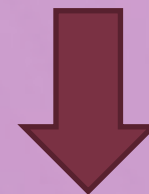
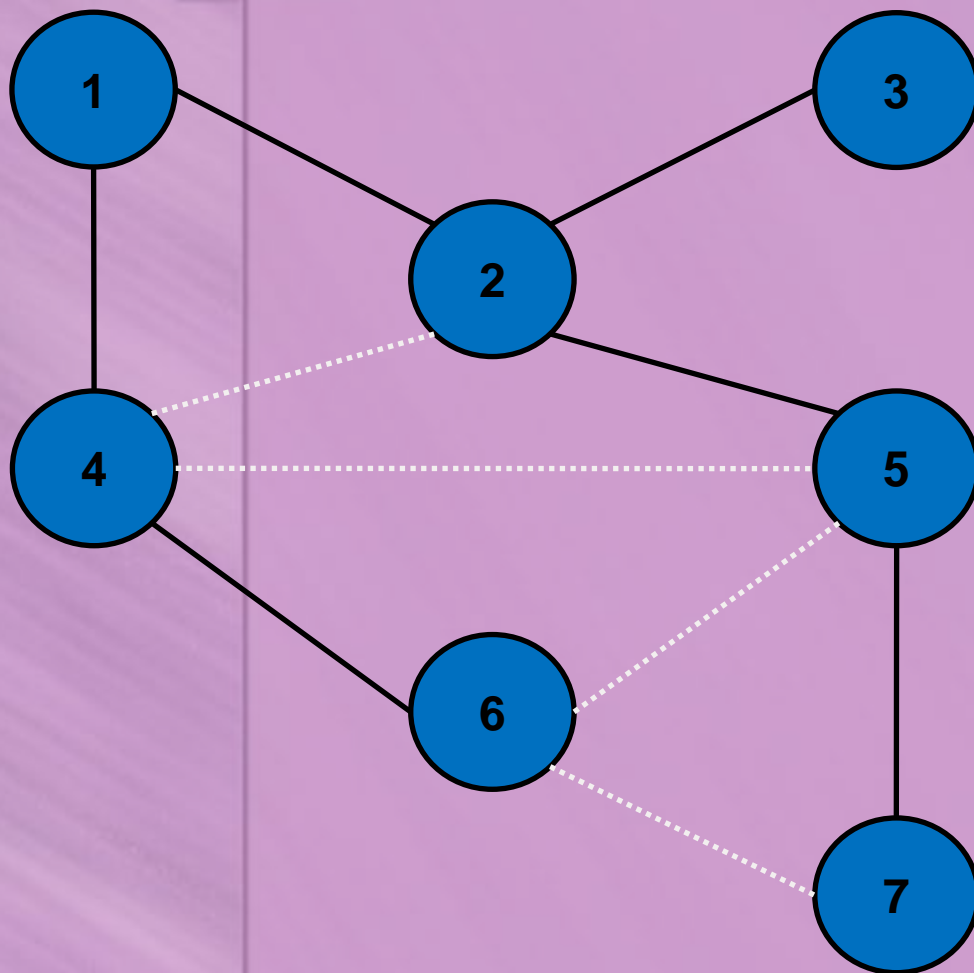
Přidáváme uzel 7

BFS - Ukázka

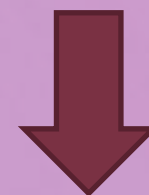


Odebíráme uzel 6

BFS - Ukázka

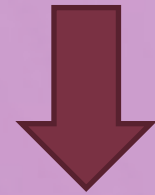
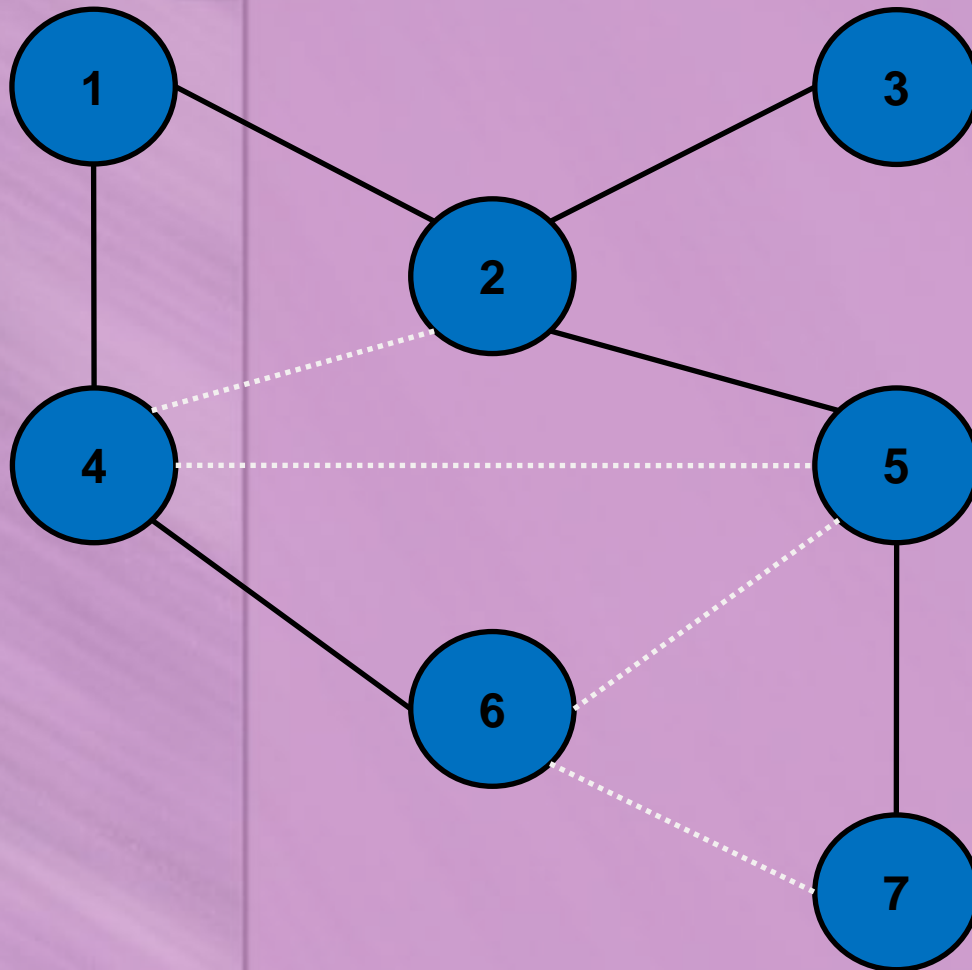


7

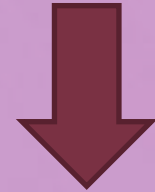


Odebíráme uzel 7

BFS - Ukázka

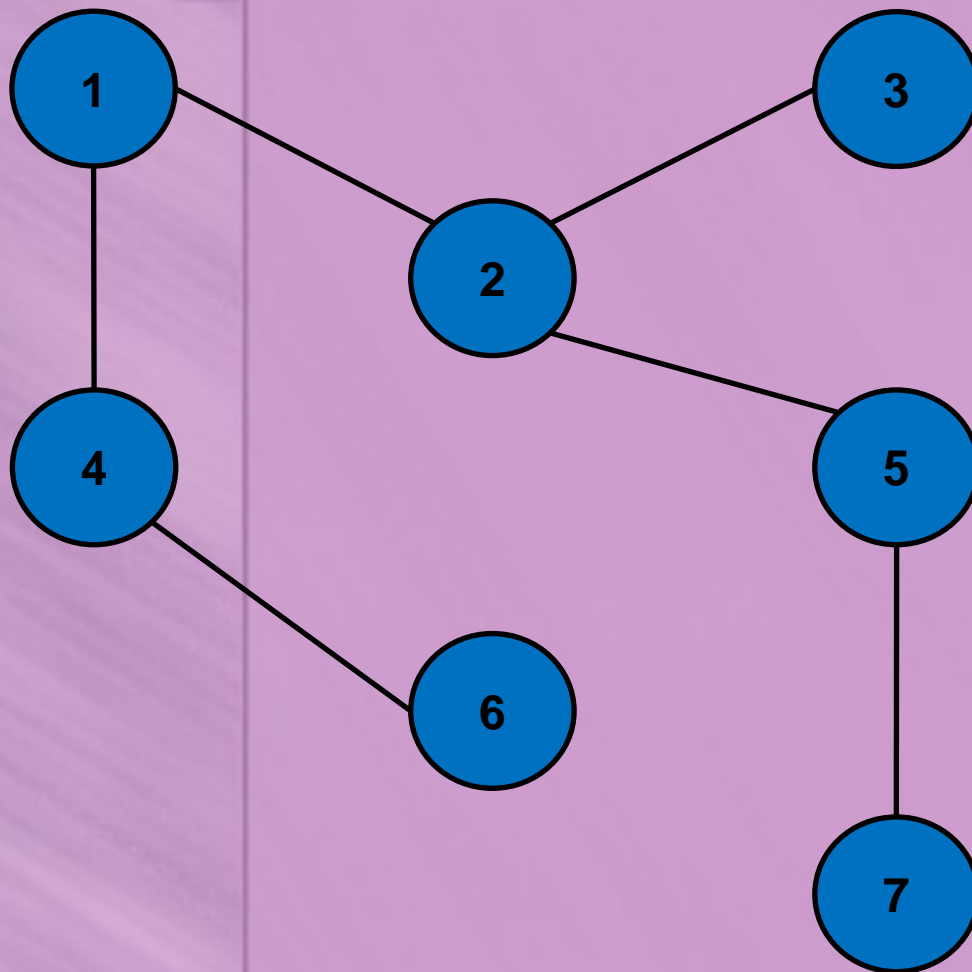


NULL




Fronta je prázdná

BFS - Ukázka



Algoritmus končí ve chvíli kdy je **prázdná** fronta.

Vznikl tzv. BF-Strom, tj. strom **nejkratších cest** z počátečního uzlu do ostatních, dostupných uzlů.



Prohledávání do hloubky

Depth - First Search
DFS

DFS - Popis

- Postupuje se stále dál od počátečního uzlu dosud neprozkoumaným směrem. Když už to dál nejde, vrátíme se pomocí backtrackingu a postupujeme zase co nejdál.
- Algoritmus používá pro vrcholy v grafu následující stavy:
 - FRESH (Ještě nebyl objeven)
 - OPEN (Právě objeven)
 - CLOSE (Už byl prozkoumán)

DFS - Pseudokód

```
void DFS (Graph G){  
  for (Node u in U(G)){  
    state[u] = FRESH;  
  }
```

1. Všechny uzly, označ jako FRESH.

```
  for (Node u in U(G)){  
    if (state[u] == FRESH){  
      DFS_Go(u);  
    }  
  }
```

2. Pro každý uzel který je FRESH zavolej proc. DFS-Go.

```
}  
void DFS_Go(Node u){  
  state[u] = OPEN;  
  for (Node v in Adj[u]){  
    if (state[v] == FRESH){  
      DFS_Go(v);  
    }  
  }
```

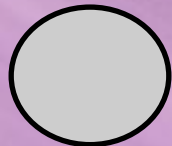
3. Pokud je stav uzlu u z parametru OPEN, tak vyhledej jeho sousedy a pokud mají stav FRESH tak pro něj zavolej rekurzivně DFS_Go.

```
  state[u] = CLOSED;  
}
```

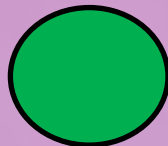
4. Uzlu u změň stav na CLOSED.

DFS - Ukázka

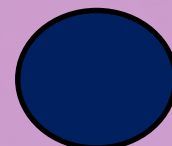
- Následuje ukázka průchodu naším bludištěm pomocí tohoto algoritmu.
- Všimněte si prosím grafu, kde bude animováno postupné prozkoumávání grafu.
- Pro větší přehlednost budeme dodržovat pravidlo, že pokud budeme mít na výběr více uzlů současně budeme pokračovat tím s nižším číslem.



FRESH

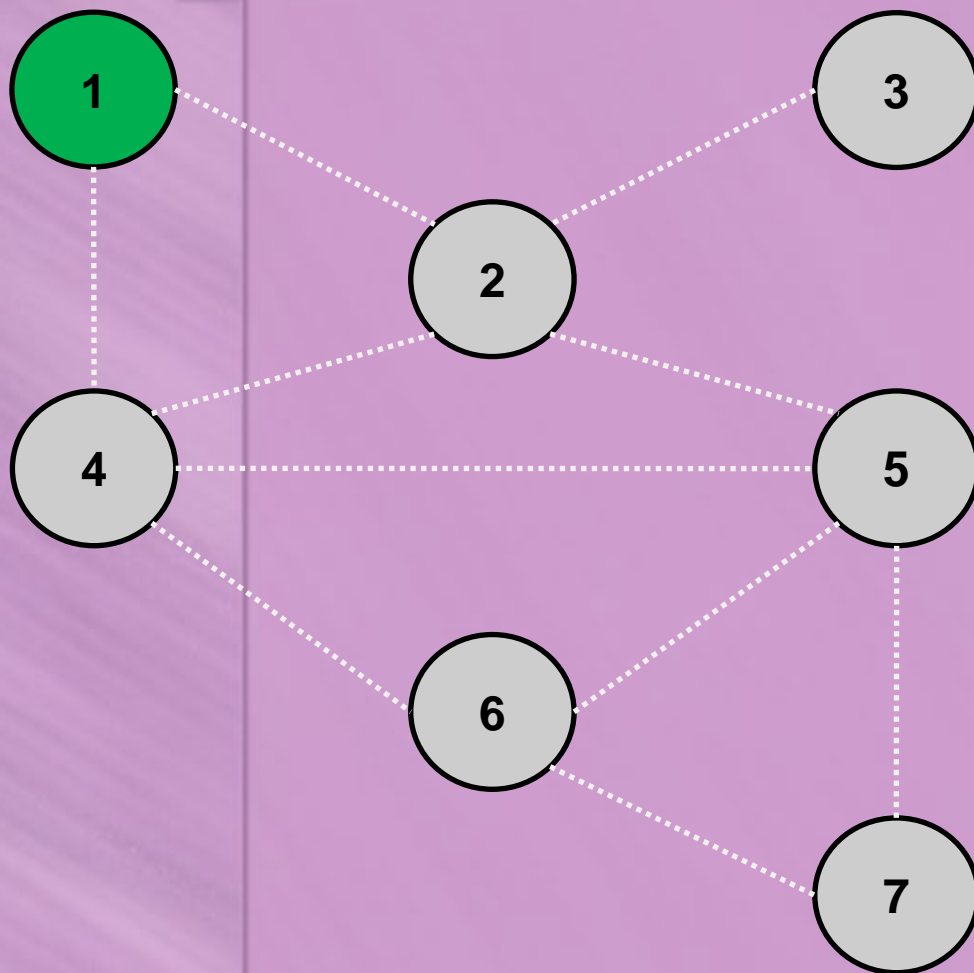


OPEN



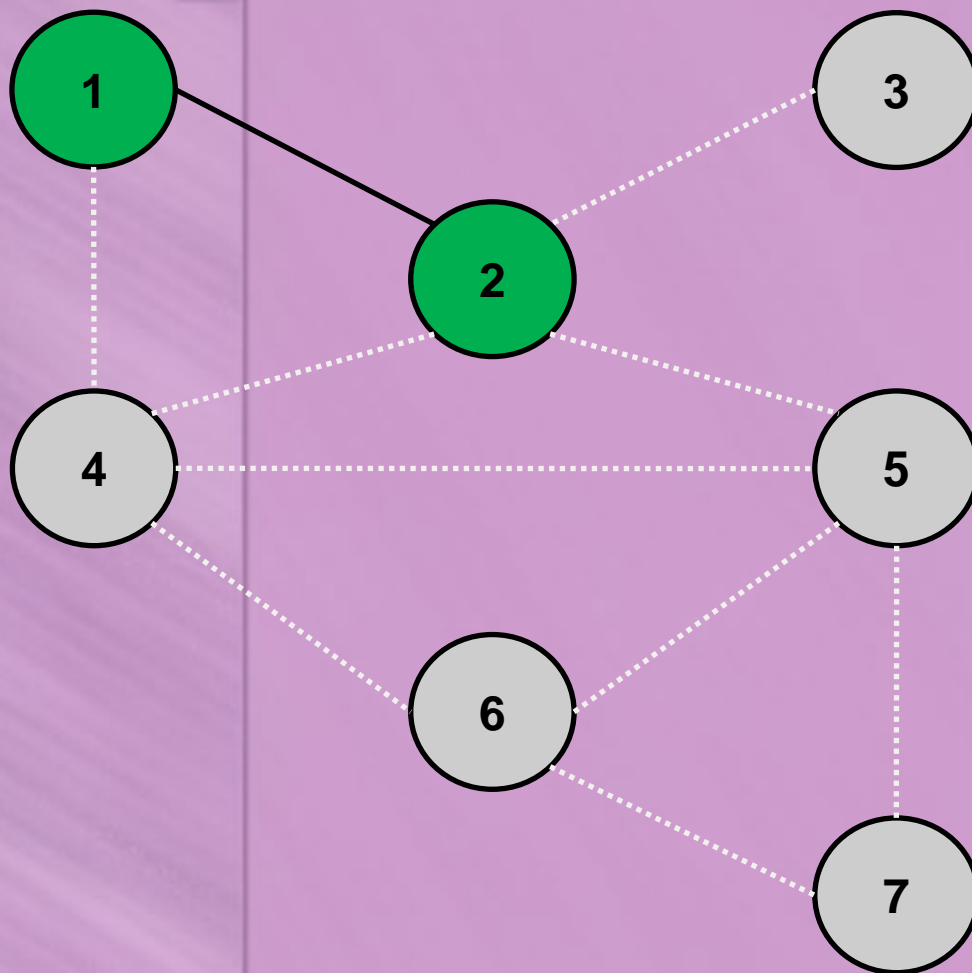
CLOSED

DFS - Ukázka



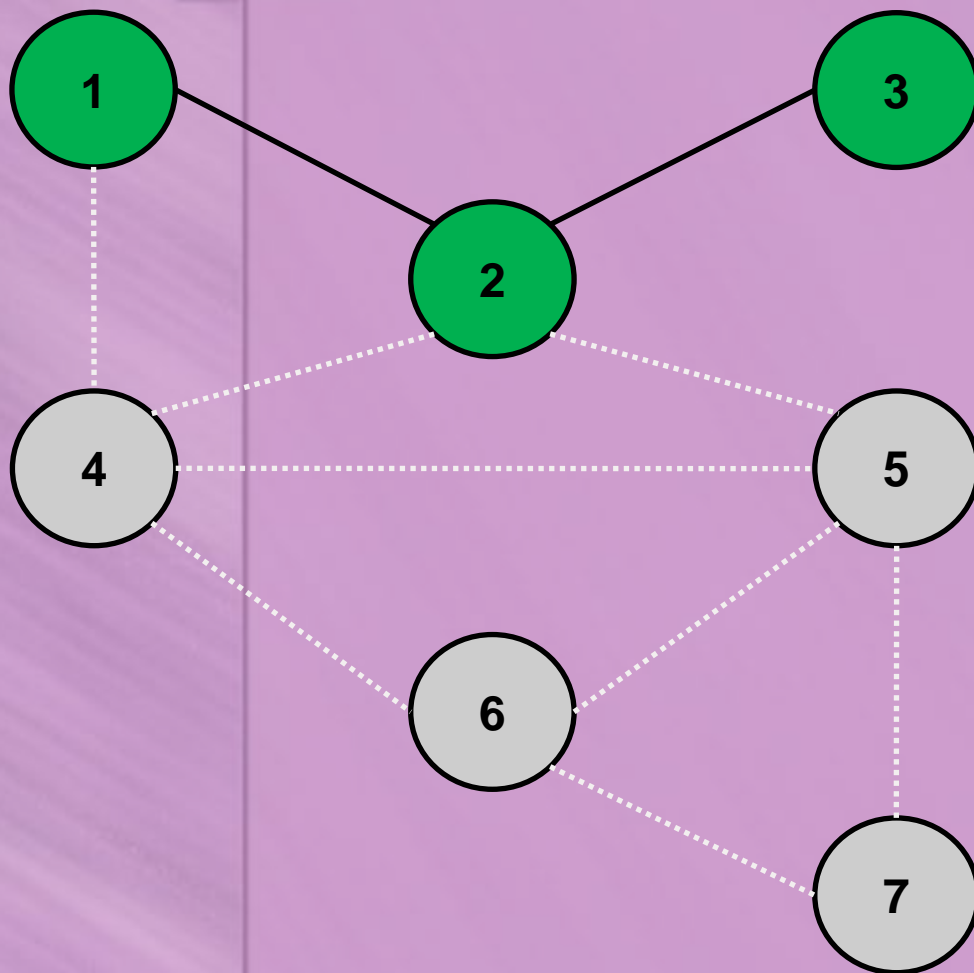
Otvíráme uzel 1

DFS - Ukázka



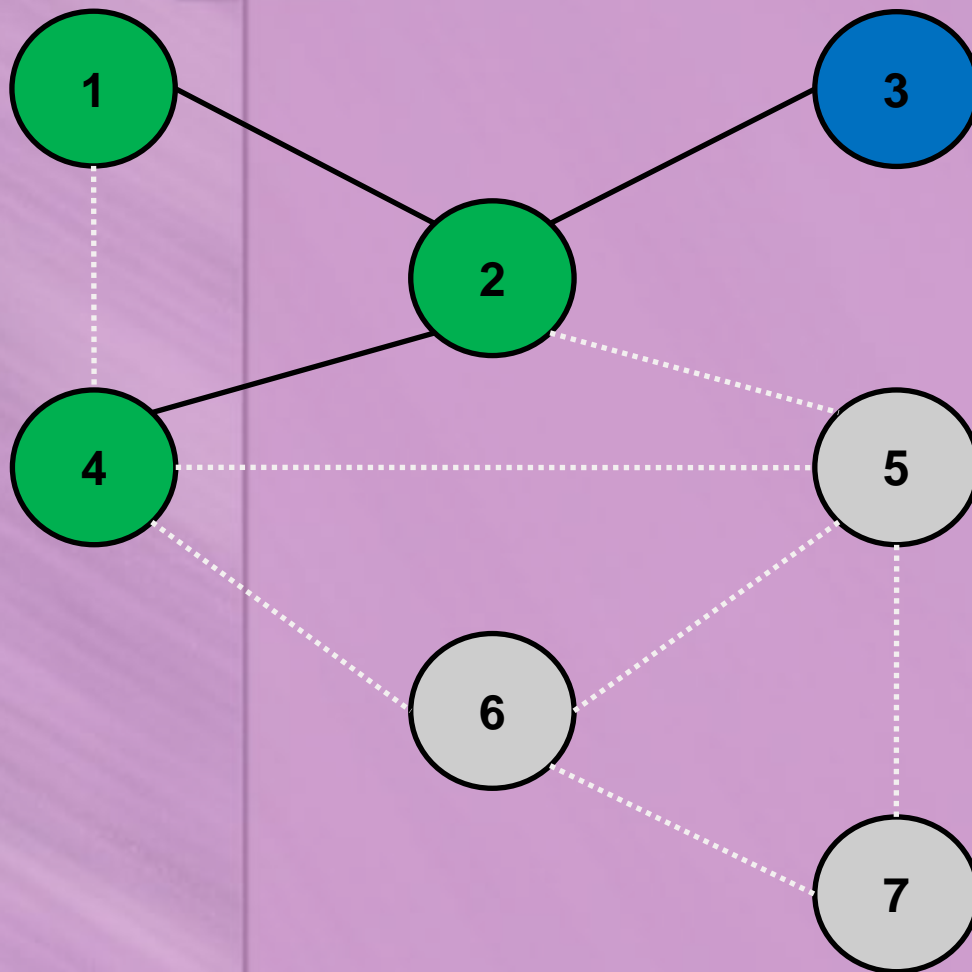
Otvíráme uzel 2

DFS - Ukázka



Otvíráme uzel 3

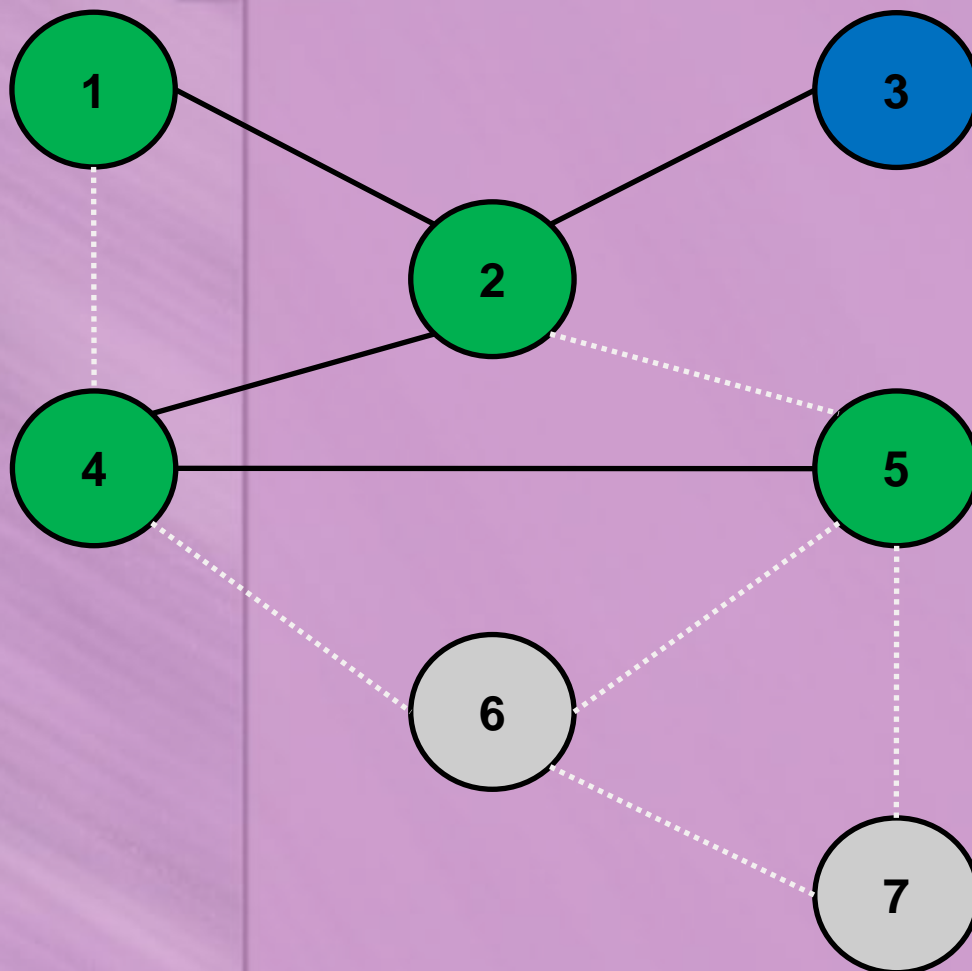
DFS - Ukázka



Uzavíráme uzel 3

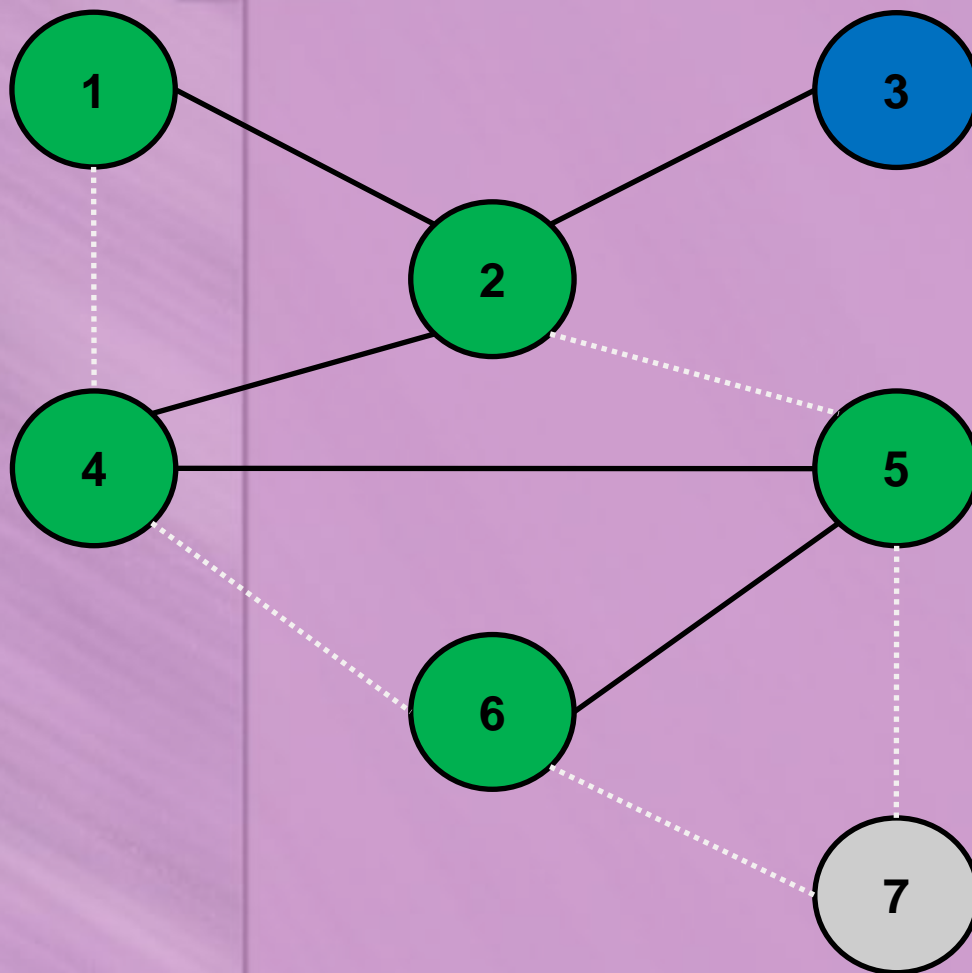
Otevíráme uzel 4

DFS - Ukázka



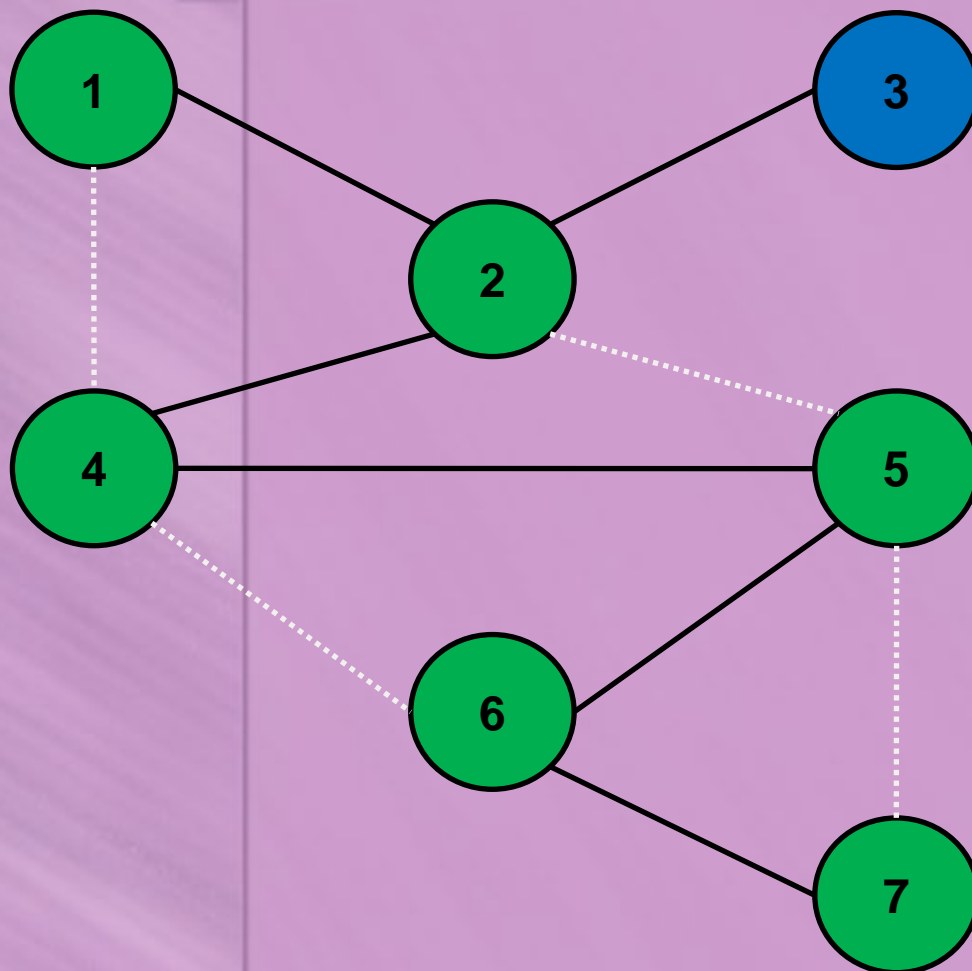
Otvíráme uzel 5

DFS - Ukázka



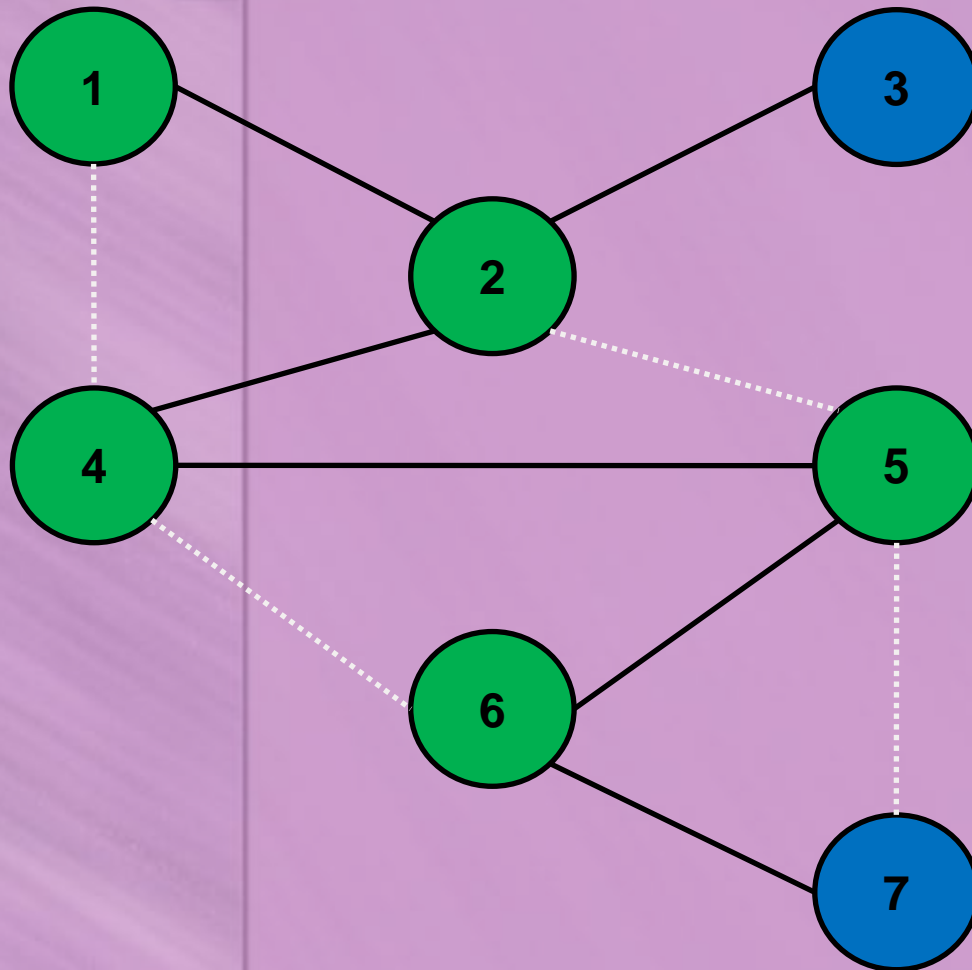
Otvíráme uzel 6

DFS - Ukázka



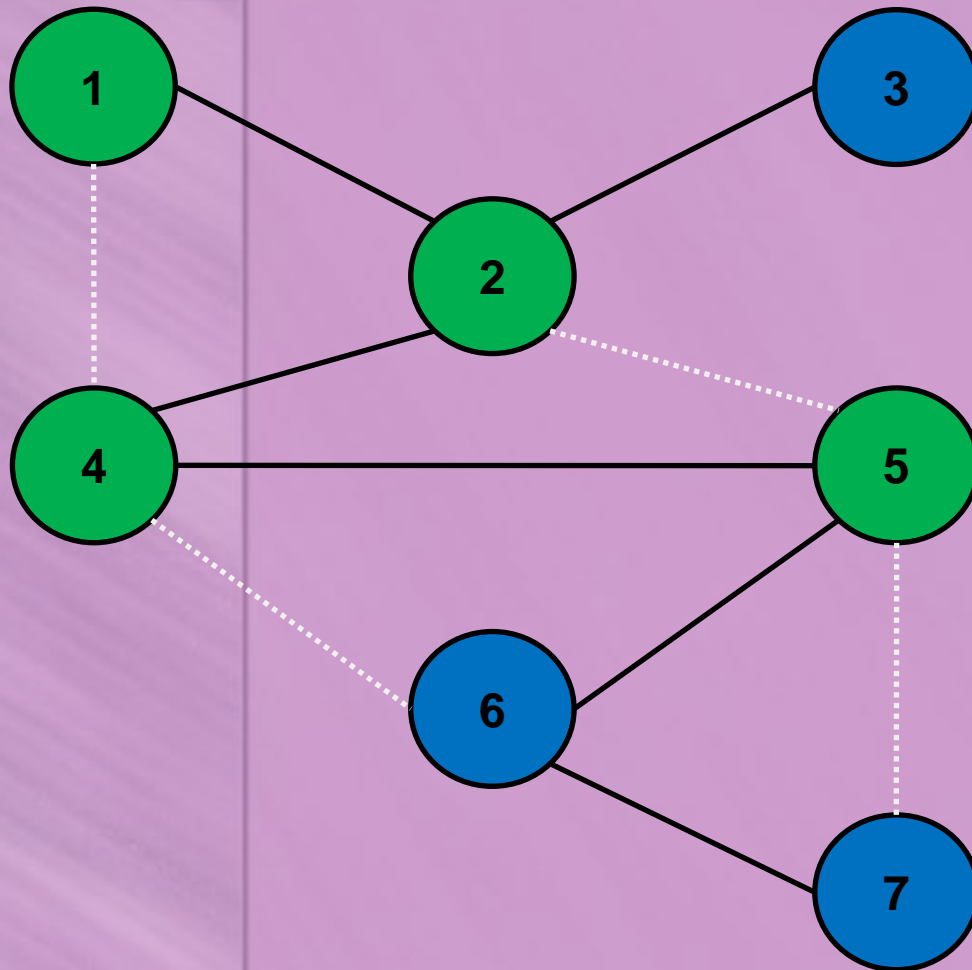
Otvíráme uzel 7

DFS - Ukázka



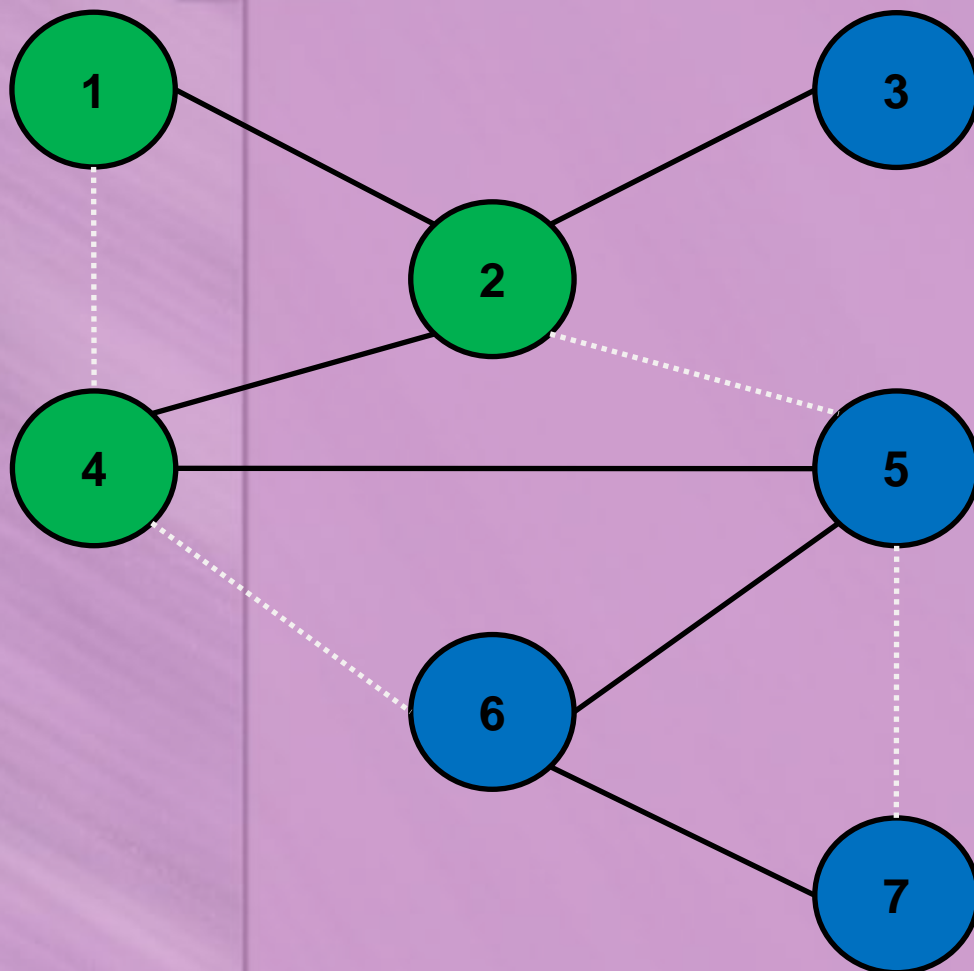
Uzavíráme uzel 7

DFS - Ukázka



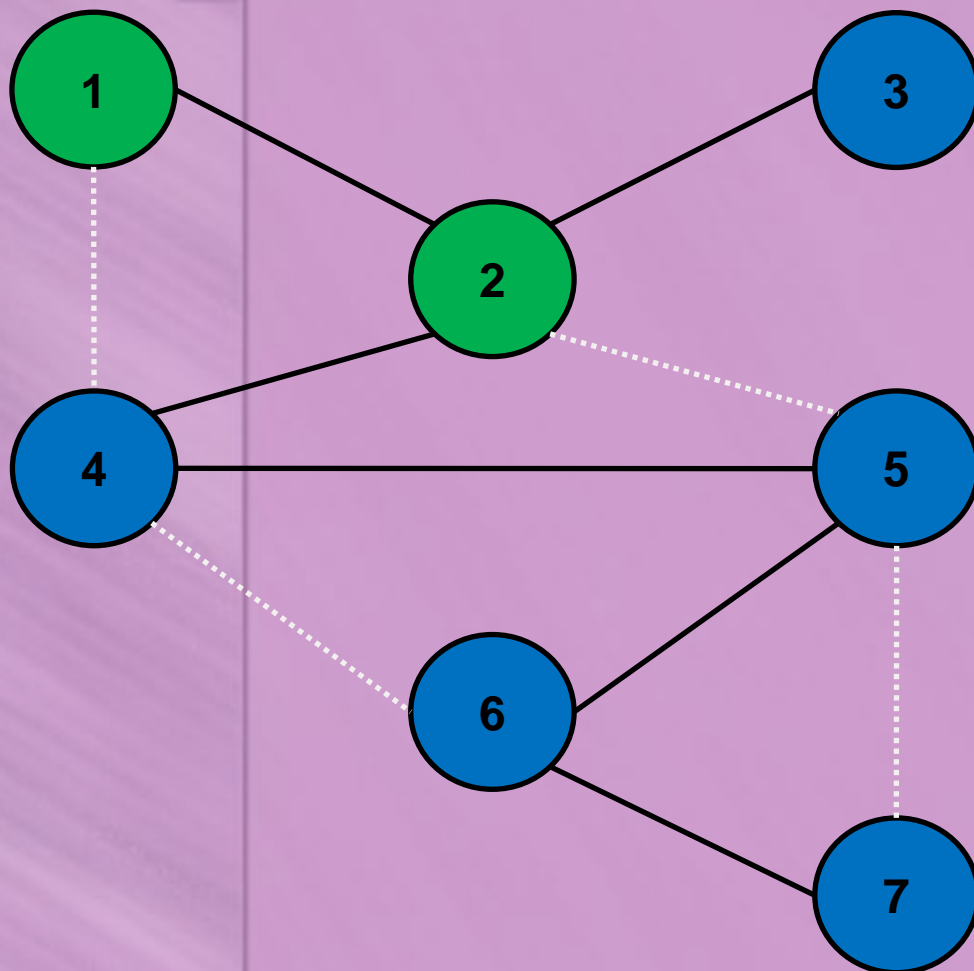
Uzavíráme uzel 6

DFS - Ukázka



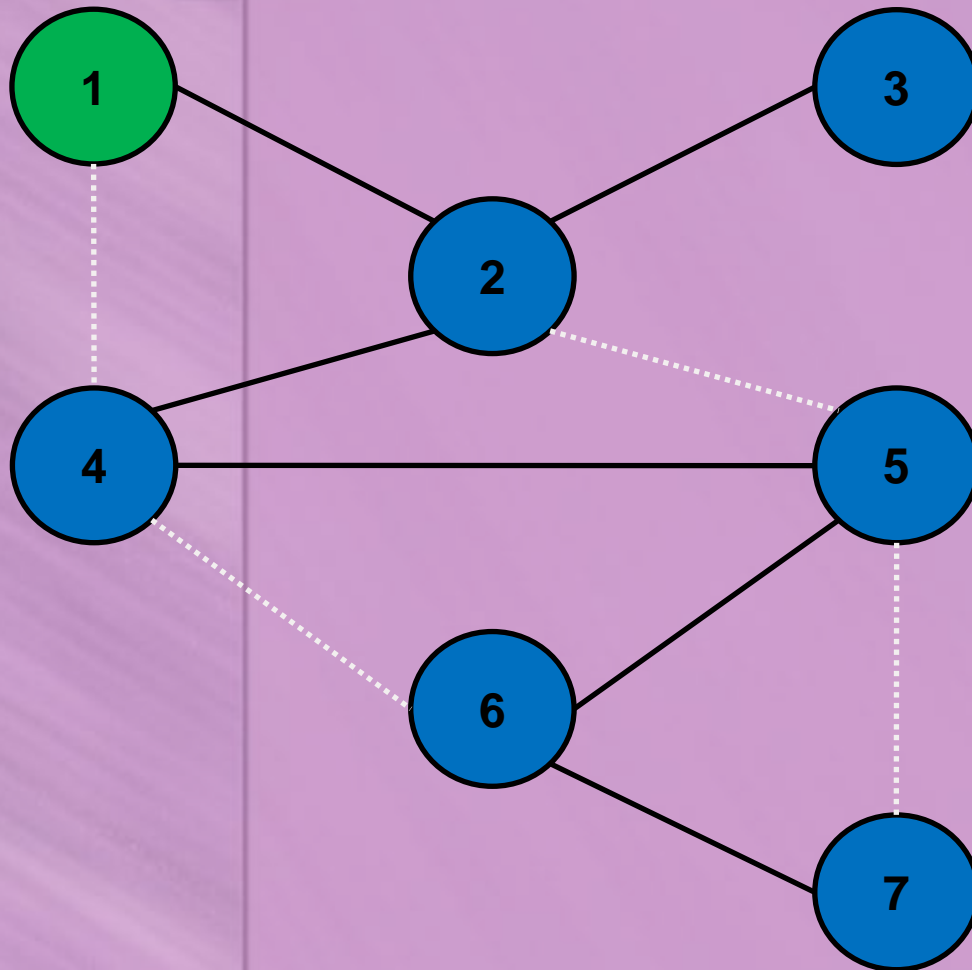
Uzavíráme uzel 5

DFS - Ukázka



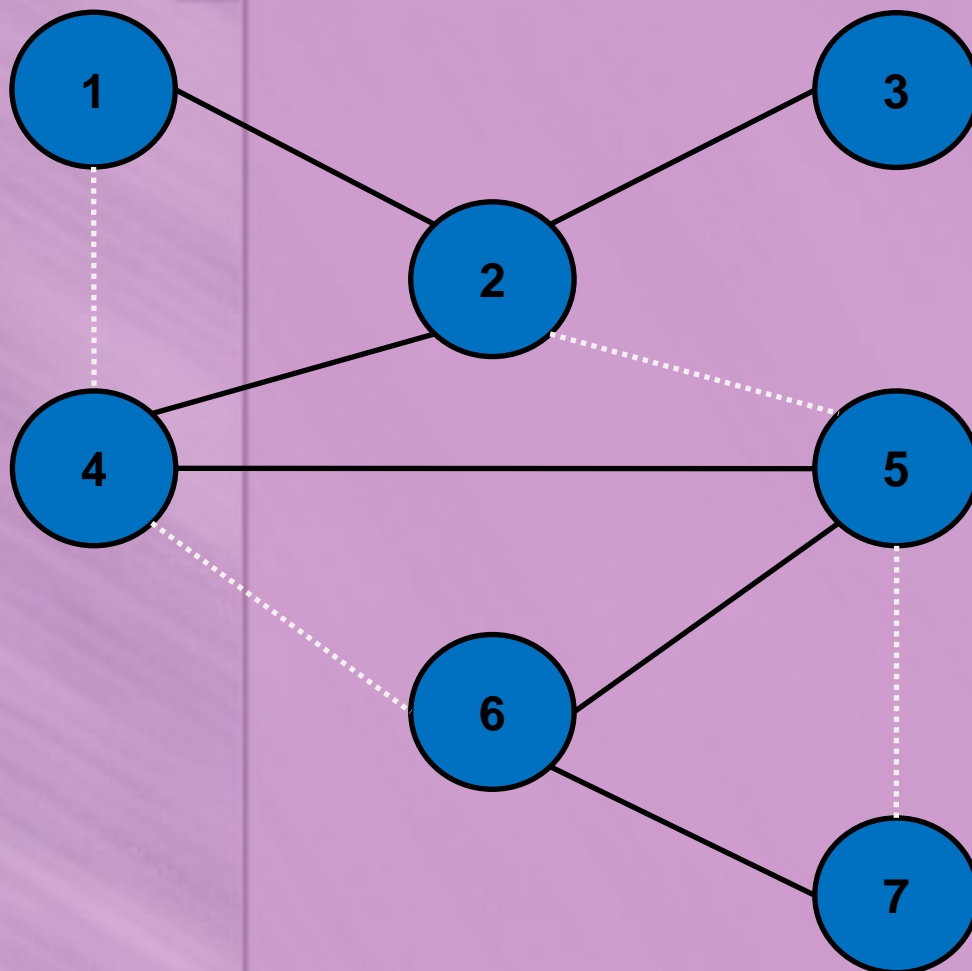
Uzavíráme uzel 4

DFS - Ukázka



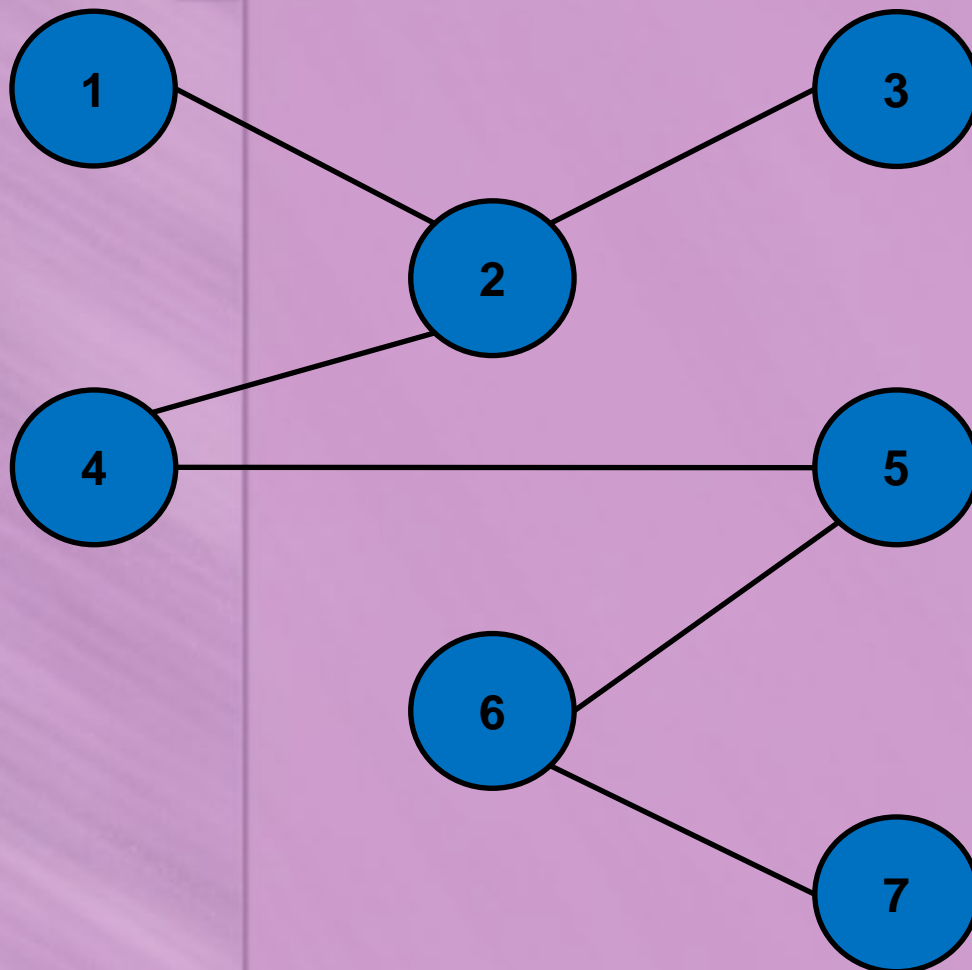
Uzavíráme uzel 2

DFS - Ukázka



Uzavíráme uzel 1

DFS - Ukázka



Algoritmus končí ve chvíli jsou všechny uzly uzavřeny.

Vznikl tzv. DF-Strom, nebo les.

Algoritmus najde cestu z výchozího uzlu do ostatních, ale tato nemusí být optimální.

DFS - Složitost

Celková složitost obou algoritmů je:

$$DFS(G) \approx BFS(G) \approx (V(G) + E(G))$$

- $V(G)$ je počet vrcholů v grafu G
- $E(G)$ je počet hran v grafu G



DOTAZY K VĚCI?

**POKUD NE, TAK DĚKUJI
ZA POZORNOST**