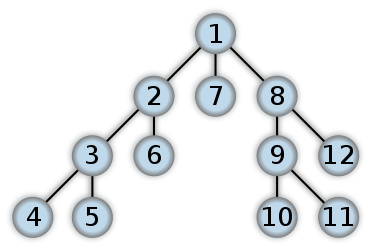
# 23. Back-track algoritmy (DFS, BFS)

## Back-track algoritmus

Zpětné vyhledávání; V množině hledají správně řešení 🡪 lze vygenerovat všechna řešení. Organizovaná hrubá síla. Opak hladového algoritmu 🡪 heuristika (zkušenost).

## DFS (Depth-first search)

* Prohledávání do hloubky
* Rekurzivní algoritmus
* Pokud existuje řešení, vždy ho najde, ovšem ne to nejkratší
* *Princip*:
  + Postupuje se stále dál od počátečního uzlu dosud neprozkoumaným směrem.
  + Když už to dál nejde, vrátí se pomocí backtrackingu a postupuje zase co nejdál.
  + Používá pro vrcholy v grafu následující stavy:
    - FRESH (Ještě nebyl objeven)
    - OPEN (Právě objeven)
    - CLOSE (Už byl prozkoumán)
* Př.: šachovnice; 8dam; eternity hra; cesta jezdce
* Lze implementovat pomocí zásobníku

## BFS (Breadth-first search)

* Prohledávání do šířky
* Postupuje systematicky (ve vlnách)
* Př.: navigace; nejrychlejší poskládání šachovnice
* Pokud existuje řešení, vždy ho najde (Nejkratší cestu)
* *Princip*:
  + Algoritmus začne v libovolném počátečním vrcholu.
  + Nejprve projde všechny sousedy startovního vrcholu, poté sousedy sousedů atd. až projde všechny dostupné vrcholy.
  + Pro vrcholy v grafu používá 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.
  + Každý krok si pamatuje předchozí
* Lze implementovat pomocí front

### https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Breadth-first_tree.svg/300px-Breadth-first_tree.svg.png

#### void DFS (Graph G) {

#### //Všechny uzly nastav jako FREH

#### for (Node u in U(G)){

#### stav[u] = FRESH;

#### }

#### for (Node u in U(G)){

#### if (stav[u] == FRESH) DFS-Projdi(u);

#### //Pro každý fresh zavolej DFS-projdi

#### }

#### void DFS-Projdi(Node u) {

#### stav[u] = OPEN; //Pokud je stav OPEN

#### for (Node v in Adj[u]){ //Vyhledej sousedy

#### if (stav[v] == FRESH) DFS-Projdi(v); //Pokud jsou FRESH; zavolej DFS-projdi

#### stav[u] = CLOSED; //Nastav CLOSE

#### }

#### void BFS (Graph G, Node s) {

#### //Všechny uzly nastav jako FREH (kromě počátečního)

#### for (Node u in U(G)-s){

#### stav[u] = FRESH;

#### }

#### stav[s] = OPEN; //Označ počáteční uzel jako fresh

#### Queue.Push(s); //Ulož ho do fronty

#### while (!Queue.Empty()) { //Dokud fronta není prázná

#### u = Queue.Pop(); //Vyzvedni uzel

#### for (v in Adj[u]) { //Najdi všechny sousedy

#### if (stav[v] == FRESH) { //Pokud mají fresh

#### stav[v] = OPEN; //Změň na open

#### Queue.Push(v); //Ulož do fronty

#### }

#### }

#### stav[u]=CLOSED; //Označ jako CLOSED

#### }

#### }