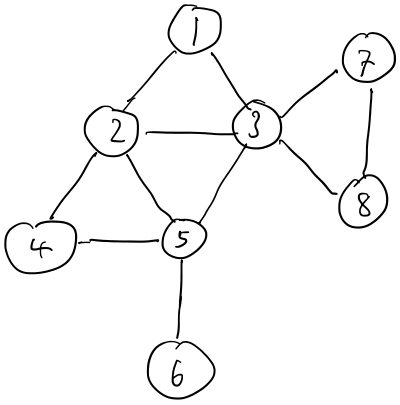
Lec06-BFS-DFS

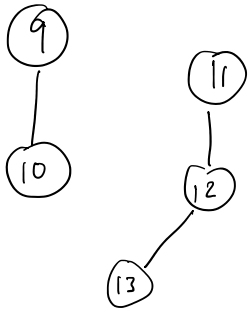
KT Ch. 3

Sunday, September 8, 2024 8:32 AM

Suppose we have a graph $G=(V,E)$

Can we find a path from a node s to a node t? (Connectivity)

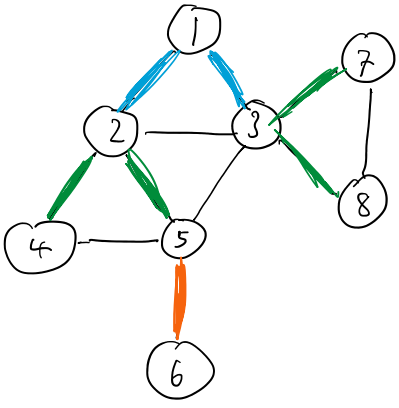


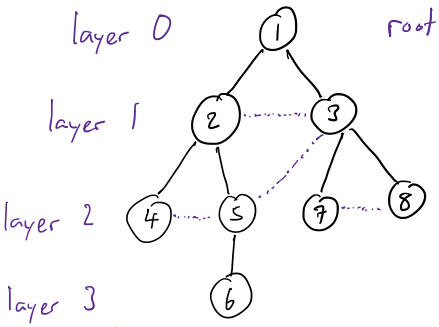


3 connected components

Breadth-first-search (BFS)

Explore outwards in layers defined by distance from root



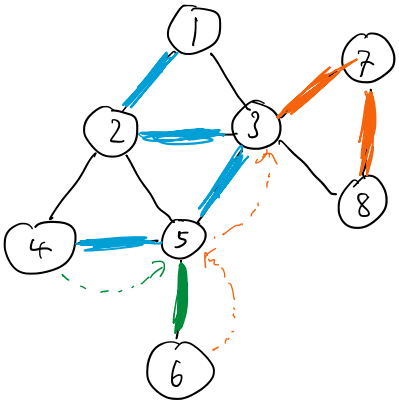


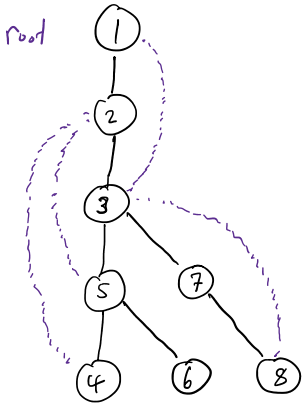
notice edges never jump layers

Thm If $(x,y) \in E$, then $|\text{layer}[x] - \text{layer}[y]| \leq 1$

Depth-First-Search (DFS)

Go as far as possible in a direction, and backtrack only as much as needed





Both BFS + DFS create search trees.

(y is a descendant of x)

x is an ancestor of y if the path from y to the root goes through x.

Thm If $(x,y) \in E$, then either x is an ancestor of y

Thm If $(x,y) \in E$, then $|\text{layer}(x) - \text{layer}(y)| \leq 1$.

proof. Suppose not.

WLOG, $\text{layer}(x) < \text{layer}(y) - 1$

But all neighbors of x will be found by $\text{layer}(x) + 1$.

$\Rightarrow \text{layer}(y) \leq \text{layer}(x) + 1$

$\Rightarrow \text{layer}(x) \geq \text{layer}(y) + 1$.

Contradiction. $\rightarrow\!\leftarrow$

Thm If $(x,y) \in E$, then either x is an ancestor of y or y is an ancestor of x in the DFS tree T_G.

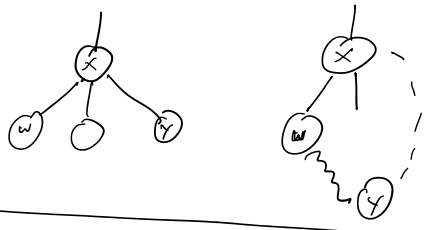
proof. WLOG, x is reached first in DFS.

(So y is unexplored when x is reached)

All nodes explored between initially reaching x and leaving x for the last time. (by backtracking past x) are descendants of x in T_G.

y must be explored before leaving x for last time because $(x,y) \in G$.

(though it might be explored thru a child)



Both BFS & DFS are tree-growing algorithms:

- Let T be the current tree
- Maintain list of frontier edges that go from T to $V-T$.
- Repeatedly choose a frontier edge (somehow) and add it to T.

Tree-growing pseudo-code

```
Tree Growing (graph G, vertex v, func nextEdge):
    T = ({v}, ∅)
    S = set of edges incident to v
    While S ≠ ∅:
        e = nextEdge(G, S)
        T = T + e        (and add relevant node)
        S = updateFrontier(G, S, e)
    return T
```

runtime depends on nextEdge & updateFrontier

BFS/DFS are $O(|V| + |E|)$ because each edge/node is processed once in updateFrontier.

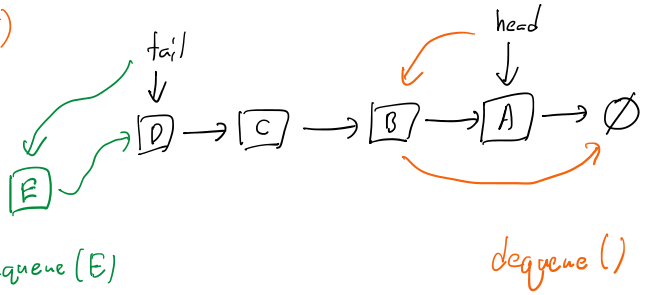
Have we seen other examples of tree-growing algorithms before?

What's nextEdge for BFS? Select least-recently discovered edge. Queue

What's nextEdge for DFS? Select most-recently discovered edge. Stack

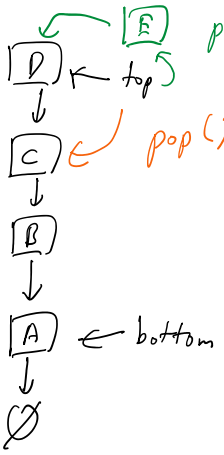
Queue (for BFS)

FIFO (First-In-First-Out)



Stack (for DFS)

LIFO (Last-In-First-Out)



Alternate recursive DFS

```
procedure DFS(G, u):
    while ∃ v an unvisited neighbor of u:
        mark v visited
        DFS(G, v)
```