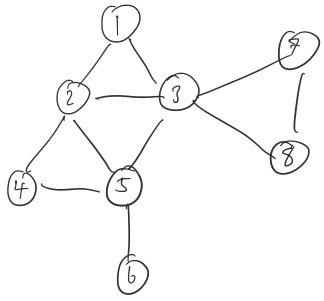
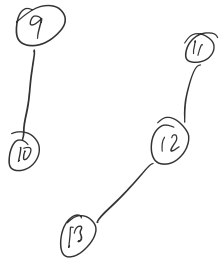
Lec06a-graph-traversal

KT Ch. 3

Thursday, September 7, 2023 11:54 PM

Suppose we have a graph $G = (V, E)$.
Can we find a path from a node s to a node t? (Connectivity)

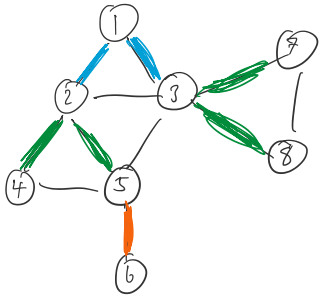


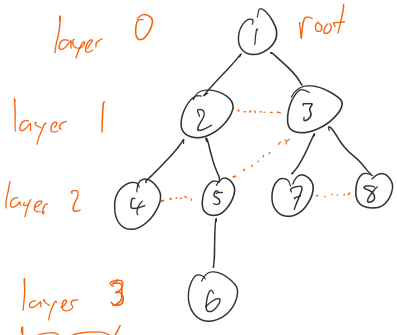


3 connected components

Breadth-first search (BFS)

Explore outward in layers defined by distance from root.





distance from root

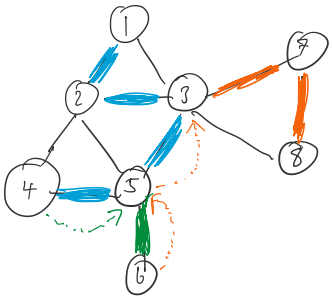
notice: edges never jump layers

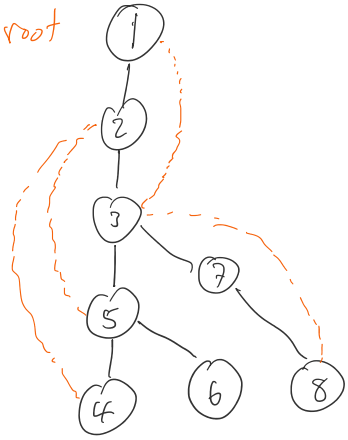
Thm If $(x, y) \in E$, then $|\text{layer}(x) - \text{layer}(y)| \leq 1$.

proof Suppose not.
WLOG, $\text{layer}(x) < \text{layer}(y) - 1$
But all neighbors of x will be found by $\text{layer}(x) + 1$.
$\Rightarrow \text{layer}(y) \leq \text{layer}(x) + 1$

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 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)

found by layer(x)+1.

$\Rightarrow$ layer(y) $\leq$ layer(x) + 1

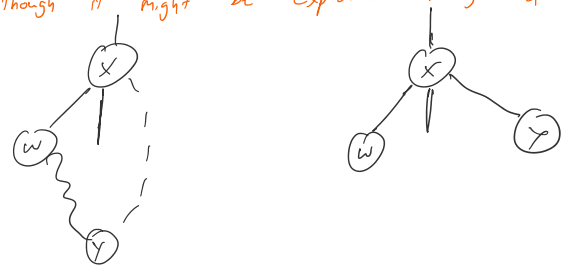
$\Rightarrow$ layer(x) $\geq$ (layer(y) − 1.

$\Rightarrow$ contradiction.

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 the last time because $(x, y) \in G$.

(though it might be explored through 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 pseudocode:

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

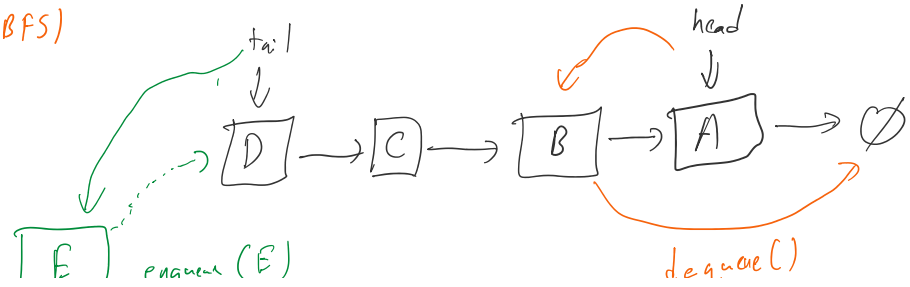
Have we seen other examples of tree-growing algorithms already?

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

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

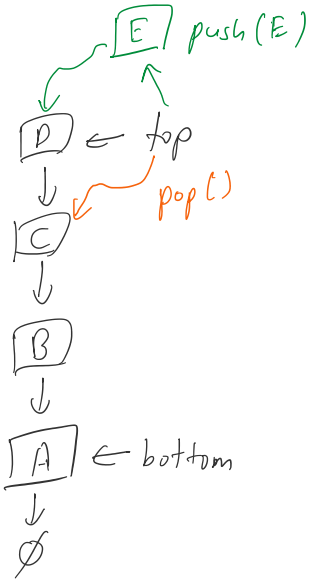
Queue: FIFO (first-in-first-out)

(for BFS)



Stack: LIFO (last in, first out)

(for DFS)



Alternate recursive DFS:

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