CS21 LECTURE #10

Program #5:

High: 100/100 Mean: 74 median: 76

Delete for Assignment Binary Search Tree:

(will not be asked to code this on final exam)

3 Cases for Deletion:

if the node to delete has

EZ
- Case 1) no children → delete it!
- Case 2) one child → splice out node, delete it!

HARD
- Case 3) two children → splice out successor node, overwrite the node to delete, then delete successor.

reference funct calls delete...

```
private delete(node n) {
    if (n.left == null || n.right == null) {
        t = n;          // one child case
    } else {
        t = successor(n); // two child case
    }

    if (t.left != null) {
        x = t.left;
    } else {
        x = t.right;
    }
}
```

n = Node of value to delete

t = target

x = child of t that will be lost when t is removed unless we re-attach it.

If t has no children, then x = null, otherwise x is that child.

```
    if (x != null) {
        x.parent = t.parent;
    }
```

If x is a node, fix its parent pointer

```
    if (t.parent == null) {
        root = x;
    } else if (t == t.parent.left) {
        t.parent.left = x;
    } else {
        t.parent.right = x;
    }

    if (t != n) {
        // copy t's data to n
    }

    delete t;
}
```

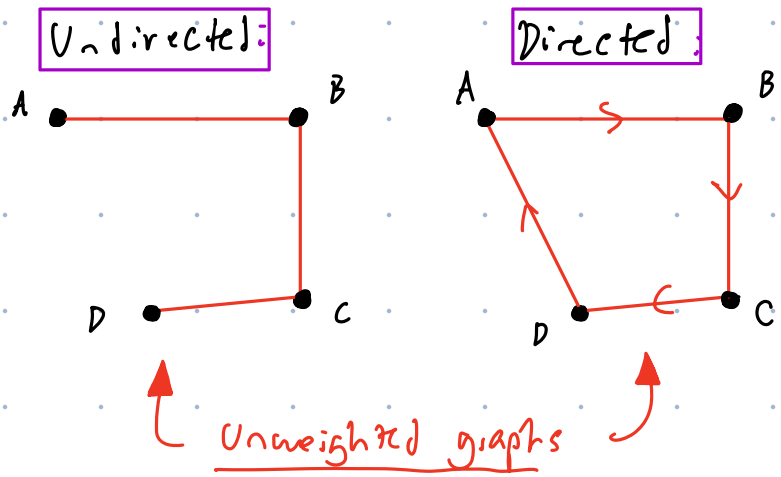
BREADTH-FIRST SEARCH

Basic Search Algorithm for a graph.

Purpose: TO learn about the structure of the graph.

Graph = collection (set) of nodes (vertices) & edges.

$G = (V, E)$



How do we represent graphs in a computer???

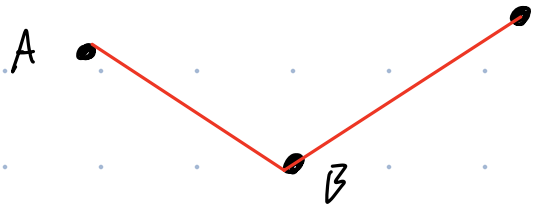
3 methods, 2 standard, 1 custom

* Adjacency list → A list for each node that lists adjacent nodes on a node!
* Adjacency matrix → Defined below...
* custom method → used in our BFS algorithm ...

Adjacent Nodes: → 2 nodes w/ an edge between them

A & B are adjacent!

Reachable Nodes: → Have a path between the two nodes.

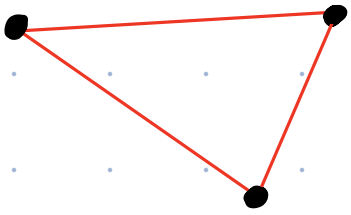


Path: $A \to B \to C$!

$C \to B \to A$!

DAG → Directed Acyclic graph!

Cycle → A path from A to B where $A = B$



Sparse Graph → $\sim O(n)$

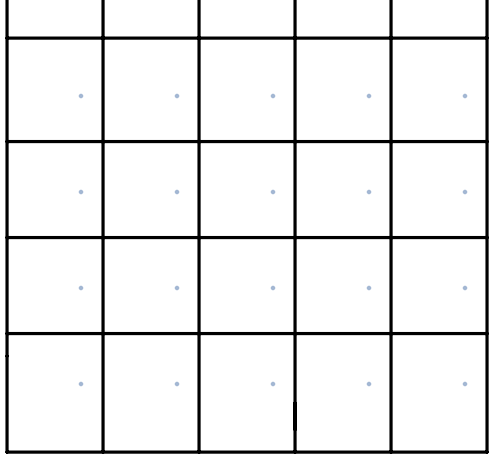
A full/complete graph. → $\frac{n(n-1)}{2} \simeq O(n^2)$

A graph where duplicate edges are <u>not</u> allowed. Our edges should be unique

Adjacency Matrix:

Dest all nodes:

$O(n^2)$ storage


all
nodes