Date 2025.04.23

*Don't have to code delete on exam for binary search tree

H.T.
↳ Minor Errors:
  ↳ Privacy Leaks ↳ Duplicate Code ↳ #includes

Not inheritantly bad, but is bad when it happens accidentally

When public function should have been private

↳ Major Errors:
  ↳ Records 'lost'
    • 'input' • 'output' • insert, search, delete

Steve wants a CLEAN disjoint set
*Squaring happens in maze

Won't test cases < 3
Not a req for your program to generate a maze
Does have to be random, will test the same case multiple times to see if they are different

Terminating/ending condition is when disjoint set is one
* If maze finishes & maze parts are unfinished that means numSets--; could have be called too early

3 Cases For Deletion

- If the node delete has ~~two children~~

EASIER
- no children – delete it!
- one child – splice ~~it~~ out node, delete it (Mid Level Difficulty)

TRICKIER
- two children – splice out successor node, overwrite the node to delete (then delete successor).

NOT ON FINAL!!!

```
private delete (node n) {
if(n.left == null || n.right == null)
    t = n; //one child or no child case
else
    t = successor (n); //two child case
if (t.left != null)
    x = t.left;
else
    x = t.right;
if (x != null)
    x.parent = t.parent;
if (t.parent == null)
    root = x;
else
    if(t = t.parent.left)
        t.parent.left = x;
    else
        t.parent.right = x;
if(target != node) //copy node target (t) into node n
//delete target delete t;
}
```

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

If x is a node, fix its parent pointer

n
node of value to delete

t
target

x
child of t that will be lost when t removed unless we reattach it.

Date

BREADTH FIRST SEARCH (BFS)

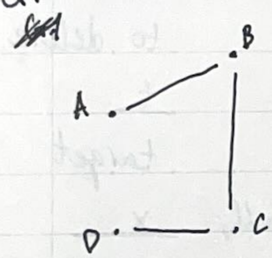
- A basic search algorithm for graphs
- Purpose of a graph search is to LEARN about the structure of the graph

GRAPH

- Collection (set) of <u>nodes</u> (vertices) and <u>edges</u>

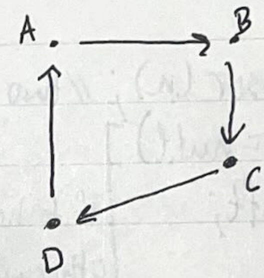
  $G = (V,E)$ ← Defined as

G1 Undirected



can be:
- unweighted
  or
- weighted

G2 Directed



HOW TO REPRESENT GRAPHS IN COMPUTER

- 3 Methods, 2 Standard, 1 custom
  - Adjacency List
  - Adjacency Matrix
  - Custom Method

★ dag
↳ directed acyclic graph
most common

ADJACENT – NODES

- Two nodes that have an edge between them.

VS

REACHABLE — NODES

- Have a <u>path</u> between the two nodes

Date

CYCLE
- Path from a to b where a == b

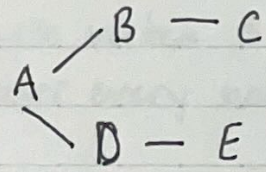
ADJACENCY LIST
- List for each node
  - Lists the adjacent nodes

A LIST FOR G1
- A) B
- B) A, C
- C) B, D
- D) C

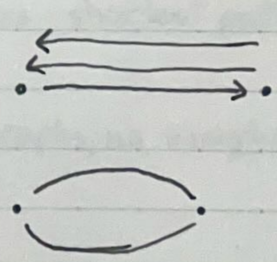
A LIST FOR G2
- A) B
- B) C
- C) D
- D) Ø

• Adj. list - common method to rep. graph
Especially usefull in sparsegraphs

SPARSE GRAPH
- Full/Complete graph
- Graph where number of edges is significantly fewer than the max possible number of edges between its vertices



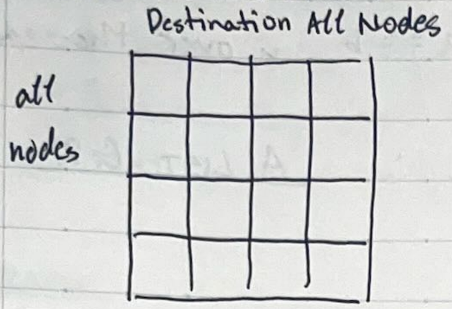
MULTIGRAPHS
- Duplicate Edges ↳ Disallowed] Unique Edges ONLY



HYPERGRAPH
- Multiple-node edges
  ↳ Disallowed

Date

ADJACENCY MATRIX



Time Complexity $O(n^2)$
↳ For storage

ANY GRAPH:

min: 0

max: $\frac{n(n-1)}{2}$ undirected

↳ $n(n-1)$ directed

$O(n^2)$ complexity