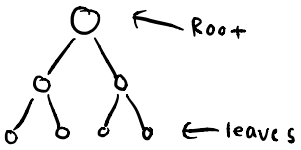
Today: Trees!

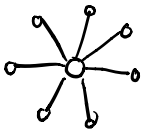
- Mt 2 Review Session 3/29
- Mock Mt Review 3/30
- Binary Trees
- BST
- Traversals
- Heaps

Trees:

- sets of minimally connected vertices (nodes) & edges
  - one way to get to any vertex, no cycles, no loops / self edges

Rooted Trees



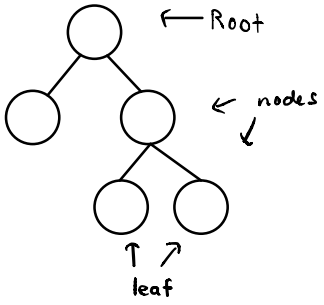


↙ Also a tree

Binary Trees:

- has Root
- has left, right, node val
  - left: Node
  - right: Node
  - node val: int

Trees



What is it? → true for all trees

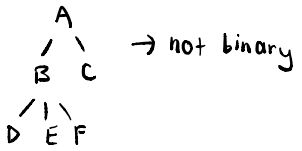
- Set of connected vertices / nodes
- minimally connected using edges ↓ what makes a tree, a tree
- one important constraint: exactly one path b/t any two nodes/vertices
- Most trees we will focus on are rooted

Many, many types:

- CS61A was like the Interface / ADT, now, we do implementations

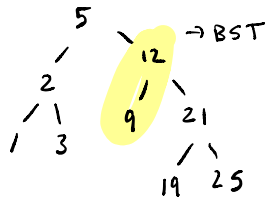
Binary Search Tree (BST) : Standard, Basic.

- every node in a binary tree has 0, 1, 2 children

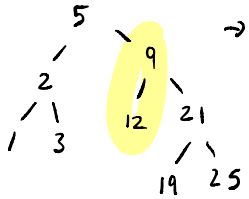


→ not binary

- every left subtree is less than you, every right subtree is greater



→ BST



→ not BST

What problem does this raise? → what to do about duplicates?

⇝ Binary Search very easy!

∴ So search is done in $\Theta(\log N)$ time.

Traversals : Visit nodes in particular order

1.) Breadth First Search (level order)

- top to bottom, left to right
- check entire level (all siblings) before moving on.

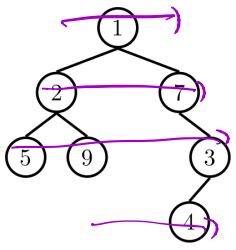
2.) Depth First Search

- Deeper nodes before shallower ones
- all the way down
- 3 types : Lets do 3.1 !

1.) Pre-order
MLR

2.) Post-order
LRM

3.) In-order
LMR



Stack (LIFO)

Pre-Order : 1 - 2 - 5 - 9 - 7 - 3 - 4

Post-Order : 5 - 9 - 2 - 4 - 3 - 7 - 1

In-order : 5 - 2 - 9 - 1 - 7 - 4 - 3

Level-order : 1 - 2 - 7 - 5 - 9 - 3 - 4

Queue (FIFO)