Lecture-08: Binary Search Trees

Overview

1. Motivation
2. Binary Trees + Terminology
3. Basic Tree methods

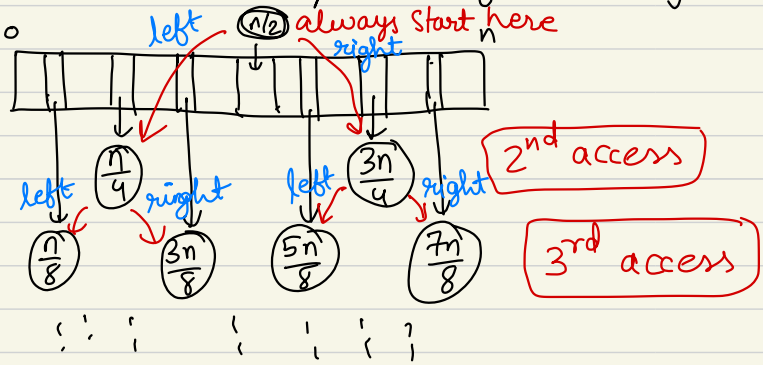
Next Week

1. Midterm 1 [1hr → take home → more info on weekend]
2. Assignment 3 (shorter) [Priority Queues]

Problem Search is fast (O(log n)), but add/remove were only O(n).

⇒ **Question:** Can we do all 3 operations in O(log n) time?

⇒ **Question:** What is "access pattern" of the binary search?



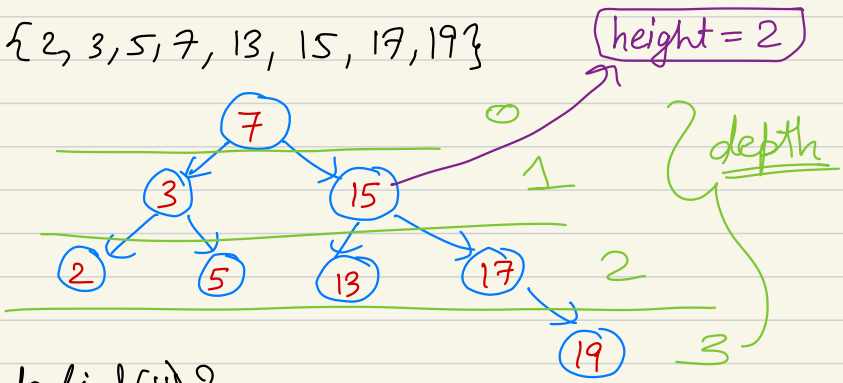
Idea: Store elements in nodes

- each node stores value and
- also "next" nodes that would be accessed in binary search w/ node search as midpoint.

Each node has stored: left child, right child, parent

- **Properties:**
  - going left → smaller values
  - going right → larger values

eg $S = \{2, 3, 5, 7, 13, 15, 17, 19\}$



- How to find(11)?

Sol
- start @ 7 (top = "root")
- $11 > 7 \Rightarrow$ go right
- read 15, $11 < 15$ so go left.
- → read 13, $11 < 13$

  no place to go → 11 not there!!

- How to add(11)?

Sol
- → imitate find until we see 11 is not there.
- → make new node storing 11, attach where we failed to find 11.

⊙ How to remove (2)?
- find(2) then remove node!

⊙ remove (15)?
- replace it with the immediate smallest or immediately largest element storing node.

⊛ Nomenclature ("rooted" binary tree)

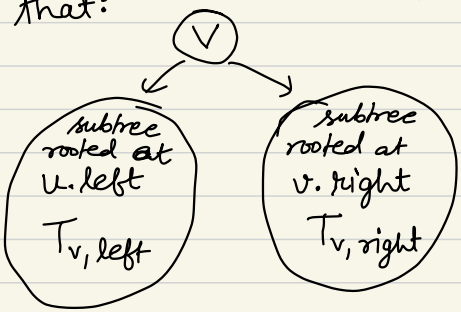
A binary tree consists of a collection of nodes with a single distinguished node called the root.

Each node has → parent node (except root's parent is null)
→ left child node (possibly null)
→ right child node (possibly null)

⇒ Must Satisfy:
⊙ Node "u" is a child of Node "v", then "v" is "u"'s parent
⊙ every node has root as ancestor (parent of parent of parent...)

⊛ Terminology
- Root: node without parent.
- Leaf: node without children.
- Internal: node which is not a "leaf".
- Depth: length of path from node to "root".
- Height: length of longest downward path from node to "leaf".
- Height of the "root" is the depth of tree.
- Subtree: rooted at node= tree of node's descendents with respect to u as a root.

A BST is a tree where each node has an associated value $u.x$ with comparable values with property that:



if u in $T_{v,left}$ then $u.x < v.x$

if w in $T_{v,right}$ then $v.x < w.x$

Given a BST.

- how to find(y)?
  - start at root u
    - if $(u.x).equals(y)$ — return $u.x$
    - if $(u.x) > y$ — go to **left** (if there) otherwise, fail.
    - if $(u.x) < y$ — go to ~~right~~ (if there) otherwise, fail
- how long does this take?
  - $O(\text{depth of tree}) = O(\text{height of root})$

Removal method on Tuesday...