Binary Search Trees

Data Structures and Algorithms

Warm Up

From CSE 143:

- what is a “binary tree”
- how do you write code to build a tree from scratch?
- how do you write code to traverse an existing tree?
- how do you write code to change an existing tree?
- What is the runtime to traverse a tree and print out every node?

Socrative:

www.socrative.com

Room Name: CSE373

Please enter your name as: Last, First

Storing Sorted Items in an Array

get() – O(logn)

put() – O(n)

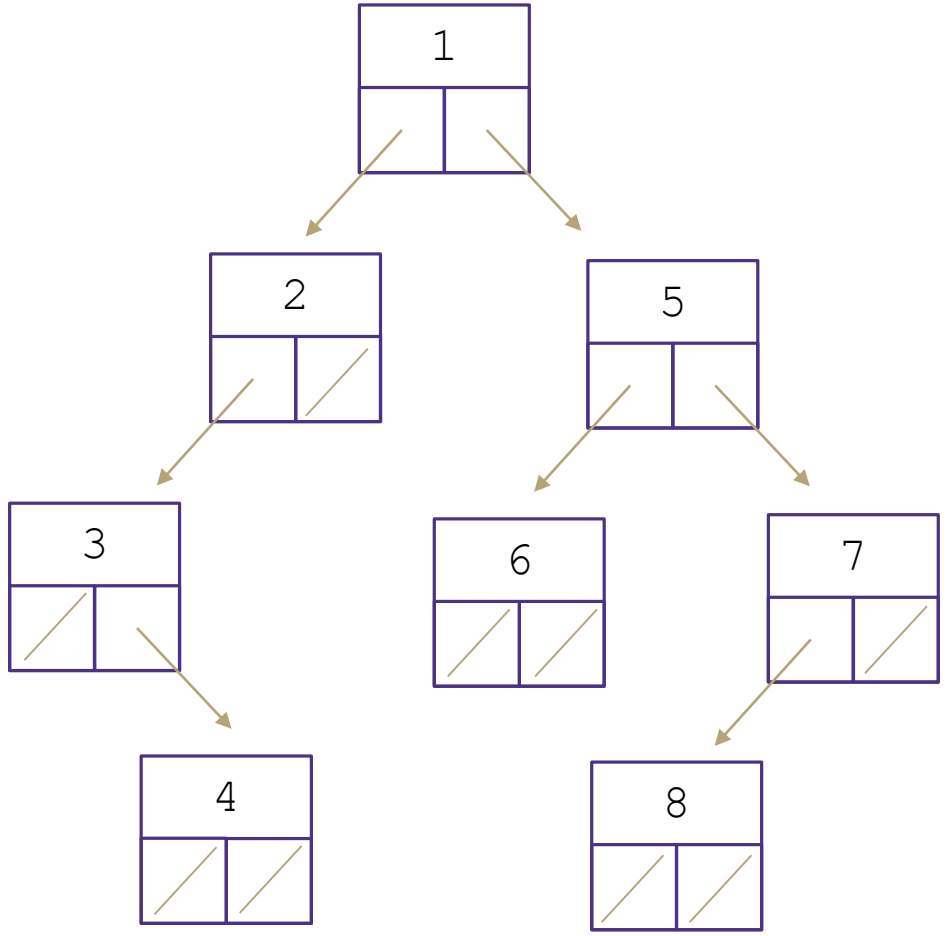
remove() – O(n)

Can we do better with insertions and removals?

Trees!

A tree is a collection of nodes

- Each node has at most 1 parent and 0 or more children



Root node: the single node with no parent, "top" of the tree

Branch node: a node with one or more children

Leaf node: a node with no children

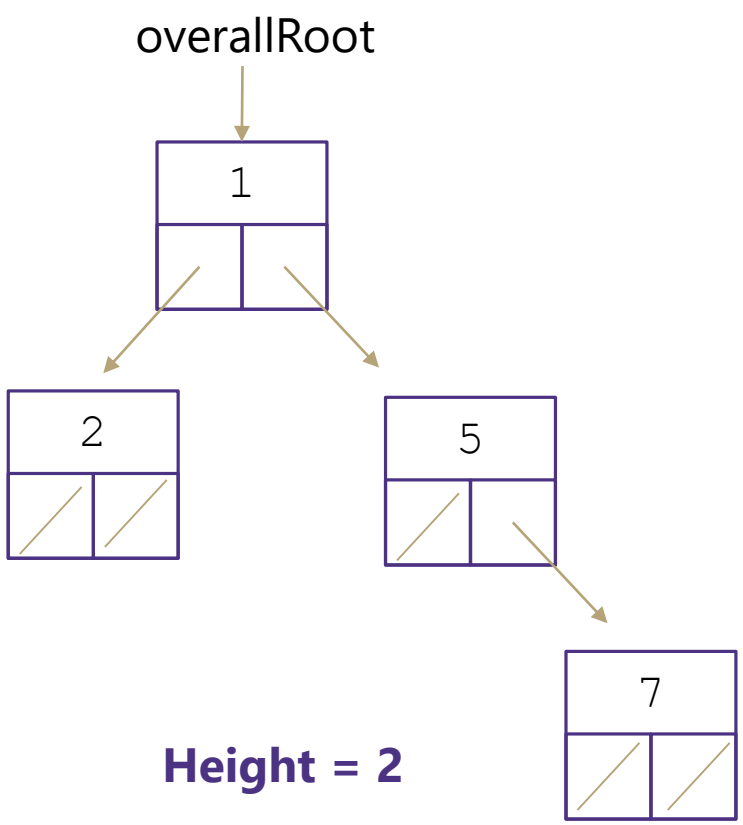
Edge: a pointer from one node to another

Subtree: a node and all it descendants

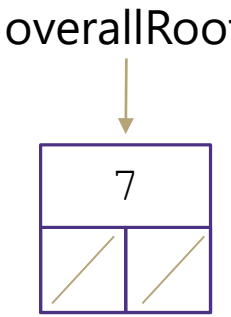
Height: the number of edges contained in the longest path from root node to some leaf node

Tree Height

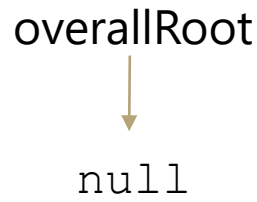
What is the height of the following trees?



Height = 2



Height = 0



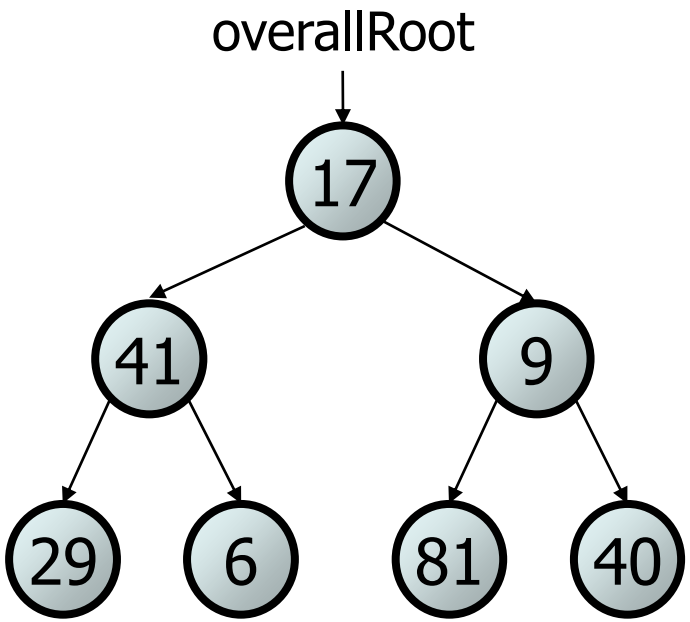
Height = -1 or NA

Traversals

traversal: An examination of the elements of a tree.
- A pattern used in many tree algorithms and methods

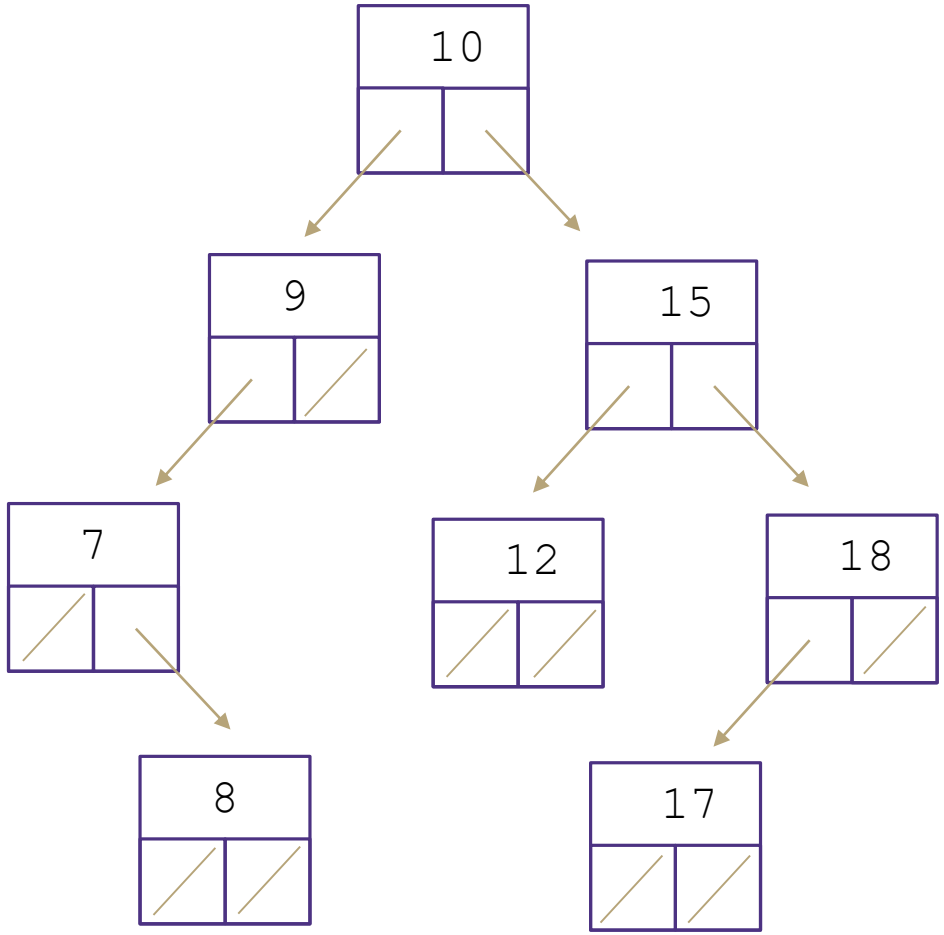
Common orderings for traversals:
- **pre-order**: process root node, then its left/right subtrees
  - `17 41 29 6 9 81 40`
- **in-order**: process left subtree, then root node, then right
  - `29 41 6 17 81 9 40`
- **post-order**: process left/right subtrees, then root node
  - `29 6 41 81 40 9 17`

Traversal Trick: Sailboat method
- Trace a path around the tree.
- As you pass a node on the proper side, process it.
  - pre-order: left side
  - in-order: bottom
  - post-order: right side



Binary Search Trees

A **binary search tree** is a binary tree that contains comparable items such that for every node, all children to the left contain smaller data and all children to the right contain larger data.



Implement Dictionary

Binary Search Trees allow us to:

- quickly find what we're looking for
- add and remove values easily

Dictionary Operations:

Runtime in terms of height, "h"

get() – O(h)

put() – O(h)

remove() – O(h)

What do you replace the node with?

Largest in left sub tree or smallest in right sub tree