CS 225

#16: Binary Search Tree

February 21, 2018 · *Wade Fagen-Ulmschneider*

Traversal vs. Search:

- **Traversal** visits every node in the tree exactly once.
- **Search** finds one element in the tree.

Breadth First Traversal + Search:

Depth First Traversal + Search:

Runtime Analysis on a Binary Tree:

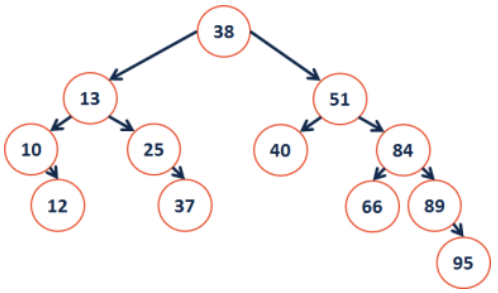
- Find an element: Best case? Worst case?
- Insertion of a sorted list of elements?
  Best case? Worst case?
- Running time bound by?

Dictionary ADT

Dictionary.h

```
3
4
5  class Dictionary {
6    public:
7
8
9
10
11
12
13
14   private:
15
16
17 };
```

A Searchable Binary Tree?



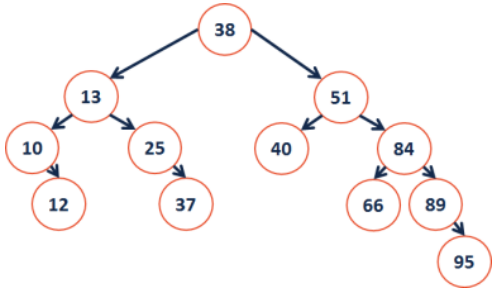
Binary Search Tree Property:

Finding an element in a BST:

BST.cpp

```
_____ _find
    (TreeNode *& root, const K & key) const {




}

_____ find(const K & key) {


}
```

Inserting an element into a BST:

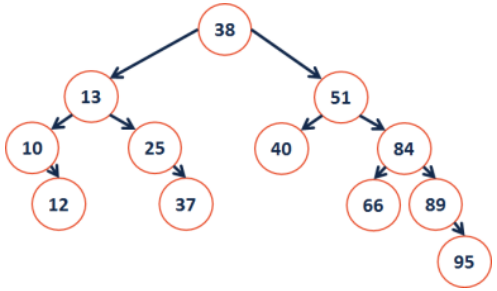


BST.cpp

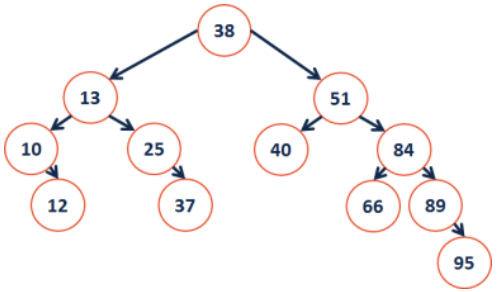
`template <class K, class V>`
`void BST::_insert(TreeNode *& root, K key, V value) {`
 	
`}`

Running time? _____ **Bound by?** _____

What if we did <u>not</u> pass a pointer by reference?



Removing an element from a BST:



`_remove(40)`

`_remove(25)`

`_remove(10)`

`_remove(13)`

One-child Remove	Two-child remove

BinaryTree.cpp

`template <class K, class V>`
`void BST::_remove(TreeNode *& root, const K & key) {`
 	
`}`

Running time? _____ **Bound by?** _____

CS 225 – Things To Be Doing:
1. Theory Exam 2 Topics List Posted (exam next week) 2. MP3 extra credit on-going; due Monday, Feb. 26 3. Upcoming Lab: lab_trees 4. Daily POTDs