#16: Binary Search Tree (BST)

October 5, 2020 · *G Carl Evans*

Traversal vs. Search:

- **Traversal** visits every node in the tree exactly once.
- **Search** finds one (or more) element(s) in the tree.

Breadth First Traversal + Search:

Depth First Traversal + Search:

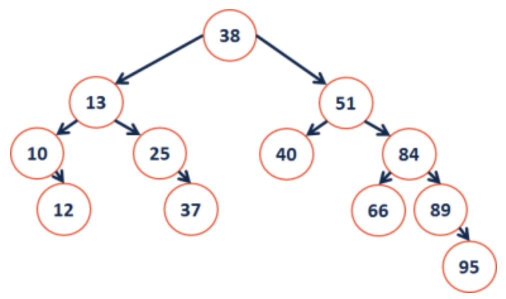
Runtime Analysis on a Binary Tree:

- 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


class Dictionary {
  public:







  private:


};
```

A Searchable Binary Tree?

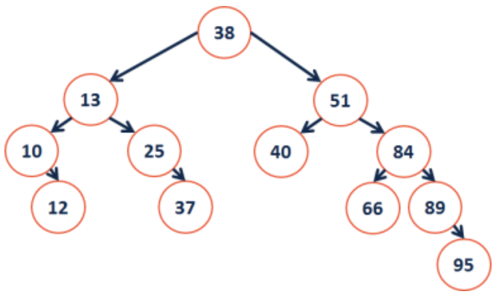


Binary Search Tree Property:

Finding an element in a BST:

```
BST.hpp
template <typename K, typename V>
____________________________________ find(const K & key) {


}

template <typename K, typename V>
___________________________________________________ _find
    (TreeNode *& root, const K & key) {




















}
```

Inserting an element into a BST:

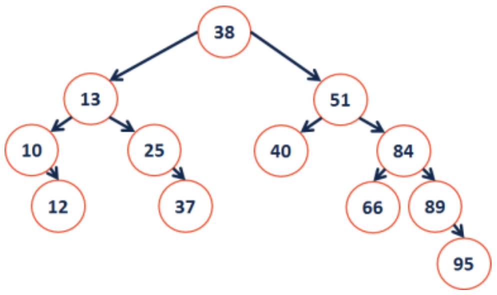


BST.hpp

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









}
```

Running time? ________________ **Bound by?** ________________

What if we did not pass a pointer by reference?



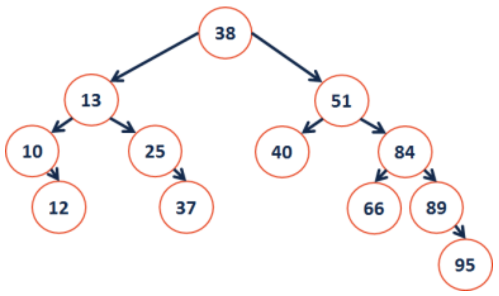
Removing an element from a BST:

`_remove(40)`

`_remove(25)`

`_remove(10)`

`_remove(13)`



One-child Remove	Two-child remove

BinaryTree.hpp

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




}
```

Running time? ________________ **Bound by?** ________________

CS 225 – Things To Be Doing:

1. Mp_list due today.
2. Getting Started/Ask Me Anything session on Tuesday 10/6 from 7-9 PM CDT
3. Daily POTDs