#13: Traversal and BST

February 21, 2022 · *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

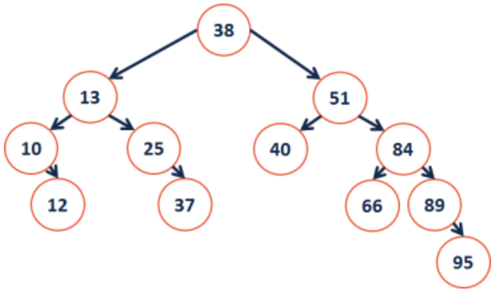
Runtime Analysis on a Binary Tree:

- Find an element: Best case? Worst case?
- Insertion of a sorted list of elements?
  Best case? Worst case?
- Traverse

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) {






















}
```

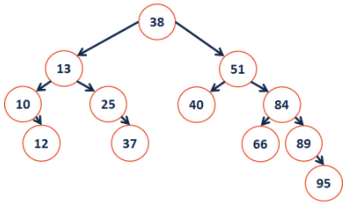
BST.cpp

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

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

What happens when we run the bugged code above?

How do we fix the code?



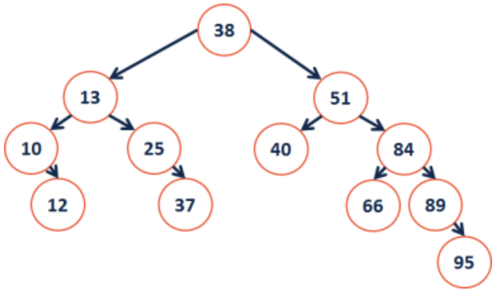
Removing an element from a BST:

`_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?** _______________

BST Analysis:

Every operation we have studied on a BST depends on:

...what is this in terms of the amount of data, **n**?

Final BST Analysis

For every height-based algorithm on a BST:

Lower Bound:

Upper Bound:

Why use a BST over a linked list?

CS 225 – Things To Be Doing:

1. mp_list due Today.
2. exam 1 reschedule window Saturday 2/26 – Monday 2/28.
3. Daily POTDs