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**

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

**Finding an element in a BST:**

```cpp
BST.hpp

template <typename K, typename V>
____________________________ find(const K & key) {

}

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

}
```
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)

Running time? ____________    Bound by? ____________

BinaryTree.hpp

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