Traversals 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**

```
class Dictionary {
public:

private:
}
```
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:

```
(BinaryTree.hpp)

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

Running time? ____________    Bound by? ___________

<table>
<thead>
<tr>
<th>One-child Remove</th>
<th>Two-child remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>_remove(40)</td>
<td></td>
</tr>
<tr>
<td>_remove(25)</td>
<td></td>
</tr>
<tr>
<td>_remove(10)</td>
<td></td>
</tr>
<tr>
<td>_remove(13)</td>
<td></td>
</tr>
</tbody>
</table>

```