

**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:**

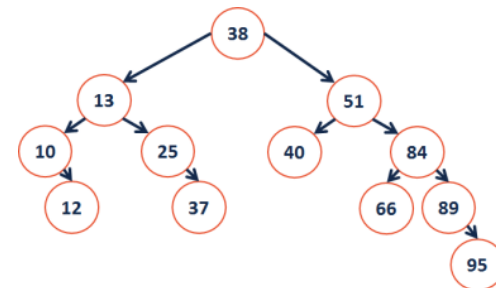
- Find an element:      Best case?              Worst case?
- Insertion of a sorted list of elements?  
                            Best case?              Worst case?
- Running time bound by?

**Dictionary ADT**

```

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

**A Searchable Binary Tree?**



**Binary Search Tree Property:**

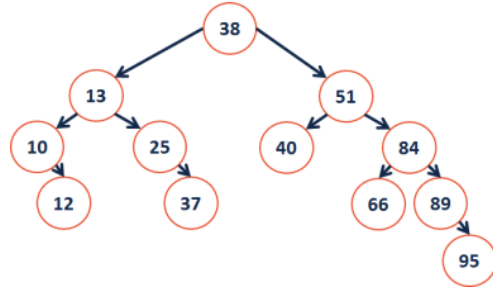
**Finding an element in a BST:**

```

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

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

**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? \_\_\_\_\_

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? \_\_\_\_\_

**Removing an element from a BST:**

- \_\_remove (40)
- \_\_remove (25)
- \_\_remove (10)
- \_\_remove (13)

