A Different Type of Traversal

Strategy:

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

```c++
void BinaryTree<T>::levelOrder(TreeNode * croot) {
}
```

---

**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**

```c++
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?

Removing an element from a BST:

```
_remove(40)
_remove(25)
_remove(10)
_remove(13)
```

How do we fix the code?

CS 225 – Things To Be Doing:

1. Exam 1 on going CBTF/Online
2. lab_trees tomorrow
3. Daily POTDs