One Algorithm, Three Traversals:

```cpp
void BinaryTree<T>::___Order(TreeNode* cur) {
    if (cur != nullptr) {
        // Code for inorder traversal
    }
}

A Different Type of Traversal

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

```cpp
class Dictionary {
    public:
    private:
};
```

A Searchable Binary Tree?

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

Binary Search Tree Property:

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) {
    }
```