EE599 HW5

Runtime analysis

Student Name: Shi Lin Chen

Student ID: 2991911997

1. Find Maximum Depth

```
void BST::NonRecursivePrintInorder(TreeNode *node) {
  level = 1;
  MaxDepth = 1;
  TreeNode* current = node;

  while(current != NULL || path.empty() != true) {
    while (current != NULL) {
      path.push(current);
      current = current->left;
      if(current!=NULL) {
        level++;
        if(MaxDepth < level) {
          MaxDepth = level;
        }
      }
    }
    std::cout << path.top()->val << " " ;
    current = path.top();
    if(current == root_) {
      level = 1;
    }
    DFSpath.push_back(current->val); // push back the path into vector in order for GTEST
    path.pop();
    if(current->right == NULL && current->left == NULL) {
      level--;
    }
    current = current->right;
    if(current!=NULL) {
        level++;
        if(MaxDepth < level) {
          MaxDepth = level;
        }
    }
  }
}
```

Basically, I use Q2 Inorder to find my Height of tree.

Therefore, the time complexity of Inorder is O(n)

Since the time complexity of DFS in a graph G(E,V) is T(n) = O(E+V), which E represents the edges and V represents the vertices. BST is also a graph, therefore T(n) = O(n+(n-1)) = O(n)

2. Inorder, recursive and non-recursive

```cpp
void BST::printInorder(TreeNode *node) { // recursive
  if(node == NULL) {
    return;
  }

  // recursive on left subtree
  printInorder(node->left);

  std::cout << node->val << " ";
   DFSpath.push_back(node->val);

  // recursive on right subtree
  printInorder(node->right);

}
```

```cpp
void BST::NonRecursivePrintInorder(TreeNode *node) {

  TreeNode* current = node;

  while(current != NULL || path.empty() != true) {
    while (current != NULL) {
      path.push(current);
      current = current->left;
      if(current!=NULL) {
        level++;
        if(MaxDepth < level) {
          MaxDepth = level;
        }
      }
    }
    std::cout << path.top()->val << " " ;
    current = path.top();
    DFSpath.push_back(current->val); // push back the path into vector in order for GTEST
    path.pop();
  }
}
```

Recursive and non-recursive will visit every node, until it visit all the nodes.

Therefore, it takes linear time, T(n) = O(n)