Algorithms 4133

HW3

Samuel Lin 010880917

In this Homework Assignment, we were given a mostly-implemented program which tested the functionality of our to-be-implemented binary search tree and AVL tree data structures.

The Binary Search Tree is a non-linear data structure designed to efficiently find a given value. From any node in the tree, the node to the right will contain a key that is greater than the current node's key, and the node to the left will contain a key that is smaller than the current node's key. Any node in the left subtree of the current node will have a key less than the current node's key. Any node in the right subtree of the current node will have a key greater than the current node's key This effectively makes finding a given value on average only take O(log n). We had to implement the basic functions of a binary search tree.

The AVL tree is a self-balancing binary search tree. Because of Binary Search tree's worst case scenario is a time complexity of O(n), the AVL tree was designed to keep the tree balanced (have it self balance itself) upon insertion and removal of nodes to the tree. This essentially means that for any subtree within the tree, there will never be an instance where the difference (balance factor) in height between any left and right subtree is greater than 1. We only had to implement the simple left rotation and right rotations as well as other basic functions for the AVL tree.

These are the results after running the main function through terminal by executing the make file.

This program was implemented and tested on an m1 MacBook Air.

BST

```
(base) smlin@Samuels-Air Homework3 % make
g++ -I./include/ -std=c++14 -DUSE_AVL=1 src/linked_list.cpp src/graph.cpp src/queue.cpp src/stack
.cpp src/bfs.cpp src/bst.cpp src/avl.cpp src/main.cpp  -o bin/main
./bin/main
Perform unit test on your BST implementation
Insert 0 into BST
Find 0. Found 0 in BST
Find 1. Not found 1 in BST

Insert 1 into BST
Find 1. Found 1 in BST
Find 2. Not found 2 in BST

Insert 2 into BST
Find 2. Found 2 in BST
Find 3. Not found 3 in BST

Insert 3 into BST
Find 3. Found 3 in BST
Find 4. Not found 4 in BST

Insert 4 into BST
Find 4. Found 4 in BST
Find 5. Not found 5 in BST

Insert 5 into BST
Find 5. Found 5 in BST
Find 6. Not found 6 in BST

Insert 6 into BST
Find 6. Found 6 in BST
Find 7. Not found 7 in BST

Insert 7 into BST
Find 7. Found 7 in BST
Find 8. Not found 8 in BST

Insert 8 into BST
Find 8. Found 8 in BST
Find 9. Not found 9 in BST

Insert 9 into BST
Find 9. Found 9 in BST
Find 10. Not found 10 in BST

Maximum value in BST: 9

Remove 0 out of BST
Find 0. Not found 0 in BST
Find 1. Found 1 in BST

Remove 1 out of BST
Find 1. Not found 1 in BST
Find 2. Found 2 in BST
```

```
Remove 2 out of BST
Find 2. Not found 2 in BST
Find 3. Found 3 in BST

Remove 3 out of BST
Find 3. Not found 3 in BST
Find 4. Found 4 in BST

Remove 4 out of BST
Find 4. Not found 4 in BST
Find 5. Found 5 in BST

Remove 5 out of BST
Find 5. Not found 5 in BST
Find 6. Found 6 in BST

Remove 6 out of BST
Find 6. Not found 6 in BST
Find 7. Found 7 in BST

Remove 7 out of BST
Find 7. Not found 7 in BST
Find 8. Found 8 in BST

Remove 8 out of BST
Find 8. Not found 8 in BST
Find 9. Found 9 in BST

Your BST implementation is correct
```

AVL

```
Perform unit test on your AVL implementation
Insert 0 into AVL
Find 0. Found 0 in AVL
Find 1. Not found 1 in AVL

Insert 1 into AVL
Find 1. Found 1 in AVL
Find 2. Not found 2 in AVL

Insert 2 into AVL
Find 2. Found 2 in AVL
Find 3. Not found 3 in AVL

Insert 3 into AVL
Find 3. Found 3 in AVL
Find 4. Not found 4 in AVL

Insert 4 into AVL
Find 4. Found 4 in AVL
Find 5. Not found 5 in AVL

Insert 5 into AVL
Find 5. Found 5 in AVL
Find 6. Not found 6 in AVL

Insert 6 into AVL
Find 6. Found 6 in AVL
Find 7. Not found 7 in AVL

Insert 7 into AVL
Find 7. Found 7 in AVL
Find 8. Not found 8 in AVL

Insert 8 into AVL
Find 8. Found 8 in AVL
Find 9. Not found 9 in AVL

Insert 9 into AVL
Find 9. Found 9 in AVL
Find 10. Not found 10 in AVL

Maximum value in AVL: 9

Your AVL implementation is correct

Perform unit test on your implementation with graph
Shortest path from 0 to 5 by : 0 1 3 4 5
Total Distance: 15


Shorest path from JBHT to  detination: JBHT -> HILL -> WJWH -> HAPG
Total Distance: 47354
You have to use OpenCV to visualize your map road
```