**INFO 6205**

**Program Structures & Algorithms**

**Summer Full 2018**

**Assignment 5**

Binary Search Tree is a node-based data structure which the left subtree of a node contains only nodes with keys lesser than the node’s key and the right subtree of a node contains only nodes with keys larger than the node’s key.

In order to make the average depth of the tree start out at lg N and degrade to sqrt(N), I initialized trees as balanced binary search trees which depth is lgN with N nodes.

**1.Conclusion**

In this assignment, I ran several experiments to verify the proposition that the depth(D) of a Binary Search Tree after *N* (Hibbard) deletions and insertions will be proportional to the square root of *N*. The deduced expression is given below:

D=sqrt(N).

In order to deduce this conclusion, I set every integer from 50 to 400 to be the nodes number of the initial trees. Each initial tree is a balanced tree, thus its initial depth is lgN. For every test, I randomly generate either put node or delete node operation, and the node that I insert or remove is also random. After the total 1000 times operations, I can get the current depth of this tree. For every tree I conduct 100 times of duplicate tests, then I get an average depth of a tree.

**2.Graph**

The entire dataset can be seen in the attached files.(bstdata.xsls)

**Example 1**

Initial nodes number=52

Initial depth=5

Theory initial depth=lg52=4.7

Final average tree depth=9.12

Theory final depth=sqrt(52)=7.21

**Example 2**

Initial nodes number=127

Initial depth=6

Theory initial depth=lg127=6

Final average tree depth=12.75

Theory final depth=sqrt(127)=11.27

**Example 3**

Initial nodes number=237

Initial depth=7

Theory initial depth=lg237=7.8

Final average tree depth=16.4

Theory final depth=sqrt(237)=15.39

**Example 4**

Initial nodes number=325

Initial depth=8

Theory initial depth=lg325=8.3

Final average tree depth=16.89

Theory final depth=sqrt(325)=18.02

Even though there’s some variations, but basically match the conclusion.

Hence Proved