Assignment 5. BST

Shuhao Xia || NUID 001899341

Part 1. Conclusion

When initialize a BST of size N (each element of the tree will range from 0-M), and after M successful insertion and deletion is conducted, the average depth (of leaf who doesn't have children) would be proportional to sqrt of N rather Lg N.

However, I cannot see such trend in my experiments. The average depth of the tree stays very stable since insertion and deletion are done randomly.

Part 2. Analysis

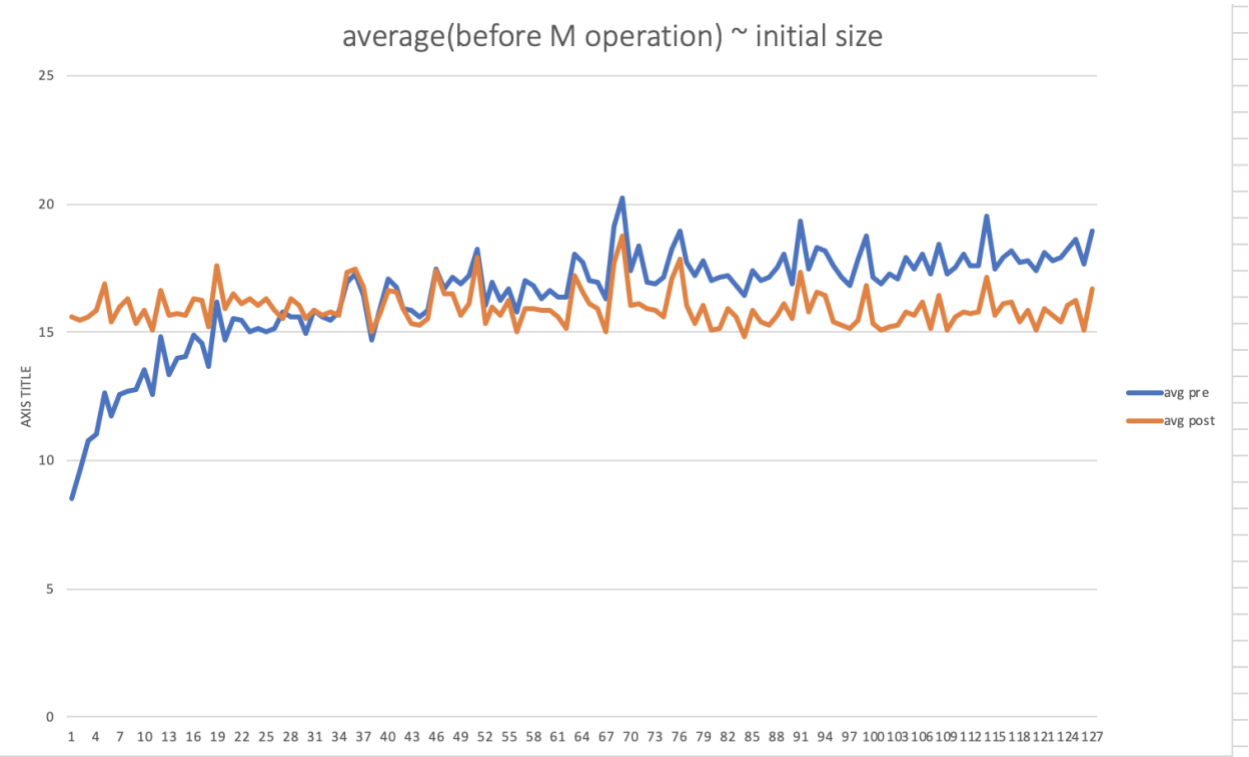
In my approach, I generate the initial BST using a map, and the value of the map is randomly generated according to the range and initial size. The map will be shuffled before construction of BST. Then I calculate the current average depth.

After that, I randomly do insertion and deletion for M times, the key is also randomly generated according to the range. If insertion or deletion failed (i.e. insert existing node or delete non-existing node), nothing will happen and it does not count for a successful operation.

After M operations, I calculate the average depth of the tree again to see if there is a trend. However, there isn't.

I did the experiment on initial size from 100 to 12800, and range from 200 to 12800, for each experiment there is 1 million insertions and 1 million deletions. However, the difference of average depth before and after M operations are slightly the same, both are proportional to lgM but not lgN.

When the initial size is rather small compared to range, the average depth is also small, but when initial size goes bigger than 1/10 of the range, it doesn't change anymore. Below shows the growth of avg depth before and after M operations when initial size increments, the range is set up to 12800.



Part 3. Important code

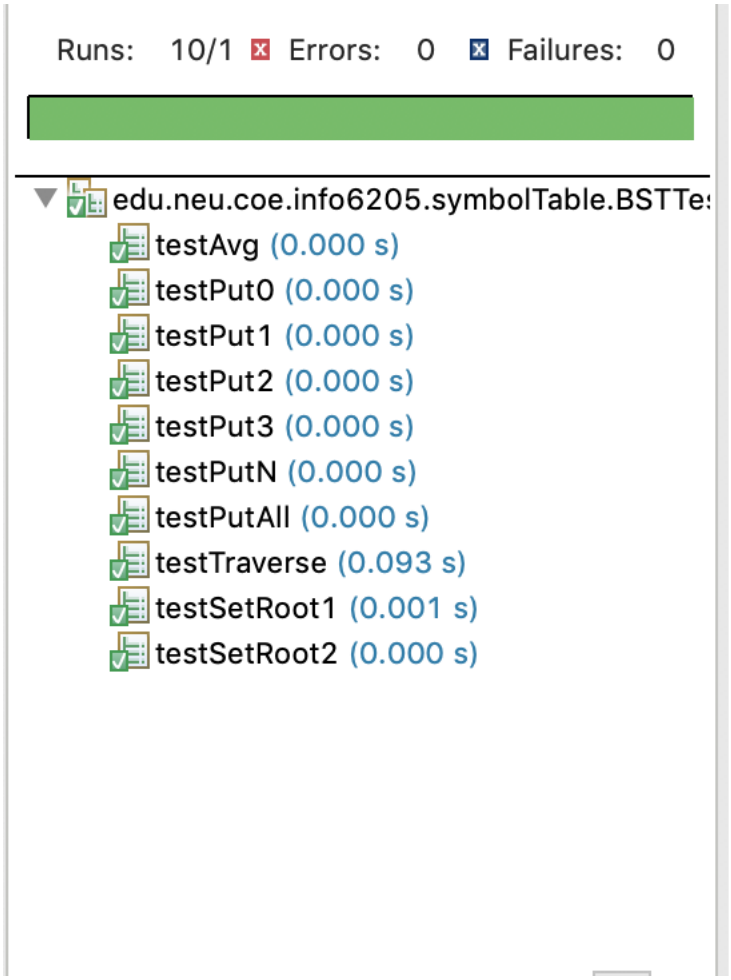
So I thought I might have misunderstood the requirements, I changed the strategy from M successful insertions and deletions to M insertions and deletions regardless of whether it success or not. But unfortunately, the result remains the same, average depth are nearly the same.

There are two csv files in root folder called ‘data.csv’ and ‘data2.csv’. data.csv represents for experiments in which insertions and deletions must be successful. ‘data2.csv’ represents for experiments in which insertions and deletions are counted regardless of whether they are successful.

Other changes are all related with BSTSimple.java. Functions to calculate avg are as below:

```java
public int depth(Node node) {
    return depth(node,root);
}

public int depth(Node node, Node key) {
    if(node == null) return 0;
    int cmp = node.key.compareTo(key.key);
    int dep = 0;
    if(cmp == 0) return 1;
    else if(cmp > 0) return 1 + depth(node,key.larger);
    else return 1 + depth(node,key.smaller);
}

public int leafDepth(Node node) {
    if(node.smaller == null && node.larger == null) return depth(node);
    else return 0;
}

public int sumDepth(Node node) {
    if(node == null) return 0;
    return leafDepth(node) + sumDepth(node.smaller) + sumDepth(node.larger);
}

public int leafCount() {
    return leafCount(root);
}

public int leafCount(Node node) {
    if(node == null) return 0;
    if(node.smaller != null || node.larger != null)
        return leafCount(node.smaller) + leafCount(node.larger);
    else return 1;
}

public double avgDepth() {
    return sumDepth(root)/(leafCount()*1.0);
}
```

Part 4. Screen shot


Runs: 10/1 Errors: 0 Failures: 0
edu.neu.coe.info6205.symbolTable.BSTTe
testAvg (0.000 s)
testPut0 (0.000 s)
testPut1 (0.000 s)
testPut2 (0.000 s)
testPut3 (0.000 s)
testPutN (0.000 s)
testPutAll (0.000 s)
testTraverse (0.093 s)
testSetRoot1 (0.001 s)
testSetRoot2 (0.000 s)