Project #4

Daniel Williamson

11/22/16

Binary Search Tree: Searching O(logN) Inserting O(logN) Traversing O(N)

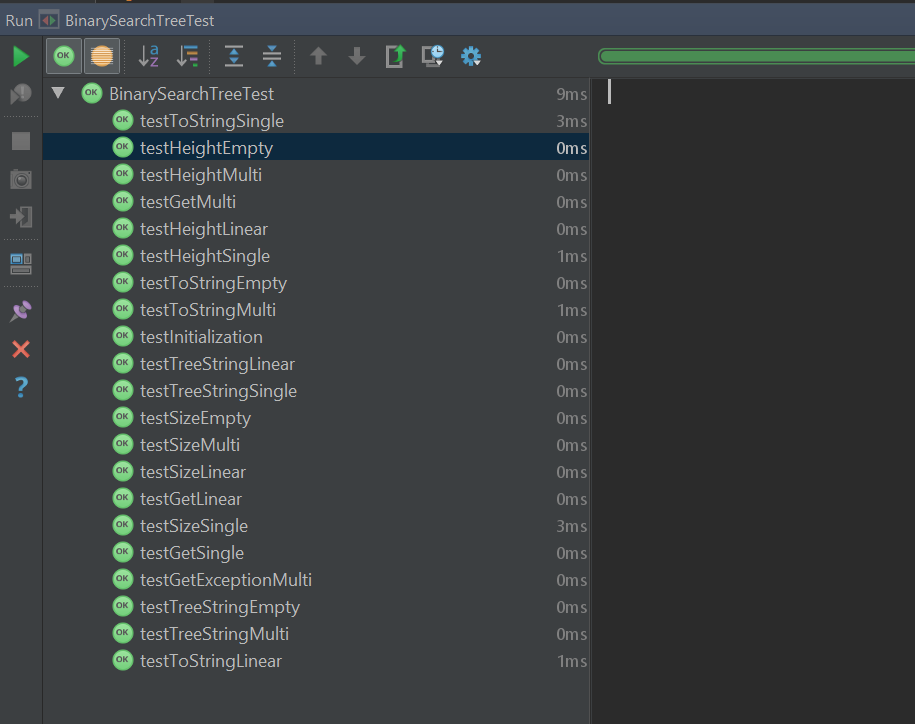
Approach:

My approach for this project was just like many others. I start by doing a lot of research. I reference the book, the internet and peers. Stephen White was one in specific that was a giant help. After figuring out how the Binary Search Tree worked. It was a rather simple concept. Basically, you always have a root. Any number smaller than the root goes to the left and anything larger than the root goes right. Each node after that has two children nodes. These child nodes each have their own two children, and so on and so forth. Each node can only have two child nodes. No more.

To begin, I needed to understand how the skeleton code was working. There were several classes in it, put, get, size, getHeight, getTreeString and a toString. We had to add a few more, along with creating another class, Node. In Node we needed to create what it meant to be a node in our tree. It was rather simple and we created a main method to make sure it was all working properly. Once we knew what a node was, we started with getting the get and put methods to work. Thankfully, the basic code was in the book on page 406. After insuring those worked with another main method. We had to work on the toString and getTreeString. By creating a recursive method for each, to make sure we went through all the nodes in the tree. And making sure when they were printed, they had the same structure as the tests.

After getting those tests passing, we moved onto the height and size. Which was rather simple. Especially the size. By adding a counter and incrementing it every time we adding a node. We would just return that value. For height we find a detailed stack overflow document at: http://stackoverflow.com/questions/2597637/finding-height-in-binary-search-tree. Which basically, we count how many times we go down each left and right side of the root. We then compare those two values for which is greater is the max height of the tree.

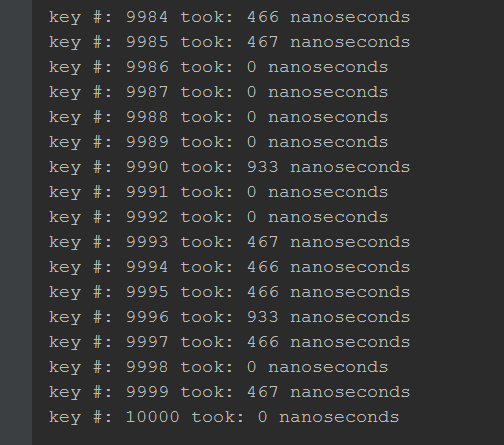
Here is a photo of the code passing all the tests:



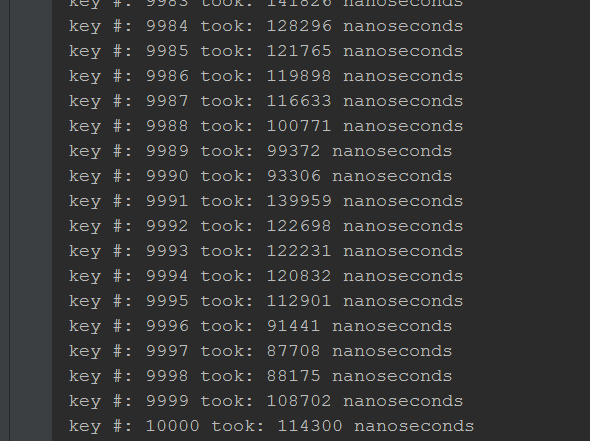
As for the profiler for Binary Search Tree.

The random tree was much quicker:

With some search times take 0 nanoseconds. It was much more efficient. Because they were placed in a more balanced way. As opposed to a linear, unbalanced one.



The linear, ascending order tree was much slower:



The reason this happens is because the random tree is more balanced. Making it much easier to search for a number. And faster since it isn’t as tall.

Red Black Trees: Searching O(logN) Inserting O(logN) Traversing O(N)

Approach:

Just like I said earlier, I started Red Black Trees with doing some research one how it needs to work. I used the internet, the book and peers. Namely Stephen White. The idea of how Red Black Trees works is rather simple. But implementing it was extremely difficult. The rules of Red Black tree are, every node is either red or black, the root is always black, if a node is red its children must be black, every path from root to leaf must have the same black height and you always insert a red node to the tree. With these rules we have to do some reconfiguring, like rotating, to make sure we follow these rules.

Unfortunately, I did not get red black trees to fully function. But, to get all but three tests to pass we created many helper methods. These helper methods do exactly what the sound like they would do. We had reconstruct methods for both a left and right child also if the grandparent was a root node or not. We had recolor methods for both a left and right node if the parent was a root or not. We had rotate methods to rotate for both a left and right child.

Essentially we had helper methods for every case needed to insert a node into a red and black tree. Now, for some reason in my code, it’s not quite keeping track of if a node is black or red, and because of this it’s setting it as null. I’ve tried many other solutions but they just don’t seem to work. Since I did not get all tests to pass, I was unable to right a profiler for this section. I was unable to work as hard as I wanted to since we had the test today as well.

