Colin McAllister

Collin Smith

CPSC 3283

M2 Programming Assignment

The goal for this module was to verify the Theorem in Chapter 12 that states, “The expected height of a randomly built binary search tree on distinct keys is .” To accomplish this, we built a program, using Java, that implemented the *Tree-Insert* method for between 500 and 20,000. At each , we built the tree 10 times and collected the average height if those 10. Then, we compared the average height of a size tree to the growth rate of and the growth rate of . As the program ran, the data was logged into a .csv file for future processing. Once it was finished, while I was analyzing the data, I noticed that the result for produced by the Java program by running Math.log(n) / Math.log(2)returned the incorrect result for each . Because of this, I only exported the average tree height and the size for each and performed the calculations using excel. The Java program can be compiled and run using a Java IDE such as jGrasp or IntelliJ. The program works as intended.

On page 300 of the textbook, right above the Theorem which we are trying to prove, it states that, “…we can show that the behavior of the average case is much closer to the best case than the worst case.” This is regarding a binary search tree built with random distinct values. Since we were using randomly generated non-distinct values, our average heights will be slightly larger, but the idea still applies. As you can see in *Figure 3,* the average height of each size tree is much closer to than it is to , and it grows at the same rate as *Figure 1* shows that the average height at each grows a little slower than , but at the same rate. Since is a constant, we can take for each, to determine if it grows faster or slower than If it grows slower than the trend line will form an incline from left to right. If it grows slower, the trend-line will form a decline from left to right. These principles also apply to the comparison between in *Figure 2* as well.

Since the worst case height for a binary search tree is , it is also important to compare the average height of size tree to the constant This comparison can be seen in *Figure 2* and shows that the average height of an sized binary search tree grows much slower than

As stated above, the theorem we are trying to prove expects the height of a randomly built binary search tree to be if the values are distinct. Since we are not enforcing the distinctness of our values, the height of the binary search trees is larger. This is because nodes with the same value will create a chain effect that will cause an imbalance and increase the height. It is my assumption that by enforcing distinct values, the average height of size binary search trees would decrease to be much close to

In conclusion, it has been shown that the height of a randomly built binary search tree of size is slightly larger than , but grows with the input size as

Figure : Average Height by Log n

Figure : Average Height by n

Figure : Average Height vs. Log n