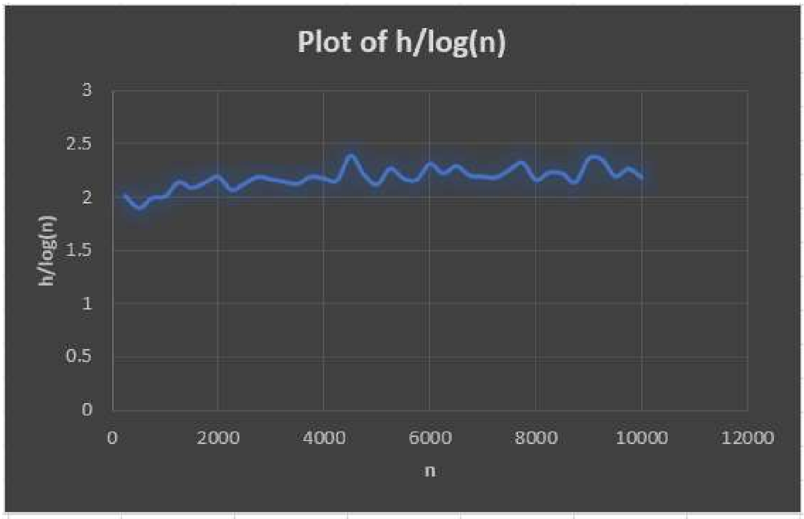
Submitted by: Wilbur Rotoni and Genji Nakano
CPSC-3283-A01
Module 2 Programming Assignment Report

The following is the plot of height(n)/ log(n). Height(n) was obtained empirically following the pseudocode mentioned in Module 2 Programming Assignment instructions:



Interpretation:

The above plot basically represents the degree of skewness of the random binary search trees that our program generated.
As we have learned in Module 2 (slide #14) lecture, there are two extreme cases of binary trees: (1) Complete Binary Tree; and (2) Chain.

Given a binary search tree with "many" nodes, n, we've learned that a complete binary tree has a height of log(n). On the other hand, a chain binary tree has a height of (n – 1).

What this means is: If, by chance, our program 'consistently' builds a complete (i.e. balanced) binary tree over and over again, the average height of the trees would be ≈ log(n), or asymptotically, ≈O(log(n)). This means that the ratio h/log(n), which we are asked to plot would be: h/log(n) ≈ log(n) /log(n) ≈ 1.
On the other hand, if by chance, our program 'consistently' builds a chain tree over and over again, the height would be (n -1), or asymptotically, ≈ O(n). This means that the ratio h/log(n) would be: h/log(n) ≈ n / log(n), which, when plotted, depicts a growing logarithmic function.
In other words, based on these observations, we can say that a randomly generated binary tree can have a degree of skewness from a minimum of 1 (perfectly balanced/complete binary tree) to a maximum of n/log(n) (perfectly skewed/chain binary tree).

Therefore, and in conclusion: As we can see in our plot above, our plot is consistently closer to 1 than it is to n/log(n); Hence, this means that the randomly generated binary trees in our program are "closer" to a complete binary tree than they are to a chain tree.
Thus, the result of our program agrees with Theorem 12. 3 (Chapter 12) because our randomly built binary trees more closely reflect a complete binary tree and as such, their averaged height is ≈ O(log(n)).

In addition:

- Yes, the program works.
- The program is written in Java using Auburn University's jGrasp IDE. Therefore, simply open the ***ProgrammingAssignment1.java*** file in jGrasp -> then Compile -> and then Run.

  The file F is a *.txt file but has been formatted like a *.csv file. This way, it can be conveniently opened (and plotted) in excel.
  Notes:

1. In line # 26 of the code (***ProgrammingAssignment1.java***), the user should change the location where the log file (F) should be saved. Currently, it is set to the programmer's local disk path/folder. **It is very strongly recommended that the user selects the same folder where he/she saved the source code *ProgrammingAssignment1.java* file.**
2. In line # 133, the user should make sure that **the file name in line # 26 and line # 133 should exactly match**.