Mathias Morache CSPC 3283 M2 Programming Report

Status and Instructions: My program collectData.py runs without issue. It is written in Python 3.7.0 and will likely be compatible with older versions of Python 3. If you do not currently have a Python interpreter installed, I recommend downloading IDLE from https://www.python.org/downloads/. No compilation is needed.

Analysis: Since at each interval, the average height taken from a sample of randomly built binary search trees is divided by $\log_2$(number of nodes **n** in the tree), my expectation was that the growth of the plot would be constant. This division should effectively cancel out the growth, since as per Theorem 12.4, the height would also be O($\log_2$(***n***)). I expected this depth ratio to converge somewhere around 2.

However, while the results I collected supported the approximate height given by the theorem, they did not support my expectation that the values would converge. In contrast, the plot (Plot A) appeared to have a gradual but steady increase. To make certain I wasn’t seeing a non-existent pattern, I reran my program several times, tweaking attributes such as the maximum value of ***n***, the sample size of ***j***, and selecting random values from within a fixed range for ***p***. While this slightly tightened the delta between values, the plot (Plot B) still was increasing.

To more efficiently collect data from higher values, I rewrote my program to increase the value of ***n*** on a logarithmic scale from 2^1 up to 2^{26}. The results from these tests showed the plot (Plot C) was increasing while the rate of growth was slowing.

This is where I believe the specification of “***n*** distinct keys” stated in the theorem becomes important. Since the keys in our binary search tree can be repeated, we may be seeing an example of the Birthday Paradox. For ***k*** possible numbers, there exists a high probability of at least one duplicate in sqrt(***k***) random choices of those numbers. This probability increases the higher we go above sqrt(**k**). If these numbers are frequently repeated, then it will lead to deeper trees, as the tree moves farther away from average-case depth and closer to worst-case depth.

In the parameters of the assignment, the value of ***n*** increases with each iteration. In the final iteration of Plot C, ***n*** = 2^{26}. Since sqrt(2^{26})= 2^{13}, once the tree we are building gets above 8,192 nodes, we can expect the number of duplicates to grow. I tested variations with a fixed range for the random number generator, but did not notice an impact on the plot.

The highest values on the plot show a slowing growth with an average depth ratio of around 2.6. My suspicion is that the plot is approaching Euler’s number, but I would need more data to verify this claim. I’m at the limits of what my hardware can efficiently accomplish, so I’d need to rewrite this code in a lower level language and/or make a threaded version. There are more tests I’d like to run, but since this is already beyond the scope of this assignment, I’ll need to do this at a latter time.