Andrew Growney

2749789

Lab 9 Report

During this experimental profiling lab, I got to experience the importance of flow chart planning and reading full descriptions. I ran into major issues with double free and corrupt memory as I attempted to use previous profiling implementation with new data structures. After reading carefully and examining the properties of my data structures, I realized that I would need to bring a new type of format in order to test my data structures in the format described. The most difficult part of organization was understanding that I could not test build and delete in different functions. I would need a continuous flow in order to assure the proper experiment was set up.

Gathering input data through random number generation was rather simple given our previous experience with it. I knew that we had to build with the same set of random numbers of size m which required an array to assure the same numbers being inserted into each one. I also discovered that I would need a heap allocated array in order to account for the size of values I would be building my structures with.

CPU recording time was the most difficult part as I could not determine what was going wrong. I quickly discovered that my machine was using the incorrect formatting for clock, and I needed to be ssh'ed into the cycle servers in order to correctly use the the C++ CPU timing that was described to us in the lab setting. This allowed me to meaningfully interpret my data to show the usefulness of each respective data structure. In order to store my data, I utilized stack allocated arrays to keep track of the trial runs that would use a function that took the averages of the trials in order to report it in the final table.

As you can see through the performance comparison part of my program, the usefulness of deletemin on Min K Heaps and delete max on Max K Heaps make the two data structures very powerful in those respects, but extremely inefficient with the opposite method. The value of having constant access time really is on display through these data structures. As simple as binary search trees are, you get to see how powerful they can be because of how well they perform in this average scenario when inserting truly random numbers. The bad case of binary search tree is not exposed through this experimentation.

To conclude, the power of experimental profiling takes careful detailing before jumping in, I dedicated much more time on this lab than anticipated because of simple misunderstandings and stubbornness to go back and map out a correct flow chart. The data structures all appear to have their strengths and weaknesses in the case of random input data sets. We see the power of Binary Search Trees having great averages for all of them without specific specialization. We also see the ups and downs of K-Heaps through their powerful constant versus linear delete functions.