Michael Linsenbardt
Lab 2 Write up
CSCI 311
10/12/12

Lab 2 Write up

For this lab, my timing information was pretty much what I expected. The quick sort was pretty fast at 0.205063 seconds for sorting by value and 0.117152 seconds for sorting by keys. Merge sort was pretty consistent at 0.299485 seconds for sorting by values and 0.298039 seconds for sorting by keys. Heap sort was the fastest at 0.136185 seconds for sorting by values and 0.141310 seconds for sorting by keys. Insertion sort was, as I suspected, the most consistently horrible at 0.64537 seconds for sorting by values and 0.61907 seconds for sorting by keys.

It makes sense to me that insertion sort would be the slowest algorithm, since it is O(n^2). It follows that the other algorithms would run much quicker, since they are each O(nlogn). It also makes sense that merge sort would be the second slowest, since it is using up memory in making a new vector each time it makes a recursive call. I was a bit surprised that heap sort was so much faster than quick sort though. It is possible that quicksort just happened to encounter a worst case scenario and was therefore a bit slower than heap sort.

For this lab I made quite a few changes to the first lab. The way I wrote my first lab, I was making a new object for every string in the input file. This really slowed down my run-time and was the reason why I was unable to submit my Lab 1 to the turn-it-in system before it shut down on me. I changed this by calling the objects outside of the while loop, which caused my run-time memory to be used up, and instead used functions which took in strings to do all of the conversions. This allowed me to use one FileReader instead of creating n of them. I also created a new class WordSort, which was responsible for all of the sorting algorithms and helper functions and I got rid of my WordCount class. I think that this class was definitely the best written class in this lab and it handled very nicely.

I think that this lab actually wasn't all too difficult. I had to think about how I was going to change my OO design, since it was horribly utilized in the last lab, and I also needed to make sure that my sorting algorithms worked. I made sure that my sorting algorithms were working first, since I figured it didn't make sense to optimize my code until I knew it worked properly. I, surprisingly, didn't have to much difficulty in getting these algorithms to work. The pseudocode in the book was very helpful in writing these algorithms, and I was able to just translate this into C++ for most of my algorithms. I did have problems getting heapsort to work, however, which forced me to work through it and see how it was working. I found this exercise very beneficial and I think it helped me to better understand how that particular algorithm worked. I also had difficulty with the timer functions. I finally realized that the reason the timer functions weren't working for me was because my librt file didn't include the gettime function. I was then forced to come to the computer lab to make sure that I could get it to work. Thankfully, it did. All in all I am glad that I did this lab and I am much more pleased with this one than I was with the last one.