PROJECT 6 ANALYSIS

CHARLES RAYMOND

1. Intro

1.1. For project 6 we had to create a program with 4 different sorting algorithms. The algorithms that I chose were bubble sort, insertion sort, selection sort, and quick sort. Each algorithm was tested on a data set of 88 doubles, and was timed based on the amount of time it took to complete the sorting.

2. Analysis

2.1. The overall times for each algorithm were the following; bubble sort 22-28 milliseconds, selection sort 14-20 milliseconds, insertion sort 8-14 milliseconds, and quick sort 1-2 milliseconds. The most drastic time difference that I encountered was the difference in quick sort compared to all of the others. Theoretically it is 0(nlogn), but I've never seen the difference empirically like in this project. We are always told that 0(n2) is terrible, and it really shows here. The only tradeoff with quick sort compared to the others is it does inefficiently use memory, and is not optimal for embedded systems. You can see that bubble sort is by far the worst algorithm, having the most swaps and taking the longest. selection sort uses less swaps and takes shorter then bubble sort. Insertion sort is twice as fast bubble sort, and uses copies instead of swaps which are less costly.

3. Conclusion

3.1. Using c++ as the choice of programming language gives the user a chance to make use of pointers. For example, in quick sort I don't pass an array for the recursive call, instead I pass a pointer to the array, which uses much less memory. I can do this because I know that an array is continuous in memory, so as long as I have a pointer to the first index, then I know where the whole array is. Other programming languages like java and python to not have this ability. The short comings of the empirical analysis was that each test was not conducted at the same exact time, but rather in sequential order. This could have a chance to effect the run time.