Eric Gustin

CPSC 223-01HW6

October 21, 2019

**Testing Strategy & Possible Improvements**

My testing strategy was slightly different from that of previous weeks. Since the algorithms that I implemented had to be able to pass the rand files and the desc files, I split my testing into three parts.

1. Write algorithm on paper / whiteboard, then type it on computer and continue to test until it passes all of my personal tests in my hw6\_tests.cpp file
2. Test the algorithm with the rand and desc files, and if they passed, then move onto step three.
3. Repeat with next algorithm.

**Implementation Issues / Challenges & How I Addressed Them**

I implemented insertion sort first. I was able to easily pass all of my personal tests, but once I tested it on my rand50k file, I was getting a memory error. This was odd and thought that it could have possibly been due to my destructor, so I moved on to the next algorithm thinking that it would be an easy fix later down the road. After I had implemented the other two algorithms, my uncommented code looked daunting, so I decided to just scrap everything that I had previously done and start from scratch. Lesson learned; I need to start commenting my code while I write it instead of waiting until I’m finished with the homework assignment.

Merge sort did not take that long to implement for me. In fact, most of my time with this algorithm was spent trying to track down a dumb bug that I introduced. I eventually found it – when calling my helper function, I did not set the function call equal to head….. I eventually noticed this, and the algorithm finally worked perfectly.

Quick sort was difficult because I couldn’t find the pseudocode in the lecture notes. So, a lot of time was spent just trying to figure out how the algorithm itself works. I would say that this algorithm took me the longest to implement and probably took 3 attempts before getting it right.