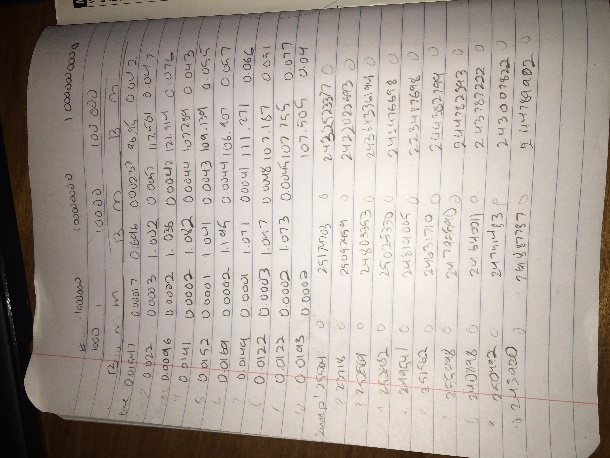
**Homework 3 Documentation**



The picture is the data from my slide, as of right now, I am not sure why I didn’t use an excel sheet to calculate my averages or write my data, but that is my table.

I was not sure if you wanted my textfiles of the generated and sorted arrays, but I excluded them for your testing to generate them myself. I did not generate them to its own file, so I apologize for the organization of the generation of files within the testing.

1000 Array:

Average Time:

|  |  |  |  |
| --- | --- | --- | --- |
| Bubble Sort | 1000 | 10000 | 100000 |
| Average Time | .01587 Seconds | 1.1412 Seconds | 109.308 Seconds |
| Average Comparisons | 1000000 | 100000000 | 10000000000 |
| Average Swaps | 249028.4 | 24703832.4 | 24482672.17 |

|  |  |  |  |
| --- | --- | --- | --- |
| Merge Sort | 1000 | 10000 | 100000 |
| Average Time | .000197 Seconds | .004137 Seconds | .0554 Seconds |
| Average Comparisons | 1000 | 10000 | 100000 |
| Average Swaps | 0 | 0 | 0 |

My random generator should be encased within the main, so running the program with make and the ./main should give you the option to test any size of arrays(ranging from 1 to 10e6).

**Case 1:**

I chose Bubble Sort because I wanted to test the inefficiency of bubble sort, it was always the go to sort in cs135 because of how easy it was, and I wanted to see if it really was all that, and my results were that it was definitely not.

**Case 2:**

I had a large amount of problems doing merge sort, and the differences were actually incredible, and quite eye opening to how some sorting methods can be super inefficient with large amounts of data structures, I could imagine a game server filled with millions of users and if some coder decided to use bubble sort how slow the servers could be.