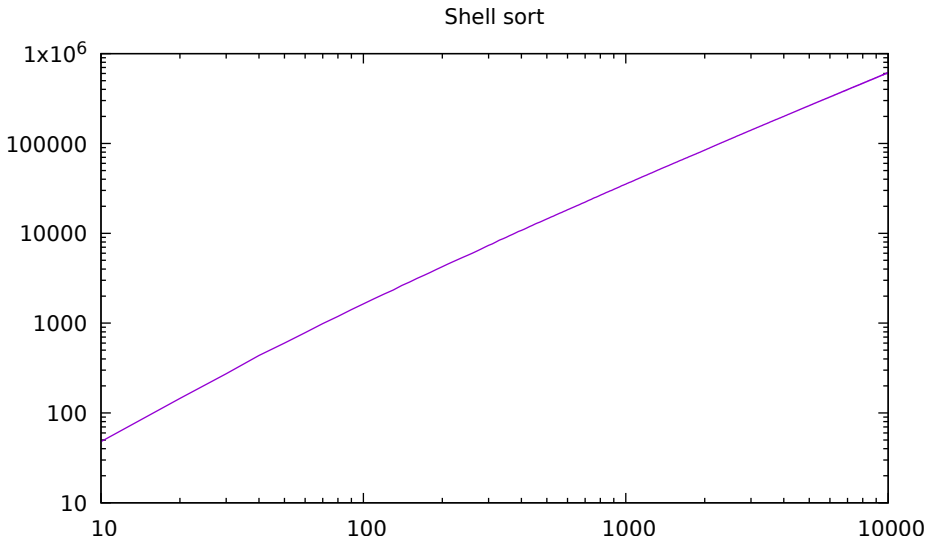
Assignment 3
Writeup
Prof Veenstra
Nguyen Vu

1. Shell Sort

I created a plot for the Shell sort with arrays with different sizes varies from 10 to 10000. It is good with small/medium size array but not as efficient as other sorts. The time complexity or performance of the sort depends on the gaps we used in the code with the best is

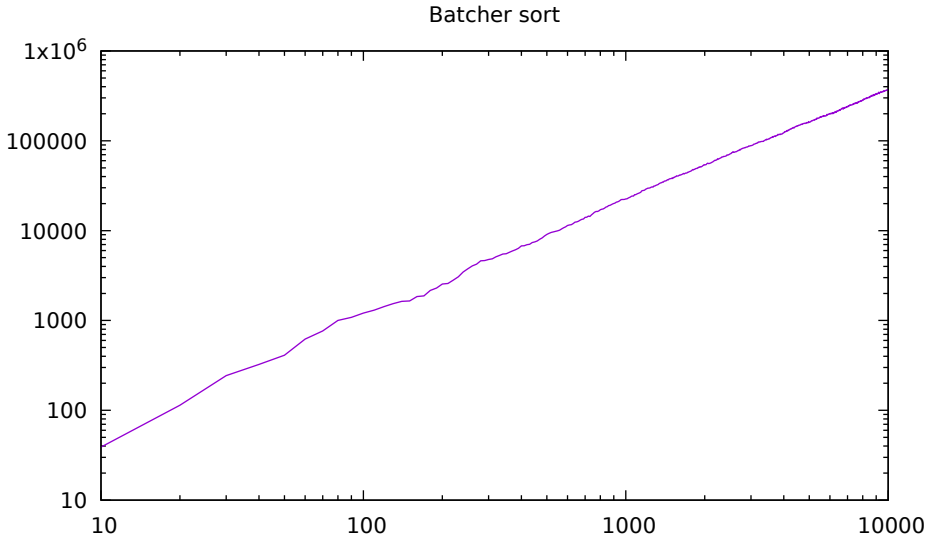
$$O(n^2)$$



2. Batcher Sort

I created a plot for the Batcher sort with arrays with different sizes varies from 10 to 10000. It is good with large size array with its fast speed but limited to inputs that are powers of 2. The time complexity or performance of the sort is

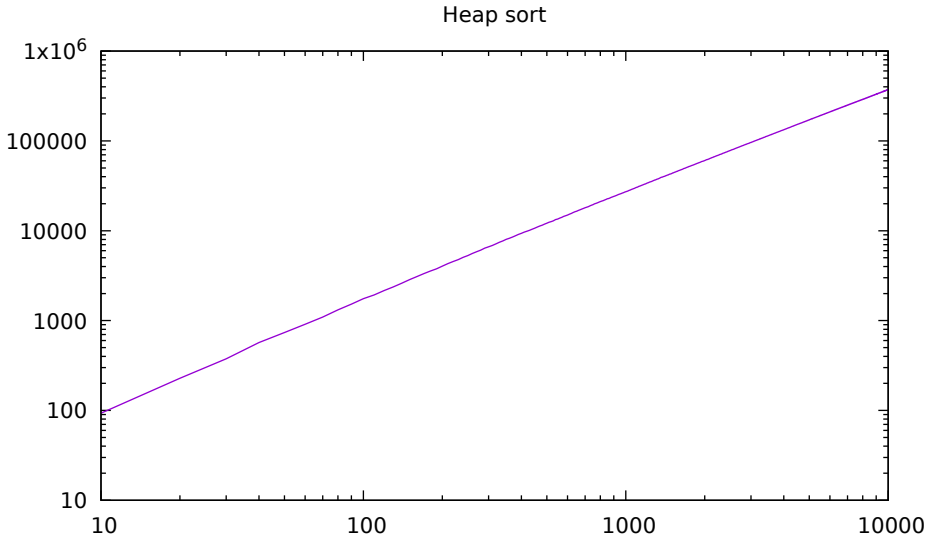
$$O(log^2(n))$$



3. Heap Sort

I created a plot for the Heap sort with arrays with different sizes varies from 10 to 10000. It is good with large size array because it it non-recursive but it is slower than Quick Sort. The time complexity or performance of the sort is

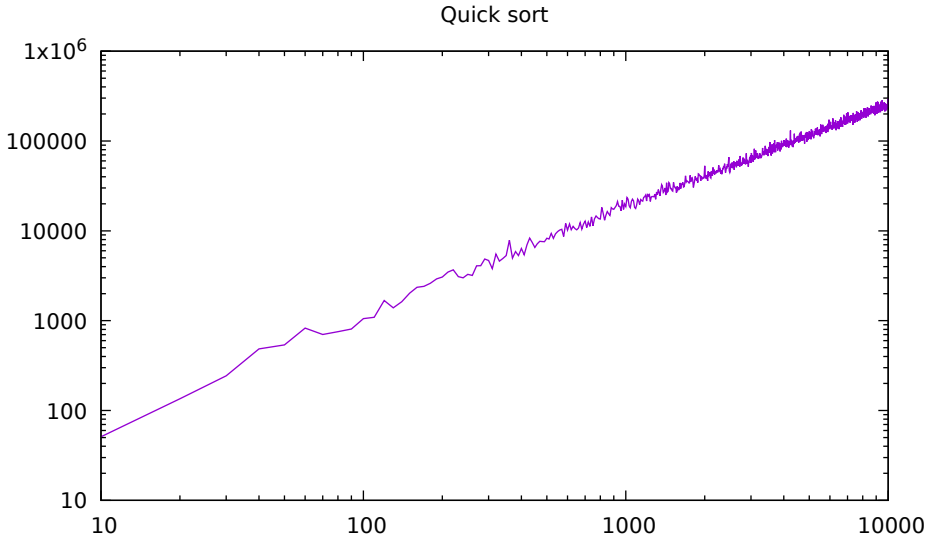
$$O(nlogn)$$



4. Quick Sort

I created a plot for the Quick sort with arrays with different sizes varies from 10 to 10000. It is extremely fast just like its name but it has very complex algorithm and rely massively on recursive. The time complexity or performance of the sort is

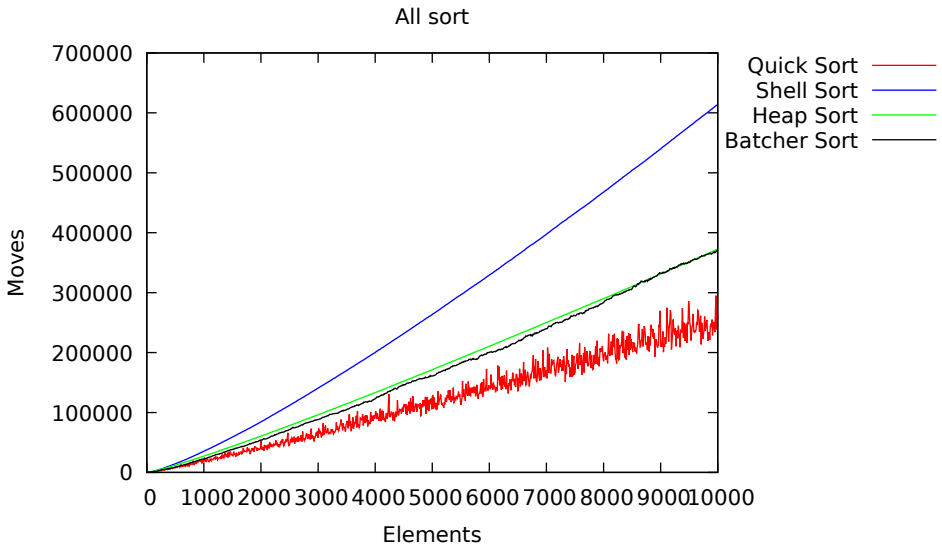
$$O(nlogn)$$



5. Conclusion

After implementing all these sorts in C, I understand about the algorithms much better. One important thing I learned is that time complexity makes the difference a lot clearer with larger data set, as the Shell sort took some time to get the data output but other sorts did it a lot faster.

One thing I learned other than time complexity is the gap sequence in Shell sort as changing it will affect the time complexity of the sort. The plot below is comparing all algorithm without axis scaling so it's easier to see the performance difference at larger data sets.


All sort
Moves
Elements
Quick Sort
Shell Sort
Heap Sort
Batcher Sort
0
100000
200000
300000
400000
500000
600000
700000
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000