Project 1 Report

Xiaowen Li

October 29, 2021

Sorting Time

Number of elements	SS	SC	SR	IS	IC	IR	QS	QC	QR
10000	0:00:03.452429	0:00:02.361607	0:00:05.111574	0:00:00.001995	00:00:00	0:00:04.796828	0:00:00.018955	N/A	0:00:00.046865
20000	0:00:14.075634	0:00:13.578764	0:00:17.949581	0:00:00.002991	0:00:00.015621	0:00:19.344745	0:00:00.036911	N/A	0:00:00.062484
30000	0:00:29.492556	0:00:27.926033	0:00:52.872940	0:00:00.003957	0:00:00.015623	0:00:49.422836	0:00:00.140592	N/A	0:00:00.062487
40000	0:00:51.534278	0:00:50.166361	0:01:32.031272	0:00:00.015621	0:00:00.015621	0:01:19.754126	0:00:00.203076	N/A	0:00:00.156213
50000	0:01:19.303204	0:01:17.576482	0:02:26.171309	00:00:00	00:00:00	0:02:18.208202	0:00:00.172106	N/A	0:00:00.187455
60000	0:01:54.989561	0:01:54.025863	0:03:32.308107	0:00:00.015621	0:00:00.031241	0:03:12.975336	0:00:00.171834	N/A	0:00:00.218368
70000	0:02:37.272483	0:02:31.242580	0:04:42.910017	00:00:00	0:00:00.015622	0:04:45.427655	0:00:00.187455	N/A	0:00:00.249941
80000	0:03:33.142269	0:03:22.891619	0:06:50.282231	0:00:00.015622	0:00:00.031243	0:06:14.025323	0:00:00.249942	N/A	0:00:00.281184
90000	0:04:33.403287	0:04:55.748776	0:10:26.596889	0:00:00.015622	0:00:00.031245	0:07:21.483132	0:00:00.297074	N/A	0:00:00.297072
100000	0:05:49.278302	0:05:19.586964	0:13:02.230891	0:00:00.016676	0:00:00.031242	0:09:59.855181	0:00:00.468640	N/A	0:00:00.296804
500000	2:31:20.592344	2:29:49.414560	5:40:13.200979	0:00:00.109349	0:00:00.109349	4:28:24.066135	0:00:01.797098	N/A	0:00:02.015700
1000000	9:35:54.601319	9:34:27.652637	15:32:06.858638	0:00:00.187456	0:00:00.281183	17:47:26.593004	0:00:03.510723	N/A	0:00:04.671380
10000000	N/A	N/A	N/A	0:00:01.897534	0:00:02.391379	N/A	0:00:49.018519	N/A	0:01:03.037933
100000000	N/A	N/A	N/A	0:00:18.575299	0:00:23.815414	N/A	0:08:39.806680	N/A	0:17:42.562546
1000000000	N/A	N/A	N/A	0:04:59.586498	0:03:35.930854	N/A	1:08:20.997106	N/A	N/A

Note:

- All run-time is obtained by running algorithms in parallel.
- Time is given in the format: hh:mm:ss.000000

Q&A

1. Which algorithm(s) performed the best on data $\leq$ 1,000,000?
For already sorted data and constant data: Insertion Sort
For random data: Quick Sort

2. Which algorithm(s) performed the best on data $<$ 1,000,000 AND $<$ 1,000,000,000?
For already sorted data and constant data: Insertion Sort
For random data: Quick Sort

3. Were you able to run anything that finished on 1,000,000,000? If you were not able to run it, why not?

I am able to run IS, IC, QS with 1,000,000,000 data set. I couldn't finish this for selection sort because the $\mathcal{O}(n^2)$ complexity tells this algorithm takes extremely long time to run. Same reason for IR. IS and IC run pretty fast due to at such situations insertion sort's complexity is $\Theta(n)$. Surprisingly, QR also couldn't finish in 8 hours. Then for QC, this does not work even the data size is small like 10000, because it reaches the worst case for quick sort – Even we choose the median of three, since all values are the same, we keeps choosing the 'worst pivot', so that one of the partitions has only a single element every time. So the complexity is $\mathcal{O}(n)$.

Even worse, for the actually implementation, the recursion depth could be n, which may trigger stack overflow.

4. Were the results what you expected? Did any of the algorithms perform significantly worse than you expected? If so, which one(s) and how/why? (Compare your results to the Justification of Big-Oh slide).

Since selection sort is always $\mathcal{O}(n^2)$ so I do expect it to be the slowest one for all cases out of three algorithms. Insertion Sort has a best case $\Theta(n)$ when the data was already sorted (or constant), so it will perform very fast in those cases. And with a worst case of $\mathcal{O}(n^2)$, insertion sort should have

a similar performance with random data just as selection sort. Then since quick sort is generally $\Theta(n\log(n))$, and we know $\log(n)$ grows slower than any polynomials so it's supposed to be the fastest one but not when it reaches the worst case - which is when all data are the same and gives $\mathcal{O}(n^2)$.

But as mentioned in Q3, QR takes surprisingly long time to finish sorting data with size 1E9. And I think the reason is kind of similar as why QC takes so long to finish - we are having too many duplicated values in the data set and which causes bad pivots. This happens because to prevent memory error(not enough mem to run the program), I set the maximum range for taking randomization to be 1000000.

5. Along with the filled in CSV file, you need to create a graph for each type of data (1 graph for random, 1 for sorted, and 1 for constant). Each graph will contain three-line graphs – 1 for quicksort, 1 for insertion sort, and 1 for selection sort. The x axis is the number of elements, and the y axis is the time taken. In total, you will have 3 graphs. Each of these need to go into your project report. The purpose of these graphs is to easily compare the running times of each algorithm for the data types.