COMP2521 Sort Detective Lab Report

by Jinghan Wang (z5286124)

In this lab, the aim is to measure the performance of two sorting programs, without access to the code, and determine which sorting algorithm each program uses.

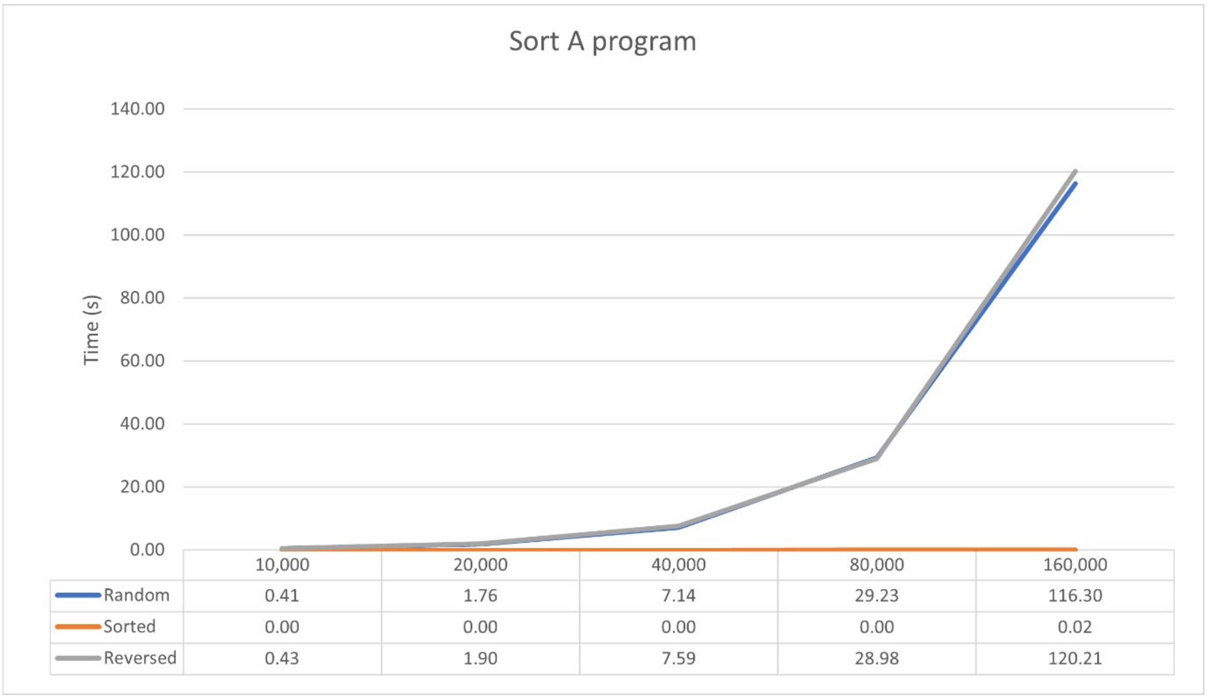
Experimental Design

To find out what sort the two programs are, I use the control variable method to solve. There are three programs in this experiment. First, Since the three orders of input data generated by the gen program are the variables of the project, I checked whether the data generated by the gen program matched the conditions I need. In addition, I check the output of program A and program B under the same input to ensure the validity of the program. These checks are used to ensure that everything except variables is working properly.

Due to less data, it is impossible to compare the time difference between each program in different sorts, I measured how each program's (Program A and Program B) execution time required by providing a large quality of data in ascending, descending and random conditions.

I provide each program 10,000, 20,000, 40,000, 80,000, 160,000 numbers of data as input in sorted, reversed and random conditions, and because of the way timing works on Unix/Linux, I repeat the same test five times to reduce the misjudgment of the results caused by error.

Experimental Results (Program A)



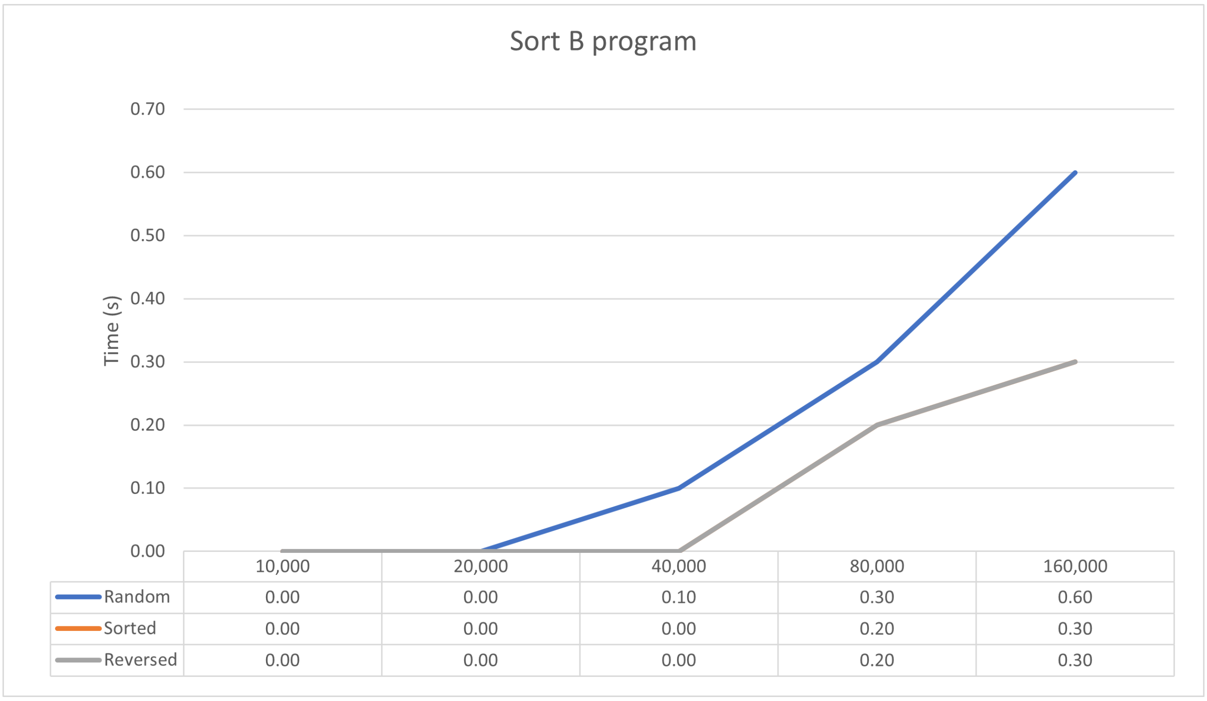
	10,000	20,000	40,000	80,000	160,000
Random	0.41	1.76	7.14	29.23	116.30
Sorted	0.00	0.00	0.00	0.00	0.02
Reversed	0.43	1.90	7.59	28.98	120.21

For the sorting A program, first of all, I found that when the input data is in order, it takes very little time to complete. When the input is random or reverse order, it takes more time. At the same time, the time required for reverse order is slightly higher than that for random order. Then, from the change of input quantity, when the quantity becomes twice the original, the time required is four times the original.

In my opinion, this is Selection sort. First, the average time complexity of sorting is O(n^2), which is consistent with my test results. When the number of inputs increases by n times, the time will also increase by the quadratic power of n. In addition, when the order is selected, the selection sorting only needs to traverse the whole data without changing the position. It is the best case for the selection sorting, and the time complexity is O(n). When sorting in reverse order, it takes a long time to move all numbers except the first one to the left of the first number and find the corresponding position. Compared with receiving, when it is random, the moved data does not need to move much when it is in reverse order, so the time required will be reduced accordingly. In reverse order, it is the worst case, and the time complexity is O(n^2). In random order, the time complexity is O(n^2) to meet the test, and the time required for reverse order is slightly higher than the random time. When $n = 2n$, the time will become four times.

Therefore, sortA is Selection sort.

Experimental Results (Program B)



	10,000	20,000	40,000	80,000	160,000
Random	0.00	0.00	0.10	0.30	0.60
Sorted	0.00	0.00	0.00	0.20	0.30
Reversed	0.00	0.00	0.00	0.20	0.30

For the sorting B program, first of all, I found that compared with the sorting a, the time is much shorter under the condition of inputting the data of the same three orders. Even in the 160000-size sample, the operation is less than 1s. At the same time, I found that when the input data are arranged in order or reverse order, the time required is less than that of random data.

In my opinion, this is merge sort. First, compared with other sorts, the advantage of merge sort is that the same steps are required for any kind of data, which consumes less time and requires higher memory. When the input data is sorted in order or reverse order, because it is sequential, when the list is completely split and then going to combine, all the numbers on one side of the combination must be smaller than the other side, and the merging is faster. For random data, it is necessary to keep comparing to find the most appropriate position during merging, the time will be higher than in order.

The time complexity of merge sort is O($nlogn$), different input data will produce time differences, but the differences will not be exponential.

Therefore, sortB is Merge sort.

Conclusions

On the basis of our experiments and our analysis above, I believe that

- sortA implements the Selection sort algorithm.
- sortB implements the Merge sort algorithm.

Appendix

Testing for input, size 10,000

Time	random	sorted	reversed
1	0.40	0.00	0.44
2	0.40	0.00	0.42
3	0.40	0.00	0.42
4	0.41	0.00	0.42
5	0.42	0.00	0.47
Average	0.41	0.00	0.43

Testing for input, size 10,000

Time	random	sorted	reversed
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.00	0.00	0.00
Average	0.00	0.00	0.00

Testing for input, size 20,000

Time	random	sorted	reversed
1	1.92	0.00	1.77
2	1.76	0.00	1.78
3	1.77	0.00	1.88
4	1.72	0.00	2.05
5	1.65	0.00	2.04
Average	1.76	0.00	1.90

Testing for input, size 20,000

Time	random	sorted	reversed
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.00	0.00	0.00
Average	0.00	0.00	0.00

Testing for input, size 40,000

Time	random	sorted	reversed
1	7.16	0.00	7.21
2	6.97	0.00	7.79
3	7.30	0.00	7.28
4	7.04	0.00	6.88
5	7.22	0.00	8.77
Average	7.14	0.00	7.59

Testing for input, size 40,000

Time	random	sorted	reversed
1	0.01	0.00	0.00
2	0.01	0.00	0.00
3	0.01	0.00	0.00
4	0.01	0.00	0.00
5	0.01	0.00	0.00
Average	0.01	0.00	0.00

Testing for input, size 80,000

Time	random	sorted	reversed
1	29.84	0.00	28.92
2	29.26	0.01	28.80
3	28.80	0.00	28.82
4	28.47	0.00	29.40
5	29.76	0.01	28.94
Average	29.23	0.00	28.98

Testing for input, size 80,000

Time	random	sorted	reversed
1	0.02	0.02	0.02
2	0.02	0.03	0.01
3	0.03	0.02	0.01
4	0.03	0.02	0.02
5	0.04	0.01	0.02
Average	0.03	0.02	0.02

Testing for input, size 160,000

Time	random	sorted	reversed
1	115.50	0.02	115.60
2	115.94	0.02	115.95
3	118.56	0.02	126.16
4	116.80	0.01	124.62
5	114.70	0.01	118.72
Average	116.30	0.02	120.21

Testing for input, size 160,000

Time	random	sorted	reversed
1	0.06	0.03	0.03
2	0.07	0.02	0.03
3	0.05	0.03	0.03
4	0.08	0.05	0.03
5	0.04	0.03	0.03
Average	0.06	0.03	0.03