COMP2521 Sort Detective Lab Report

by Tianwei Mo (z5305298), Peter Seo (z5267344)

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

We measured how each program's execution time varied as the size and initial sortedness of the input varied. We used the following kinds of input :

1. `./gen 50000 R > mydata`
2. `./gen 5000000 R > mydata`
3. `./gen 50000 A > mydata`
4. `./gen 5000000 A > mydata`
5. `./gen 50000 D > mydata`
6. `./gen 5000000 D > mydata`

We used these test cases because each sorting algorithm has a different time complexity. For example, merge sort has a time complexity of O(N * log(N)) in both best and worst cases. On the other hand, bubble sort has a time complexity of O(N^2) for worst cases and a time complexity of O(n) for best case (when the data in the array is already sorted). By generating datasets with high volume, we would be able to conduct a more accurate experiment.

The first two datasets are in random order. We have come up with two datasets each with size `50000` and `500000` so that it would be easier to observe time complexity. The other datasets are used to test for best cases and worst cases as the time complexity of most sorting algorithms depends if the dataset is already sorted (whether in ascending or descending).

Because of the way timing works on Linux, it was necessary to repeat the same test 10 times.

We also investigated the stability of the sorting programs by running particular datasets. For example:

1 asf

1 osn

1 qfs

1 rvx

etc.

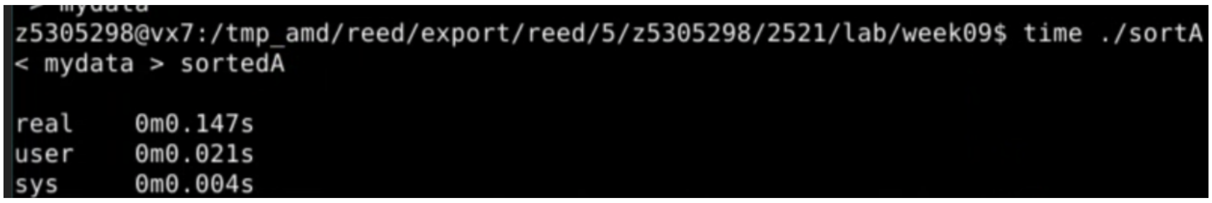
We also tried investigating how many memories are being used. Since sorting algorithms such as merge sort needs an extra memory storage, checking how memory usage could potentially help determine the type of the sorting. However, permission on using valgrind was denied.

Experimental Results

For Program A, we observed that …

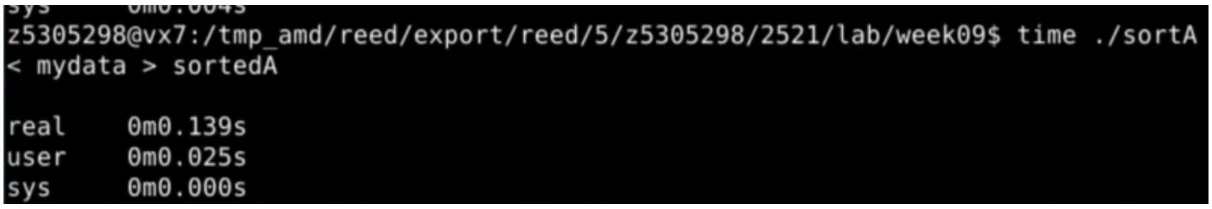
50000 R

1st time



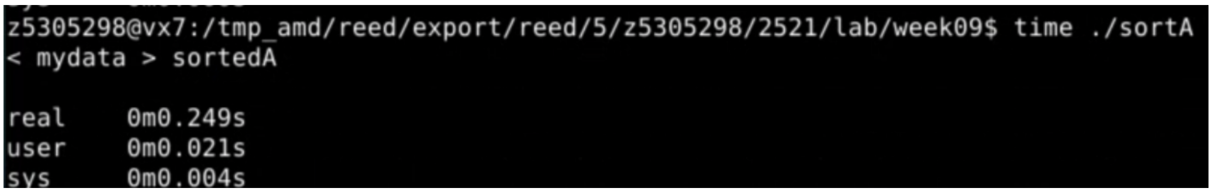
```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m0.147s
user    0m0.021s
sys     0m0.004s
```

2nd time



```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m0.139s
user    0m0.025s
sys     0m0.000s
```

3rd time

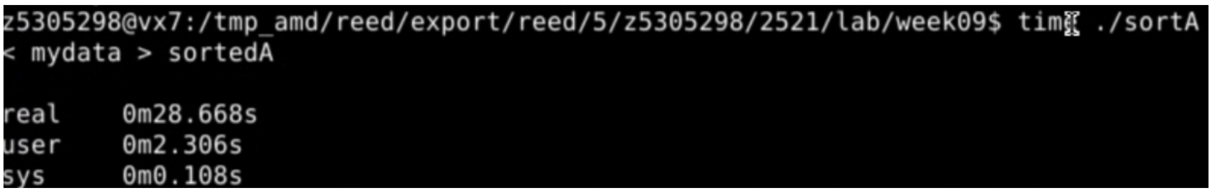


```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m0.249s
user    0m0.021s
sys     0m0.004s
```

These observations indicate that the algorithm underlying the program A consumed about 0.08s to run the dataset 50000 R.

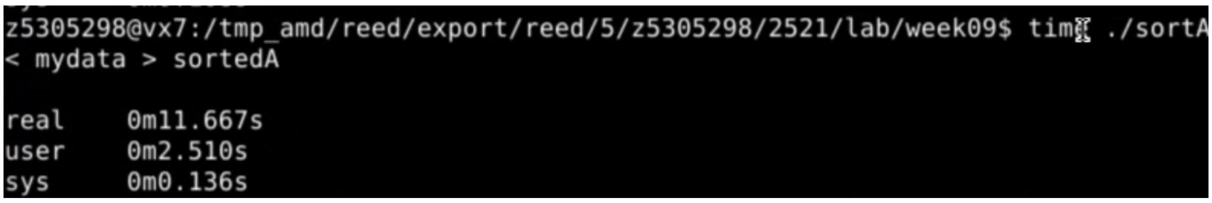
5000000 R

1st time



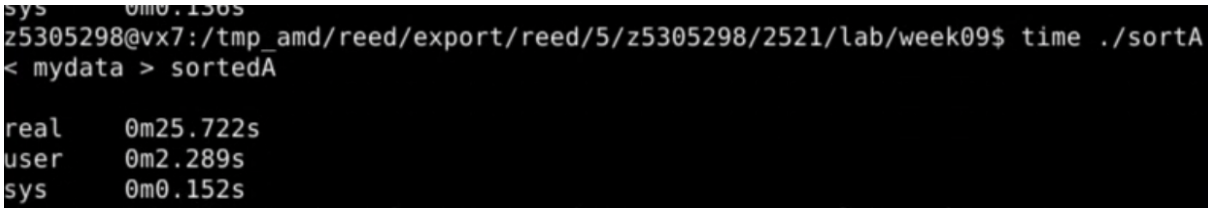
```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m28.668s
user    0m2.306s
sys     0m0.108s
```

2nd time



```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m11.667s
user    0m2.510s
sys     0m0.136s
```

3rd time

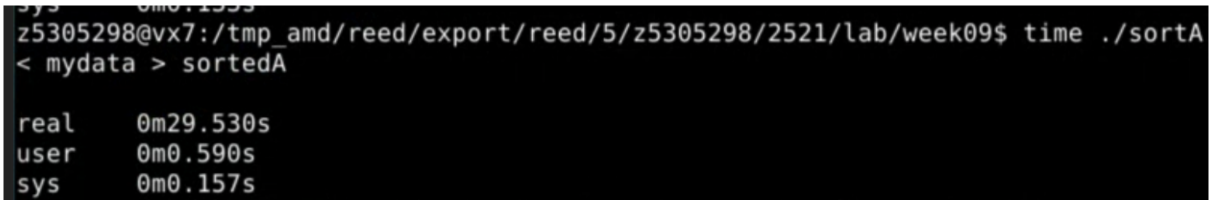


```
sys     0m0.136s
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m25.722s
user    0m2.289s
sys     0m0.152s
```

These observations indicate that the algorithm underlying the program A consumed about 25s to run the dataset 5000000 R.

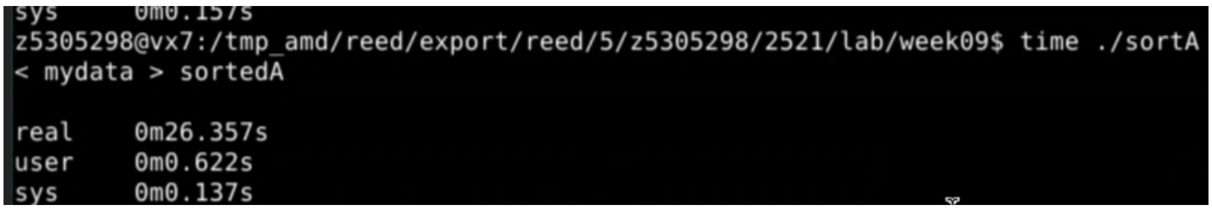
5000000 A

1st time



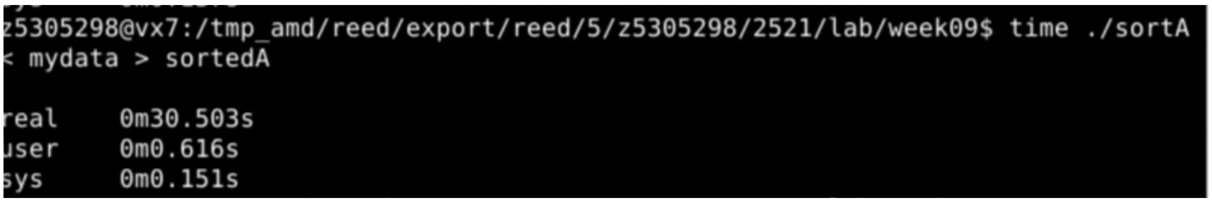
```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m29.530s
user    0m0.590s
sys     0m0.157s
```

2nd time



```
sys     0m0.157s
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m26.357s
user    0m0.622s
sys     0m0.137s
```

3rd time

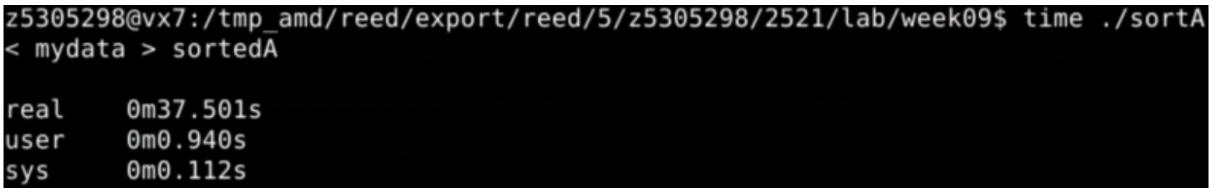


```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m30.503s
user    0m0.616s
sys     0m0.151s
```

These observations indicate that the algorithm underlying the program A consumed about 28s to run the dataset 5000000 A.

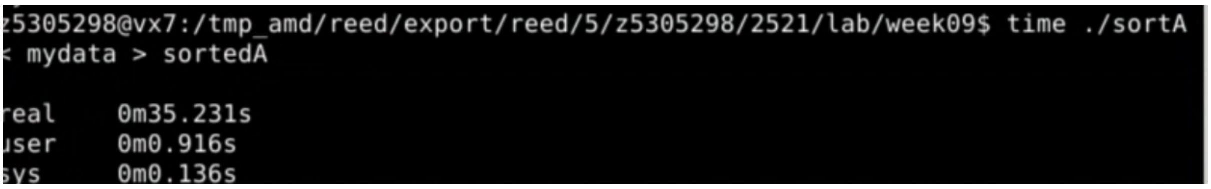
5000000 D

1st time



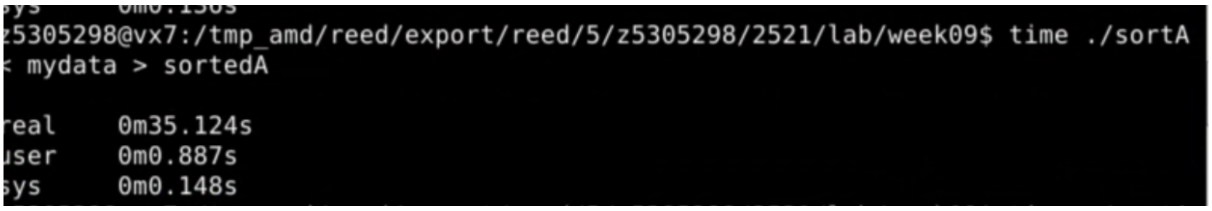
```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m37.501s
user    0m0.940s
sys     0m0.112s
```

2nd time



```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m35.231s
user    0m0.916s
sys     0m0.136s
```

3rd time



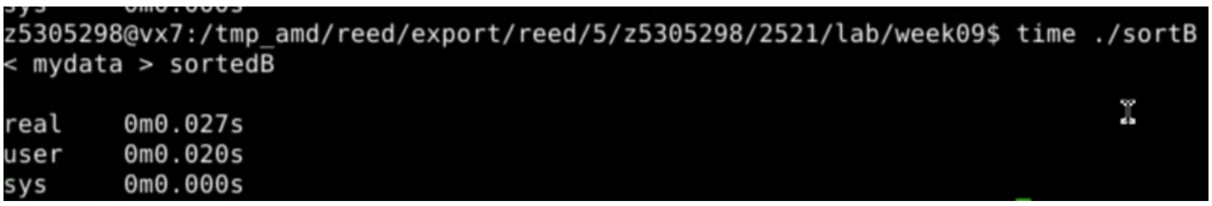
```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortA
< mydata > sortedA

real    0m35.124s
user    0m0.887s
sys     0m0.148s
```

These observations indicate that the algorithm underlying the program A consumed about 36s to run the dataset 5000000 R. Since sortA sorted ascending order datasets faster than descending order datasets, it should be insertion sort. Plus the stability tests turned out to be stable, suggesting that program A uses insertion sort.

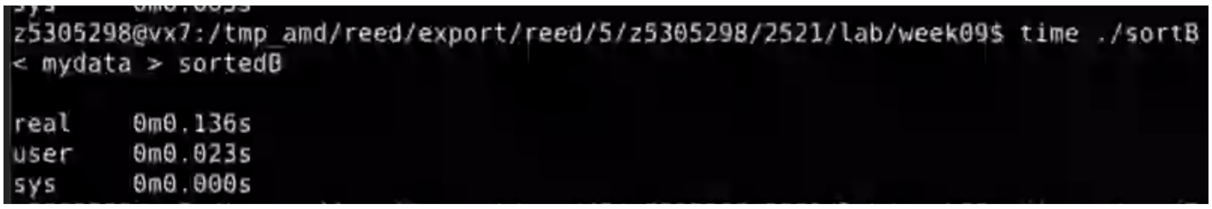
For Program B, we observed that …

50000 R

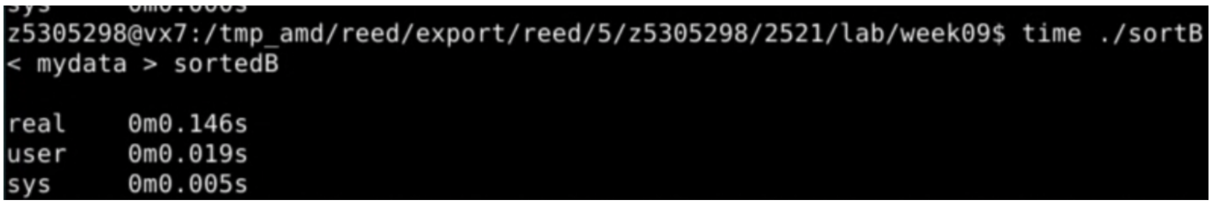
1st time


```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortB
< mydata > sortedB

real    0m0.027s
user    0m0.020s
sys     0m0.000s
```

2nd time


```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortB
< mydata > sortedB

real    0m0.136s
user    0m0.023s
sys     0m0.000s
```

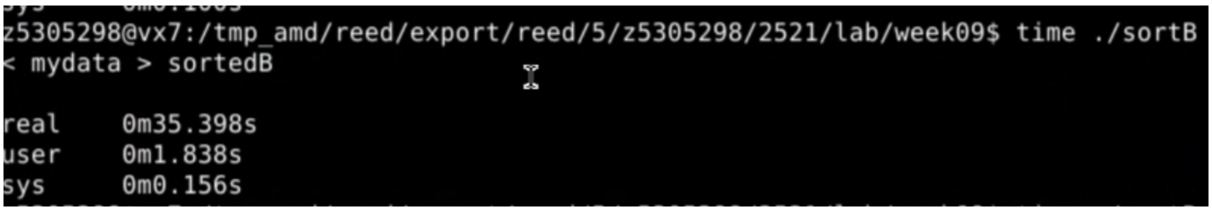
3rd time


```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortB
< mydata > sortedB

real    0m0.146s
user    0m0.019s
sys     0m0.005s
```

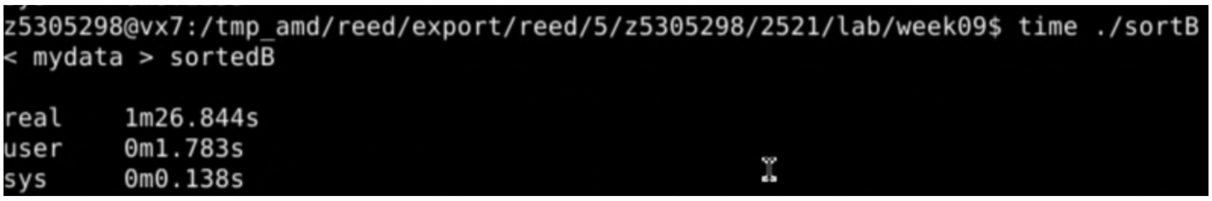
These observations indicate that the algorithm underlying the program B consumed about 0.08s to run the dataset 50000 R.

5000000 R

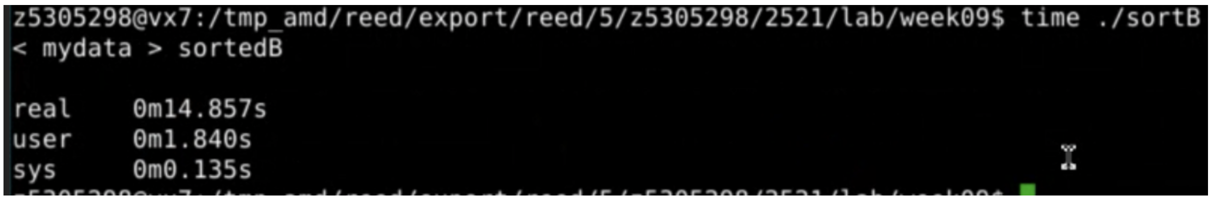
1st time


```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortB
< mydata > sortedB

real    0m35.398s
user    0m1.838s
sys     0m0.156s
```

2nd time


```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortB
< mydata > sortedB

real    1m26.844s
user    0m1.783s
sys     0m0.138s
```

3rd time



```
z5305298@vx7:/tmp_amd/reed/export/reed/5/z5305298/2521/lab/week09$ time ./sortB
< mydata > sortedB

real    0m14.857s
user    0m1.840s
sys     0m0.135s
```

These observations indicate that the algorithm underlying the program B consumed all different time to run the dataset 5000000 R. The results were so unstable that we cannot give a precise average time. Since only quicksort generates random pivot, consumed time can vary largely. Hence sort B should be randomised quicksort.

Since it was obvious that program B is a randomised quicksort, no further experiments were needed.

Conclusions

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

- sortA implements the *insertion sort* sorting algorithm.
- sortB implements the *randomised quicksort* sorting algorithm.