ECE590: Project 1 Report

Jiaxi Yin (jy280)

Yutong Zhang(yz566)

1 Introduction

In this project, five different sorting algorithms are implemented and compared. Algorithms including Selection sort, Insertion sort, Bubble sort, Merge sort and Quick sort. Their theoretical time complexity is analyzed, and experimental performance is tested with different number of trials. Then, conclusions about the best algorithm are made based on testing results and discussion.

2 Methodology

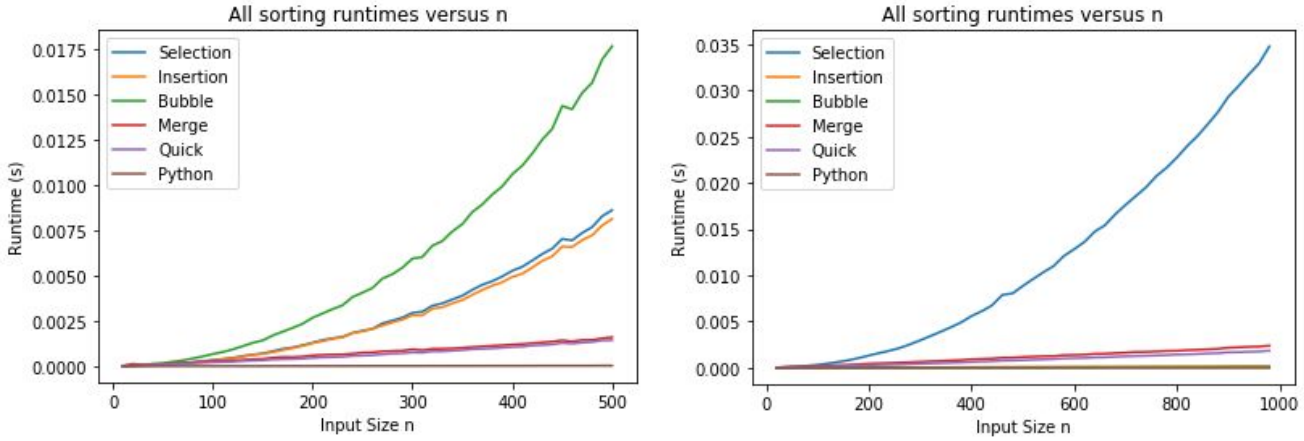
For each of the algorithms, we do test with 30 trials and compute the experimental average time. One trial may be affected by other irrelevant factors other than computing speed. This will cause a big bias. Thus, averaging across multiple trials helps minimize the effect of irrelevant factors. Another point worth noting is that performing a computationally expensive task in the background will slow down the computer and increase the runtimes. We should ensure the background is cleared while running the test task.

We also compare the runtime of algorithms with their theoretical runtimes. Because the operation speed and performance of different computers are variable, analyzing theoretical values gives a more accurate result. Comparing experimental runtimes and theoretical runtimes also helps to check the correctness of the codes implemented. Moreover, for large dataset, theoretical runtimes provide more useful comparisons because running all the algorithms and comparing them are inefficient. For small dataset or unconventional dataset, experimental runtimes provide more useful comparisons because actual operation is performed. It is convincing.

The value of n matters. If n is small, experimental runtimes for different algorithms are tiny because very few operations are needed. We cannot make comparisons among algorithms. The larger the n, the greater the difference in runtimes of these algorithms. Thus, our tests are implemented on different input sizes for algorithms, including large size inputs. Moreover, we report the theoretical runtimes for large n. As and the theoretical runtimes for small size inputs will be derived as a constant for all algorithms. The essential theoretical runtime difference among algorithms is the different performance of algorithms when dealing with large input sizes.

3 Results and Discussion

Algorithm	Average Case	Best Case	Worst Case
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n)$	$O(n^2)$
Bubble Sort	$O(n^2)$	$O(n)$	$O(n^2)$
Merge Sort	$O(nlogn)$	$O(nlogn)$	$O(nlogn)$
Quick Sort	$O(nlogn)$	$O(nlogn)$	$O(n^2)$



The table above is time complexity of average cases, best case and worst case for five algorithms that are tested. And the left figure above is runtimes of unsorted arrays for six algorithms(including python built-in function) with 30 trials while the right figure above is runtimes of sorted arrays with 30 trials. The table below is algorithms' log-log slop generated from tests with input size larger than 200.

As we can see on the table above, Merge sort and Quick sort have theoretical time complexity O(nlogn) on average, whereas Selection sort, Insertion sort and Bubble sort have O(n^2) on average. Thus, for sorting unsorted lists, we expect that Quick sort and Merge sort have lower experimental run time than run time of the other three algorithms. The test results meet our expectation on this point, in the left figure, we can clearly find that the runtimes of Quick sort and Merge sort are significantly lower than those of Selection sort, Insertion sort and Bubble sort. Also, the table below also meets the expectation, as the log-log slopes of Selection sort, Insertion sort and Bubble sort are all approximately 2, and the log-log slops of Merge sort and Quick sort are both approximately 1.

Algorithm	log-log Slope (n>200)
Selection Sort	2.195598
Insertion Sort	2.121172
Bubble Sort	2.160650
Merge Sort	1.199753
Quick Sort	1.344397

Based on the implementation of algorithms, we expect that Merge sort, Quick sort and Selection sort are indifferent in runtimes between unsorted arrays and sorted arrays, whereas Insertion sort and Bubble sort perform better with sorted arrays than unsorted arrays. These are consistent with theoretical time complexity. In experimental runtime figures above, we can see that this expectation is fulfilled.

4 Conclusion

According to the theoretical analysis and testing results, Merge sort is the best sorting algorithm while Selection sort is the worst sorting algorithm. The time complexity of Merge sort is the lowest for average case, best case and worst case among five algorithms. The time complexity of Selection sort is the highest for average case, best case and worst case among five algorithms.