Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Estd: 1991 A.D.

LAB-2

Algorithms and Complexity

[Code No: COMP 314]

Submitted by:

Nigam Niraula (32)

Submitted to:

Dr. Rajani Chulyadyo

Department of Computer Science and Engineering

Submission Date: 05/23/2024

LAB DESCRIPTION:

In the second lab, we enhanced on what we learned in the first lab and used unittest library to test different algorithms. Here, we implemented the quick-sort and merge-sort algorithm and wrote the test cases for it accordingly. The algorithms and its corresponding functions were checked as per the requirement using random array, sorted array and empty array. Once the algorithm was successfully tested, the analysis of all the sorting algorithms implemented till now was done. For this, a random array was generated with fixed size (first size being 1000) and the size was increased by 1000 after each iteration. For each iteration, time taken for quick- sort, merge-sort , best cases of merge and quick sort , worst cases for merge and quick sort and finally time taken for insertion sort and selection sort was calculated. Then, graphs for sizes of array vs time was plotted to deduce a relation.

OUTPUT:

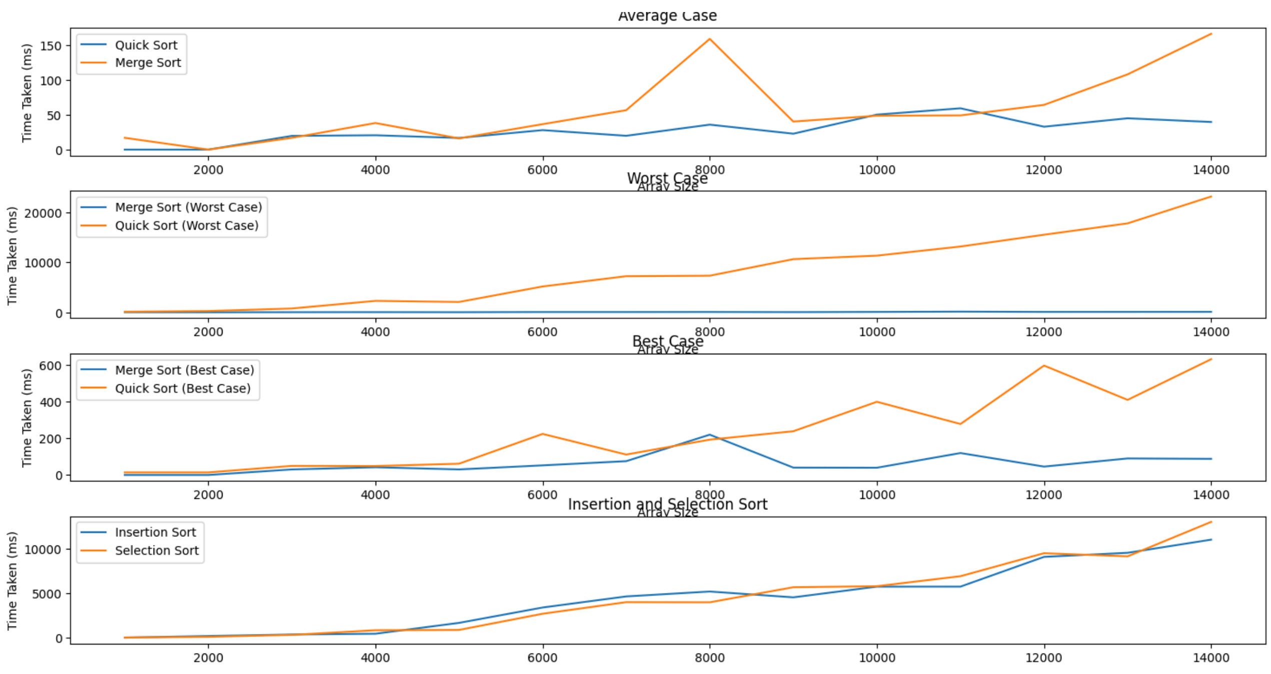

Fig: Graph for different cases

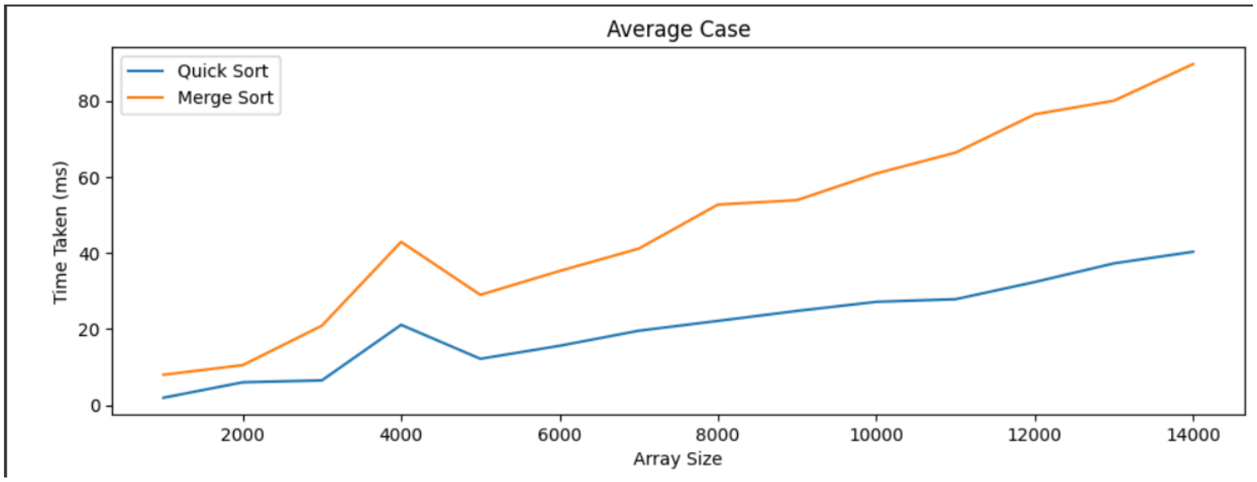

Fig: Array Size vs Time for Quick and Merge Sort (Average Case)

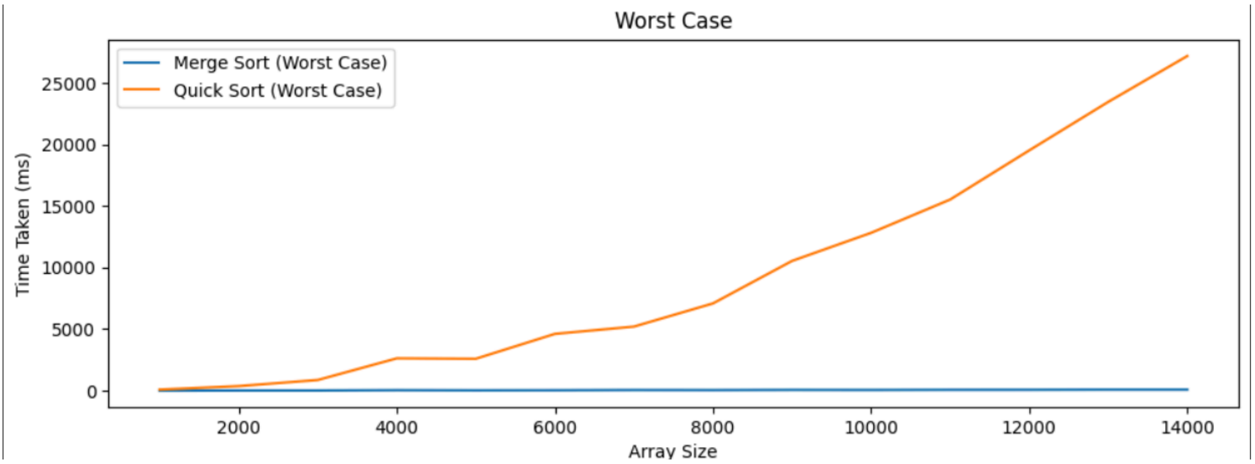

Fig: Array Size vs Time for Quick and Merge Sort (Worst Case)

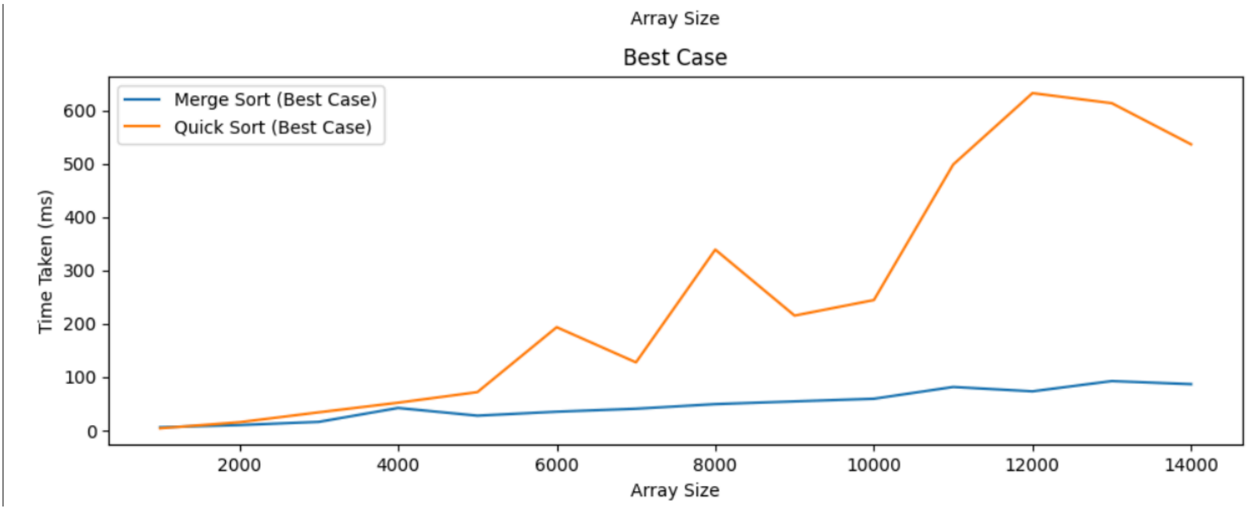

Fig: Array Size vs Time for Quick and Merge Sort (Best Case)

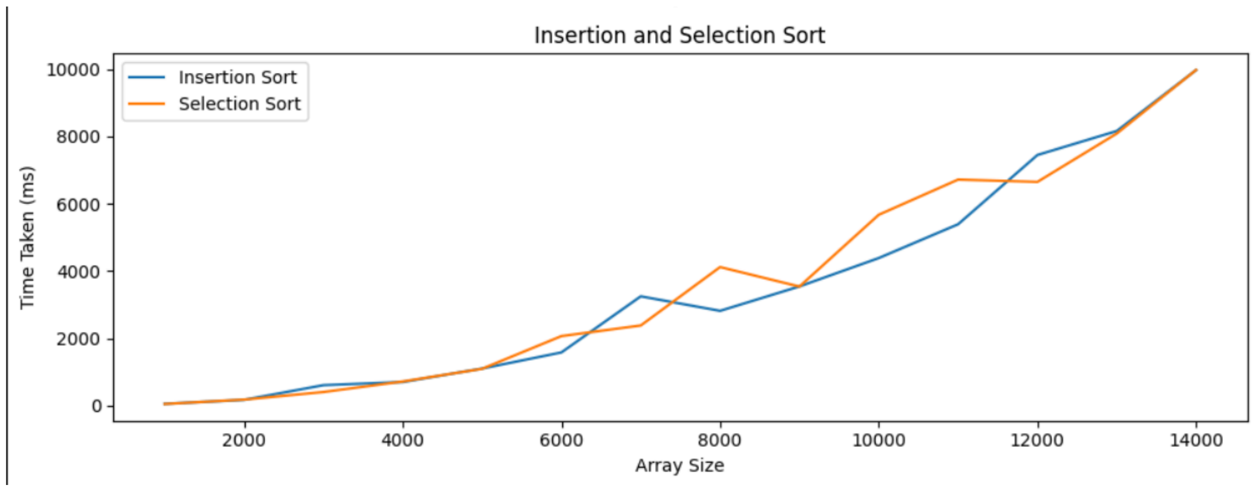

Fig: Array Size vs Time for Insertion and Selection Sort (Average Case)

OBSERVATION:

It was observed that in general for average cases merge and quick sort performed way better than the insertion and selection sort. It is also to be noted that even in the worst case the merge sort performed better than the insertion and selection sort but quick sort performed similarly. The merge sort and quick sort performed exceptionally well for the best cases. In general both algorithms that follow divide and conquer strategy showed a log linear graph. Thus, it could be stated that in average cases both have time complexity O(nlogn). However, it has to be noted that merge sort also has space complexity of O(n). In worst cases, quick sort followed O(n^2) and merge sort still retained O(nlogn) complexity.

SOURCE CODE:

The source code can be found at:
https://github.com/ninix07/Complexity-Labs/tree/master/Lab-2