Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

LAB 2

[Code No: COMP 314]

Submitted by

Paribartan Timalsina

Roll no:56

Group: Computer Engineering

Level: III year/II sem

Submitted to

Dr.Rajani Chulyadyo

Department of Computer Science and Engineering

Purpose

The purpose of this report is to document the implementation, testing, and performance measurement of two sorting algorithms: Quick Sort and Merge Sort. This includes developing the algorithms in Python, writing test cases to verify correctness, generating random input data, and measuring the execution times for both best and worst-case scenarios.

Merge sort and Quicksort:

Merge sort is a sorting algorithm that follows divide and conquer strategy and works recursively by dividing the input array into smaller subarrays and sorting those subarrays then merging them back together to obtain sorted array.Merge sort process is to divide the array into two halves,sort each half and then merge the sorted halves back together.

Quick sort is the sorting algorithm based on the divide and conquer algorithm that picks an element as pivot and partitions the given array around the pivot by placing the pivot in its correct position in sorted array.In quick sort the partition() places the pivot at its correct position and put smaller elements at left of pivot and the greater elements at right.It is done recursively on each side of pivot and finally array is sorted.

Introduction:

In Lab 2, we wrote different algorithms and tested the validity of those algorithms through different test cases. At first, an algorithm for Quick Sort and Merge Sort was written along with the algorithms for partition and merge, and then these functions were tested using the unittest library. The test cases were done for different types of numbers, and all of these cases were run. These tests validated our algorithms and told us if they were correct or not. For validation, we used arrays of different lengths and the numbers in different orders with both positive and negative numbers and compared the returned array as output.

The test cases were written for testing quick sort,merge sort,partition and merge and the test were carried out.The output of running test cases for sorting algorithms are:

```
PS D:\6TH SEM\ALGORITHM AND TIME COMPLEXITY 6TH SEM\algorithm lab 2\tests> python test_sorting.py
.......
----------------------------
Ran 7 tests in 0.001s

OK
PS D:\6TH SEM\ALGORITHM AND TIME COMPLEXITY 6TH SEM\algorithm lab 2\tests>
```

After testing our sorting algorithms we did the analysis of algorithms. For this the number of elements in the array size were made different and their execution time was noted.The array size was increased from 10 to 1000 by the increment of array size by 100 in each iteration.After noting all the execution times using different algorithms like merge sort,quick sort and their best and worst cases it was compared with the insertion sort and selection sort of Lab1 and results were discussed.

Output

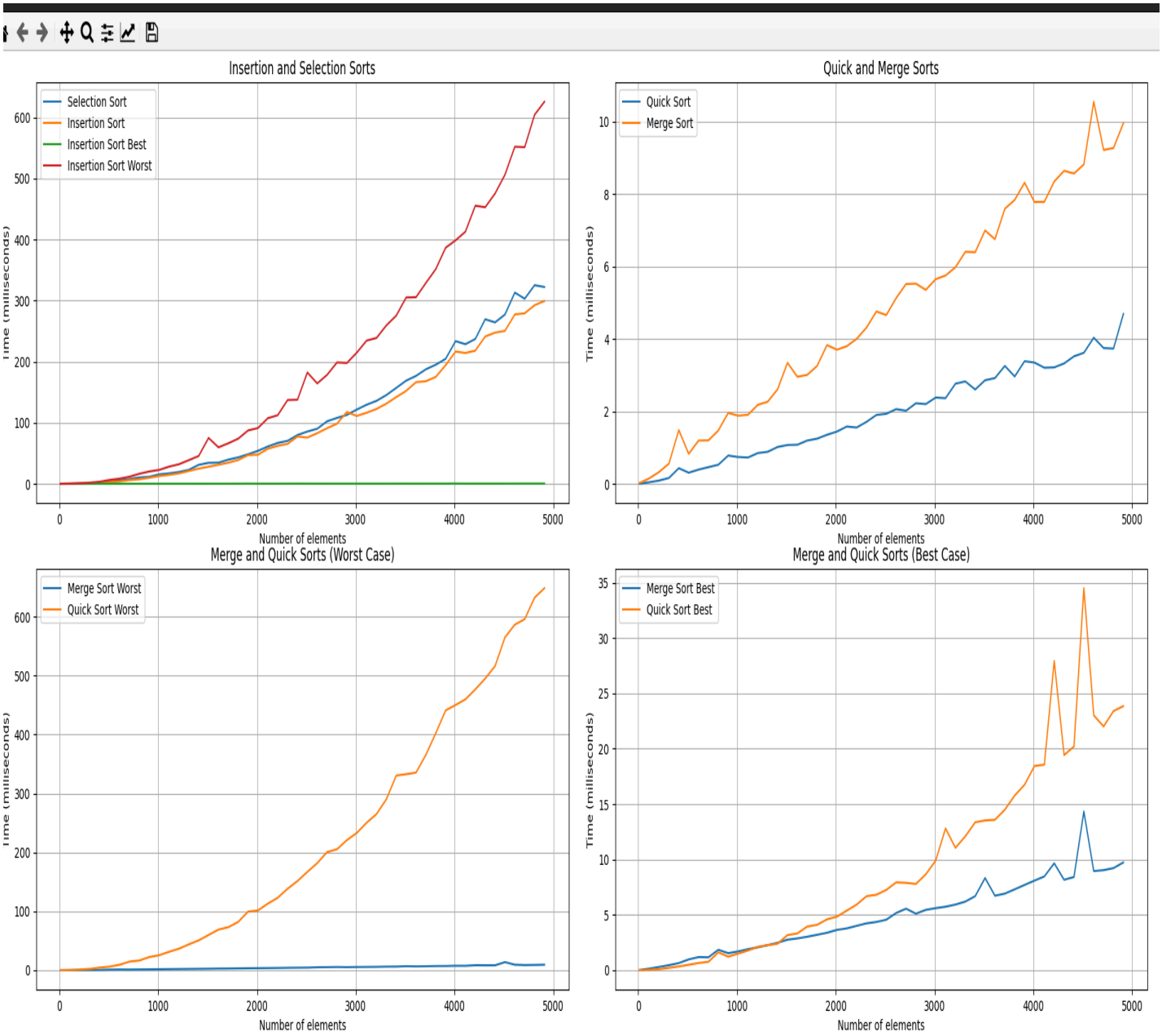

Figure:Array size vs execution time graph

Observations and Results:

From the graph obtained we can see that in the average case the quick sort and merge sort performs way better than the insertion and selection sort.The graph of both quick sort and merge sort shows the nature of log linear (nlogn) where n is the input array size whereas the graph of insertion sort and selection sort shows the quadratic nature so we can compare the time complexity between them in average case where we can see that time complexity of insertion sort and selection sort is $O(n^2)$ whereas for quick sort and merge sort is O(nlogn).In comparison of quick sort and merge sort both shows the same nature of curve in average case where the execution of merge sort is taking slightly longer time than the quick sort.It may be because of the fact that merge sort also has the space complexity of O(n)

The best case of both quick sort and merge sort is shown in graph where we can see that nature of curve is log linear.The best case of merge sort and the average case has the same nature but in the quick sort best case it is taking longer time even than the average case it might be due to implementation error or due to the problem of recursion limit that we face during computation.The worst case of quick sort has the quadratic property which is also shown in the graph and the execution time taken is longer as compared to average and best case.On the other hand the merge sort worst case is log linear and it is shown in the graph so we can conclude by saying that the insertion sort and selection sort are outperformed by merge sort and quick sort in average case but in the best case the insertion sort is linear and it outperforms all other algorithms.Similarly in the worst case the merge sort is way better than quicksort as seen in the graph.

Algorithm	Time Complexity		
	Best	Average	Worst
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$

Quick Sort	O(n log n)	O(n log n)	$O(n^2)$
Merge Sort	O(n log n)	O(n log n)	O(n log n)

The source code is available at:
https://github.com/Paribartan-Timalsina/COMP314-LABS.git