# Kathmandu University
## Department of Computer Science and Engineering

**Dhulikhel, Kavre**

**Report of Lab2 : Merge Sort and Quicksort**

[Code No: COMP 314]


**Submitted by:**

Abiral Adhikari (02)


**Submitted to:**

Dr. Rajani Chulyadyo

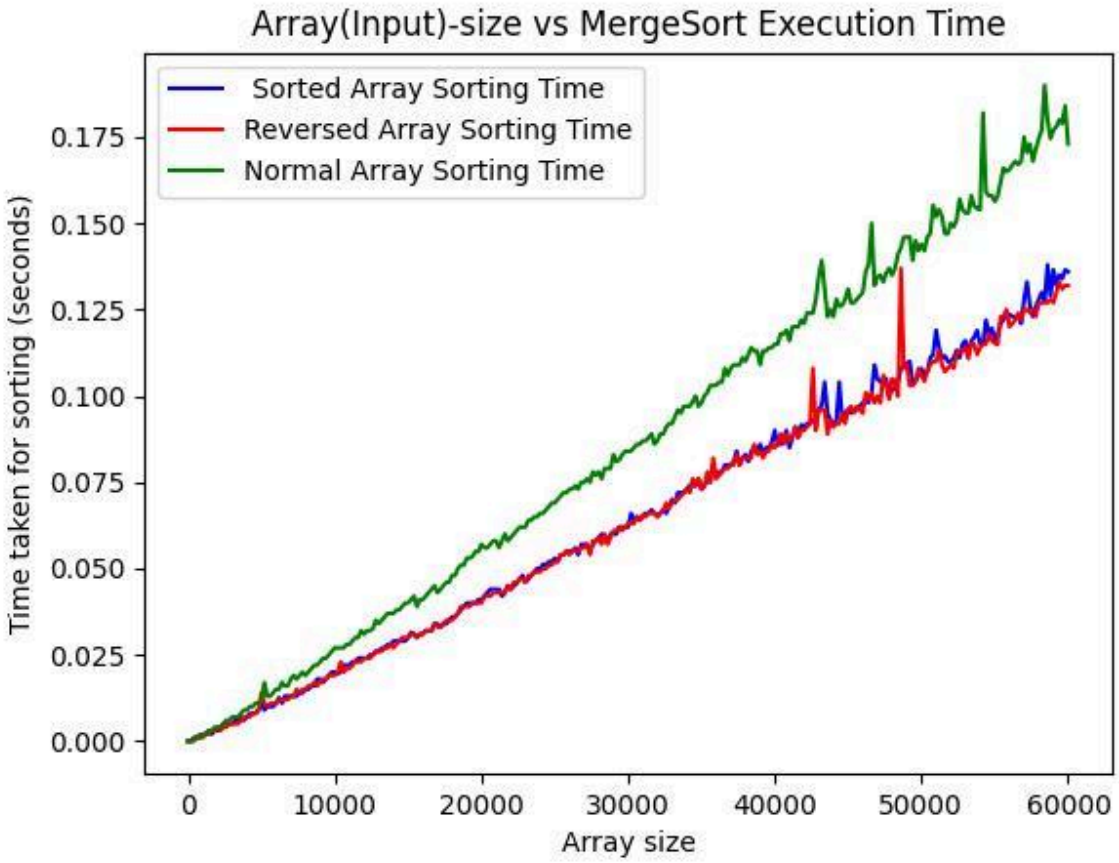
Department of Computer Science and Engineering

**Submission Date:** 23/05/2024

**Task 3 :**Generate some random inputs for your program and apply both quick sort and merge sort algorithms to sort the generated sequence of data. Record the execution times of both algorithms for best and worst cases on inputs of different sizes. Plot an input-size vs execution-time graph.

## Merge Sort:

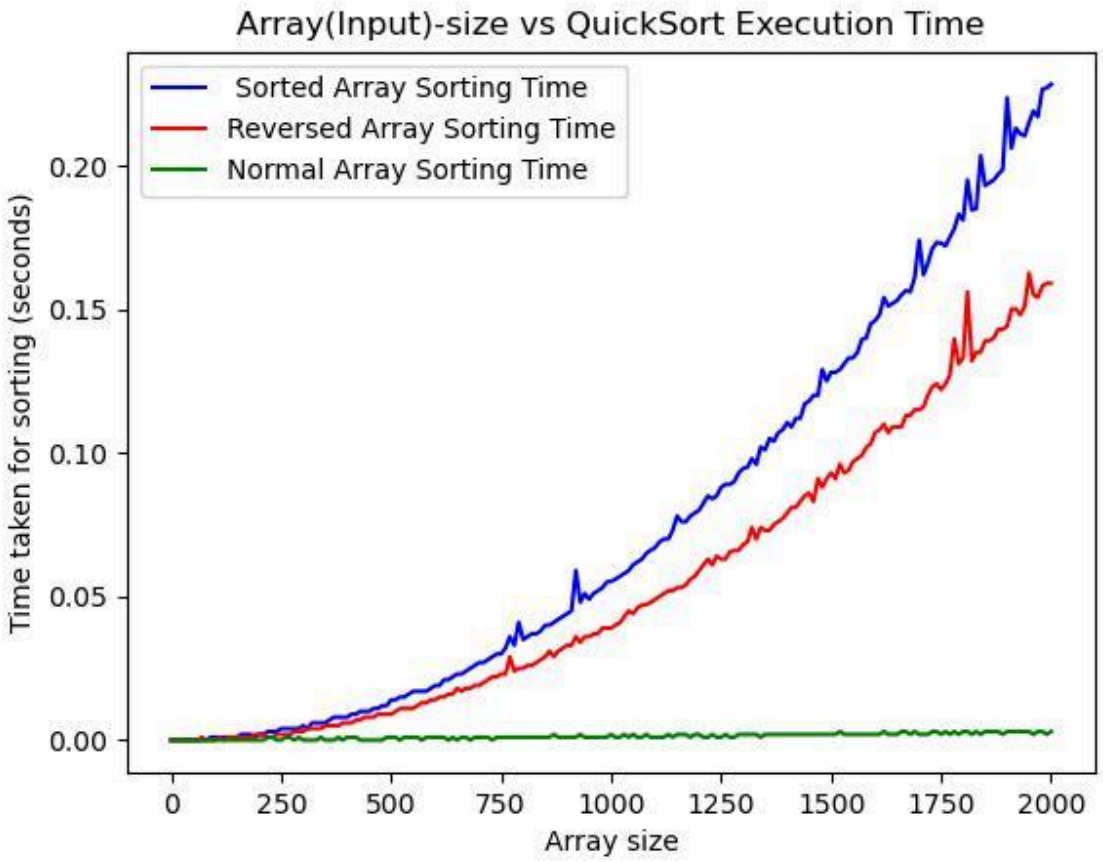
Merge sort is a divide-and-conquer sorting algorithm where an unsorted list is broken down recursively into sublists containing one element, then merging those sublists back together in sorted order. This process continues until the entire list is sorted. The time-complexities for best, average and worst case of merge sort is shown below:



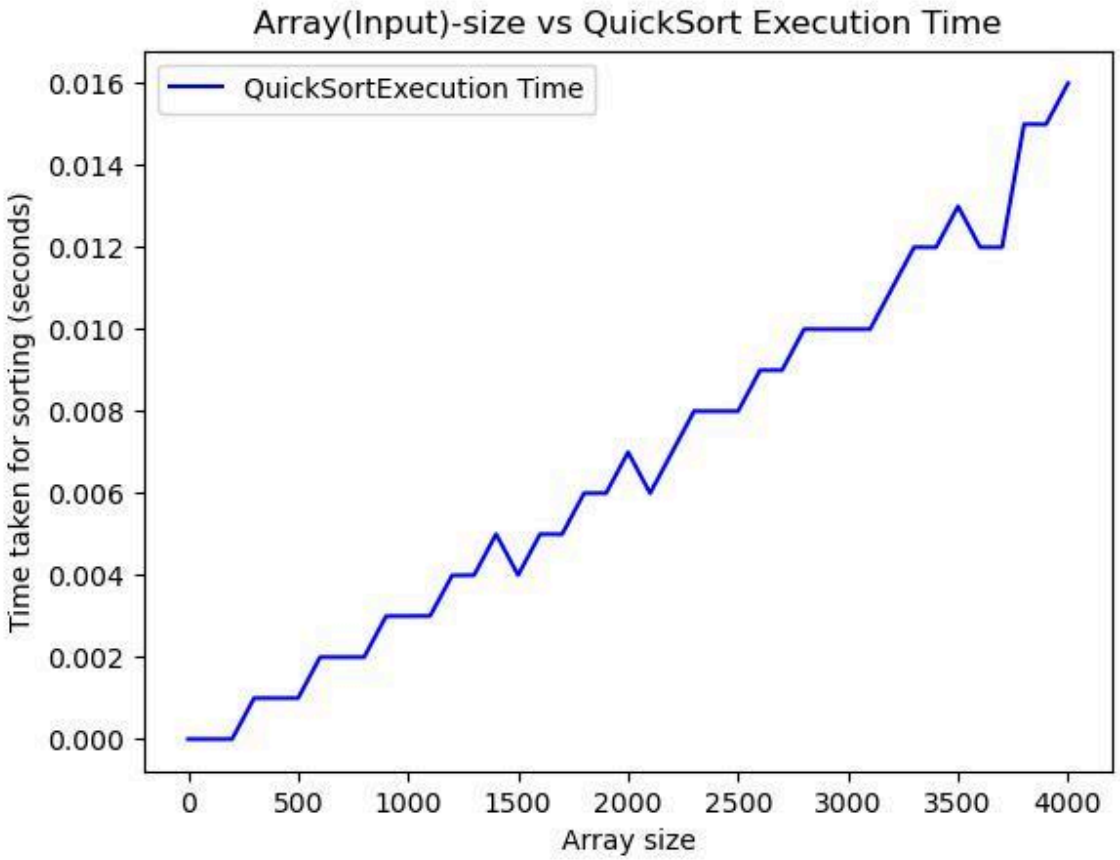

From the above graph we see that even though there is slight difference in the sorting time of the sorted, reverse-sorted and normally distributed array the nature of the graph is same and we see that the time-complexity of the merge sort in all cases(best, average and worst) is O(n.logn). So no matter the case the sorting time complexity of the merge sort is the same.

## Quick Sort:

Quicksort is a divide-and-conquer sorting algorithm where a pivot element is selected from the list and pivot is swapped with the last element. Then the list is rearranged such that elements less than pivot are on its left and elements greater are on its right. Then the process is repeated recursively for the left and right sub-lists.This also uses divide and conquer strategy similar to merge sort. The time-complexities for best, average and worst case of quick sort shown below:



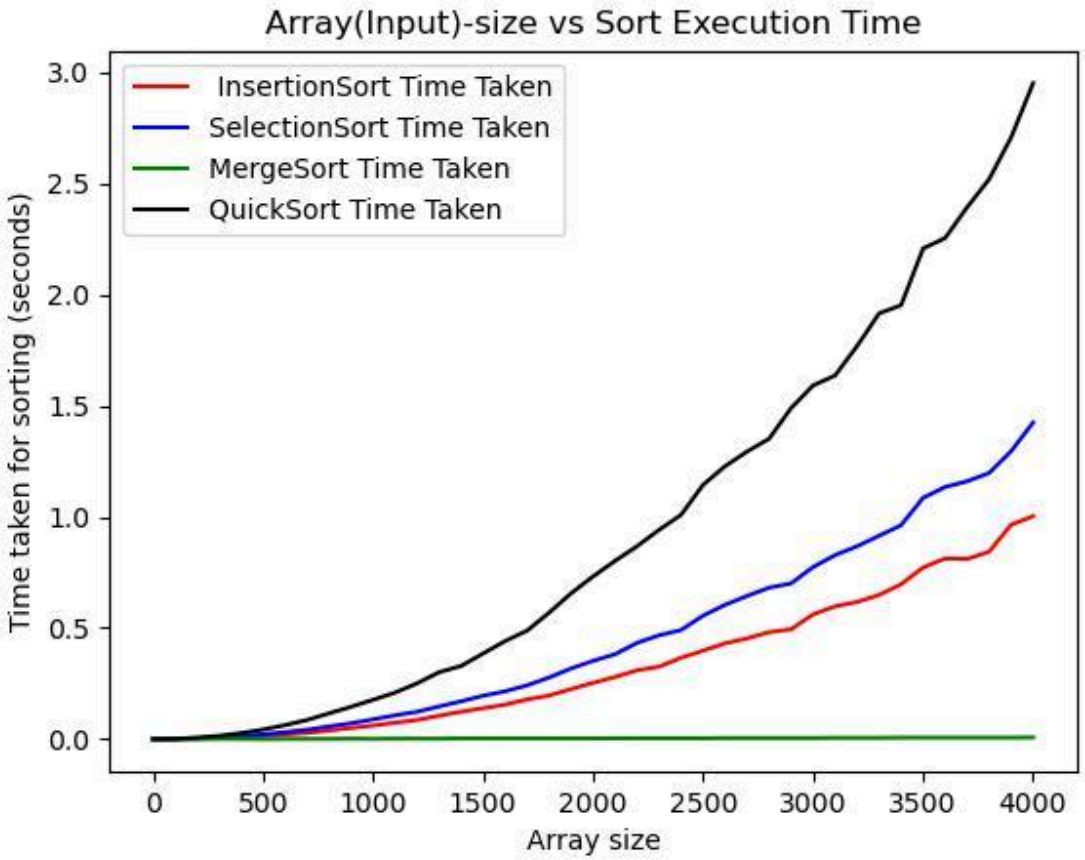

For the quicksort sorted array in any order makes the worst case for the sorting as the pivot element selected is the largest or smallest element so the partition gives maximum skewness. From the graph we see that the worst case of the quick sort is O(n^2). Though the graph shows for the average case of sorting to be constant it's due to presence for a large range of data in time taken. For the average case we plot again for it only as following:




From above graph we see the best case and average case of the Quick Sorting is O(n.logn)

**Task 3 :**Compare your results with those from Lab 1, and explain your observations.

Following graph can be used to compare the results between the sorts used in Lab1 and Lab2:




From the above graph we see that the time complexities for worst case for insertion sort, selection sort and quicksort is the O(n^2) while that of merge sort is O(n.logn) In general we find the quick sort to be better than all the sorts implemented followed by merge sort , insertion sort and selection sort. But merge sort is the best considering all the cases from the observation of the graph plot of array-size vs time taken for the sorting.