Name: Shaan Agarwal
UID: 2021300001
Batch: Comps A (A1)
Experiment No: 2
Date Of Performance: 9th February, 2023

Aim: To compare two sorting algorithms – Merge Sort and Quick sort, based on their running time.

Theory:
Merge sort and quick sort are yet another crucial sorting algorithms that are comparatively much faster than the insertion or selection sort. These algorithms are based on the divide and conquer approach of solving a problem. In other words, recursion plays an important role in their implementation. Below is a brief description of these two sorting algorithms.

Merge Sort:
As mentioned earlier, it words on the divide and conquer approach. In this type of sorting, the array that is to be sorted is broken into smaller and smaller group of elements virtually till each group consists of a single element. Further, these groups are merges in such a fashion that each resulting group is a sorted

group of elements. Doing this recursively ends up in sorting the entire array.

Quick Sort:
In quick sort, the elements are placed at their respective places based on recursive partition. This partition happens around a reference element known as pivot. Considering the sorting being done for obtaining the ascending order of elements, the ones lesser than the pivot are placed to its left, while others are placed to its right. For partitioning an array around a pivot, usually two pointers are used. Any element can be chosen to be the pivot. However, usually first or last element is selected to act as the pivot. When the process of partitioning the array is completed, we get the left unsorted array and the right unsorted array. These are also sorted recursively using the logic of quick sort.

Algorithm:

1) Merge Sort

```
MergeSort(A, p, r):
    if p > r
        return
    q = (p + r) / 2
    mergeSort(A, p, q)
    mergeSort(A, q + 1, r)
    merge(A, p, q, r)
```

2) Quick Sort

```
quicksort(array, leftmostIndex, rightmostIndex)
  if (leftmostIndex < rightmostIndex)
    pivotIndex <- partition(array,leftmostIndex,
rightmostIndex)
    quicksort(array, leftmostIndex, pivotIndex – 1)
    quicksort(array, pivotIndex, rightmostIndex)

partition(array, leftmostIndex, rightmostIndex)
  set rightmostIndex as pivotIndex
```

```
    storeIndex <- leftmostIndex – 1
    for i <- leftmostIndex + 1 to rightmostIndex
    if element[i] < pivotElement
      swap element[i] and element[storeIndex]
      storeIndex++
    swap pivotElement and element[storeIndex + 1]
return storeIndex + 1
```

Program -> The program is attached as a separate file.

Result-> Thus we have implemented merge sort and quick sort algorithm. We have ran it on 1,00,000 randomly generated numbers and done the analysis on it.

Conclusion : After performing the above experiment, I have thoroughly understood the implementation of merge sort and quick sort algorithm, and have understood the analysis between the two of them.