***Abstract:***

The purpose of this assignment is to implement the execution of radix sort using two stable sorting algorithms i.e., **Count Sort** and **Insertion Sort.** The algorithms are tested against an input of “m” random strings each having a maximum length of “n”. The Count Sort version is tested for values of n being 100000, 250000, 500000, 750000, 1000000, 2500000, 5000000 and m being 50, 70, 90, 100. The Insertion Sort version is tested for value of n being 10000, 25000, 50000, 75000, 10000 against value of m being 25, 50, 75.

***Results:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **M** | | | |
| **N** | **50** | **70** | **90** | **100** |
| **100,000** | **0.4ms** | **0ms** | **0.4ms** | **0ms** |
| **250,000** | **0ms** | **0ms** | **0.2ms** | **0ms** |
| **500,000** | **0.2ms** | **0ms** | **0ms** | **0.2ms** |
| **750,000** | **0ms** | **0ms** | **0ms** | **0ms** |
| **1,000,000** | **0ms** | **0ms** | **0.2ms** | **0ms** |
| **2,500,000** | **0ms** | **0ms** | **0.2ms** | **0ms** |
| **5,000,000** | **0ms** | **0ms** | **0ms** | **0.2ms** |

*Table1: For Radix Sort Using Count Sort as Stable Sorting Algorithm*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **M** | | |
| **N** | **25** | **50** | **75** |
| **10,000** | **0.4ms** | **0ms** | **0.4ms** |
| **25,000** | **0ms** | **0ms** | **0.2ms** |
| **50,000** | **0.2ms** | **0ms** | **0ms** |
| **75,000** | **0ms** | **0ms** | **0ms** |
| **100,000** | **0ms** | **0ms** | **0.2ms** |

*Table2: For Radix Sort Using Insertion Sort as Stable Sorting Algorithm*

***Discussions:***

Since the execution time of each execution is ranging from [0ms - 0.4ms], the execution can not be visualized graphically but it can be inferred that in radix sort, upon each iteration, the maximum number of integers which will be sorted are in the range of [1,m] and each digit will be in the range of [ ASCII (a), ASCII (z) ].

***Evaluation of Results:***

Since the testing condition and environment for both sorting algorithms is different, there is no possible comparison between the two execution techniques. The individual analysis can be done, and it is safe to assume that the insertion sort would take more time as the number of strings into the array increases, and as the number of elements increase, the comparisons to be done in each iteration will also increase.  
On the other hand, with the use of count sort, there is no such comparison in real and the size of array is not going to form any influence on the overall number of comparison as it will only increase the count of some specific element at position “d” of the string in process.

**Solution to Recurrences**

**Part01:**

**Part02:**

**Part03:**

**Part04:**