ITWS Project

Pradyumn Agrawal 15UCS096 Susmit Lavania 15UCS148 Akash Bhudaulia 15UEC009

I. Introduction to the problem

We were given five sorting algorithms in which four were provided with algorithms and we were asked to code the one named selection sort. We are asked to make a library file containing all the functions of sorting algorithms returning the number of comparisons value for each sorting algorithm for a provided amount of input data. Since all five functions sort an array of integers which were initialied here as **data[n]**. Your program should compute the total number of comparisons made between the elements of the array for each function until it sort out the full array. For example, if input array is 14,18,1,5,78 then output array is 1,5,14,18,78. This array is fully sorted by all the algorithms but the ways are different and thus the **Time Complexity** varies with respect to the amount of data given and the algorithm used. We uses count variable for each sorting algorithm which is intialised to zero value at starting (e.g. **countbubble**) is intialised variable for **bubble sort**.In this way total number of comparisons taking place are calculated. We have five variables for five functions. You may make function calls to each of these functions from main() as follows:

II. More Points

1) We will calculate each one of sorting function separately to achieve the foolproof results.
2) We will take same raw data for all the sorting algorithms to arrive to correct conclusion.
3) Collect the total comparisons result for different value of n (input amount data) for each algorithm and output each of them in .txt's files.
4) We will plot the **GNU Plot** for each of the function to know the efficiency of each sorting algorithm.On X axis we will have the number of elements in the array or the value of n and on Y axis the total number of comparisons to sort the array.

III. Preparing the data

We will finally plot the data of all the data with varying values of input amount data. We will input the data with the help of **bash script** and then finally iterate the data with varying values of n having stepping of 100 starting from 1 and targetting to the value of 10000.

We will need *global* variable such as countbubble (for only bubble sorting). Total time taken by your program :On Linux, command time ./a.out will give you the amount of time taken by your program to run which includes time by **user**,**real** and **sys**.

Details of behaviour of algorithms with labelling is given in figure.

IV. Results

We achieve the results that most of the time **Merge Sort** is fastest sorting algorithm till we have studied so far. But **Quick Sort** is also is a fine algorithm for less amount of data according the data of what we have achieved in the graph.