

Sorting Algo Task

*13/1/2022
Data Structures and
Algorithms
Presented to: Dr.Hesham
Farag*

Contents

Objective: 3

Requirement 4

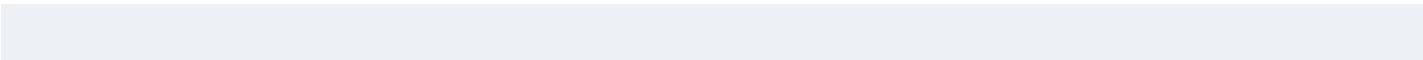
Overview 4

Input..... 5

Sorted Output 5

Plotting and Graphing 6

Comparison Between Different Algorithms 7



Project Contributors and Developers:

Name	ASU ID	UEL ID	Role
Omar Ashraf Mabrouk	19P8102	2140624	Developed GUI , Programmed Interface and bug handling
Hussein Ahmed Hassan Selim	19P9614	2140571	Coded Bubble, Insertion, Selection Sort
Abdelrahman Mohamed Salah	19P9131	2140523	Coded Heap, Merge, Counting Sort.
Zakaria Sobhy Abd El-Salam Soliman Madkour	19P2676	2140654	Coded Files Reading and Output, Developed Counting and Radix Sort to work with negative numbers
Mahmoud Mohamed Seddik	19P3374	2140584	Coded Radix Sort and Selection Sort, Graphed The Output, Tested the software for any bugs

Objective:

The objective is to build an application to compare the efficiency of different sorting algorithms. The application should be able to compare the efficiency of sorting algorithms with their respective asymptotic behavior.

Requirement:

1. Ability to create test data files. Optionally test data files may be created using Microsoft Excel.
2. The user can choose either to compare the efficiency of an algorithm with another algorithm, or to compare an algorithm with its asymptotic efficiency.
3. Ability to select an algorithm out of predefined set of algorithms. A list of sort algorithm must include, but not limited to, the insertion sort, the merge sort, the bubble sort, the quick sort, and the heap sort.
4. The efficiency should be measured as the number of steps taken by the algorithm to sort the test data.
5. The results should be displayed graphically.

Overview:

The Program support different types of inputs such as an input from csv file or just an input written in the input field, it also can generate random inputs and write it in the input box, the user have the option to include -ve numbers by checking the check box under 'Number of Randoms'.

Each function is discussed in details in its related section.

Sorting Algo

Input:

Output:

Get Data from csv file

Write Data to Random.csv

Number of Randoms:

☐ Include -ve numbers

Step:

Generate Randoms

Insertion Sort

Merge Sort

Bubble Sort

Counting Sort

Radix Sort

Quick Sort

Selection Sort

Heap Sort

Execution Time :

Number of Steps :

☐ merge sort ☐ insertion sort ☐ heap sort ☐ bubble sort

☐ quick sort ☐ selection sort ☐ radix sort ☐ counting sort

☐ $O(n^2)$ ☐ $O(n\log)$ ☐ $O(n)$

Plot

Input:

If the input given contains any negative numbers, the user needs to check the **Include -ve numbers** checkbox so **the radix and counting sort can work with -ve numbers**, if this is not checked and the input contains -ve numbers then the result of the counting and radix will be unreliable.

Ways to generate Input:

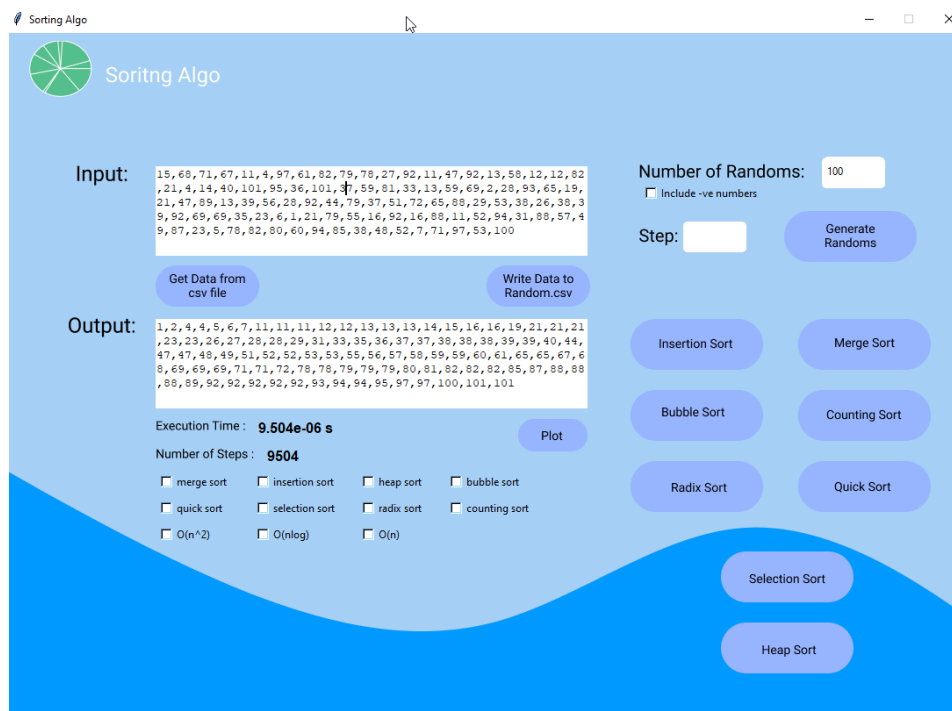
1. Input can be given manually in the input box.
2. Input can be given from csv file
3. Input can be generated using generate random functionality and specifying the number of random numbers wanted.

Note: The user can also write the random numbers to a csv file named **Random.csv** which will be generated during execution if not found.

Sorted Output:

If the user wants to see the sorted array, he can choose any sorting algorithms to sort with by clicking on its corresponding button on the left side of the dashboard. This will write the sorted array into the output box.

For each sorting algorithms, the program will calculate the execution time and number of steps as follows:



Sorting Algo

Input: 15,68,71,67,11,4,97,61,82,79,78,27,92,11,47,92,13,58,12,12,82,21,4,14,40,101,96,36,101,47,59,81,33,13,59,69,2,28,93,65,19,21,47,89,13,39,56,28,52,44,79,37,51,72,65,88,29,53,38,26,38,3,9,92,69,69,35,23,6,1,21,79,55,16,92,16,88,11,52,94,31,88,57,4,9,87,23,5,78,82,80,60,94,85,38,48,52,7,71,97,53,100

Number of Randoms: 100

☐ Include -ve numbers

Step: Generate Randoms

Get Data from csv file Write Data to Random.csv

Output: 1,2,4,4,5,6,7,11,11,11,12,12,13,13,13,14,15,16,16,19,21,21,21,23,23,26,27,28,28,29,31,33,35,36,37,37,38,38,38,39,39,40,44,47,47,48,49,51,52,52,53,53,55,56,57,58,59,59,60,61,65,65,67,68,69,69,71,71,72,78,78,79,79,80,81,82,82,82,85,87,88,88,88,89,92,92,92,92,93,94,94,95,97,97,100,101,101

Execution Time: 9.504e-06 s

Number of Steps: 9504

Plot

☐ merge sort ☐ insertion sort ☐ heap sort ☐ bubble sort

☐ quick sort ☐ selection sort ☐ radix sort ☐ counting sort

☐ O(n^2) ☐ O(nlog) ☐ O(n)

Insertion Sort Merge Sort

Bubble Sort Counting Sort

Radix Sort Quick Sort

Selection Sort

Heap Sort

Plotting and Graphing:

To plot any type and number of graphs follow these steps:

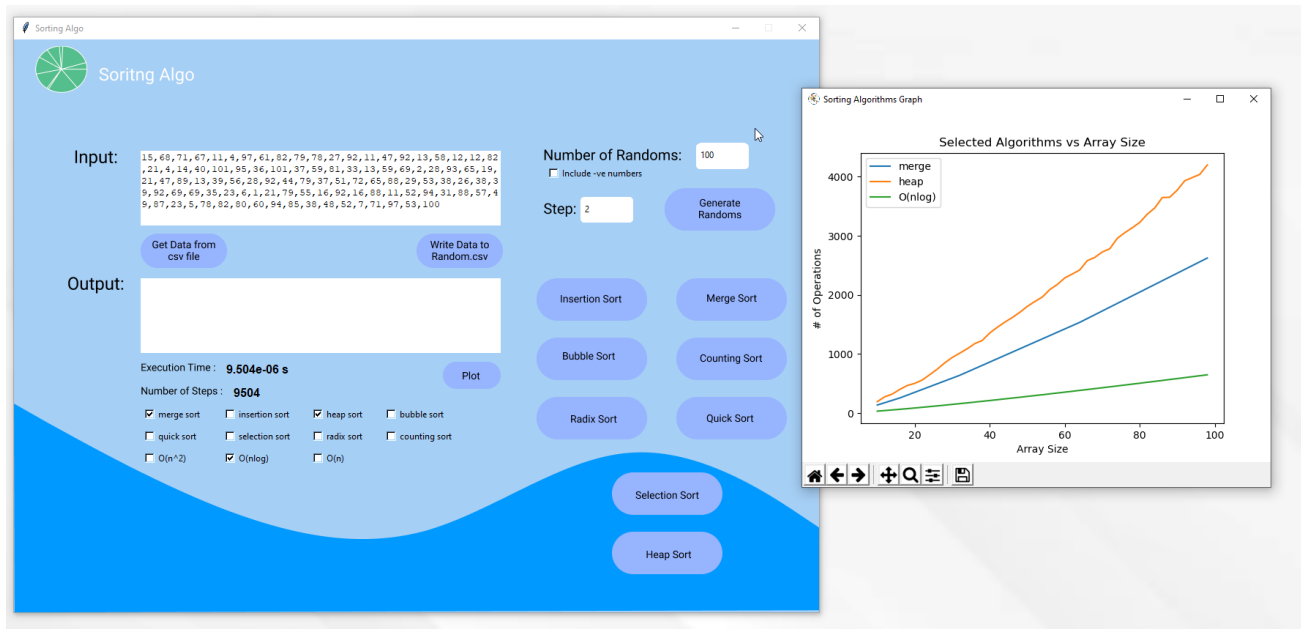
1. The user needs to generate at least a 100 number
2. Specify Step Size (MUST)
3. Choose the sorting/complexity to plot from the checkboxes at the bottom of the program
4. Then click on plot and the graph will be generated with legend

Note: For plotting to work, the user need to generate at least a 100 number in the input.

Note: The Step Size Cannot Be Zero

Note: if in radix sort the number of digits which the input consists of fluctuate, that will result in a spike in the graph.

As follow:



Comparison Between Different Algorithms:

Name	Time Complexity (Best)	Time Complexity (Average)	Time Complexity (Worst)	Space Complexity	Stability
Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	Stable
Selection Sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	Unstable
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	Stable
Merge Sort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$	Stable
Quick Sort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(\log(n))$	Unstable
Heap Sort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(1)$	Unstable
Counting Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$	Stable
Radix Sort	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$	Stable