Design and Analysis of Algorithms (CS-2009)

Project Report

By: Ayihan Ul Haq (BCS-5F)

INTRODUCTION

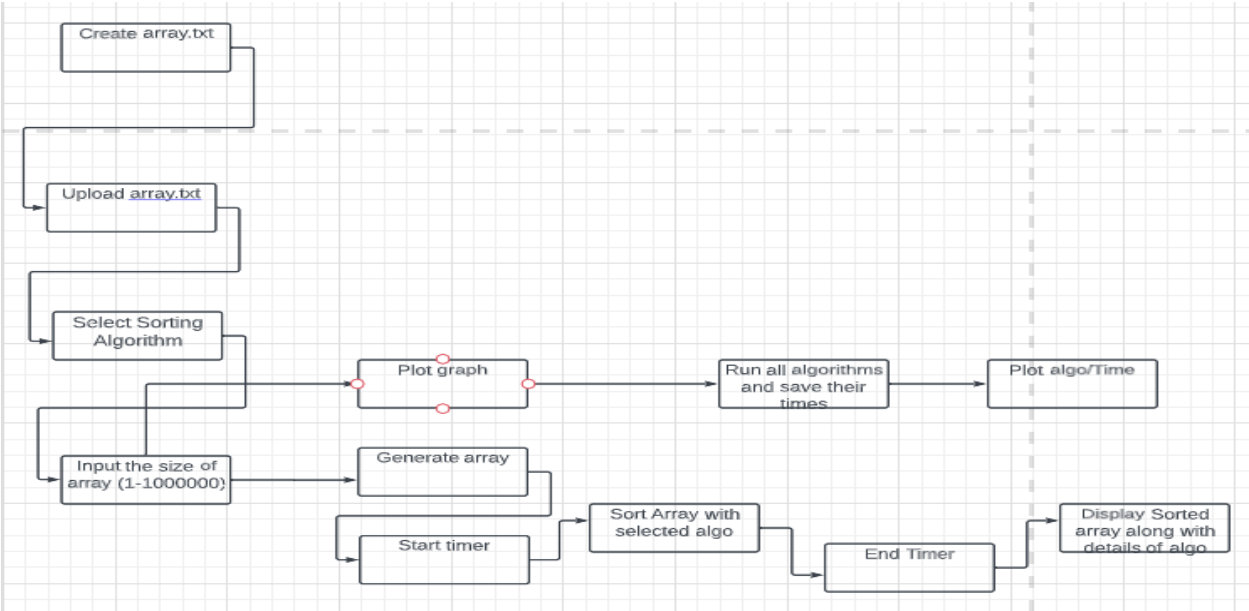
This is the project report compiled for the “Design & Analysis of Algorithms CS-2009” compulsory course project in the Fall 2022 semester which features the comparison of time, Time Complexity and Space Complexity of 10 different algorithms.

ABSTRACT

This project aims to clearly show the comparison between 10 different sorting algorithms; Insertion sort, Bubble sort, Merge Sort, Heap Sort, Quick sort, Radix Sort, Bucket Sort, Courting sort, 7.4.5. from book(hybrid quick sort), 8.2.4. from the book(hybrid count sort). The input array file can contain elements from a range of 1-1000000 that are randomly generated from values 0 - 1000 for accurate representation of computation time of each algorithm.

DESIGN

This project uses Node.Js 18.0, Chart.js and JavaScript for building a interactive webuser interface. The flow starts from generating a array file that is separated by linesNext the user has to upload that file to the website and then select which sortingalgorithm they want to test. The user then enters the desired size of the array they want the algorithm to compute on and clicks generate arrays. The unsorted array and The sorted array is displayed after being computed along with the time taken by the algorithm, its time complexity and space complexity. The user can also choose to press the plot graph button which inturn will run all the algorithms and plot them in a bar chart where the comparison of time is much more clearly visible.



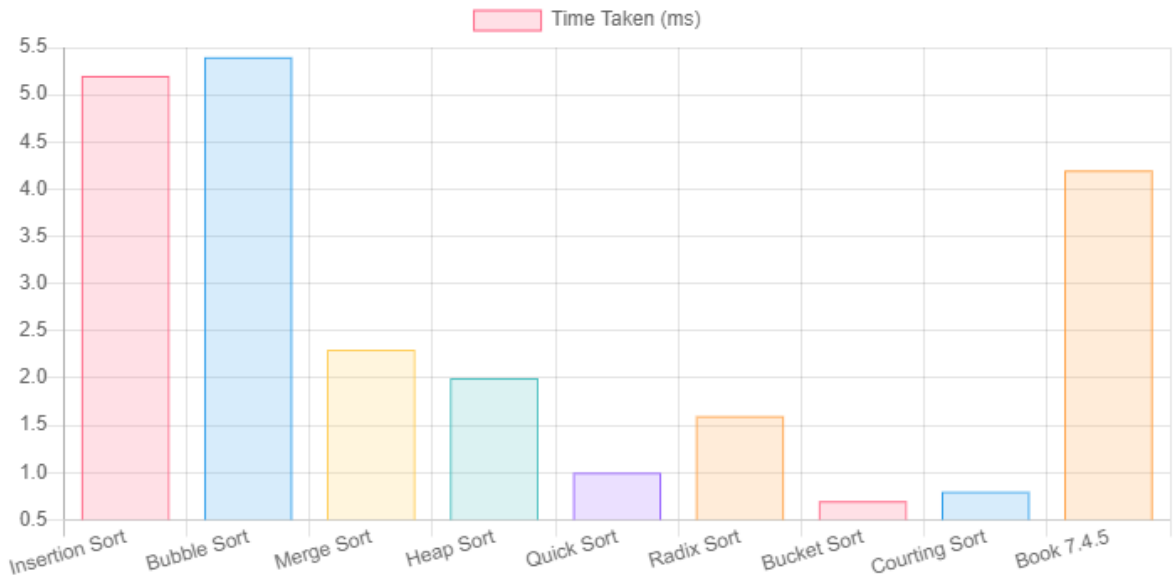
EXPERIMENTAL SETUP

JavaScript being a single threaded language provides a fair comparison between all the different sorting algorithms used, Chart.Js was used to plot all the time graph and the site is hosted on github pages

https://au79912.github.io/algorithms-project/

RESULTS & DISCUSSION

Sample result output on a array of 1000 elements



CONCLUSION

Using a variety of array sizes it can be evidently seen that $O(n^2)$ perform significantly better than O(nlogn) algorithms as long as the array size is comparatively small but they begin to rapidly deteriorate as the array size increases. On average Quicksort performed the best on large arrays and Insertionsort performed the best on smaller arrays in respect to time taken.

REFERENCES

https://developer.mozilla.org/en-US

https://www.chartjs.org/docs/latest/