**FAST National University of Computer and Emerging Sciences**



**Research Topic:** To what extent time complexities can affect performance of sorting algorithms**?**

**Instructor:** Sir Sohail Afzal

**Group Members:**

1. Aahil Halani (20K-0308)
2. Faizan Hakim (20K-0290)

**Date: 6th December, 2022**

**Table of Contents**

1. Introduction -------------------------------------------------------------------------- 3
2. Description ------------------------------------------------------------------------- 3
3. Languages and Tools --------------------------------------------------------------- 3
4. Results ------------------------------------------------------------------------------ 4
5. Analysis ---------------------------------------------------------------------------- 9
6. Evaluation -------------------------------------------------------------------------- 9
7. Conclusion ---------------------------------------------------------------------------- 9
8. References ---------------------------------------------------------------------------- 9
9. **Introduction**

Visualizing the functions of different sorting algorithms, they carry out to sort the data and output their time and space complexity. Purpose of this project is also to check which algorithm works best by changing the size of data and identify the reason for their speed and efficiency by visualizing the steps they take.

1. **Description**

We selected 8 sorting algorithms. Three of them were linear while the rest were comparison based:

The algorithms were as follows:

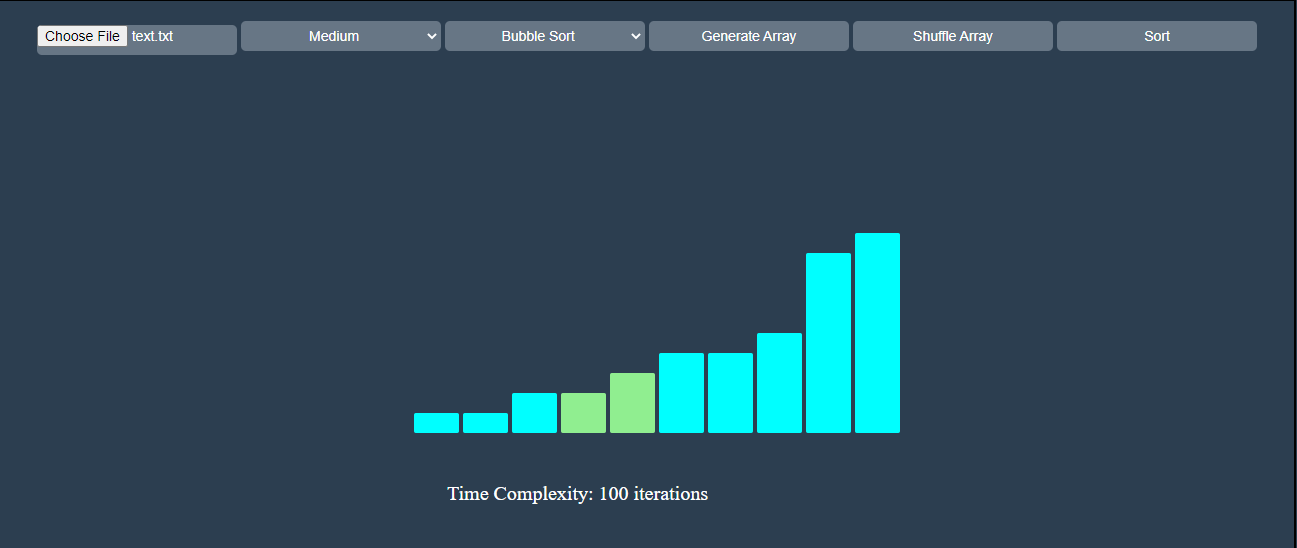
* Bubble Sort:
* Insertion Sort
* Heap Sort
* Count Sort
* Radix Sort
* Bucket Sort
* Quick Sort
* Merge Sort

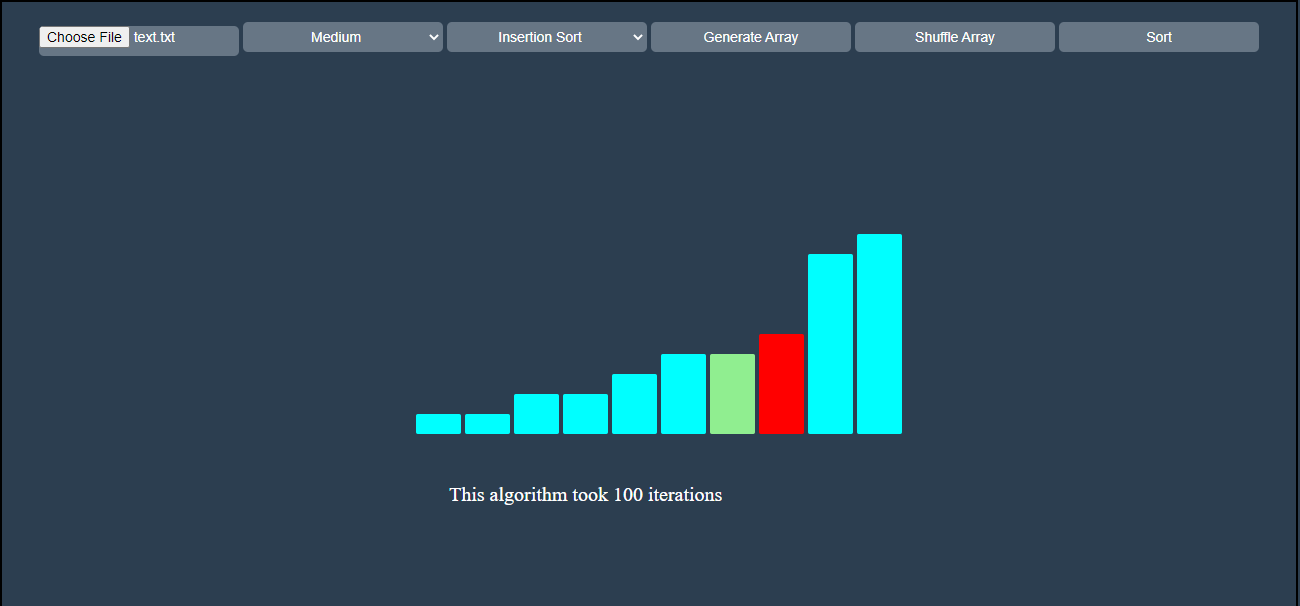
We also tested another two algorithms by optimizing two of them (Count Sort and Quick Sort) from above to observe whether the changes were beneficial or not.

1. **Languages and Tools**

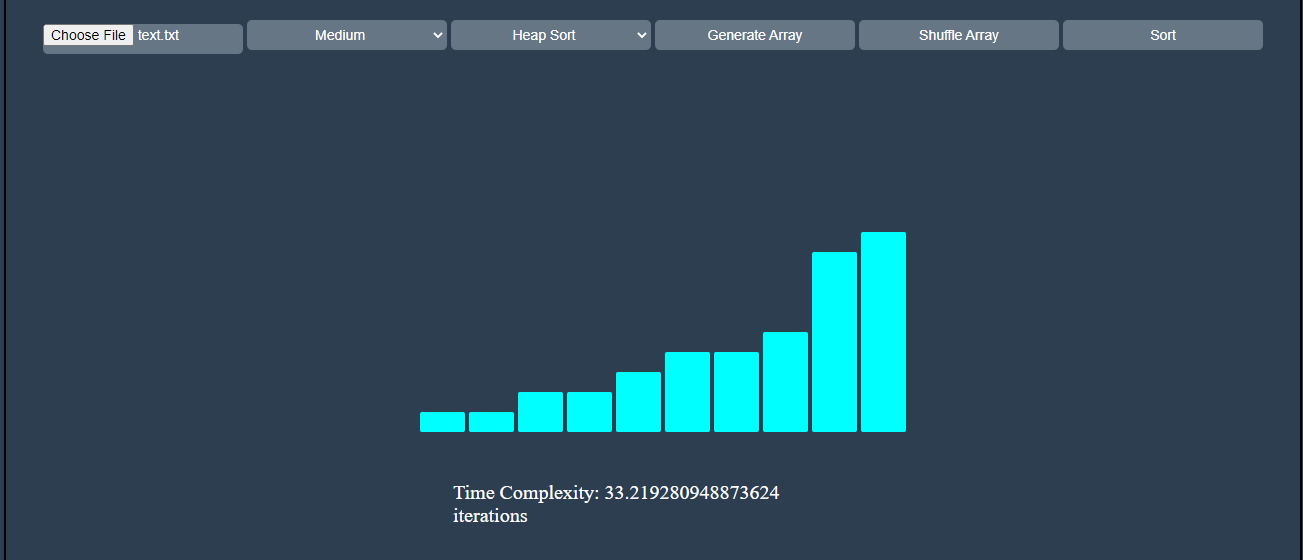
* HTML
* CSS
* Java Script
* IDE: Visual Studio Code

1. **Results**

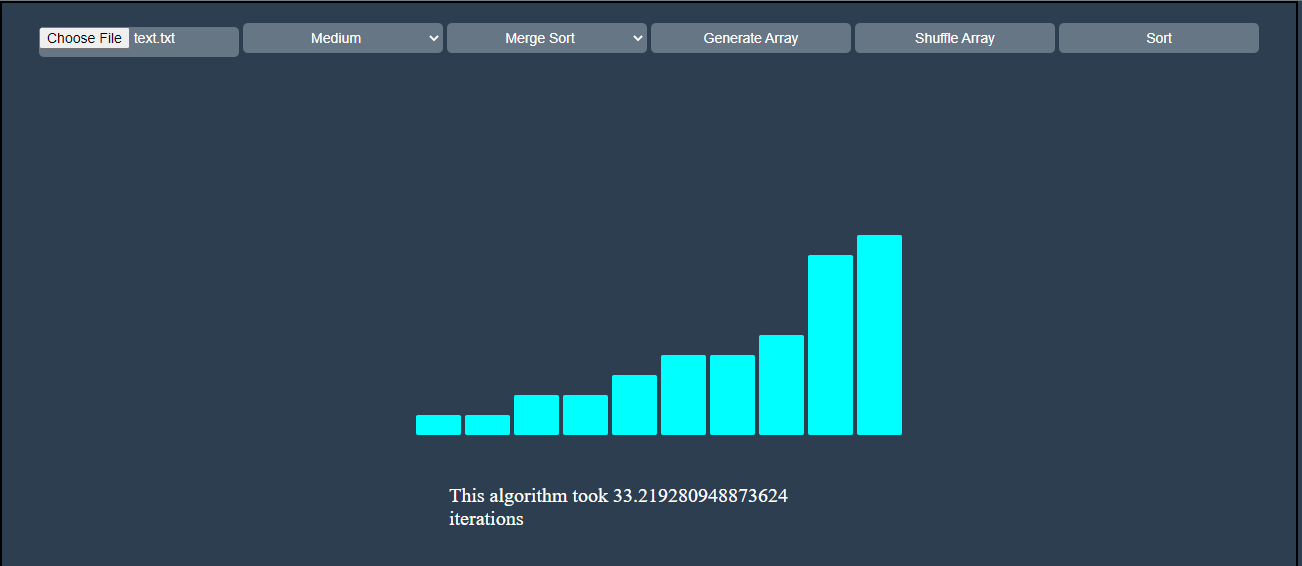
* **Bubble Sort**
* **Insertion Sort**



* **Heap Sort**



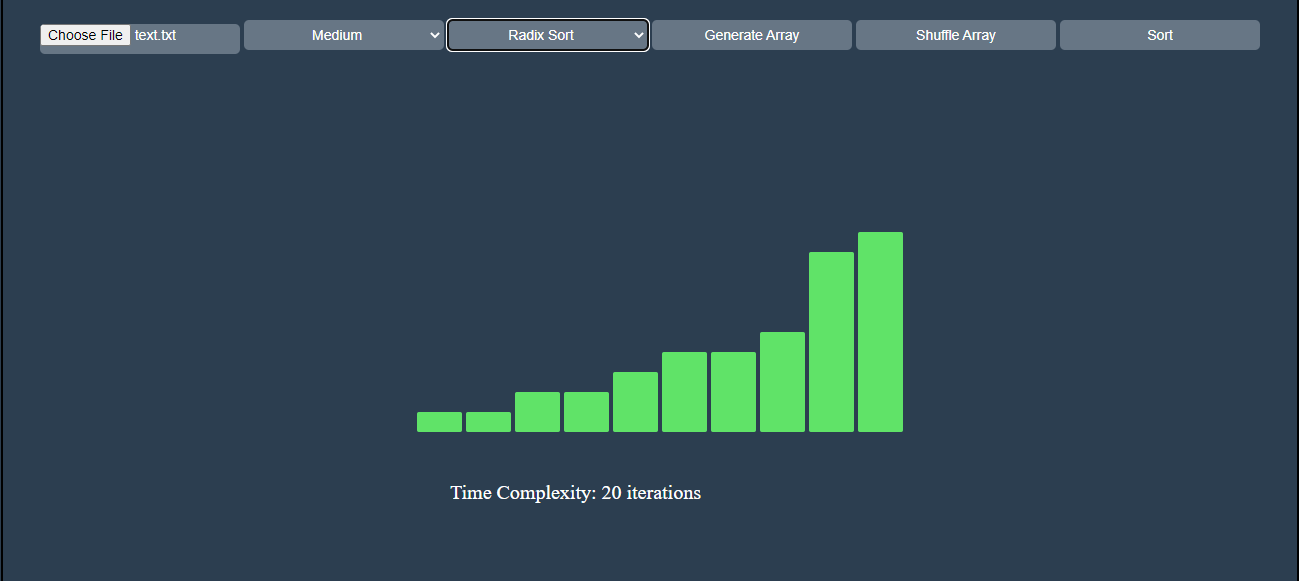
* **Merge Sort**



* **Quick Sort**
* **Count Sort**



* **Radix Sort**

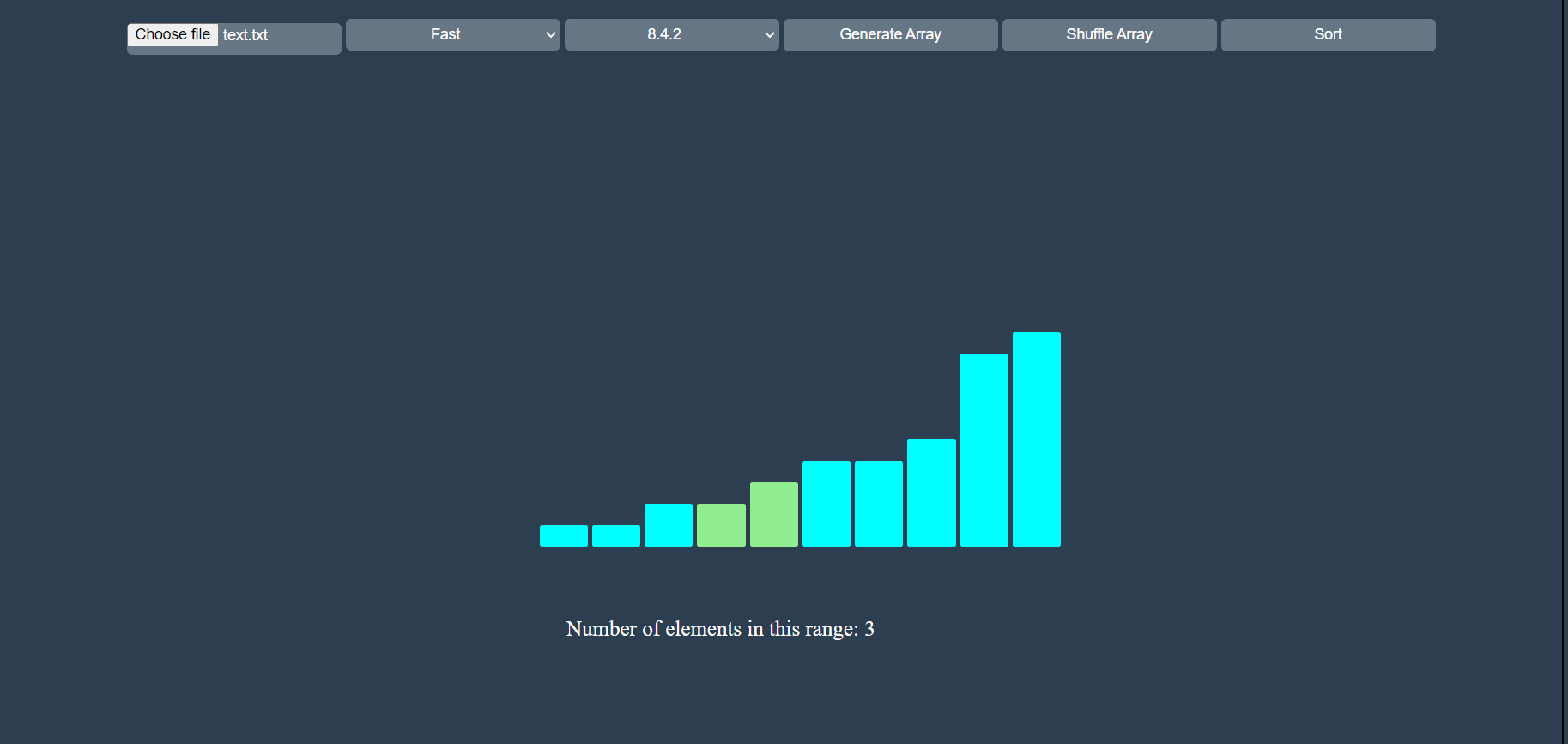


* **Bucket Sort**
* **7.4.5**



* **8.4.2**

**Range – 3 to 4**



1. **Analysis**

Algorithms with the highest time of complexity O () took greater number of iterations to sort the data. These algorithms include Bubble sort and Insertion Sort. Comparatively, algorithms such as Merge Sort and Heap Sort having time complexity of O ( performed better. However, linear time sorting techniques such as Count Sort was the quickest to sort the data set.

1. **Evaluation**

While working on a large data set, an optimized algorithm can increase system computation time and throughput significantly. If there is less emphasis on memory space bound, then linear time algorithms such as Count Sort may work best for sorting large data. However, this is not always possible therefore next best time complexity algorithms such as Quick Sort or Heap Sort might be more ideal in place algorithms than can sort large amount of data quickly.

1. **Conclusion**

Linear time algorithms (O (n)) work the quickest in terms of time, but are costly in terms of space. O ( are optimized in terms of space and time and are most likely to be used frequently. O () should be avoided wherever possible as they consume big proportion of system resources and computation cycles.

1. **References**

* geeksforgeeks
* ww3school
* sortingalgorithms