# **Design and Analysis of Algorithm**

# **Fall 2022**

# **PROJECT REPORT**

Abbas Doosa (20k-1660) (BSE-5A)

Abbas Zaidi (20k-1628) (BSE-5B)

## **ABSTRACT**

The problem is to visualize the sorting algorithms. The goal is to design a software application that visualizes all the Ten sorting algorithms on an input file and eventually generate the steps that lead to the sorted output.

## **INTRODUCTION**

An algorithm for sorting is one that arranges the items on a list. The most often used ordering systems are lexicographical and numerical, both in ascending and decreasing order. The effectiveness of other algorithms that require input data to be in sorted lists must be maximized, hence efficient sorting is crucial. Additionally, canonicalizing data and creating output that is intelligible by humans also benefit from sorting.

**Complexity of Sorting Algorithms**

The efficiency of any sorting algorithm is determined by the time complexity and space complexity of the algorithm.

1. Time Complexity: The amount of time it takes an algorithm to execute in relation to the size of the input is referred to as time complexity. It can be portrayed in a variety of ways:

Big-O notation (O)

Omega notation (Ω)

Theta notation (Θ)

2. Space Complexity: When referring to an algorithm's overall memory usage during execution, we use the term "space complexity." Both the input and the auxiliary memory are included.

The additional memory that the algorithm uses in addition to the input data is known as the auxiliary memory. Usually, while calculating an algorithm's space complexity, auxiliary memory is considered.

## **Program Design**

This Project is made on JavaScript with CSS used for the output and designing of the visualization. The description of this system is that it starts with a user-interface that allows user to select the input file that they want to sort after that the algorithm converts that file into an array to generate the bar for sorting once the bars are generated then the user select the algorithm and the speed with which the sorting would occur finally visualization of sorting can be seen with each step.

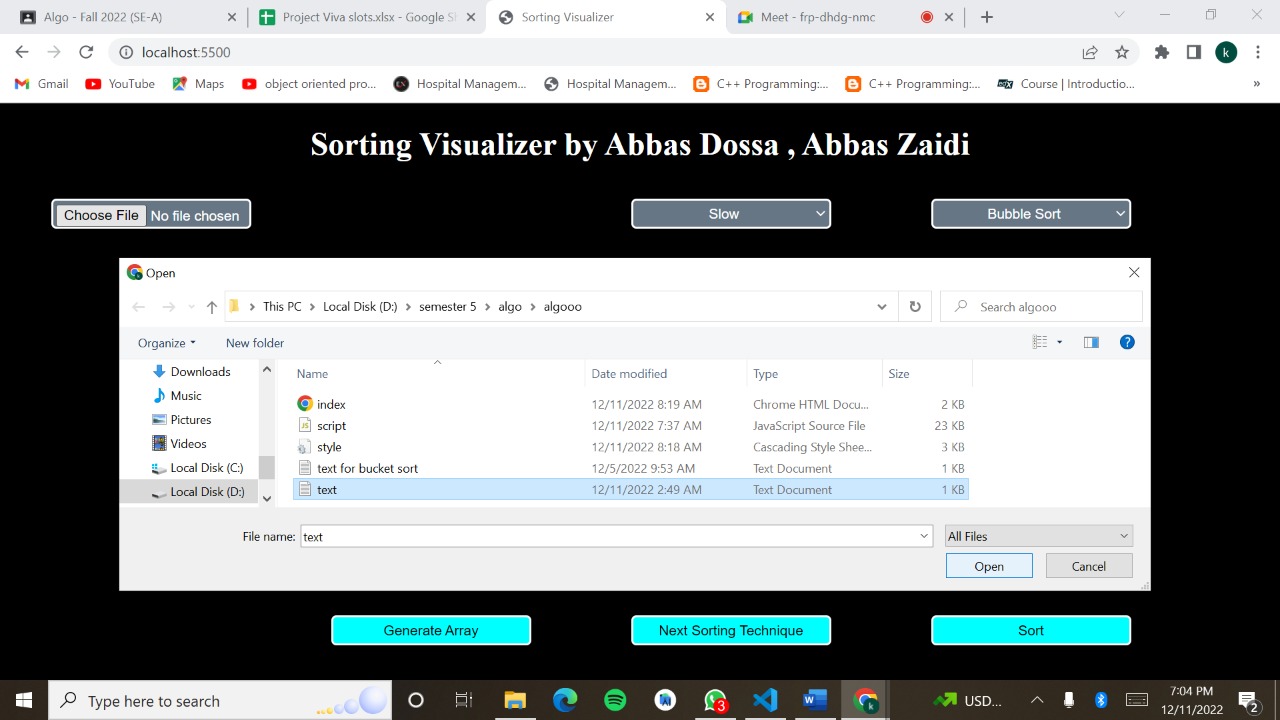
## **EXPERIMENTAL SETUP**

**Step 1: User Interface opens where the user has Option to Select the text file containing numbers**

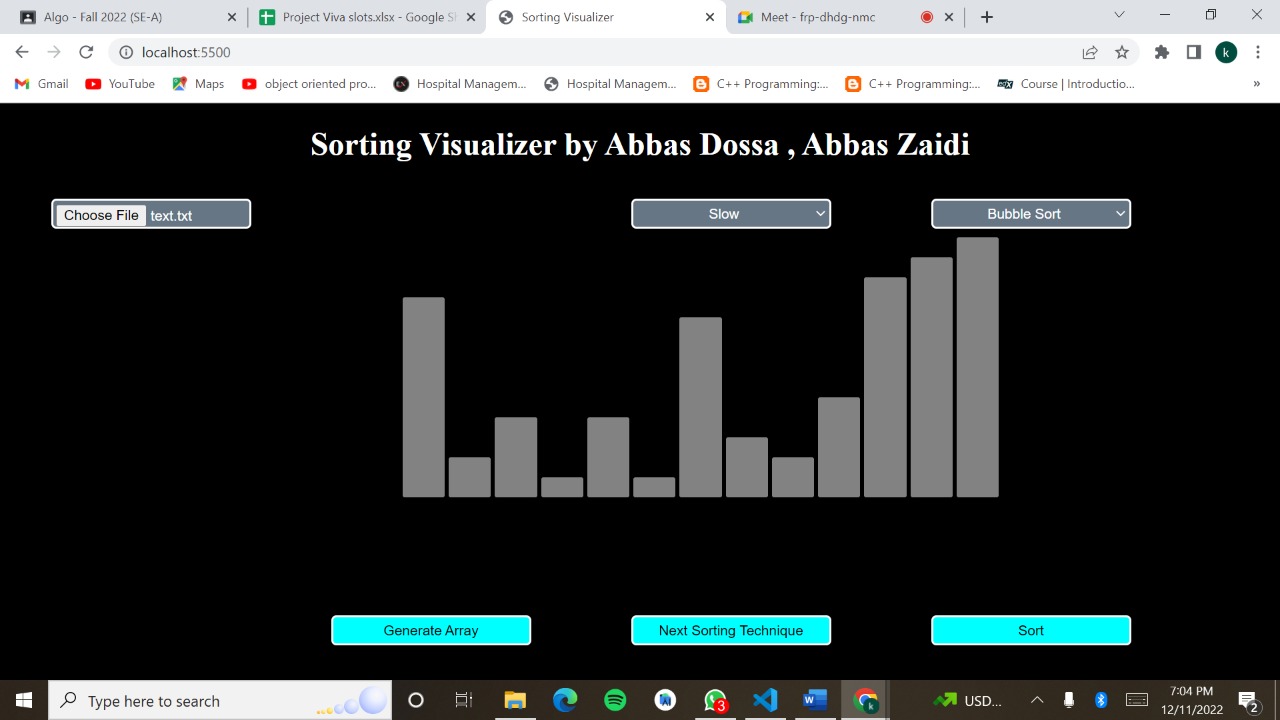
A screenshot of a computer

Description automatically generated with medium confidence

**Step 2: User selects the text file that contains the numbers to be sorted**

****

**Step 3: User generates the Array**



**Step 4: User selects the speed either Fast or Slow**

A screenshot of a computer

Description automatically generated with medium confidence

**Step 5: User Select the Sorting Algorithm out of the 10 Possibilities**

A screenshot of a computer

Description automatically generated with medium confidence

**Step 6: Click on Sort to Perform sorting and finally the sorted array is displayed along with the time and space complexities.**

A screenshot of a computer

Description automatically generated with medium confidence

## **Results:**

**All the Sorting Technique will result in the Same output which can be scene in the following output**

A screenshot of a computer

Description automatically generated with medium confidence

**Only the Bucket sort will have a different output as numbers can only be in the range of 0-1 there will have a different input file below attached is the final output of the bucket sort**



## **CONCLUSION**

To get a Sorted result we have multiple sorting algorithms to select from

but the most appropriate is the one that saves both the time and the resources of computation. Quick and merge sort are the quickest comparison-based sorting algorithms although non-comparison-based algorithms have linear time complexities but could result in huge time complexities which is not ideal.

## **REFERENCES**

* GeeksForGeeks
* TutorialsPoint