**PROJECT REPORT**

**20K-0278 | MUHAMMAD AYYAN TAHIR | BCS-5A**

**20K-0275 | SYED ABDULLAH | BCS-5A**

**ABSTRACT:**

The motivation behind this project was to show the effectiveness of animated sorting algorithms for teaching. To illustrate this, we decided to make a web-based animation application, implementing several sorting algorithms. A bar graph is used to visualize data, after which a data sorting and algorithm can be applied. The final animation is then carried out showing the step-by-step representation of each element present in the dataset**.**

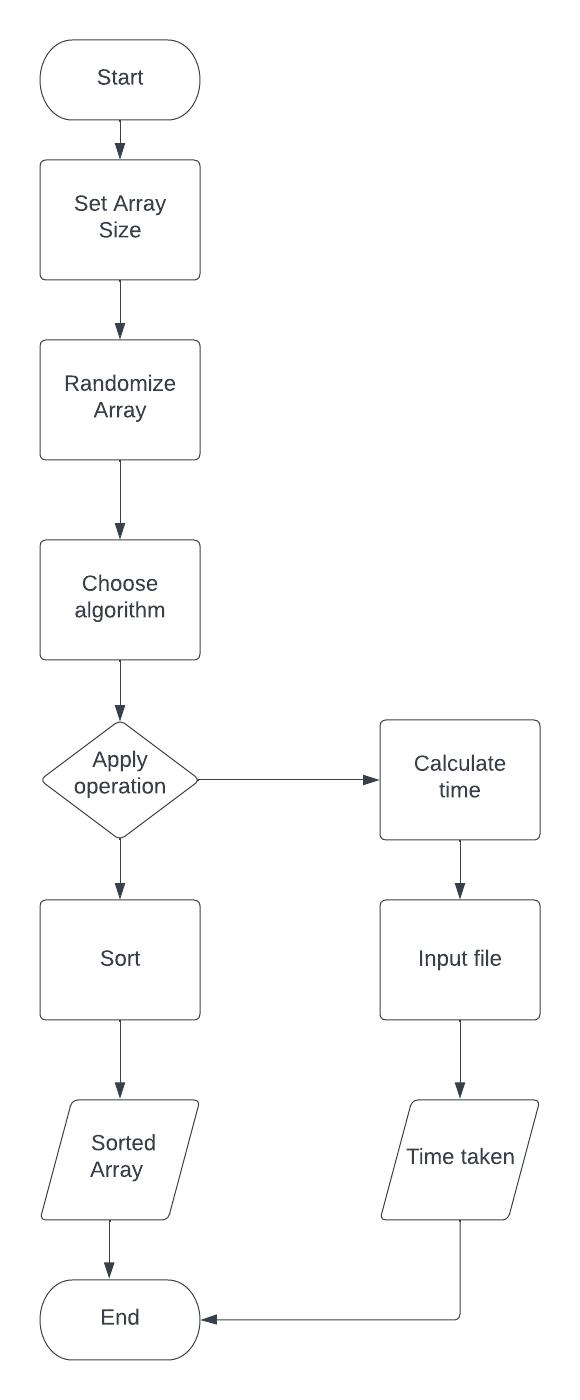
**INTRODUCTION:**

Sorting algorithms are widely used in computer software nowadays. For example, if you open file explorer on your PC, you may notice that files are sorted in various ways. Throughout their studies in computer science, students must learn various sorting algorithms. Because many students face sorting problems during the course of algorithm design, there is an understanding that visual representation is an important part of the learning process. The project's main goal was to create a program that would serve as a tool for understanding how most well-known sorting algorithms work. The demonstration software is designed to be user-friendly and simple to use. To get the most out of learning, try each sorting algorithm on your data.

**PROGRAMMING DESIGN:**

Front-End: HTML,CSS  
Back-End : Javascript  
IDE : VS Code

A step by step diagram of operations taking place in the system

  
**EXPERIMENTAL SETUP:**

This web application was designed in such a way that the user can easily interact with it and arrive at their desired assertion, which is usually constructed on the basis of execution time . For visual representation of step by step sorting , the elements of the array are denoted by the length of the bars and the visualisation can be viewed by the user at their own suitable pace by adjusting the sorting speed from the slider. Because visualization speed is a significant factor in time elapsed , the backend code divides the total time taken by the speed factor and outputs it to Web page for the user to view.

Below is the list of the algorithms we have executed:

* **Bubble Sort:**

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

Time Complexity: O(n2)

Space Complexity: O(1)

* **Insertion Sort:**

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Time Complexity: O(n2)

Space Complexity: O(1)

* **Heap Sort:**

Heap sort is a comparison-based sorting technique based on Binary Heap data structure. It is similar to the selection sort where we first find the minimum element and place the minimum element at the beginning. Repeat the same process for the remaining elements.

Time Complexity: O(nlogn)

Space Complexity: O(1)

* **Merge Sort:**

The Merge Sort algorithm is a sorting algorithm that is based on the Divide and Conquer paradigm. In this algorithm, the array is initially divided into two equal halves and then they are combined in a sorted manner.

Time Complexity: O(n log(n))

Space Complexity: O(n)

* **Quick Sort:**

QuickSort, like Merge Sort, is a Divide and Conquer algorithm. It selects an element to act as a pivot, partitions the given array around the pivot, and then sorts the array so that the pivot is in the correct position at each iteration.

Time Complexity: O(n log(n))

Space Complexity: O(n log(n))

* **Counting Sort:**

Counting sort is a sorting technique based on keys between a specific range. It works by counting the number of objects having distinct key values (a kind of hashing). Then do some arithmetic operations to calculate the position of each object in the output sequence.

Time Complexity: O(n+k)

Space Complexity: O(n+k)

* **Radix Sort:**

In subroutines when counting sort cannot be utilised alone, such as when an integer array's values span from 0 to 100000, radix sort sorts arrays based on the modulus increasing by a power of 10 in each iteration.

Time Complexity: O(n\*k/d)

Space Complexity: O(n+2d)

* **Bucket Sort:**

Bucket Sort is a sorting algorithm that divides the unsorted array elements into several groups called buckets. Each bucket is then sorted by using any of the suitable sorting algorithms or recursively applying the same bucket algorithm.Finally, the sorted buckets are combined to form a final sorted array.

Time Complexity: O(n+k)

Space Complexity: O(n+k)

* **Modified Quick Sort (7.4.5):**

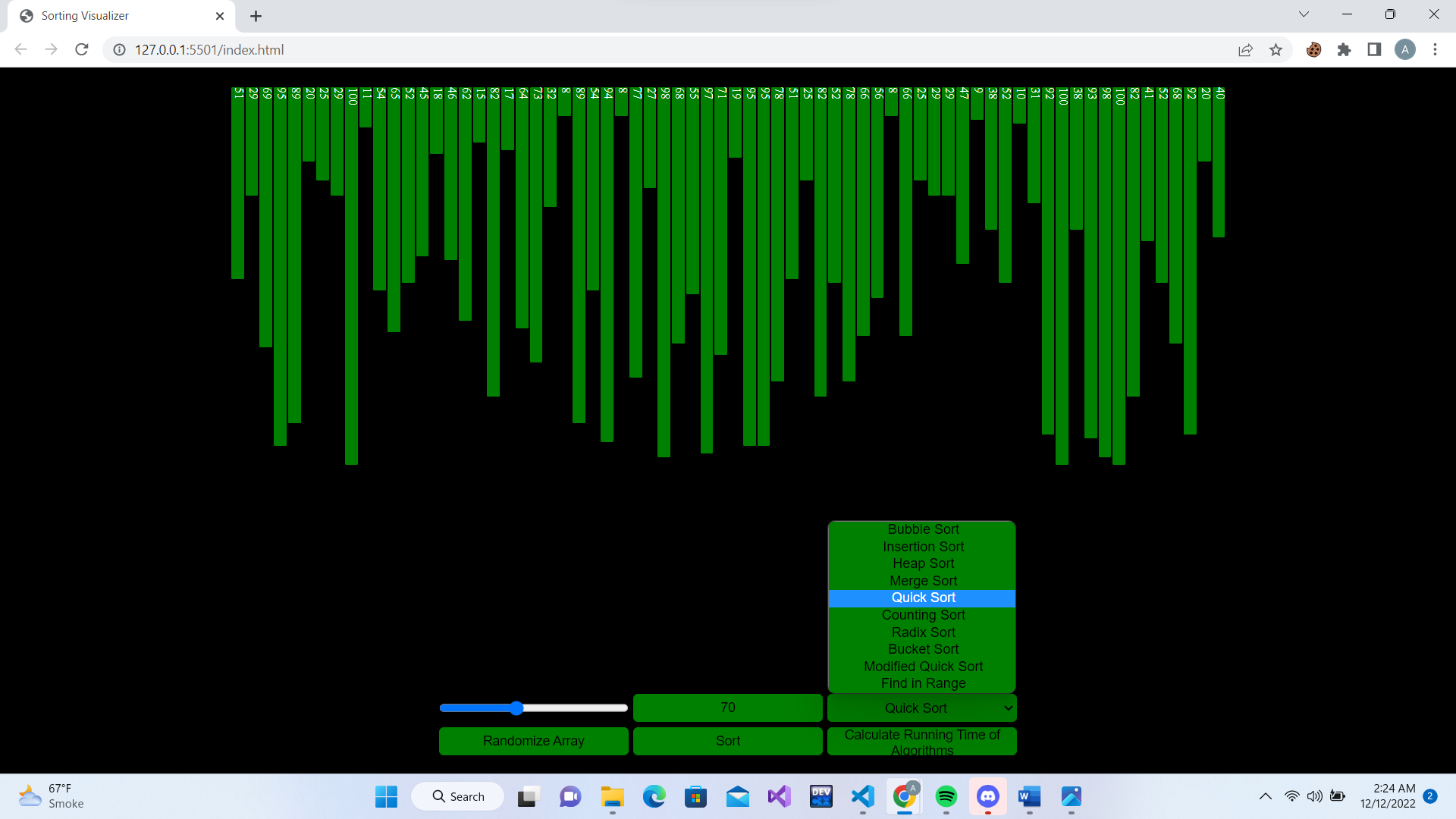
This algorithm is basically a slight modification of the conventional quicksort algorithm . In quicksort the data is broken down into partitions but here at one time the partition limit is reached and from then onwards Insertion sort subroutine is applied.  
Time Complexity : O.nk C n lg.n=k

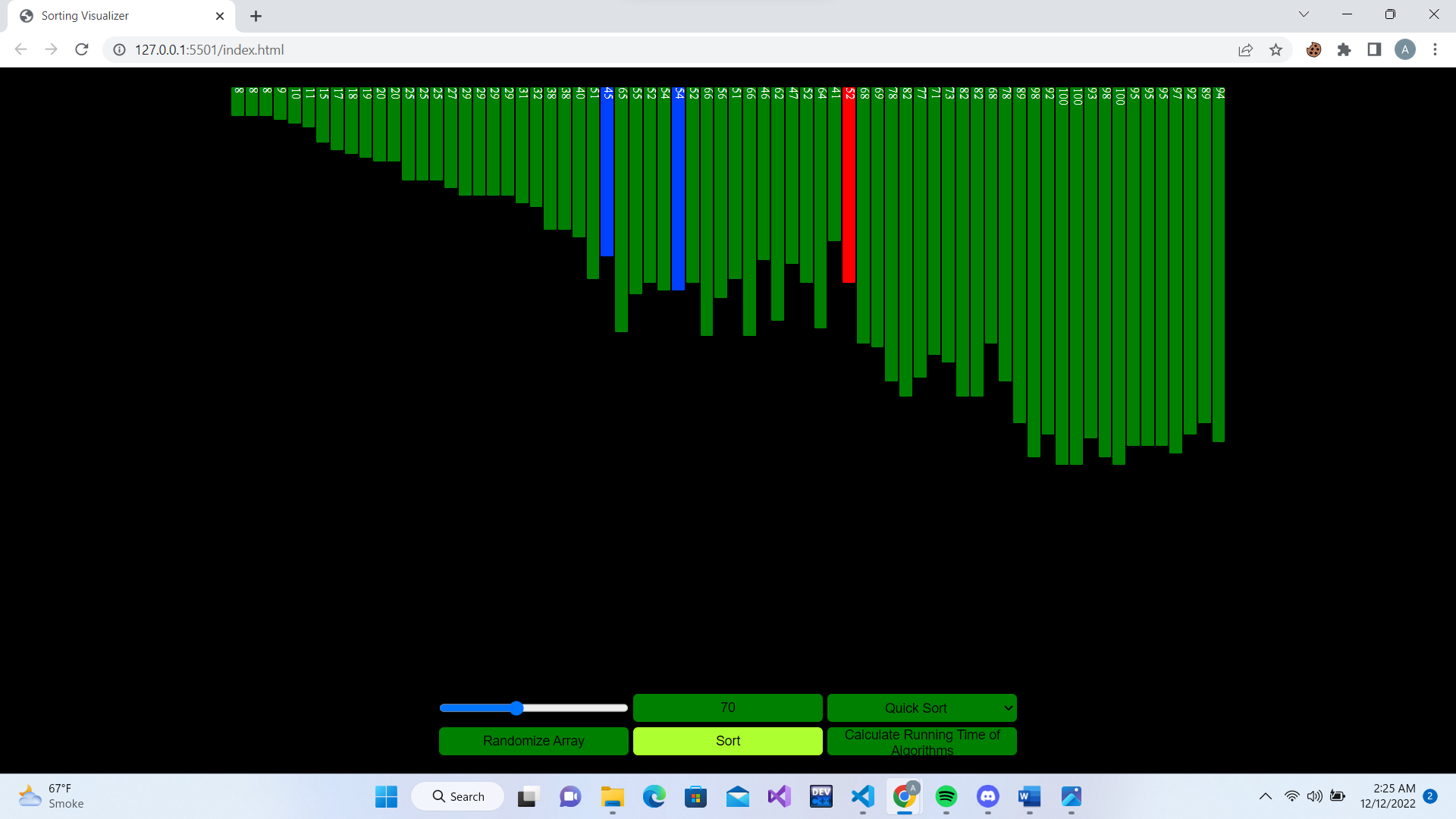
* **Find In Range(8.2.4):**

This algorithm takes an input of two numbers from the user which is the range and then compares the value of the data in array according to the given range and then stores their count at a specific index. This algorithm basically works like a pre processing of counting sort algorithm except the sorting part.

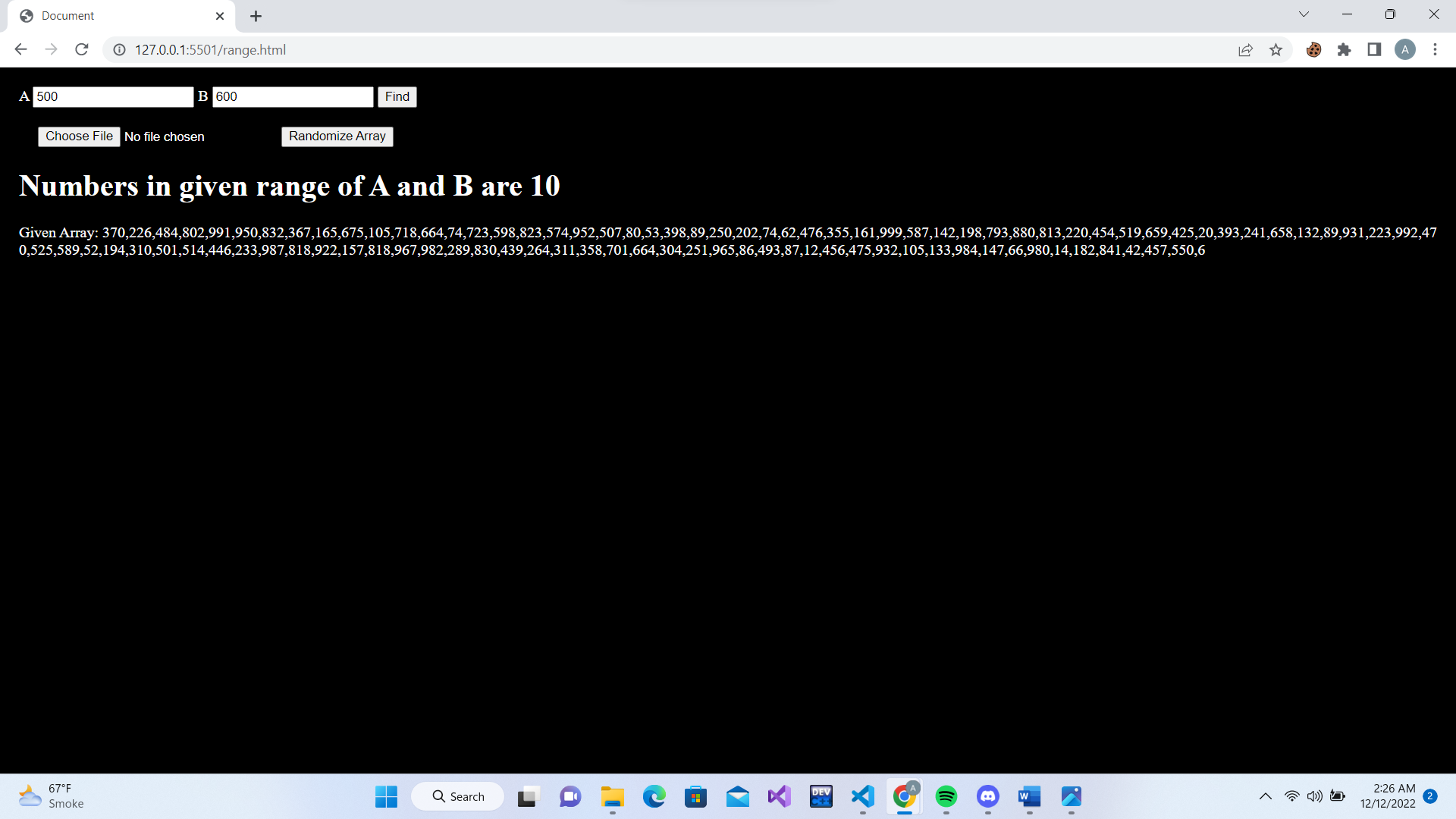
Time Complexity : O(1)

**RESULTS:**

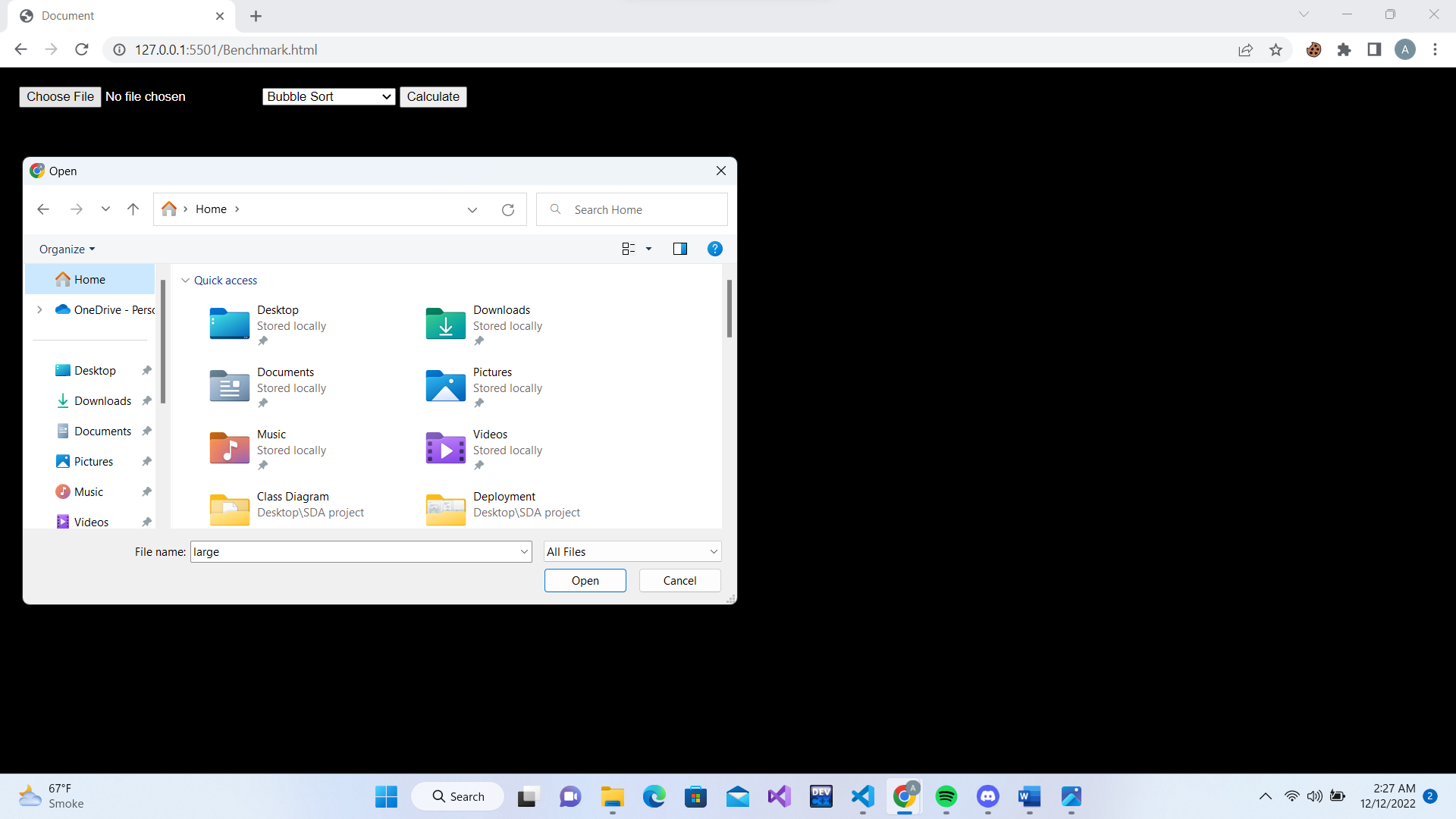
****

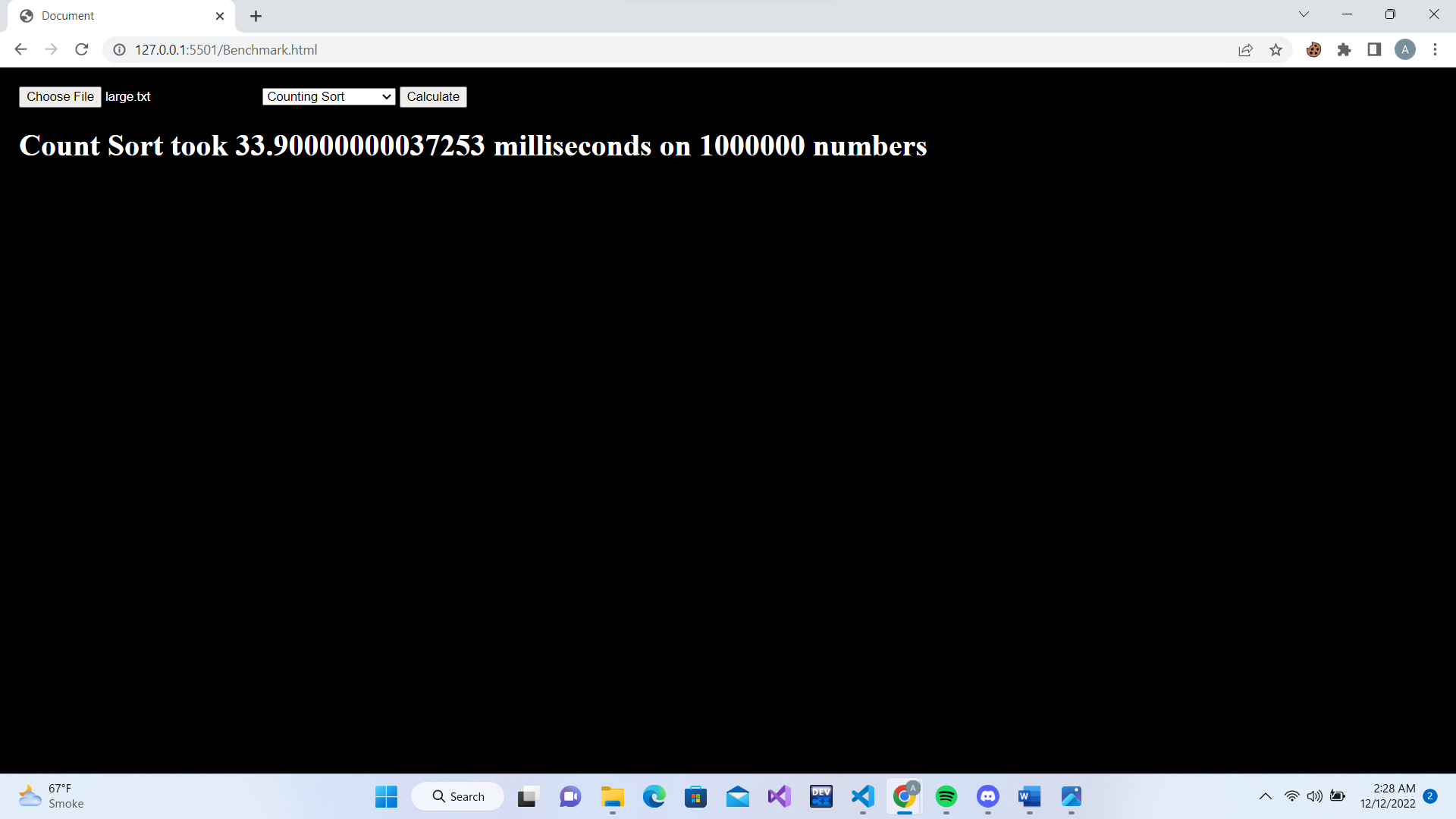
****

Step by Step visualization of your selected algorithm



Finds the numbers present in the given range in O(1) time





Takes File as an input and calculates execution time

**CONCLUSION:**

The product makes the understanding of algorithms easier through the step by step visualization and is aimed at the educational institutions and the self-learning computer science students who can use this tool in order to study different sorting algorithms comprehensively in their relevant courses.

**REFERENCES:**

<https://www.geeksforgeeks.org/sorting-algorithms/>