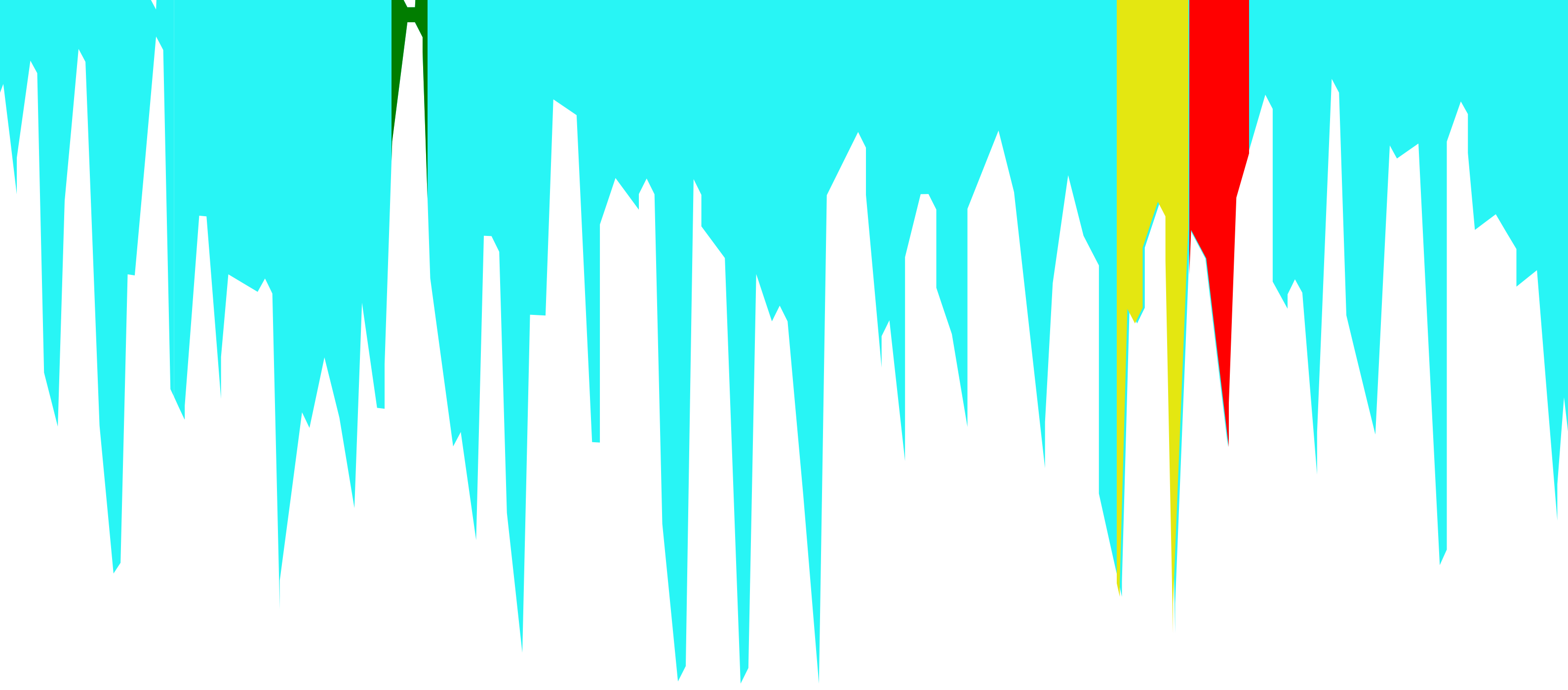


SORTING VISUALIZER

Aditya Rajesh Nair

TyBsc:CS, Roll No: 01

'A' GRADE
NAAC Re-accredited

NATIONAL EDUCATION SOCIETY'S

Ratnam College of Arts, Science & Commerce
Bhandup, Mumbai – 400 078.

Department of Computer Science

CERTIFICATE

Class: *Roll / Seat No.:* *Year:*

This is to certify that Mr./Ms. ____________________________ has satisfactorily completed the project work ______________________for partial fulfillment of the 3 years Degree Course Bachelor in Computer Science for the University of Mumbai for the year _________ to _________.

Place: ________________

Date: ________________

Project Guide

In-charge,
Dept. of Computer Science

Signature of Examiner with Date

Table of Contents

Sr.No	Topic	Page No	Signature
1	Introduction	02	
2	Languages Used	03	
3	Explaining Algorithms	04-06	
4	Requirement	07	
5	Code	07-27	
6	Advantage Of Sorting Visualizer	28	
7	Screen Shots	29-30	
8	References	31	

Table of Contents

CONTENTS

01.
Introduction

02.
Languages Used

03.
Explaining Algorithms

04.
Requirement

05.
Code

06.
Advantage Of Sorting Visualizer

07.
Screen Shots

08.
References

Introduction:

Sorting Visualizer helps the user to understand and visualize the Sorting Algorithms using Bar Graph.

There are currently hundreds of different sorting algorithms, each with its own specific characteristics. Algorithms are classified according to two metrics: space complexity and time complexity. Space and time complexity can also be further subdivided into 3 different cases: Best Case, Average Case and Worst Case.

Sorting algorithms can be difficult to understand and it's easy to get confused. I believe visualizing sorting algorithms can be a great way to better understand their functioning while having fun!

Languages Used:

HTML: HTML is the standard markup language for documents designed to be displayed in a web browser.

CSS: CSS is a style sheet language used for describing the presentation of a document written in a markup language.

JavaScript: JavaScript, often abbreviated to JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.

Explaining Sorting Algorithms:

Bubble Sort Algorithm: 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.

How Bubble Sort Works?

Consider an array arr[] = {5, 1, 4, 2, 8}

Bubble sort starts with very first two elements, comparing them to check which one is greater.

(5 1 4 2 8) –› (1 5 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 › 1.

(1 5 4 2 8) –› (1 4 5 2 8), Swap since 5 › 4

(1 4 5 2 8) –› (1 4 2 5 8), Swap since 5 › 2

(1 4 2 5 8) –› (1 4 2 5 8), Now, since these elements are already in order (8 › 5), algorithm does not swap them.

Selection Sort Algorithm: The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

- The subarray which is already sorted.
- Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

How Selection Sort Works?

Lets consider the following array as an example: arr[] = {64, 25, 12, 22, 11}

For the first position in the sorted array, the whole array is traversed from index 0 to 4 sequentially. The first position where 64 is stored presently, after traversing whole array it is clear that 11 is the lowest value.

64 25 12 22 11

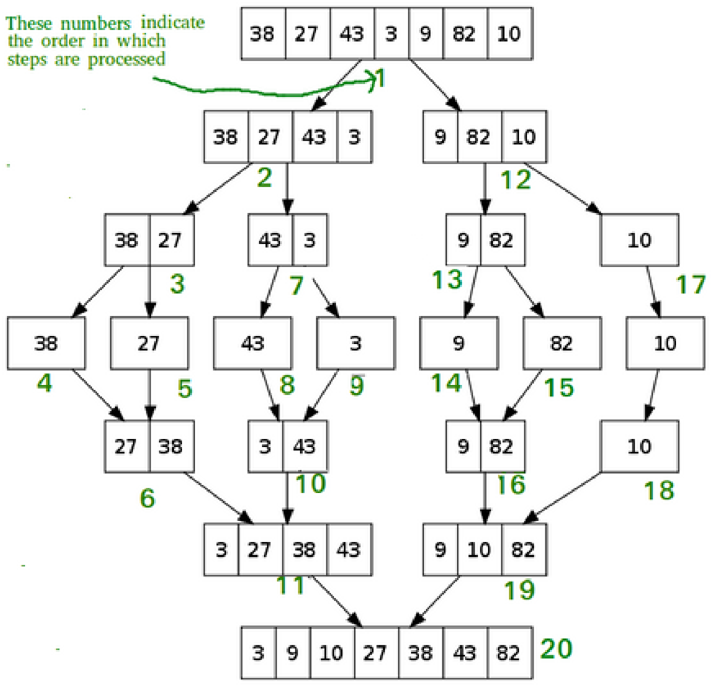
Thus, replace 64 with 11. After one iteration 11, which happens to be the least value in the array, tends to appear in the first position of the sorted list.

11 25 12 22 64

Merge Sort Algorithm: The Merge Sort algorithm is a sorting algorithm that is considered as an example of the divide and conquer strategy. So, in this algorithm, the array is initially divided into two equal halves and then they are combined in a sorted manner. We can think of it as a recursive algorithm that continuously splits the array in half until it cannot be further divided. This means that if the array becomes empty or has only one element left, the dividing will stop, i.e. it is the base case to stop the recursion. If the array has multiple elements, we split the array into halves and recursively invoke the merge sort on each of the halves. Finally, when both the halves are sorted, the merge operation is applied. Merge operation is the process of taking two smaller sorted arrays and combining them to eventually make a larger one.

How Merge Sort Works?

for an example array {38, 27, 43, 3, 9, 82, 10}



REQUIREMENTS:

Microsoft Visual Studio with all framework downloaded

CODE:

Bubble Sort:

```
async function bubble() {
    console.log('In bubbe()');
    const ele = document.querySelectorAll(".bar");
    for(let i = 0; i < ele.length-1; i++){
        console.log('In ith loop');
        for(let j = 0; j < ele.length-i-1; j++){
            console.log('In jth loop');
            ele[j].style.background = 'blue';
            ele[j+1].style.background = 'blue';
            if(parseInt(ele[j].style.height) > parseInt(ele[j+1].style.height)){
                console.log('In if condition');
                await waitforme(delay);
                swap(ele[j], ele[j+1]);
            }
            ele[j].style.background = 'cyan';
            ele[j+1].style.background = 'cyan';
        }
        ele[ele.length-1-i].style.background = 'green';
    }
ele[0].style.background = 'green';
}

const bubSortbtn = document.querySelector(".bubbleSort");
bubSortbtn.addEventListener('click', async function(){
    disableSortingBtn();
    disableSizeSlider();
    disableNewArrayBtn();
    await bubble();
    enableSortingBtn();
    enableSizeSlider();
    enableNewArrayBtn();
});
```

Selection Sort:

```
async function selection(){
    console.log('In selection()');
    const ele = document.querySelectorAll(".bar");
    for(let i = 0; i < ele.length; i++){
        console.log('In ith loop');
        let min_index = i;
        // Change color of the position to swap with the next min
        ele[i].style.background = 'blue';
        for(let j = i+1; j < ele.length; j++){
            console.log('In jth loop');
            // Change color for the current comparision (in consideration for
min_index)
            ele[j].style.background = 'red';

            await waitforme(delay);
            if(parseInt(ele[j].style.height) < parseInt(ele[min_index].style.height)){
                console.log('In if condition height comparision');
                if(min_index !== i){
                    // new min_index is found so change prev min_index color back
to normal
ele[min_index].style.background = 'cyan';
}
min_index = j;
}
else{
// if the currnent comparision is more than min_index change is back to
normal
ele[j].style.background = 'cyan';
}
}
await waitforme(delay);
swap(ele[min_index], ele[i]);
// change the min element index back to normal as it is swapped
ele[min_index].style.background = 'cyan';
// change the sorted elements color to green
ele[i].style.background = 'green';
}
}
```

```
const selectionSortbtn = document.querySelector(".selectionSort");
selectionSortbtn.addEventListener('click', async function(){
 disableSortingBtn();
 disableSizeSlider();
 disableNewArrayBtn();
 await selection();
 enableSortingBtn();
 enableSizeSlider();
 enableNewArrayBtn();
});
```

Merge Sort:

```
//let delay = 30;
async function merge(ele, low, mid, high){
    console.log('In merge()');
    console.log(`low=${low}, mid=${mid}, high=${high}`);
    const n1 = mid - low + 1;
    const n2 = high - mid;
    console.log(`n1=${n1}, n2=${n2}`);
    let left = new Array(n1);
    let right = new Array(n2);

    for(let i = 0; i < n1; i++){
        await waitforme(delay);
        console.log('In merge left loop');
        console.log(ele[low + i].style.height + ' at ' + (low+i));
        // color
        ele[low + i].style.background = 'orange';
        left[i] = ele[low + i].style.height;
    }
    for(let i = 0; i < n2; i++){
        await waitforme(delay);
        console.log('In merge right loop');
console.log(ele[mid + 1 + i].style.height + ' at ' + (mid+1+i));
        // color
        ele[mid + 1 + i].style.background = 'yellow';
        right[i] = ele[mid + 1 + i].style.height;
    }
    await waitforme(delay);
    let i = 0, j = 0, k = low;
    while(i < n1 && j < n2){
        await waitforme(delay);
        console.log('In merge while loop');
        console.log(parseInt(left[i]), parseInt(right[j]));

        // To add color for which two r being compared for merging

        if(parseInt(left[i]) <= parseInt(right[j])){
            console.log('In merge while loop if');
            // color
            if((n1 + n2) === ele.length){
                ele[k].style.background = 'green';
            }
            else{
                ele[k].style.background = 'lightgreen';
}
await waitforme(delay);
    let i = 0, j = 0, k = low;
    while(i < n1 && j < n2){


            ele[k].style.height = left[i];
            i++;
            k++;
        }
        else{
            console.log('In merge while loop else');
            // color
            if((n1 + n2) === ele.length){
                ele[k].style.background = 'green';
            }
            else{
                ele[k].style.background = 'lightgreen';
            }
            ele[k].style.height = right[j];
            j++;
            k++;
        }
    }
```

```
while(i < n1){
        await waitforme(delay);
        console.log("In while if n1 is left");
        // color
        if((n1 + n2) === ele.length){
            ele[k].style.background = 'green';
        }
        else{
            ele[k].style.background = 'lightgreen';
        }
        ele[k].style.height = left[i];
        i++;
        k++;
    }
    while(j < n2){
        await waitforme(delay);
        console.log("In while if n2 is left");
        // color
        if((n1 + n2) === ele.length){
            ele[k].style.background = 'green';
        }
        else{
            ele[k].style.background = 'lightgreen';
```

```
 }
 ele[k].style.height = right[j];
 j++;
 k++;
 }
}
async function mergeSort(ele, l, r){
 console.log('In mergeSort()');
 if(l >= r){
 console.log(`return cause just 1 elemment l=${l}, r=${r}`);
 return;
 }
 const m = l + Math.floor((r - l) / 2);
 console.log(`left=${l} mid=${m} right=${r}`, typeof(m));
 await mergeSort(ele, l, m);
 await mergeSort(ele, m + 1, r);
 await merge(ele, l, m, r);
}
const mergeSortbtn = document.querySelector(".mergeSort");
mergeSortbtn.addEventListener('click', async function(){
 let ele = document.querySelectorAll('.bar');
let l = 0;
 let r = parseInt(ele.length) - 1;
 disableSortingBtn();
 disableSizeSlider();
 disableNewArrayBtn();
 await mergeSort(ele, l, r);
 enableSortingBtn();
 enableSizeSlider();
 enableNewArrayBtn();
});
```

Sorting:

```
// swap function util for sorting algorithms takes input of 2 DOM elements with .style.height
feature
function swap(el1, el2) {
    console.log('In swap()');

    let temp = el1.style.height;
    el1.style.height = el2.style.height;
    el2.style.height = temp;

}

// Disables sorting buttons used in conjunction with enable, so that we can disable during
sorting and enable buttons after it
function disableSortingBtn(){
    document.querySelector(".bubbleSort").disabled = true;
    document.querySelector(".mergeSort").disabled = true;
    document.querySelector(".selectionSort").disabled = true;
}

// Enables sorting buttons used in conjunction with disable
function enableSortingBtn(){
    document.querySelector(".bubbleSort").disabled = false;
    document.querySelector(".mergeSort").disabled = false;
    document.querySelector(".selectionSort").disabled = false;
}
```

```
// Disables size slider used in conjunction with enable, so that we can disable during sorting
and enable buttons after it
function disableSizeSlider(){
    document.querySelector("#arr_sz").disabled = true;
}

// Enables size slider used in conjunction with disable
function enableSizeSlider(){
    document.querySelector("#arr_sz").disabled = false;
}

// Disables newArray buttons used in conjunction with enable, so that we can disable during
sorting and enable buttons after it
function disableNewArrayBtn(){
    document.querySelector(".newArray").disabled = true;
}

// Enables newArray buttons used in conjunction with disable
function enableNewArrayBtn(){
    document.querySelector(".newArray").disabled = false;
}

// Used in async function so that we can so animations of sorting, takes input time in ms (1000
= 1s)
```

```
 return new Promise(resolve => {
        setTimeout(() => { resolve('') }, milisec);
    })
}

// Selecting size slider from DOM
let arraySize = document.querySelector('#arr_sz');

// Event listener to update the bars on the UI
arraySize.addEventListener('input', function(){
    console.log(arraySize.value, typeof(arraySize.value));
    createNewArray(parseInt(arraySize.value));
});

// Default input for waitforme function (260ms)
let delay = 260;

// Selecting speed slider from DOM
let delayElement = document.querySelector('#speed_input');

// Event listener to update delay time
delayElement.addEventListener('input', function(){
    console.log(delayElement.value, typeof(delayElement.value));
    delay = 320 - parseInt(delayElement.value);
});
```

```
// Creating array to store randomly generated numbers
let array = [];

// Call to display bars right when you visit the site
createNewArray();

// To create new array input size of array
function createNewArray(noOfBars = 60) {
    // calling helper function to delete old bars from dom
    deleteChild();

    // creating an array of random numbers
    array = [];
    for (let i = 0; i < noOfBars; i++) {
        array.push(Math.floor(Math.random() * 250) + 1);
    }
    console.log(array);

    // select the div #bars element
    const bars = document.querySelector("#bars");

    // create multiple element div using loop and adding class 'bar col'
    for (let i = 0; i < noOfBars; i++) {
        const bar = document.createElement("div");
        bar.style.height = `${array[i]*2}px`;
bar.classList.add('bar');
        bar.classList.add('flex-item');
        bar.classList.add(`barNo${i}`);
        bars.appendChild(bar);
    }
}

// Helper function to delete all the previous bars so that new can be added
function deleteChild() {
    const bar = document.querySelector("#bars");
    bar.innerHTML = '';
}

// Selecting newarray button from DOM and adding eventlistener
const newArray = document.querySelector(".newArray");
newArray.addEventListener("click", function(){
    console.log("From newArray " + arraySize.value);
    console.log("From newArray " + delay);
    enableSortingBtn();
    enableSizeSlider();
    createNewArray(arraySize.value);
});
```

Style.css:

```
body {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 20px;
    padding: 0 20px 30px 0;
    line-height: 1.4;
}
.flex-container{
    margin-top: 20px;
    display: flex;
    flex-wrap: nowrap;
    width: 100%;
    height: 500px;
    justify-content: center;
    transition: 2s all ease;
}

.flex-item{
    background: cyan;
    border: 1pt solid black;
    width: 10px;
    transition: 0.1s all ease;
}
```

```
.row{
 display: grid;
 grid-template-columns: 1fr 1fr 2fr;
}

#input{
 display: flex;
 padding: 10px;
 justify-content: space-around;
}
```

VS.Html:

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sorting Visualizer by Aditya Rajesh Nair</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqIj61tLrc4wSX0szH/Ev+nYRRuWlolflfl"
crossorigin="anonymous">
    <link rel="stylesheet" href="style.css">
</head>
<header>
    <h1 align="center">Sorting Visualizer</h1>
    <nav>
        <div class="row">
            <div class="col gap-2 d-sm-flex" id="newArray">
                <button type="button" class="btn btn-outline-success btn-dark newArray">New
Array</button>
            </div>
            <div class="col" id="input">
                <span id="size">Size
                    <input id="arr_sz" type="range" min="10" max="100" step=1 value=60>
                </span>
```

```
 <span id="speed">Speed
 <input id="speed_input" type="range" min="60" max="300" stepDown=10 value=60>
 </span>
 </div>
 <div class="col gap-2 d-sm-flex justify-content-end">
 <button type="button" class="btn btn-outline-primary btn-dark bubbleSort">Bubble
Sort</button>
 <button type="button" class="btn btn-outline-primary btn-dark selectionSort">Selection
Sort</button>
 <button type="button" class="btn btn-outline-primary btn-dark mergeSort">Merge
Sort</button>
 </div>
 </div>
 </nav>
</header>
</foot>
<footer> <h3 align="center">A Project By Aditya Rajesh Nair</h3>
</foot>
<body class="p-3 mb-2 bg-dark text-white">

 <div id="bars" class="flex-container"></div>

 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta2/dist/js/bootstrap.bundle.min.js" integrity="sha384-
b5kHyXgcpbZJO/tY9Ul7kGkf1S0CWuKcCD38l8YkeH8z8QjE0GmW1gYU5S9FOnJ0"
```

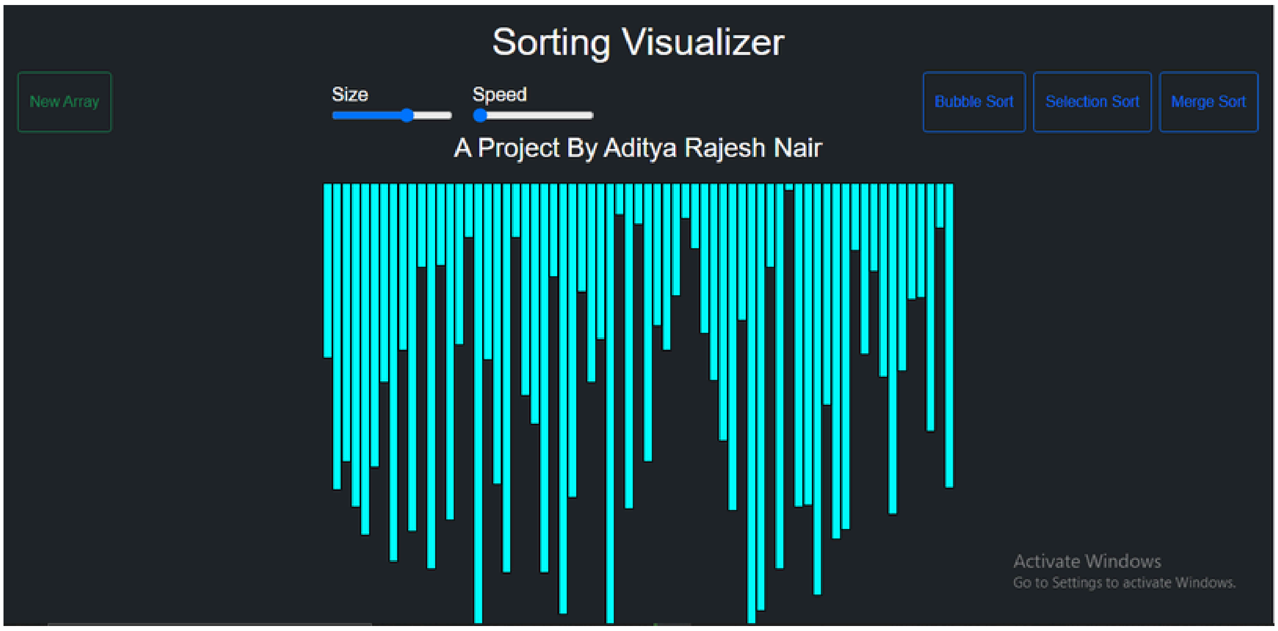
```
crossorigin="anonymous"></script>
 <script src="js_files/sorting.js"></script>
 <script src="js_files/bubble.js"></script>
 <script src="js_files/merge.js"></script>
 <script src="js_files/selection.js"></script>
</body>
</html>
```

Advantage Of Sorting Visualizer:

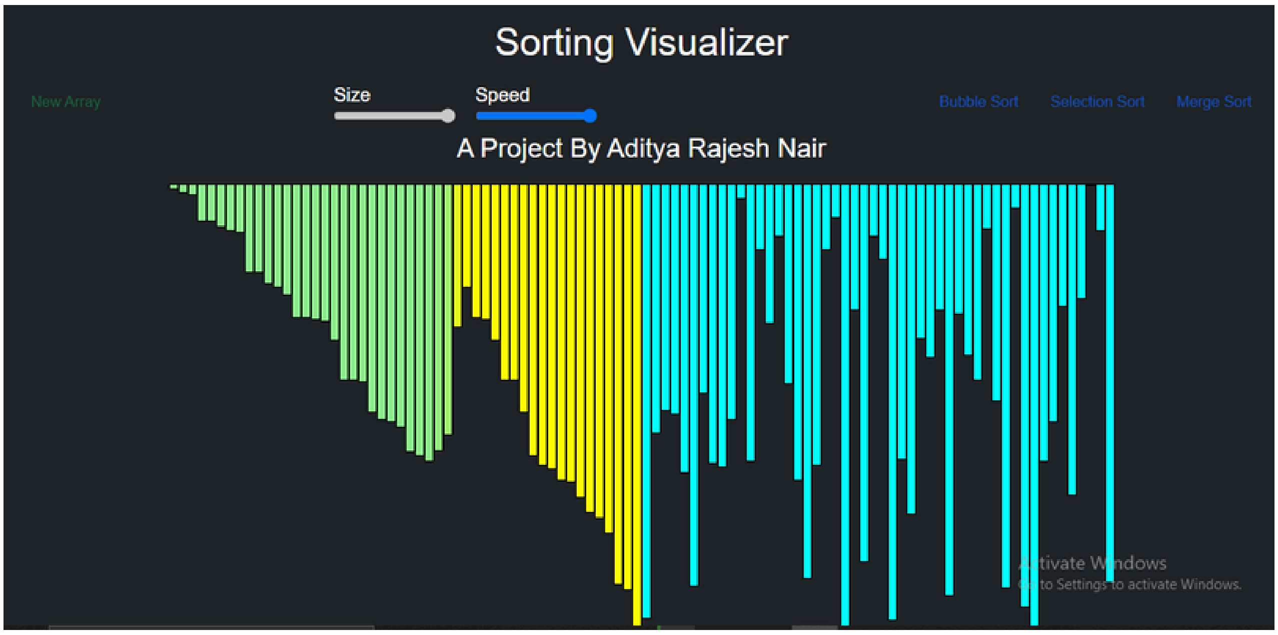
- Helps the user to understand and visualize the Sorting Algorithms.
- Can be used in Colleges to make students understand Sorting Algorithms in a Effective way.

SCREEN SHOTS:

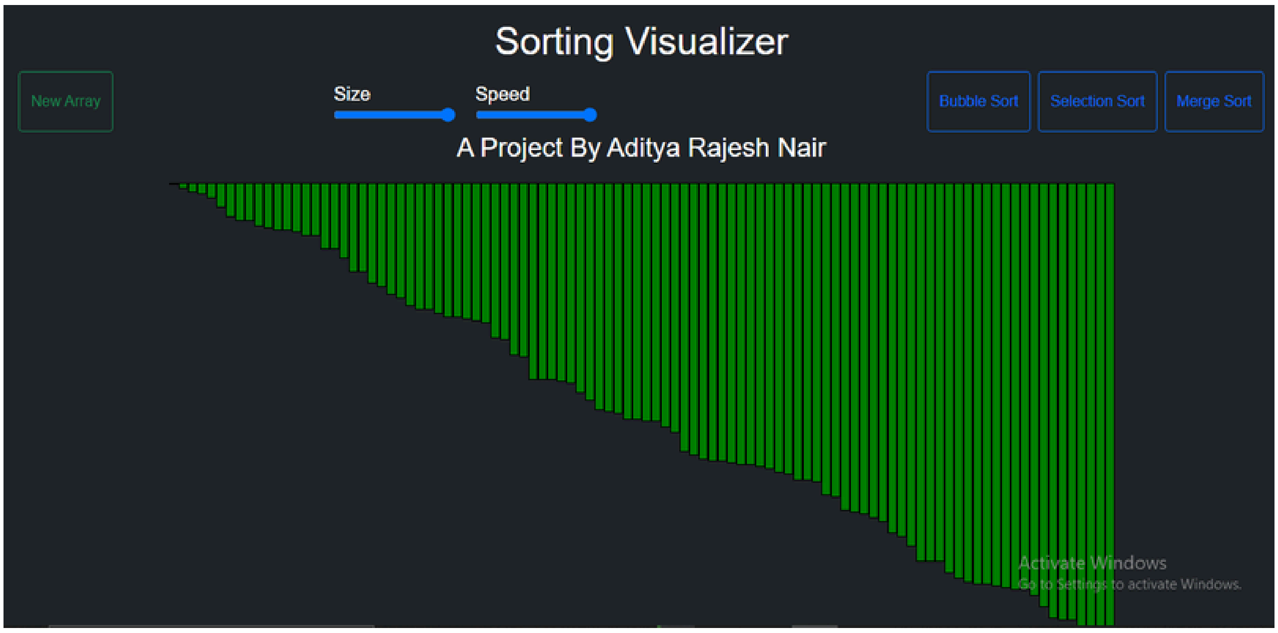
Main:



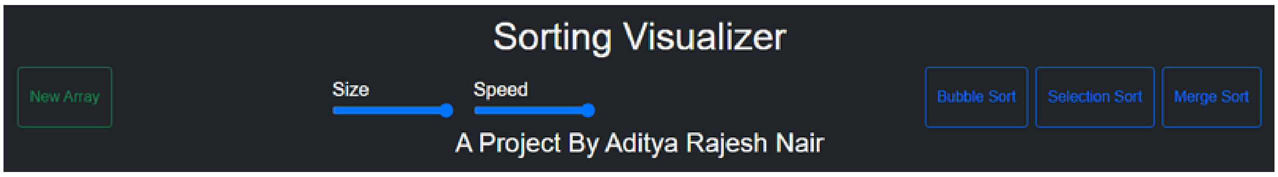
During Process:



End of Process:



Navigation Bar:



References:

- Greeks for Greeks:

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

https://www.geeksforgeeks.org/bubble-sort/?ref=lbp

https://www.geeksforgeeks.org/selection-sort/?ref=lbp

https://www.geeksforgeeks.org/merge-sort/?ref=lbp

- Youtube

https://www.youtube.com/watch?v=cW16SGqr_Lg