| | |
|---|---|
| Warsaw University of Technology | Advanced Internet Programming |
| Project 3 Sorting using JavaScript | Maciej Iwańczyk 311258 |
| Date: 08.03.2024 | Grade: |

## 1. Introduction:

In today's project I had to implement two sorting algorithms in JavaScript - Bubble-Sort and Selection-Sort. I have decided to make all three parameters of a random number the amount of numbers, the minimum number and the maximum number to be inputted by the user. This way it would be a lot faster for me to compare the results and execution times of different algorithms for different parameters of the random number generator.

## 2.HTML:

The usage of HTML in this code is very limited. I have only used some form and paragraph elements in order to input and display results.

```
95  <form onsubmit="handleSubmit(event)">
96    <label for="amount">Enter the amount of numbers to generate:</label>
97    <input type="number" id="amount" name="amount" required>
98    <br>
99    <label for="min">Enter the minimum value:</label>
100   <input type="number" id="min" name="min" required>
101   <br>
102   <label for="max">Enter the maximum value:</label>
103   <input type="number" id="max" name="max" required>
104   <br>
105   <button type="submit">Generate and Sort</button>
106 </form>
107
108 <div id="results"></div>
109 <p id="bubbleSortTime"></p>
110 <p id="selectionSortTime"></p>
```

## 3.JavaScript:

JavaScript was used in order to implement the core functionalities of the application.Functions were developed for generating random lists, performing Bubble Sort and Selection Sort, and handling form submission. Sorting algorithms were implemented efficiently to ensure optimal performance. Execution times were recorded using the performance.now() method to measure the efficiency of sorting operations. I have also implemented a very simple JavaScript in order to prevent wrong values that could potentially be inputted by the user.

```
<script>
  // Function to generate a random list of numbers
  function generateRandomList(length, min, max) {
    let list = [];
    for (let i = 0; i < length; i++) {
      list.push(Math.floor(Math.random() * (max - min + 1)) + min);
    }
    return list;
  }

  // Function to perform Bubble Sort
  function bubbleSort(arr) {
    let startTime = performance.now();
    let len = arr.length;
    for (let i = 0; i < len; i++) {
      for (let j = 0; j < len - i - 1; j++) {
        if (arr[j] > arr[j + 1]) {
          // Swap
          let temp = arr[j];
          arr[j] = arr[j + 1];
          arr[j + 1] = temp;
        }
      }
    }
    let endTime = performance.now();
    let executionTime = endTime - startTime;
    document.getElementById('bubbleSortTime').textContent = "Bubble Sort execution time: " + executionTime + " milliseconds";
    return arr;
  }
```

The first one is a function that generates the random numbers from the parameters inputted by the user and initializes an empty array called “list”. Then it fills an array with those numbers.

The second function performs the Bubble Sort algorithm to sort the input array arr. It records the start time of the sorting process using performance.now(). It iterates over the array using nested loops to compare adjacent elements and swap them if necessary. After completing the sorting process, it records the end time and calculates the execution time.
It updates the HTML element with the ID bubbleSortTime to display the execution time. Finally, it returns the sorted array.

```
  // Function to perform Selection Sort
  function selectionSort(arr) {
    let startTime = performance.now();
    let len = arr.length;
    for (let i = 0; i < len - 1; i++) {
      let minIndex = i;
      for (let j = i + 1; j < len; j++) {
        if (arr[j] < arr[minIndex]) {
          minIndex = j;
        }
      }
      if (minIndex !== i) {
        // Swap
        let temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
      }
    }
    let endTime = performance.now();
    let executionTime = endTime - startTime;
    document.getElementById('selectionSortTime').textContent = "Selection Sort execution time: " + executionTime + " milliseconds";
    return arr;
  }

  // Function to display only the first 5 and last 5 numbers of an array
  function displayFirstAndLast(arr) {
    let displayArr = arr.slice(0, 5).concat(["(...)"]).concat(arr.slice(-5));
    return displayArr;
  }
```

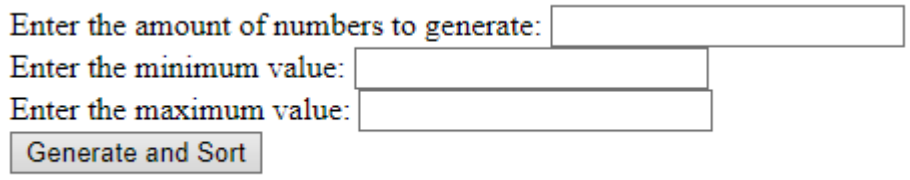
The first function sorts the input array arr using the Selection Sort algorithm. It records the start time of the sorting process. It iterates over the array to find the minimum element and swaps it with the element at the current position. After sorting, it records the end time and calculates the execution time. It updates the HTML element with the ID selectionSortTime to display the execution time. Finally, it returns the sorted array.

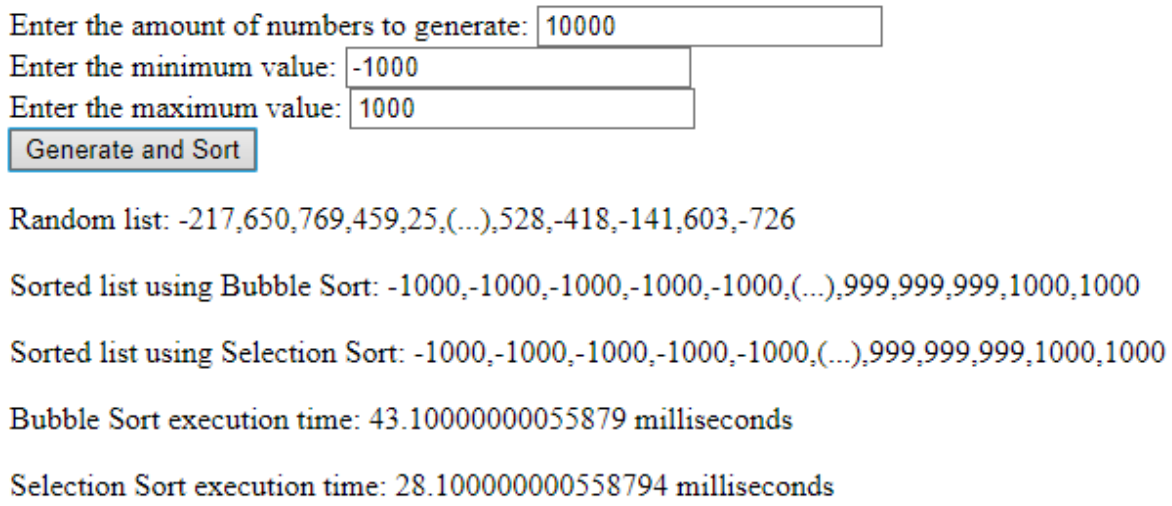
The second function I have made myself in order to display only the first and last five numbers of an array while also indicating it with (...) sign using concat which joins two or more strings together.

```
    // Function to handle form submission
  function handleSubmit(event) {
    event.preventDefault();
    const amount = parseInt(document.getElementById("amount").value);
    const min = parseInt(document.getElementById("min").value);
    const max = parseInt(document.getElementById("max").value);
    if (isNaN(amount) || amount <= 0 || isNaN(min) || isNaN(max) || min >= max) {
      alert("Please enter valid values. Amount should be a positive number, and min should be less than max.");
      return;
    }
    let randomList = generateRandomList(amount, min, max);
    let output = "<p>Random list: " + displayFirstAndLast(randomList) + "</p>";

    // Bubble Sort
    let sortedListBubble = bubbleSort([...randomList]);
    output += "<p>Sorted list using Bubble Sort: " + displayFirstAndLast(sortedListBubble) + "</p>";

    // Selection Sort
    let sortedListSelection = selectionSort([...randomList]);
    output += "<p>Sorted list using Selection Sort: " + displayFirstAndLast(sortedListSelection) + "</p>";

    document.getElementById("results").innerHTML = output;
  }
</script>
```

Here is the function that makes sure that the data inputted by the user is correct and only if it is it calls for generateRandomList function and later uses the output of that function in order to call Bubble Sort and Selection Sort functions. It constructs an HTML string to display the generated random list and the sorted lists. And finally, it updates the HTML element with the ID results to display the output.

## 4. The final project:




## Here is the basis for our comparisons

Enter the amount of numbers to generate: 100000
Enter the minimum value: -1000
Enter the maximum value: 1000
Generate and Sort

Random list: 372,-69,-285,611,-834,(...),615,310,-269,-83,-889

Sorted list using Bubble Sort: -1000,-1000,-1000,-1000,-1000,(...),1000,1000,1000,1000,1000

Sorted list using Selection Sort: -1000,-1000,-1000,-1000,-1000,(...),1000,1000,1000,1000,1000

Bubble Sort execution time: 6942.200000000186 milliseconds

Selection Sort execution time: 2397.2000000001863 milliseconds

**Here I increased the amount of numbers to generate ten times and the execution times increased dramatically indicating that both methods would not be too efficient in handling large amounts of data.**

Enter the amount of numbers to generate: 10000
Enter the minimum value: -1000000
Enter the maximum value: 1000000
Generate and Sort

Random list: 250576,-64065,-977837,789081,479325,(...),731448,-291483,508662,61664,311974

Sorted list using Bubble Sort: -999852,-999837,-998783,-998244,-998157,(...),999595,999644,999829,999846,999944

Sorted list using Selection Sort: -999852,-999837,-998783,-998244,-998157,(...),999595,999644,999829,999846,999944

Bubble Sort execution time: 49.700000000186265 milliseconds

Selection Sort execution time: 28.600000000558794 milliseconds

**Here I increased the minimum and maximum value of a number and it did not seem to affect the execution times at all.**