.

Lesson 04 Demo 07

Implementing Count Sort Algorithm

Objective: To demonstrate the count sort algorithm and explain its time and space complexity using JavaScript

Tools required: Visual Studio Code and Node.js

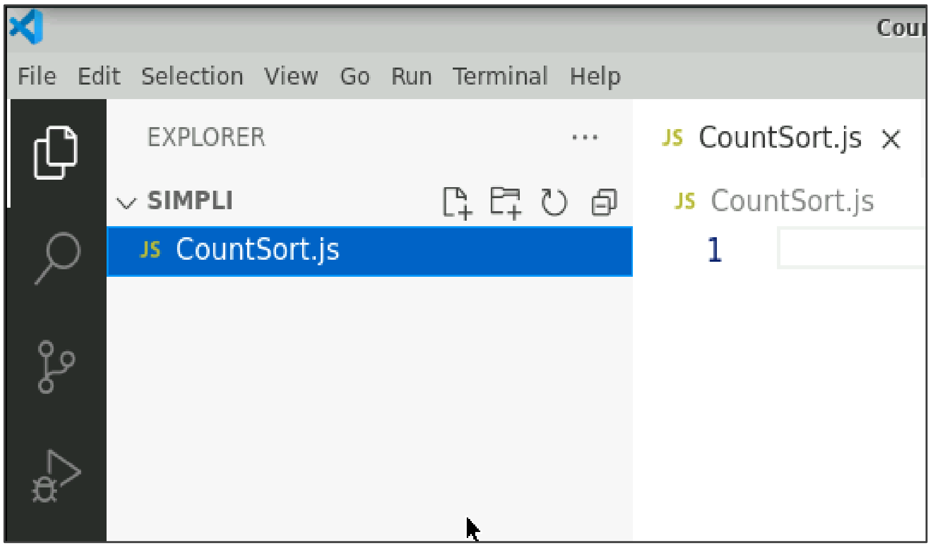
Prerequisites: Basic understanding of arrays and loops in JavaScript

Steps to be followed:

1. Create and execute the JS file

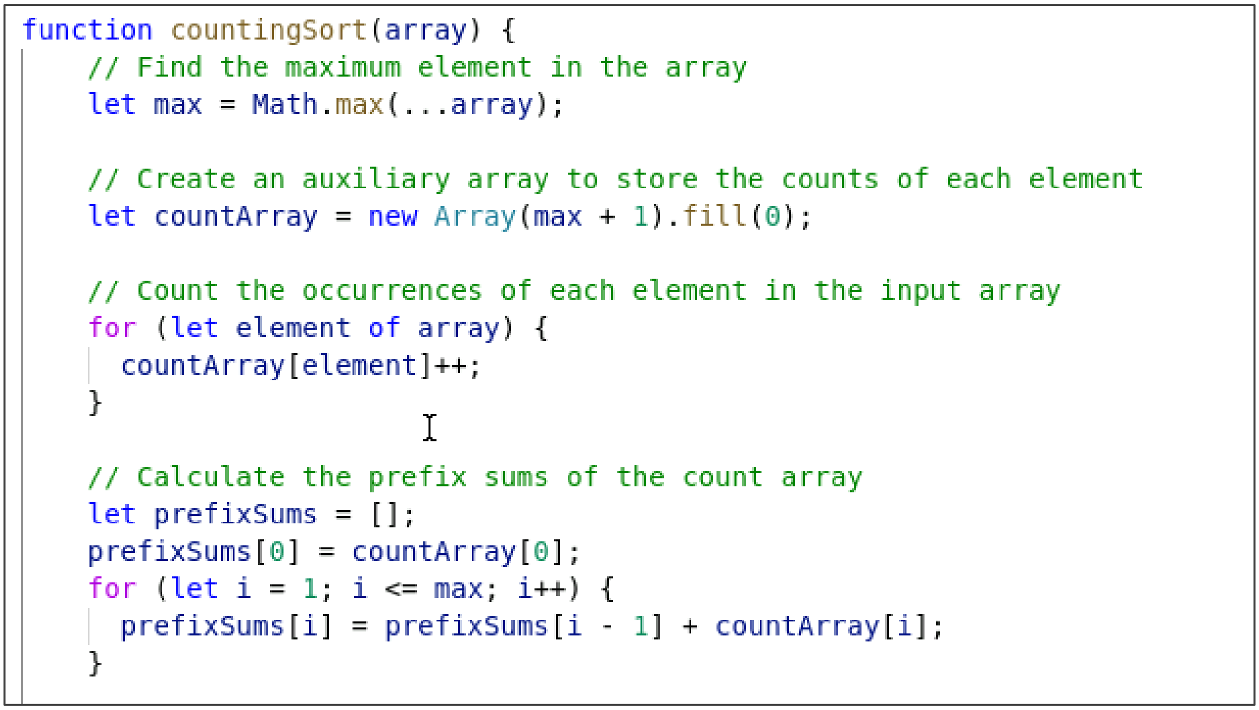
Step 1: Create and execute the JS file

1.1 Open the Visual Studio Code editor and create a JavaScript file named **CountSort.js**

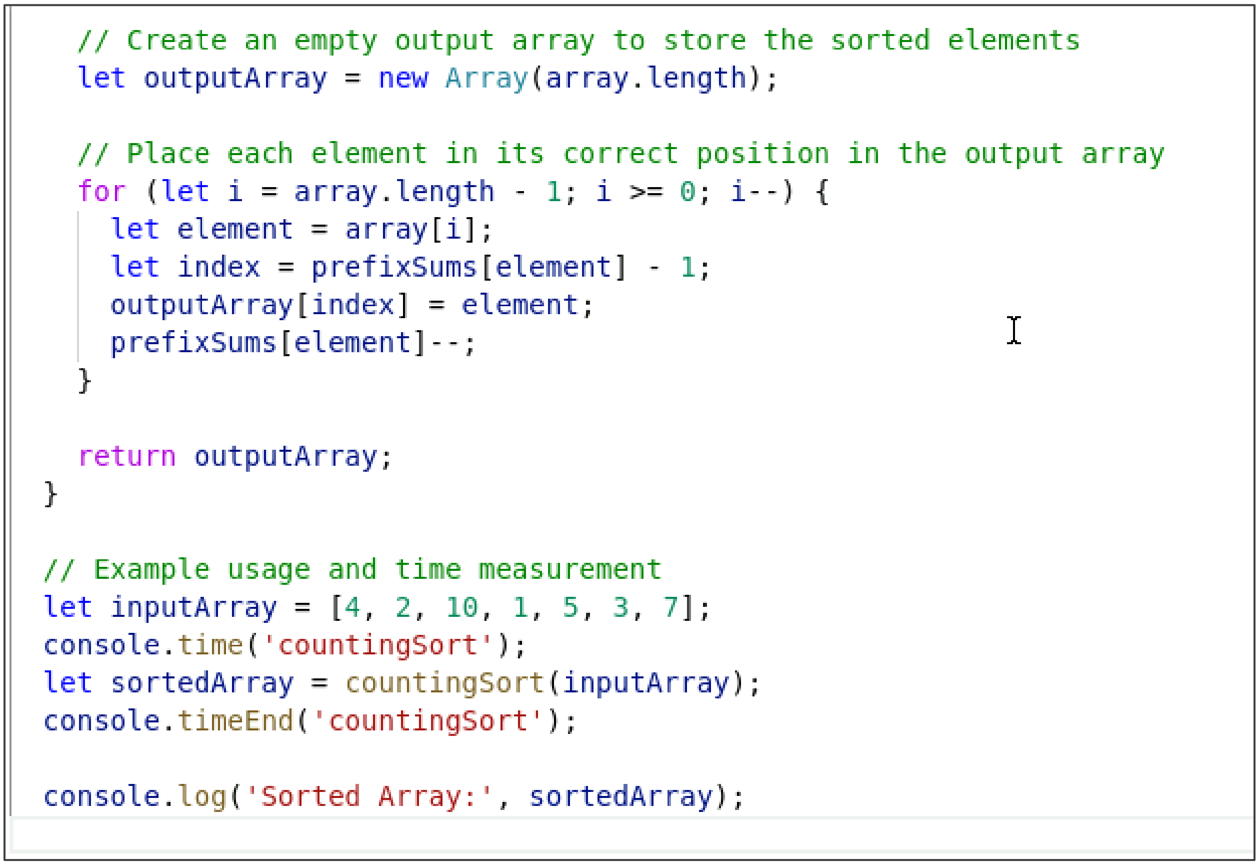


1.2 Write the code given below in the **CountSort.js** file:

```
function countingSort(array) {
    // Find the maximum element in the array
    let max = Math.max(...array);

    // Create an auxiliary array to store the counts of each element
    let countArray = new Array(max + 1).fill(0);

    // Count the occurrences of each element in the input array
    for (let element of array) {
      countArray[element]++;
    }

    // Calculate the prefix sums of the count array
    let prefixSums = [];
    prefixSums[0] = countArray[0];
    for (let i = 1; i <= max; i++) {
      prefixSums[i] = prefixSums[i - 1] + countArray[i];
    }

    // Create an empty output array to store the sorted elements
    let outputArray = new Array(array.length);

    // Place each element in its correct position in the output array
    for (let i = array.length - 1; i >= 0; i--) {
      let element = array[i];
      let index = prefixSums[element] - 1;
      outputArray[index] = element;
      prefixSums[element]--;
    }

    return outputArray;
  }

  // Example usage and time measurement
  let inputArray = [4, 2, 10, 1, 5, 3, 7];
  console.time('countingSort');
  let sortedArray = countingSort(inputArray);
  console.timeEnd('countingSort');

  console.log('Sorted Array:', sortedArray)
```

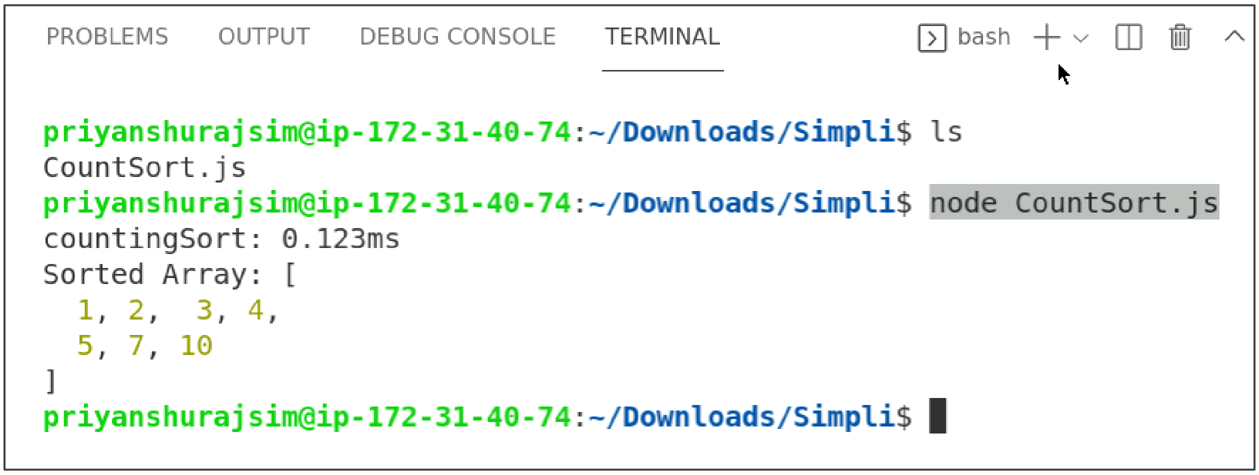


```
function countingSort(array) {
    // Find the maximum element in the array
    let max = Math.max(...array);

    // Create an auxiliary array to store the counts of each element
    let countArray = new Array(max + 1).fill(0);

    // Count the occurrences of each element in the input array
    for (let element of array) {
      countArray[element]++;
    }

    // Calculate the prefix sums of the count array
    let prefixSums = [];
    prefixSums[0] = countArray[0];
    for (let i = 1; i <= max; i++) {
      prefixSums[i] = prefixSums[i - 1] + countArray[i];
    }
```



```
  // Create an empty output array to store the sorted elements
  let outputArray = new Array(array.length);

  // Place each element in its correct position in the output array
  for (let i = array.length - 1; i >= 0; i--) {
    let element = array[i];
    let index = prefixSums[element] - 1;
    outputArray[index] = element;
    prefixSums[element]--;
  }

  return outputArray;
}

// Example usage and time measurement
let inputArray = [4, 2, 10, 1, 5, 3, 7];
console.time('countingSort');
let sortedArray = countingSort(inputArray);
console.timeEnd('countingSort');

console.log('Sorted Array:', sortedArray);
```

1.3 Save the file and execute it in the terminal using the following command:
node CountSort.js



```
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$ ls
CountSort.js
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$ node CountSort.js
countingSort: 0.123ms
Sorted Array: [
  1, 2,  3, 4,
  5, 7, 10
]
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$ 
```

In the example, we used the count sort algorithm in JavaScript to arrange the items in an array. It has a time complexity of O(n + k) and a space complexity of O(n).

By following these steps, you have successfully implemented and executed the count sort algorithm in JavaScript, including measuring its execution time.