Lesson 04 Demo 04

Implementing Merge Sort Algorithm

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

Tools required: Visual Studio Code and Node.js

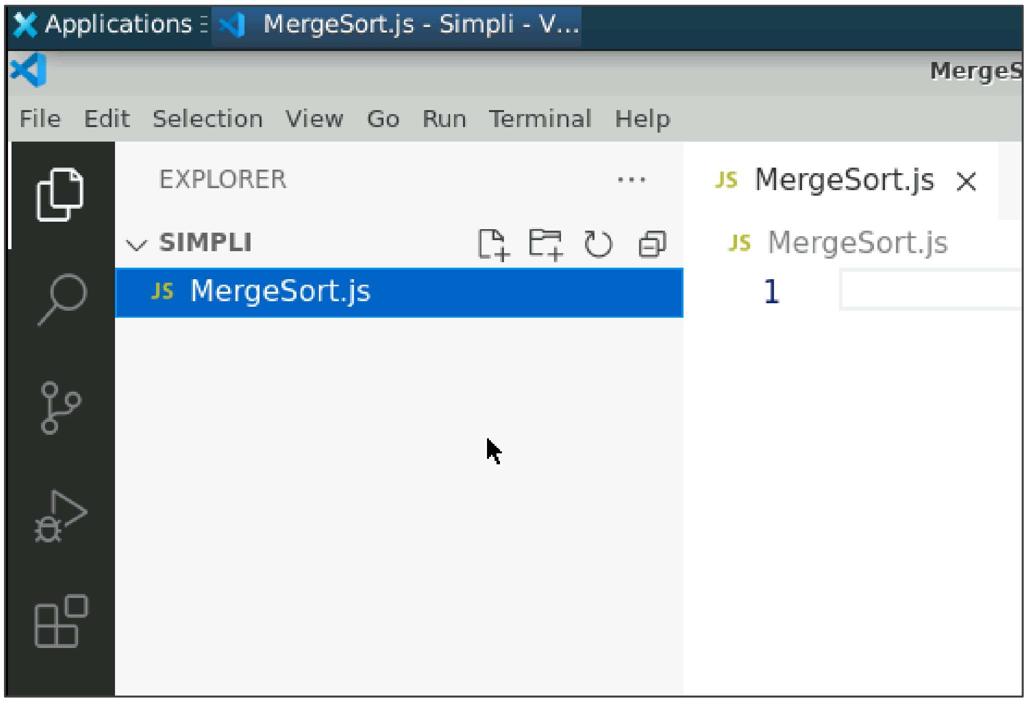
Prerequisites: Basic understanding of arrays and loops in JavaScript

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 **MergeSort.js**


Applications
MergeSort.js - Simpli - V...
MergeS
File Edit Selection View Go Run Terminal Help
EXPLORER
SIMPLI
MergeSort.js
MergeSort.js
MergeSort.js

.

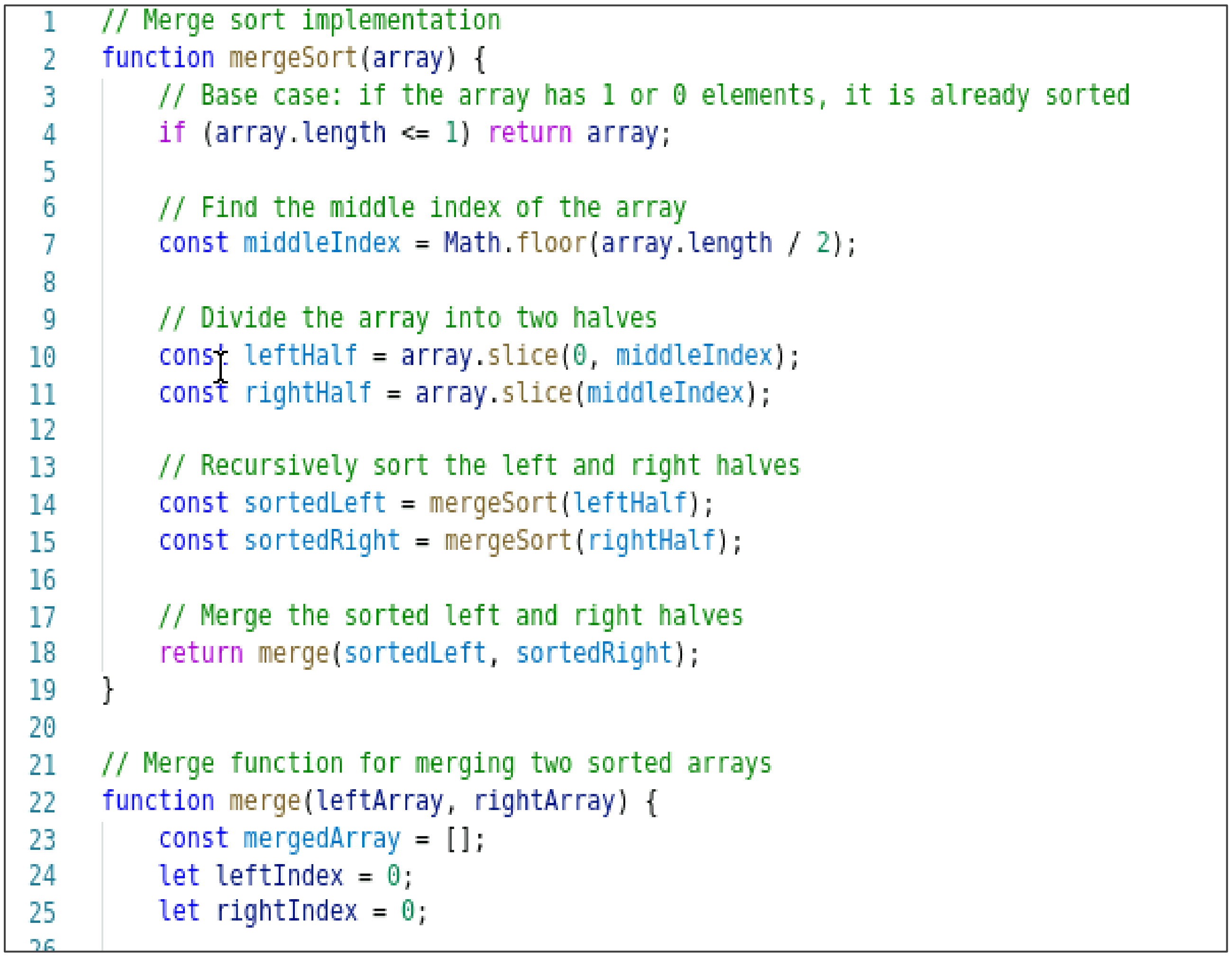
1.2 Write the code given below in the **MergeSort.js** file:

```
// Merge sort implementation
function mergeSort(array) {
  // Base case: if the array has 1 or 0 elements, it is already sorted
  if (array.length <= 1) return array;

  // Find the middle index of the array
  const middleIndex = Math.floor(array.length / 2);

  // Divide the array into two halves
  const leftHalf = array.slice(0, middleIndex);
  const rightHalf = array.slice(middleIndex);

  // Recursively sort the left and right halves
  const sortedLeft = mergeSort(leftHalf);
  const sortedRight = mergeSort(rightHalf);

  // Merge the sorted left and right halves
  return merge(sortedLeft, sortedRight);
}

// Merge function for merging two sorted arrays
function merge(leftArray, rightArray) {
  const mergedArray = [];
  let leftIndex = 0;
  let rightIndex = 0;

  // Merge the two sorted arrays
  while (leftIndex < leftArray.length && rightIndex < rightArray.length) {
    if (leftArray[leftIndex] <= rightArray[rightIndex]) {
      mergedArray.push(leftArray[leftIndex]);
      leftIndex++;
    } else {
      mergedArray.push(rightArray[rightIndex]);
      rightIndex++;
    }
  }

 // Concatenate the remaining elements from both arrays (if any)
  return
mergedArray.concat(leftArray.slice(leftIndex)).concat(rightArray.slice(rightIndex));
}
```

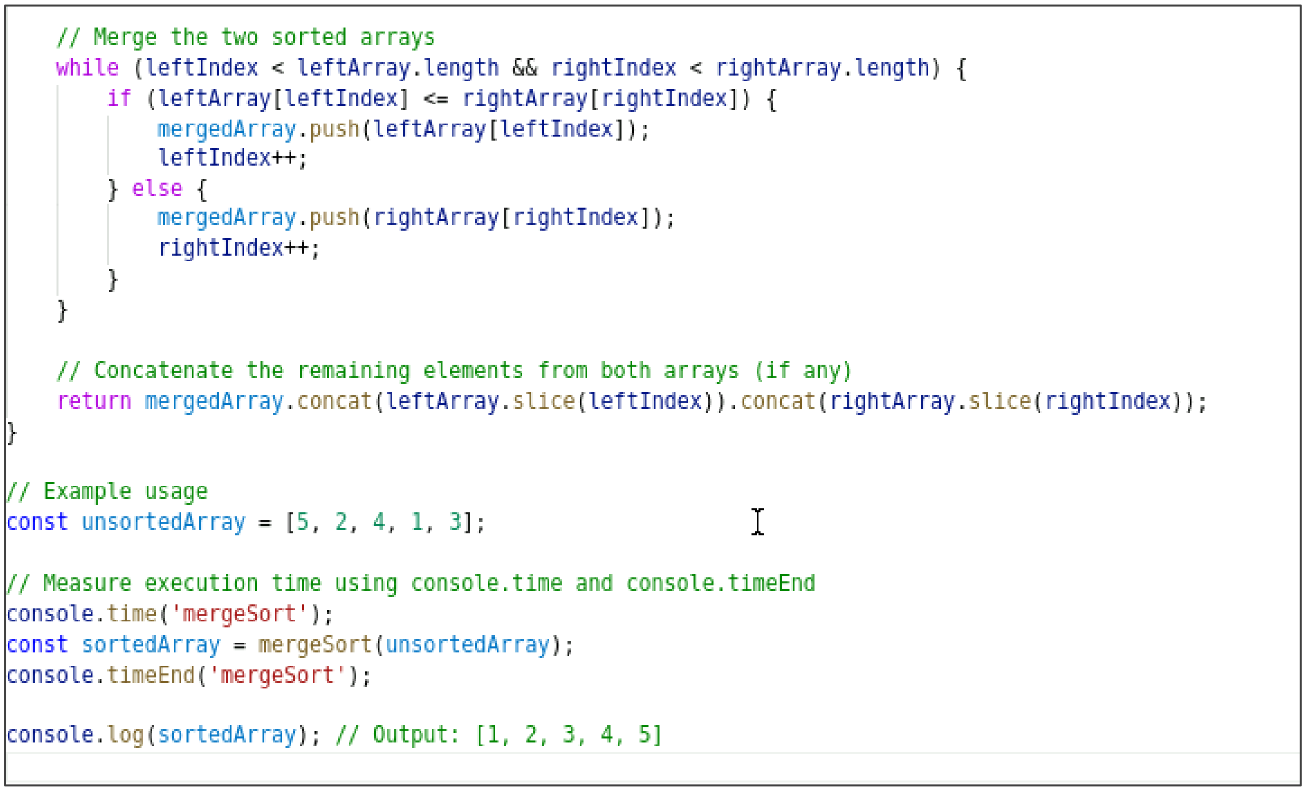
// Example usage
const unsortedArray = [5, 2, 4, 1, 3];

// Measure execution time using console.time and console.timeEnd
console.time('mergeSort');
const sortedArray = mergeSort(unsortedArray);
console.timeEnd('mergeSort');

console.log(sortedArray); // Output: [1, 2, 3, 4, 5]

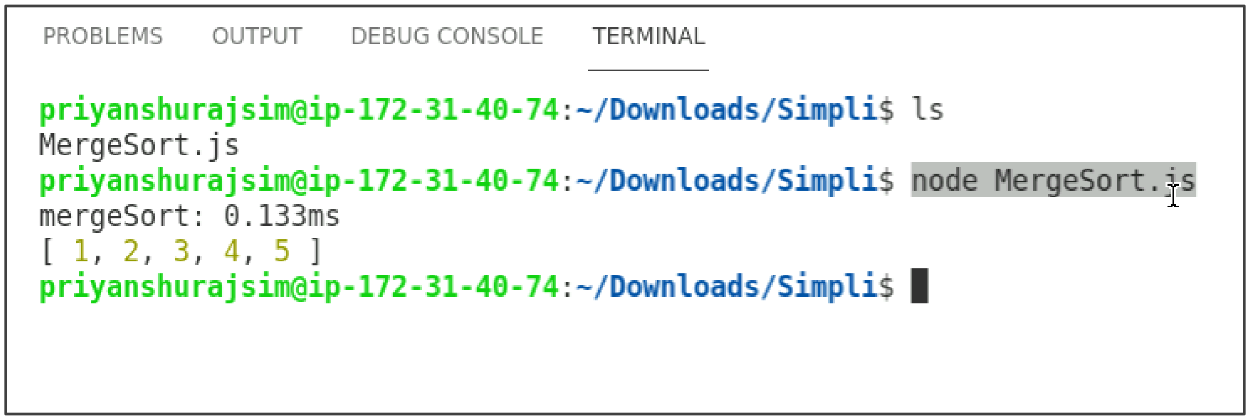


```
// Merge sort implementation
function mergeSort(array) {
    // Base case: if the array has 1 or 0 elements, it is already sorted
    if (array.length <= 1) return array;

    // Find the middle index of the array
    const middleIndex = Math.floor(array.length / 2);

    // Divide the array into two halves
    const leftHalf = array.slice(0, middleIndex);
    const rightHalf = array.slice(middleIndex);

    // Recursively sort the left and right halves
    const sortedLeft = mergeSort(leftHalf);
    const sortedRight = mergeSort(rightHalf);

    // Merge the sorted left and right halves
    return merge(sortedLeft, sortedRight);
}

// Merge function for merging two sorted arrays
function merge(leftArray, rightArray) {
    const mergedArray = [];
    let leftIndex = 0;
    let rightIndex = 0;
```



```
    // Merge the two sorted arrays
    while (leftIndex < leftArray.length && rightIndex < rightArray.length) {
        if (leftArray[leftIndex] <= rightArray[rightIndex]) {
            mergedArray.push(leftArray[leftIndex]);
            leftIndex++;
        } else {
            mergedArray.push(rightArray[rightIndex]);
            rightIndex++;
        }
    }

    // Concatenate the remaining elements from both arrays (if any)
    return mergedArray.concat(leftArray.slice(leftIndex)).concat(rightArray.slice(rightIndex));
}

// Example usage
const unsortedArray = [5, 2, 4, 1, 3];

// Measure execution time using console.time and console.timeEnd
console.time('mergeSort');
const sortedArray = mergeSort(unsortedArray);
console.timeEnd('mergeSort');

console.log(sortedArray); // Output: [1, 2, 3, 4, 5]
```

1.3 Save the file and execute it on the terminal using the command given below:
node MergeSort.js



```
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$ ls
MergeSort.js
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$ node MergeSort.js
mergeSort: 0.133ms
[ 1, 2, 3, 4, 5 ]
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$
```

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

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