Lesson 04 Demo 02

Implementing Selection Sort Algorithm

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

Tools required: Visual Studio Code and Node.js

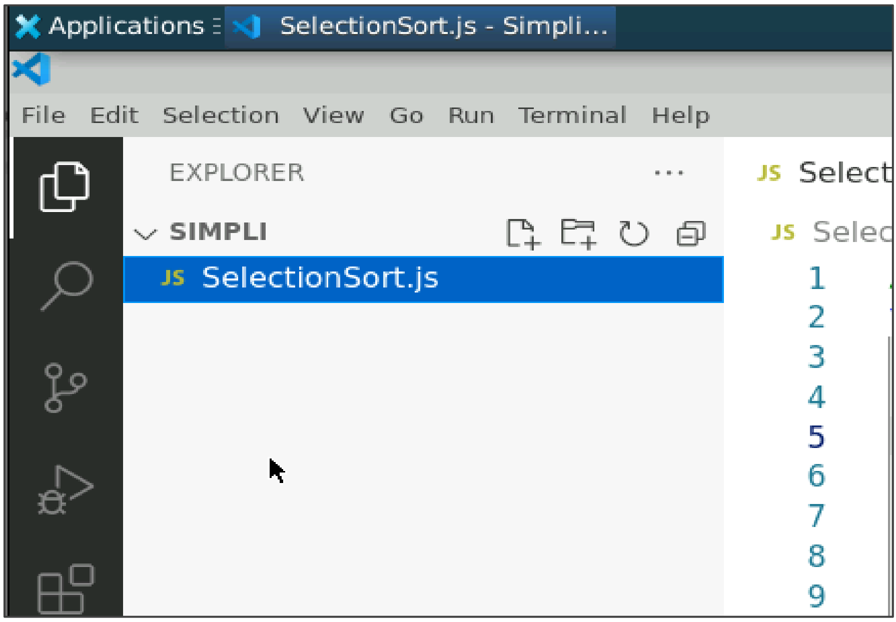
Prerequisites: Basic understanding of arrays and loops in JavaScript

Steps to be followed:

1. Create and execute JS file

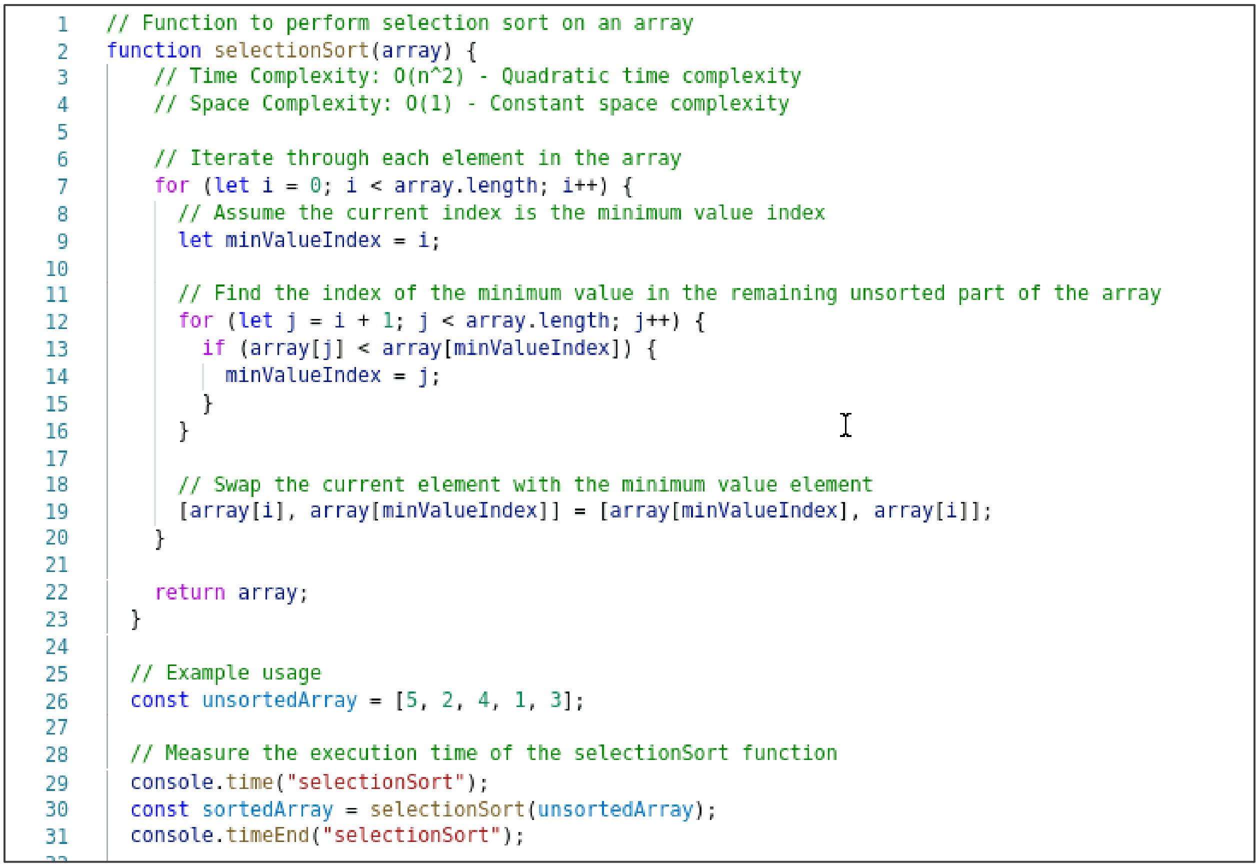
Step 1: Create and execute JS file

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



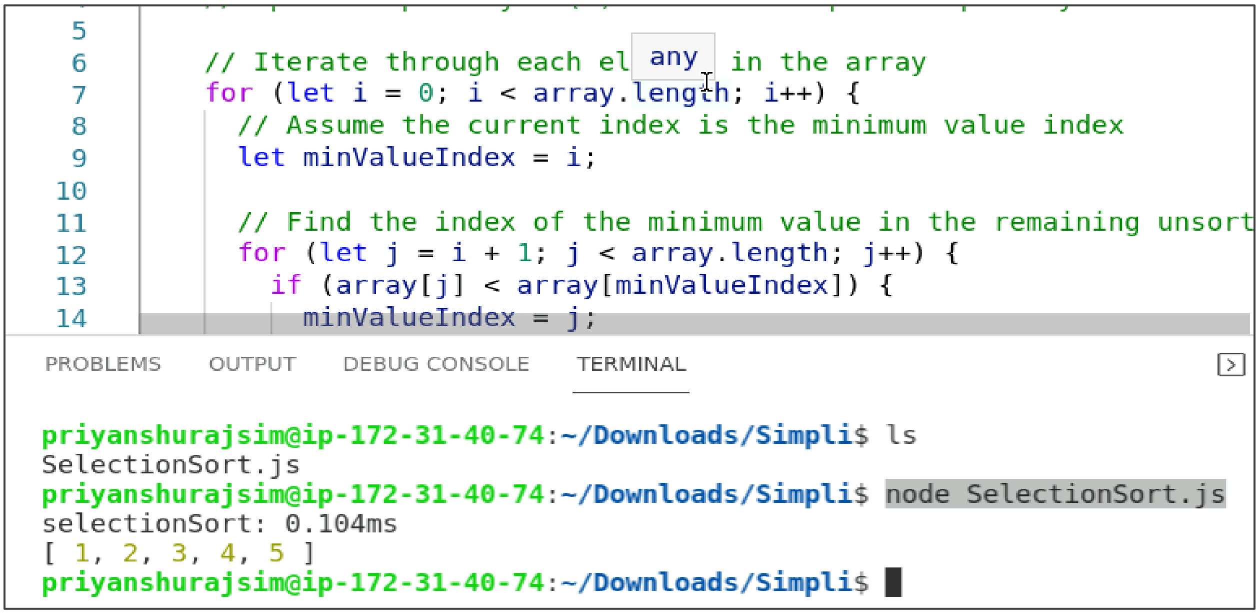
1.2 Write the code given below in the **SelectionSort.js** file:

```
// Function to perform selection sort on an array
function selectionSort(array) {
  // Time Complexity: O(n^2) - Quadratic time complexity
  // Space Complexity: O(1) - Constant space complexity

  // Iterate through each element in the array
  for (let i = 0; i < array.length; i++) {
   // Assume the current index is the minimum value index
   let minValueIndex = i;

   // Find the index of the minimum value in the remaining unsorted part of the array
   for (let j = i + 1; j < array.length; j++) {
    if (array[j] < array[minValueIndex]) {
     minValueIndex = j;
    }
   }

   // Swap the current element with the minimum value element
   [array[i], array[minValueIndex]] = [array[minValueIndex], array[i]];
  }

  return array;
 }

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

 // Measure the execution time of the selectionSort function
 console.time("selectionSort");
 const sortedArray = selectionSort(unsortedArray);
 console.timeEnd("selectionSort");

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



```
// Function to perform selection sort on an array
function selectionSort(array) {
    // Time Complexity: O(n^2) - Quadratic time complexity
    // Space Complexity: O(1) - Constant space complexity

    // Iterate through each element in the array
    for (let i = 0; i < array.length; i++) {
      // Assume the current index is the minimum value index
      let minValueIndex = i;

      // Find the index of the minimum value in the remaining unsorted part of the array
      for (let j = i + 1; j < array.length; j++) {
        if (array[j] < array[minValueIndex]) {
          minValueIndex = j;
        }
      }

      // Swap the current element with the minimum value element
      [array[i], array[minValueIndex]] = [array[minValueIndex], array[i]];
    }

    return array;
  }

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

  // Measure the execution time of the selectionSort function
  console.time("selectionSort");
  const sortedArray = selectionSort(unsortedArray);
  console.timeEnd("selectionSort");
```

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



```

    // Iterate through each el      in the array
    for (let i = 0; i < array.length; i++) {
      // Assume the current index is the minimum value index
      let minValueIndex = i;

      // Find the index of the minimum value in the remaining unsort
      for (let j = i + 1; j < array.length; j++) {
        if (array[j] < array[minValueIndex]) {
          minValueIndex = j;
```

```
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$ ls
SelectionSort.js
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$ node SelectionSort.js
selectionSort: 0.104ms
[ 1, 2, 3, 4, 5 ]
priyanshurajsim@ip-172-31-40-74:~/Downloads/Simpli$
```

In our example, we used the selection sort algorithm in JavaScript to arrange the items in an array. Its time complexity of O(n^2) and a space complexity of O(1).

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