.

Lesson 04 Demo 02

Implementing Selection Sort Algorithm

Objective: To sort an array using selection sort in JavaScript, understanding its time and space complexity for organizing fixed-size datasets such as weekly task lists or student scores

Tools required: Visual Studio Code and Node.js

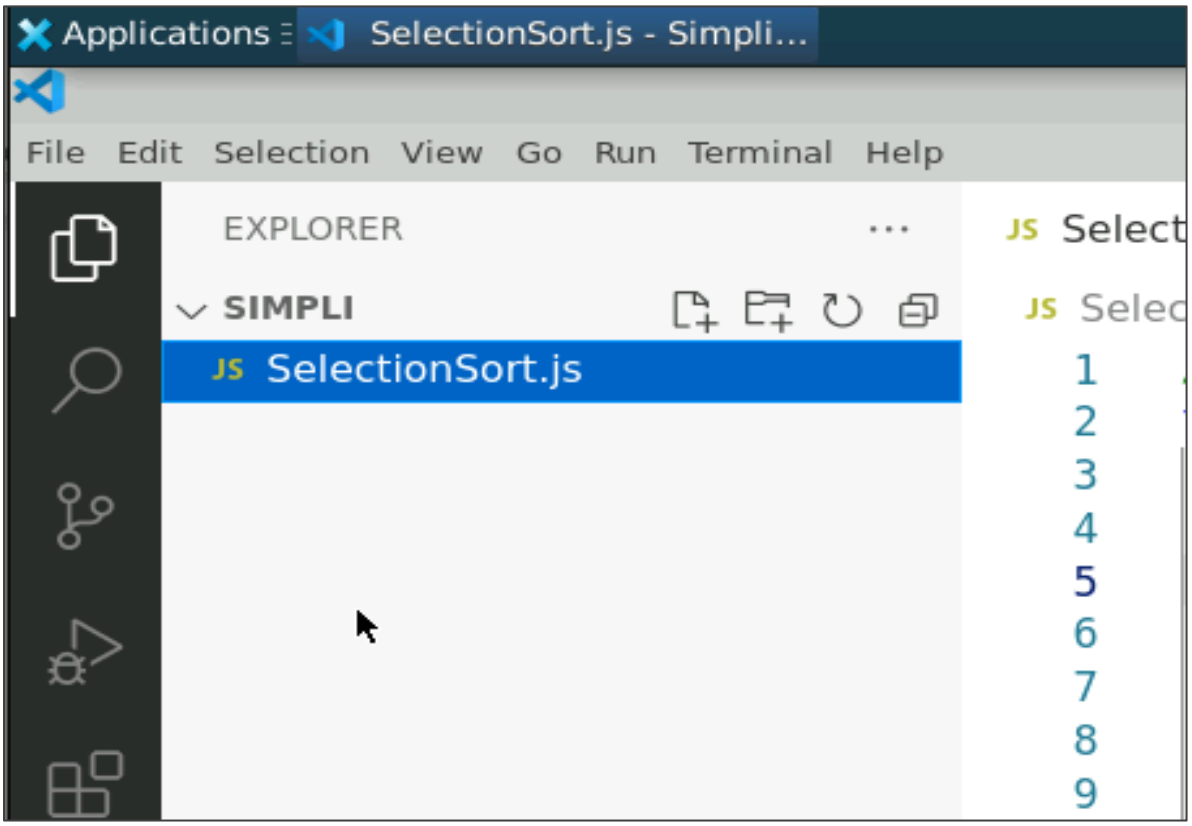
Prerequisites: A basic understanding of arrays and loops in JavaScript

Steps to be followed:

1. Create a JavaScript file and execute it

Step 1: Create a JavaScript file and execute it

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



.

1.2 Add the following code to the 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);
```

.

```javascript
// 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");
```

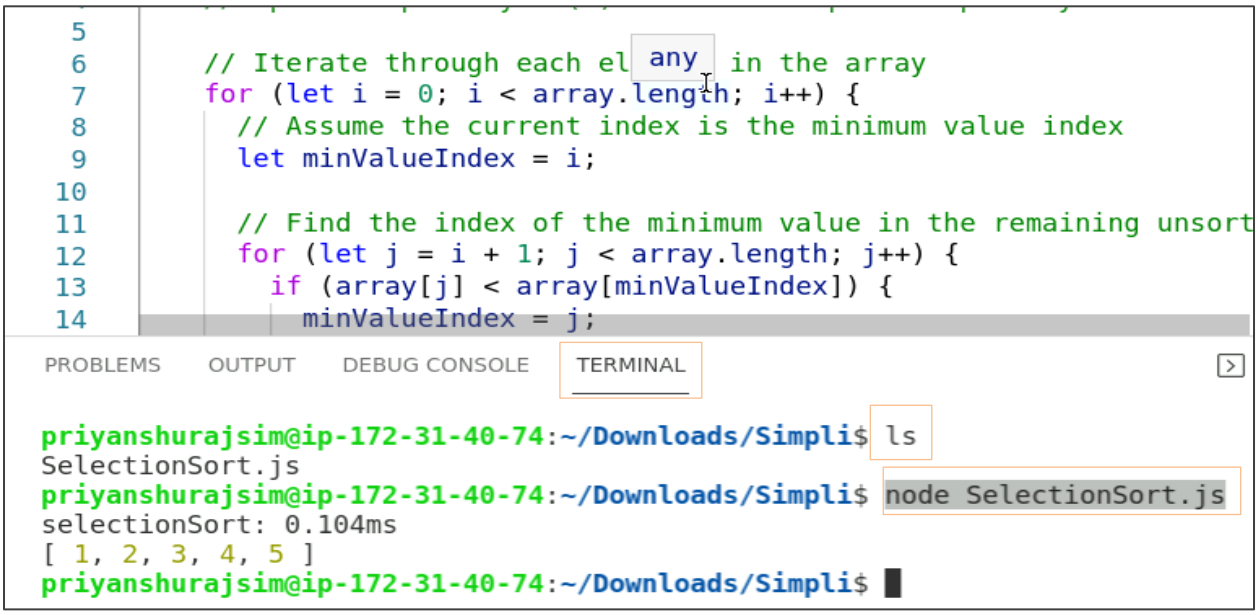
```javascript
  // Example usage
  const unsortedArray = [5, 2, 4, 1, 3];

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

  console.log(sortedArray);
```

.

1.3 Press **Ctrl** + **S** to save the file and then execute it in the **TERMINAL** using the following commands:

ls

node SelectionSort.js



By following these steps, you have successfully used the selection sort algorithm in JavaScript to organize small datasets like task lists or scores, understood how it arranges data through comparisons, and learned that it has a time complexity of $O(n^2)$ and space complexity of $O(1)$.