

## Lesson 04 Demo 06

### Implementing Heap Sort Algorithm

**Objective:** To sort data using the heap sort algorithm in JavaScript for managing priority-based tasks such as job scheduling or leaderboard rankings

**Tools required:** Visual Studio Code and Node.js

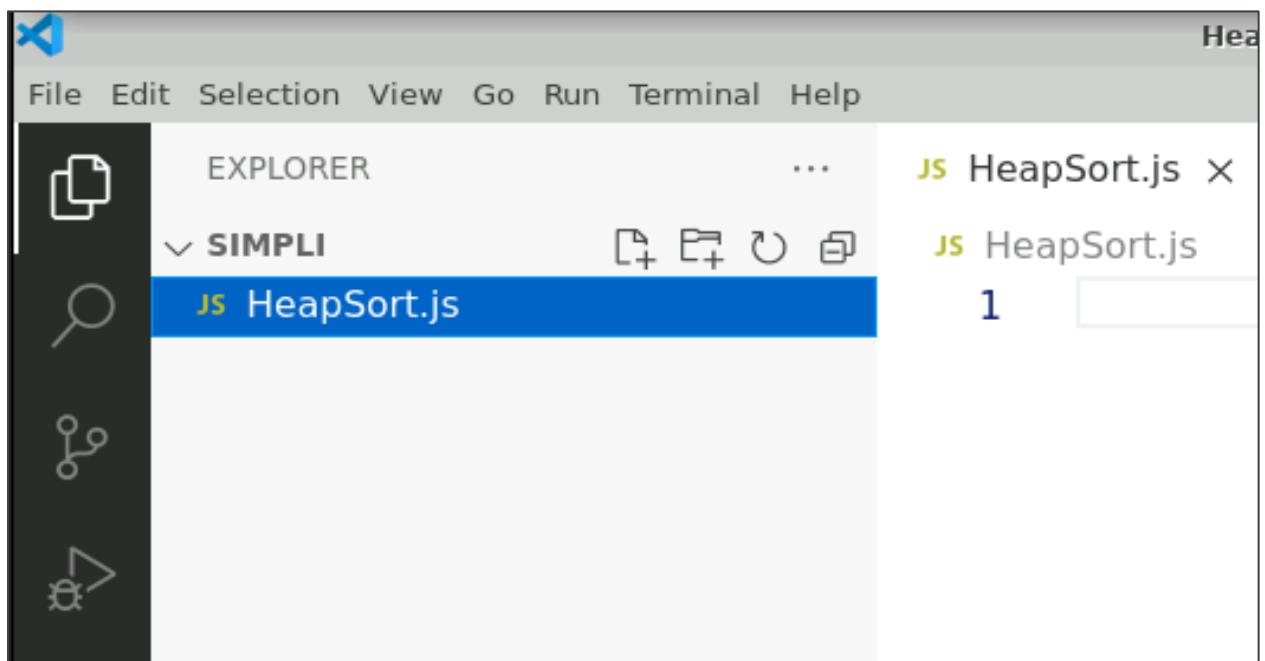
**Prerequisites:** A basic understanding of arrays and loops in JavaScript

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 **HeapSort.js**



1.2 Add the following code to the file:

```
// Function to swap two elements in an array
function swap(arr, i, j) {
    const temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

// Function to heapify a subtree rooted at index i
function heapify(arr, n, i) {
    const left = 2 * i + 1;
    const right = 2 * i + 2;

    let largest = i;

    // Compare with left child
    if (left < n && arr[left] > arr[largest]) {
        largest = left;
    }

    // Compare with right child
    if (right < n && arr[right] > arr[largest]) {
        largest = right;
    }

    // If the largest element is not the root, swap and heapify the affected subtree
    if (largest !== i) {
        swap(arr, i, largest);
        heapify(arr, n, largest);
    }
}

// Function to perform heap sort on the given array
function heapSort(arr) {
    const n = arr.length;

    // Build a max heap (rearrange array)
    for (let i = Math.floor(n / 2) - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }
}
```

```

// One by one extract elements from the heap
for (let i = n - 1; i >= 0; i--) {
    // Move the current root to the end
    swap(arr, 0, i);

    // Call heapify on the reduced heap
    heapify(arr, i, 0);
}

// Example usage
const arr = [3, 0, 2, 5, -1, 4, 1];
console.log('Unsorted array:', arr);
// Measure the execution time of heapSort
console.time('Heap Sort');
heapSort(arr);
console.timeEnd('Heap Sort');

console.log('Sorted array:', arr);

```

```

// Function to swap two elements in an array
function swap(arr, i, j) {
    const temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

// Function to heapify a subtree rooted at index i
function heapify(arr, n, i) {
    const left = 2 * i + 1;
    const right = 2 * i + 2;

    let largest = i;

    // Compare with left child
    if (left < n && arr[left] > arr[largest]) {
        largest = left;
    }

    // Compare with right child
    if (right < n && arr[right] > arr[largest]) {
        largest = right;
    }

    // If the largest element is not the root, swap and heapify the affected subtree
    if (largest !== i) {
        swap(arr, i, largest);
        heapify(arr, n, largest);
    }
}

```

```

// Function to perform heap sort on the given array
function heapSort(arr) {
    const n = arr.length;

    // Build a max heap (rearrange array)
    for (let i = Math.floor(n / 2) - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }

    // One by one extract elements from the heap
    for (let i = n - 1; i >= 0; i--) {
        // Move the current root to the end
        swap(arr, 0, i);

        // Call heapify on the reduced heap
        heapify(arr, i, 0);
    }
}

// Example usage
const arr = [3, 0, 2, 5, -1, 4, 1];
console.log('Unsorted array:', arr);

// Measure the execution time of heapSort
console.time('Heap Sort');
heapSort(arr);
console.timeEnd('Heap Sort');

console.log('Sorted array:', arr);

```

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

**ls**

**node HeapSort.js**

The screenshot shows the VS Code interface with the TERMINAL tab selected. The terminal window displays the output of the 'ls' command, listing various files and directories in the current directory.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● labuser@ip-172-31-24-201:~$ ls
BubbleSort.js  Music      mytailwindapp
Desktop        Pictures   node_modules
Documents      Public     package-lock.json
Downloads      Templates  package.json
HeapSort.js    Videos    snap
Heap Sort.js   express-app thinclient drives

```

The screenshot shows a terminal window within a code editor interface. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The terminal content is as follows:

```
MergeSort.js  my-app
● labuser@ip-172-31-24-201:~$ node HeapSort.js
Debugger attached.
Unsorted array: [
  3, 0, 2, 5,
  -1, 4, 1
]
```

The command `node HeapSort.js` is highlighted with a red box. On the right side of the terminal, there is a sidebar with two items: "bash" and "JavaScript...". The "JavaScript..." item is selected, indicated by a blue background.

By following these steps, you have successfully used the heap sort algorithm in JavaScript to efficiently organize data such as job queues or leaderboard rankings, and learned that it has a time complexity of  $O(n \log n)$  and a space complexity of  $O(1)$ .