Sorting In JavaScript

Sort an Array

```
const arr = [-2, -7, 1000, 5]
console.log(arr.sort()) // -2, -7, 1000, 5
console.log(arr.sort((a, b) => a - b)) // -7, -2 , 5, 1000
console.log(arr.sort((a, b) => b - a)) // 1000, 5, -2, -7

const strArr = ["mango", "apple", "banana"]
console.log(strArr.sort()) // "apple", "banana", "mango"
```

Sort a String

```
const str = "Vishal"
console.log(str.split("").sort().join("")) // "Vahils
```

Bubble Sort In JavaScript

```
const bubbleSort = (arr) => {
    let swapped;
    do {
        swapped = false;
        for (let i = 0; i < arr.length - 1; i++) {
```

```
            if (arr[i] > arr[i + 1]) {
                [arr[i], arr[i + 1]] = [arr[i + 1], arr[i]]
                swapped = true;
            }
        }
    } while (swapped)
    return arr;
}

console.log(bubbleSort(arr)) // -7, -2 , 5, 1000
```

Selection Sort in JavaScript

```
const selectionSort = (arr) => {
    for (let i = 0; i < arr.length - 1; i++) {
        let minIndex = i;
        for (let j = i + 1; j < arr.length; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        [arr[minIndex], arr[i]] = [arr[i], arr[minIndex]]
    }
    return arr;
}

console.log(selectionSort(arr)) // -7, -2 , 5, 1000
```

Insertion Sort In JavaScript

```
const insertionSort = (arr) => {
    for(let i=1; i<arr.length; i++){
        let current = arr[i];
        let j = i-1;
        while(j >= 0 && arr[j] > current){
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = current;
    }
    return arr;
}

console.log(insertionSort(arr)) // -7, -2 , 5, 1000
```

Merge Sort in JavaScript

```
const mergeSort = (arr) => {
    if (arr.length < 2) {
        return arr;
    }
    let mid = Math.floor(arr.length / 2);
    let left = mergeSort(arr.slice(0, mid))
    let right = mergeSort(arr.slice(mid))
    return merge(left, right)
}

const merge = (left, right) => {
    const result = []
    let leftIndex = 0, rightIndex = 0;
    while (leftIndex < left.length && rightIndex < right.length) {
        if (left[leftIndex] < right[rightIndex]) {
            result.push(left[leftIndex])
            leftIndex++;
        }
        else {
            result.push(right[rightIndex])
            rightIndex++;
        }
    }

    while (leftIndex < left.length) {
        result.push(left[leftIndex])
        leftIndex++;
    }

    while (rightIndex < right.length) {
        result.push(right[rightIndex])
        rightIndex++;
    }

    return result;
}

const arr1 = [29, 10, 8, 16, 37, 14, 4, 45]
console.log(mergeSort(arr1))
```

Merge Sort in JavaScript (Space Optimised)

```
const mergeSortInplace = (arr, low, high) => {
    if (low < high) {
        let mid = Math.floor((low + high) / 2);
        mergeSortInplace(arr, low, mid)
        mergeSortInplace(arr, mid + 1, high)
```

```
        mergeInplace(arr, low, mid, high)
    }
}

const mergeInplace = (arr, low, mid, high) => {
    const result = []
    let leftIndex = low, rightIndex = mid + 1;
    while (leftIndex <= mid && rightIndex <= high) {
        if (arr[leftIndex] < arr[rightIndex]) {
            result.push(arr[leftIndex])
            leftIndex++;
        }
        else {
            result.push(arr[rightIndex])
            rightIndex++;
        }
    }

    while (leftIndex <= mid) {
        result.push(arr[leftIndex])
        leftIndex++;
    }

    while (rightIndex <= high) {
        result.push(arr[rightIndex])
        rightIndex++;
    }

    for (let i = low; i <= high; i++) {
        arr[i] = result[i - low];
    }
}

const arr1 = [29, 10, 8, 16, 37, 14, 4, 45]
console.log(mergeSortInplace(arr1, 0, arr.length - 1))
console.log(arr1)
```

Quick Sort in JavaScript

```
const quickSort = (arr) => {
    if(arr.length < 2){
        return arr;
    }
    let pivotIndex = Math.floor(Math.random() * arr.length);
    let left = [], right = [];
    for(let i=0; i<arr.length; i++){
        if(i === pivotIndex)
            continue;

        if(arr[i] < arr[pivotIndex]){
            left.push(arr[i])
        }
```

```
        else{
            right.push(arr[i])
        }
    }

    return [...quickSort(left), arr[pivotIndex], ...quickSort(right)]
}

console.log(quickSort(arr1))
```

Practice Question

- How Many Numbers are smaller than the current number
- Merge Sorted Array
- Sort an Array
- Largest Number
- Sort Color