**Assignment18**

**1.**

const mergeIntervals = function(intervals) {

intervals.sort((a, b) => a[0] - b[0]);

const mergedIntervals = [];

let currentInterval = intervals[0];

for (let i = 1; i < intervals.length; i++) {

const [currentStart, currentEnd] = currentInterval;

const [nextStart, nextEnd] = intervals[i];

if (currentEnd >= nextStart) {

currentInterval[1] = Math.max(currentEnd, nextEnd);

} else {

mergedIntervals.push(currentInterval);

currentInterval = intervals[i];

}

}

mergedIntervals.push(currentInterval);

return mergedIntervals;

};

console.log(mergeIntervals([[1, 3], [2, 6], [8, 10], [15, 18]]));

Output: [[1, 6], [8, 10], [15, 18]]

console.log(mergeIntervals([[1, 4], [4, 5]]));

Output: [[1, 5]]

2.

const sortColors = function(nums) {

let low = 0;

let mid = 0;

let high = nums.length - 1;

while (mid <= high) {

if (nums[mid] === 0) {

swap(nums, low, mid);

low++;

mid++;

} else if (nums[mid] === 1) {

mid++;

} else if (nums[mid] === 2) {

swap(nums, mid, high);

high--;

}

}

};

const swap = function(nums, i, j) {

const temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

};

const nums1 = [2, 0, 2, 1, 1, 0];

sortColors(nums1);

console.log(nums1);

Output: [0, 0, 1, 1, 2, 2]

const nums2 = [2, 0, 1];

sortColors(nums2);

console.log(nums2);

Output: [0, 1, 2]

3.

const firstBadVersion = function(n) {

let left = 1;

let right = n;

while (left <= right) {

const mid = left + Math.floor((right - left) / 2);

if (isBadVersion(mid)) {

right = mid - 1;

} else {

left = mid + 1;

}

}

return left;

};

const badVersion1 = 4;

const result1 = firstBadVersion(5);

console.log(result1);

Output: 4

const badVersion2 = 1;

const result2 = firstBadVersion(1);

console.log(result2);

Output: 1

4.

const maximumGap = function(nums) {

if (nums.length < 2) {

return 0;

}

const min = Math.min(...nums);

const max = Math.max(...nums);

const range = Math.max(1, Math.floor((max - min) / (nums.length - 1)));

const minBucket = new Array(nums.length - 1).fill(Infinity);

const maxBucket = new Array(nums.length - 1).fill(-Infinity);

for (let num of nums) {

if (num === min || num === max) {

continue;

}

const index = Math.floor((num - min) / range);

minBucket[index] = Math.min(minBucket[index], num);

maxBucket[index] = Math.max(maxBucket[index], num);

}

let maxDiff = 0;

let prevMax = min;

for (let i = 0; i < nums.length - 1; i++) {

if (minBucket[i] === Infinity && maxBucket[i] === -Infinity) {

continue;

}

maxDiff = Math.max(maxDiff, minBucket[i] - prevMax);

prevMax = maxBucket[i];

}

maxDiff = Math.max(maxDiff, max - prevMax);

return maxDiff;

};

const nums1 = [3, 6, 9, 1];

const result1 = maximumGap(nums1);

console.log(result1);

Output: 3

const nums2 = [10];

const result2 = maximumGap(nums2);

console.log(result2);

Output: 0

5.

const containsDuplicate = function(nums) {

const uniqueSet = new Set();

for (let num of nums) {

if (uniqueSet.has(num)) {

return true;

}

uniqueSet.add(num);

}

return false;

};

const nums1 = [1, 2, 3, 1];

const result1 = containsDuplicate(nums1);

console.log(result1);

Output: true

const nums2 = [1, 2, 3, 4];

const result2 = containsDuplicate(nums2);

console.log(result2);

Output: false

const nums3 = [1, 1, 1, 3, 3, 4, 3, 2, 4, 2];

const result3 = containsDuplicate(nums3);

console.log(result3);

Output: true

**6.**

const findMinArrowShots = function(points) {

if (points.length === 0) {

return 0;

}

points.sort((a, b) => a[1] - b[1]);

let end = points[0][1];

let arrows = 1;

for (let i = 1; i < points.length; i++) {

if (points[i][0] > end) {

arrows++;

end = points[i][1];

} else {

end = Math.min(end, points[i][1]);

}

}

return arrows;

};

const points1 = [[10, 16], [2, 8], [1, 6], [7, 12]];

const result1 = findMinArrowShots(points1);

console.log(result1);

Output: 2

const points2 = [[1, 2], [3, 4], [5, 6], [7, 8]];

const result2 = findMinArrowShots(points2);

console.log(result2);

Output: 4

const points3 = [[1, 2], [2, 3], [3, 4], [4, 5]];

const result3 = findMinArrowShots(points3);

console.log(result3);

Output: 2

**7.**

const lengthOfLIS = function(nums) {

const n = nums.length;

if (n === 0) {

return 0;

}

const dp = new Array(n).fill(1);

for (let i = 1; i < n; i++) {

for (let j = 0; j < i; j++) {

if (nums[i] > nums[j]) {

dp[i] = Math.max(dp[i], dp[j] + 1);

}

}

}

return Math.max(...dp);

};

const nums1 = [10, 9, 2, 5, 3, 7, 101, 18];

const result1 = lengthOfLIS(nums1);

console.log(result1);

Output: 4

const nums2 = [0, 1, 0, 3, 2, 3];

const result2 = lengthOfLIS(nums2);

console.log(result2);

Output: 4

const nums3 = [7, 7, 7, 7, 7, 7, 7];

const result3 = lengthOfLIS(nums3);

console.log(result3);

Output: 1

**8.**

const find132pattern = function(nums) {

const n = nums.length;

const stack = [];

let second = Number.NEGATIVE\_INFINITY;

for (let i = n - 1; i >= 0; i--) {

if (nums[i] < second) {

return true;

}

while (stack.length > 0 && nums[i] > stack[stack.length - 1]) {

second = stack.pop();

}

stack.push(nums[i]);

}

return false;

};

const nums1 = [1, 2, 3, 4];

const result1 = find132pattern(nums1);

console.log(result1);

Output: false

const nums2 = [3, 1, 4, 2];

const result2 = find132pattern(nums2);

console.log(result2);

Output: true

const nums3 = [-1, 3, 2, 0];

const result3 = find132pattern(nums3);

console.log(result3);

Output: true