Naga Rajasekhar

2D Arrays - 6th Assignment

**1st Solution**

function reconstructPermutation(s) {

const n = s.length;

const perm = new Array(n + 1);

let start = 0;

let end = n;

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

if (s[i] === 'I') {

perm[i] = start;

start++;

} else {

perm[i] = end;

end--;

}

}

perm[n] = start;

return perm;

}

const s = "IDID";

const perm = reconstructPermutation(s);

console.log(perm);

**2nd Solution**

function searchMatrix(matrix, target) {

const m = matrix.length;

const n = matrix[0].length;

let left = 0;

let right = m \* n - 1;

while (left <= right) {

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

const row = Math.floor(mid / n);

const col = mid % n;

const num = matrix[row][col];

if (num === target) {

return true;

} else if (num < target) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return false;

}

const matrix = [

[1, 3, 5, 7],

[10, 11, 16, 20],

[23, 30, 34, 60],

];

const target = 3;

const found = searchMatrix(matrix, target);

console.log(found);

**3rd Solution**

function validMountainArray(arr) {

const n = arr.length;

let i = 0;

if (n < 3) {

return false;

}

while (i < n - 1 && arr[i] < arr[i + 1]) {

i++;

}

if (i === 0 || i === n - 1) {

return false;

}

while (i < n - 1 && arr[i] > arr[i + 1]) {

i++;

}

return i === n - 1;

}

const arr = [2, 1];

const isValidMountain = validMountainArray(arr);

console.log(isValidMountain);

**4th Solution**

function findMaxLength(nums) {

const countMap = {};

countMap[0] = -1;

let maxLen = 0;

let count = 0;

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

count += nums[i] === 1 ? 1 : -1;

if (countMap.hasOwnProperty(count)) {

const prevIndex = countMap[count];

const currLen = i - prevIndex;

maxLen = Math.max(maxLen, currLen);

} else {

countMap[count] = i;

}

}

return maxLen;

}

const nums = [0, 1];

const maxLength = findMaxLength(nums);

console.log(maxLength);

**5th Solution**

function minProductSum(nums1, nums2) {

nums1.sort((a, b) => a - b);

nums2.sort((a, b) => a - b);

let minProductSum = Infinity;

for (let i = 0; i < nums1.length; i++) {

const productSum = nums1[i] \* nums2[nums2.length - i - 1];

minProductSum = Math.min(minProductSum, productSum);

}

return minProductSum;

}

const nums1 = [5, 3, 4, 2];

const nums2 = [4, 2, 2, 5];

const result = minProductSum(nums1, nums2);

console.log(result);

**6th Solution**

function findOriginalArray(changed) {

if (changed.length % 2 !== 0) {

return [];

}

const original = [];

const count = new Map();

for (const num of changed) {

count.set(num, (count.get(num) || 0) + 1);

}

for (const num of changed) {

if (count.get(num) === 0) {

continue;

}

const originalNum = num / 2;

if (count.get(originalNum) > 0) {

original.push(originalNum);

count.set(originalNum, count.get(originalNum) - 1);

count.set(num, count.get(num) - 1);

} else {

return [];

}

}

return original;

}

const changed = [1, 3, 4, 2, 6, 8];

const original = findOriginalArray(changed);

console.log(original);

**7th Solution**

function generateMatrix(n) {

const matrix = Array(n).fill().map(() => Array(n).fill(0));

let num = 1;

let top = 0;

let bottom = n - 1;

let left = 0;

let right = n - 1;

while (num <= n \* n) {

for (let i = left; i <= right; i++) {

matrix[top][i] = num++;

}

top++;

for (let i = top; i <= bottom; i++) {

matrix[i][right] = num++;

}

right--;

for (let i = right; i >= left; i--) {

matrix[bottom][i] = num++;

}

bottom--;

for (let i = bottom; i >= top; i--) {

matrix[i][left] = num++;

}

left++;

}

return matrix;

}

const n = 3;

const matrix = generateMatrix(n);

console.log(matrix);

**8th Solution**

function multiplySparseMatrices(mat1, mat2) {

const m = mat1.length;

const k = mat1[0].length;

const n = mat2[0].length;

const result = Array(m).fill().map(() => Array(n).fill(0));

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

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

for (let l = 0; l < k; l++) {

if (mat1[i][l] !== 0 && mat2[l][j] !== 0) {

result[i][j] += mat1[i][l] \* mat2[l][j];

}

}

}

}

return result;

}

const mat1 = [

[1, 0, 0],

[-1, 0, 3]

];

const mat2 = [

[7, 0, 0],

[0, 0, 0],

[0, 0, 1]

];

const result = multiplySparseMatrices(mat1, mat2);

console.log(result);