**Question 1**

A permutation perm of n + 1 integers of all the integers in the range [0, n] can be represented as a string s of length n where:

* s[i] == 'I' if perm[i] < perm[i + 1], and
* s[i] == 'D' if perm[i] > perm[i + 1].

Given a string s, reconstruct the permutation perm and return it. If there are multiple valid permutations perm, return **any of them**.

**Example 1:**

**Input:** s = "IDID"

**Output:**

[0,4,1,3,2]

Program: class Solution:

def diStringMatch(self, s: str) -> List[int]:

mini=0

maxi=len(s)

l = []

for i in s:

if i =="I":

l.append(mini)

mini+=1

else:

l.append(maxi)

maxi-=1

return l+[maxi]

INPUT : s = "IDID"

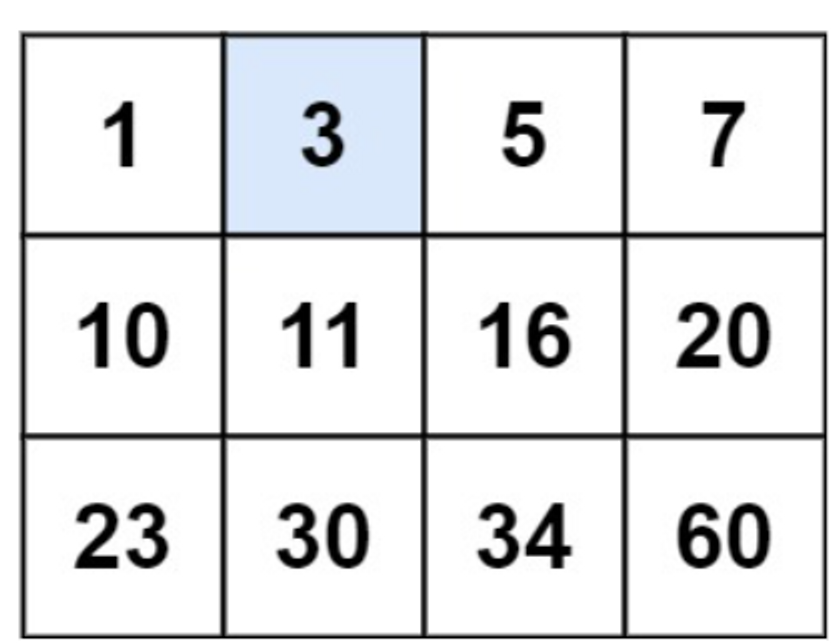
OUTPUT : [0,4,1,3,2]

**Question2: You are given an m x n integer matrix matrix with the following two properties:**

* **Each row is sorted in non-decreasing order.**
* **The first integer of each row is greater than the last integer of the previous row.**

**Given an integer target, return true *if* target *is in* matrix *or* false *otherwise*.**

**You must write a solution in O(log(m \* n)) time complexity.**

**Example: **

Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3

Output: true

**Program: class Solution:**

**def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:**

**m,n = len(matrix), len(matrix[0])**

**left, right = 0, m \* n - 1**

**while left < right:**

**mid = (left + right) >> 1**

**x, y = divmod(mid, n)**

**if matrix[x][y] >= target:**

**right = mid**

**else:**

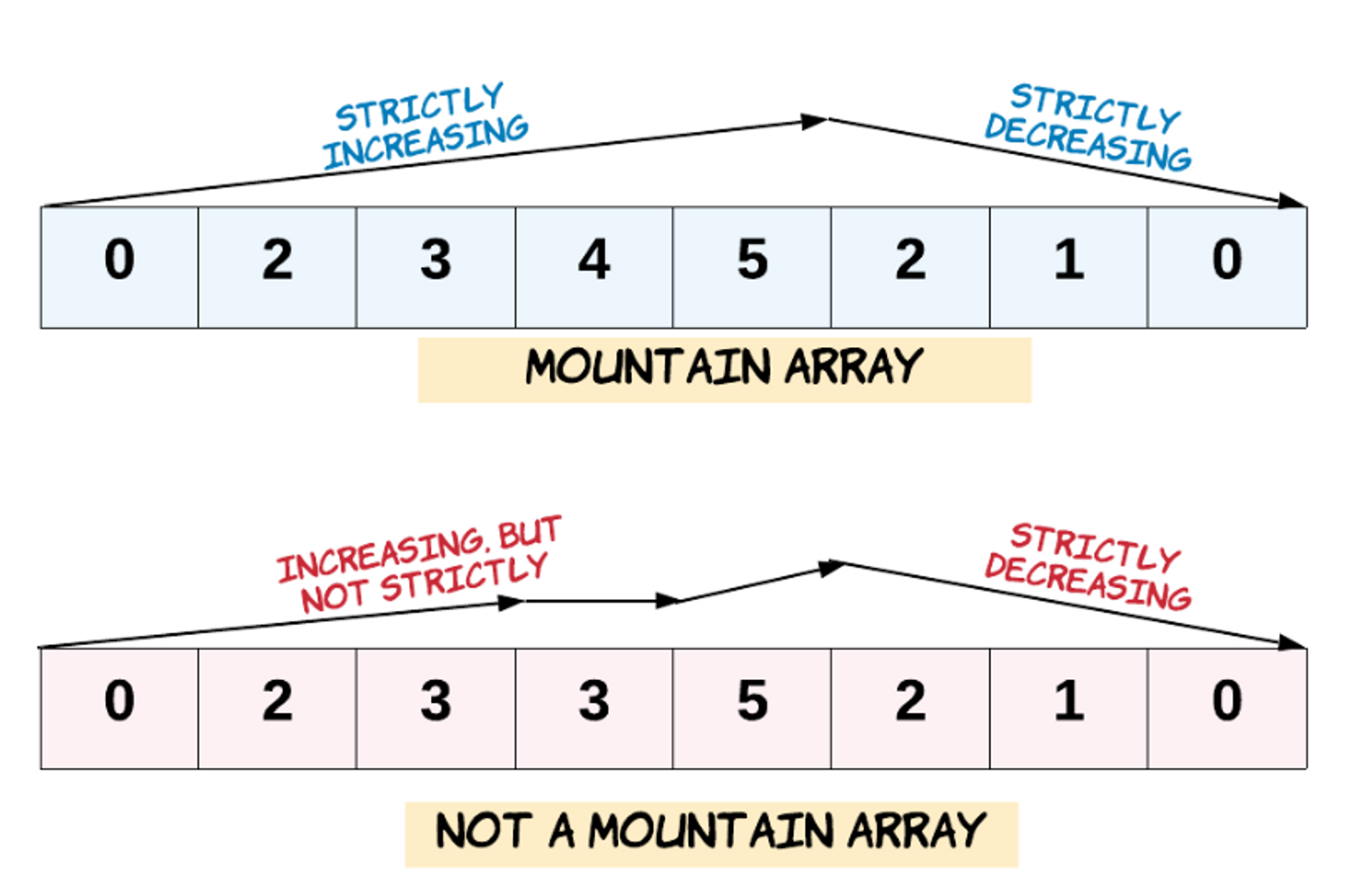
**left = mid + 1**

**return matrix[left // n][left % n ] == target**

**INPUT: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3**

**OUTPUT: true**

**Question 3 . Given an array of integers arr, return *true if and only if it is a valid mountain array*.Recall that arr is a mountain array if and only if:**

* **arr.length >= 3**
* **There exists some i with 0 < i < arr.length - 1 such that:**
  + **arr[0] < arr[1] < ... < arr[i - 1] < arr[i]**
  + **arr[i] > arr[i + 1] > ... > arr[arr.length - 1]**
  + ****

**Example 1: Input: arr = [2,1]**

**Output: false**

**Program: class Solution:**

**def validMountainArray(self, arr: List[int]) -> bool:**

**i = 1**

**while i < len(arr) and arr[i] > arr[i-1]:**

**i += 1**

**if i == 1 or i == len(arr):**

**return False**

**while i < len(arr) and arr[i] < arr[i-1]:**

**i += 1**

**return i == len(arr)**

**INPUT: arr = [2,1]**

**OUTPUT: False**

**Question 4. Given a binary array nums, return *the maximum length of a contiguous subarray with an equal number of* 0 *and* 1.**

**Example 1:**

**Input: nums = [0,1]**

**Output: 2**

**Explanation:**

**[0, 1] is the longest contiguous subarray with an equal number of 0 and 1.**

**Program:   
class Solution:**

**def findMaxLength(self, nums: List[int]) -> int:**

**sum\_map = {0:-1}**

**max\_len = 0**

**distance = 0**

**partial\_sum = 0**

**for i, num in enumerate(nums):**

**if num == 1:**

**partial\_sum += 1**

**else:**

**partial\_sum -= 1**

**if partial\_sum in sum\_map:**

**distance = i - sum\_map[partial\_sum]**

**max\_len = max(max\_len, distance)**

**else:**

**sum\_map[partial\_sum] = i**

**return max\_len**

**INPUT: nums = [0,1]   
OUTPUT : 2**

**Question 5**

**The product sum of two equal-length arrays a and b is equal to the sum of a[i] \* b[i] for all 0 <= i < a.length (0-indexed).**

* **For example, if a = [1,2,3,4] and b = [5,2,3,1], the product sum would be 1*5 + 2*2 + 3*3 + 4*1 = 22.**

**Given two arrays nums1 and nums2 of length n, return *the minimum product sum if you are allowed to rearrange the order of the elements in* nums1.**

**Example 1:**

**Input: nums1 = [5,3,4,2], nums2 = [4,2,2,5]**

**Output: 40**

**Explanation:**

**We can rearrange nums1 to become [3,5,4,2]. The product sum of [3,5,4,2] and [4,2,2,5] is 3*4 + 5*2 + 4*2 + 2*5 = 40.**

**Program: class Solution:**

**def minProductSum(self, nums1: List[int], nums2: List[int]) -> int:**

**nums1.sort()**

**nums2.sort()**

**n, res = len(nums1), 0**

**for i in range(n):**

**res += nums1[i] \* nums2[n - i - 1]**

**return res**

**Input: nums1 = [5,3,4,2], nums2 = [4,2,2,5]**

**Output: 40**

**Question 6. An integer array original is transformed into a doubled array changed by appending twice the value of every element in original, and then randomly shuffling the resulting array.**

**Given an array changed, return original *if* changed *is a doubled array. If* changed *is not a doubled array, return an empty array. The elements in* original *may be returned in any order*.**

**Example 1:**

**Input: changed = [1,3,4,2,6,8]**

**Output: [1,3,4] Explanation: One possible original array could be [1,3,4]:**

* **Twice the value of 1 is 1 \* 2 = 2.**
* **Twice the value of 3 is 3 \* 2 = 6.**
* **Twice the value of 4 is 4 \* 2 = 8.**

**Other original arrays could be [4,3,1] or [3,1,4].**

**Program: class Solution:**

**def findOriginalArray(self, changed: List[int]) -> List[int]:**

**ans = []**

**q = collections.deque()**

**for num in sorted(changed):**

**if q and num == q[0]:**

**q.popleft()**

**Else:**

**q.append(num \* 2)**

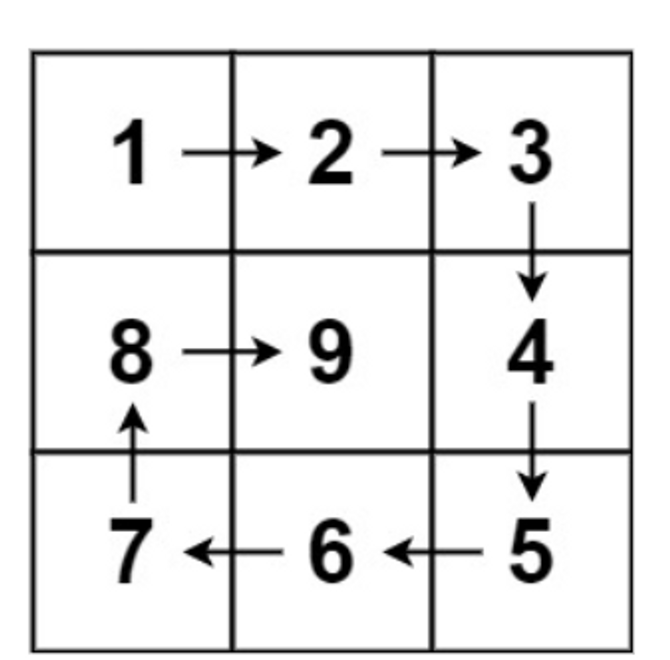
**ans.append(num)**

**return [] if q else ans**

**INPUT: changed = [1,3,4,2,6,8], OUTPUT : [1,3,4]**

**Question 7. Given a positive integer n, generate an n x n matrix filled with elements from 1 to n2 in spiral order.**

**Example 1:**

****

**Input: n = 3**

**Output: [[1,2,3],[8,9,4],[7,6,5]]**

**Program: class Solution:**

**def generateMatrix(self, n: int) -> List[List[int]]:**

**ans = [[0] \* n for \_ in range(n)]**

**count = 1**

**for min in range(n // 2):**

**max = n - min - 1**

**for i in range(min, max):**

**ans[min][i] = count**

**count += 1**

**for i in range(min, max):**

**ans[i][max] = count**

**count += 1**

**for i in range(max, min, -1):**

**ans[max][i] = count**

**count += 1**

**for i in range(max, min, -1):**

**ans[i][min] = count**

**count +=**

**if n & 1:**

**ans[n // 2][n // 2] = count**

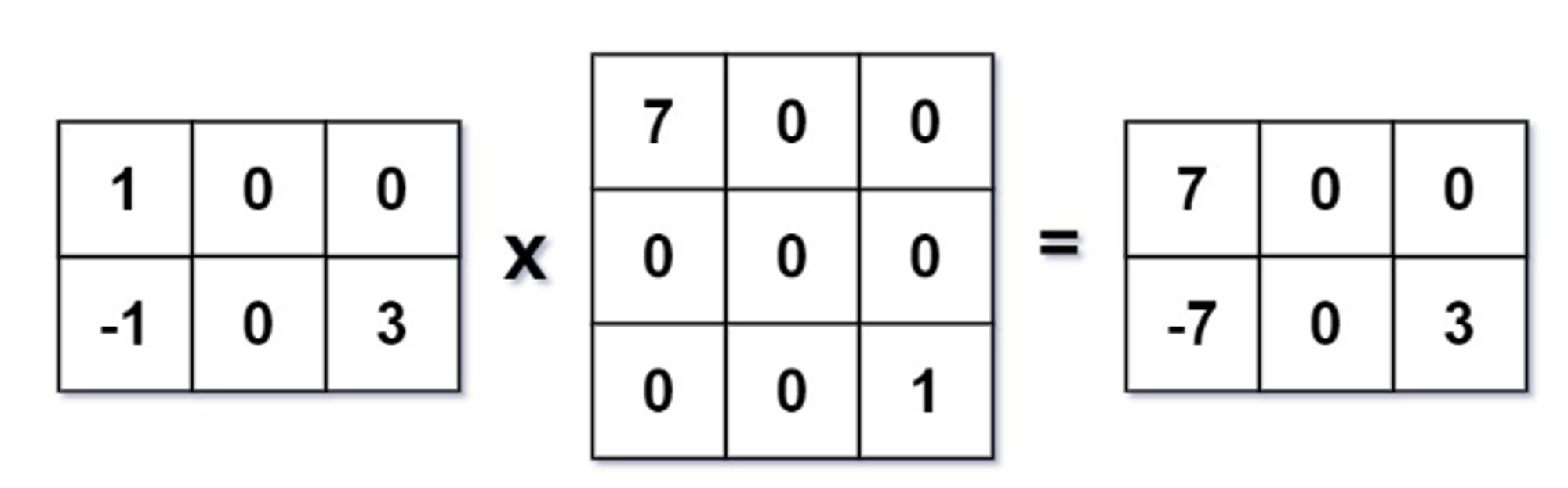
**return ans**

**INPUT : n = 3**

**OUTPUT - [[1,2,3],[8,9,4],[7,6,5]]**

**Question 8.Given two** [**sparse matrices**](https://en.wikipedia.org/wiki/Sparse_matrix) **mat1 of size m x k and mat2 of size k x n, return the result of mat1 x mat2. You may assume that multiplication is always possible.**

**Example 1:**

****

**Input: mat1 = [[1,0,0],[-1,0,3]], mat2 = [[7,0,0],[0,0,0],[0,0,1]]**

**Output: [[7,0,0],[-7,0,3]]**

**class Solution:**

**def multiply(self, mat1: List[List[int]], mat2: List[List[int]]) -> List[List[int]]:**

**r1, c1, c2 = len(mat1), len(mat1[0]), len(mat2[0])**

**res = [[0] \* c2 for \_ in range(r1)]**

**mp = defaultdict(list)**

**for i in range(r1):**

**for j in range(c1):**

**if mat1[i][j] != 0:**

**mp[i].append(j)**

**for i in range(r1):**

**for j in range(c2):**

**for k in mp[i]:**

**res[i][j] += mat1[i][k] \* mat2[k][j]**

**return res**

**Input: mat1 = [[1,0,0],[-1,0,3]], mat2 = [[7,0,0],[0,0,0],[0,0,1]]**

**Output: [[7,0,0],[-7,0,3]]**