Q1. 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:

Class Solution:

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

i, n = 0, len(s)

ans = []

for c in s:

if c == 'I':

ans.append(i)

i+=1

else:

ans.append(n)

n-=1

ans.append(i)

return ans

Q2. You are given an m x n integer matrix matrix with the following two properties: Do binary search  
class Solution:

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

row,col=len(matrix),len(matrix[0])

left,right=0,row\*col-1

while left<=right:

mid=(left+right)//2

num=matrix[mid//col][mid%col]

if num==target:

return True

if num>target:

right=mid-1

else:

left=mid+1

return False

Q3. 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:

class Solution:

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

strictly\_increasing = strictly\_decreasing = False

for i in range(1, len(arr)):

if arr[i] > arr[i - 1]:

if strictly\_decreasing:

return False

strictly\_increasing = True

elif arr[i] < arr[i - 1]:

if not strictly\_increasing:

return False

strictly\_decreasing = True

else:

return False

return True if strictly\_increasing and strictly\_decreasing else False

Q4. Given a binary array nums, return *the maximum length of a contiguous subarray with an equal number of* 0 *and* 1.

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

count = 0

map = { 0: -1}

max\_length = 0

for i, number in enumerate( nums ):

if number:

count += 1

else:

count -= 1

if count in map:

max\_length = max( max\_length, ( i - map[count] ) )

else:

map[ count ] = i

return max\_length

Q5. 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.

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

Q6. 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.

class Solution:

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

if len(changed) % 2 == 1:

return []

data = Counter(changed)

result = []

for k in sorted(data):

if data[k] < 0:

return []

value = k \* 2

while data[k] > 0:

if data[value] == 0:

return []

result.append(k)

data[k] -= 1

data[val

Q7 Given a positive integer n, generate an n x n matrix filled with elements from 1 to n2 in spiral order.

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

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

top = 0

bottom = n - 1

left = 0

right = n - 1

num = 1

while top <= bottom and left <= right:

for i in range(left, right + 1):

matrix[top][i] = num

num += 1

top += 1

for i in range(top, bottom + 1):

matrix[i][right] = num

num += 1

right -= 1

if top <= bottom:

for i in range(right, left - 1, -1):

matrix[bottom][i] = num

num += 1

bottom -= 1

if left <= right:

for i in range(bottom, top - 1, -1):

matrix[i][left] = num

num += 1

left += 1

return matrix

Q8. 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.

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)]

for i in range(r1):

for j in range(c2):

for k in range(c1):

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

return res