Q1

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

class Solution:

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

for row in matrix:

if target in row:

return True

return False

Q3

class Solution:

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

if len(arr)<=2 or max(arr)==arr[0] or max(arr)==arr[len(arr)-1]:

return False

f=True

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

if f and arr[i]>=arr[i+1]:

f=False

if not f and arr[i]<=arr[i+1]:

return False

return True

Q4

class Solution:

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

m,c=0,0

d={0:-1}

for i in range(len(nums)):

if nums[i]==0:

c-=1

else:

c+=1

if c in d:

m=max(m,i-d[c])

else:

d[c]=i

return m

Q5

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

class Solution:

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

        changed.sort()

        stk,res=deque([]),[]

        for i in changed:

            if stk and stk[0]\*2==i:

                b=stk.popleft()

                res.append(b)

            else:

                stk.append(i)

        return res if not stk else []

Q7

def spiralOrder(matrix):

result = []

rows = len(matrix)

columns = len(matrix[0])

up = 0

left = 0

right = columns - 1

down = rows - 1

while len(result) < rows \* columns:

# Traverse from left to right.

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

result.append(matrix[up][col])

# Traverse downwards.

for row in range(up + 1, down + 1):

result.append(matrix[row][right])

# Make sure we are now on a different row.

if up != down:

# Traverse from right to left.

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

result.append(matrix[down][col])

# Make sure we are now on a different column.

if left != right:

# Traverse upwards.

for row in range(down - 1, up, -1):

result.append(matrix[row][left])

left += 1

right -= 1

up += 1

down -= 1

return result

Q8

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