SABUDH

1. **Given an array arr[] containing integers and an integer k, your task is to find the length of the longest subarray where the sum of its elements is equal to the given value k. If there is no subarray with sum equal to k, return 0.**

   Examples:

   Input: arr[] = [10, 5, 2, 7, 1, -10], k = 15
   Output: 6
   Explanation: Subarrays with sum = 15 are [5, 2, 7, 1], [10, 5] and [10, 5, 2, 7, 1, -10]. The length of the longest subarray with a sum of 15 is 6.

   Input: arr[] = [-5, 8, -14, 2, 4, 12], k = -5
   Output: 5
   Explanation: Only subarray with sum = 15 is [-5, 8, -14, 2, 4] of length 5.

   Input: arr[] = [10, -10, 20, 30], k = 5
   Output: 0
   Explanation: No subarray with sum = 5 is present in arr[].

   Constraints:

   - 1 ≤ arr.size() ≤ 10^5
   - -10^4 ≤ arr[i] ≤ 10^4
   - -10^9 ≤ k ≤ 10^9

2. **Given an integer array, find a triplet having the maximum product.**

   Test Case : 1
   Input: { -4, 1, -8, 9, 6 }
   Output: The triplet having the maximum product is (-8, -4, 9)

   Test Case : 2

Input: { 1, 7, 2, -2, 5 }
Output: The triplet having the maximum product is (7, 5, 2)

3. Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.

   Example 1:

   Input: nums = [-1,0,1,2,-1,-4]
   Output: [[-1,-1,2],[-1,0,1]]

   Explanation:
   nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.
   nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.
   nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.
   The distinct triplets are [-1,0,1] and [-1,-1,2].
   Notice that the order of the output and the order of the triplets does not matter.

   Example 2:
   Input: nums = [0,1,1]
   Output: []
   Explanation: The only possible triplet does not sum up to 0.

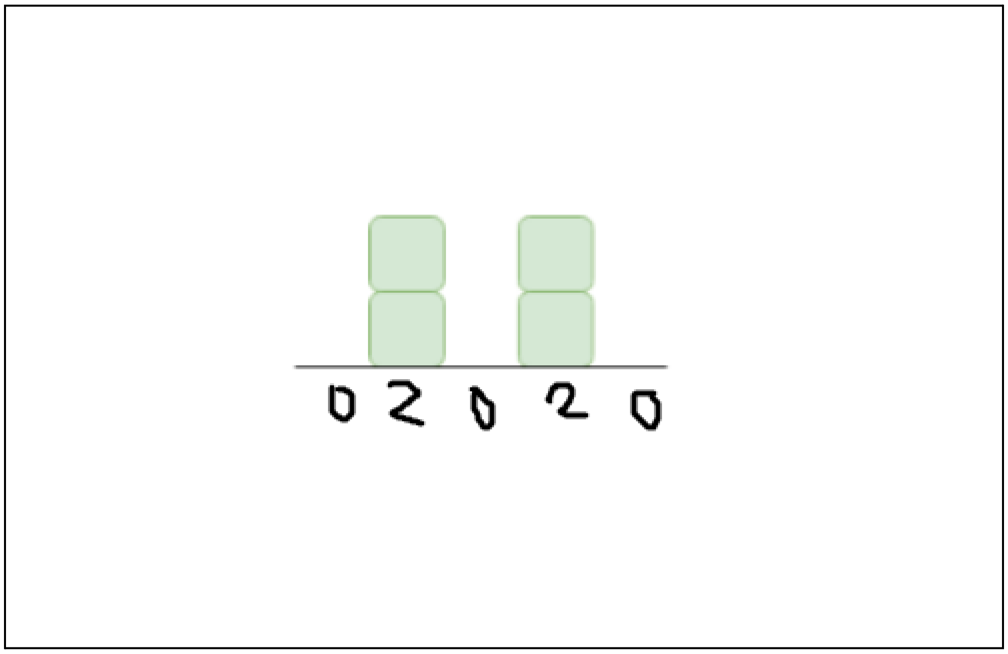
   Example 3:
   Input: nums = [0,0,0]
   Output: [[0,0,0]]
   Explanation: The only possible triplet sums up to 0.

   Constraints:
   - 3 <= nums.length <= 3000
   - -10^5<= nums[i] <= 10^5

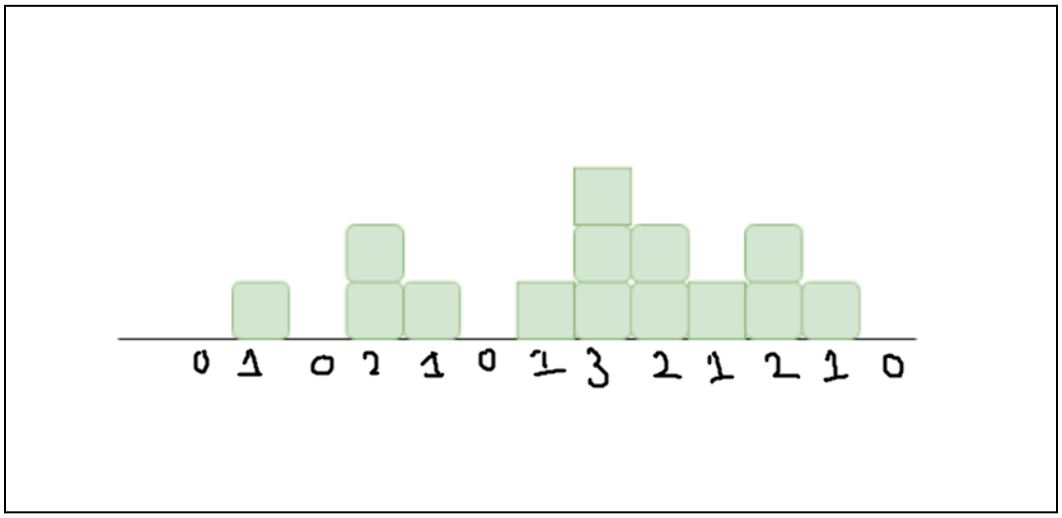
4. Given an array of N non-negative integers arr[] representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after rain.

   Test Case :1
   Input: arr[] = {0,2, 0, 2,0}

Output: 2



Test Case : 2
Input: arr[] = {0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1, 0}
Output: 6



5. A permutation of an array of integers is an arrangement of its members into a sequence or linear order.

    - For example, for arr = [1,2,3], the following are all the permutations of arr: [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].

    The next permutation of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the next permutation of that array is the permutation that follows it in the sorted container. If such an arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

    - For example, the next permutation of arr = [1,2,3] is [1,3,2].
    - Similarly, the next permutation of arr = [2,3,1] is [3,1,2].
    - While the next permutation of arr = [3,2,1] is [1,2,3] because [3,2,1] does not have a lexicographical larger rearrangement

    Given an array of integers nums, *find the next permutation of* nums.
    The replacement must be in place and use only constant extra memory.

    Example 1:
    Input: nums = [1,2,3]
    Output: [1,3,2]

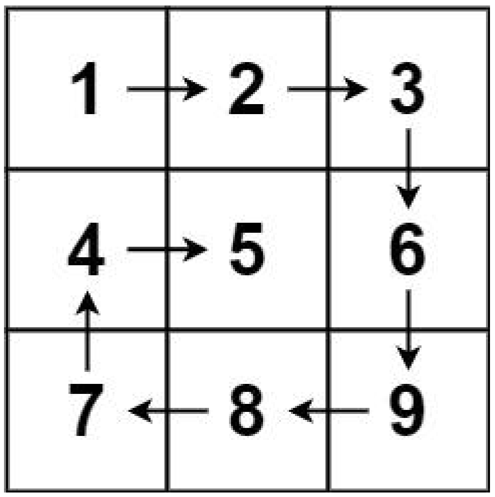
    Example 2:
    Input: nums = [3,2,1]
    Output: [1,2,3]

    Example 3:
    Input: nums = [1,1,5]
    Output: [1,5,1]

    Constraints:
    1 <= nums.length <= 100
    0 <= nums[i] <= 100

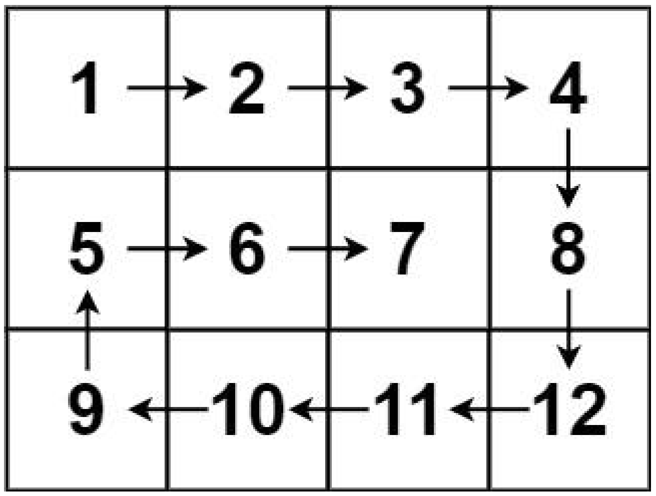
6. Given an m x n matrix, return *all elements of the* matrix *in spiral order*.

Example 1:



Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
Output: [1,2,3,6,9,8,7,4,5]

Example 2:



Input: matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
Output: [1,2,3,4,8,12,11,10,9,5,6,7]

Constraints:

- m == matrix.length
- n == matrix[i].length
- 1 <= m, n <= 10
- -100 <= matrix[i][j] <= 100

7. You are given two positive integer arrays spells and potions, of length n and m, respectively, where spells[i] represents the strength of the ith spell and potions[j] represents the strength of the jth potion.

   You are also given an integer success. A spell and potion pair is considered successful if the product of their strengths is at least success.

   Return *an integer array* pairs *of length* n, *where* pairs[i] *is the number of potions that will form a successful pair with the* ith *spell*

   Example 1:
   Input: spells = [5,1,3], potions = [1,2,3,4,5], success = 7
   Output: [4,0,3]

   Explanation:
   - 0th spell: 5 * [1,2,3,4,5] = [5,10,15,20,25]. 4 pairs are successful.
   - 1st spell: 1 * [1,2,3,4,5] = [1,2,3,4,5]. 0 pairs are successful.
   - 2nd spell: 3 * [1,2,3,4,5] = [3,6,9,12,15]. 3 pairs are successful.
   Thus, [4,0,3] is returned.

   Example 2:
   Input: spells = [3,1,2], potions = [8,5,8], success = 16
   Output: [2,0,2]

   Explanation:
   - 0th spell: 3 * [8,5,8] = [24,15,24]. 2 pairs are successful.
   - 1st spell: 1 * [8,5,8] = [8,5,8]. 0 pairs are successful.
   - 2nd spell: 2 * [8,5,8] = [16,10,16]. 2 pairs are successful.
   Thus, [2,0,2] is returned.

   Constraints:
   - n == spells.length

SABUDH

- m == potions.length
- 1 <= n, m <= 105
- 1 <= spells[i], potions[i] <= 105
- 1 <= success <= 1010

All the Best !!!