Self-Practice No. : **3**

Topics Covered : **Basic Math, Control Flow, Arrays, Functions, Bit Manipulation**

Date : **10-05-2024**

Solve the following problems

Q No.	Question Detail	Level
1	**Friendly pair** **Problem statement :** Vino wants to **check if the given two numbers are friendly pair or not.** Friendly pair(Amicable numbers) are two different numbers related in a way such that the Ratio's sum of the proper divisors divided by number itself for each is same. Proper divisors of a number are all the positive integers that divide the number evenly, excluding the number itself. The sum of proper divisors of a number is the total sum of all its proper divisors. For example, the sum of proper divisors of 6 is 1 + 2 + 3 = 6. **Input :** n = 6, m = 28 **Output** : Yes **Explanation:** Divisor of 6 are 1, 2, 3, 6. Divisor of 28 are 1, 2, 4, 7, 14, 28. Sum of divisor of 6 and 28 are 12 and 56 respectively. Abundancy index of 6 and 28 are 2. So they are friendly pair. **Input :** n = 18, m = 26 **Output :** No	Hard

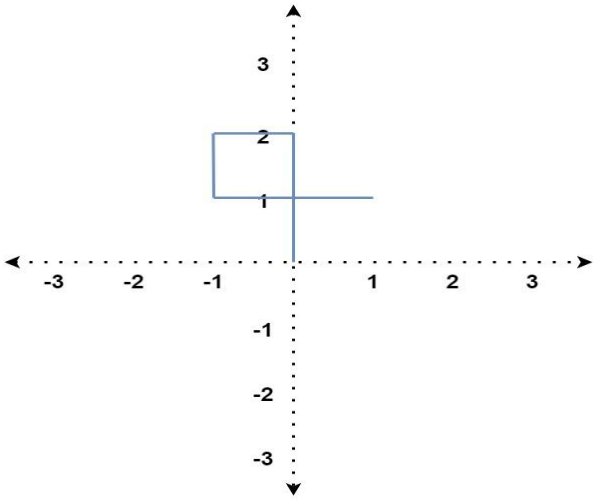
	Constraints: • $1<=N1,N2<=10^8$	
2	**K pattern** **Problem statement :** Print a pattern of stars in the shape of the letter 'K'. The pattern should consist of asterisks (*) forming the uppercase and lowercase letter 'K'. The program should take the height of the letter 'K' as input and print the corresponding pattern. **Input 1:** height = 5 **Output** ***** **** *** ** * ** *** **** ***** **Input2:** height = 3 **Output2:** *** ** * ** ***	Hard

	Constraints: • 1 <= height <= 26	
3	**Count the number of subarrays** **Problem Statement :** Given an array A[] of N integers and a range(L, R). The task is to find the number of subarrays having sum in the range L to R (inclusive). **Example 1:** **Input:** N = 3, L = 3, R = 8 A[] = {1, 4, 6} **Output:** 3 **Explanation:** The subarrays are [1,4], [4] and [6] **Example 2:** **Input:** N = 4, L = 4, R = 13 A[] = {2, 3, 5, 8} **Output:** 6 **Explanation:** The subarrays are [2,3], [2,3,5], [3,5],[5], [5,8] and [8] **Constraints:** 1 ≤ N ≤ 10^6 1 ≤ A[] ≤ 10^9 1 ≤ L ≤ R ≤ 10^15	Hard
4	**Self Crossing** **Problem Statement :** You are given an array of integers distance.	Hard

You start at the point (0, 0) on an X-Y plane, and you move distance[0] meters to the north, then distance[1] meters to the west, distance[2] meters to the south, distance[3] meters to the east, and so on. In other words, after each move, your direction changes counter-clockwise.

Return true if your path crosses itself or false if it does not.

Example 1:

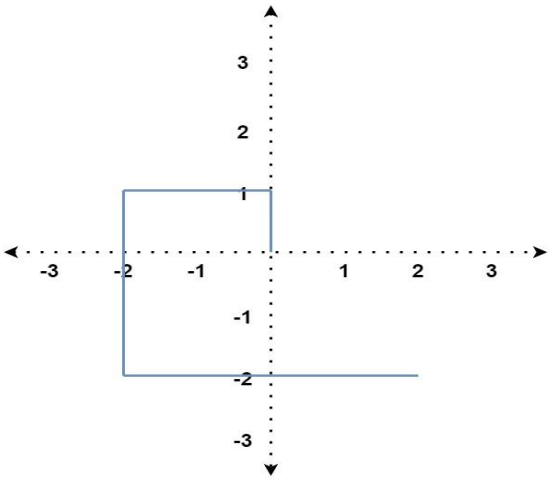


Input: distance = [2,1,1,2]

Output: true

Explanation: The path crosses itself at the point (0, 1).

Example 2:



	Input: distance = [1,2,3,4] **Output:** false **Explanation:** The path does not cross itself at any point. **Constraints:** • 1 <= distance.length <= 10^5 • 1 <= distance[i] <= 10^5	
5	**Maximum Score** **Problem statement :** Tom is playing a board game in which two lists of distinct numbers 'A' and 'B' arranged in a non-descending order are given. The game has certain rules and the player has to pick some numbers from the given list and the score is the sum of unique picked numbers. The rules are: 1. Choose any list 'A' or 'B'. 2. Traverse from left to right. 3. After picking a number, if the picked number is present in both the arrays, you are allowed to traverse to the other array. You are given arrays,'A' and 'B' of size 'N' and 'M' respectively. Your task is to find the maximum score Tom can achieve. Since the answer can be very large, print answer % (10^9 + 7). For **Example** If the given array are 'A' = [1,3,5,7,9] and 'B' = [3,5,100]".The maximum score can be achieved is 109[1+3+5+100]. Detailed explanation (Input/output format, Notes, Images)	Hard

Sample Input 1:

2

5 3

1 3 5 7 9

3 5 100

4 3

1 2 3 5

1 3 10

Sample Output 1:

109

16

Explanation of sample input 1:

For the first test case,

The numbers Tom will be pick are [1,3,5,100].The sum of unique numbers are 109.

Hence, the answer is 109.

For the second test case:

The numbers Tom will be pick are [1,2,3,3,10].The sum of unique numbers are 16(1+2+3+10).

Hence, the answer is 16.

Sample Input 2:

2

5 4

2 4 5 8 10

	4 6 8 9 5 5 1 2 3 4 5 6 7 8 9 10 **Sample Output 2:** 30 40 **Constraints:** 1 <= T <= 10 1 <= N,M <= 10^4. 1 <= A[i], B[i] <= 10^6.	
6	**Minimize OR of Remaining Elements Using Operations** **Problem Statement :** You are given a 0-indexed integer arraynums and an integer k. In one operation, you can pick any index i of nums such that 0 <= i < nums.length - 1 and replace nums[i] and nums[i + 1] with a single occurrence of nums[i] & nums[i + 1], where & represents the bitwise AND operator. Return the minimum possible value of the bitwise OR of the remaining elements of nums after applying at most k operations. **Example 1:** **Input:** nums = [3,5,3,2,7], k = 2 **Output**: 3 **Explanation:** Let's do the following operations:	Hard

1. Replace nums[0] and nums[1] with (nums[0] & nums[1]) so that nums becomes equal to [1,3,2,7].
2. Replace nums[2] and nums[3] with (nums[2] & nums[3]) so that nums becomes equal to [1,3,2].

The bitwise-or of the final array is 3.
It can be shown that 3 is the minimum possible value of the bitwise OR of the remaining elements of nums after applying at most k operations.

Example 2:

Input: nums = [7,3,15,14,2,8], k = 4
Output: 2
Explanation: Let's do the following operations:

1. Replace nums[0] and nums[1] with (nums[0] & nums[1]) so that nums becomes equal to [3,15,14,2,8].
2. Replace nums[0] and nums[1] with (nums[0] & nums[1]) so that nums becomes equal to [3,14,2,8].
3. Replace nums[0] and nums[1] with (nums[0] & nums[1]) so that nums becomes equal to [2,2,8].
4. Replace nums[1] and nums[2] with (nums[1] & nums[2]) so that nums becomes equal to [2,0].

The bitwise-or of the final array is 2.
It can be shown that 2 is the minimum possible value of the bitwise OR of the remaining elements of nums after applying at most k operations.

Constraints:

- 1 <= nums.length <= 10^5
- 0 <= nums[i] < 230

	• 0 <= k < nums.length	
7	**Cake Distribution Problem** **Problem Statement :** Geek is organizing a birthday party, so his friends brought a cake for him. The cake consists of N chunks, whose individual sweetness is represented by the sweetness array. Now at the time of distribution, Geek cuts the cake into K + 1 pieces to distribute among his K friends. One piece he took for himself. Each piece consists of some consecutive chunks. He is very kind, so he left the piece with minimum sweetness for him. You need to find the maximum sweetness that the Geek can get if he distributes the cake optimally. **Example 1:** **Input:** N = 6, K = 2 sweetness[] = {6, 3, 2, 8, 7, 5} **Output:** 9 **Explanation:** Geek can divide the cake to [6, 3], [2, 8], [7, 5] with sweetness level 9, 10 and 12 respectively. **Example 2:** **Input:** N = 7, K = 3 sweetness[] = {1, 2, 4, 7, 3, 6, 9} **Output:**	Hard

	7 **Explanation:** Geek can divide the cake to [1, 2, 4], [7], [3, 6], [9] with sweetness level 7, 7, 9 and 9 respectively. **Constraints:** 1 <= N <= 10^5 0 <= K < N 1 <= sweetness[i] <= 10^9	
8	**Minimize Manhattan Distances** **Problem Statement :** You are given a array points representing integer coordinates of some points on a 2D plane, where points[i] = [xi, yi]. The distance between two points is defined as their Manhattan distance. Return the minimum possible value for maximum distance between any two points by removing exactly one point. **Example 1:** **Input:** points = [[3,10],[5,15],[10,2],[4,4]] **Output**: 12 **Explanation:** The maximum distance after removing each point is the following: • After removing the 0th point the maximum distance is between points (5, 15) and (10, 2), which is |5 - 10| + |15 - 2| = 18. • After removing the 1st point the maximum distance is between points (3,	Hard

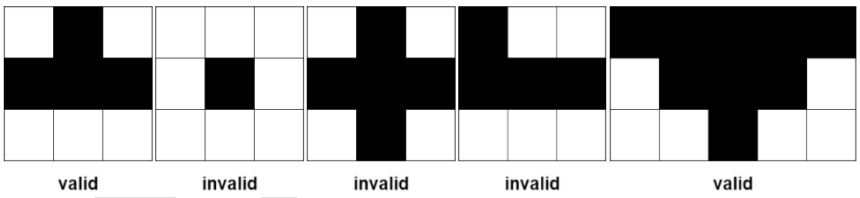
	10) and (10, 2), which is |3 - 10| + |10 - 2| = 15. • After removing the 2nd point the maximum distance is between points (5, 15) and (4, 4), which is |5 - 4| + |15 - 4| = 12. • After removing the 3rd point the maximum distance is between points (5, 15) and (10, 2), which is |5 - 10| + |15 - 2| = 18. 12 is the minimum possible maximum distance between any two points after removing exactly one point. **Example 2:** **Input:** points = [[1,1],[1,1],[1,1]] **Output:** 0 **Explanation:** Removing any of the points results in the maximum distance between any two points of 0. **Constraints:** 3 <= points.length <= 10^5 points[i].length == 2 1 <= points[i][0], points[i][1] <= 10^8	
9	**Count Fertile Pyramids in a Land** **Problem Statement :** A farmer has a rectangular grid of land with m rows and n columns that can be divided into unit cells. Each cell is either fertile (represented by a 1) or barren (represented by a 0). All cells outside the grid are considered barren. A pyramidal plot of land can be defined as a set of cells with the following criteria: 1. The number of cells in the set has to be greater than 1 and all cells must be fertile.	Hard

2. The apex of a pyramid is the topmost cell of the pyramid. The height of a pyramid is the number of rows it covers. Let (r, c) be the apex of the pyramid, and its height be h. Then, the plot comprises of cells (i, j) where r <= i <= r + h - 1 and c - (i - r) <= j <= c + (i - r).

An inverse pyramidal plot of land can be defined as a set of cells with similar criteria:

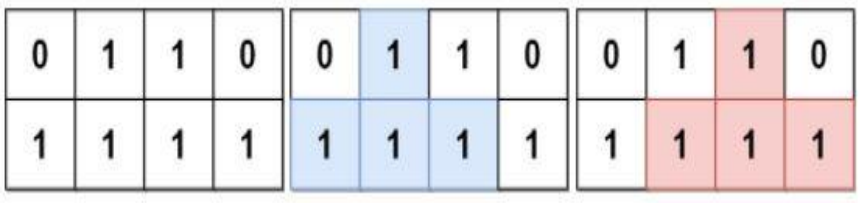
1. The number of cells in the set has to be greater than 1 and all cells must be fertile.
2. The apex of an inverse pyramid is the bottommost cell of the inverse pyramid. The height of an inverse pyramid is the number of rows it covers. Let (r, c) be the apex of the pyramid, and its height be h. Then, the plot comprises of cells (i, j) where r - h + 1 <= i <= r and c - (r - i) <= j <= c + (r - i).

Some examples of valid and invalid pyramidal (and inverse pyramidal) plots are shown below. Black cells indicate fertile cells.



Given a 0-indexed m x n binary matrix grid representing the farmland, return the total number of pyramidal and inverse pyramidal plots that can be found in grid.

Example 1:



Input: grid = [[0,1,1,0],[1,1,1,1]]
Output: 2
Explanation: The 2 possible pyramidal plots are shown in blue and red respectively.
There are no inverse pyramidal plots in this grid.
Hence total number of pyramidal and inverse pyramidal plots is 2 + 0 = 2.

Example 2:

Input: grid = [[1,1,1],[1,1,1]]
Output: 2
Explanation: The pyramidal plot is shown in blue, and the inverse pyramidal plot is shown in red.
Hence the total number of plots is 1 + 1 = 2.

Constraints:

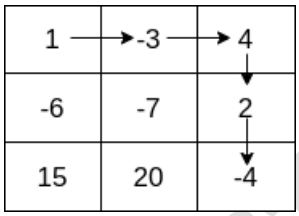
m == grid.length
n == grid[i].length
1 <= m, n <= 1000
1 <= m * n <= 10^5
grid[i][j] is either 0 or 1.

10	**Minimum HP** **Problem statement :** You are playing a board game where you die when your hp (health points) becomes 0 or negative. Now you start with some initial hp, and	Hard

start moving in the board starting from the top-left corner. You have to reach the bottom right corner of the board by only moving right and down. Whenever you arrive on a cell in the board your hp may increase or decrease by the amount written on the cell. If the cell has a positive value it means your hp increases, if it has a negative value it means your hp decreases. You need to return the minimum initial health required so that you can reach the rightmost corner.

For **Example:**

In the example below, if we follow the path shown in the figure, then the minimum initial hp required will be 3.



Detailed explanation (Input/output format, Notes, Images)

Sample Input 1 :

2

3 3

1 -3 4

-6 -7 2

15 20 -4

1 1

-4

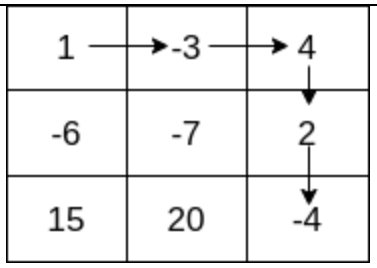
Sample Output 1 :

3

5

Explanation For Sample Input 1 :

In test case 1:



If we follow the path shown in the figure then the minimum initial hp required will be 3.

In test case 2:

There is only 1 element in the board, if where minimum 5 initial hp is required. Because 5 + (-4) = 1, if we have anything lower than 5, we will have 0 or negative hp at the end.

Sample Input 2 :

2
3 4
-1 -7 3 -10
10 -13 3 -14
-13 6 -6 9
5 5
3 -1 2 14 19
-8 -5 5 -15 -19
-9 -10 -5 -18 -8
6 20 13 1 -9
0 13 5 6 -1

Sample Output 2 :

5
1

Constraints :

1 <= T <= 5
1 <= N, M <= 1000
-3000 <= board[i][j] <= 3000

	Where 'board[i][j]' represent the value of board at 'jth' column of 'ith' row.	