Exercise 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	**Seating Arrangement** **Problem statement :** You are required to create a program that determines the various arrangements in which a group of 'n' individuals can be seated in 'r' seats within a classroom or theater setting. The program should calculate and display these possible arrangements. **Input 1 :** n=4, r=2 **Output 1:** Possible arrangements : 6 **Input 2:** n=5 , r=3 **Output 2:** Possible arrangements: 10 **Constraints:** $1 \leq n, r \leq 10$	Hard
2	**Reaching Points** **Problem statement :** Given four integers sx, sy, tx, and ty, return true if it is possible to convert the point (sx, sy) to the point (tx, ty) through some operations, or false otherwise. The allowed operation on some point (x, y) is to convert it to either (x, x + y) or (x + y, y). **Example 1:**	Hard

	Input: sx = 1, sy = 1, tx = 3, ty = 5 **Output:** true **Explanation:** One series of moves that transforms the starting point to the target is: (1, 1) -> (1, 2) (1, 2) -> (3, 2) (3, 2) -> (3, 5) **Example 2:** **Input:** sx = 1, sy = 1, tx = 2, ty = 2 **Output:** false **Example 3:** **Input:** sx = 1, sy = 1, tx = 1, ty = 1 **Output:** true **Constraints:** 1 <= sx, sy, tx, ty <= 10^9	
3	**Ways To Express** **Problem statement :** You are given the number 'N'. The task is to find the number of ways to represent 'N' as a sum of two or more consecutive natural numbers. **Example:** N = 9 '9' can be represented as: 2 + 3 + 4 = 9 4 + 5 = 9 The number of ways to represent '9' as a sum of consecutive natural numbers is '2'. So, the answer is '2'. **Note:** 1. The numbers used to represent 'N' should be natural numbers (Integers greater than equal to 1). **Sample Input 1:** 2	Hard

	21 15 **Sample Output 1:** 3 3 **Explanation Of Sample Input 1:** **Test Case 1:** '21' can be represented as: 1 + 2 + 3 + 4 + 5 + 6 = 21 6 + 7 + 8 = 21 10 + 11 = 21 The number of ways to represent '21' as a sum of consecutive natural numbers is '3'. So, the answer is '3'. **Test Case 2:** '15' can be represented as: 1 + 2 + 3 + 4 + 5 = 15 4 + 5 + 6 = 15 7 + 8 = 15 The number of ways to represent '15' as a sum of consecutive natural numbers is '3'. So, the answer is '3'. **Sample Input 2:** 2 18 16 **Sample Output 2:** 2 0 **Constraints:** 1 <= T <= 1000 3 <= N <= 10 ^ 5 Where 'T' is the number of test cases, and 'N' is the given number.	
4	**Butterfly star pattern** **Problem statement**: Create a program to print a butterfly star pattern consisting of stars in a symmetrical butterfly shape. The pattern should be	Hard

	printed such that the stars form wings on both sides of the centerline. The number of stars in each wing should decrease towards the centerline.
	
	Input 1 : rows = 5
	Output 1:
	`*  *`
	`**  **`
	`***  ***`
	`****  ****`
	`**********`
	`****  ****`
	`***  ***`
	`**  **`
	`*  *`
	
	Input 2: rows =7
	Output 2 :
	
	`*  *`
	`**  **`
	`***  ***`
	`****  ****`
	`*****  *****`
	`**********`
	`**********`
	`*****  *****`
	`****  ****`
	`***  ***`
	`**  **`
	`*  *`	

5	**Sum of two prime numbers** **Problem statement:** Rita has to design a code that takes a positive integer or number which is required to be given by the user. Then the code should further identify whether that digit can be expressed as the sum of two prime numbers. If the inserted number can be expressed as sum of two prime	Hard

	numbers then, print the integer can be expressed as sum of two prime numbers as a result. **Sample Input 1:** 34 **Sample Output 1:**3,31 **Sample Input1:** 20 **Sample Output2:** 7,13 **Constraints:** N>=1	
6	**Consecutive Numbers Sum** **Problem Statement:** Given an integer n, return the number of ways you can write n as the sum of consecutive positive integers. **Example 1:** Input: n = 5 Output: 2 Explanation: 5 = 2 + 3 **Example 2:** Input: n = 9 Output: 3 Explanation: 9 = 4 + 5 = 2 + 3 + 4 **Example 3:** Input: n = 15 Output: 4 Explanation: 15 = 8 + 7 = 4 + 5 + 6 = 1 + 2 + 3 + 4 + 5 **Constraints:** 1 <= n <= 10^9	Hard

7	**Minimum One Bit Operations to Make Integers Zero** **Problem Statement:** Given an integer n, you must transform it into 0 using the following operations any number of times: Change the rightmost (0th) bit in the binary representation of n. Change the ith bit in the binary representation of n if the (i-1)th bit is set to 1 and the (i-2)th through 0th bits are set to 0. Return the minimum number of operations to transform n into 0. **Example 1:** Input: n = 3 Output: 2 Explanation: The binary representation of 3 is "11". "11" -> "01" with the 2nd operation since the 0th bit is 1. "01" -> "00" with the 1st operation. **Example 2:** Input: n = 6 Output: 4 Explanation: The binary representation of 6 is "110". "110" -> "010" with the 2nd operation since the 1st bit is 1 and 0th through 0th bits are 0. "010" -> "011" with the 1st operation. "011" -> "001" with the 2nd operation since the 0th bit is 1. "001" -> "000" with the 1st operation. **Constraints:** 0 <= n <= 10^9	Hard

8	**Median of 2 Sorted Arrays of Different Sizes** **Problem Statement :** Given two sorted arrays array1 and array2 of size m and n respectively. Find the median of the two sorted arrays. **Example 1:** **Input:** m = 3, n = 4 array1[] = {1,5,9} array2[] = {2,3,6,7} **Output:** 5 **Explanation:** The middle element for {1,2,3,5,6,7,9} is 5 **Example 2:** **Input:** m = 2, n = 4 array1[] = {4,6} array2[] = {1,2,3,5} **Output:** 3.5 **Constraints:** 0 ≤ m,n ≤ 10^6 1 ≤ array1[i], array2[i] ≤ 10^9	Hard
9	**Subarrays With Equal 0's, 1's and 2's** **Problem statement :** You are given an array containing 'N' integers. In the array, the elements are 0, 1 and 2. You have a simple task, find the count of non-empty subarrays containing an equal number of 0's, 1's and 2's. The subarray of ARR is a contiguous part of the array ARR, i. e. the array ARRi, ARRi+1, ... , ARRj for some 0 ≤ i ≤ j ≤ N - 1. For **Example :** If 'N' = 5, 'ARR' = {1, 1, 0, 2, 1} There are exactly two subarrays that contain an equal number of 0's, 1's and 2's.	Hard

The first subarray is from 'ARR[1]' to 'ARR[3]', ie: {1, 0, 2}.

The second subarray is from 'ARR[2]' to 'ARR[4]', ie: {0, 2, 1}.

Sample Input 1 :

2

5

1 1 0 2 1

4

1 1 0 0

Sample Output 1 :

2

0

Explanation For Sample Input 1 :

For test case 1 :

We will print 2 because:

There are exactly two subarrays that contain an equal number of 0's, 1's and 2's.

The first subarray is from ARR[1] to ARR[3], ie: {1, 0, 2}, and the second subarray is from ARR[2] to ARR[4], ie: {0, 2, 1}

For test case 2 :

We will print 0 because:

The input array doesn't contain any element equal to 2, so it's impossible to form a non-empty subarray with an equal number of 0's, 1's and 2's.

Sample Input 2 :

2

6

1 0 2 1 0 2

6

	1 1 0 0 2 2 **Sample Output 2 :** 5 1 **Constraints :** $1 \leq T \leq 10$ $1 \leq N \leq 1000$ ARR[i] = {0, 1, 2}	
10	**Chocolate & Sweetness** **Problem statement :** Nim wants to try some sweets in a sweet shop. There are 'N' types of sweets available at that shop. Each sweet has a sweetness level and expiry date. Nim is fond of sweets, so he sets a specific sweetness level 'X' and is only likely to buy the sweets having a sweetness level greater than equal to 'X'. But Nim also wants fresher sweets, so he only buys having expiry date strictly greater than 'Y'. Can you help Nim to find the number of sweets suitable for him to buy? You are given two arrays, 'SWEET' and 'EXPIRY', both having 'N' values corresponding to the sweetness and expiry date of ith sweet. Nim will ask 'Q' queries with 'X' as the minimum sweetness level and 'Y' as the minimum expiry date. Your task is to answer all 'Q' queries and tell the number of sweets satisfying the given condition for each query. For **Example** For the given if N = '5', 'SWEET' = [1,3,6,7,2] and 'EXPIRY = [10,7,2,6,4]. And the query is 'X'=2 and 'Y' =6 ,then the number of sweets satisfying the condition is only 1 (having sweetness 3 and expiry date 7). **Sample Input 1:** 2	Hard

5 2
1 3 6 7 2
10 7 2 6 4
2 6
3 9
3 1
1 5 7
11 7 10
3 6

Sample Output 1:

1 0
2

Explanation of sample input 1:

For the first test case,

For the first query, the number of sweets having sweetness greater than 2 and expiry greater than 6 is only 1(Sweet number 4).

For the second query, there is no sweet satisfying both the condition. Hence the answer is 0.

Hence the answer is [1,0].

For the second test case:

The number of sweets satisfying both the conditions is 2. (2nd and the 3rd sweet.)

Hence the answer is [2].

Sample Input 2:

2
5 2
2 9 4 8 7
3 8 3 2 8
7 9
3 3
4 3
9 2 6 4
1 10 10 2
1 2
7 2

	2 2 **Sample Output 2:** 0 2 2 0 2 **Constraints:** 1 <= T <= 10 1 <= N <= 100000. 1 <= Q <= 100000. 1 <= SWEET[i] <=100000 1 <= EXPIRY[i] <= 100000	
11	**Smallest Positive Integer that cannot be represented as Sum** **Problem Statement :** Given an array of size N, find the smallest positive integer value that is either not presented in the array or cannot be represented as a sum(coz sum means you are adding two or more elements) of some elements from the array. **Example 1:** **Input:** N = 6 array[] = {1, 10, 3, 11, 6, 15} **Output:** 2 **Explanation:** 2 is the smallest integer value that cannot be represented as sum of elements from the array. **Example 2:** **Input:** N = 3 array[] = {1, 1, 1} **Output:** 4 **Explanation:** 1 is present in the array.	Hard

	2 can be created by combining two 1s. 3 can be created by combining three 1s. 4 is the smallest integer value that cannot be represented as sum of elements from the array. **Constraints:** 1 ≤ N ≤ 10^6 1 ≤ array[i] ≤ 10^9 The array may contain duplicates.	
12	**Candy** **Problem Statement :** There are N children standing in a line. Each child is assigned a rating value given in the integer array ratings. You are giving candies to these children subjected to the following requirements: • Each child must have atleast one candy. • Children with a higher rating than its neighbors get more candies than their neighbors. Return the minimum number of candies you need to have to distribute. **Example 1:** **Input:** N = 3 ratings = [1, 0, 2] **Output:** 5 **Explanation:** You can allocate to the first, second and third child with 2, 1, 2 candies respectively. **Example 2:** **Input:** N = 3 ratings = [1, 2, 2] **Output:** 4 **Explanation:**	Hard

	You can allocate to the first, second and third child with 1, 2, 1 candies respectively. The third child gets 1 candy because it satisfies the above two conditions. **Constraints:** 1 ≤ N ≤ 10^6 0 ≤ ratingsi ≤ 10^9	
13	**Maximum Sum Rectangle** **Problem statement :** You are given an M X N matrix of integers ARR. Your task is to find the maximum sum rectangle. Maximum sum rectangle is a rectangle with the maximum value for the sum of integers present within its boundary, considering all the rectangles that can be formed from the elements of that matrix. A rectangle is a 2-D polygon with opposite sides parallel and equal to each other. For **example:** Consider following matrix: 1 2 -1 -4 -20 -8 -3 4 2 1 3 8 10 1 3 -4 -1 1 7 -6 The rectangle (1,1) to (3,3) is the rectangle with the maximum sum, i.e. 29. 1 2 -1 -4 -20 -8 |-3 4 2 | 1 3 | 8 10 1 | 3 -4 |-1 1 7 | -6 **Sample Input 1:** 3 1 1	Hard

1
1 2
-1 1
2 2
1 -1
2 2

Sample Output 1:

1
1
4

Explanation of Input 1:

(i) Maximum sum rectangle is (0,0)-(0,0)
(ii) Maximum sum rectangle is (0,1)-(0,1)
(iii) Maximum sum rectangle is (1,0)-(1,1)

Sample Input 2:

3
1 2
0 0
2 2
1 0
0 1
4 5
1 2 -1 -4 -20
-8 -3 4 2 1
3 8 10 1 3
-4 -1 1 7 -6

Sample Output 2:

0
2
29

Explanation of Input 2:

(i) Maximum sum rectangle is (0,0)-(0,1)
(ii) Maximum sum rectangle is (0,0)-(1,1)
(iii) Maximum sum rectangle is (1,1)-(3,3)

Constraints:

	1 <= T <= 10 1 <= M, N <= 100 -10^5 <= ARR[i] <=10^5	
14	**Sum of Floored Pairs** **Problem Statement :** Given an integer array nums, return the sum of floor(nums[i] / nums[j]) for all pairs of indices 0 <= i, j < nums.length in the array. Since the answer may be too large, return it modulo 10^9 + 7. The floor() function returns the integer part of the division. **Example 1:** **Input:** nums = [2,5,9] **Output:** 10 **Explanation:** floor(2 / 5) = floor(2 / 9) = floor(5 / 9) = 0 floor(2 / 2) = floor(5 / 5) = floor(9 / 9) = 1 floor(5 / 2) = 2 floor(9 / 2) = 4 floor(9 / 5) = 1 We calculate the floor of the division for every pair of indices in the array then sum them up. **Example 2:** **Input:** nums = [7,7,7,7,7,7,7] **Output:** 49 **Constraints:** • 1 <= nums.length <= 10^5 • 1 <= nums[i] <= 10^5	Hard
15	**Maximum subset XOR** **Problem Statement :** Given an array arr[] of N positive integers. Find an integer denoting the maximum XOR subset value in the given array arr[]. **Example 1:**	Hard

	Input : N = 3 arr[] = {2, 4, 5} Output : 7 **Explanation :** The subset {2, 5} has maximum subset XOR value. **Example 2 :** **Input :** N= 3 arr[] = {9, 8, 5} **Output :** 13 **Explanation :** The subset {8, 5} has maximum subset XOR value. **Contraints :** 1 <= N <= 10^5 1 <= arr[i] <= 10^6	
16	**Poor Pigs** **Problem Statement:** There are buckets buckets of liquid, where exactly one of the buckets is poisonous. To figure out which one is poisonous, you feed some number of (poor) pigs the liquid to see whether they will die or not. Unfortunately, you only have minutesToTest minutes to determine which bucket is poisonous. You can feed the pigs according to these steps: Choose some live pigs to feed. For each pig, choose which buckets to feed it. The pig will consume all the chosen buckets simultaneously and will take no time. Each pig can feed from any number of buckets, and each bucket can be fed from by any number of pigs. Wait for minutesToDie minutes. You may not feed any other pigs during this time. After minutesToDie minutes have passed, any pigs that have been fed the poisonous bucket will die, and all others will survive. Repeat this process until you run out of time.	Hard

	Given buckets, minutesToDie, and minutesToTest, return the minimum number of pigs needed to figure out which bucket is poisonous within the allotted time. **Example 1:** Input: buckets = 4, minutesToDie = 15, minutesToTest = 15 Output: 2 Explanation: We can determine the poisonous bucket as follows: At time 0, feed the first pig buckets 1 and 2, and feed the second pig buckets 2 and 3. At time 15, there are 4 possible outcomes: - If only the first pig dies, then bucket 1 must be poisonous. - If only the second pig dies, then bucket 3 must be poisonous. - If both pigs die, then bucket 2 must be poisonous. - If neither pig dies, then bucket 4 must be poisonous. **Example 2:** Input: buckets = 4, minutesToDie = 15, minutesToTest = 30 Output: 2 Explanation: We can determine the poisonous bucket as follows: At time 0, feed the first pig bucket 1, and feed the second pig bucket 2. At time 15, there are 2 possible outcomes: - If either pig dies, then the poisonous bucket is the one it was fed. - If neither pig dies, then feed the first pig bucket 3, and feed the second pig bucket 4. At time 30, one of the two pigs must die, and the poisonous bucket is the one it was fed. **Constraints:** 1 <= buckets <= 1000 1 <= minutesToDie <= minutesToTest <= 100	
17	**Count Array Pairs Divisible by K** **Problem Statement :** Given a 0-indexed integer array nums of length n and an integer k, return the number of pairs (i, j) such that: • 0 <= i < j <= n - 1 and	Hard

	• nums[i] * nums[j] is divisible by k. **Example 1:** **Input:** nums = [1,2,3,4,5], k = 2 **Output:** 7 **Explanation:** The 7 pairs of indices whose corresponding products are divisible by 2 are (0, 1), (0, 3), (1, 2), (1, 3), (1, 4), (2, 3), and (3, 4). Their products are 2, 4, 6, 8, 10, 12, and 20 respectively. Other pairs such as (0, 2) and (2, 4) have products 3 and 15 respectively, which are not divisible by 2. **Example 2:** **Input:** nums = [1,2,3,4], k = 5 **Output:** 0 **Explanation:** There does not exist any pair of indices whose corresponding product is divisible by 5. **Constraints:** • 1 <= nums.length <= 10^5 • 1 <= nums[i], k <= 10^5	
18	**Max min Height** **Problem Statement :** You have a garden with n flowers lined up in a row. The height of ith flower is ai units. You will water them for k days. In one day you can water w continuous flowers (you can do this only once in a single day). Whenever you water a flower its height increases by 1 unit. You need to maximize the height of the smallest flower all the time. **Example 1:** **Input:** n=6 k=2 w=3 a[]={2,2,2,2,1,1} **Output:** 2	Hard

	Explanation: Water last three flowers for first day & first three flowers for second day.The new heights will be {3,3,3,3,2,2} **Example 2:** **Input:** n=2 k=5 w=1 a[]={5,8} **Output:** **9** **Explanation:** For the first four days water the first flower then water the last flower once. **Constraints:** 1 <= n<= 10^5 1<=w<=n 1<=k<=10^5 1 <= a[i] <= 10^9	
19	**Split Array Largest Sum** **Problem Statement :** Given an array arr[] of N elements and a number K., split the given array into K subarrays such that the maximum subarray sum achievable out of K subarrays formed is minimum possible. Find that possible subarray sum. **Example 1:** **Input:** N = 4, K = 3 arr[] = {1, 2, 3, 4} **Output:** 4 **Explanation:** Optimal Split is {1, 2}, {3}, {4}. Maximum sum of all subarrays is 4, which is minimum possible for 3 splits. **Example 2:**	Hard

	Input: N = 3, K = 2 A[] = {1, 1, 2} **Output:** 2 **Explanation:** Splitting the array as {1,1} and {2} is optimal. This results in a maximum sum subarray of 2. **Constraints:** 1 ≤ N ≤ 10^5 1 ≤ K ≤ N 1 ≤ arr[i] ≤ 10^4	
20	**Max Circular Subarray Sum** **Problem Statement :** Given an array arr[] of N integers arranged in a circular fashion. Your task is to find the maximum contiguous subarray sum **Example 1:** **Input:** N = 7 arr[] = {8,-8,9,-9,10,-11,12} **Output:** 22 **Explanation:** Starting from the last element of the array, i.e, 12, and moving in a circular fashion, we have max subarray as 12, 8, -8, 9, -9, 10, which gives maximum sum as 22. **Example 2:** **Input:** N = 8 arr[] = {10,-3,-4,7,6,5,-4,-1} **Output:** 23 **Explanation**: Sum of the circular subarray with maximum sum is 23	Hard

Constraints:

1 <= N <= 10^6

-10^6 <= Arr[i] <= 10^6

Sample Code:

```
static int normalMaxSum(int a[],int n){
    int res=a[0],maxEnd=a[0];
    for(int i=1;i<n;i++){
        maxEnd=Math.max(maxEnd+a[i],a[i]);
        res=Math.max(res,maxEnd);
    }
    return res;
}
static int circularSubarraySum(int a[], int n) {
    int max_normal=normalMaxSum(a,n);
    if(max_normal<0){
        return max_normal;
    }
    int arr_sum=0;
    for(int i=0;i<n;i++){
        arr_sum+=a[i];
        a[i]=-a[i];
    }
    int maxCircular=arr_sum+normalMaxSum(a,n);
    return Math.max(maxCircular,max_normal);
}
```

21	**Mining Diamonds** **Problem statement :** There are 'N' diamonds in a mine. The size of each diamond is given in the form of integer array 'A'. If the miner mines a diamond, then he gets 'size of previous unmined diamond * size of currently mined diamond * size of next unmined diamond' number of coins. If there isn't any next or previous unmined diamond then their size is replaced by 1 while calculating the number of coins.	Hard

Vladimir, a dumb miner was assigned the task to mine all diamonds. Since he is dumb he asks for your help to determine the maximum number of coins that he can earn by mining the diamonds in an optimal order.

For **example:**

Suppose 'N' = 3, and 'A' = [7, 1, 8]

The optimal order for mining diamonds will be [2, 1, 3].

State of mine - [7, 1, 8] [7, 8] [8]

Coins earned - (7*1*8) + (1*7*8) + (1*8*1) = 56 + 56 + 8 = 120

Hence output will be 120.

Sample Input 1 :

2

3

7 1 8

2

9 1

Sample Output 1 :

120

18

Explanation For Sample Input 1 :

For First Case - Same as explained in above example.

For the second case - 'N' = 2, and 'A' = [9, 1]

The optimal order for mining diamonds will be [2, 1].

State of mine - [9, 1] [9]

Coins earned - (1*9*1) + (1*9*1) = 9 + 9 = 18

Hence output will be 18..

Sample Input 2 :

2

5

1 2 3 4 5

4

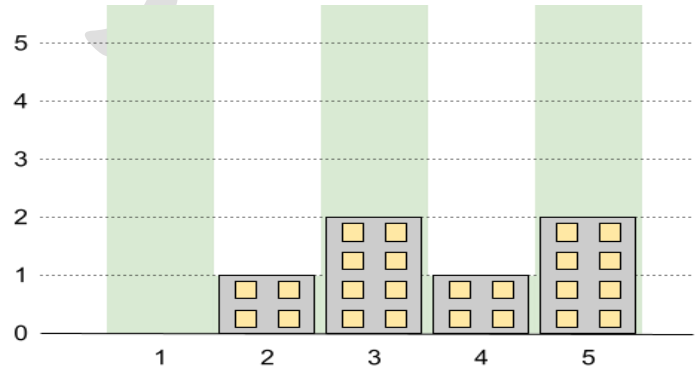
1 5 2 8

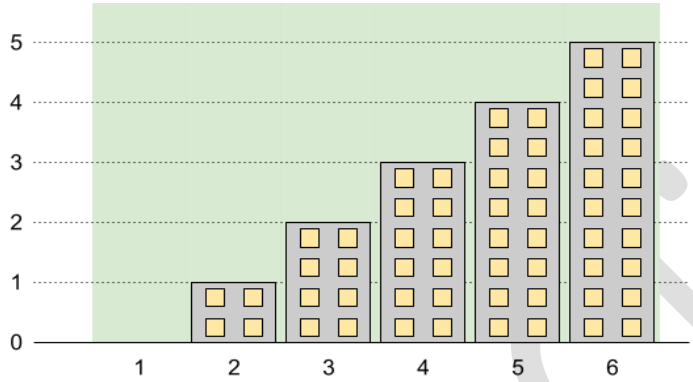
Sample Output 2 :

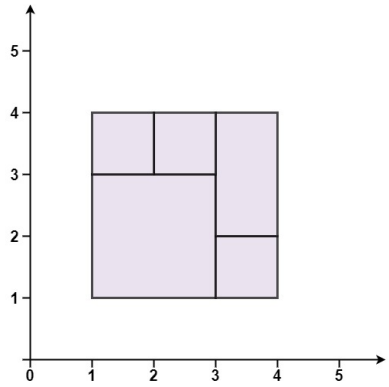
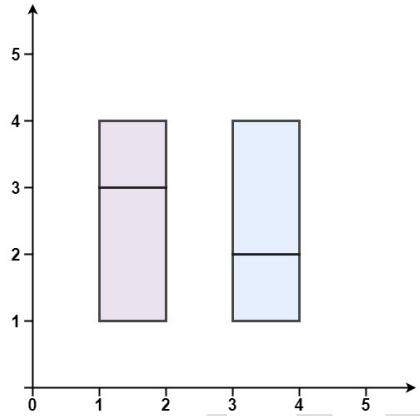
110

136

22	**Matrix Exponentiation** **Problem statement :** Sai is playing a game, he is standing on index 1 in a linear array of size 'N', then he rolls a dice and moves forward according to the number on the top of the dice. He repeats this operation until he reached index 'N'. He asked you to find the expected number of moves required to reach position 'N' starting from '1'. **Note:** 1. The expected number of moves can be fractional. 2. Sai cannot go outside the array i.e if he is at (n - 1)-th position he can only move if he gets 1 as an outcome in dice. **Sample Input 1:** 2 2 8 **Sample Output 1:** 1 1 **Explanation of Sample Input 1:** **Test case 1:** Here the answer is equal to the expected number of dice rolls to get the first 1 as an outcome because we cannot go outside the array. And the expected number of dice rolls to get the first '1' is 6. i.e. On average, we need to roll a dice 6 times to get at position 2. **Test case 2.** If we are 7 steps away, then we can get from 1 by getting '6' with probability (1 / 6) and expected dice roll equal to 6, similarly, from 2 we can get 5, from 3 by 4, and so on. with all same probability i.e. 1 / 6 and expected no. of dice roll 6. Now the average value will be 1 + ((6 * 6) / 6 = 7. Hence, for 7 steps away, our answer is 7. **Sample Input 2:** 1 3	Hard

	Sample Output 2: 1 **Explanation of Sample Input 2:** **Test case 1:** As the Input is less than 7, the expected number of operations will be 6. **Constraints:** 1 <= T <= 5 1 <= N <= 10^9	
23	**Maximum Building Height** **Problem Statement :** You want to build n new buildings in a city. The new buildings will be built in a line and are labeled from 1 to n. However, there are city restrictions on the heights of the new buildings: • The height of each building must be a non-negative integer. • The height of the first building must be 0. • The height difference between any two adjacent buildings cannot exceed 1. Additionally, there are city restrictions on the maximum height of specific buildings. These restrictions are given as a 2D integer array restrictions where restrictions[i] = [idi, maxHeighti] indicates that building idi must have a height less than or equal to maxHeighti. It is guaranteed that each building will appear at most once in restrictions, and building 1 will not be in restrictions. Return the maximum possible height of the tallest building. **Example 1:**  	Hard

	Input: n = 5, restrictions = [[2,1],[4,1]] **Output:** 2 **Explanation:** The green area in the image indicates the maximum allowed height for each building. We can build the buildings with heights [0,1,2,1,2], and the tallest building has a height of 2. **Example 2:**  **Input:** n = 6, restrictions = [] **Output:** 5 **Explanation:** The green area in the image indicates the maximum allowed height for each building. We can build the buildings with heights [0,1,2,3,4,5], and the tallest building has a height of 5. **Constraints:** • 2 <= n <= 10^9 • 0 <= restrictions.length <= min(n - 1, 10^5) • 2 <= idi <= n • idi is unique. • 0 <= maxHeighti <= 10^9	
24	**Perfect Rectangle** **Problem Statement :** Given an array rectangles where rectangles[i] = [xi, yi, ai, bi] represents an axis-aligned rectangle. The bottom-left point of the rectangle is (xi, yi) and the top-right point of it is (ai, bi).Return true if all the rectangles together form an exact cover of a rectangular region.	Hard

	Example 1:  **Input**: rectangles = [[1,1,3,3],[3,1,4,2],[3,2,4,4],[1,3,2,4],[2,3,3,4]] **Output**: true **Explanation:** All 5 rectangles together form an exact cover of a rectangular region. **Example 2:**  **Input:** rectangles = [[1,1,2,3],[1,3,2,4],[3,1,4,2],[3,2,4,4]] **Output**: false **Explanation**: Because there is a gap between the two rectangular regions. **Constraints:** 1 <= rectangles.length <= 2 * 10^4 rectangles[i].length == 4 -10^5 <= xi, yi, ai, bi <= 10^5	
25	**Print Spiral** **Problem statement :** For a given two-dimensional integer array/list of size (N x M), print it in a spiral form. That is, you need to print in the order followed for every iteration: a. First row(left to right)	Hard

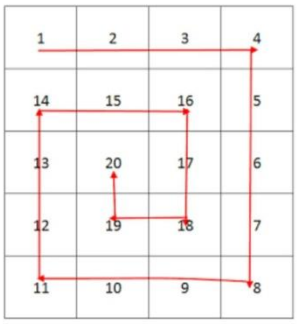
b. Last column(top to bottom)

c. Last row(right to left)

d. First column(bottom to top)

Mind that every element will be printed only once.

Refer to the Image:



Output : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Sample Input 1:

1

4 4

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Sample Output 1:

1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Sample Input 2:

2

3 3

1 2 3

4 5 6

7 8 9

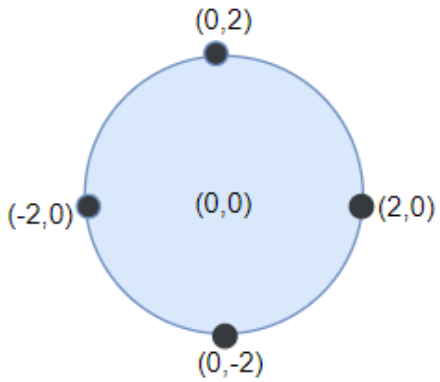
3 1

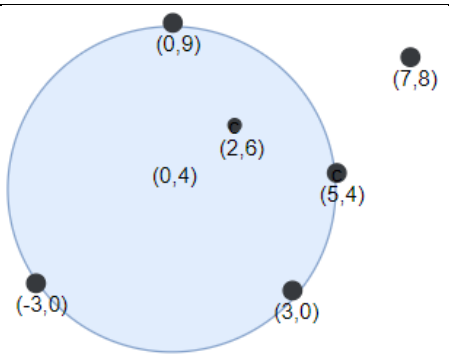
10

20

30

Sample Output 2:

	1 2 3 6 9 8 7 4 5 10 20 30 **Constraints :** 1 <= t <= 10^2 0 <= N <= 10^3 0 <= M <= 10^3	
26	**Maximum Number of Darts Inside of a Circular Dartboard** **Problem Statement :** Alice is throwing n darts on a very large wall. You are given an array darts where darts[i] = [xi, yi] is the position of the ith dart that Alice threw on the wall. Bob knows the positions of the n darts on the wall. He wants to place a dartboard of radius r on the wall so that the maximum number of darts that Alice throws lie on the dartboard. Given the integer r, return the maximum number of darts that can lie on the dartboard. **Example 1:**  **Input:** darts = [[-2,0],[2,0],[0,2],[0,-2]], r = 2 **Output**: 4 **Explanation:** Circle dartboard with center in (0,0) and radius = 2 contain all points. **Example 2:**	Hard



Input: darts = [[-3,0],[3,0],[2,6],[5,4],[0,9],[7,8]], r = 5

Output: 5

Explanation: Circle dartboard with center in (0,4) and radius = 5 contain all points except the point (7,8).

Constraints:

- 1 <= darts.length <= 100
- darts[i].length == 2
- -104 <= xi, yi <= 10^4
- All the darts are unique
- 1 <= r <= 5000

27	**Paint House** **Problem statement :** Alex has started a painting business recently. He got a contract to paint 'N' houses in a city. Alex has 'K' colors to choose from. But the client has a condition that no two adjacent houses have the same color. The cost of painting each house with a certain color is represented by an N x K cost matrix. For example, costs[0][0] is the cost of painting house 0 with color 0; costs[1][2] is the cost of painting house 1 with color 2, and so on. Your task is to help Alex to find the minimum cost required to paint houses according to this condition. **For Example :** Let say N = 2 and K = 3 and costs = [[1,5,3] , [2,9,4]] In this case,	Hard

Alex can paint house 0 into color 0, paint house 1 into color 2. Minimum cost: 1 + 4 = 5;

Or paint house 0 into color 2, paint house 1 into color 0. Minimum cost: 3 + 2 = 5 .

Note :

Assume 0 based indexing

Sample Input 1 :

2

2 3

1 5 3

2 9 4

3 3

1 4 5

2 3 5

6 7 8

Sample Output 1 :

5

10

Explanation Of Sample Input 1 :

Test Case 1 :

In this case, Alex can paint house 0 into color 0, paint house 1 into color 2. Minimum cost: 1 + 4 = 5;

Or paint house 0 into color 2, paint house 1 into color 0. Minimum cost: 3 + 2 = 5

Hence the minimum cost will be 5.

Test case 2 :

In this case, Alex can paint house 0 into color 0, paint house 1 into color 1, paint house 2 in color 0. Minimum cost : 1 + 3 + 6 = 10.

Sample Input 2 :

2

2 3

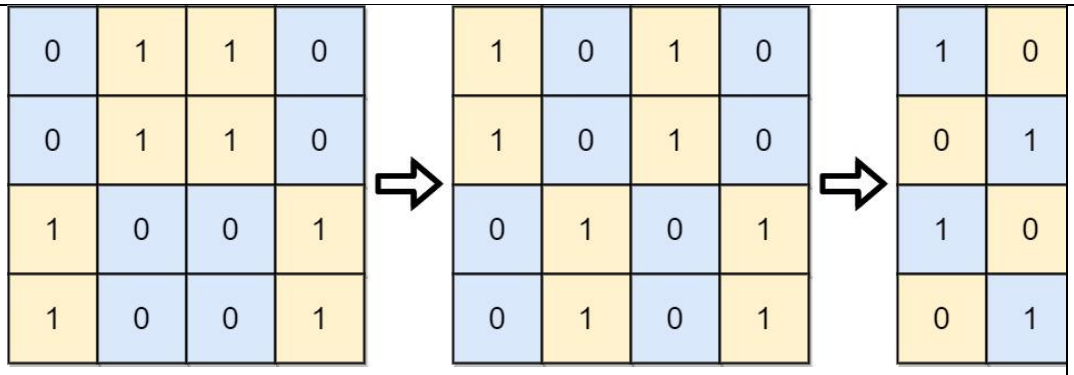
1 2 3

10 11 12

1 2

	4 2 **Sample output 2 :** 12 2	
28	**Bricks Falling when hit** **Problem Statement:** You are given an m x n binary grid, where each 1 represents a brick and 0 represents an empty space. A brick is stable if: • It is directly connected to the top of the grid, or • At least one other brick in its four adjacent cells is stable. You are also given an array hits, which is a sequence of erasures we want to apply. Each time we want to erase the brick at the location hits[i] = (rowi, coli). The brick on that location (if it exists) will disappear. Some other bricks may no longer be stable because of that erasure and will fall. Once a brick falls, it is immediately erased from the grid (i.e., it does not land on other stable bricks). Return an array result, where each result[i] is the number of bricks that will fall after the ith erasure is applied. Note that an erasure may refer to a location with no brick, and if it does, no bricks drop. **Example 1:** **Input:** grid = [[1,0,0,0],[1,1,1,0]], hits = [[1,0]] **Output:** [2] **Explanation:** Starting with the grid: [[1,0,0,0], [1,1,1,0]] We erase the underlined brick at (1,0), resulting in the grid: [[1,0,0,0], [0,1,1,0]] The two underlined bricks are no longer stable as they are no longer connected to the top nor adjacent to another stable brick, so they will fall. The resulting grid is: [[1,0,0,0], [0,0,0,0]] Hence the result is [2].	Hard

	Example 2: **Input**: grid = [[1,0,0,0],[1,1,0,0]], hits = [[1,1],[1,0]] **Output:** [0,0] **Explanation:** Starting with the grid: [[1,0,0,0], [1,1,0,0]] We erase the underlined brick at (1,1), resulting in the grid: [[1,0,0,0], [1,0,0,0]] All remaining bricks are still stable, so no bricks fall. The grid remains the same: [[1,0,0,0], [1,0,0,0]] Next, we erase the underlined brick at (1,0), resulting in the grid: [[1,0,0,0], [0,0,0,0]] Once again, all remaining bricks are still stable, so no bricks fall. Hence the result is [0,0].	
29	**Transform to Chessboard** You are given an n x n binary grid board. In each move, you can swap any two rows with each other, or any two columns with each other. Return the minimum number of moves to transform the board into a chessboard board. If the task is impossible, return -1. A chessboard board is a board where no 0's and no 1's are 4-directionally adjacent. **Example 1:**	Hard



Input: board = [[0,1,1,0],[0,1,1,0],

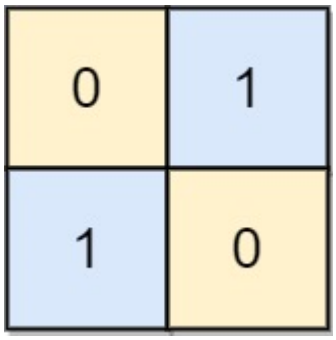
Output: 2

Explanation: One potential sequence of moves is shown.

The first move swaps the first and second column.

The second move swaps the second and third row.

Example 2:

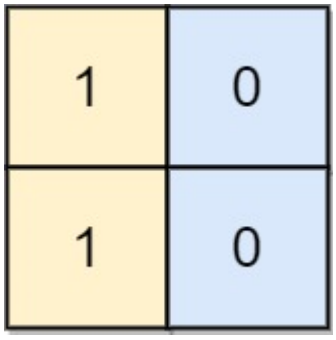


Input: board = [[0,1],[1,0]]

Output: 0

Explanation: Also note that the board with 0 in the top left corner, is also a valid chessboard.

Example 3:



Input: board = [[1,0],[1,0]]

Output: -1

	Explanation: No matter what sequence of moves you make, you cannot end with a valid chessboard. **Constraints:** n == board.length n == board[i].length 2 <= n <= 30 board[i][j] is either 0 or 1.	
30	**Kth Smallest Amount With Single Denomination Combination** **Problem statement :** You are given an integer array coins representing coins of different denominations and an integer k. You have an infinite number of coins of each denomination. However, you are not allowed to combine coins of different denominations. Return the kth smallest amount that can be made using these coins. **Example 1:** Input: coins = [3,6,9], k = 3 Output: 9 Explanation: The given coins can make the following amounts: Coin 3 produces multiples of 3: 3, 6, 9, 12, 15, etc. Coin 6 produces multiples of 6: 6, 12, 18, 24, etc. Coin 9 produces multiples of 9: 9, 18, 27, 36, etc. All of the coins combined produce: 3, 6, 9, 12, 15, etc. Example 2: Input: coins = [5,2], k = 7 Output: 12 Explanation: The given coins can make the following amounts: Coin 5 produces multiples of 5: 5, 10, 15, 20, etc. Coin 2 produces multiples of 2: 2, 4, 6, 8, 10, 12, etc. All of the coins combined produce: 2, 4, 5, 6, 8, 10, 12, 14, 15, etc. **Constraints:** 1 <= coins.length <= 15 1 <= coins[i] <= 25 1 <= k <= 2 * 10^9 coins contains pairwise distinct integers.	Hard