1) Missing in Array

Given an array of size **n-1** such that it only contains distinct integers in the range of **1 to n**. Return the missing element.

Examples:

Input: n = 5, arr[] = {1,2,3,5} **Output:** 4 **Explanation :** All the numbers from 1 to 5 are present except 4.
Input: n = 2, arr[] = {1} **Output:** 2 **Explanation :** All the numbers from 1 to 2 are present except 2.
Expected Time Complexity: O(n)
Expected Auxiliary Space: O(1)

Constraints:
1 ≤ n ≤ 10^6
1 ≤ arr[i] ≤ 10^6

2) Count the triplets

Given an array **Arr** consisting of **N** distinct integers. The task is to count all the triplets such that sum of two elements equals the third element.

Example 1:

Input: N = 4 arr[] = {1, 5, 3, 2} **Output:** 2 **Explanation**: There are 2 triplets: 1 + 2 = 3 and 3 +2 = 5

Example 2:

Input: N = 3 arr[] = {2, 3, 4} **Output:** 0 **Explanation**: No such triplet exits
Your Task:
You don't need to read input or print anything. Your task is to complete the function **countTriplet()** which takes the array **arr[]** and **N** as inputs and returns the triplet count

Expected Time Complexity: O(N2)

Expected Auxiliary Space: O(1)

Constraints:
$1 \leq N \leq 10^3$
$1 \leq arr[i] \leq 10^5$

3) Merge Without Extra Space

Given two sorted arrays **arr1[]** and **arr2[]** of sizes **n** and **m** in non-decreasing order. Merge them in sorted order without using any extra space. Modify arr1 so that it contains the first N elements and modify arr2 so that it contains the last M elements.

Example 1:

Input: n = 4, arr1[] = [1 3 5 7] m = 5, arr2[] = [0 2 6 8 9] **Output**: arr1[] = [0 1 2 3] arr2[] = [5 6 7 8 9] **Explanation**: After merging the two non-decreasing arrays, we get, 0 1 2 3 5 6 7 8 9.

Example 2:

Input: n = 2, arr1[] = [10 12] m = 3, arr2[] = [5 18 20] **Output**: arr1[] = [5 10] arr2[] = [12 18 20] **Explanation**: After merging two sorted arrays we get 5 10 12 18 20.

Your Task:

You don't need to read input or print anything. You only need to complete the function**merge()** that takes arr1, arr2, n and m as input parameters and modifies them in-place so that they look like the sorted merged array when concatenated.

Expected Time Complexity: O((n+m) log(n+m))
Expected Auxilliary Space: O(1)

Constraints:
$1 <= n, m <= 10^5$
$0 <= arr1_i, arr2_i <= 10^7$

4) Reverse array in groups

Given an array arr[] of positive integers of size N. Reverse every sub-array group of size K.

Note: If at any instance, there are no more subarrays of size greater than or equal to K, then reverse the last subarray (irrespective of its size). You shouldn't return any array, modify the given array in-place.

Example 1:

Input: N = 5, K = 3 arr[] = {1,2,3,4,5} **Output:** 3 2 1 5 4 **Explanation:** First group consists of elements 1, 2, 3. Second group consists of 4,5.

Example 2:

Input: N = 4, K = 3 arr[] = {5,6,8,9} **Output:** 8 6 5 9

Your Task:
You don't need to read input or print anything. The task is to complete the function **reverseInGroups**() which takes the array, N and K as input parameters and modifies the array in-place.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
1 ≤ N, K ≤ 10^7
1 ≤ A[i] ≤ 10^18

5) Convert array into Zig-Zag fashion

Given an array **arr** of distinct elements of size **n**, the task is to rearrange the elements of the array in a zig-zag fashion so that the converted array should be in the below form:

arr[0] < arr[1] > arr[2] < arr[3] > arr[4] < . . . . arr[n-2] < arr[n-1] > arr[n].

Note: Modify the given arr[] only, If your transformation is correct, the output will be 1 else the output will be 0.

Examples

Input: n = 7, arr[] = {4, 3, 7, 8, 6, 2, 1} **Output:** 1 **Explanation:** After modification the array will look like 3 < 7 > 4 < 8 > 2 < 6 > 1, the checker in the driver code will produce 1.
Input: n = 5, arr[] = {4, 7, 3, 8, 2} **Output:** 1 **Explanation:** After modification the array will look like 4 < 7 > 3 < 8 > 2 hence output will be 1.

Expected Time Complexity: O(n)
Expected Auxiliary Space: O(1)

Constraints:
1 <= n <= 10^6
0 <= arr$_i$ <= 10^9

6) Spirally traversing a matrix

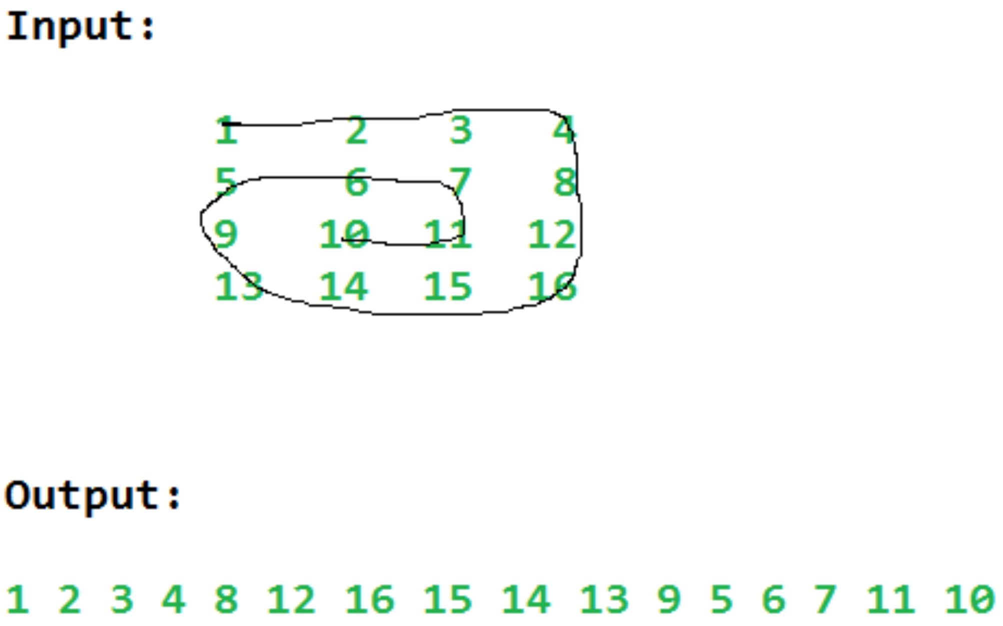
Difficulty: **Medium**Accuracy: **35.2%**Submissions:**248K**+Points: **4**

Given a matrix of size r*c. Traverse the matrix in spiral form.

Example 1:

Input: r = 4, c = 4 matrix[][] = {{1, 2, 3, 4},   {5, 6, 7, 8},   {9, 10, 11, 12},   {13, 14, 15,16}} **Output**: 1 2 3 4 8 12 16 15 14 13 9 5 6 7

11 10 **Explanation**:



Input:

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

Output:

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

Example 2:

Input: r = 3, c = 4 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 **Explanation**: Applying same technique as shown above, output for the 2nd testcase will be 1 2 3 4 8 12 11 10 9 5 6 7.

Your Task:
You dont need to read input or print anything. Complete the function **spirallyTraverse()** that takes **matrix, r** and **c** as input parameters and returns a list of integers denoting the spiral traversal of matrix.

Expected Time Complexity: O(r*c)
Expected Auxiliary Space: O(r*c), for returning the answer only.

Constraints:
1 <= r, c <= 100
0 <= matrix$_i$ <= 100

7) Largest Number formed from an Array

Difficulty: **Medium**Accuracy: **37.82%**Submissions:**157K+**Points: **4**

Given an array of strings **arr[]** of length **n** representing non-negative integers, arrange them in a manner, such that, after concatanating them in order, it results in the **largest possible number**. Since the result may be very large, return it as a string.

Example 1:

Input: n = 5 arr[] = {"3", "30", "34", "5", "9"} **Output:** "9534330"
Explanation:
Given numbers are {"3", "30", "34", "5", "9"},
the arrangement "9534330" gives the largest value.

Example 2:

Input: n = 4 arr[] = {"54", "546", "548", "60"} **Output:** "6054854654" **Explanation:**
Given numbers are {"54", "546", "548", "60"}, the arrangement "6054854654" gives the largest value.

Your Task:

You don't need to read input or print anything. Your task is to complete the function **printLargest()** which takes the array of strings **arr[]** as a parameter and **returns a string** denoting the answer.

Expected Time Complexity: O(n*log(n)).
Expected Auxiliary Space: O(n).

Constraints:
1 ≤ n ≤ 10^5
0 ≤ arr[i] ≤ 10^18
Sum of all the elements of the array is greater than 0.

8) Pythagorean Triplet

Given an array **arr** of **n** integers, write a function that returns true if there is a **triplet (a, b, c)** from the array (where a, b, and c are on different indexes) that satisfies **a2 + b2 = c2**, otherwise return false.

Example 1:

Input: N = 5 Arr[] = {3, 2, 4, 6, 5} **Output:** Yes **Explanation:** a=3, b=4, and c=5 forms a pythagorean triplet.

Example 2:

Input: N = 3 Arr[] = {3, 8, 5} **Output:** No **Explanation:** No such triplet possible.

Your Task:

You don't have to take any input or print any thing. You have to complete the function **checkTriplet()** which takes an array **arr,** a single integer **n,** as input parameters and returns **boolean** denoting answer to the problem.

Note: The driver will print "**Yes**" or "**No**" instead of corresponding to the boolean value returned.

Expected Time Complexity: O(n+max(Arr[i])2)

Expected Auxiliary Space: O(max(Arr[i]))

Constraints:

1 <= n <= 10^5

1 <= arr[i] <= 1000

9) Find maximum possible stolen value from houses (House Robber)

There are **N** houses built in a line, each of which contains some value in it. A thief is going to steal the maximum value of these houses, but he can’t steal in two adjacent houses because the owner of the stolen houses will tell his two neighbors left and right sides. The task is to find what is the maximum stolen value.

Examples:

Input: *hval[] = {6, 7, 1, 3, 8, 2, 4}*
Output: *19*
Explanation: *The thief will steal 6, 1, 8 and 4 from the house.*

Input: *hval[] = {5, 3, 4, 11, 2}*
Output: *16*
Explanation: *Thief will steal 5 and 11*

10) Chocolate Distribution Problem

Given an **array of N integers** where each value represents the **number of chocolates** in a packet. Each packet can have a **variable number** of chocolates. There are **m students**, the task is to **distribute** chocolate packets such that:

- Each student gets **one** packet.
- The **difference between** the number of chocolates in the packet with **maximum chocolates** and the packet with **minimum chocolates** given to the students is **minimum**.

Examples:

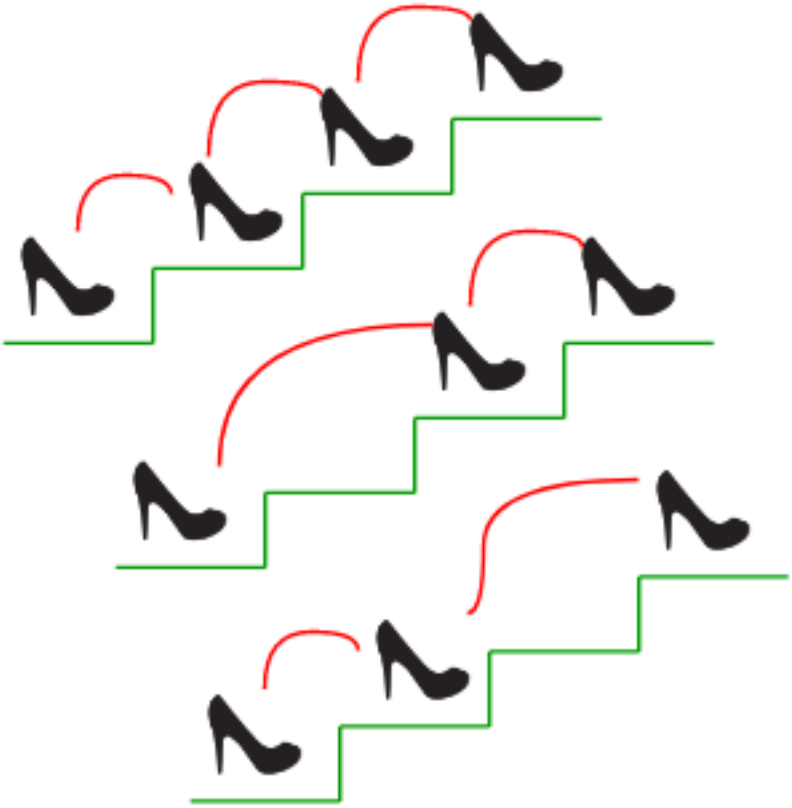
Input : *arr[] = {7, 3, 2, 4, 9, 12, 56} , m = 3*
Output: *Minimum Difference is 2*
Explanation:
We have seven packets of chocolates and we need to pick three packets for 3 students
If we pick 2, 3 and 4, we get the minimum difference between maximum and minimum packet sizes.

Input : *arr[] = {3, 4, 1, 9, 56, 7, 9, 12} , m = 5*
Output: *Minimum Difference is 6*

Input : *arr[] = {12, 4, 7, 9, 2, 23, 25, 41, 30, 40, 28, 42, 30, 44, 48, 43, 50} , m = 7*
Output: *Minimum Difference is 10*

11) Climbing Stairs to reach at the top.

There are **n** stairs, a person standing at the bottom wants to climb stairs to reach the **nth** stair. The person can climb either **1** stair or **2** stairs at a time, the task is to count the number of ways that a person can reach at the top.



Consider the example shown in the diagram. The value of n is 3. There are 3 ways to reach the top.

Examples:

Input: *n = 1*
Output: *1 There is only one way to climb 1 stair*

Input: *n=2*
Output: *2 There are two ways: (1, 1) and (2)*

Input: *n = 4*
Output: *5 (1, 1, 1, 1), (1, 1, 2), (2, 1, 1), (1, 2, 1), (2, 2)*

12) Word Break Problem | DP-32

Given an input string and a dictionary of words, find out if the input string can be segmented into a space-separated sequence of dictionary words. See following examples for more details.
This is a famous Google interview question, also being asked by many other companies now a days.

```
Consider the following dictionary 
{ i, like, sam, sung, samsung, mobile, ice, 
  cream, icecream, man, go, mango}
Input:  ilike
Output: Yes 
The string can be segmented as "i like".
Input:  ilikesamsung
Output: Yes
The string can be segmented as "i like samsung" 
or "i like sam sung".
```

13) Egg Dropping Puzzle

The following is a description of the instance of this famous puzzle involving N = 2 eggs and a building with K = 36 floors.
Suppose that we wish to know which stories in a 36-story building are safe to drop eggs from, and which will cause the eggs to break on landing. We make a few assumptions:

- An egg that survives a fall can be used again.
- A broken egg must be discarded.
- The effect of a fall is the same for all eggs.
- If an egg breaks when dropped, then it would break if dropped from a higher floor.
- If an egg survives a fall then it would survive a shorter fall.
- It is not ruled out that the first-floor windows break eggs, nor is it ruled out that the 36th-floor does not cause an egg to break.

If only one egg is available and we wish to be sure of obtaining the right result, the experiment can be carried out in only one way. Drop the egg from the first-floor window; if it survives, drop it from the second-floor window. Continue upward until it breaks. In the worst case, this method may require 36 droppings. Suppose 2 eggs are available. What is the least number of egg droppings that are guaranteed to work in all cases?
The problem is not actually to find the critical floor, but merely to decide floors from which eggs should be dropped so that the total number of trials is minimized.

Note: In this post, we will discuss a solution to a general problem with 'N' eggs and 'K' floors

14) Matrix Chain Multiplication

Given the dimension of a sequence of matrices in an array **arr[]**, where the dimension of the **i**th matrix is **(arr[i-1] * arr[i])**, the task is to find the most efficient way to multiply these matrices together such that the total number of element multiplications is minimum.

Examples:

Input: *arr[] = {40, 20, 30, 10, 30}*
Output: *26000*
Explanation:*There are 4 matrices of dimensions 40×20, 20×30, 30×10, 10×30.*
Let the input 4 matrices be A, B, C and D.
The minimum number of multiplications are obtained by putting parenthesis in following way (A(BC))D.
*The minimum is 20*30*10 + 40*20*10 + 40*10*30*

Input: *arr[] = {1, 2, 3, 4, 3}*
Output: *30*
Explanation: *There are 4 matrices of dimensions 1×2, 2×3, 3×4, 4×3.*
Let the input 4 matrices be A, B, C and D.
The minimum number of multiplications are obtained by putting parenthesis in following way ((AB)C)D.
*The minimum number is 1*2*3 + 1*3*4 + 1*4*3 = 30*

Input: *arr[] = {10, 20, 30}*
Output: *6000*
Explanation: *There are only two matrices of dimensions 10×20 and 20×30.*
*So there is only one way to multiply the matrices, cost of which is 10*20*30*

15) Count number of ways to cover a distance

Given a distance ‘dist’, count total number of ways to cover the distance with 1, 2 and 3 steps.

Examples:

Input: *n = 3*
Output: *4*
Explanation: *Below are the four ways*
=> 1 step + 1 step + 1 step
=> 1 step + 2 step
=> 2 step + 1 step
=> 3 step

Input: *n = 4*
Output: *7*
Explanation: *Below are the four ways*
=> 1 step + 1 step + 1 step + 1 step
=> 1 step + 2 step + 1 step
=> 2 step + 1 step + 1 step
=> 1 step + 1 step + 2 step
=> 2 step + 2 step
=> 3 step + 1 step
=> 1 step + 3 step