# Arrays:

## Problem

Given an array **arr[]** of size **N,**the task is to check if it is possible to make all array elements consecutive by performing any of the following operations:

* Leave the element unchanged (arr[i] = arr[i])
* Increment the element by 1
* Decrement the element by 1

**Note:**For every array element, one of the above operations can be applied only once.

#### Observations:

There are only three possibilities:

* If first element is incremented then the whole sequence should look like arr[0] + 1, arr[0] + 2, arr[0] + 3, and so on
* If first element is left unchanged then the whole sequence should look like arr[0], arr[0] + 1, arr[0] + 2, arr[0] + 3, and so on.
* If first element is decremented then the whole sequence should look like arr[0]-1, arr[0], arr[0] + 1, arr[0] + 2, arr[0] + 3, and so on.

If it is possible to convert the given array in any of the above three possibilities then the output will be “YES” else “NO“.

## Problem

Given three arrays A[], B[] and C[] of size N each and two integers X and Y, the task is to, find the minimum cost to reach the end of the arrays with the following conditions:

Select one of the three elements at every index.

For switching from 1st array to 2nd or vice versa, extra X points are required and

For switching from 2nd array to 3rd or vice versa, extra Y points are required.

The first element (at index 0) must be selected from 1st array only.

Note: It’s not possible to switch from 1st to 3rd array directly or vice versa.

#### Solution:

Follow the below steps to implement the idea:

Create a 2D array (say min\_pnts[N][3]).

Traverse the array from the last index to the first index and in each iteration:

The value of min\_pnts[i][0] depends on min\_pnts[i+1][0] and min\_pnts[i+1][1], value of min\_pnts[i][1] depends on min\_pnts[i+1][1], min\_pnts[i+1][0] and min\_pnts[i+1][2], value of min\_pnts[i][2] depends on min\_pnts[i+1][2] and min\_pnts[i+1][1].

Check which path required minimum points (i.e. with switching array or not).

Store the minimum points required if we select the current index element from the 1st, 2nd, or 3rd array.

Return the minimum points required to traverse the array.

3.[Count ways to partition Binary Array into subarrays containing K 0s each](https://www.geeksforgeeks.org/count-ways-to-partition-binary-array-into-subarrays-containing-k-0s-each/).

4. [Calculate GCD of all pairwise sum of given two Arrays](https://www.geeksforgeeks.org/calculate-gcd-of-all-pairwise-sum-of-given-two-arrays/)

## Problem

Given an array arr[]of integers with length N and an integer X, the task is to calculate the number of subarrays with median greater than or equal to the given integer X.

#### Solution:

Follow the below steps to implement the above idea:

Replace each element of an array with 1 if it is greater than or equal to X, else replace it with -1.

Based on the above idea, for the new array, median of any subarray to be greater than or equal to X, its sum of elements should be greater than or equal to 0.

For calculating the number of subarray with a sum greater than or equal to 0:

Find prefix sum up to each index of the new array.

Traverse the newly created prefix array starting from index 1 and calculate the number of elements before it with a value less than or equal to the current value.

Add all those in the final answer as they will also form a subarray with the current one satisfying all conditions.

After finding it for an index, add the current value to a multiset.

Return the final answer.

## Problem

Given four arrays A[], B[], C[], D[] and an integer K. The task is to find the number of combinations of four unique indices p, q, r, s such that A[p] + B[q] + C[r] + D[s] ≤ K.

Input: A = {2, 3}, B = {5, 2}, C = {0}, D = {1, 2}, K = 6

Output: 3

Explanation: The following are the required combinations:

{2, 2, 0, 1}, {2, 2, 0, 2}, {3, 2, 0, 1}

#### Solution:

by using Divide and Conquer and Binary Search. Follow the steps mentioned below to solve the problem:

* Generate all possible pair combinations for A, B, and C, D.
* Assume each array has length n, then we will have two arrays, each with length n\*n. Let it be merge1 and merge2.
* Sort one of the merge array, let’s say merge2.
* Iterate through the unsorted merge1 array and find how many elements from merge2 can be paired up with a sum less than or equal to K. It can easily be done by using binary search.

## Problem

Given two numbers N and K, the task is to count the number of all possible arrays of size N such that each element is a positive integer less than or equal to K and is either a multiple or a divisor of its neighbours. Since the answer can be large, print it modulo 10^9 + 7.(solved)

## Problem

Given an integer N and an array arr[] of M pairs of type (Ai, Bi), the task is to generate the lexicographically smallest possible permutation of 1 to N such that every Ai, occurs before every Bi. ( top sort using bfs )