## ICPC ASIA DHAKA REGIONAL CONTEST 2023

## Problem A: Gona Guni

#### Author: Md Mahamudur Rahaman Sajib

#### Alternate writers: Nafis Sadique

Category: DP, Polynomials

#### Explanation:

***Author’s explanation:*** [Problem A: ICPC ASIA DHAKA REGIONAL CONTEST 2023](https://docs.google.com/document/d/1KnDS9hhybYkq3VK0qPtESAzEbuo3MGj12GOCGCegtZU/edit?usp=sharing)

***Alternative explanation:***

1. Instead of counting sum{(the minimum vertex cover)^m}, we can use the [ordered pair technique](https://codeforces.com/blog/entry/62792). This insight leads to solving a different problem: select a set of vertices of upto m size, in how many different subsets of vertices of the tree these vertices are guaranteed to be selected as the minimum vertex cover. Then we count the sum over the sizes of all sets of vertices.
2. What we really want is given a set of vertices from the minimum vertex cover, can we get back the original sub-tree. That way we can generate all possible subtrees and in turn count the subset of vertices that will contain this given set as minimum vertex cover. This turns out to be difficult to do, unless you consider the maximum matching instead of minimum vertex cover and always assign the vertex closer to the root (of the whole tree) as the cover vertex.
3. This approach can lead to a bottom up sibling dp solution. From each node we need to calculate 2 things, the number of ways to select k <= m maximum matching with the subtree below the vertex and the sibling vertices with the current vertices matching with one of the children or the current vertices becoming available for matching with the parent. Merging the count from the children and the siblings requires convolution, thankfully with some small optimization we can avoid doing fft.
4. This dp calculation is a little tricky, as you need to think about what happens in a lot of situations. Like, whether to cut off the edge from the parent and remove the subtree rooted at the current vertex, when the current vertex is matched with a child, do we count it as part of the k set or we don’t, is the vertex included in the subset or not.
5. At the end, you will have the count of selecting exactly k different vertices and the number of subsets where they will be guaranteed to be a part of the minimum vertex cover, from there we can calculate the number of ordered lists of size m from k vertices (from the ordered pair technique). The sum of them is the result.
6. To avoid doing fft, we can do this trick: instead of multiplying mXm, we avoid the suffix with 0s. So, we find the last non-zero position of both arrays, say non\_zero\_a and non\_zero\_b, so we multiply two arrays of size non\_zero\_aXnon\_zero\_b. We can also stop multiplying if the multiplication will be added to any position beyond m.
7. Alternative solution is roughly 210 lines long.

## Problem B: ASCII Table

#### Author: Shahriar Manzoor

#### Alternate writers: Rumman Mahmud, Raihat Zaman Neloy

#### Explanation:

Category: Observation and modulo arithmetic.

## Problem C: Bears

#### Author:Iftekhar Hakim Kaowsar

#### Alternate writers: Kazi Md Irshad, Pritom Kundu

#### Explanation:

put all divisor pair in list, sort it, and find the LIS. But you have to delete duplicates from the input, otherwise it may hit TL. Complexity: O(Nlg^2)

## Problem D: Pyramid

#### Author: Shahriar Manzoor

#### Alternate writers: Kazi Md Irshad, Rumman Mahmud

#### Explanation:

Math, Tarnary Search.

## Problem E: Crazy General

#### Author: Anindya Das

#### Alternate writers: Nafis Sadique, Kazi Md Irshad , Raihat Zaman Niloy

#### Explanation:

Catagory: DP, stirling number , Group theory

The way the problem is described leads us to sum over all possible values of k, x, y: S1r(k, x) \* S1r(n - k, y) \* C(n, k), where S1r(k, x) is the reduced Stirling number of 1st kind, i.e. Stirling Number of 1st kind with the limitation that every cycle needs to have at least r members. C(n, k) is the binomial coefficient. Say, x + y = z. The sum can be simplified to S1r(n, z) \* C(z, x) for all possible values of z and x. Now S1r(n, k) = (n - 1) \* S1r(n - 1, k) + C(n - 1, r - 1) \* (r - 1)! \* S1r(n - r, k - 1). We can precompute this table and Binomial Coefficients.

We still need to sum over all z and x, but here we can use sum over x: C(z, x) = 2^z. Then we need to sum over only z for each case.

## 

## Problem F: Uncle Bob and XOR Sum

#### Author: Mohammad Rumman Mahmud

#### Alternate writers: Raihat Zaman Neloy, Pritom Kundu

#### Explanation:

Catagory: Math, Gaussian, XOR Basis.

Given two arrays of integers **A** of length **N** and **B** of length **K**, how many subsets of array **A** are there so that the xor sum of the subset does not contain as a submask where . Which means , for all .

That means if we choose a subset then

(

or, (

This can be solved using the XOR Basis. Instead of finding a good subset xor sum, we can find the total number of bad subset xor sum and then just subtract it from the total number of subsets.

As we could have duplicate results when calculating bad subsets, we need to carefully subtract them. Let's say we have calculated the number of bad subsets that has as a submask and the number of bad subsets that has as a submask. There could be subsets that has both and as a submask. Hence, we need to subtract them. How can we do that? We know that if a subset xor sum contains both and as submask then and . We can derive a new equation from them

or

That means we need to find the subsets which contains as submask and then subtract it. More formally, we need to use the Inclusion–exclusion principle.

Complexity:

We need to calculate the basis times. Hence, the complexity per test case is where is the number of rows/equations we need to add. This could be as big as 32 because all numbers will fit in a 32 bits signed integer. If we use bitset, the complexity will be reduced by a constant factor of 32/64.

However, the above solution is too slow for the given constraints for this problem. We can actually optimise the solution even further. In our previous solution for each of the basises we are inserting where and is the OR of elements of the chosen subset of the array **B**. In reality, what we are doing is, we are inserting submasks of . Instead of doing this for all , we can actually precalculate the basis of A. Then for each subset of B we need to calculate basis over if the ith bit of is set (where ). If we can make then the number of subsets that can make XOR Sum would be **2(n-rank)**, otherwise 0.

Proof: (Yet to complete)

Complexity of precalculating the basis of A:

Complexity of finding the final answer: , here we are able to reduce N to L.

Overall complexity:

## Problem G: Pess Chuzzle

#### Author: Kazi Md Irshad

#### Alternate writers: Nafis Sadique

#### Explanation:

Category: Constructive Algorithm

If P[i] != i then this rook must make at least 1 horizontal move. If a rook makes one vertical move then it must do another vertical move to come down to the bottom most row.

We claim that no rook makes extra horizontal moves and makes exactly 0 or 2 vertical moves. Let us minimize the number of rooks requiring 2 vertical moves. If such a construction exists then it must be optimal because adding vertical rook moves cannot decrease the fixed requirement of horizontal moves.

Let us call rooks which forever stay in the bottom most row as horizontal rooks and the rest vertical rooks.

We take the longest increasing subsequence L of P. the elements not belonging in L will be our vertical rooks. This maximizes the number of horizontal rooks. If we take more than |L| horizontal rooks then there will exist at least a pair which must cross each other on the bottom row. But that is not possible.

For only the vertical rooks, first we take all i such that i <= P[i]. we process these rooks with decreasing order of P[i]. take rook from (i, 1) to (i, 8) then to (P[i], 8)

Then for the vertical rooks, we take all i such that i >= P[i]. we process in increasing order of P[i]. take rook from (i, 1) to (i, 7) then to (P[i], 7).

Now all vertical rooks have left the bottommost row. Now move the horizontal rooks (maintaining a stack helps) then bring down all vertical rooks. This solution works with only 3 rows and 8 is just a bait.

## Problem H: Island Invasion

#### Author: Pritom Kundu,

#### Alternate writers:Md Sabbir Rahman

#### Explanation:

Category: Geometry, Implementational.

Curse the setter.

## Problem I: Qwiksort

#### Author: Md Sabbir Rahman

#### Alternate writers: Pritom Kundu, Raihat Zaman Niloy

#### Explanation:

Category: Constructive Algorithm, Observation, Easy

We can follow bubble sorting like, first sort [1, n] range then [n/2, n+n/2], then [n+1, 2n], then sort [1, n], then [n/2, n+n/2], then [1, n] again. (when n is odd, will need some extra ops, but will totally require 8 ops)

## Problem J: Point Table

#### Author: Mohammad Ashraful Islam

#### Alternate writers: Hasinur Rahman

#### Explanation:

Category: Observation, Easy

Only 3 pair of matches, AB, BC and CA. Check all possible outcomes.

## Problem K: Strategies in Sequential Games

#### Author: Siam Habib

#### Alternate writers: Kazi Md Irshad, Pritom Kundu

#### Explanation:

Category: DP, DS

There is a brute force **O(nk)** dp solution which will get TLE. We can speed it up to an **O(n)** solution by using virtual trees. [Solution to Problem K](https://docs.google.com/document/d/1E-1BmIdQfPkqBwRDrqDLIZLRMKkGtyIWR9PWhBOraCs/edit?usp=sharing)