1 1A. Gaussian elimination

Discussed during the lecture

2 1B. Binary Gaussian

Firstly, let's notice that binary Gaussian is even easier than the usual one. As coefficients are always 0 or 1, the coefficient by which we multiply the row will be 0 or 1. So, we can drop it and subtract one row from another (of course, if there is 1 in the position where we want to set 0).

After performing Gaussian for the system with m equations for n variables we may end up with some rows that have complete zeros as coefficient and some value in the column $n + 1$. There is no solution if and only if for one of such rows the value in the last column is not zero.

There are multiple solutions if the number of rows with non-zero elements is less than n.

Now back to the problem, here we can say that variables x_i can be 0/1 - if we take the i-th value or not. Now for each bit we have an equation where coefficients are values of this bit in the input and the target value is the value of this bit in the target value.

3 1C. The Road to Berland is Paved With Good Intentions

Source: https://codeforces.com/problemset/problem/228/E

Let's notice that it doesn't make sense to use one city more than once. So, we can have variable x_i — do we use city i or not. Now if road (u, v) has $c = 1$ we get equation $x_u \oplus x_v = 0$ as we either use both of them or none. In case $c = 0$ we get $x_u \oplus x_v = 1$. Now you should simply solve this system.

There is also a solution that doesn't use Gaussian (copied from the editorial of the round):

The vertices from the result call switched-on. Firstly, note that every vertex should be switched-on no more than once. Then, consider every edge (x, y) of color c. We want to make its color 1. So, if $c = 1$ we should switch on vertices x and y or don't switch them on simultaneously. If $c = 0$ we should switch on x or y. So, consider some vertex v and try to switch it on or not. Thus, we can uniquely determine the state of every vertex of the same connected component as v. If we face some collision, we can't get the solution, you should print `Impossible`. The solution can be realized using bfs with complexity $O(N + M)$.

4 2A. Xor queries

As explained in the lecture notes, you can calculate prefix xors:

$$p_i = a_0 \oplus a_1 \oplus \ldots \oplus a_i = p_{i-1} \oplus a_i$$

Now the XOR on the subsegment $[l, r]$ is exactly $p_r \oplus p_{l-1}$. Be careful with the case when $l = 0$.

5 2B. Dima and a Bad XOR

Editorial is copied from the source: https://codeforces.com/contest/1151/problem/B

Let's take the first number in each array.

Then, if we have current XOR strictly greater than zero we can output an answer.

And if there is some array, such that it contains at least two distinct numbers, you can change the first number in this array to number, that differs from it, and get XOR $0 \oplus x > 0$.

Else, each array consists of the same numbers, so all possible XORs are equal to 0, and there is no answer.

Complexity is $O(nm)$.

6 2C. Make Good

Editorial is copied from the source: https://codeforces.com/contest/1270/problem/C

Let the sum of numbers be S, and their $\oplus$ be X.

Solution 1: If $S \leq 2X$ and S was even, it would be enough to add into the array 2 numbers $\frac{2XS}{2}, \frac{2XS}{2}$: X wouldn't change, and the sum would become $2X$.

How to achieve this? Let's add to the array number $250 + (S mod 2)$. If new sum and $\oplus$ of all numbers are S_1 and X_1 correspondently, then we know, that $S_1 \leq 2 \cdot 250 \leq 2X_1$, and S_1 is even. We spent 3 numbers.

Solution 2: It's enough to add to the array numbers X and $S+X$. Indeed, $S+X+(S+X) = 2(S+X)$, and $X \oplus X \oplus (S+X) = (S+X)$. We spent only 2 numbers!

Both solutions have asymptotic $O(n)$

7 2D. Little Xor

Editorial is copied from the source: https://codeforces.com/contest/252/problem/A Let's iterate over all segments in our array. For each of them we'll find the xor of all its elements. Then we need to output the maximal xor we've seen.

8 2E. Mahmoud and Ehab and the xor

Editorial is copied from the source: https://codeforces.com/contest/862/problem/C

$n = 2, x = 0$ is the only case with answer "NO" .

Let $pw = 2^{17}$.

First print $1, 2, \ldots, n-3$ (The first $n-3$ positive integers), Let their bitwise-xor sum be y, If $x = y$ you can add $pw, pw * 2, pw \oplus (pw * 2)$, Otherwise you can add $0, pw, pw \oplus x \oplus y$. We handled the case $x = y$ in a different way because if we add $0, pwpw \oplus x \oplus y$ in this case, Then it's like adding $0, pw, pw$, pw appears twice so we'll get wrong answer.

Handle $n = 1$ (print x) and $n = 2$ (print 0 and x) .

9 3A. Max XOR

If you read the articles about XOR-basis mentioned in the lecture notes, you should know that the answer is greedily XORing all elements in the basis if you keep them ordered by their highest bit.

10 3B. Count different

The answer is 2^k where k is the size of the basis.

11 3C. Mahmoud and Ehab and yet another xor task

Source: https://codeforces.com/contest/959/problem/F Editorial is copied from the second link in the lecture notes.

We can answer the queries online. Iterate through the prefix of the array, and for each prefix remember the basis vectors of that prefix. Then, iterate through the queries. To answer a query, we check if x is actually representable by the prefix of l elements or not. We can do it the same way we add element to the basis but if we find some new bit we return fasle instead of adding element to the basis.

If it's not representable, then the answer to the query is 0. If it is representable, then the answer will be 2^{l-b}, where b is the basis size for the first l elements. It is so, because for each subset of the $l - b$ non-basis vectors in the prefix, we find a unique linear combination to yield xor-sum x.

12 3D. Weird game

Source (but I simplified this problem by removing the statement about Nim-game): https://codeforces.com/cont
You can read editorial (with another brief explanation of XOR-basis) here

13 3E. Ivan and Burgers

Source: https://codeforces.com/contest/1100/problem/F

The main idea is to use Divide-and-Conquer method. Editorial: here