XORting

Time limit: 2 seconds
Memory limit: 1024 MiB

This is an interactive problem.

Iliyan has played a prank on Jonas: he has hidden a permutation $p_0, p_1, \ldots, p_{N-1}$ of the integers from 0 to $N-1$. Jonas is desperate to recover it, and you have agreed to help him.

Iliyan will not reveal the permutation outright, but he has agreed to answer questions of the following kind: you can choose two indices i and j such that $0 \le i, j < N$, and he tells you the bitwise XOR (exclusive OR) of p_i and p_j.

> The bitwise XOR (exclusive OR) of two numbers, denoted by $\oplus$, performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if exactly one of the bits is 1, but will be 0 if both are 0 or both are 1. For example, $75 \oplus 29 = 86$, since $1001011_{(2)} \oplus 0011101_{(2)} = 1010110_{(2)}$.
>
> In C++, you may calculate the XOR of `a` and `b` by using the `^` operator.

Iliyan is willing to answer **at most** Q questions.

However, these questions cannot always determine the hidden permutation uniquely. We define a permutation $a_0, a_1, \ldots, a_{N-1}$ to be *indistinguishable* from p if $a_i \oplus a_j = p_i \oplus p_j$ for every pair of indices i and j such that $0 \le i, j < N$. Every answer Iliyan gives is identical whether the hidden permutation is p or a, so no sequence of questions can tell the two apart. Note that p is always indistinguishable from itself, and that there may be no other such permutation.

Your task is to find the lexicographically smallest permutation indistinguishable from p.

> A permutation $x_0, x_1, \ldots, x_{N-1}$ is lexicographically smaller than $y_0, y_1, \ldots, y_{N-1}$ if, at the first position k where they differ, $x_k < y_k$. For example, $x = [2, 0, 1, 4, 3]$ is lexicographically smaller than $y = [2, 0, 3, 1, 4]$, because the first position at which they differ is $k = 2$, and $x_2 = 1 < 3 = y_2$.

The hidden permutation is fixed before your program starts and does not change in response to your questions.

Implementation details

Your program must implement the function `solve`:

```
std::vector<int> solve(int N)
```

- N: the number of elements of the permutation.

This function is called exactly T times, once for each hidden permutation. It must return the lexicographically smallest permutation indistinguishable from the permutation hidden for that call.

To communicate with the jury's program, you are provided with the function:

```
int get_xor(int i, int j)
```

Each time the function `get_xor` is called, it returns $p_i \oplus p_j$. Both arguments must be valid indices of the permutation. If this condition is not met, your solution will receive the verdict `Output isn't correct: Invalid argument`. You can assume that the function responds in constant time $O(1)$.

The value of Q is fixed for each subtask and is not passed to your program.

Constraints

- $1 \leq N_i \leq 2^{20}$ for every $1 \leq i \leq T$, where N_i is the value of N in the i-th call to `solve`
- $1 \leq T \leq 2^{10}$
- $T \cdot \max(N_1, N_2, \ldots, N_T) \leq 2^{25}$.
- During the i-th call of `solve` in a test you may ask at most Q_i questions, where Q_i is either N_i or N_i^2, depending on the subtask.

Example

Consider the following interaction, in which `solve` is called twice:

Participant program	Jury program
	`solve(4)`
`get_xor(0, 1)`	`2`
`get_xor(0, 2)`	`3`
`get_xor(0, 3)`	`1`
`get_xor(1, 2)`	`1`
`get_xor(1, 3)`	`3`
`get_xor(2, 3)`	`2`
`return {0, 2, 3, 1}`	
	`solve(1)`
`return {0}`	

Explanation of the first call

The hidden permutation is $[2, 0, 1, 3]$, but the lexicographically smallest one indistinguishable from it is $[0, 2, 3, 1]$. Six questions were asked for $N_1 = 4$, which is allowed because $Q_1 = N_1^2 = 16$ in this subtask.

Explanation of the second call

The only possible permutation with $N_2 = 1$ is $[0]$.

Subtasks

Subtask	Points	N_i	T	Q_i	Additional constraints
0	0	-	-	N_i^2	The example.
1	7	$\leq 2^3$	2^{10}	N_i^2	-
2	18	$\leq 2^7$	2^5	N_i^2	-
3	5	$\leq 2^{11}$	2^5	N_i^2	-
4	5	$\leq 2^{11}$	2^5	N_i	-
5	5	$\leq 2^{18}$	2^5	N_i	$N_i = 2^k$ for some integer k.
6	5	$\leq 2^{18}$	2^5	N_i	N_i is odd.
7	30	$\leq 2^{18}$	2^5	N_i	-
8	25	$\leq 2^{20}$	2^5	N_i	-

Sample grader

The input format is the following:

- line 1: two integers – the values of T and Q_{type}, where Q_{type} is either 1 or 2;
- the following $2T$ lines describe the tests:
  - line $2i$: one integer – the value of N_i for the i-th test;
  - line $2i + 1$: N_i integers – the permutation $p_0, p_1, \ldots, p_{N_i - 1}$ for the i-th test.

If $Q_{type} = 1$, then $Q_i = N_i$ for the i-th call of `solve`; if $Q_{type} = 2$, then $Q_i = N_i^2$.

The output format is the following:

- line i: the permutation returned by your program on the i-th test.