

XORting

Time limit: 2 seconds

Memory limit: 1024 MiB

Это интерактивная задача.

Илиян подшутил над Йонасом: он спрятал перестановку $p_0, p_1, \dots, p_{N-1}$ целых чисел от 0 до $N - 1$. Йонас отчаянно хочет её восстановить, и вы согласились ему помочь.

Илиян не будет раскрывать перестановку напрямую, но согласился отвечать на вопросы следующего вида: вы можете выбрать два индекса i и j такие, что $0 \leq i, j < N$, и он скажет вам побитовое исключающее ИЛИ (XOR) чисел p_i и p_j .

Побитовое исключающее ИЛИ (XOR) двух чисел, обозначаемое $\oplus$, выполняет логическую операцию исключающего ИЛИ над каждой парой соответствующих битов. Для каждой позиции, если ровно один из битов равен 1, то результат равен 1. А если оба бита равны 0 или оба равны 1, то результат равен 0. Например, $75 \oplus 29 = 86$, потому что $1001011_{(2)} \oplus 0011101_{(2)} = 1010110_{(2)}$.

В C++ вы можете вычислить XOR a и b , используя оператор $\wedge$

Илиян готов ответить максимум на Q вопросов.

Однако эти вопросы не всегда могут однозначно определить скрытую перестановку. Мы определяем перестановку $a_0, a_1, \dots, a_{N-1}$ как *неразличимую* от p , если $a_i \oplus a_j = p_i \oplus p_j$ для каждой пары индексов i и j таких, что $0 \leq i, j < N$. Независимо от того, является ли скрытая перестановка p или a , все ответы Илияна будут одинаковы и поэтому никакая последовательность вопросов не может их различить. Заметим, что p всегда неразличима от самой себя, и что может не быть других неразличимых перестановок.

Ваша задача — найти лексикографически наименьшую перестановку, неразличимую от p .

Перестановка $x_0, x_1, \dots, x_{N-1}$ лексикографически меньше $y_0, y_1, \dots, y_{N-1}$, если на первой позиции k , в которой они различаются, $x_k < y_k$. Например, $x = [2, 0, 1, 4, 3]$ лексикографически меньше $y = [2, 0, 3, 1, 4]$, потому что первая позиция, в которой они различаются, это $k = 2$, и $x_2 = 1 < 3 = y_2$.

Скрытая перестановка зафиксирована до запуска вашей программы и не меняется в зависимости от ваших вопросов.

Implementation details

Ваша программа должна реализовать функцию `solve`:

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

- N : количество элементов перестановки.

Эта функция вызывается ровно T раз программой жюри, один раз для каждой скрытой перестановки. Она должна вернуть лексикографически наименьшую перестановку, неразличимую от скрытой перестановки этого вызова.

Для связи с программой жюри вам предоставляется функция:

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

Каждый раз при вызове функции `get_xor` она возвращает $p_i \oplus p_j$. Оба аргумента должны быть допустимыми индексами перестановки. Если это условие не выполняется, ваше решение получит вердикт `Output isn't correct: Invalid argument`. Вы можете предположить, что функция отвечает за время $O(1)$.

Для каждой подзадачи Q имеет фиксированное значение, которое ваша программа не получает.

Constraints

- $1 \leq N_i \leq 2^{20}$ для каждого $1 \leq i \leq T$, где N_i — значение N в i -м вызове `solve`
- $1 \leq T \leq 2^{10}$
- $T \cdot \max(N_1, N_2, \dots, N_T) \leq 2^{25}$.
- Во время i -го вызова `solve` в тесте вы можете задать максимум Q_i вопросов, где Q_i равно либо N_i , либо N_i^2 , в зависимости от подзадачи.

Example

Рассмотрим следующее взаимодействие, в котором `solve` вызывается дважды:

Participant program	Jury program
	<code>solve(4)</code>
<code>get_xor(0, 1)</code>	2
<code>get_xor(0, 2)</code>	3
<code>get_xor(0, 3)</code>	1
<code>get_xor(1, 2)</code>	1
<code>get_xor(1, 3)</code>	3
<code>get_xor(2, 3)</code>	2
<code>return {0, 2, 3, 1}</code>	
	<code>solve(1)</code>
<code>return {0}</code>	

Объяснение первого вызова

Скрытая перестановка — $[2, 0, 1, 3]$, но лексикографически наименьшая из неразличимых с ней — $[0, 2, 3, 1]$. Для $N_1 = 4$ было задано шесть вопросов, что допустимо, потому что $Q_1 = N_1^2 = 16$ в этой подзадаче.

Объяснение второго вызова

Единственная возможная перестановка с $N_2 = 1$ — это $[0]$.

Subtasks

Subtask	Points	N_i	T	Q_i	Additional constraints
0	0	—	—	N_i^2	Пример.
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	$N_i = 2^k$ для некоторого целого числа k .
6	5	$\leq 2^{18}$	2^5	N_i	N_i нечетное
7	30	$\leq 2^{18}$	2^5	N_i	—
8	25	$\leq 2^{20}$	2^5	N_i	—

Sample grader

Формат входных данных следующий:

- строка 1: два целых числа — значения T и Q_{type} , где Q_{type} равно 1 или 2;
- следующие $2T$ строк описывают тесты:
 - строка $2i$: одно целое число — значение N для i -го теста;
 - строка $2i + 1$: N целых чисел — перестановка $p_0, p_1, \dots, p_{N-1}$ для i -го теста.

Если $Q_{type} = 1$, то $Q_i = N_i$ для i -го вызова `solve`; если $Q_{type} = 2$, то $Q_i = N_i^2$.

Формат выходных данных следующий:

- строка i : перестановка, возвращённая вашей программой на i -м тесте.