XOR-тировка

Време: 2 секунди
Памет: 1024 MiB

Това е интерактивна задача.

Илиян е направил номер на Йонас: скрил е пермутация $p_0, p_1, \ldots, p_{N-1}$ на целите числа от 0 до $N-1$. Йонас е твърдо решен да я възстанови, а ти вече си се съгласил да му помогнеш.

Разбира се, Илиян няма да разкрие пермутацията веднага, но е съгласен да отговаря на въпроси от следния вид: по избрани два индекса i и j такива че $0 \leq i, j < N$, той ще ти каже побитовото XOR (изключващо ИЛИ) на p_i и p_j.

> Побитовото XOR (изключващо ИЛИ) на две числа, означавано с $\oplus$, извършва логическата операция изключващо ИЛИ върху всяка двойка съответстващи си битове. Резултатът на всяка позиция е 1, ако точно един от битовете е 1, и е 0, ако и двата са 0 или и двата са 1. Например $75 \oplus 29 = 86$, тъй като $1001011_{(2)} \oplus 0011101_{(2)} = 1010110_{(2)}$.
>
> В C++ можете да изчислявате XOR на `a` и `b`, като използвате оператора `^`.

Илиян е готов да отговори на **най-много** Q въпроса.

Тези въпроси обаче не винаги могат да определят скритата пермутация еднозначно. Дефинираме, че пермутацията $a_0, a_1, \ldots, a_{N-1}$ е *неразличима* от p, ако $a_i \oplus a_j = p_i \oplus p_j$ за всяка двойка индекси i и j, такива че $0 \leq i, j < N$. Всеки отговор, който Илиян дава, е един и същ, независимо дали скритата пермутация е p или a, така че никаква последователност от въпроси не може да ги различи. Забележете, че p винаги е неразличима от самата себе си, както и че може да няма друга такава пермутация.

Вашата задача е да намерите лексикографски най-малката пермутация, неразличима от p.

> Една пермутация $x_0, x_1, \ldots, x_{N-1}$ е лексикографски по-малка от $y_0, y_1, \ldots, y_{N-1}$, ако на първата позиция k, където те се различават, е изпълнено $x_k < y_k$. Например, $x = [2, 0, 1, 4, 3]$ е лексикографски по-малка от $y = [2, 0, 3, 1, 4]$, защото първата позиция, на която те се различават, е $k = 2$, и $x_2 = 1 < 3 = y_2$.

Скритата пермутация е фиксирана преди стартирането на вашата програма и не се променя на база вашите въпроси.

Детайли по имплементацията

Вашата програма трябва да имплементира функцията `solve`:

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

- N: броят на елементите на пермутацията.

За всеки тест, функцията ще бъде извикана точно T пъти от програмата на журито, по веднъж за всяка скрита пермутация. За всяко извикване, тя трябва да върне лексикографски най-малката пермутация, неразличима от съответната скрита пермутация.

За да комуникирате с програмата на журито, ви е предоставена следната функция:

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

Всеки път, когато функцията `get_xor` бъде извикана, тя връща $p_i \oplus p_j$. И двата аргумента трябва да бъдат валидни индекси на пермутацията. Ако това условие не е изпълнено, вашето решение ще получи резултат `Output isn't correct: Invalid argument`. Може да приемете, че функцията отговаря за константно време $O(1)$.

Стойността на Q е фиксирана за всяка подзадача, но не се подава на вашата програма.

Ограничения

- $1 \leq N_i \leq 2^{20}$ за всяко $1 \leq i \leq T$, където N_i е стойността на N при i-тото извикване на функцията `solve`
- $1 \leq T \leq 2^{10}$
- $T \cdot \max(N_1, N_2, \ldots, N_T) \leq 2^{25}$.
- За i-тото извикване на функцията `solve` в даден тест, може да задавате най-много Q_i въпроса, където Q_i е или N_i, или N_i^2, в зависимост от подзадачата.

Пример

Нека разгледаме следното взаимодействие, при което функцията `solve` е извикана два пъти.

Участник	Жури
	`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}`	

Обяснение на първото извикване

Скритата пермутация е $[2, 0, 1, 3]$, но лексикографски най-малката, която е неразличима от нея, е $[0, 2, 3, 1]$. Зададени са шест въпроса при $N_1 = 4$, което е позволено, защото $Q_1 = N_1^2 = 16$ в тази подзадача.

Обяснение на второто извикване

Единствената възможна пермутация с $N_2 = 1$ е $[0]$.

Подзадачи

Подзадача	Точки	N_i	T	Q_i	Допълнителни ограничения
0	0	-	-	N_i^2	Примерът от условието.
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	$N_i = 2^k$ за някое цяло число k.
6	5	$\leq 2^{18}$	2^5	N_i	N_i е нечетно.
7	30	$\leq 2^{18}$	2^5	N_i	-
8	25	$\leq 2^{20}$	2^5	N_i	-

Примерен грейдър

Входният формат е следният:

- ред 1: две цели числа – стойностите на T и Q_{type}, където Q_{type} е или 1, или 2;
- следващите $2T$ реда описват тестовете:
  - ред $2i$: едно цяло число – стойността на N_i за i-тия тест;
  - ред $2i + 1$: N_i цели числа – пермутацията $p_0, p_1, \ldots, p_{N_i - 1}$ за i-тия тест.

Ако $Q_{type} = 1$, тогава $Q_i = N_i$ за i-тото извикване на `solve`; ако $Q_{type} = 2$, тогава $Q_i = N_i^2$.

Изходният формат е следният:

- ред i: пермутацията, върната от вашата програма при i-тия тест.