

XORування

Обмеження часу: 2 секунди

Обмеження пам'яті: 1024 MiB

Це інтерактивна задача.

Ілля розіграв Дашу: він сховав перестановку $p_0, p_1, \dots, p_{N-1}$ цілих чисел від 0 до $N - 1$. Даша відчайдушно намагається її відновити, і ви погодилися їй допомогти.

Ілля не розкриє перестановку безпосередньо, але погодився відповідати на запитання такого типу: ви можете вибрати два індекси i та j такі, що $0 \leq i, j < N$, а він повідомить вам побітове XOR (виключне АБО) чисел p_i та p_j .

Побітове XOR (виключне АБО) двох чисел, що позначається символом $\oplus$, виконує логічну операцію виключного АБО над кожною парою відповідних бітів. Результат у кожній позиції дорівнює 1, якщо рівно один із бітів дорівнює 1, і дорівнює 0, якщо обидва біти дорівнюють 0 або обидва дорівнюють 1. Наприклад, $75 \oplus 29 = 86$, оскільки $1001011_{(2)} \oplus 0011101_{(2)} = 1010110_{(2)}$.

У C++ ви можете обчислити XOR чисел a та b за допомогою оператора $\wedge$.

Ілля готовий відповісти **щонайбільше** на Q запитань.

Однак ці запитання не завжди дають змогу однозначно визначити приховану перестановку. Назвемо перестановку $a_0, a_1, \dots, a_{N-1}$ як таку, що неможливо відрізнити від p (нерозрізнювальною), якщо $a_i \oplus a_j = p_i \oplus p_j$ для кожної пари індексів i та j таких, що $0 \leq i, j < N$. Кожна відповідь Іллі буде однаковою незалежно від того, чи є прихованою перестановкою p або a , тому жодна послідовність запитань не дасть змоги розрізнити ці дві перестановки. Зауважте, що p завжди нерозрізнювальна сама із собою, а інших таких перестановок може не існувати.

Ваше завдання — знайти лексикографічно найменшу перестановку, нерозрізнювальну з p .

Перестановка $x_0, x_1, \dots, x_{N-1}$ є лексикографічно меншою за $y_0, y_1, \dots, y_{N-1}$, якщо в першій позиції k , у якій вони відрізняються, виконується $x_k < y_k$. Наприклад, $x = [2, 0, 1, 4, 3]$ є лексикографічно меншою за $y = [2, 0, 3, 1, 4]$, оскільки перша позиція, у якій вони відрізняються, — це $k = 2$, і $x_2 = 1 < 3 = y_2$.

Прихована перестановка фіксується до початку роботи вашої програми й не змінюється у відповідь на ваші запитання.

Деталі реалізації

Ваша програма повинна реалізувати функцію `solve`:

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

- N : кількість елементів перестановки.

Ця функція викликається рівно T разів — по одному разу для кожної прихованої перестановки. Вона повинна повернути лексикографічно найменшу перестановку, нерозрізнявальну з перестановкою, прихованою під час відповідного виклику.

Для взаємодії з програмою журі вам надається функція:

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

Кожного разу, коли викликається функція `get_xor`, вона повертає $p_i \oplus p_j$. Обидва аргументи повинні бути коректними індексами перестановки. Якщо ця умова не виконується, ваше рішення отримає вердикт `Output isn't correct: Invalid argument`. Можете вважати, що функція працює за сталий час $O(1)$.

Значення Q є фіксованим для кожної підзадачі й не передається вашій програмі.

Обмеження

- $1 \leq N_i \leq 2^{20}$ для кожного $1 \leq i \leq T$, де N_i — значення N під час i -го виклику `solve`
- $1 \leq T \leq 2^{10}$
- $T \cdot \max(N_1, N_2, \dots, N_T) \leq 2^{25}$.
- Під час i -го виклику `solve` у тесті ви можете поставити щонайбільше Q_i запитань, де Q_i дорівнює або N_i , або N_i^2 , залежно від підзадачі.

Приклад

Розглянемо таку взаємодію, у якій `solve` викликається двічі:

Програма учасника	Програма журі
	<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>	

Пояснення першого виклику

Прихованою перестановкою є $[2, 0, 1, 3]$, але лексикографічно найменшою перестановкою, нерозрізнявальною з нею, є $[0, 2, 3, 1]$. Для $N_1 = 4$ було поставлено шість запитань, що дозволено, оскільки в цій підзадачі $Q_1 = N_1^2 = 16$.

Пояснення другого виклику

Єдиною можливою перестановкою для $N_2 = 1 \in [0]$.

Підзадачі

Підзадача	Бали	N_i	T	Q_i	Додаткові обмеження
0	0	-	-	N_i^2	Приклад.
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 деякого цілого числа k .
6	5	$\leq 2^{18}$	2^5	N_i	N_i є непарним.
7	30	$\leq 2^{18}$	2^5	N_i	-
8	25	$\leq 2^{20}$	2^5	N_i	-

Приклад градера

Формат вхідних даних:

- рядок 1: два цілих числа — значення T та Q_{type} , де Q_{type} дорівнює або 1, або 2;
- наступні $2T$ рядків описують тести:
 - рядок $2i$: одне ціле число — значення N_i для i -го тесту;
 - рядок $2i + 1$: N_i цілих чисел — перестановка $p_0, p_1, \dots, p_{N_i-1}$ для i -го тесту.

Якщо $Q_{type} = 1$, то $Q_i = N_i$ для i -го виклику `solve`; якщо $Q_{type} = 2$, то $Q_i = N_i^2$.

Формат вихідних даних:

- рядок i : перестановка, повернена вашою програмою на i -му тесті.