

XORting

Временско ограничење: 2 секунде

Меморијско ограничење: 1024 MiB

Ово је интерактивни задатак.

Илијан је направио шалу на Јонасов рачун: сакрио је пермутацију $p_0, p_1, \dots, p_{N-1}$ целих бројева од 0 до $N - 1$. Јонас се мучи да је пронађе и ти си пристао да му помогнеш.

Илијан неће одмах открити пермутацију, али је пристао да одговара на питања следеће врсте: можеш да изабереш два индекса i и j таква да је $0 \leq i, j < N$ и он ће ти рећи резултат битовског XOR-а (ексклузивне дисјункције) бројева p_i и p_j .

Битовски XOR (ексклузивна дисјункција) два броја, у ознаци $\oplus$, врши ексклузивну дисјункцију над сваким паром бита. Резултат на свакој позицији је 1 ако је тачно један улазни бит 1, иначе 0 ако су оба бита 0 или 1. На пример, $75 \oplus 29 = 86$, јер је $1001011_{(2)} \oplus 0011101_{(2)} = 1010110_{(2)}$. У језику C++ XOR бројева a и b може се израчунати коришћењем оператора $\wedge$.

Илијан је вољан да одговори на **највише** Q питања.

Ипак, ова питања не могу увек једнозначно да одреде скривену пермутацију. Кажемо да је пермутација $a_0, a_1, \dots, a_{N-1}$ **неразлучива** (немогуће направити разлику) од пермутације p ако је $a_i \oplus a_j = p_i \oplus p_j$ за сваки пар индекса i и j таквих да је $0 \leq i, j < N$. Сваки Илијанов одговор је идентичан било да је скривена пермутација p или a , према томе не постоји низ питања која могу да их разликују. Приметимо да је p увек неразлучива од саме себе и да можда нема друге такве пермутације.

Твој задатак је да пронађеш лексикографски најмању пермутацију неразлучиву од p .

Пермутација $x_0, x_1, \dots, x_{N-1}$ је лексикографски мања од $y_0, y_1, \dots, y_{N-1}$ ако за прву позицију k на којој се разликују важи $x_k < y_k$. На пример, $x = [2, 0, 1, 4, 3]$ је лексикографски мања од $y = [2, 0, 3, 1, 4]$, јер је прва позиција на којој се разликују $k = 2$ и важи $x_2 = 1 < 3 = y_2$.

Скривена пермутација је фиксирана пре него што твој програм крене да се извршава и не мења се током одговарања на твоја питања.

Детаљи имплементације

Твој програм мора да имплементира функцију `solve`:

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

- N : број елемената пермутације.

Ова функција се позива тачно T пута, једном за сваку скривену пермутацију. Она мора да врати лексикографски најмању пермутацију неразлучиву од скривене пермутације у том позиву.

За комуникацију са жиријевим програмом на располагању ти је функција:

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

Сваки пут када се функција `get_xor` позове, она враћа $p_i \oplus p_j$. Оба аргумента морају бити валидни индекси пермутације. Ако овај услов није испуњен, твоје решење ће добити одговор `Output isn't correct: Invalid argument`. Можеш да претпоставиш да функција одговара у константном времену $O(1)$.

Вредност Q је фиксирана за сваки подзадатак и не прослеђује се твој програму.

Ограничења

- $1 \leq N_i \leq 2^{20}$ за све $1 \leq i \leq T$, где је N_i вредност N у i -том позиву функције `solve`
- $1 \leq T \leq 2^{10}$
- $T \cdot \max(N_1, N_2, \dots, N_T) \leq 2^{25}$.
- Током i -тог позива функције `solve` у тесту можеш да поставиш највише Q_i питања, где је Q_i или N_i или N_i^2 , у зависности од подзадатка.

Пример

Размотримо следећу интеракцију у којој се функција `solve` позива двапут:

Твој програм	Жиријев програм
	<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>	

Објашњење првог позива

Скривена пермутација је $[2, 0, 1, 3]$, али је лексикографски најмања неразлучива од ње $[0, 2, 3, 1]$. Шест питања је постављено за $N_1 = 4$, што је дозвољено јер је $Q_1 = N_1^2 = 16$ у овом подзадатку.

Објашњење другог позива

Једина могућа пермутација за $N_2 = 1$ је $[0]$.

Подзадаци

Подзадатак	Поени	N_i	T	Q_i	Додатна ограничења
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	$N_i = 2^k$ за неки цео број k .
6	5	$\leq 2^{18}$	2^5	N_i	N_i је непарно.
7	30	$\leq 2^{18}$	2^5	N_i	-
8	25	$\leq 2^{20}$	2^5	N_i	-

Пример грејдера

Формат улаза је следећи:

- линија 1: два цела броја – вредности T и Q_{type} , где је Q_{type} или 1 или 2;
- наредних $2T$ линија описују тестове:
 - линија $2i$: један цео број – вредност N_i за i -ти тест;
 - линија $2i + 1$: N_i целих бројева – пермутација $p_0, p_1, \dots, p_{N_i-1}$ за i -ти тест.

Ако је $Q_{type} = 1$, онда је $Q_i = N_i$ за i -ти позив функције `solve`; ако је $Q_{type} = 2$, онда је $Q_i = N_i^2$.

Формат излаза је следећи:

- линија i : пермутација коју враћа твој програм за i -ти тест.