

XORting

Időlimit: 2 másodperc

Memórialimit: 1024 MiB

Ez egy interaktív feladat.

Iliyan megviccelte Jonast: kiválasztotta a 0 és $N - 1$ közötti egész számoknak egy titkos $p_0, p_1, \dots, p_{N-1}$ permutációját. Jonas szeretné kitalálni a permutációt, és te beleegyeztél, hogy segítesz neki.

Iliyan nem fogja megmondani a permutációt, de a következő típusú kérdésekre válaszol: kiválaszthatsz két indexet, i -t és j -t úgy, hogy $0 \leq i, j < N$, és ő megmondja neked a p_i és p_j számok XOR (kizáró vagy) értékét.

Két szám XOR-ját (kizáró vagy-át) a $\oplus$ művelet jelöli, ami az azonos helyiértékű bitekre a logikai kizáró vagy műveletet hajtja végre. Az eredmény az egyes pozíciókban 1, ha pontosan az egyik bit 1, viszont 0, ha mindkettő 0 vagy mindkettő 1. Például $75 \oplus 29 = 86$, mivel $1001011_{(2)} \oplus 0011101_{(2)} = 1010110_{(2)}$.

C++-ban az a és b XOR-értékét a $\wedge$ operátor segítségével számíthatjuk ki.

Iliyan **legfeljebb** Q kérdésre hajlandó válaszolni.

De sajnos ezekkel a kérdésekkel nem mindig lehet a titkos permutációt egyértelműen meghatározni. Egy $a_0, a_1, \dots, a_{N-1}$ permutációt akkor tekintünk *megkülönböztethetetlennek* p -től, ha minden olyan i és j párra, amelyre $0 \leq i, j < N$, teljesül, hogy $a_i \oplus a_j = p_i \oplus p_j$. Vagyis minden Iliyan által adott válasz azonos, függetlenül attól, hogy a titkos permutáció p vagy a , így egyetlen kérdéssorozat sem képes megkülönböztetni a két permutációt. Definíció szerint p mindig megkülönböztethetetlen önmagától, és lehet, hogy nincs más ilyen permutáció.

A feladatod a lexikografikusan legkisebb, p -től megkülönböztethetetlen permutáció megtalálása.

Az $x_0, x_1, \dots, x_{N-1}$ permutáció lexikografikusan kisebb, mint az $y_0, y_1, \dots, y_{N-1}$ permutáció, ha az első olyan k pozíció, ahol eltérnek, ott $x_k < y_k$ teljesül. Például $x = [2, 0, 1, 4, 3]$ lexikografikusan kisebb, mint $y = [2, 0, 3, 1, 4]$, mert az első olyan pozíció, ahol eltérnek, az $k = 2$, és $x_2 = 1 < 3 = y_2$.

A p permutáció mindig előre rögzített és a futtatás során nem változik.

Megvalósítás

A `solve` függvényt kell megvalósítanod:

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

- N : a permutáció elemeinek száma.

Ez a függvény T -szer lesz meghívva, minden titkos p permutációra egyszer. A függvénynek vissza kell adnia a lexikografikusan legkisebb, p -tól megkülönböztethetetlen permutációt.

Az értékelőprogrammal való kommunikációhoz a következő függvény áll rendelkezésedre:

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

A `get_xor` függvény minden egyes meghívásakor a $p_i \oplus p_j$ értéket adja vissza. Mindkét paraméternek a permutáció érvényes indexének kell lennie. Ha ez nem teljesül, akkor az `Output isn't correct: Invalid argument` értékelést kapod. Feltehető, hogy a függvény $O(1)$ időben válaszol.

Q értéke minden részfeladatnál rögzített, és ez nem kerül bele a programodba.

Korlátok

- $1 \leq N_i \leq 2^{20}$ minden $0 < i \leq T$ esetén, ahol N_i az i -edik `solve` hívás során N értéke
- $1 \leq T \leq 2^{10}$
- $T \cdot \max(N_1, N_2, \dots, N_T) \leq 2^{25}$.
- Egy tesztben a `solve` függvény i -edik meghívása során legfeljebb Q_i kérdést tehetsz fel, ahol Q_i a részfeladattól függően N_i , vagy N_i^2 .

Példa

Nézzük a következő interakciót, amelyben a `solve` függvényt kétszer hívják meg:

Saját programod	Értékelőprogram
	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}	

Az első példa magyarázata:

A titkos permutáció $[2, 0, 1, 3]$, de a lexikografikusan legkisebb, p -től megkülönböztethetetlen permutáció $[0, 2, 3, 1]$. Hat kérdést tettünk fel $N_1 = 4$ esetén, ami megengedett, mivel ebben a részfeladatban $Q_1 = N_1^2 = 16$.

A második példa magyarázata:

Az egyetlen lehetséges permutáció $N_2 = 1$ esetén a $[0]$.

Részfeladatok

Részfeladat	Pontszám	N_i	T	Q_i	További korlátok
0	0	-	-	N_i^2	A példa.
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	$N_i = 2^k$ valamilyen k egész esetén.
6	5	$\leq 2^{18}$	2^5	N_i	N_i páratlan.
7	30	$\leq 2^{18}$	2^5	N_i	-
8	25	$\leq 2^{20}$	2^5	N_i	-

Mintaértékelő

A bemenet formátuma a következő:

- 1. sor: két egész szám – a T és a Q_{type} értékei, ahol Q_{type} értéke 1 vagy 2;
- a következő $2T$ sor a teszteseteket írja le:
 - $2i$. sor: egy egész szám – az i -edik teszteset N_i értéke;
 - $2i + 1$. sor: N_i egész szám – az i -edik tesztben a p permutáció: $p_0, p_1, \dots, p_{N_i-1}$.

Ha $Q_{type} = 1$, akkor $Q_i = N_i$ a `solve` i -edik hívásakor; ha $Q_{type} = 2$, akkor $Q_i = N_i^2$.

A kimenet formátuma a következő:

- i -edik sor: a program által az i -edik tesztnél visszaadott permutáció.