XORტირება

დროის ლიმიტი: 2 წამი
მეხსიერების ლიმიტი: 1024 MiB

ეს არის ინტერაქტიული ამოცანა.

ილიანმა იონასს ხუმრობა მოუწყო: მან დამალა 0-დან $(N-1)$-მდე მთელი რიცხვების პერმუტაცია $p_0, p_1, \ldots, p_{N-1}$. იონასს ძალიან სურს მისი აღდგენა და შენ დათანხმდი, რომ დაეხმარები მას.

ილიანი პირდაპირ არ გაამჟღავნებს პერმუტაციას, მაგრამ დათანხმდა, რომ უპასუხებს შემდეგი სახის კითხვებს: შენ შეგიძლია აირჩიო ორი ინდექსი i და j, ისეთი, რომ $0 \le i, j < N$ და ის გეტყვის p_i-ისა და p_j-ის ბიტურ XOR-ს (გამომრიცხავი OR).

> ორი რიცხვის ბიტური XOR (გამომრიცხავი OR), რომელიც აღინიშნება როგორც $\oplus$, ასრულებს ლოგიკურ გამომრიცხავ OR ოპერაციას თითოეულ შესაბამის ბიტთა წყვილზე. შედეგი თითოეულ პოზიციაზე არის 1, თუ ბიტებიდან ზუსტად ერთია 1, მაგრამ იქნება 0, თუ ორივე არის 0 ან ორივე არის 1. მაგალითად, $75 \oplus 29 = 86$, ვინაიდან $1001011_{(2)} \oplus 0011101_{(2)} = 1010110_{(2)}$.
>
> C++-ში, თქვენ შეგიძლიათ გამოთვალოთ `a`-სა და `b`-ს XOR `^` ოპერატორის გამოყენებით.

ილიანი მზადაა უპასუხოს **მაქსიმუმ** Q კითხვას.

თუმცა, ეს კითხვები ყოველთვის ვერ დაადგენს დამალულ პერმუტაციას ცალსახად. ჩვენ ვამბობთ, რომ პერმუტაცია $a_0, a_1, \ldots, a_{N-1}$ *განურჩეველია* p-სგან, თუ $a_i \oplus a_j = p_i \oplus p_j$ ინდექსების ყოველი ისეთი i და j წყვილისთვის, რომ $0 \le i, j < N$. ასეთ შემთხვევაში, ილიანის ყოველი პასუხი ერთი და იგივეა მიუხედავად იმისა, დამალული გადანაცვლება p არის თუ a. ამიტომ, კითხვების არცერთ მიმდევრობას არ შეუძლია ამ ორის ერთმანეთისგან გარჩევა. გაითვალისწინეთ, რომ p ყოველთვის განურჩეველია საკუთარი თავისგან და შესაძლოა სხვა ასეთი გადანაცვლება არ არსებობდეს.

თქვენი ამოცანაა იპოვოთ ლექსიკოგრაფიულად უმცირესი გადანაცვლება, რომელიც განურჩეველია p-სგან.

> პერმუტაცია $x_0, x_1, \ldots, x_{N-1}$ ლექსიკოგრაფიულად ნაკლებია, ვიდრე $y_0, y_1, \ldots, y_{N-1}$, თუ პირველ პოზიცია k-ზე, სადაც ისინი განსხვავდებიან, $x_k < y_k$. მაგალითად, $x = [2, 0, 1, 5, 3]$ ლექსიკოგრაფიულად ნაკლებია, ვიდრე $y = [2, 0, 3, 1, 5]$, რადგან პირველი პოზიცია, სადაც ისინი განსხვავდებიან, არის $k = 2$ და $x_2 = 1 < 3 = y_2$.

დამალული გადანაცვლება დაფიქსირებულია თქვენი პროგრამის დაწყებამდე და არ იცვლება თქვენი დასმული კითხვების მიხედვით.

იმპლემენტაციის დეტალები

თქვენმა პროგრამამ იმპლემენტაცია უნდა გაუკეთოს ფუნქციას `solve`:

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

- N: პერმუტაციის ელემენტების რაოდენობა.

ეს ფუნქცია ჟიურის პროგრამის მიერ გამოიძახება ზუსტად T-ჯერ, თითოჯერ თითოეული დამალული პერმუტაციისთვის. მან უნდა დააბრუნოს ლექსიკოგრაფიულად უმცირესი პერმუტაცია, რომელიც განურჩეველია იმ გამოძახებისთვის დამალული პერმუტაციისგან.

ჟიურის პროგრამასთან კომუნიკაციისთვის თქვენ გეძლევათ ფუნქცია:

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

ყოველ ჯერზე, როცა ფუნქცია `get_xor` გამოიძახება, ის აბრუნებს $p_i \oplus p_j$-ს. ორივე არგუმენტი უნდა იყოს პერმუტაციის სწორი ინდექსი. თუ ეს პირობა არ სრულდება, თქვენი ამონახსნი მიიღებს ვერდიქტს `Output isn't correct: Invalid argument`. შეგიძლიათ ჩათვალოთ, რომ ფუნქცია პასუხს იძლევა მუდმივ დროში $O(1)$.

Q-ს მნიშვნელობა ფიქსირებულია თითოეული ქვეამოცანისთვის და არ გადაეცემა თქვენს პროგრამას.

შეზღუდვები

- $1 \leq N_i \leq 2^{20}$ ყოველი $0 \leq i < T$-სათვის, სადაც N_i არის N-ის მნიშვნელობა `solve`-ის i-ურ გამოძახებაში
- $1 \leq T \leq 2^{10}$
- $T \cdot \max(N_0, N_1, \ldots, N_{T-1}) \leq 2^{25}$.
- ტესტში `solve`-ის i-ური გამოძახებისას შეგიძლიათ დასვათ არაუმეტეს Q შეკითხვისა, სადაც Q არის ან N_i ან N_i^2, ქვეამოცანის მიხედვით.

მაგალითი

განვიხილოთ შემდეგი ინტერაქცია, რომელშიც `solve` გამოიძახება ორჯერ:

მონაწილის პროგრამა	ჟიურის პროგრამა
	`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}`	

პირველი გამოძახების ახსნა

დამალული პერმუტაციაა $[2, 0, 1, 3]$, მაგრამ ლექსიკოგრაფიულად ყველაზე პატარა, რომელიც მისგან განურჩეველია, არის $[0, 2, 3, 1]$. ექვსი კითხვა დაისვა $N_1 = 4$-სათვის, რაც დაშვებულია, რადგან ამ ქვეამოცანაში $Q = N_1^2 = 16$.

მეორე გამოძახების ახსნა

ერთადერთი შესაძლო პერმუტაცია $N_2 = 1$-ისთვის არის $[0]$.

ქვეამოცანები

ქვეამოცანა	ქულები	N	T	Q	დამატებითი შეზღუდვები
0	0	-	-	N^2	მაგალითი.
1	7	$\leq 2^3$	2^{10}	N^2	-
2	18	$\leq 2^7$	2^5	N^2	-
3	5	$\leq 2^{11}$	2^5	N^2	-
4	5	$\leq 2^{11}$	2^5	N	-
5	5	$\leq 2^{18}$	2^5	N	$N = 2^k$ რომელიღაც მთელი რიცხვისთვის k.
6	5	$\leq 2^{18}$	2^5	N	N კენტია.
7	30	$\leq 2^{18}$	2^5	N	-
8	25	$\leq 2^{20}$	2^5	N	-

სანიმუშო გრეიდერი

შეტანის ფორმატი შემდეგია:

- ხაზი 1: ორი მთელი რიცხვი – T-სა და Q_{type}-ის მნიშვნელობები, სადაც Q_{type} არის ან 1, ან 2;
- შემდეგი $2T$ რაოდენობის ხაზი აღწერს ტესტებს:
  - ხაზი $2i$: ერთი მთელი რიცხვი – N-ის მნიშვნელობა i-ური ტესტისთვის;
  - ხაზი $2i+1$: N რაოდენობის მთელი რიცხვი – გადანაცვლება $p_0, p_1, \ldots, p_{N-1}$ i-ური ტესტისთვის.

თუ $Q_{type} = 1$, მაშინ $Q = N_i$ `solve`-ის i-ური გამოძახებისთვის; თუ $Q_{type} = 2$, მაშინ $Q = N_i^2$.

გამოტანის ფორმატი შემდეგია:

- ხაზი i: პერმუტაცია, რომელსაც აბრუნებს თქვენი პროგრამა i-ურ ტესტზე.