

XOR-lama

Vaxt limiti: 2 saniyə

Yaddaş limiti: 1024 MiB

Bu interaktiv məsələdir.

Tuncay Mahmud-a zarafat edib: o, 0 -dan $N - 1$ -ə qədər olan tam ədədlərin $p_0, p_1, \dots, p_{N-1}$ permutasiyasını gizlədib. Mahmud onu bərpa etmək üçün çox həvəslidir və siz ona kömək etməyə razılaşmısınız.

Tuncay permutasiyanı birbaşa açıqlamayacaq, lakin aşağıdakı növ suallara cavab verməyə razılıb: $0 \leq i, j < N$ şərtini ödəyən iki i və j indeksi seçə bilərsiniz və o sizə p_i ilə p_j -nin bitwise XOR (istisna VƏ YA) nəticəsini deyəcək.

İki ədədin bitwise XOR-u (istisna VƏ YA), $\oplus$ ilə işarə olunur, hər uyğun bit cütü üzərində məntiqi istisna VƏ YA əməliyyatını yerinə yetirir. Hər mövqedə nəticə 1 -dir, əgər bitlərdən yalnız biri 1 -dirsə, əks halda (hər ikisi 0 və ya hər ikisi 1 olduqda) 0 -dir. Məsələn, $75 \oplus 29 = 86$, çünki $1001011_{(2)} \oplus 0011101_{(2)} = 1010110_{(2)}$.

C++-da, a və b -nin XOR-unu $\wedge$ operatorundan istifadə edərək hesablaya bilərsiniz.

Tuncay **ən çox Q** suala cavab verməyə hazırdır.

Bununla belə, bu suallar həmişə gizli permutasiyanı unikal şəkildə müəyyən edə bilmir. $a_0, a_1, \dots, a_{N-1}$ permutasiyasını p -dən *fərqləndirilə bilməyən* adlandırırıq, əgər $0 \leq i, j < N$ şərtini ödəyən hər i və j indeks cütü üçün $a_i \oplus a_j = p_i \oplus p_j$ olarsa. Tuncay-nın verdiyi hər cavab, gizli permutasiya p və ya a olmasından asılı olmayaraq eynidir, ona görə də heç bir sual ardıcılığı ikisini bir-birindən ayıra bilməz. Qeyd edək ki, p həmişə özündən fərqləndirilə bilməzdir və bundan başqa belə bir permutasiya olmaya da bilər.

Sizin tapşırığınız p -dən fərqləndirilə bilməyən leksikoqrafik olaraq **ən kiçik** permutasiyanı tapmaqdır.

$x_0, x_1, \dots, x_{N-1}$ permutasiyası $y_0, y_1, \dots, y_{N-1}$ permutasiyasından leksikoqrafik olaraq kiçikdir, əgər onların fərqləndiyi ilk k mövqeyində $x_k < y_k$ olarsa. Məsələn, $x = [2, 0, 1, 5, 3]$ leksikoqrafik olaraq $y = [2, 0, 3, 1, 5]$ -dən kiçikdir, çünki onların fərqləndiyi ilk mövqe $k = 2$ -dir və $x_2 = 1 < 3 = y_2$.

Gizli permutasiya proqramınız başlamazdan əvvəl sabitlənir və suallarınıza cavab olaraq dəyişir.

Implementasiya təfərrüatları

Proqramınız `solve` funksiyasını implementasiya etməlidir:

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

- N : permutasiyanın elementlərinin sayı.

Bu funksiya münsiflər heyətinin proqramı tərəfindən tam olaraq T dəfə, hər gizli permutasiya üçün bir dəfə çağırılır. O, həmin çağırış üçün gizlədilmiş permutasiyadan fərqləndirilə bilməyən leksikoqrafik olaraq ən kiçik permutasiyanı qaytarmalıdır.

Münsiflər heyətinin proqramı ilə əlaqə saxlamaq üçün sizə aşağıdakı funksiya verilir:

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

`get_xor` funksiyası hər çağırıldığında, $p_i \oplus p_j$ qaytarır. Hər iki argument permutasiyanın etibarlı indeksləri olmalıdır. Bu şərt ödənilmədikdə, həlliniz `Output isn't correct: Invalid argument` nəticəsini alacaq. Funksiyanın sabit $O(1)$ vaxtında cavab verdiyini fərz edə bilərsiniz.

Q -nun qiyməti hər alt-tapşırıq üçün sabitdir və proqramınıza ötürülmür.

Məhdudiyyətlər

- $1 \leq N_i \leq 2^{20}$, `solve`-ə i -ci çağırış üçün, hər $0 \leq i < T$ üçün, burada N_i – i -ci çağırışda N -in qiymətidir;
- $1 \leq T \leq 2^{10}$;
- $T \cdot \max(N_0, N_1, \dots, N_{T-1}) \leq 2^{25}$.
- Testdə `solve`-in i -ci çağırışı zamanı ən çox Q sual verə bilərsiniz, burada Q ya N_i , ya da N_i^2 -dir, alt-tapşırıqdan asılı olaraq.

Nümunə

`solve` funksiyasının iki dəfə çağırıldığı aşağıdakı qarşılıqlı əlaqəni nəzərdən keçirək:

İştirakçı proqramı	Münsiflər proqramı
	<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>	

Birinci çağırışın izahı

Gizli permutasiya $[2, 0, 1, 3]$ -dür, lakin ondan fərqləndirilə bilməyən leksikoqrafik olaraq ən kiçiyi $[0, 2, 3, 1]$ -dir. $N_1 = 4$ üçün altı sual verildi ki, bu icazəlidir, çünki bu alt-tapşırıqda $Q = N_1^2 = 16$ -dır.

İkinci çağırışın izahı

$N_2 = 1$ olduqda mümkün olan yeganə permutasiya $[0]$ -dir.

Alt-tapşırıqlar

Alt-tapşırıq	Xal	N	T	Q	Əlavə məhdudiyyətlər
0	0	-	-	N^2	Nümunə.
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	$N = 2^k$, hər hansı tam k üçün.
6	5	$\leq 2^{18}$	2^5	N	N təkdir.
7	30	$\leq 2^{18}$	2^5	N	-
8	25	$\leq 2^{20}$	2^5	N	-

Nümunə qiymətləndirici

Giriş formatı aşağıdakı kimidir:

- sətir 1: iki tam ədəd – T və Q_{type} -in qiymətləri, burada Q_{type} ya 1, ya da 2-dir;
- sonrakı $2T$ sətir testləri təsvir edir:
 - sətir $2i$: bir tam ədəd – i -ci test üçün N -in qiyməti;
 - sətir $2i + 1$: N tam ədəd – i -ci test üçün $p_0, p_1, \dots, p_{N-1}$ permutasiyası.

Əgər $Q_{type} = 1$ -dirsə, `solve`-in i -ci çağırışı üçün $Q = N_i$ -dir; əgər $Q_{type} = 2$ -dirsə, $Q = N_i^2$ -dir.

Çıxış formatı aşağıdakı kimidir:

- sətir i : programınızın i -ci testdə qaytardığı permutasiya.