

XORting

Zaman sınırı: 2 saniye

Bellek sınırı: 1024 MiB

Bu etkileşimli bir problemdir.

Iliyan, Jonas'a bir şaka yapmıştır: 0 ile $N - 1$ arasındaki tam sayıların bir permütasyonu olan $p_0, p_1, \dots, p_{N-1}$ permütasyonunu gizlemiştir. Jonas bu permütasyonu bulmak istemektedir ve siz de ona yardım etmeyi kabul ettiniz.

Iliyan permütasyonu doğrudan açıklamayacaktır, ancak şu tür soruları yanıtlamayı kabul etmiştir: $0 \leq i, j < N$ olacak şekilde iki indeks i ve j seçebilirsiniz ve Iliyan size p_i ile p_j değerlerinin bit düzeyinde XOR (exclusive OR) sonucunu söyler.

İki sayının $\oplus$ ile gösterilen bit düzeyinde XOR (exclusive OR) işlemi, karşılık gelen her bit çifti üzerinde mantıksal exclusive OR işlemini uygular. Her bir konumdaki sonuç, bitlerden tam olarak biri 1 ise 1; her ikisi de 0 veya her ikisi de 1 ise 0 olur. Örneğin, $75 \oplus 29 = 86$ 'dır, çünkü $1001011_{(2)} \oplus 0011101_{(2)} = 1010110_{(2)}$.

C++'ta `a` ve `b` değerlerinin XOR sonucunu `^` operatörünü kullanarak hesaplayabilirsiniz.

Iliyan **en fazla** Q soruyu yanıtlamaya razıdır.

Ancak bu sorular, gizli permütasyonu her zaman tek bir şekilde belirlemeye yetmeyebilir. Her $0 \leq i, j < N$ indeks çifti için $a_i \oplus a_j = p_i \oplus p_j$ eşitliği sağlanıyorsa, $a_0, a_1, \dots, a_{N-1}$ permütasyonuna p 'den *ayırt edilemez* deriz. Iliyan'ın vereceği her cevap, gizli permütasyon ister p ister a olsun aynı olacağından, hiçbir soru dizisi bu iki permütasyonu birbirinden ayırt edemez. p 'nin her zaman kendisinden ayırt edilemez olduğuna ve başka hiçbir böyle permütasyon bulunmayabileceğine dikkat edin.

Göreviniz, p 'den ayırt edilemeyen sözlük sırasına göre en küçük permütasyonu bulmaktır.

Bir $x_0, x_1, \dots, x_{N-1}$ permütasyonu, $y_0, y_1, \dots, y_{N-1}$ permütasyonundan, farklı oldukları ilk k konumunda $x_k < y_k$ ise sözlük sırasına göre daha küçüktür. Örneğin, $x = [2, 0, 1, 4, 3]$, $y = [2, 0, 3, 1, 4]$ 'ten sözlük sırasına göre daha küçüktür; çünkü farklı oldukları ilk konum $k = 2$ 'dir ve $x_2 = 1 < 3 = y_2$ 'dir.

Gizli permütasyon, programınız başlamadan önce sabitlenir ve sorularınıza bağlı olarak değişmez.

Uygulama detayları

Programınız `solve` fonksiyonunu gerçekleştirmelidir:

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

- N : permütasyondaki eleman sayısı.

Bu fonksiyon, her gizli permütasyon için bir kez olmak üzere tam olarak T kez çağrılır. Her çağrıda, o çağrı için gizlenmiş olan permütasyondan ayırt edilemeyen sözlük sırasına göre en küçük permütasyonu döndürmelidir.

Jüri programıyla iletişim kurmanız için aşağıdaki fonksiyon sağlanmıştır:

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

`get_xor` fonksiyonu her çağrıldığında $p_i \oplus p_j$ değerini döndürür. Her iki argüman da permütasyonun geçerli indeksleri olmalıdır. Bu koşul sağlanmazsa çözümünüz `Output isn't correct: Invalid argument` sonucunu alacaktır. Fonksiyonun sabit zamanda $O(1)$ yanıt verdiğini varsayabilirsiniz.

Q değeri her alt görev için sabittir ve programınıza parametre olarak verilmez.

Kısıtlar

- Her $1 \leq i \leq T$ için $1 \leq N_i \leq 2^{20}$, burada N_i , `solve` fonksiyonunun i 'inci çağrısındaki N değeridir.
- $1 \leq T \leq 2^{10}$
- $T \cdot \max(N_1, N_2, \dots, N_T) \leq 2^{25}$.
- Bir testte `solve` fonksiyonunun i 'inci çağrısı sırasında en fazla Q_i soru sorabilirsiniz; burada Q_i , alt göreve bağlı olarak N_i veya N_i^2 değerlerinden biridir.

Örnek

`solve` fonksiyonunun iki kez çağrıldığı aşağıdaki etkileşimi ele alalım:

Katılımcı programı	Jüri programı
	<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>	

İlk çağrının açıklaması

Gizli permütasyon $[2, 0, 1, 3]$ 'tür, ancak ondan ayırt edilemeyen sözlük sırasına göre en küçük permütasyon $[0, 2, 3, 1]$ 'dir. $N_1 = 4$ için altı soru sorulmuştur; bu alt görevde $Q_1 = N_1^2 = 16$ olduğundan buna izin verilir.

İkinci çağrının açıklaması

$N_2 = 1$ için mümkün olan tek permütasyon $[0]$ 'dır.

Alt görevler

Alt görev	Puan	N_i	T	Q_i	Ek kısıtlar
0	0	-	-	N_i^2	Örnek.
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	$N_i = 2^k$, burada k bir tam sayıdır.
6	5	$\leq 2^{18}$	2^5	N_i	N_i tektir.
7	30	$\leq 2^{18}$	2^5	N_i	-
8	25	$\leq 2^{20}$	2^5	N_i	-

Örnek değerlendirici

Girdi formatı aşağıdaki gibidir:

- satır 1: iki tam sayı – T ve Q_{type} değerleri; burada Q_{type} ya 1 ya da 2'dir;
- sonraki $2T$ satır testleri açıklar:
 - satır $2i$: bir tam sayı – i 'inci test için N_i değeri;
 - satır $2i + 1$: N_i adet tam sayı – i 'inci test için $p_0, p_1, \dots, p_{N_i-1}$ permütasyonu.

Eğer $Q_{type} = 1$ ise, `solve` fonksiyonunun i 'inci çağrısı için $Q_i = N_i$; eğer $Q_{type} = 2$ ise $Q_i = N_i^2$ olur.

Çıktı formatı aşağıdaki gibidir:

- satır i : programınızın i 'inci testte döndürdüğü permütasyon.