Alice, Bob, ve Circuit (Devre)

Cyberland Devre Vakfı n üyeden oluşur. Her üyenin favori sayısı ve benzersiz bir adı vardır (favori sayılar farklı olmayabilir).

Üyeler arasında m tane mektup gönderilmiştir. Her mektubun bir göndericisi ve bir alıcısı vardır ve mektubun içeriği gönderenin favori sayısıdır.

Her üye, aldığı içeriğin (gönderenlerin favori sayısı) toplamını hesaplar ve bu toplamın 65536 (yani 2^{16}) 'a göre modunu alarak kendi sonuç sayısını hesaplar.

Göreviniz tüm sonuç sayılarını bulmaktır.

Ancak durum göründüğü kadar basit değildir. Alice, Bob ve Circuit (Devre) bu sorunu biraz daha karmaşık bir şekilde çözmeye karar verir:

- Alice, tüm n üyeyi (isim ve favori sayı) bilir, ancak mektuplar hakkında hiçbir şey bilmez. Onun devreye uzunluğu 10^5'den fazla olmayan bir ikili dizgi (String) göndermesi gerekir.
- Bob tüm m mektubu (gönderen ve alıcının adı) bilir, ancak üyeler hakkında hiçbir şey bilmez. Onun devreye uzunluğu 10^5'den fazla olmayan bir ikili dizgi (String) göndermesi gerekir.
- Devre, Alice ve Bob tarafından gönderilen ikili dizgileri (String) alabilir ve ardından çıktı olarak $16n$ bit içeren bir ikili dizgi oluşturabilir. Bununla birlikte, sınırlı hesaplama gücü nedeniyle devre yalnızca temel mantıksal işlemleri (ör. AND, OR, NOT) gerçekleştirme yeteneğine sahiptir.

Aşağıda devrenin nasıl çalıştığı ayrıntılı olarak açıklanmıştır.

Devre Detayları

Kapı, bir devrenin temel elemanıdır. Bir kapı, sıfır veya iki boolean girişi ve bir boolean çıkışından oluşur. İki tür kapı vardır: girdi kapıları ve hesaplama kapıları.

- Girdi kapılarının girişi yoktur ve Alice ve Bob tarafından gönderilen ikili dizgilerden gelen bitleri temsil eder.
  - 0 ile $(l_A + l_B - 1)$ arasında etiketlenmiş $l_A + l_B$ girdi kapısı olacaktır, burada l_A, l_B sırasıyla Alice ve Bob'dan gelen dizgilerin uzunluklarıdır.
  - $0 \le i < l_A$, için i-inci kapının çıktısı, Alice'ten gelen dizginin i-inci bitidir;
  - $0 \le i < l_B$ için, $(i + l_A)$-inci kapının çıktısı, Bob'dan gelen dizginin i-inci bitidir.
- Hesaplama kapılarının iki girdisi vardır ve hesaplama sürecini temsil eder.
  - Hesaplama kapıları için etiketler $(l_A + l_B)$'dan başlar.
  - Her hesaplama kapısı için, o kapıya bağımlı iki kapının etiketini ve $p (0 \le p \le 15)$ işlem türünü vermelisiniz.
    - Döngüsel bağımlılıkları önlemek için, iki bağımlı kapının etiketi hesaplama kapısının etiketinden daha küçük olmalıdır.
    - İki bağımlı kapının çıktıları sırasıyla x_0 ve x_1 ise $(x_0, x_1 \in \{0, 1\})$, o zaman hesaplama kapısının çıktısı:

$$f(p, x_0, x_1) = \left\lfloor \frac{p}{2^{x_0 + 2x_1}} \right\rfloor \bmod 2$$

Aşağıda sizin için kullanışlı olabilecek bazı örnekler vardır:

x_0	x_1	x_0 AND x_1 $f(8, x_0, x_1)$	x_0 OR x_1 $f(14, x_0, x_1)$	x_0 XOR x_1 $f(6, x_0, x_1)$	NOT x_0 $f(5, x_0, x_1)$
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

Kodlama Detayları

Not:

- Tüm dizi indisleri 0'dan başlar. Örneğin, `a`, `n` uzunluğunda bir diziyse, `a[0]` ila `a[n-1]` geçerli verilerdir, bu aralığın ötesindeki indislere erişmek out-of-bounds hatasına neden olabilir .
- Tüm dizgiler null karakter `\0` ile sonlanır.

Aşağıdaki prosedüleri kodlamalısınız.

Alice

```
int alice(const int n, const char names[][5], const unsigned short numbers[], bool outputs_alice[]);
```

Yön	Value (Değer)	Uzunluk	Anlam	Kısıt
Girdi	`n`	1	n	$0 \leq n \leq 700$
	`names`	n	Her üyenin adı.	Her isim benzersizdir, sadece küçük harf ile İngilizce karakterleri içerir, ve maksimum 4 karakter uzunluğundadır.
	`numbers`	n	Her üyenin favori sayısı.	Her sayı 0 ile 65535 arasındadır.
Çıktı	`outputs_alice`	l_A	İkili dizgi devreye gönderilir.	
	(Return değer)	1	l_A	**l_A'ın 10^5'ı geçmediğinden emin olmalısınız ve n aynı olduğunda l_A sabitlenmelidir.**

Bob

```
int bob(const int m, const char senders[][5], const char recipients[][5], bool outputs_bob[]);
```

Yön	Value (Değer)	Uzunluk	Anlam	Kısıt
Girdi	`m`	1	m	$0 \leq m \leq 1000$
	`senders`	m	Her mektupta göndericinin adı vardır.	Tüm isimler Alice'in girdisinde görülür.
	`recipients`	m	Her mektupta alıcının adı vardır.	
Çıktı	`outputs_bob`	l_B	İkili dizgi devreye gönderilir.	
	(Return değer)	1	l_B	**l_B'ın 10^5'ı geçmediğinden emin olmalısınız ve m aynı olduğunda l_B'ın sabitlenmesi gerekir.**

Circuit (Devre)

Devrenin hesaplama işleminin genel bir devre gibi olmasını sağlamak için Alice ve Bob'dan Circuit'e gönderilen ikili dizgileri doğrudan elde edemezsiniz. Siz sadece bu iki dizginin uzunluklarını biliyorsunuz ve devre yapısını çıkartıyorsunuz.

```
int circuit(const int la, const int lb, int operations[], int operands[][2], int outputs_circuit[][16]);
```

Yön	Value (Değer)	Uzunluk	Anlam	Kısıt
Girdi	`la`	1	l_A	
	`lb`	1	l_B	
Çıktı	`operations`	l	Devredeki her kapı tarafından gerçekleştirilen işlemin türü.	0 ile 15 arasında tam sayı.

Yön	Value (Değer)	Uzunluk	Anlam	Kısıt
	`operands`	l	Devredeki her kapı tarafından kullanılan işleç (operand).	Sayı, geçerli kapının etiketinden küçük olmalıdır.
	`outputs_circuit`	n	Devre çıktısının kapı etiketi.	`outputs_circuit[i][j]`, i-inci üye için nihai sonucun j-inci bitini (saymaya en önemsiz bitten başlayarak) belirtir. Üyeler, Alice'in girdisine göre sıralanır.
	(Return değer)	1	l, toplam kapı sayısını temsil eder (girdi kapıları dahil).	$l \leq 2 \times 10^7$ **olduğundan emin olmanız gerekir**

`operations` ve `operands` dizilerinde $l_A + l_B$'den küçük indislere sahip kapıların bilgilerini değiştirebilseniz de, değerlendirici (grader) bu tür değişiklikleri göz ardı eder.

Örnek

Aşağıdaki çağrıları göz önüne alın:

```
alice(3, {"alic", "bob", "circ"}, {10000, 20000, 30000}, outputs_alice);
bob(5, {"alic", "bob", "bob", "circ", "circ"}, {"circ", "circ", "alic", "circ", "circ"}, outputs_bob);
```

Bu aşağıdaki senaryoyu temsil eder:

- Alice 3 üye olduğunu biliyor, `alic` adlı üyenin favori sayısı 10000, vb. `alice()` için olası bir çıktı şudur:
  - `alice()` return değeri 2'dir ve bu $l_A = 2$'ı temsil eder.
  - `alice()` fonksiyonu içinde, `outputs_alice[0] = 1, outputs_alice[1] = 0` olarak ayarlayın. Bu sonuçtaki ikili dizginin 10 olduğunu gösterir.
- Bob 5 mektup olduğunu biliyor, ilk mektup `alic` 'ten `circ` 'e dir, vb. `bob()` için olası bir çıktı şudur:
  - `bob()` return değeri 3'dür ve $l_B = 3$'ü temsil eder.
  - `bob()` fonksiyonu içinde, `outputs_bob[0] = 1, outputs_bob[1] = 1, outputs_bob[2] = 0` olarak ayarlayın. Bu sonuçtaki ikili dizginin 110 olduğunu gösterir.

`alice()` ve `bob()` 'un önceki çıktılarına dayalı olarak aşağıdaki çağrı olacaktır:

```
circuit(2, 3, operations, operands, outputs_circuit);
```

Bu fonksiyon için doğru bir çıktı şöyle olabilir:

- `circuit()` nin return değeri 7'dir, yani 5 ve 6 olarak etiketlenmiş iki hesaplama kapısı ekliyoruz.
- `circuit()` içinde, `operations`, `operands`, and `outputs_circuit` şu şekilde ayarlayın:
  - `operations = {-1, -1, -1, -1, -1, 8, 14}`, burada giriş kapılarından göz ardı edilen bilgileri temsil etmek için `-1` kullanırız;
  - `operands = {{-1, -1}, {-1, -1}, {-1, -1}, {-1, -1}, {-1, -1}, {0, 4}, {2, 5}};`
  - `outputs_circuit = {{5, 5, 5, 5, 5, 6, 5, 5, 5, 6, 6, 6, 5, 5, 6, 5}, ...}`. Dizi biraz uzun, tam dizi için eklerdeki `abc.cpp`yi kontrol edebilirsiniz.

Çıktıya göre, hesaplama prosedürü şu şekildedir:

- 0 kapısından ve 4 kapısına girdi alan tür 8 hesaplama kapısı ekleyin. 0 kapısının çıktısı, Alice'ten gelen dizginin 1 olan 0-inci bitidir; 4 kapısının çıktısı, Bob'dan gelen dizginin 2-inci bitidir, bu 0'dır. Yani 5 kapısının çıktısı $f(8, 0, 1) = 0 \text{ AND } 1 = 0$'dır.
- 2 kapısından ve 5 kapısına girdi alan tür 14 hesaplama kapısı ekleyin. 2 kapısının çıktısı Bob'dan gelen dizginin 0-ınci bitidir, bu 1'dır; 5 kapısının çıktısı 0'dır. Yani 6 kapısının çıktısı $f(14, 1, 0) = 1 \text{ OR } 0 = 1$.
- `output_circuit[0]`, `alic`in nihai sonucunu temsil eder, bu $(0100111000100000)_2 = 20000$'dır. `alic`, `bob`dan yalnızca bir mektup aldığından, `alic` in nihai sonucu 20000'dır.
- Hiç mektup almadığına göre `bob` un nihai sonucu 0 olmalıdır; `circ` in nihai sonucu $(10000 + 20000 + 30000 + 30000) \bmod 65536 = 24464$ olmalıdır.

Eklerdeki `abc.cpp` bu örneği geçebilir, ancak diğer test senaryolarını geçebileceğini garanti etmiyoruz.

Kısıtlar

Tüm test durumları için:

- $0 \le n \le 700, 0 \le m \le 1000$.
- Tüm adlar farklıdır, yalnızca küçük İngilizce harflerden oluşur ve maksimum 4 karakter uzunluğundadır.
- Her üyenin favori sayısı 0 ile 65535 aralığındadır.
- Tüm gönderenlerin ve alıcıların adları, Alice'in `names` girdi dizisinde görünür.
- `alice()` ve `bob()` , sırasıyla 2048 MB'lık bir bellek sınırına ve $0,02$ saniyelik bir zaman sınırına sahiptir.
- `circuit()` 2048 MB bellek sınırıa ve 7 saniye zaman sınırına sahiptir.

Son değerlendirme için, tek bir test senaryosunda `alice()` ve `bob()` birden çok kez çağrılabilir. Her çağrı için $0,02$ saniyelik zaman sınırı vardır.

Altgörevler

Altgörev Tür A (12 puan)

Alt görev 1,2,3, $n = 1$ olduğu A alt görev türündedir.

Her alt görev aşağıdaki ek kısıtlamalara sahiptir:

- Altgörev 1 (4 puan): $m = 0$.
- Altgörev 2 (4 puan): $0 \le m \le 1$.
- Altgörev 3 (4 puan): $0 \le m \le 1000$.

Altgörev Tür B (54 puan)

Alt görev 4,5,6 B alt görev türündedir. Burada,

- $0 \le n \le 30, \frac{n}{2} \le m \le n^2$.
- Aynı gönderici ve alıcıya sahip iki mektup yoktur.
- Bob'un girdisinde tüm üye adları görünür (yani, her üye ya en az bir mektup gönderir ya da en az bir mektup alır).

Her alt görev aşağıdaki ek kısıtlamalara sahiptir:

- Altgörev 4 (24 puan): $n = 26$, Tüm üyelerin adları tek karakterli küçük harflerdir ve Alice'in girdisinde `a` dan `z` ye kadar sıralanır.
- Altgörev 5 (24 puan): $n = 26$.
- Altgörev 6 (6 puan): Özel kısıt yoktur.

Altgörev Tür C (34 puan)

Alt görev 7,8,9 C alt görev türündedir. Burada, $0 \le n \le 700, 0 \le m \le 1000$.

Her alt görev aşağıdaki ek kısıtlamalara sahiptir:

- Altgörev 7 (18 puan): $n = 676$, tüm üyelerin adları iki küçük harften oluşur ve Alice'in girdisinde sözlük sırasına göre görünürler (yani, `aa`, `ab`, `ac`, ..., `az`, `ba`, ..., `bz`, `ca`, ..., `zz`).
- Altgörev 8 (10 puan): $n = 676$.
- Altgörev 9 (6 puan): Ek sınır yoktur.

Örnek Değerlendirici

Örnek değerlendirici, girdiyi aşağıdaki formatta okur:

- Satır 1: $n\ m$
- Satır $2 + i (0 \le i \le n - 1)$: $names_i\ numbers_i$
- Satır $2 + n + i (0 \le i \le m - 1)$: $senders_i\ recipients_i$.

Örnek değerlendirici aşağıdaki formatta çıktı verir:

- Program başarılı bir şekilde tamamlanırsa, örnek değerlendirici, her üye için kodladığınız fonksiyonlar tarafından hesaplanan nihai sonucu temsil eden, her biri bir tamsayı içeren n satır çıktı verecektir.
- Aksi takdirde, değerlendirici stdout'a hiçbir çıktı vermez ve hata mesajlarını dizindeki (directory) `abc.log` dosyasına yazacaktır.
- Ek olarak, örnek değerlendirici l_A, l_B, l değerlerini ve her bir fonksiyonun çalışma süresini `abc.log` içine yazar.

Örnek değerlendirici, aynı n / m, için l_A / l_B 'nin eşit olması gerektiğini ve bellek sınırını kontrol etmeyecektir.