Mesaj

Aişe ve Basma birbirleriyle yazışan iki arkadaştır. Aişe'nin Basma'ya göndermek bir M mesajı vardır, bu da S bit serisidir (yani bu seri sıfırlar veya birlerden oluşur). Aişe, Basma'ya **paket** göndererek onunla iletişim kuruyor. Bir paket 0 ile 30 arasında indekslenen 31 bitlik bir seridir. Aişe, Basma'ya bir miktar paket göndererek M mesajını göndermek istiyor.

Ne yazık ki Kleopatra, Aişe ile Basma arasındaki iletişimi bozar ve paketleri **kirletebilme** yeteneğine sahiptir. Yani Cleopatra her pakette tam olarak 15 tane indeksteki biti değiştirebilir. Spesifik olarak, anlamı aşağıdaki şekilde verilen her elemanın 0 ya da 1 olduğu 31 uzunluğunda bir C dizisi vardır:

- $C[i] = 1$, i indeksli bitin Kleopatra tarafından değiştirilebileceğini gösterir. Biz bunlara Kleopatra'nın **kontrol ettiği** indeksler diyoruz.
- $C[i] = 0$, i indeksli bitin Kleopatra tarafından değiştirilemeyeceğini gösterir.

C dizisi tam olarak 15 tane bir ve 16 tane sıfırdan oluşmaktadır. M mesajı gönderilirken, Cleopatra tarafından kontrol edilen indeks kümesi tüm paketler için aynı kalır. Aişe, Kleopatra'nın hangi 15 indeksi kontrol ettiğini tam olarak biliyor. Basma ise sadece 15 indeksin Kleopatra tarafından kontrol edildiğini biliyor, ama hangi indeksler olduğunu bilmiyor.

A nın Aişe'nin göndermeye karar verdiği bir paket olduğunu varsayalım (ki buna **orijinal paket** diyoruz). B nin Basma tarafından alınan paket olduğunu varsayalım (ki biz buna **kirli paket** diyoruz). Her i için, $0 \leq i < 31$ olacak şekilde:

- eğer Kleopatra i indeksli biti kontrol etmiyorsa ($C[i] = 0$), Basma, Aişe tarafından gönderilen bit i yi alır ($B[i] = A[i]$),
- Aksi takdirde, eğer Kleopatra i indeksli biti kontrol ediyorsa ($C[i] = 1$), $B[i]$ nin değeri Kleopatra tarafından belirlenir.

Her paketi gönderdikten hemen sonra, Aişe, kirli paketin ne olduğunu öğrenir.

Aişe bütün paketleri gönderdikten sonra, Basma, tüm kirli paketleri **gönderildikleri sırayla** alır ve orijinal mesaj M'yi yeniden inşa etmek zorundadır.

Göreviniz bir strateji tasarlamak ve uygulamaktır: Bu strateji, Aişe'nin Basma'ya M mesajını göndermesine izin verecektir ve böylece Basma, kirlenmiş paketlerden M'yi kurtarabilecektir. Özellikle iki prosedürü kodlamanız gerekir. Birinci prosedür Aişe'nin yaptıklarını gerçekleştirecektir. M mesajı ve C dizisi verilir, ve mesajı Basma'ya iletmek için bazı paketler göndermesi gerekir. İkinci

prosedür Basma'nın yaptıklarını gerçekleştirecektir. Kirli paketler verilir ve orijinal M mesajını kurtarmalıdır.

Kodlama Detayları

Kodlamanız gereken ilk prosedür şudur:

```
void send_message(std::vector<bool> M, std::vector<bool> C)
```

- M : Aişe'nin Basma'ya iletmek istediği mesajı belirten S uzunluğunda bir dizi.
- C : Kleopatra'nın kontrolündeki bitlerin indekslerini gösteren 31 uzunluğunda bir dizi.
- Bu prosedür her test durumunda **en fazla 2100 kez** çağrılabilir.

Bu prosedür bir paket göndermek için aşağıdaki prosedürü çağırmalıdır:

```
std::vector<bool> send_packet(std::vector<bool> A)
```

- A : Aişe'nin gönderdiği bitleri temsil eden orijinal bir paket (uzunluğu 31 olan bir dizi)
- Bu prosedür, Basma'nın alacağı bitleri temsil eden kirlenmiş bir paket B döner.
- Bu prosedür her `send_message` her çağrısında en fazla 100 kez çağrılabilir.

Kodlamanız gereken ikinci prosedür şudur:

```
std::vector<bool> receive_message(std::vector<std::vector<bool>> R)
```

- R : kirlenmiş paketleri tanımlayan dizi. Paketler, Aişe'nin tek bir `send_message` çağrısında gönderdiği paketlerden kaynaklanmaktadır ve Aişe'nin gönderdiği **sırayla** verilmiştir. R nin her bir elemanı, kirlenmiş bir paketi temsil eden, uzunluğu 31 olan bir dizidir.
- Bu prosedür orijinal mesaj M ye eşit olan S bitlik bir dizi dönmelidir.
- Bu prosedür her test durumunda **birden çok kez**, ve karşılık gelen her bir `send_message` çağrısı için **tam olarak bir kez** çağrılabilir. `receive_message` **prosedür çağrılarının sıralaması** ona denk gelen `send_message` çağrılarının sırasının aynı olması gerekmez.

Notlandırma sisteminde `send_message` ve `receive_message` prosedürlerinin **ayrı programlarda** çağrıldığını unutmayın.

Kısıtlar

- $1 \leq S \leq 1024$
- C nin tam olarak 31 elemanı vardır ve bunların 16'sı 0'a ve 15'i 1'e eşittir.

Altgörevler ve Puanlama

Eğer test durumlarından herhangi birinde, `send_packet` prosedürüne yapılan çağrılar yukarıda belirtilen kurallara uymuyorsa, veya `receive_message` prosedürüne yapılan çağrılardan herhangi birinin dönme değeri yanlışsa, bu test durumu için çözümünüz 0 puan alacaktır.

Aksi takdirde, Q nuntüm test durumları üzerindeki `send_message` çağrıları arasında `send_packet` prosedürüne yapılan çağrıların maksimum sayısı olduğunu varsayalım. Ayrıca X in şuna eşit olduğunu varsayalım:

- 1 , eğer $Q \leq 66$
- 0.95^{Q-66} , eğer $66 < Q \leq 100$
- 0 , eğer $100 < Q$ ise

Puan şu şekilde hesaplanır:

Altgörev	Puan	Ek Kısıtlar
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Ek kısıt yoktur.

Bazı durumlarda graderin davranışının **uyarlanabilir (adaptif)** olduğunu unutmayın. Bu, `send_packet` tarafından dönen değerlerin girdi argümanları ve bu prosedüre yapılan önceki çağrıların dönme değerine bağlı olabileceği anlamına gelir.

Örnek

Aşağıdaki çağrıyı göz önüne alın.

```
send_message([0, 1, 1, 0],
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

Aişe'nin Basma'ya iletmeye çalıştığı mesaj $[0, 1, 1, 0]$ dır. 0 ile 15 arasındaki indekslere sahip bitler Kleopatra tarafından değiştirilemez, 16 ile 30 arasındaki indeksli bitler ise Kleopatra tarafından değiştirilebilir.

Bu örnek için, Kleopatra'nın davranışının deterministik olduğunu varsayalım, ve kontrol ettiği ardışık bitleri 0 ve 1 arasında dönüşümlü olarak doldurur, yani şunları atar 0'ı kontrol ettiği ilk indekse atar (bizim örneğimizde indeks 16), 1'i kontrol ettiği ikinci indekse atar (indeks 17), 0'ı kontrol ettiği üçüncü indekse atar (indeks 18), ve benzeri.

Aişe, orijinal mesajdan iki biti tek bir pakette göndermeye aşağıdaki şekilde karar verebilir: ilk biti kontrol ettiği ilk 8 indekse gönderecek ve ikinci biti kontrol ettiği ve takip eden 8 indekse gönderecektir.

Aişe daha sonra şu paketi göndermeyi seçer:

```
send_packet([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Kleopatra'nın son 15 indeksteki bitleri değiştirebileceğini unutmayın. Böylece Aişe bunları keyfi olarak ayarlayabilir, çünkü bunlar üzerine yazılabilir (değişebilir). Kleopatra'nın varsayılan stratejisiyle prosedür şunu döner: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Aişe daha öncekine benzer şekilde M 'nin son iki bitini ikinci pakette göndermeye karar verir:

```
send_packet([1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Kleopatra'nın varsayılan stratejisiyle prosedür şunu döner: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Aişe daha fazla paket gönderebilir ama o göndermemeyi tercih ediyor.

Daha sonra değerlendirici aşağıdaki prosedür çağrısını yapar:

```
receive_message([[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                  0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
                 [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                  0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]])
```

Basma, M mesajını şu şekilde kurtarır. Her paketten, üst üste iki kez geçen ilk biti, ve üst üste iki kez geçen son biti alır. Yani ilk paketten $[0,1]$ bitlerini ve ikinci paketten $[1,0]$ bitlerini alıyor. Bunları bir araya getirerek $[0,1,1,0]$ mesajını kurtarır, bu ise `receive_message` çağrısı için doğru dönme değeridir.

Kleopatra'nın varsayılan stratejisi ve 4 uzunluğundaki mesajlar için gösterilebilir ki, Basma'nın bu yaklaşımı, C değerinden bağımsız olarak M yi doğru bir şekilde kurtarır. Ancak genel durumda bu doğru değildir.

Örnek Değerlendirici

Örnek değerlendirici uyarlanabilir değildir. Bunun yerine Kleopatra'nın davranışı deterministiktir. ve kontrol ettiği ardışık bitleri, yukarıdaki örnekte anlatıldığı gibi, dönüşümlü olarak 0 ve 1 bitleriyle doldurur.

Girdi formatı: **Girdinin ilk satırı senaryo sayısını belirten bir tamsayı T içerir.** Bunu T senaryo takip eder. Her biri aşağıdaki formatta sunulmaktadır:

```
S
M[0]  M[1]  ...  M[S-1]
C[0]  C[1]  ...  C[30]
```

Çıktı formatı: Örnek değerlendirici, T senaryosunun her birinin sonucunu girdide sağlandığı sırayla aşağıdaki formatta yazar:

```
K L
D[0]  D[1]  ...  D[L-1]
```

Burada, K `send_packet` için yapılan çağrıların sayısıdır, D `receive_message` tarafından dönen mesajdır ve L bunun uzunluğunu gösterir.