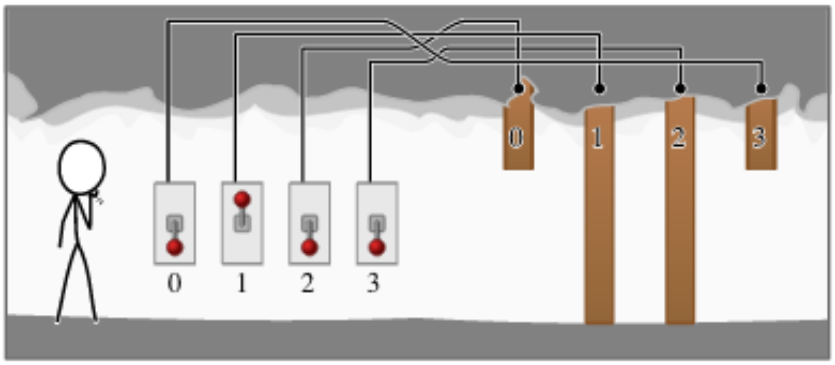
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Turkish — 1.0

Kaldığınız yurtlardan sınavın yapılacağı yere yürürken kayboldunuz ve yanlışlıkla üniversitenin altında yer alan gizli bir mağaranın girişine geldiniz. Mağaranın girişi, birbiri arkasına ardışık yerleştirilmiş `N` tane kapı; ve herbiri ayrı bir kapıya bağlı `N` tane anahtar içeren bir güvenlik sistemi ile korunmaktadır.



Kapılar, `0, 1, …, (N - 1)` olarak numaralanmıştır ve size en yakın kapı 0 numaralı kapıdır. Anahtarlar da `0, 1, …, (N - 1)` olarak numaralandırılmıştır, fakat siz hangi anahtarın hangi kapıya bağlı olduğunu bilmiyorsunuz.

Anahtarlar mağaranın girişinde bulunmaktadır. Her bir anahtar ya *yukarı* ya da *aşağı* vaziyette olabilmektedir. Bu vaziyetlerden sadece birisi doğrudur. Eğer anahtar doğru vaziyette ise anahtara bağlı olan kapı açılacak; yanlış vaziyette ise anahtara bağlı olan kapı kapanacaktır. Her bir anahtar için doğru vaziyet farklı olabilir, ve siz hangi vaziyetin doğru olduğunu bilmiyorsunuz.

Bu güvenlik sistemini öğrenmeye merak sardınız. Bunun için, anahtarları istediğiniz kombinasyona getirerek mağara içine doğru yürüyüp ilk kapalı kapının hangisi olduğunu görebiliyorsunuz. Kapılar şeffaf olmadığı için ilk kapalı kapıyı gördüğünüzde bu kapının arkasındaki kapıları göremiyorsunuz.

Sizin göreviniz ise, en fazla `70,000` kombinasyon deneyerek, her bir anahtarın doğru vaziyetini ve her bir anahtarın bağlı olduğu kapıyı bulmaktır.

Gerçekleştirim

`exploreCave()` prosedürünü gerçekleştiren bir dosya göndermelisiniz. Bu prosedür `tryCombination()` fonksiyonunu en fazla 70,000 kez çağırabilir ve en sonunda `answer()` prosedürünü çağırmalıdır. Fonksiyonlar ve prosedürler aşağıda tanımlanmıştır.

Grader Fonksiyonu: `tryCombination()`

C/C++
```
int tryCombination(int S[]);
```

Pascal
```
function tryCombination(var S: array of LongInt) : LongInt;
```

Tanım

Grader bu fonksiyonu size sağlayacaktır. Bu fonksiyon anahtarların herhangi bir vaziyet kombinasyonunu denediğinizde karşılık gelen ilk kapalı kapıyı döndürecektir. Eğer tüm kapılar açık ise, `-1` döndürecektir. Bu fonksiyon `O(N)` zamanda çalışır.

Bu fonksiyonu en fazla `70,000` kez çağırabilirsiniz.

Parametreler

- `S`: `N` uzunluğunda bir dizi, dizinin elemanları karşılık gelen anahtarın vaziyetidir. Yani, `S[i]` değeri `i`. nci anahtarın vaziyetidir. `0` değeri anahtarın yukarı, ve `1` değeri anahtarın aşağı olduğunu belirtmek içindir.
- *Dönen değer*: İlk kapalı kapının numarası, ya da eğer tüm kapılar açıksa `-1`.

Grader Prosedürü: `answer()`

C/C++
```
void answer(int S[], int D[]);
```

Pascal
```
procedure answer(var S, D: array of LongInt);
```

Tanım

Bu prosedürü tüm kapıları açacak anahtar vaziyeti kombinasyonu, ve her bir anahtarın bağlı olduğu kapıyı bulduğunuzda çağırmalısınız.

Parametreler

- `S`: `N` uzunluğunda bir dizi, dizinin elemanları anahtarların doğru vaziyetleri olmalıdır. Formatı `tryCombination()` fonskiyonununda anlatılan format ile aynıdır.
- `D`: `N` uzunluğunda bir dizi, anahtarların bağlı olduğu kapıları belirtir. Yani, `D[i]` değeri `i`.nci anahtarın bağlı olduğu kapıdır.
- *Dönen değer*: Bu prosedür programı sonlandırdığından bir şey döndürmez.

Sizin Prosedürünüz: `exploreCave()`

C/C++
```
void exploreCave(int N);
```

Pascal
```
procedure exploreCave(N: longint);
```

Tanım

Gönderdiğiniz dosya bu fonksiyonu gerçekleştirmelidir.

Bu fonksiyon, `tryCombination()` fonksiyonunu doğru anahtar vaziyetleri ve anahtarların bağlı olduğu kapıları bulmak için çağırmalıdır. Cevabı bulduğunda ise `answer()` prosedürünü bir kez çağırmalıdır.

Parametreler

- `N`:Mağaradaki anahtar/kapı sayısı.

Örnekteki Durum

Kapılar ve anahtarlar resimdeki örnekte olduğu gibi ayarlanmış olsun:

Çağrılan Fonksiyon	Dönen Değer	Açıklama
`tryCombination([1, 0, 1, 1])`	`1`	Bu resimdeki duruma karşılık gelir: 0, 2 ve 3.ncü anahtarlar aşağı, 1. anahtar ise yukarı. Bu fonksiyon `1` döndürür, çünkü soldan sağa doğru ilk kapalı kapı 1.nci kapıdır.
`tryCombination([0, 1, 1, 0])`	`3`	0, 1 ve 2. kapılar açılır, fakat 3. kapı kapanır.
`tryCombination([1, 1, 1, 0])`	`-1`	0. anahtarı aşağı vaziyete getirmek tüm kapıları açar ve böylelikle `-1` döner.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Program sonlanır)*	Buradan doğru kombinasyonun `[1, 1, 1, 0]` olduğunu , ve 0, 1, 2 ve 3 nolu anahtarların sırasıyla 3, 1, 0 ve 2 nolu kapılara bağlı olduğunu buluruz.

Kısıtlar

- Süre sınırı: 2 saniye
- Hafıza sınırı: 32 MiB
- $1 \le N \le 5,000$

Altgörevler

Altgörev	Puan	İlave Girdi Kısıtları
1	12	Her bir `i`.nci anahtar, `i`.nci kapıya bağlıdır. Sizin asıl göreviniz doğru anahtar vaziyeti kombinasyonunu bulmaktır.
2	13	Doğru anahtar vaziyeti kombinasyonu her zaman `[0, 0, 0, …, 0]` dır. Sizin asıl göreviniz hangi anahtarın hangi kapıya bağlı olduğunu bulmaktır.
3	21	$N \le 100$
4	30	$N \le 2,000$
5	24	*(Kısıt yok)*

Test etme

Bilgisayarınızdaki örnek grader, `cave.in` adındaki girdi dosyasını okumaktadır. Bu dosyanın formatı:

- satır 1: `N`
- satır 2: `S[0] S[1] … S[N - 1]`
- satır 3: `D[0] D[1] … D[N - 1]`

Burada `N` kapı/anahtar sayısıdır, `S[i]` ise `i`.nci anahtarın doğru vaziyetidir, ve `D[i]` ise `i`.nci anahtarın bağlı olduğu kapıdır.

Örneğin, yukarıdaki ilk örnek aşağıdaki formatta verilmelidir:

```
4
1 1 1 0
3 1 0 2
```

Programlama Dili Notları

C/C++	`"cave.h"` dosyasını include etmelisiniz.
Pascal	Önce `unit Cave` i tanımlamalısınız, ayrıca grader prosedürlerini `uses GraderHelpLib` ile import etmelisiniz. Bütün dizi indisleri `0`'dan başlar.

Örnekler için bilgisayarınızdaki çözüm şablonlarını inceleyebilirsiniz.