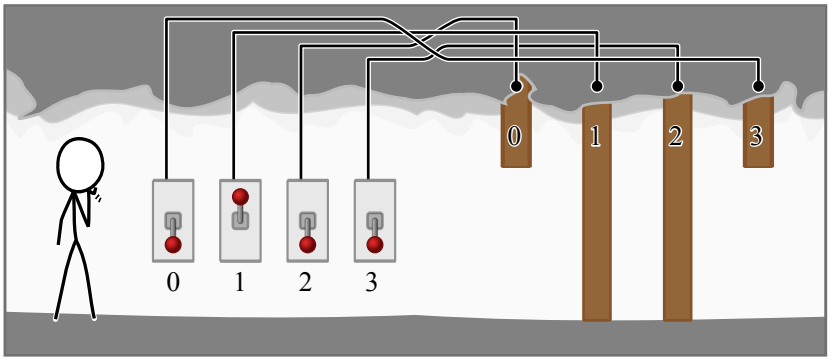
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

Hang động

Tiếng Việt — 1.0

Trong khi lạc đường đi từ ký túc xá đến Trung tâm UQ, bạn tình cờ đi qua lối vào dẫn tới một hệ thống hang động bí mật chạy ngầm sâu dưới ngôi trường. Lối vào bị chặn bởi một hệ thống an ninh bao gồm N cánh cửa liên tiếp nhau, cánh cửa này tiếp đến cánh cửa kia; và N công tắc, mỗi công tắc được nối với một cánh cửa khác nhau.



Các cánh cửa được đánh số theo thứ tự $0, 1, \ldots, (N - 1)$, cánh cửa 0 ở gần với bạn nhất. Các công tắc cũng được đánh số $0, 1, \ldots, (N - 1)$, tuy nhiên bạn không biết mỗi công tắc được nối với cánh cửa nào.

Tất cả các công tắc được đặt ở lối vào của hang động. Mỗi công tắc hoặc ở vị trí *trên* hoặc ở vị trí *dưới*. Chỉ có một trong các vị trí đó là đúng cho mỗi công tắc. Nếu một công tắc ở vị trí đúng thì cánh cửa nối với công tắc đó sẽ được mở, và nếu công tắc ở vị trí sai thì cánh cửa nối với nó sẽ đóng. Vị trí đúng có thể khác nhau đối với các công tắc khác nhau, và bạn không biết những vị trí nào là đúng.

Bạn muốn hiểu về hệ thống an ninh này. Để làm việc đó, bạn có thể đặt lại các công tắc theo bất cứ tổ hợp nào, và sau đó đi vào hang động để xem cánh cửa đầu tiên bị đóng là cánh cửa nào. Các cánh cửa không trong suốt: một khi bạn gặp cánh cửa đầu tiên bị đóng, bạn không thể nhìn thấy bất cứ cánh cửa nào phía sau nó.

Bạn có thời gian để thử $70,000$ tổ hợp của các công tắc, nhưng không được thử nhiều hơn. Nhiệm vụ của bạn là xác định vị trí đúng cho mỗi công tắc, và mỗi công tắc được nối với cánh cửa nào.

Cài đặt

Bạn cần nộp file cài đặt thủ tục `exploreCave()`. Thủ tục này có thể gọi hàm chấm điểm `tryCombination()` tối đa 70,000 lần, và phải kết thúc bằng việc gọi thủ tục chấm điểm `answer()`. Các hàm và thủ tục này được mô tả ở phía dưới.

Hàm chấm điểm: `tryCombination()`

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

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

Mô tả

Hệ thống chấm điểm sẽ cung cấp hàm này. Nó cho phép bạn thử một tổ hợp của các công tắc; và sau đó đi vào hang động để xác định cánh cửa đầu tiên bị đóng. Nếu tất cả các cánh cửa đều mở, thì hàm này sẽ trả lại giá trị `-1`. Hàm này chạy trong thời gian `O(N)`; tức là, thời gian chạy trong trường hợp tồi nhất tỉ lệ với `N`.

Hàm này có thể được gọi nhiều nhất là `70,000` lần.

Các thông số

- `S`: Mảng có độ dài `N`, cho biết vị trí của mỗi công tắc. Phần tử `S[i]` tương ứng với công tắc `i`. Giá trị `0` cho biết công tắc ở vị trí trên, và giá trị `1` cho biết công tắc ở vị trí dưới.
- *Giá trị trả lại*: Chỉ số của cánh cửa đầu tiên bị đóng, hoặc `-1` nếu tất cả các cánh cửa đều mở.

Thủ tục chấm điểm: `answer()`

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

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

Mô tả

Gọi thủ tục này khi bạn đã xác định được tổ hợp của các công tắc để mở được tất cả các cánh cửa, và mỗi công tắc được nối với cánh cửa nào.

Các thông số

- `S`: Mảng có độ dài `N`, cho biết vị trí đúng của mỗi công tắc. Định dạng giống như trong hàm `tryCombination()` được mô tả ở trên.
- `D`: Mảng có độ dài `N`, cho biết cánh cửa mà mỗi công tắc nối tới. Cụ thể là, phần tử `D[i]` cần chứa chỉ số của cánh cửa mà công tắc `i` nối tới nó.
- *Giá trị trả lại*: Thủ tục này không trả lại giá trị, nhưng sẽ khiến chương trình kết thúc.

Thủ tục mà bạn phải xây dựng: `exploreCave()`

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

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

Mô tả

Bạn phải cài đặt và nộp thủ tục này.

Thủ tục này cần sử dụng chương trình chấm `tryCombination()` để xác định vị trí đúng cho mỗi công tắc và cánh cửa mà mỗi công tắc nối tới, và phải gọi `answer()` khi đã xác định được thông tin này.

Các thông số

- `N`: Số lượng các công tắc và cánh cửa trong hang động.

Phần ví dụ

Giả sử các cánh cửa và các công tắc được sắp xếp như trong hình vẽ ở trên:

Gọi hàm	Giá trị trả lại	Giải thích
`tryCombination([1, 0, 1, 1])`	`1`	Thử nghiệm này tương ứng với hình vẽ. Các công tắc 0, 2 và 3 ở vị trí dưới, còn công tắc 1 ở vị trí trên. Hàm này sẽ trả lại `1`, cho biết cánh cửa số 1 là cánh cửa đầu tiên tính từ bên trái bị đóng.
`tryCombination([0, 1, 1, 0])`	`3`	Các cánh cửa số 0, 1 và 2 đều mở, trong khi cánh cửa số 3 bị đóng.
`tryCombination([1, 1, 1, 0])`	`-1`	Chuyển công tắc 0 về vị trí dưới sẽ làm cho tất cả các cánh cửa đều mở, điều này được nhận biết bởi giá trị trả lại bằng `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Kết thúc chương trình)*	Chúng ta đoán rằng tổ hợp đúng là `[1, 1, 1, 0]`, và các công tắc 0, 1, 2 và 3 được nối tương ứng với các cánh cửa 3, 1, 0 và 2.

Các ràng buộc

- Giới hạn về thời gian: 2 giây
- Giới hạn về bộ nhớ: 32 MiB (megabyte)
- $1 \le N \le 5{,}000$

Subtasks

Subtask	Điểm	Các ràng buộc thêm về dữ liệu đầu vào
1	12	Với mỗi `i`, công tắc `i` được nối với cánh cửa `i`. Nhiệm vụ của bạn chỉ đơn giản là xác định được tổ hợp đúng.
2	13	Tổ hợp đúng luôn là `[0, 0, 0, …, 0]`. Nhiệm vụ của bạn chỉ đơn giản là xác định xem mỗi công tắc được nối với cánh cửa nào.
3	21	$N \le 100$
4	30	$N \le 2{,}000$
5	24	*(Không có)*

Thực nghiệm

Chương trình chấm mẫu trên máy của bạn sẽ đọc dữ liệu vào từ file `cave.in`, file được ghi nhận theo khuôn dạng sau:

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

Ở đây `N` là số lượng các cánh cửa và các công tắc, `S[i]` là vị trí đúng của công tắc `i`, và `D[i]` là cánh cửa mà công tắc `i` được nối tới.

Chẳng hạn, ví dụ ở trên cần được cho theo khuôn dạng sau:

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

Chú ý về ngôn ngữ

C/C++	Bạn phải `#include "cave.h"`.
Pascal	Bạn phải định nghĩa `unit Cave`, và bạn cũng phải nạp các chương trình con chấm điểm bằng cách `uses GraderHelpLib`. Tất cả các mảng được đánh chỉ số bắt đầu từ `0` (không phải `1`).

Xem các lời giải mẫu trên máy của bạn, như là ví dụ.