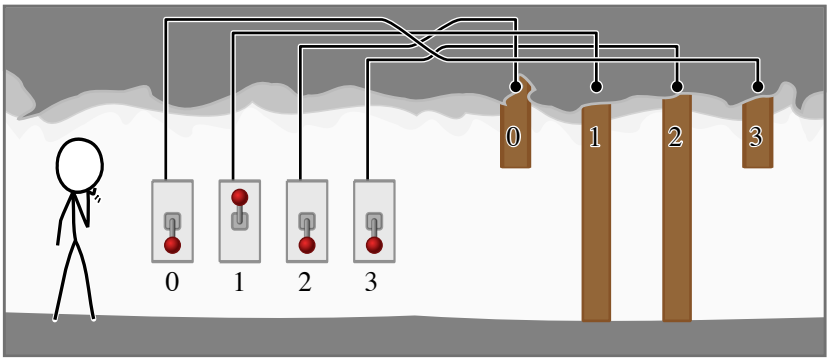
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Italian — 1.0

Vagando perduto lungo il tragitto dal college all'UQ Centre, ti sei imbattuto nell'ingresso di un sistema segreto di tunnel che si estende in profondità al di sotto dell'università. L'entrata è bloccata da un sistema di sicurezza consistente in `N` porte consecutive (ciascuna dietro alla precedente) ed `N` interruttori connessi ciascuno con una porta distinta.



Le porte sono numerate nell'ordine `0, 1, ..., (N - 1)`, dove la porta 0 è la più vicina all'ingresso del tunnel. Gli interruttori sono analogamente numerati `0, 1, ..., (N - 1)`, ma non sai quale interruttore sia connesso a quale porta.

Gli interruttori sono tutti posti in un unico punto all'ingresso del tunnel. Ogni interruttore può essere *in alto* oppure *in basso*, ed esattamente una e una sola di queste due posizioni è *corretta* per ogni interruttore. Se un interruttore è nella posizione corretta la porta a lui connessa sarà aperta, altrimenti sarà chiusa. La posizione corretta varia per ogni interruttore, e non è noto quali siano le posizioni corrette.

Vorresti trovare il modo di comprendere i dettagli del funzionamento di questo sistema di sicurezza. Per farlo, puoi impostare gli interruttori a una qualunque combinazione, e quindi camminare lungo il tunnel per verificare quale è la prima porta chiusa. Le porte non sono trasparenti, quindi non è possibile carpire informazioni sulle porte dietro di essa.

Hai tempo di provare `70 000` differenti posizioni degli interruttori, ma non di più. Il tuo compito è di determinare la posizione corretta e a quale porta è connesso ogni interruttore.

Implementazione

Devi sottoporre un file che implementi la procedura `exploreCave()`. Questa procedura può richiamare la funzione del grader `tryCombination()` fino a 70 000 volte, e deve terminare chiamando la procedura del grader `answer()`. Queste funzioni e procedure sono descritte nel seguito.

Funzione del grader: `tryCombination()`

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

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

Descrizione

Il grader conterrà questa funzione, che ti consente di provare una disposizione di interruttori e quindi entrare nel tunnel per determinare la prima porta chiusa. Se tutte le porte sono aperte, la funzione restituisce `-1`. Questa funzione è implementata in tempo `O(N)`; cioè il tempo di esecuzione è proporzionale ad `N` nel caso peggiore.

Può essere chiamata al massimo `70 000` volte.

Parametri

- `S`: Un array di lunghezza `N`, che indica la posizione di ogni interruttore. L'elemento `S[i]` corrisponde all'interruttore `i`. Un valore di `0` indica che l'interruttore è in alto, mentre un valore di `1` indica che l'interruttore è in basso.
- *Restituisce*: Il numero della prima porta chiusa, o `-1` se tutte le porte sono aperte.

Procedura del grader: `answer()`

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

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

Descrizione

Questa procedura deve essere chiamata quando hai identificato la posizione degli interruttori che apre tutte le porte e la porta a cui ogni interruttore è connesso.

Parametri

- `S`: Un array di lunghezza `N`, riportante la posizione corretta per ogni interruttore. Il formato corrisponde a quello della funzione `tryCombination()` descritta sopra.
- `D`: Un array di lunghezza `N`, riportante la porta a cui ogni interruttore è connesso. Precisamente, l'elemento `D[i]` deve corrispondere al numero della porta a cui l'interruttore `i` è connesso.
- *Restituisce*: Questa procedura non ha valore di ritorno, ma farà terminare il programma.

Procedura: `exploreCave()`

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

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

Descrizione

La tua sottoposizione deve implementare questa procedura.

Questa procedura deve usare la routine del grader `tryCombination()` per determinare la posizione corretta di ogni interruttore e a quale porta ciascun interruttore è connesso, e quindi chiamare `answer()` appena ha determinato queste informazioni.

Parametri

- `N`: Il numero di interruttori e porte nel tunnel.

Sessioni di esempio

La seguente sessione di esempio fa riferimento al disegno presente all'inizio.

Chiamata a funzione	Valore di ritorno	Spiegazione
`tryCombination([1, 0, 1, 1])`	`1`	Come nel disegno, gli interruttori 0, 2 e 3 sono in basso, mentre l'interruttore 1 è in alto. La funzione ritorna `1`, dato che la porta 1 è la prima chiusa da sinistra.
`tryCombination([0, 1, 1, 0])`	`3`	Le porte 0, 1 e 2 sono aperte, mentre la porta 3 è chiusa.
`tryCombination([1, 1, 1, 0])`	`-1`	Abbassare l'interruttore 0 fa aprire tutte le porte, come indicato dal valore di ritorno `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Il programma termina)*	Il programma risponde che le posizioni corrette sono quindi `[1, 1, 1, 0]`, e gli interruttori 0, 1, 2 e 3 sono connessi alle porte 3, 1, 0 e 2 rispettivamente.

Limiti

- Tempo limite: 2 secondi
- Limite di memoria: 32 MiB
- $1 \le N \le 5\,000$

Subtask

Subtask	Punteggio	Limiti aggiuntivi
1	12	Per ogni `i`, l'interruttore `i` è connesso alla porta `i`. Il tuo compito è quindi quello di determinare la disposizione corretta degli interruttori.
2	13	La disposizione corretta degli interruttori è sempre `[0, 0, 0, …, 0]`. Il tuo compito è quindi quello di determinare a quale porta è connesso ciascun interruttore.
3	21	$N \le 100$
4	30	$N \le 2\,000$
5	24	*(Nessun limite aggiuntivo)*

Testing

Il grader di esempio legge l'input dal file `cave.in`, che deve essere nel seguente formato:

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

Qui `N` è il numero di porte e interruttori, `S[i]` è la posizione corretta per l'interruttore `i`, e `D[i]` è la porta a cui l'interruttore `i` è connesso.

L'esempio riportato sopra può essere fornito nel seguente formato:

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

Note relative al linguaggio

C/C++	Devi inserire `#include "cave.h"`.
Pascal	Devi definire `unit Cave`, e devi anche importare le routine del grader tramite `uses GraderHelpLib`. Tutti gli array sono numerati a partire da `0` (e non `1`).

Vedi i template di soluzione nel tuo computer per alcuni esempi.