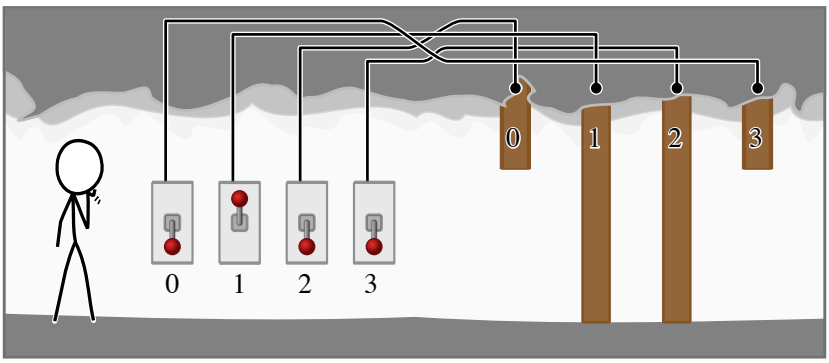
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Português — 1.0

Enquanto estava perdido na longa caminhada desde o colégio até ao UQ Centre, você descobriu a entrada para um sistema de cavernas secretas debaixo da universidade. A entrada está bloqueada por um sistema de segurança que consiste em `N` portas consecutivas, cada uma atrás da anterior; e `N` interruptores, cada um ligado a uma porta diferente.



As portas são numeradas `0, 1, ..., (N - 1)` por ordem, com a porta 0 sendo a mais próxima de si. Os interruptores são também numerados `0, 1, ..., (N - 1)`, embora você não saiba qual interruptor está ligado a qual porta.

Os interruptores estão localizados na entrada da caverna. Cada interruptor pode estar na posição *cima* ou *baixo*. Apenas uma destas posições é a correta para cada interruptor. Se um interruptor está na posição correcta então a porta a que está ligado irá estar aberta, se o interruptor está na posição incorreta então a porta a que está ligado irá estar fechada. A posição correta pode ser diferente para diferentes interruptores, e você não sabe que posições são as corretas.

Você deseja compreender este sistema de segurança. Para isso, pode colocar os interruptores numa qualquer combinação, e depois andar para dentro da caverna e observar qual é a primeira porta fechada. As portas não são transparentes: uma vez que encontre a primeira porta fechada, você não consegue ver nenhuma das portas que estão por trás.

Você tem tempo para tentar `70,000` combinações de interruptores, e nenhuma mais. A sua tarefa é determinar qual a posição correta de cada interruptor e também a que porta está ligado cada interruptor.

Implementação

Você deve submeter um ficheiro que implemente o procedimento `exploreCave()`. Este pode chamar a função do avaliador `tryCombination()` até 70,000 vezes e deve terminar chamando a função do avaliador `answer()`. Estas funções e procedimentos são descritas a seguir.

Função do avaliador: `tryCombination()`

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

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

Descrição

O avaliador irá providenciar esta função. Ela permite que você tente uma combinação de interruptores e que depois entre na caverna para determinar a primeira porta fechada. Se todas as portas estão abertas, a função retorna `-1`. Esta função executa em tempo `O(N)`, isto é, o tempo de execução é proporcional a `N`.

Esta função pode ser chamada no máximo `70,000` vezes.

Parâmetros

- `S`: Um vetor de tamanho `N`, indicando a posição de cada interruptor. O elemento `S[i]` corresponde ao interruptor `i`. Um valor de `0` indica que o interruptor está para cima, e um valor de `1` indica que o interruptor está para baixo.
- *Retorna*: o número da primeira porta fechada, ou `-1` se todas as portas estão abertas.

Procedimento do avaliador: `answer()`

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

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

Descrição

Chame este procedimento quando tiver identificado a combinação de interruptores que abre todas as portas, e a porta a que cada interruptor está ligado.

Parâmetros

- `S`: um vetor de tamanho `N`, indicando a posição correcta de cada interruptor. O formato corresponde ao da função `tryCombination()` atrás descrita.
- `D`: Um vetor de tamanho `N`, indicando a porta a que cada interruptor está ligado. Especificamente, o elemento `D[i]` deve conter o número do interruptor `i` a que está ligado.
- *Retorna*: Este procedimento não retorna, mas antes causa a terminação do programa.

O seu procedimento: `exploreCave()`

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

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

Descrição

A sua submissão deve implementar este procedimento.

Esta função deve usar o procedimento do avaliador `tryCombination()` para determinar qual a posição correta de cada interruptor e a porta a que cada interruptor está ligado, e deve chamar `answer()` uma vez que tenha determinado esta informação.

Parâmetros

- `N`: O número de interruptores e portas da cave.

Sessão Exemplo

Suponha que as portas e os interruptores estão dispostos como na figura anterior.

Chamada de função	Retorno	Explicação
`tryCombination([1, 0, 1, 1])`	`1`	Corresponde à imagem. Os interruptores 0, 2 e 3 estão em baixo, enquanto o interruptor 1 está em cima. A função retorna `1`, indicando que a porta 1 é a primeira porta fechada a partir da esquerda.
`tryCombination([0, 1, 1, 0])`	`3`	Portas 0, 1 2 estão todas abertas, enquanto a porta 3 está fechada.
`tryCombination([1, 1, 1, 0])`	`-1`	Mover o interruptor 0 para baixo faz com que todas as portas fiquem abertas, o que é indicado pelo valor de retorno `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(o programa sai)*	Nós adivinhamos que a combinação correcta é `[1, 1, 1, 0]`, e que os interruptores 0, 1, 2 e 3 ligam às portas 3, 1, 0 e 2

Restrições

- Limite de tempo: 2 segundos
- Limite de memória: 32 MiB
- $1 \le N \le 5{,}000$

Sub-tarefas

Sub-tarefa	Pontos	Restrições de entrada adicionais
1	12	Para cada `i`, o interruptor `i` está ligado à porta `i`. A sua tarefa é simplesmente descobrir a combinação correta.
2	13	A combinação correcta será sempre `[0, 0, 0, …, 0]`. A sua tarefa é simplesmente determinar qual interruptor liga com qual porta.
3	21	$N \le 100$
4	30	$N \le 2{,}000$
5	24	*(Nenhuma)*

Experimentação

O avaliador exemplo no seu computador irá ler o input a partir do ficheiro `cave.in`, que deve estar no seguinte formato:

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

Aqui `N` é o número de portas e de interruptores, `S[i]` é a posição correta para o interruptor `i`, e `D[i]` é a porta à qual o interruptor `i` está ligado.

O exemplo de cima seria providenciado no seguinte formato.

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

Notas das Linguagens

C/C++	Você deve incluir `#include "cave.h"`.
Pascal	Você deve definir a `unit Cave`, e deve também importar os procedimentos do avaliador via `uses GraderHelpLib`. Todos os vetores estão numerados a começar do `0` (e não `1`).

Veja os templates de solução na sua máquina para exemplos.