Giro in bici (ciclismo)

Limite di tempo:	1.0 secondi
Limite di memoria:	256 MiB

Giorgio è appassionato di ciclismo, tanto che ogni giorno prende un treno per raggiungere una nuova località e da lì farsi un bel giro in bici. La mappa della località in cui si è recato oggi consiste di N incroci collegati da M strade bidirezionali (non esistono sensi unici per le biciclette). Ognuno degli incroci, inoltre, si trova a una diversa altitudine H_i.

Nel suo giro, Giorgio partirà dalla stazione (corrispondente all'incrocio numero 0) e procederà ogni volta dirigendosi verso l'incrocio con altitudine più bassa tra quelli collegati all'incrocio corrente (Giorgio non ama le salite!), eccettuato quello da cui al momento arriva (in altre parole, non fa mai inversione a U). Il giro in bici prosegue quindi fintanto che si troverebbe costretto a effettuare un'inversione a U oppure quando ritorna in un incrocio da cui è già passato. In quale incrocio finirà il giro in bici?

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

> ☞ Tra gli allegati a questo task troverai un template (`ciclismo.c`, `ciclismo.cpp`, `ciclismo.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	`int pedala(int N, int M, int H[], da[], int a[]);`
Pascal	`function pedala(N, M: longint; var H, da, a: array of longint): longint;`

In cui:

- L'intero N rappresenta il numero di incroci presenti.
- L'intero M rappresenta il numero di strade presenti.
- L'array `H`, indicizzato da 0 a $N-1$, contiene le altitudini degli incroci.
- Gli array `da` e `a`, indicizzati da 0 a $M-1$, contengono la descrizione delle strade (per cui la strada i-esima collega gli incroci `da[`i`]` e `a[`i`]`).
- La funzione dovrà restituire l'incrocio in cui termina il giro in bici, che verrà stampato sul file di output.

Dati di input

Il file `input.txt` è composto da $M+2$ righe. La prima riga contiene i due interi N e M. La seconda riga contiene gli N interi H_i separati da uno spazio. Le successive M righe contengono ciascuna i due interi `da[`i`]` e `a[`i`]`.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Assunzioni

- $2 \leq N \leq 10\,000$.
- $1 \leq M \leq 100\,000$.
- $0 \leq H_i \leq 1\,000\,000$ per ogni $i = 0 \ldots N-1$.
- $0 \leq \texttt{da}_i, \texttt{a}_i < N$ e $\texttt{da}_i \neq \texttt{a}_i$ per ogni $i = 0 \ldots M-1$.
- Le altitudini sono tutte distinte e non ci sono strade ripetute.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1** [**10 punti**]: Casi d'esempio.
- **Subtask 2** [**20 punti**]: $N \leq 10$.
- **Subtask 3** [**40 punti**]: $N \leq 100$.
- **Subtask 4** [**30 punti**]: Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
5 6 20 10 40 5 33 0 1 1 2 0 4 2 3 3 1 4 3	0
6 6 21 17 19 13 43 18 0 1 1 2 0 3 1 4 3 4 4 5	2

Spiegazione

Nel **primo caso di esempio**, il giro percorre gli incroci 0 – 1 – 3 – 4 – 0 e poi termina (dato che ha chiuso un ciclo).

Nel **secondo caso di esempio**, il giro percorre gli incroci 0 – 3 – 4 – 1 – 2 e poi termina (dato che non può più muoversi).