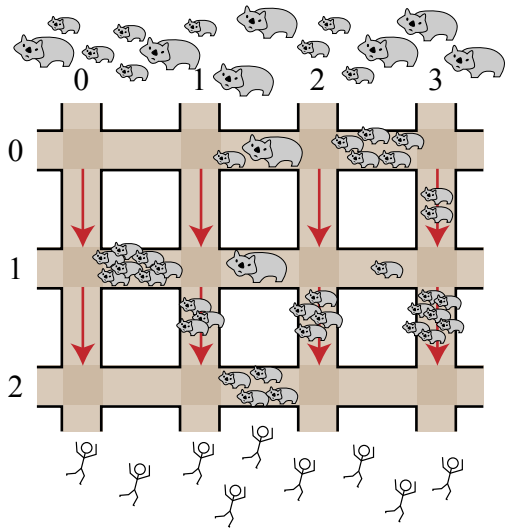
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia

wombats

Italian — 1.0

La metropoli di Brisbane è stata conquistata da enormi vombati mutanti, e tu devi condurre il popolo alla salvezza.

Le strade di Brisbane sono disposte secondo una griglia: `R` strade orizzontali, numerate `0`, ..., `(R - 1)` nell'ordine da nord a sud, e `C` strade verticali, numerate `0`, ..., `(C - 1)` nell'ordine da ovest a est, come nella figura sottostante.



I vombati procedono da nord, e la popolazione sta scappando verso sud. La popolazione scappa lungo le strade orizzontali in qualunque direzione, ma lungo le strade verticali si muove *unicamente verso sud*, ossia verso la salvezza.

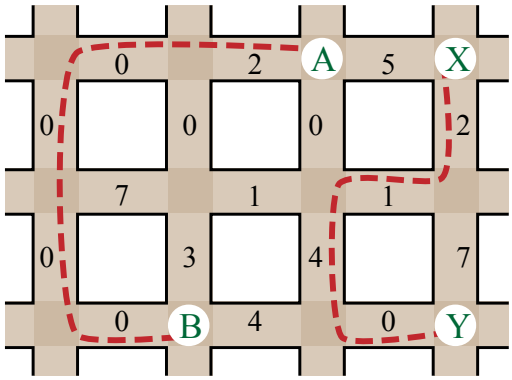
L'incrocio tra la strada orizzontale `P` e la strada verticale `Q` è denotato `(P,Q)`. Ogni segmento di strada tra due incroci contiene un certo numero di vombati che potrebbe cambiare nel tempo. Il tuo compito è guidare ogni persona da un incrocio a nord assegnato (lungo la strada orizzontale `0`) a un incrocio a sud assegnato (lungo la strada orizzontale `R-1`), guidandola lungo un tragitto che incontri il minor numero possibile di vombati.

Viene fornita la dimensione della griglia e il numero di vombati iniziale in ogni segmento di strada. Successivamente viene fornita una serie di `E` eventi, ciascuno dei quali può essere:

- un *change*, che modifica il numero di vombati in un segmento di strada; oppure
- un *escape*, in cui devi guidare una persona da un incrocio assegnato lungo la strada orizzontale `0` a un incrocio assegnato lungo la strada orizzontale `R - 1`, di modo che incontri il minor numero possibile di vombati.

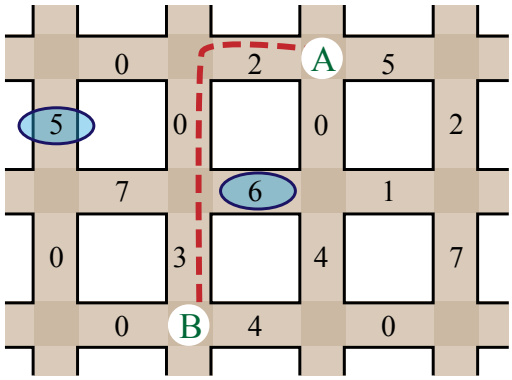
Devi gestire questi eventi implementando le routines `init()`, `changeH()`, `changeV()` and `escape()`, descritte sotto.

Esempio



Il disegno soprastante mostra una griglia iniziale con `R = 3` strade orizzontali e `C = 4` strade verticali, con indicato su ogni segmento il numero di vombati. Consideriamo la seguente successione di eventi:

- Una persona nell'incrocio `A = (0, 2)` intende fuggire verso l'incrocio `B = (2, 1)`. Il minor numero di vombati che deve incontrare è `2`, come indicato dalla linea tratteggiata.
- Un'altra persona nell'incrocio `X = (0, 3)` intende fuggire verso l'incrocio `Y = (2, 3)`. Il minor numero di vombati che deve incontrare è `7`, nuovamente indicato da una linea tratteggiata.
- Accadono due eventi *change*: il numero di vombati nel segmento più in alto della strada verticale `0` diventa `5`, e il numero di vombati nel segmento intermedio della strada orizzontale `1` diventa `6` (vedi i numeri cerchiati nella figura sottostante).



- Una terza persona intende fuggire dall'incrocio `A = (0, 2)` all'incrocio `B = (2, 1)`. Ora il minimo numero di vombati che deve incontrare è `5`, come indicato dalla nuova linea tratteggiata.

Implementazione

Devi sottoporre un file che implementi le procedure `init()`, `changeH()` e `changeV()` e la funzione `escape()`, come segue:

Procedura: `init()`

C/C++
```
void init(int R, int C, int H[5000][200], int V[5000][200]);
```

Pascal
```
type wombatsArrayType = array[0..4999, 0..199] of LongInt;

procedure init(R, C : LongInt; var H, V : wombatsArrayType);
```

Descrizione

Questa procedura fornisce la configurazione iniziale della griglia, e consente di inizializzare variabili globali e strutture di dati se necessario. Questa procedura viene chiamata un'unica volta, prima di ogni chiamata a `changeH()`, `changeV()` o `escape()`.

Parametri

- `R`: Il numero di strade orizzontali.
- `C`: Il numero di strade verticali.
- `H`: Un array bidimensionale di dimensioni `R × (C - 1)`, dove `H[P][Q]` è il numero di vombati sul segmento di strada orizzontale tra gli incroci `(P, Q)` e `(P, Q + 1)`.
- `V`: Un array bidimensionale di dimensioni `(R - 1) × C`, dove `V[P][Q]` è il numero di vombati lungo il segmento di strada verticale tra gli incroci `(P, Q)` and `(P + 1, Q)`.

Procedura: `changeH()`

C/C++ `void changeH(int P, int Q, int W);`

Pascal `procedure changeH(P, Q, W: LongInt);`

Descrizione

Questa procedura viene chiamata quando cambia il numero di vombati lungo il segmento di strada orizzontale tra gli incroci `(P, Q)` and `(P, Q + 1)`.

Parametri

- `P`: Indica la strada orizzontale influenzata (`0 ≤ P ≤ R - 1`).
- `Q`: Indica tra quali due strade verticali giace il segmento (`0 ≤ Q ≤ C - 2`).
- `W`: Il nuovo numero di vombati su questo segmento di strada (`0 ≤ W ≤ 1,000`).

Procedura: `changeV()`

C/C++ `void changeV(int P, int Q, int W);`

Pascal `procedure changeV(P, Q, W: LongInt);`

Descrizione

Questa procedura viene chiamata quando cambia il numero di vombati lungo il segmento di strada verticale tra gli incroci (P, Q) and $(P + 1, Q)$.

Parametri

- `P`: Indica tra quali due strade orizzontali giace il segmento ($0 \le P \le R - 2$).
- `Q`: Indica la strada verticale influenzata ($0 \le Q \le C - 1$).
- `W`: Il nuovo numero di vombati su questo segmento di strada ($0 \le W \le 1{,}000$).

Funzione: `escape()`

C/C++ `int escape(int V1, int V2);`

Pascal `function escape(V1, V2 : LongInt) : LongInt;`

Descrizione

La funzione deve determinare il minimo numero di vombati che una persona è costretta a incontrare spostandosi dall'incrocio $(0,V1)$ a $(R-1,V2)$.

Parametri

- `V1`: Indica il luogo di partenza sulla riga 0 ($0 \le V1 \le C-1$).
- `V2`: Indica il luogo di arrivo sulla riga $R-1$ ($0 \le V2 \le C-1$).
- *Restituisce*: Il minimo numero di vombati che la persona deve incontrare.

Sessione di esempio

La seguente sessione descrive l'esempio precedente:

Function Call	Returns
`init(3, 4, [[0,2,5], [7,1,1], [0,4,0]], [[0,0,0,2], [0,3,4,7]])`	
`escape(2,1)`	`2`
`escape(3,3)`	`7`
`changeV(0,0,5)`	
`changeH(1,1,6)`	
`escape(2,1)`	`5`

Limiti

- Tempo limite: 20 secondi
- Limite di memoria: 256 MiB
- $2 \le R \le 5\,000$
- $1 \le C \le 200$
- Al massimo 500 cambiamenti (chiamate a `changeH()` o `changeV()`)
- Al massimo 200 000 chiamate a `escape()`
- Al massimo 1 000 vombati su un qualsiasi segmento, in qualsiasi momento

Subtask

Subtask	Punti	Limiti aggiuntivi sull'input
1	9	$C = 1$
2	12	$R,C \le 20$, e non ci sono chiamate a `changeH()` o `changeV()`
3	16	$R,C \le 100$, e ci sono al massimo 100 chiamate a `escape()`
4	18	$C = 2$
5	21	$C \le 100$
6	24	*(Nessuno)*

Testing

Il grader di esempio sul tuo computer legge l'input dal file `wombats.in`, che deve rispettare il seguente formato:

- riga 1: `R C`
- riga 2: `H[0][0] … H[0][C-2]`
- …
- riga $(R + 1)$: `H[R-1][0] … H[R-1][C-2]`
- riga $(R + 2)$: `V[0][0] … V[0][C-1]`
- …
- riga $(2R)$: `V[R-2][0] … V[R-2][C-1]`
- riga successiva: `E`
- le `E` righe successive: un evento per riga, nell'ordine in cui gli eventi accadono

Se $C = 1$, le righe vuote che contengono il numero di vombati lungo le strade orizzontali (righe dalla `2` alla $R + 1$) non sono necessarie.

La riga relativa al singolo evento deve avere uno dei seguenti formati:

- per indicare `changeH(P, Q, W)`: `1 P Q W`
- per indicare `changeV(P, Q, W)`: `2 P Q W`
- per indicare `escape(V1, V2)`: `3 V1 V2`

In particolare, l'esempio precedente viene codificato in questo modo:

```
3 4
0 2 5
7 1 1
0 4 0
0 0 0 2
0 3 4 7
5
3 2 1
3 3 3
2 0 0 5
1 1 1 6
3 2 1
```

Note relative al linguaggio

C/C++	Devi inserire `#include "wombats.h"`.
Pascal	Devi definire `unit Wombats`. Tutti gli array sono numerati a partire da `0` (non da `1`).

Vedi i template delle soluzioni sulla tua macchina per alcuni esempi.