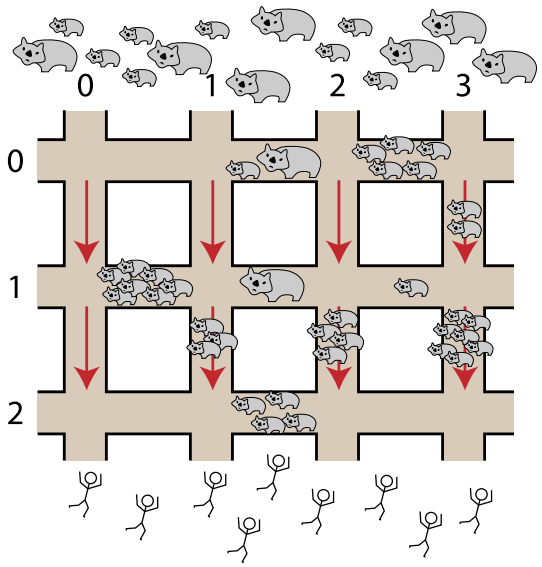
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia

Gấu túi

Tiếng Việt — 1.0

Một đàn gấu túi bị biến đổi gien đã tràn vào thành phố Brisbane, và bạn phải đưa mọi người đi lánh nạn đến địa điểm an toàn.

Các đường phố ở Brisbane được bố trí trên một lưới ô vuông lớn. Có `R` đường ngang chạy từ đông sang tây được đánh số `0, ..., (R - 1)` theo thứ tự từ bắc xuống nam, và `C` đường dọc chạy từ bắc xuống nam được đánh số `0, ..., (C - 1)` theo thứ tự từ tây sang đông, như chỉ ra ở hình vẽ dưới đây.



Các con gấu túi xâm chiếm từ hướng bắc, và mọi người thì chạy trốn về hướng nam. Mọi người có thể di chuyển trên các đường ngang theo hướng bất kỳ, nhưng trên các đường dọc họ *chỉ được di chuyển về hướng nam*, đi về hướng an toàn.

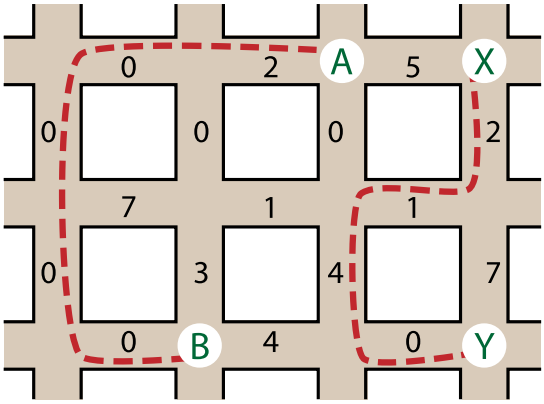
Nút giao giữa đường ngang `P` với đường dọc `Q` được ký hiệu là `(P, Q)`. Mỗi đoạn đường nối hai nút giao có một số con gấu túi, và số lượng này có thể thay đổi theo thời gian. Nhiệm vụ của bạn là hướng dẫn từng người đi từ một nút giao cho trước ở phía bắc (trên đường ngang số `0`) đến một nút giao cho trước ở phía nam (trên đường ngang số `R-1`) sao cho tổng số lượng các con gấu túi trên đường họ đi qua là ít nhất có thể được.

Để bắt đầu, bạn sẽ được cho biết kích thước của lưới ô vuông và số lượng các con gấu túi trên mỗi đoạn đường. Sau đó bạn sẽ được cho một chuỗi gồm `E` sự kiện, mỗi sự kiện có một trong hai dạng sau:

- một sự kiện *thay đổi* sẽ thay đổi số lượng các con gấu trên một đoạn đường nào đó; hoặc
- một sự kiện *chạy trốn* cho biết một người vừa đến một nút giao trên đường ngang `0`, và bạn phải tìm một đường đi đến một nút giao cho trước trên đường ngang `R - 1` sao cho số lượng con gấu túi trên đường người này đi qua là nhỏ nhất.

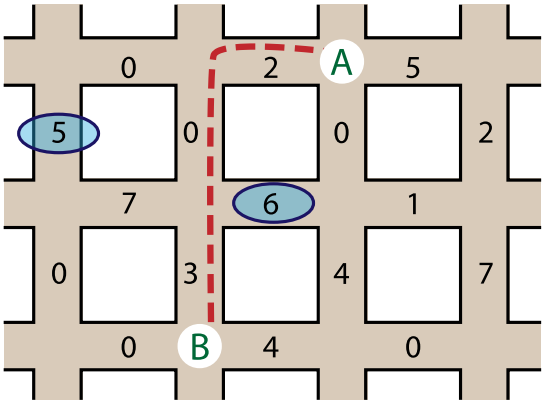
Bạn phải xử lý các sự kiện này bằng cách cài đặt các chương trình con `init()`, `changeH()`, `changeV()` và `escape()`, theo yêu cầu dưới đây.

Các ví dụ



Hình vẽ trên mô tả một bản đồ ban đầu có `R = 3` đường ngang và `C = 4` đường dọc, với số lượng các con gấu túi được ghi trên mỗi đoạn đường tương ứng. Xét chuỗi các sự kiện sau:

- Một người đến nút giao `A = (0, 2)` và muốn chạy trốn đến nút giao `B = (2, 1)`. Số lượng nhỏ nhất các con gấu túi trên đường người đó có thể đi qua là `2`, như được mô tả bằng đường gạch nối.
- Một người khác đến nút giao `X = (0, 3)` và mong muốn chạy trốn đến nút giao `Y = (2, 3)`. Số lượng nhỏ nhất các con gấu túi trên đường người đó có thể đi qua là `7`, như được mô tả bằng đường gạch nối.
- Hai sự kiện thay đổi xảy ra: số lượng gấu túi ở đoạn trên cùng của đường dọc `0` thay đổi thành `5`, và số lượng gấu túi ở đoạn giữa của đường ngang `1` thay đổi thành `6`. Xem các con số được khoanh tròn trong hình vẽ dưới đây.



- Người thứ ba đến nút giao `A = (0, 2)` và muốn chạy trốn đến nút giao `B = (2, 1)`. Bây giờ số lượng nhỏ nhất các con gấu túi trên đường người đó có thể đi qua là `5`, như được mô tả bằng đường gạch nối.

Cài đặt

Bạn cần nộp một file cài đặt các thủ tục `init()`, `changeH()` và `changeV()` và hàm `escape()`, như sau:

Thủ tục mà bạn phải xây dựng: `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);
```

Mô tả

Thủ tục này cho bạn biết bố trí ban đầu của bản đồ, và cho phép bạn khởi tạo các biến tổng thể và cấu trúc dữ liệu. Nó sẽ chỉ được gọi một lần, trước mọi lệnh gọi đến `changeH()`, `changeV()` hoặc `escape()`.

Các tham số

- `R`: Số lượng các đường ngang.
- `C`: Số lượng các đường dọc.
- `H`: Mảng hai chiều có kích thước `R × (C - 1)`, trong đó `H[P][Q]` cho biết số lượng gấu túi trên đoạn đường ngang nằm giữa nút giao `(P, Q)` và `(P, Q + 1)`.

- `V` : Mảng hai chiều có kích thước `(R - 1) × C` , trong đó `V[P][Q]` cho biết số lượng gấu túi trên đoạn đường dọc giữa hai nút giao `(P, Q)` và `(P + 1, Q)` .

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

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

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

Mô tả

Thủ tục này sẽ được gọi khi số lượng gấu túi thay đổi trên đoạn đường ngang giữa nút giao `(P, Q)` và `(P, Q + 1)` .

Các tham số

- `P` : Cho biết con đường ngang nào bị ảnh hưởng (`0 ≤ P ≤ R - 1`).
- `Q` : Cho biết đoạn đường nằm giữa hai con đường dọc nào (`0 ≤ Q ≤ C - 2`).
- `W` : Số lượng cập nhật các con gấu túi trên đoạn đường này (`0 ≤ W ≤ 1,000`).

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

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

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

Mô tả

Thủ tục này sẽ được gọi khi số lượng gấu túi thay đổi trên đoạn đường dọc giữa nút giao `(P, Q)` và `(P + 1, Q)` .

Các tham số

- `P` : Cho biết đoạn đường nằm giữa hai con đường ngang nào (`0 ≤ P ≤ R - 2`).
- `Q` : Cho biết con đường dọc nào bị ảnh hưởng (`0 ≤ Q ≤ C - 1`).
- `W` : Số lượng cập nhật các con gấu túi trên đoạn đường này (`0 ≤ W ≤ 1,000`).

Hàm mà bạn phải xây dựng: `escape()`

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

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

Mô tả

Hàm này cần tính số lượng các con gấu túi nhỏ nhất có thể được mà một người gặp phải trên đường di chuyển từ nút giao `(0, V1)` đến `(R-1, V2)`.

Các tham số

- `V1`: cho biết người này bắt đầu ở đâu trên đường ngang 0 (`0 ≤ V1 ≤ C-1`).
- `V2`: cho biết người này kết thúc ở đâu trên đường ngang `R-1` (`0 ≤ V2 ≤ C-1`).
- *Returns (giá trị trả về)*: Số lượng gấu túi nhỏ nhất trên đường người này phải đi qua.

Phần ví dụ

Phần dưới đây mô tả ví dụ ở trên:

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`

Các điều kiện ràng buộc

- Giới hạn thời gian: 20 giây
- Giới hạn bộ nhớ: 256 MiB (MegaByte)
- `2 ≤ R ≤ 5,000`
- `1 ≤ C ≤ 200`
- Không quá 500 thay đổi (các lần gọi đến `changeH()` hoặc `changeV()`)
- Không quá 200,000 lần gọi đến `escape()`

- Không quá 1,000 con gấu túi trên mỗi đoạn đường tại mỗi thời điểm.

Subtasks

Subtask	Điểm	Các ràng buộc thêm về đầu vào
1	9	$C = 1$
2	12	$R,C \le 20$, và sẽ không có lần gọi nào đến `changeH()` hoặc `changeV()`
3	16	$R,C \le 100$, và sẽ có tối đa 100 lần gọi đến `escape()`
4	18	$C = 2$
5	21	$C \le 100$
6	24	*(Không có)*

Thực nghiệm

Chương trình chấm mẫu trên máy tính của bạn sẽ đọc dữ liệu đầu vào từ file `wombats.in`, mà phải được cho dưới khuôn dạng sau:

- dòng 1: `R C`
- dòng 2: `H[0][0] … H[0][C-2]`
- …
- dòng $(R + 1)$: `H[R-1][0] … H[R-1][C-2]`
- dòng $(R + 2)$: `V[0][0] … V[0][C-1]`
- …
- dòng $(2R)$: `V[R-2][0] … V[R-2][C-1]`
- dòng tiếp theo: `E`
- `E` dòng tiếp theo: một sự kiện trên một dòng, theo thứ tự xuất hiện của các sự kiện.

Nếu $C = 1$, các dòng trống chứa số lượng gấu túi trên các đường ngang (dòng `2` đến $R + 1$) là không cần thiết.

Dòng chứa mỗi sự kiện phải theo một trong các khuôn dạng sau: :

- để mô tả `changeH(P, Q, W)`: `1 P Q W`
- để mô tả `changeV(P, Q, W)`: `2 P Q W`

- để mô tả `escape(V1, V2)`: `3 V1 V2`

Chẳng hạn, ví dụ ở trên cần được ghi nhận theo khuôn dạng sau đây:

```
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
```

Các lưu ý về ngôn ngữ lập trình

C/C++	Bạn phải `#include "wombats.h"`.
Pascal	Bạn phải định nghĩa `unit Wombats`. Tất cả các mảng được đánh chỉ số bắt đầu từ `0` (không phải `1`).

Hãy xem mẫu các lời giải trên máy tính của bạn như là những ví dụ.