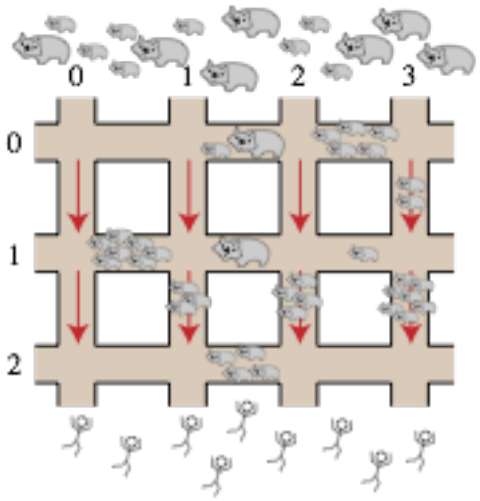
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia

wombats

Deutsch — 1.1

Brisbane wurde von mutierten Wombats (das sind australische Beuteltiere) eingenommen, und du sollst die Bevölkerung in Sicherheit bringen.

Die Straßen von Brisbane bilden ein Gitter. Es gibt `R` waagerechte Straßen von Ost nach West, die von Norden nach Süden mit `0, ..., (R - 1)` nummeriert sind, und `C` senkrechte Straßen von Nord nach Süd, die von Westen nach Osten mit `0, ..., (C - 1)` nummeriert sind (siehe Bild unten).



Die Wombats dringen von Norden her ein, und die Bevölkerung flieht nach Süden. Auf den waagerechten Straßen können sich die Leute in beide Richtungen bewegen, auf den senkrechten Straßen aber *nur in Richtung Süden*.

Die Kreuzung der waagerechten Straße `P` mit der senkrechten Straße `Q` wird mit `(P,Q)` bezeichnet. Auf jedem Straßenabschnitt zwischen zwei Kreuzungen befindet sich eine bestimmte Anzahl Wombats; diese Anzahl kann sich mit der Zeit ändern. Du sollst einzelne Personen von einer gegebenen Kreuzung im Norden (auf der waagerechten Straße `0`) zu einer anderen gegebenen Kreuzung im Süden (auf der waagerechten Straße `R - 1`) führen, und zwar entlang einer *optimalen Route*, also einer, auf der sich so wenige Wombats wie möglich befinden.

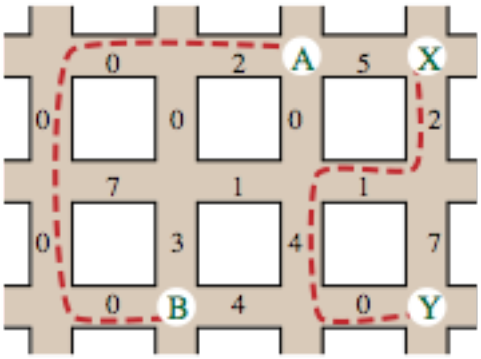
Am Anfang bekommst du die Größe des Gitters und für jeden Straßenabschnitt die Anzahl der Wombats. Danach folgen `E` Ereignisse, und zwar entweder

- ein *change*, d.h.: die Anzahl der Wombats auf einem Straßenabschnitt ändert sich; oder

- ein *escape*, d.h.: eine Person muss von einer gegebenen Kreuzung auf der waagerechten Straße `0` zu einer anderen gegebenen Kreuzung auf der waagerechten Straße `R - 1` gelangen, und zwar auf einer optimalen Route (mit möglichst wenigen Wombats).

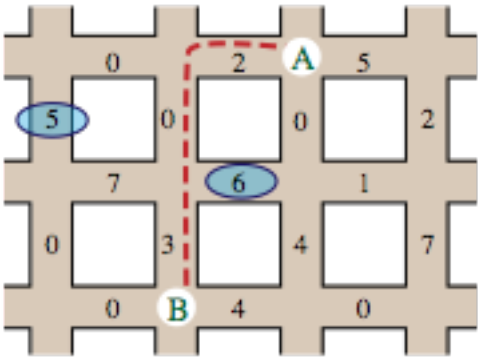
Um mit diesen Ereignissen umzugehen, musst du die Routinen `init()`, `changeH()`, `changeV()` und `escape()` implementieren, wie weiter unten beschrieben.

Beispiele



Das obige Bild zeigt ein Gitter mit `R = 3` waagerechten und `C = 4` senkrechten Straßen, wobei auf jedem Straßenabschnitt die anfängliche Zahl von Wombats angegeben ist. Dann folgen diese Ereignisse:

- *escape*: Eine Person muss von der Kreuzung `A = (0, 2)` zur Kreuzung `B = (2, 1)`. Eine optimale Route (mit `2` Wombats) ist im Bild oben als gestrichelte Linie eingezeichnet.
- *escape*: Eine andere Person muss von der Kreuzung `X = (0, 3)` zur Kreuzung `Y = (2, 3)`. Eine optimale Route (mit `7` Wombats) ist im Bild oben ebenfalls als gestrichelte Linie eingezeichnet.
- Zweimal *change*: Die Anzahl der Wombats auf dem oberen Abschnitt der senkrechten Straße `0` ändert sich zu `5`, und die Anzahl der Wombats auf dem mittleren Abschnitt der waagerechten Straße `1` ändert sich zu `6` (siehe die markierten Zahlen im folgenden Bild).



- *escape*: Eine dritte Person muss von der Kreuzung `A = (0, 2)` zur Kreuzung `B = (2, 1)`. Nun ist eine andere Route optimal (mit `5` Wombats).

Implementierung

Du sollst eine Datei einsenden, welche die Prozeduren `init()`, `changeH()` und `changeV()` sowie die Funktion `escape()` folgendermaßen implementiert:

Deine Prozedur: `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);
```

Beschreibung

Diese Prozedur spezifiziert den Anfangszustand des Gitters, und erlaubt dir jegliche globalen Variablen und Datenstrukturen zu initialisieren. Sie wird einmalig aufgerufen, und zwar vor dem ersten Aufruf von `changeH()`, `changeV()` oder `escape()`.

Parameter

- `R`: Die Anzahl waagerechter Straßen.
- `C`: Die Anzahl senkrechter Straßen.
- `H`: Ein zwei-dimensionales Array der Größe `R × (C - 1)`. `H[P][Q]` ist die Anzahl an Wombats auf dem waagerechten Straßenabschnitt zwischen Kreuzungen `(P, Q)` und `(P, Q + 1)`.
- `V`: Ein zwei-dimensionales Array der Größe `(R - 1) × C`. `V[P][Q]` ist die Anzahl an Wombats auf dem senkrechten Straßenabschnitt zwischen Kreuzungen `(P, Q)` und `(P + 1, Q)`.

Deine Prozedur: `changeH()`

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

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

Beschreibung

Diese Prozedur wird aufgerufen, wenn sich die Anzahl an Wombats auf dem waagerechten Straßenabschnitt zwischen Kreuzungen (P, Q) und $(P, Q + 1)$ ändert.

Parameter

- `P`: Besagt welche waagerechte Straße betroffen ist ($0 \le P \le R - 1$).
- `Q`: Besagt zwischen welchen zwei senkrechten Straßen der Straßenabschnitt liegt ($0 \le Q \le C - 2$).
- `W`: Die neue Anzahl an Wombats auf diesem Straßenabschnitt ($0 \le W \le 1.000$).

Deine Prozedur: `changeV()`

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

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

Beschreibung

Diese Prozedur wird aufgerufen, wenn sich die Anzahl an Wombats auf dem senkrechten Straßenabschnitt zwischen Kreuzungen (P, Q) und $(P + 1, Q)$ ändert.

Parameter

- `P`: Besagt zwischen welchen zwei waagerechten Straßen der Straßenabschnitt liegt ($0 \le P \le R - 2$).
- `Q`: Besagt welche senkrechte Straße betroffen ist ($0 \le Q \le C - 1$).
- `W`: Die neue Anzahl an Wombats auf diesem Straßenabschnitt ($0 \le W \le 1.000$).

Deine Funktion: `escape()`

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

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

Beschreibung

Diese Funktion soll die Anzahl an Wombats auf einer optimalen Route von der Kreuzung `(0, V1)` zur Kreuzung `(R-1, V2)` bestimmen.

Parameter

- `V1`: Besagt wo die Person in der waagerechten Straße 0 startet (`0 ≤ V1 ≤ C-1`).
- `V2`: Besagt wo die Person in der waagerechten Straße `R-1` ankommt (`0 ≤ V2 ≤ C-1`).
- *Rückgabe*: Die Anzahl an Wombats auf einer optimalen Route.

Beispiel-Session

Folgender Ablauf beschreibt obiges Beispiel:

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`

Beschränkungen

- Zeitlimit: 20 Sekunden
- Speicherlimit: 256 MiB
- `2 ≤ R ≤ 5.000`
- `1 ≤ C ≤ 200`
- Maximal 500 Änderungen (Aufrufe von `changeH()` oder `changeV()`)
- Maximal 200.000 Aufrufe von `escape()`
- Zu keiner Zeit mehr als 1.000 Wombats pro Straßenabschnitt.

Teilaufgaben

Teilaufgabe	Punkte	Zusätzliche Beschränkungen
1	9	$C = 1$
2	12	$R,C \leq 20$, keine Aufrufe von `changeH()` oder `changeV()`
3	16	$R,C \leq 100$, maximal 100 Aufrufe von `escape()`
4	18	$C = 2$
5	21	$C \leq 100$
6	24	*(Keine)*

Testen

Der Beispielgrader auf deiner Maschine liest die Eingabe aus der Datei `wombats.in`, aufgebaut wie folgt:

- Zeile 1: `R C`
- Zeile 2: `H[0][0] … H[0][C-2]`
- …
- Zeile $(R + 1)$: `H[R-1][0] … H[R-1][C-2]`
- Zeile $(R + 2)$: `V[0][0] … V[0][C-1]`
- …
- Zeile $(2R)$: `V[R-2][0] … V[R-2][C-1]`
- folgende Zeile: `E`
- folgende `E` Zeilen: Ein Ereignis pro Zeile, in genau der Reihenfolge wie sie eintreten.

Gilt $C = 1$, sind die leeren Zeilen mit der Anzahl an Wombats auf waagerechten Straßenabschnitten (Zeilen `2` bis $R + 1$) nicht nötig.

Das Format jeder Ereigniszeile muss eines der folgenden Formate berücksichtigen:

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

Das Format des obigen Beispiels sollte folgendes sein:

```
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
```

Sprachspezifische Bemerkungen

C/C++	Du musst `#include "wombats.h"` verwenden.
Pascal	Du musst `unit Wombats` definieren. Alle Arrays sind mit `0` beginnend indiziert. (nicht `1`).

Siehe Lösungsvorlagen auf deiner Maschine für Beispiele.