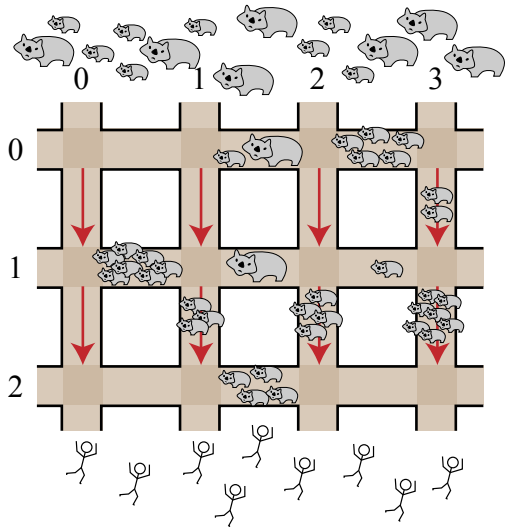
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia

wombats

Czech — 1.0

Město Brisbane bylo napadeno velkými zmutovanými lidožravými vombaty. Armáda bohužel ani přes obrovské ztráty nedokázala jejich útok odrazit, a tak nezbývá nic jiného, než se dát na zběsilý útěk.

V Brisbane je `R` horizontálních ulic vedoucích z východu na západ a `C` vertikálních ulic vedoucích ze severu na jih. Horizontální ulice jsou číslované od severu čísly `0, …, (R - 1)` a vertikální ulice jsou číslované od západu čísly `0, …, (C - 1)`, viz obrázek.



Vombati zaútočili ze severu, takže lidé musí utíkat na jih. Na horizontálních ulicích se mohou pohybovat libovolným směrem, na vertikálních mohou utíkat pouze směrem na jih, směrem do bezpečí.

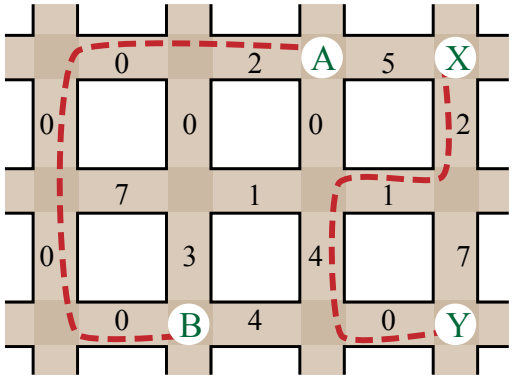
Křížení horizontální ulice `P` s vertikální ulicí `Q` označme jako `(P, Q)`. Každý úsek mezi dvěma kříženími je obsazen určitým počtem vombatů a tento počet se může v průběhu času měnit. Vaším úkolem je pomoct utéct osobě nacházející se na zadaném průsečíku nejsevernější ulice (ulice číslo `0`) do zadaného průsečíku nejjižnější ulice (ulice číslo `R-1`) tak, aby cestou potkal (a ochutnalo ho) co nejméně vombatů.

Na začátku dostanete počet horizontálních a vertikálních ulic a počty vombatů na všech úsecích mezi kříženími. Následuje posloupnost `E` událostí. Rozlišujeme dva druhy událostí:

- *změna* (*change*), která změní počet vombatů v nějakém úseku.
- *útěk* (*escape*), kdy nějaký obyvatel potřebuje utéct z určitého křížení na horizontální ulici č. `0` do určitého křížení na horizontální ulici č. `R-1` tak, aby potkal co nejméně vombatů.

Tyto události musíte ošetřit implementací následujících funkcí: `init()`, `changeH()`, `changeV()` a `escape()`, jejichž popis následuje.

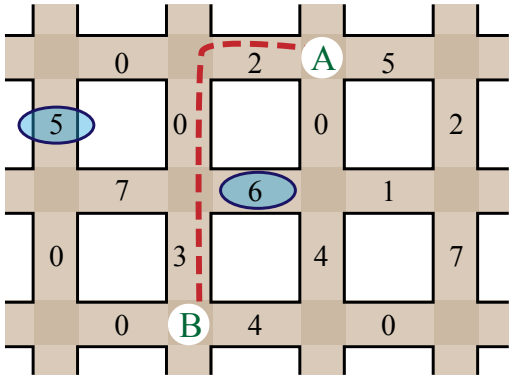
Příklad



Obrázek výše ukazuje mapu s `R = 3` horizontálními ulicemi, `C = 4` vertikálními a s počtem vombatů , který je uveden na každém úseku.

Uvažujte následující posloupnost událostí:

- Obyvatel potřebuje utéct z křížení `A = (0, 2)` do křížení `B = (2, 1)`. Nejmenší počet vombatů, které při útěku potká, je `2`, jak je znázorněno přerušovanou čarou.
- Jiný obyvatel potřebuje utéct z křížení `X = (0, 3)` do křížení `Y = (2, 3)`. Nejmenší počet vombatů, které potká, je `7`. Jeho útěk je opět znázorněn přerušovanou čarou.
- Nastaly dvě změny: počet vombatů na nejsevernějším úseku vertikální ulice č. `0` se změnil na `5` a počet vombatů na prostředním úseku horizontální ulice č. `1` se změnil na `6`. Tyto změny jsou znázorněny zakroužkovanými čísly na níže uvedeném obrázku.



- Třetí osoba potřebuje utéct z křížení `A = (0, 2)` do křížení `B = (2, 1)`. Nyní při útěku potká alespoň `5` vombatů. Útěk je znázorněn přerušovanou čárou.

Implementace

Odevzdejte soubor implementující následující funkce `init()`, `changeH()`, `changeV()` a `escape()`:

Vaše funkce: `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);
```

Popis

Tato funkce obdrží popis města a umožní vám inicializovat vaše globální proměnné a datové struktury. Tato funkce bude zavolána pouze jednou před funkcemi `changeH()`, `changeV()` a `escape()`.

Parametry

- `R`: počet horizontálních ulic.
- `C`: počet vertikálních ulic.
- `H`: dvojrozměrné pole velikosti `R × (C - 1)`, kde `H[P][Q]` určuje počet vombatů na horizontálním úseku mezi křížením `(P, Q)` a `(P, Q + 1)`.
- `V`: dvojrozměrné pole velikosti `(R - 1) × C`, kde `V[P][Q]` určuje počet vombatů na vertikálním úseku mezi křížením `(P, Q)` a `(P + 1, Q)`.

Vaše funkce: changeH()

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

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

Popis

Tato funkce bude zavolána, když se změní počet vombatů na úseku horizontální ulice mezi křížením `(P, Q)` a `(P, Q + 1)`.

Parametry

- `P`: určuje, na které horizontální ulici se počet vombatů změnil (`0 ≤ P ≤ R - 1`).
- `Q`: určuje, mezi kterými vertikálními ulicemi se počet vombatů změnil (`0 ≤ Q ≤ C - 2`).
- `W`: určuje nový počet vombatů na zadném úseku (`0 ≤ W ≤ 1 000`).

Vaše funkce: `changeV()`

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

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

Popis

Tato funkce bude zavolána, když se změní počet vombatů na úseku vertikální ulice mezi křížením `(P, Q)` a `(P + 1, Q)`.

Parametry

- `P`: určuje, mezi kterými horizontálními ulicemi se počet vombatů změnil (`0 ≤ P ≤ R - 2`).
- `Q`: určuje, na které horizontální ulici se počet vombatů změnil (`0 ≤ Q ≤ C - 1`).
- `W`: určuje nový počet vombatů na zadném úseku (`0 ≤ W ≤ 1 000`).

Vaše funkce: `escape()`

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

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

Popis

Tato funkce má za úkol spočítat kolik nejméně vombatů musí obyvatel potkat při útěku z křížení `(0, V1)` do křížení `(R-1, V2)`

Parametry

- `V1`: určuje z jakého křížení s horizontální ulicí č. `0` utíká (`0 ≤ V1 ≤ C-1`).
- `V2`: určuje do jakého křížení s horizontální ulicí č. `R-1` utíká (`0 ≤ V2 ≤ C-1`).
- *Návratová hodnota*: nejmenší počet vombatů, které obyvatel při útěku musí potkat.

Příklad

Následující příklad ukazuje posloupnost volání funkcí ve výše uvedené ukázce:

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`

Omezení

- Časový limit: 15 sekund
- Paměťový limit: 256 MiB
- $2 \le R \le 5\,000$
- $1 \le C \le 200$
- Nejvýše 500 změn (tj. volání funkcí `changeH()` nebo `changeV()`)
- Nejvýše 200 000 volání funkce `escape()`
- Nejvýše 1 000 vombatů na libovolném úseku v libovolnou chvíli

Podúlohy

Podúloha	Počet bodů	Dodatečné předpoklady
1	9	`C = 1`
2	12	`R,C ≤ 20` a žádné volání funkcí `changeH()` a `changeV()`
3	16	`R,C ≤ 100` a nejvýše 100 volání funkce `escape()`
4	18	`C = 2`
5	21	`C ≤ 100`
6	24	*(žádné)*

Testování

Vzorový testovač, který jste dostali k dispozici v adresáři této úlohy, bude číst soubor `wombats.in`, který musí mít následující formát:

- řádek 1: `R C`
- řádek 2: `H[0][0] … H[0][C-2]`
- …
- řádek `(R + 1)`: `H[R-1][0] … H[R-1][C-2]`
- řádek `(R + 2)`: `V[0][0] … V[0][C-1]`
- …
- řádek `(2R)`: `V[R-2][0] … V[R-2][C-1]`
- následující řádek: `E`
- následujících `E` řádků: jedna událost na každém z těchto řádků v pořadí, ve kterém nastaly

Pokud `C = 1`, pak prázdné řádky obsahující počty vombatů na horizontálních ulicích (tj. řádky `2` až `R + 1`) nejsou nezbytné.

Řádek pro každou událost musí být v následujícím formátu:

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

Předchozímu příkladu odpovídá následující vstupní soubor:

```
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
```

Poznámky k jednotlivým programovacím jazykům

C/C++	Vaše řešení musí obsahovat řádek `#include "wombats.h"`.
Pascal	Musíte definovat `unit Wombats`. Všechna pole jsou číslována od `0` (tj. nikoliv od `1`).

Viz kostry řešení v adresáři této úlohy.