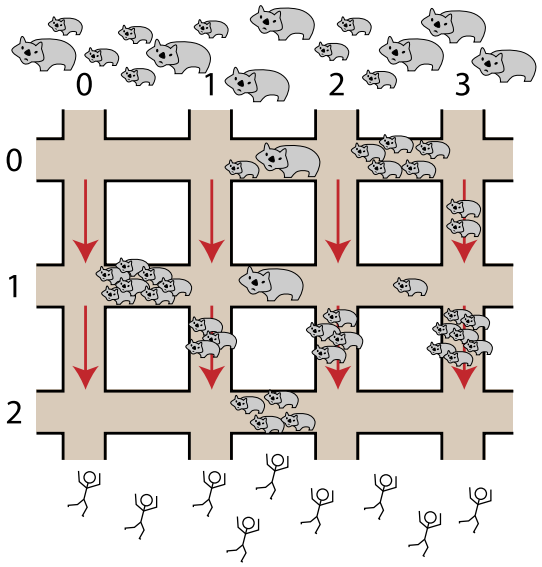
wombats

Latvian — 1.0

Lieli vombati-mutanti ir pārņēmuši Brisbenas pilsētu, un jums ir jānogādā tās iedzīvotāji drošībā.

Brisbenas ceļi ir veido lielu taisnstūrveida režģī. Ir `R` horizontāli ceļi kuri rietumu-austrumu virzienā un sanumurēti `0, ..., (R - 1)` no ziemeļiem uz dienvidiem, un `C` vertikāli ceļi ziemeļu-dienvidu virzienā un sanumurēti `0, ..., (C - 1)` no rietumiem uz austrumiem, kā ir redzams zemāk dotajā attēlā.



Vombati ir iekarojuši pilsētu no ziemeļiem un iedzīvotāji mūk uz dienvidiem. Iedzīvotāji var skriet pa horizontāliem ceļiem abos virzienos, bet viņi *skries tikai uz dienvidiem*, uz drošību, pa vertikāliem ceļiem.

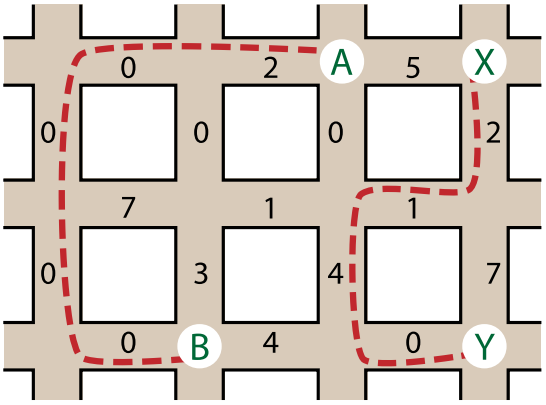
Horizontāla ceļa `P` un vertikāla ceļa `Q` krustojums ir apzīmēts ar `(P, Q)`. Katrs ceļa segments starp diviem krustojumiem satur kādu vombatu skaitu, un šie skaitļi var ar laiku mainīties. Jūsu uzdevums ir nogādāt katru personu no kāda dota krustojuma ziemeļos (uz horizontāla ceļa `0`) uz kādu dotu krustojumu dienvidos (uz horizontāla ceļa `R - 1`) izmantojot maršrutu, uz kura var sastapt pēc iespējas mazāku vombatu skaitu.

Sākumā ir doti režģa izmēri un vombatu skaits katrā ceļa segmentā. Pēc tam tiks doti `E` notikumi, kur katrs no notikumiem būs vai nu:

- *maiņa*, kur mainās vombatu skaits kādā ceļa segmentā; vai nu
- *bēgšana*, kur kāds iedzīvotājs ierodas dotā krustojumā uz horizontāla ceļa `0`, un jums ir jāatrod maršruts uz dotu punktu uz horizontāla ceļa `R - 1` kas satur pēc iespējas mazāku vombatu skaitu.

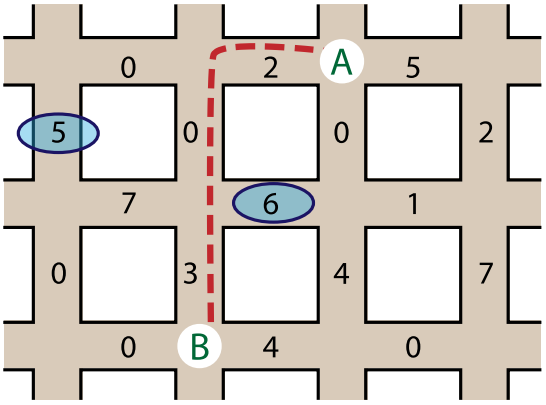
Jums ir jāapstrādā šie notikumi implementējot procedūras vai funkcijas `init()`, `ChangeH()`, `changeV()` un `escape()`, kā ir aprakstīts zemāk..

Piemēri



Attēlā augstāk redzams sākotnējais režģis ar `R = 3` horizontāliem un `C = 4` vertikāliem ceļiem, ar atzīmētu vombatu skaitu katrā segmentā. Apskatiet sekojošu notikumu secību:

- Iedzīvotājs ierodas krustojumā `A = (0, 2)` un vēlas aizbēgt uz krustojumu `B = (2, 1)`. Mazākais vombatu skaits ko viņš var sastapt ir *2*, kas ir parādīts ar raustītu līniju.
- Cits iedzīvotājs ierodas krustojumā `X = (0, 3)` un vēlas aizbēgt uz krustojumu `Y = (2, 3)`. Mazākais vombatu skaits ko viņš var sastapt ir *7*, kas atkal ir parādīts ar raustītu līniju.
- Notiek divi maiņas notikumi: vombatu skaits vertikāla ceļa `0` augšējā segmentā mainās uz `5`, un vombatu skaits horizontāla ceļa `1` vidējā segmentā mainās uz `6`. Skatīt apvilktos numurus zemāk dotajā attēlā.



- Trešais iedzīvotājs ierodas krustojumā `A = (0, 2)` un vēlas aizbēgt uz krustojumu `B = (2, 1)`. Tagad mazākais vombatu skaits ko viņš var sastapt ir *5*, kas ir parādīts ar jaunu raustītu līniju.

Implementācija

Jums ir jāiesūta fails kurš implementē procedūras `init()`, `changeH()` un `changeV()` ka arī funkciju `escape()`, sekojošā veidā:

Jūsu procedūra: `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);
```

Apraksts

Šī procedūra dod jums sākotnēju kartes izvietojumu, un atļauj jums inicializēt jebkurus globālos mainīgus un datu struktūras. Tā tiks izsaukta tikai vienreiz, pirms jebkādiem `changeH()`, `changeV()` vai `escape()` izsaukumiem.

Parametri

- `R`: Horizontālu ceļu skaits.
- `C`: Vertikālu ceļu skaits.
- `H`: Divdimensionāls masīvs ar izmēru $R \times (C - 1)$, kur `H[P][Q]` dod vombatu skaitu horizontālā ceļa segmentā starp krustojumiem (P, Q) un $(P, Q + 1)$.
- `V`: Divdimensionāls masīvs ar izmēru $(R - 1) \times C$, kur `V[P][Q]` dod vombatu skaitu vertikālā ceļa segmentā starp krustojumiem (P, Q) un $(P + 1, Q)$.

Jūsu procedūra: `changeH()`

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

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

Apraksts

Šī procedūra tiks izsaukta kad mainās vombatu skaits horizontālā ceļa segmentā starp krustojumiem (P, Q) un $(P, Q + 1)$.

Parametri

- `P`: Nosaka kuram horizontālam ceļam pieder maināmais segments ($0 \le P \le R - 1$).

- `Q`: Nosaka starp kuriem diviem vertikāliem ceļiem atrodas maināmais segments ($0 \le Q \le C - 2$).
- `W`: Jaunais vombatu skaits šajā ceļa segmentā ($0 \le W \le 1,000$).

Jūsu procedūra: `changeV()`

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

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

Apraksts

Šī procedūra tiks izsaukta kad mainās vombatu skaits uz vertikālā ceļa segmentā starp krustojumiem (P, Q) un $(P + 1, Q)$.

Parametri

- `P`: Nosaka starp kuriem diviem horizontāliem ceļiem atrodas maināmais segments ($0 \le P \le R - 2$).
- `Q`: Nosaka kuram vertikālam ceļam pieder maināmais segments ($0 \le Q \le C - 1$).
- `W`: Jaunais vombatu skaits šajā ceļa segmentā ($0 \le W \le 1,000$).

Jūsu funkcija: `escape()`

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

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

Apraksts

Šai funkcijai ir jānoskaidro minimālais vombatu skaits, kurš ir jāsastop iedzīvotājam ceļojot no krustojuma $(0, V1)$ uz $(R - 1, V2)$.

Parametri

- `V1`: Norāda, kur iedzīvotājs uzsāk savu ceļojumu uz horizontāla ceļa `0` ($0 \le V1 \le C - 1$).
- `V2`: Norāda, kur iedzīvotāks pabeidz savu ceļojumu uz horizontāla ceļa `R = 1` ($0 \le V2 \le C - 1$).
- *Rezultāts*: Mazākais vombatu skaits, kas jāsastop iedzīvotājam.

Piemēra sesija

Zemāk esošajā sesijā aprakstīts augstāk redzamais piemērs:

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`

Ierobežojumi

- Laika ierobežojums: 20 sekundes
- Atmiņas ierobežojums: 256 MiB
- $2 \le R \le 5,000$
- $1 \le C \le 200$
- Ne vairāk par 500 maiņām (procedūru `changeH()` vai `changeV()` izsaukumi)
- Ne vairāk par 200,000 funkcijas `escape()` izsaukumu
- Ne vairāk kā 1,000 vombatu katrā segmentā jebkurā brīdī

Apakšuzdevumi

Apakšuzdevums	Punkti	Papildus ievaddatu ierobežojumi
1	9	$C = 1$
2	12	$R, C \le 20$, un nebūs procedūru `changeH()` vai `changeV()` izsaukumu
3	16	$R, C \le 100$, un būs ne vairāk kā 100 funkcijas `escape()` izsaukumu
4	18	$C = 2$
5	21	$C \le 100$
6	24	*(Nav papildus ierobežojumu)*

Eksperimentēšana

Piemēra testētājs uz jūsu datora nolasīs ievaddatus no faila `wombats.in`, kam ir jābūt šādā formātā:

- 1. rinda: `R C`
- 2. rinda: `H[0][0] … H[0][C-2]`
- …
- `(R + 1)`. rinda: `H[R-1][0] … H[R-1][C-2]`
- `(R + 2)`. rinda: `V[0][0] … V[0][C-1]`
- …
- `(2R)`. rinda: `V[R-2][0] … V[R-2][C-1]`
- nākamā rinda: `E`
- nākamas `E` rindas: viens notikums rindā, notikumu notikšanas secībā

Ja `C = 1`, tad tukšas rindas, kas apzīmē vombatu skaitu uz horizontāliem ceļiem (no `2`. līdz `R + 1`. rindām), nav nepieciešamas.

Katram notikumam rindai ir jābūt vienā no šādiem formātiem:

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

Augstāk esošais piemērs jāapraksta šādā veidā:

```
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
```

Piezīmes valodām

C/C++	Jums jāiekļauj `#include "wombats.h"`.
Pascal	Jums jādefinē `unit Wombats`. Visi masīvi tiek numurēti sākot no `0` (nevis `1`).

Iepazīstieties ar risinājumu piemēru šabloniem uz jūsu datora.