International Olympiad in Informatics 2013

July 2013 6-13

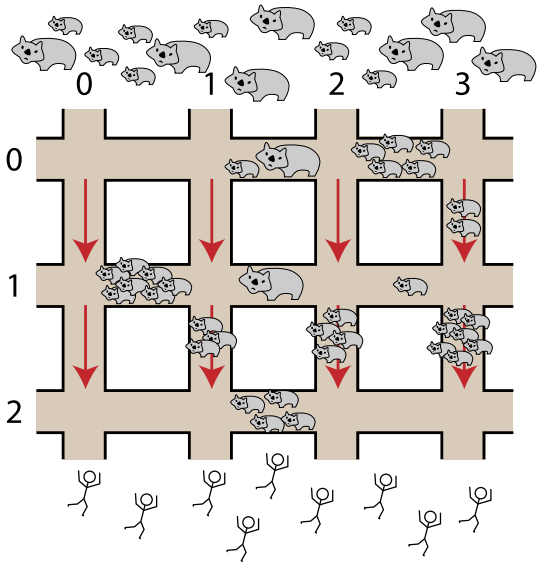
Brisbane, Australia

وامبت‌ها

Persian — ۱.۰

شهر بریزبین توسط وامبت‌های (خرس‌های استرالیایی) جهش‌یافته و غول‌پیکر اشغال شده است. شما باید ساکنین شهر را به منطقه‌ی امن برسانید.

جاده‌ها در بریزبین به شکل یک توری (grid) هستند، یعنی تمام جاده‌ها به صورت افقی یا عمودی قرار گرفته‌اند. در این شهر R جاده‌ی افقی از شرق به غرب وجود دارد که از بالا به پایین با شماره‌های 0...R-1 شماره‌گذاری شده‌اند. به همین ترتیب C جاده‌ی عمودی از شمال به جنوب وجود دارند که از چپ به راست با اعداد 0...C-1 شماره‌گذاری شده‌اند. شیوه‌ی شماره‌گذاری جاده‌ها در شکل زیر نشان داده شده است:



وامبت‌ها از سمت شمال به شهر حمله کرده‌اند و شهروندان به سمت جنوب فرار می‌کنند. شهروندان می‌توانند درون جاده‌های افقی، به هر دو سمت حرکت کنند، اما در جاده‌های عمودی، فقط به سمت جنوب و منطقه‌ی امن حرکت می‌کنند.

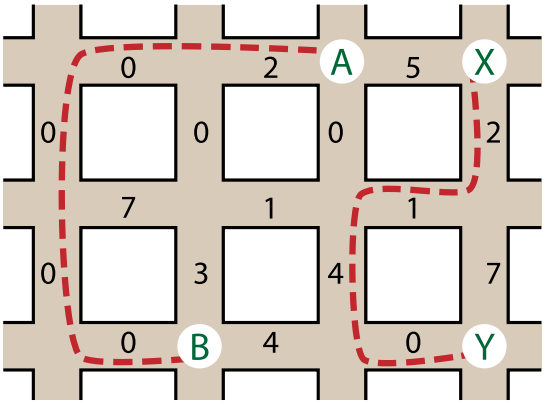
محل تقاطع جاده‌ی افقی P با جاده‌ی عمودی Q را با (P, Q) نشان می‌دهیم. در هر قسمت از یک جاده بین دو تقاطع، تعدادی وامبت قرار دارند که این تعداد می‌تواند در طول زمان عوض شود. شما باید به هر شهروند کمک کنید تا با حرکت در جاده‌ها، از یک تقاطع در شمال (روی جاده افقی 0) به یک تقاطع داده شده در جنوب (روی جاده‌ی افقی R-1) برود، طوری که در طول مسیر، به کم‌ترین تعداد وامبت برخورد کند.

برای شروع، اندازه توری و تعداد وامبت‌ها در هر جاده به شما داده می‌شود. در ادامه E رویداد به شما گزارش می‌شود که هر رویداد به یکی از دو صورت زیر است:

- *change*: تعداد وامبت‌ها در یک قسمت از یک جاده بین دو تقاطع تغییر می‌کند.
- *escape*: یک شهروند وارد یک تقاطع روی جاده افقی 0 می‌شود و شما باید برای او، یک مسیر پیدا کنید که به یک تقاطع مشخص روی جاده‌ی افقی R-1 برسد و شهروند در طول مسیر به کم‌ترین تعداد وامبت ممکن برخورد کند.

شما باید برای این رویدادها، توابع `init()`، `changeH()`، `changeV()` و `escape()` را به شکلی که در ادامه توضیح داده می‌شود، پیاده‌سازی کنید.

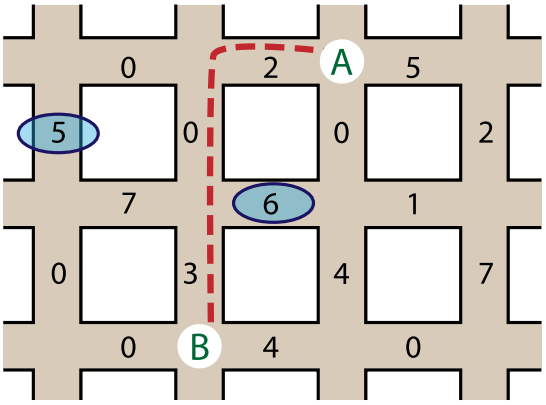
مثال‌ها



شکل بالا یک جدول اولیه با `R=۳` جاده‌ی افقی و `C=۴` جاده‌ی عمودی را نشان می‌دهد. تعداد وامبت‌های درون هر جاده، روی آن جاده نشان داده شده است. دنباله‌ی رویدادهای زیر را در نظر بگیرید:

- یک شهروند به تقاطع `A=(۰, ۲)` می‌رسد و می‌خواهد به تقاطع `B=(۲, ۱)` فرار کند. کم‌ترین تعداد وامبتی که او می‌تواند در مسیر ببیند برابر با `۲` است که با خط‌چین نشان داده شده است.
- یک شهروند دیگر به تقاطع `X=(۰, ۳)` می‌رسد و می‌خواهد به تقاطع `Y=(۲, ۳)` فرار کند. کم‌ترین تعداد وامبتی که او می‌تواند در مسیر ببیند برابر با `۷` است و مجدداً با خط‌چین نشان داده شده است.

- دو رویداد تغییر رخ می‌دهند: ابتدا در قسمت بالایی جاده‌ی عمودی `۰`، تعداد وامبت‌ها به `۵` تغییر می‌یابد. سپس در قسمت میانی جاده‌ی افقی `۱`، تعداد وامبت‌ها برابر `۶` می‌شود. به اعداد درون دایره در شکل زیر توجه کنید.



- شهروند سوم به تقاطع `A=(۰, ۲)` می‌رسد و می‌خواهد به تقاطع `B=(۲, ۱)` فرار کند. حال کم‌ترین تعداد وامبتی که او می‌تواند ببیند برابر `۵` است. مسیر جدید در شکل بالا با خط‌چین نشان داده شده است.

پیاده‌سازی

شما باید یک فایل که در آن توابع `init()`، `changeH()`، `changeV()` و `escape()` پیاده‌سازی شده‌اند را به سامانه‌ی داوری ارسال کنید.

تابع شما: `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);
```

توضیحات

این تابع، چیدمان اولیه‌ی نقشه را به شما می‌دهد و به شما اجازه می‌دهد تا متغیرهای سراسری (global variables) و داده‌ساختارهای خود را مقداردهی کنید. این تابع تنها یک بار و قبل از توابع `changeH()`، `changeV()` یا `escape()` صدا زده می‌شود.

پارامترها

- `R`: تعداد جاده‌های افقی.
- `C`: تعداد جاده‌های عمودی.
- `H`: یک آرایه‌ی دو بعدی با ابعاد `R × (C - 1)`، طوری که `H[P][Q]` تعداد وامبت‌ها روی قسمت افقی جاده بین تقاطع‌های `(P, Q)` و `(P, Q + 1)` را نشان می‌دهد.
- `V`: یک آرایه‌ی دو بعدی با ابعاد `(R - 1) × C`، طوری که `V[P][Q]` تعداد وامبت‌ها روی قسمت عمودی جاده بین تقاطع‌های `(P, Q)` و `(P + 1, Q)` را نشان می‌دهد.

تابع شما: `changeH()`

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

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

توضیحات

این تابع زمانی صدا زده می‌شود که تعداد وامبت‌ها در قسمت افقی جاده بین تقاطع‌های `(P, Q)` و `(P, Q + 1)` تغییر می‌کند.

پارامترها

- `P`: نشان می‌دهد کدام جاده‌ی افقی تغییر کرده است (`0 ≤ P ≤ R - 1`).

- `Q` : نشان می‌دهد قسمت تغییرکرده، بین کدام جاده‌های عمودی قرار دارد (`0 ≤ Q ≤ C - 2`).
- `W` : تعداد جدید وامبت‌ها در این قسمت از جاده (`0 ≤ W ≤ 1,000`).

تابع شما: `changeV()`

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

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

توضیحات

این تابع زمانی صدا زده می‌شود که تعداد وامبت‌ها در قسمت عمودی جاده بین تقاطع‌های `(P, Q)` و `(P + 1, Q)` تغییر می‌کند.

پارامترها

- `P` : نشان می‌دهد قسمت تغییر کرده، بین کدام جاده‌های افقی قرار دارد (`0 ≤ P ≤ R - 2`).
- `Q` : نشان می‌دهد کدام جاده عمودی تغییر کرده است (`0 ≤ Q ≤ C - 1`).
- `W` : تعداد جدید وامبت‌ها در این قسمت از جاده (`0 ≤ W ≤ 1,000`).

تابع شما: `escape()`

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

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

توضیحات

این تابع کم‌ترین تعداد وامبت ممکن در یک مسیر از تقاطع `(0, V1)` به تقاطع `(R-1, V2)` را برای یک شهروند محاسبه می‌کند.

پارامترها

- `V1` : نشان می‌دهد که شهروند از چه محلی در جاده‌ی افقی `0` شروع می‌کند (`0 ≤ V1 ≤ C-1`).
- `V2` : نشان می‌دهد که مسیر شهروند در چه محلی در جاده‌ی افقی `R-1` تمام می‌شود (`0 ≤ V2 ≤ C-1`).
- خروجی تابع: کم‌ترین تعداد وامبتی که شهروند باید ببیند.

اجرای نمونه

اجرای زیر مثال بالا را توصیف می‌کند:

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`

محدودیت‌ها

- محدودیت زمان: ۲۰ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- $2 \le R \le 5,000$
- $1 \le C \le 200$
- حد اکثر `500` تغییر (فراخوانی توابع `changeH()` یا `changeV()`) انجام می‌شود.
- حداکثر `200,000` فراخوانی تابع `escape()` انجام می‌شود.
- در هر لحظه حداکثر `1,000` وامبت در هر قسمت از هر جاده قرار دارند.

زیرمسئله‌ها

زیرمسئله	نمره	محدودیت‌های اضافی ورودی
۱	۹	$C = 1$
۲	۱۲	$R,C \le 20$, و هیچ فراخوانی `changeH()` یا `changeV()` انجام نمی‌شود.
۳	۱۶	$R,C \le 100$, و حداکثر ۱۰۰ فراخوانی `escape()` انجام می‌شود.
۴	۱۸	$C = 2$
۵	۲۱	$C \le 100$
۶	۲۴	(بدون محدودیت اضافی)

آزمایش

مصحح نمونه روی کامپیوتر شما، ورودی را از فایل `wombats.in` می‌خواند که باید به شکل زیر باشد:

- خط ۱: `R C`
- خط ۲: `H[0][0] … H[0][C-2]`
- ...
- خط `R + 1`: `H[R-1][0] … H[R-1][C-2]`
- خط `R + 2`: `V[0][0] … V[0][C-1]`
- ...
- خط `2R`: `V[R-2][0] … V[R-2][C-1]`
- خط بعدی: `E`
- در `E` خط بعد: در هر خط، یک رویداد، به ترتیب وقوع آن‌ها به شما داده می‌شود.

اگر `C = 1` باشد، وجود خط‌های خالی برای نشان دادن تعداد وامبت‌ها در جاده‌های افقی (خط‌های `2` تا `R + 1`) الزامی نیست.

خط مربوط به هر رویداد، باید به یکی از صورت‌های زیر باشد:

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

به طور نمونه، مثال بالا باید به شکل زیر در ورودی داده شود:

```
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
```

نکات زبان

`C/C++` عبارت `#include "wombats.h"` را باید به برنامه اضافه کنید.

Pascal شما باید `unit Wombats` را تعریف کنید. تمام آرایه‌ها از `0` (و نه `1`) شروع می‌شوند.

برای دیدن مثال‌ها به راه‌حل‌های نمونه (برروی کامپیوتر خود) مراجعه کنید.