International Olympiad in Informatics 2013

July 2013 6-13

Brisbane, Australia

رؤیـا

Persian — ۱.۰

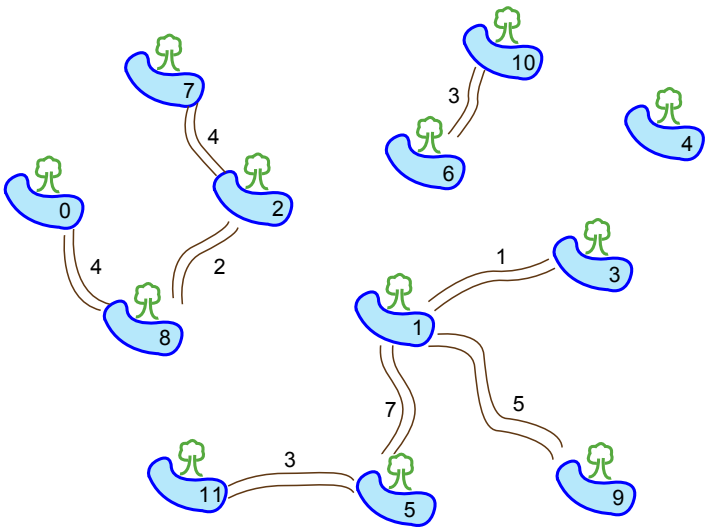
این داستان در سال‌های خیلی دور رخ می‌دهد؛ زمانی که دنیا تازه به وجود آمده بود و کسی IOI را در رؤیا هم نمی‌دید.

یک مار در سرزمینی زندگی می‌کند که N برکه دارد. برکه‌ها با اعداد 0 تا $N - 1$ شماره‌گذاری شده‌اند. M جاده‌ی دوطرفه این برکه‌ها را به هم متصل می‌کند و مار می‌تواند روی این جاده‌ها حرکت کند. هر زوج برکه با حداکثر یک دنباله از جاده‌ها (به صورت مستقیم یا غیر مستقیم) به هم متصل شده‌اند، هرچند برخی زوج برکه‌ها می‌توانند اصلاً به هم متصل نباشند (بنابراین $M \leq N-1$). عبور از هر جاده تعداد مشخصی روز برای مار طول می‌کشد. این تعداد می‌تواند برای هر جاده متفاوت باشد.

کانگورو، دوست مار، می‌خواهد $N - M - 1$ جاده‌ی جدید بسازد طوری که مار بتواند بین هر زوج برکه حرکت کند. کانگورو می‌تواند بین هر دو برکه‌ی دل‌خواه جاده بسازد و پیمایش هر جاده‌ی جدیدی که کانگورو می‌سازد L روز برای مار طول می‌کشد.

علاوه بر این، کانگورو می‌خواهد که حرکت مار را تا حد ممکن سریع کند. بنابراین او جاده‌های جدید را طوری می‌سازد که بیش‌ترین زمان حرکت بین هر دو برکه کم‌ترین مقدار ممکن را داشته باشد. شما باید به کانگورو کمک کنید که جاده‌ها را به طریق گفته‌شده بسازد و بعد از ساخته‌شدن جاده‌ها، بیش‌ترین زمان حرکت بین برکه‌ها را بیابد.

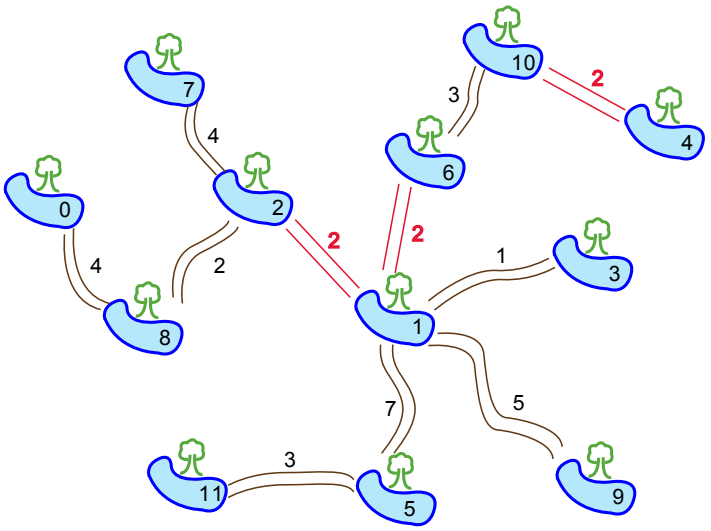
مثال‌ها



در شکل بالا $N = 12$ برکه و $M = 8$ جاده وجود دارد. فرض کنید عبور از هر جاده‌ی جدید $L = 2$ روز برای مار طول می‌کشد. حال کانگورو می‌تواند سه جاده‌ی جدید به صورت زیر بسازد:

- بین برکه‌های ۱ و ۲

- بین برکه‌های ۱ و ۶
- بین برکه‌های ۴ و ۱۰



شکل بالا مجموعه‌ی نهایی جاده‌ها را نشان می‌دهد. بیش‌ترین زمان حرکت در این شکل ۱۸ روز است که بین برکه‌های ۰ و ۱۱ رخ می‌دهد. این کم‌ترین جواب ممکن است: مستقل از این که کانگورو چه جاده‌هایی را بسازد، دو برکه وجود خواهند داشت که حرکت بین آن‌ها برای مار حداقل ۱۸ روز طول می‌کشد.

پیاده‌سازی

شما باید تابع `travelTime()` را به شکل زیر در یک فایل پیاده‌سازی و ارسال کنید.

تابع شما: `travelTime()`

C/C++

```
int travelTime(int N, int M, int L,
        int A[], int B[], int T[]);
```

Pascal

```
function travelTime(N, M, L : LongInt;
        var A, B, T : array of LongInt) : LongInt;
```

توضیحات

این تابع باید بیش‌ترین زمان حرکت (به روز) بین هر زوج برکه را محاسبه کند، با این فرض که کانگورو `N - M - 1` جاده را طوری اضافه کرده است که همه‌ی برکه‌ها به هم متصل شوند و بیش‌ترین زمان حرکت بین برکه‌ها کم‌ترین مقدار ممکن را داشته باشد.

پارامترها

- `N` : تعداد برکه‌ها.
- `M` : تعداد جاده‌های اولیه که از قبل وجود دارند.

- `L` : تعداد روزهایی که طول می‌کشد تا مار یک جاده‌ی جدید را بپیماید.
- `A` ،`B` و `T` : آرایه‌هایی به طول `M` که نقاط پایانی و زمان پیمایش جاده‌های موجود را مشخص می‌کند، طوری که `i`اُمین جاده برکه‌های `A[i-1]` و `B[i-1]` را به هم متصل می‌کند و زمان پیمایش آن در هر یک از دو جهت `T[i-1]` روز است.
- خروجی: بیش‌ترین زمان حرکت بین هر زوج برکه، همان طور که در بالا توضیح داده شد.

اجرای نمونه

اجرای زیر مثال بالا را توصیف می‌کند:

Parameter	Value
N	`12`
M	`8`
L	`2`
A	`[0, 8, 2, 5, 5, 1, 1, 10]`
B	`[8, 2, 7, 11, 1, 3, 9, 6]`
T	`[4, 2, 4, 3, 7, 1, 5, 3]`
Returns	`18`

محدودیت‌ها

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۶۴ مگابایت
- `1 ≤ N ≤ 100,000`
- `0 ≤ M ≤ N - 1`
- `0 ≤ A[i], B[i] ≤ N - 1`
- `1 ≤ T[i] ≤ 10,000`
- `1 ≤ L ≤ 10,000`

زیرمسئله‌ها

زیرمسئله	امتیاز	محدودیت‌های ورودی اضافی
۱	۱۴	`M = N - 2` و به هر برکه دقیقاً یا یک یا دو جاده‌ی اولیه منتهی می‌شود. به عبارت دیگر، دو مجموعه از برکه‌های همبند وجود دارند و در هر مجموعه جاده‌ها یک مسیر بدون انشعاب را تشکیل می‌دهند.
۲	۱۰	`N ≤ 100` و `M = N - 2`
۳	۲۳	`M = N - 2`
۴	۱۸	به هر برکه حداکثر یک جاده‌ی اولیه منتهی می‌شود.
۵	۱۲	`N ≤ 3,000`
۶	۲۳	(بدون محدودیت اضافی)

آزمایش

مصحح نمونه روی کامپیوتر شما ورودی را از فایل `dreaming.in` می‌خواند که محتوای آن به شکل زیر است:

- خط ۱: `N M L`
- خطوط ۲ تا `M + 1`: `A[i] B[i] T[i]`

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

```
12 8 2
0 8 4
8 2 2
2 7 4
5 11 3
5 1 7
1 3 1
1 9 5
10 6 3
```

نکات زبان

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

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

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

International Olympiad in Informatics 2013

July 2013 6-13
Brisbane, Australia

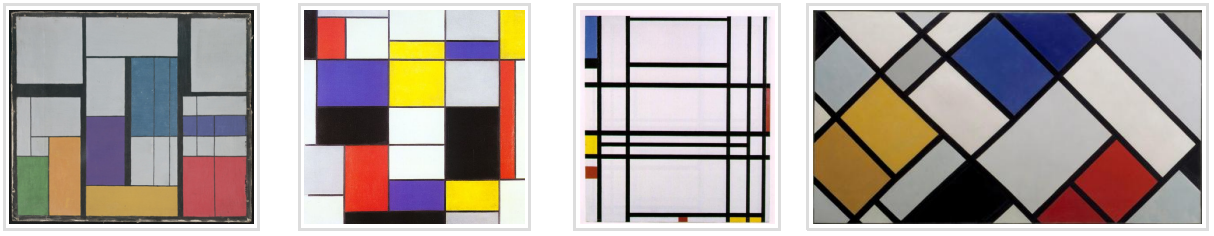
کلاس هنر

Persian — ۱.۱

شما به زمان آزمون تاریخ هنر نزدیک می‌شوید، اما بیش از آن که در مدرسه به کلاس هنر توجه کرده باشید، بیش‌تر ذهن‌تان را درگیر کلاس کامپیوتر کرده‌اید. حالا باید برنامه‌ای بنویسید که به جای شما آزمون دهد.

آزمون شامل تعدادی نقاشی است. هر نقاشی نمونه‌ای از چهار سبک متمایز است که با شماره‌های ۱، ۲، ۳ و ۴ مشخص می‌شوند.

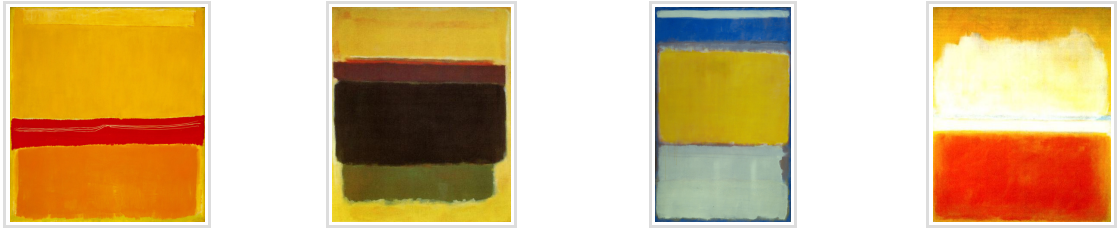
سبک ۱ شامل هنر مدرن نئوپلاستیک است که در زیر نمونه‌هایی از آن را مشاهده می‌کنید:



سبک ۲ شامل مناظر امپرسیونیست است که در زیر نمونه‌هایی از آن را مشاهده می‌کنید:

سبک ۳ شامل نقاشی‌های اکسپرسیونیست است که در زیر نمونه‌هایی از آن را مشاهده می‌کنید:

سبک ۴ شامل نقاشی‌های رنگ زمینه است که نمونه‌هایی از آن را در زیر مشاهده می‌کنید:



وظیفه‌ی شما این است که به ازای یک نقاشی دیجیتال داده‌شده، سبک آن را تعیین کنید.

جاج IOI تصاویر زیادی را از هر سبک جمع‌آوری کرده است. ۹ تصویر تصادفی از هر سبک انتخاب و برای استفاده‌ی دستی و تست کردن برنامه در کامپیوتر شما قرار داده شده است. بقیه‌ی تصاویر به منظور نمره‌دهی به برنامه‌ی شما داده می‌شود.

تصویر به صورت یک آرایه‌ی H×W از پیکسل‌ها داده می‌شود. سطرها از بالا به پایین با اعداد 0 تا H - 1 و ستون‌ها از چپ به راست با با اعداد 0 تا W - 1 شماره‌گذاری شده‌اند.

پیکسل‌ها با استفاده از آرایه‌های دو بعدی R ، G و B توصیف می‌شوند که به ترتیب میزان قرمز، سبز و آبی بودن هر پیکسل را مشخص می‌کنند. محدوده‌ی این مقادیر از 0 (عدم وجود قرمز، سبز یا آبی) تا 255 (بیشینه‌ی میزان قرمز، سبز یا آبی بودن) می‌باشد.

پیاده‌سازی

شما باید تابع style() به شکل زیر را در یک فایل پیاده‌سازی و ارسال کنید.

تابع شما: style()

C/C++

```
int style(int H, int W,
        int R[500][500], int G[500][500], int B[500][500]);
```

Pascal

```
type artArrayType = array[0..499, 0..499] of longint;

function style(H, W : LongInt;
        var R, G, B : artArrayType) : LongInt;
```

توضیحات

این تابع باید سبک تصویر را مشخص کند.

پارامترها

- H : تعداد سطرهای پیکسل‌ها در تصویر
- W : تعداد ستون‌های پیکسل‌ها در تصویر
- R : یک آرایه‌ی دو بعدی با اندازه‌ی H×W که میزان رنگ قرمز در هر پیکسل را مشخص می‌کند.
- G : یک آرایه‌ی دو بعدی با اندازه‌ی H×W که میزان رنگ سبز در هر پیکسل را مشخص می‌کند.
- B : یک آرایه‌ی دو بعدی با اندازه‌ی H×W که میزان رنگ آبی در هر پیکسل را مشخص می‌کند.
- خروجی: سبک تصویر که باید یکی از مقادیر 1 ، 2 ، 3 و 4 ، همان‌گونه که در بالا توضیح داده شد باشد.

توجه کنید که R[i][j] ، G[i][j] و B[i][j] به پیکسلی که در سطر i و ستون j قرار دارد اشاره می‌کند و یک عدد صحیح بین 0 و 255 خواهد بود.

محدودیت‌ها

- محدودیت زمان: ۵ ثانیه
- محدودیت حافظه: ۶۴ مگابایت
- $100 \le H \le 500$
- $100 \le W \le 500$

امتیازدهی

زیرمسئله‌ای وجود ندارد. به جای زیرمسئله، امتیاز شما براساس تعداد تصاویری که به درستی دسته‌بندی کرده باشید محاسبه خواهد شد.

فرض کنید شما P درصد از تصاویر را به درستی دسته‌بندی کرده باشید ($0 \le P \le 100$):

- اگر $P < 25$ ، امتیاز شما صفر خواهد بود.
- اگر $25 \le P < 50$ ، امتیاز شما بین صفر و 10 براساس مقیاس خطی محاسبه خواهد شد. به طور مشخص امتیاز شما بزرگ‌ترین عدد صحیح نابزرگ‌تر از $10 \times (P - 25) / 25$ خواهد بود.
- اگر $50 \le P < 90$ ، امتیاز شما بین 10 و 100 براساس مقیاس خطی محاسبه خواهد شد. به طور مشخص امتیاز شما بزرگ‌ترین عدد صحیح نابزرگ‌تر از $10 + (90 \times (P - 50) / 40)$ خواهد بود.
- اگر $P \ge 90$ ، امتیاز شما 100 خواهد بود.

آزمایش

مصحح نمونه روی کامپیوتر شما، فایل ورودی را از فایل `artclass.jpg` می‌خواند. این فایل باید شامل یک تصویر به فرمت JPEG باشد.

شما مجاز به استفاده از هر ابزار گرافیکی موجود برای مطالعه‌ی تصاویر هستید، اما استفاده از این ابزارها برای حل مسئله الزامی نیست. (منوی "Applications > Graphics" را ببینید.)

نکات زبان

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

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

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

International Olympiad in Informatics 2013

July 2013 6-13

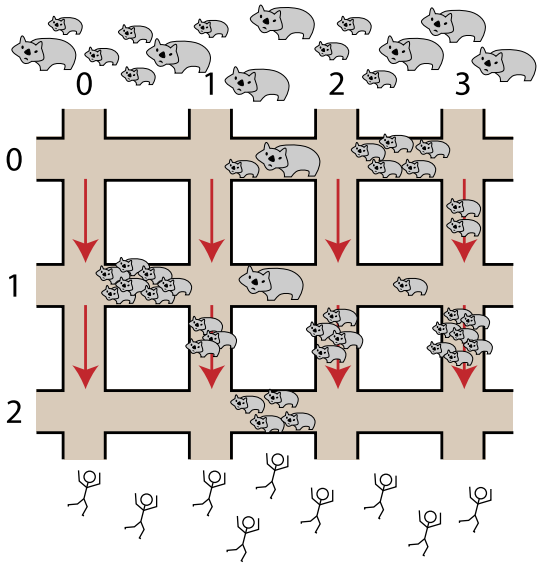
Brisbane, Australia

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.

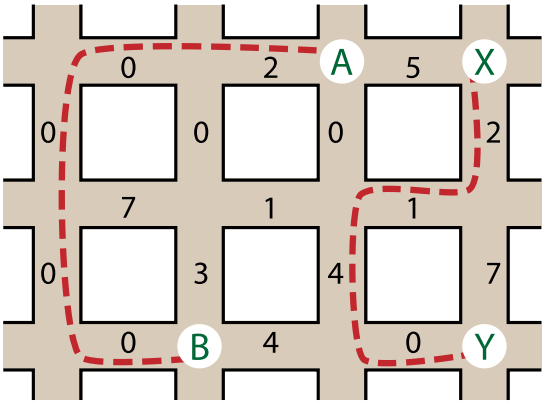
محل تقاطع جاده‌ی افقی P با جاده‌ی عمودی Q را با (P, Q) نشان می‌دهیم. در هر قسمت از یک جاده بین دو تقاطع، تعدادی وامبت قرار دارند که این تعداد می‌تواند در طول زمان عوض شود. شما باید به هر شهروند کمک کنید تا با حرکت در جاده‌ها، از یک تقاطع در شمال (روی جاده افقی 0) به یک تقاطع داده شده در جنوب (روی جاده‌ی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

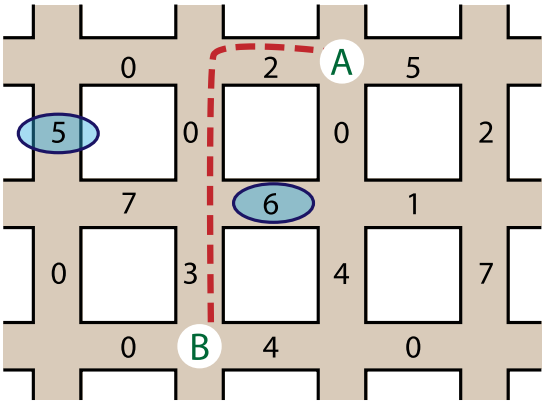
مثال‌ها



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.