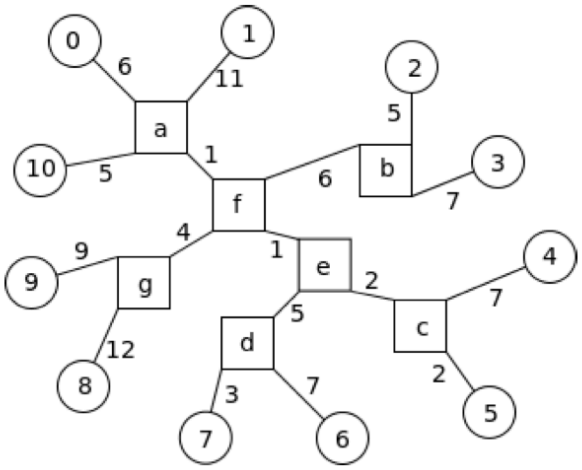
شهرها

در قزاقستان N شهر کوچک وجود دارد که از ۰ تا $N-1$ شماره‌گذاری شده‌اند. همچنین تعداد نامشخصی کلان‌شهر نیز وجود دارند. ما به شهرهای کوچک و کلان‌شهرها به طور کلی «شهر» اطلاق می‌کنیم.

تمام شهرها در قزاقستان توسط شبکه‌ای از بزرگراه‌های دوطرفه به هم متصل شده‌اند. هر بزرگراه دو شهر متمایز را به هم متصل می‌کند و هر دو شهری توسط حداکثر یک بزرگراه به یکدیگر متصل‌اند. برای هر دو شهر a و b، یک مسیر یکتا برای رسیدن از a به b وجود دارد، به شرط آن‌که از هیچ بزرگراهی در این مسیر دو بار استفاده نشود.

می‌دانیم که هر شهر کوچک به دقیقا یک شهر دیگر متصل است. همچنین هر کلان‌شهر به حداقل سه شهر متصل است.

شکل زیر شبکه‌ای شامل ۱۱ شهر کوچک و ۷ کلان‌شهر را نشان می‌دهد. شهرهای کوچک به صورت دایره با برچسب‌های عددی، و کلان‌شهرها به صورت مربع با برچسب‌های حرفی نشان داده شده‌اند.



طول هر بزرگراه یک عدد صحیح مثبت است. فاصله‌ی بین هر دو شهر برابر است با مجموع طول بزرگراه‌ها روی کوتاه‌ترین مسیری که آن دو شهر را به هم متصل می‌کند.

برای هر کلان‌شهر C، مقدار $r(C)$ را برابر با فاصله‌ی دورترین شهر کوچک از C تعریف می‌کنیم. یک کلان‌شهر C را «مرکز» می‌نامیم اگر $r(C)$ بین تمام کلان‌شهرها کم‌ترین باشد. فاصله‌ی یک مرکز تا دورترین شهر کوچک از آن را با R نشان می‌دهیم. بنابراین R کوچک‌ترین مقدار از بین تمام مقادیر $r(C)$ است.

در مثال بالا دورترین شهر کوچک از کلان‌شهر a شهر شماره‌ی ۸ است، و فاصله‌ی بین این دو شهر برابر است با $r(a) = ۱ + ۴ + ۱۲ = ۱۷$. برای کلان‌شهر g نیز $r(g) = ۱۷$ است (یکی از شهرهای کوچکی که از g دورترین است، شهر شماره‌ی ۶ است.) در مثال بالا تنها مرکز، کلان‌شهر f با مقدار $r(f) = ۱۶$ است. بنابراین در مثال بالا مقدار R برابر ۱۶ است.

با حذف یک مرکز، شبکه به چند مؤلفه‌ی همبند تقسیم می‌شود. یک مرکز را «متوازن» می‌نامیم، اگر هر یک از این مؤلفه‌ها حداکثر شامل $\lfloor \frac{N}{۲} \rfloor$ شهر کوچک باشد. (دقت کنید که ما در این تعریف فقط شهرهای کوچک را می‌شمریم.)

در مثال قبل، کلان‌شهر f یک مرکز است. اگر f را حذف کنیم، شبکه به چهار مؤلفه‌ی همبند افراز می‌شود. این چهار مؤلفه شامل مجموعه شهرهای کوچک زیر خواهند بود: $\{۰, ۱, ۱۰\}$، $\{۲, ۳\}$، $\{۴, ۵, ۶, ۷\}$، و $\{۸, ۹\}$. هیچ یک از این مؤلفه‌ها بیش‌تر از $۵ = \lfloor \frac{۱۱}{۲} \rfloor$ شهر کوچک ندارد، بنابراین f یک مرکز متوازن است.

مسئله

تنها اطلاعاتی که در آغاز در مورد شبکه‌ی شهرها و بزرگراه‌ها دارید، تعداد شهرهای کوچک یعنی N است. شما اطلاعی از تعداد کلان‌شهرها و همچنین چینش بزرگراه‌های کشور ندارید. شما تنها می‌توانید با پرس‌وجو درباره‌ی فاصله‌ی بین جفت شهرهای کوچک در مورد شبکه اطلاعات جدید به دست بیاورید.

شما قرار است موارد زیر را انجام دهید:

- در تمام زیرمسئله‌ها: تعیین مقدار R
- در زیرمسئله‌های ۳ تا ۶: تعیین این که آیا شبکه دارای یک مرکز متوازن است.

شما باید تابع `hubDistance` را پیاده‌سازی کنید. ارزیاب در هر بار اجرا تعدادی مورد آزمون را ارزیابی می‌کند. تعداد موارد آزمون در هر اجرا حداکثر ۴۰ است. برای هر مورد آزمون، ارزیاب تابع `hubDistance` را دقیقا یک‌بار فراخوانی می‌کند. مطمئن شوید که تابع شما متغیرهای موردنیاز را در هر بار فراخوانی مقداردهی اولیه می‌کند.

- `hubDistance(N, sub)`
  - `N`: تعداد شهرهای کوچک.
  - `sub`: شماره‌ی زیرمسئله (در قسمت زیرمسئله‌ها توضیح داده شده است).
  - اگر `sub` برابر ۱ یا ۲ باشد، تابع می‌تواند مقدار R یا $-R$ را برگرداند.
  - اگر `sub` بزرگ‌تر از ۲ باشد، اگر یک مرکز متوازن وجود داشت، تابع باید مقدار R و در غیر این صورت مقدار $-R$ را برگرداند.

شما می‌توانید درون تابع `hubDistance` با فراخوانی تابع ارزیاب `getDistance(i, j)` در مورد شبکه‌ی بزرگراه‌ها اطلاعات کسب کنید. این تابع فاصله‌ی بین دو شهر کوچک i و j را برمی‌گرداند. اگر i و j برابر باشند، تابع مقدار ۰ را برمی‌گرداند. همچنین وقتی که آرگومان‌ها غیرمعتبر باشند، تابع مقدار ۰ برمی‌گرداند.

زیرمسئله‌ها

در هر مورد آزمون:

- N عدد صحیحی بین ۶ و ۱۱۰ است.
- فاصله‌ی بین هر دو شهر کوچک متمایز بین ۱ و ۱,۰۰۰,۰۰۰ است.

تعداد پرس‌وجوهایی که برنامه‌ی شما می‌تواند انجام دهد محدود است. این محدودیت بسته به زیرمسئله متفاوت است و برای هر زیرمسئله در جدول زیر مشخص شده است. اگر برنامه‌ی شما از محدودیت تعداد پرس‌وجوها تجاوز کند، برنامه خاتمه یافته و فرض می‌شود که برنامه جواب غلط داده است.

زیرمسئله	امتیاز	تعداد پرس‌وجو	یافتن مرکز متوازن	محدودیت‌های دیگر
۱	۱۳	$\frac{N(N-۱)}{۲}$	خیر	ندارد
۲	۱۲	$\lceil \frac{۷N}{۲} \rceil$	خیر	ندارد
۳	۱۳	$\frac{N(N-۱)}{۲}$	بله	ندارد
۴	۱۰	$\lceil \frac{۷N}{۲} \rceil$	بله	هر کلان‌شهر دقیقا به سه شهر دیگر متصل است.
۵	۱۳	$۵N$	بله	ندارد
۶	۳۹	$\lceil \frac{۷N}{۲} \rceil$	بله	ندارد

ارزیاب نمونه

توجه کنید که شماره‌ی زیرمسئله بخشی از ورودی است. ارزیاب نمونه رفتارش را برحسب شماره‌ی زیرمسئله تغییر می‌دهد. ارزیاب نمونه ورودی را از فایل `towns.in` با قالب زیر می‌خواند:

- خط ۱: شماره‌ی زیرمسئله و تعداد موارد آزمون
- خط ۲: $N_۱$، تعداد شهرهای کوچک در اولین مورد آزمون.
- $N_۱$ خط بعد: عدد j اُم ($۱ \leq j \leq N_۱$) در خط i اُم از این خط‌ها ($۱ \leq i \leq N_۱$) فاصله بین شهرهای کوچکِ $i-۱$ و $j-۱$ است.
- موارد آزمون بعدی در ادامه می‌آیند. آن‌ها هم به همان قالب اولین مورد آزمون داده می‌شوند.

ارزیاب نمونه برای هر مورد آزمون، مقدار خروجی تابع `hubDistance` و تعداد فراخوانی‌های انجام‌شده را در سطرهای جدا چاپ می‌کند .

فایل ورودی مربوط به مثال بالا به صورت زیر است:

```
1 1
11
0 17 18 20 17 12 20 16 23 20 11
17 0 23 25 22 17 25 21 28 25 16
18 23 0 12 21 16 24 20 27 24 17
20 25 12 0 23 18 26 22 29 26 19
17 22 21 23 0 9 21 17 26 23 16
12 17 16 18 9 0 16 12 21 18 11
20 25 24 26 21 16 0 10 29 26 19
16 21 20 22 17 12 10 0 25 22 15
23 28 27 29 26 21 29 25 0 21 22
20 25 24 26 23 18 26 22 21 0 19
11 16 17 19 16 11 19 15 22 19 0
```

این قالب با مشخص کردن لیست بزرگراه‌ها کاملاً متفاوت است. مسلماً شما می‌توانید ارزیاب نمونه را ویرایش کنید تا از قالب ورودی دیگری استفاده کند.

مرتب‌سازی

«آیژان» یک دنباله شامل N عدد صحیح $S[۰]$، $S[۱]$، ...، $S[N-۱]$ در اختیار دارد. این دنباله شامل اعداد متمایز از ۰ تا $N-۱$ است. او می‌خواهد این دنباله را با استفاده از جابه‌جایی جفت عناصر دنباله (swap) به ترتیب صعودی مرتب کند. دوستش «اِرمِک» نیز قصد دارد برخی از جفت عناصر دنباله را (نه لزوما در راستای کمک به آیژان) جابه‌جا کند.

ارمک و آیژان قصد دارند دنباله را در چندین مرحله تغییر دهند. در هر مرحله، ابتدا ارمک یک جابه‌جایی انجام می‌دهد و سپس آیژان یک جابه‌جایی دیگر انجام می‌دهد. به طور دقیق‌تر، شخصی که جابه‌جایی را انجام می‌دهد، دو اندیس معتبر انتخاب می‌کند و عناصر قرار گرفته در آن دو اندیس را جابه‌جا می‌کند. توجه کنید که این دو اندیس لزوما متمایز نیستند. در صورت برابری اندیس‌ها، یک عنصر با خودش جابه‌جا می‌شود و این کار دنباله را تغییر نمی‌دهد.

آیژان می‌داند که مرتب‌سازی دنباله‌ی S برای ارمک مهم نیست. او همچنین اندیس‌های انتخابی ارمک را از قبل می‌داند. ارمک قصد دارد در M مرحله شرکت کند. این مراحل را از ۰ تا $M-۱$ شماره‌گذاری می‌کنیم. به ازای هر i بین ۰ تا $M-۱$ (شامل هر دو)، ارمک اندیس‌های $X[i]$ و $Y[i]$ را در مرحله‌ی i انتخاب خواهد کرد.

آیژان قصد دارد که دنباله‌ی S را مرتب کند. قبل از هر مرحله، اگر آیژان متوجه شود که دنباله به صورت صعودی مرتب شده است، او به فرآیند مرتب‌سازی خاتمه خواهد داد. با فرض این‌که دنباله‌ی S و اندیس‌هایی که ارمک قصد انتخابشان را دارد به شما داده می‌شود، وظیفه‌ی شما پیدا کردن دنباله‌ای از جابه‌جایی‌ها است که آیژان می‌تواند با استفاده از آن‌ها دنباله‌ی S را مرتب کند. علاوه بر این، در برخی از زیرمسئله‌ها شما باید کوتاهترین دنباله‌ی جابه‌جایی‌ها را پیدا کنید. شما می‌توانید فرض کنید که دنباله‌ی S با M مرحله یا کمتر، قابل مرتب‌سازی است.

توجه کنید که اگر آیژان متوجه شود که بعد از جابه‌جایی ارمک دنباله‌ی S مرتب شده است، او می‌تواند دو اندیس یکسان را جابه‌جا کند (برای نمونه، ۰ و ۰). در نتیجه دنباله‌ی S بعد از پایان این مرحله مرتب شده است و در نتیجه آیژان به هدفش می‌رسد. هم‌چنین توجه کنید در صورتی که دنباله‌ی S در ابتدا مرتب باشد، کمترین تعداد مرحله‌ی مورد نیاز ۰ خواهد بود.

مثال ۱

فرض کنید:

- دنباله‌ی اولیه $S = ۴, ۳, ۲, ۱, ۰$ است.
- ارمک تمایل دارد $M = ۶$ جابه‌جایی انجام دهد.
- دنباله‌های X و Y ای که اندیس‌های انتخابی ارمک را نشان می‌دهند عبارتند از: $X = ۰, ۱, ۲, ۳, ۰, ۱$ و $Y = ۱, ۲, ۳, ۴, ۱, ۲$. به عبارت دیگر، جفت اندیس‌هایی که ارمک قصد انتخاب آن‌ها را دارد عبارتند از (۰, ۱)، (۱, ۲)، (۲, ۳)، (۳, ۴)، (۰, ۱) و (۱, ۲).

در این سناریو آیژان می‌تواند دنباله‌ی S را در سه مرحله به دنباله ۰, ۱, ۲, ۳, ۴ تبدیل کند. او می‌تواند این کار با انتخاب اندیس‌های (۰, ۴)، (۱, ۳) و سپس (۳, ۴) انجام دهد.

جدول زیر نشان می‌دهد که ارمک و آیژان چگونه دنباله را تغییر داده‌اند.

مرحله	بازیکن	جابه‌جایی	دنباله (از چپ به راست)
شروع			۴, ۳, ۲, ۱, ۰
۰	ارمک	(۰, ۱)	۳, ۴, ۲, ۱, ۰
۰	آیژان	(۰, ۴)	۰, ۴, ۲, ۱, ۳
۱	ارمک	(۱, ۲)	۰, ۲, ۴, ۱, ۳
۱	آیژان	(۱, ۳)	۰, ۱, ۴, ۲, ۳
۲	ارمک	(۲, ۳)	۰, ۱, ۲, ۴, ۳
۲	آیژان	(۳, ۴)	۰, ۱, ۲, ۳, ۴

مثال ۲

فرض کنید:

- دنباله‌ی اولیه $S = ۳, ۰, ۴, ۲, ۱$ است.
- ارمک تمایل دارد $M = ۵$ جابه‌جایی انجام دهد.
- جفت اندیس‌هایی که ارمک قصد انتخاب آن‌ها را دارد عبارتند از (۱, ۱)، (۴, ۰)، (۳, ۲)، (۴, ۱) و (۴, ۰).

در این سناریو آیژان می‌تواند دنباله‌ی S را در سه مرحله مرتب کند. برای مثال با انتخاب جفت اندیس‌های (۴, ۱)، (۲, ۴) و سپس (۲, ۲). جدول زیر نشان می‌دهد که ارمک و آیژان چگونه دنباله را تغییر داده‌اند.

مرحله	بازیکن	جابه‌جایی	دنباله (از چپ به راست)
شروع			۳, ۰, ۴, ۲, ۱
۰	ارمک	(۱, ۱)	۳, ۰, ۴, ۲, ۱
۰	آیژان	(۱, ۴)	۳, ۱, ۴, ۲, ۰
۱	ارمک	(۴, ۰)	۰, ۱, ۴, ۲, ۳
۱	آیژان	(۴, ۲)	۰, ۱, ۳, ۲, ۴
۲	ارمک	(۲, ۳)	۰, ۱, ۲, ۳, ۴
۲	آیژان	(۲, ۲)	۰, ۱, ۲, ۳, ۴

مسئله

دنباله‌ی S، عدد M، و دنباله‌های X و Y به شما داده شده است. دنباله‌ای از جابه‌جایی‌ها را پیدا کنید که به کمک آن‌ها آیژان می‌تواند دنباله‌ی S را مرتب کند. در زیرمسئله‌های ۵ و ۶ شما باید کوتاه‌ترین دنباله‌ی ممکن از جابه‌جایی‌ها را پیدا کنید.

شما باید تابع `findSwapPairs` را پیاده‌سازی کنید:

- `findSwapPairs(N, S, M, X, Y, P, Q)` — این تابع دقیقا یک بار از طرف ارزیاب فراخوانی می‌شود.

- N: طول دنباله‌ی S
- S: آرایه‌ای از اعداد صحیح شامل دنباله‌ی اولیه‌ی S
- M: تعداد جابه‌جاهایی که ارمک قصد دارد انجام دهد.
- X، Y: آرایه‌هایی به طول M از اعداد صحیح. به ازای هر $0 \leq i \leq M-1$، ارمک قصد دارد در مرحله‌ی i اندیس‌های $X[i]$ و $Y[i]$ را جابه‌جا کند.
- P، Q: آرایه‌ای از اعداد صحیح. از این آرایه‌ها، برای گزارش دنباله‌ی جابه‌جایی‌هایی که آیژان توسط آن‌ها می‌تواند دنباله‌ی S را مرتب کند استفاده کنید. مقدار R را برابر با طول دنباله‌ی جابه‌جایی‌هایی که برنامه‌ی شما پیدا کرده است، در نظر بگیرید. برای هر i بین 0 تا $R-1$ (شامل هر دو)، اندیس‌هایی که آیژان در مرحله‌ی i انتخاب می‌کند باید در $P[i]$ و $Q[i]$ ذخیره شوند. شما می‌توانید فرض کنید که برای هر یک از آرایه‌های P و Q، تعداد M عنصر در حافظه اختصاص داده شده است.
- این تابع باید مقدار R (که در بالا تعریف شده) را به عنوان خروجی برگرداند.

زیرمسئله‌ها

زیرمسئله	امتیاز	N	M	محدودیت‌های دیگر روی X و Y	محدودیت روی R
1	8	$1 \leq N \leq 5$	$M = N^2$	$X[i] = Y[i] = 0$ for all i	$R \leq M$
2	12	$1 \leq N \leq 100$	$M = 30N$	$X[i] = Y[i] = 0$ for all i	$R \leq M$
3	16	$1 \leq N \leq 100$	$M = 30N$	$X[i] = 0, Y[i] = 1$ for all i	$R \leq M$
4	18	$1 \leq N \leq 500$	$M = 30N$	none	$R \leq M$
5	20	$6 \leq N \leq 2,000$	$M = 3N$	none	minimum possible
6	26	$6 \leq N \leq 200,000$	$M = 3N$	none	minimum possible

شما می‌توانید فرض کنید که جوابی با M مرحله و یا کمتر وجود دارد.

ارزیاب نمونه

ارزیاب نمونه ورودی را از فایل sorting.in با فرمتی که در زیر آمده است، می‌خواند:

- سطر ۱: N
- سطر ۲: S[0] تا S[N-1]
- سطر ۳: M
- سطر ۴ تا $M+3$: X[i] سپس Y[i]

ارزیاب نمونه خروجی‌های زیر را چاپ می‌کند:

- سطر ۱: مقدار R، خروجی تابع findSwapPairs
- سطر $i+2$، برای $0 \leq i < R$: P[i] سپس Q[i]

اسب‌ها

منصور مانند نیاکانش عاشق پرورش اسب است. او در حال حاضر مالک بزرگ‌ترین گله‌ی اسب در قزاقستان است. ولی همیشه این طور نبوده است. N سال پیش، منصور جوانی بود با تنها یک اسب و آرزو داشت که روزی پولدار شود.

فرض کنید سال‌ها را به ترتیب زمانی از ۰ تا $N-1$ شماره‌گذاری می‌کنیم (یعنی سال $N-1$ آخرین سال است). آب و هوای سال‌های مختلف بر میزان رشد گله اثر می‌گذارد. برای هر سال i، منصور عدد صحیح مثبت $X[i]$ را به عنوان ضریب رشد آن سال به یاد می‌آورد. این به این معنی است که اگر در ابتدای سال i گله شامل h اسب باشد، در انتهای این سال $h \times X[i]$ اسب در گله خواهد بود.

فروش اسب‌ها تنها در پایان سال ممکن است. برای هر سال i، منصور یک عدد صحیح مثبت $Y[i]$ را به عنوان قیمت فروش اسب در انتهای سال i به یاد می‌آورد. این به این معنی است که در پایان سال i، منصور می‌توانسته است هر تعداد اسبی را به مبلغ $Y[i]$ به ازای هر اسب بفروشد.

منصور دوست دارد بداند که اگر در طی این N سال اسب‌هایش را در بهترین زمان‌های ممکن فروخته بود، بیشترین مقدار پولی که می‌توانست داشته باشد چه‌قدر است. شما این افتخار را داشته‌اید که در طی تعطیلات مهمان منصور باشید، و او از شما خواسته است که به این سؤال پاسخ دهید.

هر چه از شب می‌گذرد، حافظه‌ی منصور بهتر می‌شود و او M بار خاطراتش را به‌روزرسانی می‌کند. هر به‌روزرسانی مقدار یکی از $X[i]$ها یا یکی از $Y[i]$ها را تغییر می‌دهد. پس از هر به‌روزرسانی، منصور دوباره از شما می‌پرسد که بیشترین مقدار پولی که می‌توانست داشته باشد چه‌قدر است. این به‌روزرسانی‌ها «افزاینده» هستند، یعنی هر بار که می‌خواهید به سؤال جواب دهید، باید همه‌ی به‌روزرسانی‌های قبلی را اعمال کرده باشید. توجه کنید که مقدار یک $X[i]$ یا یک $Y[i]$ ممکن است بیش از یک بار تغییر کند.

جواب درست سؤال منصور ممکن است عدد بسیار بزرگی باشد. برای این که لازم نباشد با عددهای بزرگ کار کنید، از شما خواسته شده است که جواب را تنها در پیمانه‌ی $۱۰^۹ + ۷$ محاسبه کنید.

مثال

فرض کنید $N = ۳$ سال با اطلاعات زیر داریم:

سال	۰	۱	۲
X	۲	۱	۳
Y	۳	۴	۱

برای مقادیر اولیه‌ی فوق، منصور بیشترین مقدار پول را با فروختن هر دو اسبش در پایان سال ۱ به دست می‌آورد. کل فرایند به صورت زیر است:

- در ابتدا، منصور یک اسب دارد.

- پس از پایان سال ۰، او $X[۰] = ۲$ × ۱ اسب خواهد داشت.
- پس از پایان سال ۱، او $X[۱] = ۲$ × ۲ اسب خواهد داشت.
- او می‌تواند هر دوی این اسب‌ها را در پایان سال ۱ بفروشد. میزان درآمد او $Y[۱] = ۸$ × ۲ خواهد بود.

حالا فرض کنید که $M = ۱$ به‌روزرسانی داریم: مقدار $Y[۱]$ باید به ۲ تغییر کند. پس از این به‌روزرسانی، مقادیر X و Y به این صورت هستند:

سال	۰	۱	۲
X	۲	۱	۳
Y	۳	۲	۱

پس از این تغییر، یک جواب بهینه این است که ۱ اسب را در پایان سال ۰ و پس از آن ۳ اسب را در پایان سال ۲ بفروشیم. کل فرایند به صورت زیر خواهد بود:

- در ابتدا، منصور یک اسب دارد.
- پس از پایان سال ۰، او $X[۰] = ۲$ × ۱ اسب خواهد داشت.
- او می‌تواند یکی از این دو اسب را به قیمت $Y[۰] = ۳$ بفروشد و یک اسب را نگه دارد.
- پس از پایان سال ۱، او $X[۱] = ۱$ × ۱ اسب خواهد داشت.
- پس از پایان سال ۲، او $X[۲] = ۳$ × ۱ اسب خواهد داشت.
- او می‌تواند این سه اسب را در پایان سال ۲ به قیمت $Y[۲] = ۳$ × ۳ بفروشد. میزان کل درآمد او $۳ + ۳ = ۶$ خواهد بود.

مسئله

مقادیر N، X، Y و لیست به‌روزرسانی‌ها به شما داده شده است. قبل از اولین به‌روزرسانی و پس از هر یک از به‌روزرسانی‌ها، بیشترین مقدار پولی که منصور می‌توانست از فروش اسب‌هایش به دست آورد را در پیمانه‌ی $۱۰^۹ + ۷$ محاسبه کنید. شما باید سه تابع `init`، `updateX` و `updateY` به شرح زیر را پیاده‌سازی کنید:

- `init(N, X, Y)` — ارزیاب این تابع را در ابتدا فقط یک بار فراخوانی می‌کند.
  - `N`: تعداد سال‌ها
  - `X`: آرایه‌ای به طول N. برای هر $۰ \le i \le N - ۱$، $X[i]$ ضریب رشد گله در سال i است.
  - `Y`: آرایه‌ای به طول N. برای هر $۰ \le i \le N - ۱$، $Y[i]$ قیمت یک اسب در پایان سال i است.
  - دقت کنید که `X` و `Y` مقادیر اولیه‌ای که منصور به شما داده (قبل از هر گونه به‌روزرسانی) هستند.
  - پس از خاتمه‌ی تابع `init`، آرایه‌های `X` و `Y` معتبر باقی می‌مانند و شما اگر بخواهید می‌توانید مقدارشان را تغییر دهید.
  - این تابع باید بیشترین مقدار درآمد ممکن برای منصور به ازای این مقادیر اولیه را در پیمانه‌ی $۱۰^۹ + ۷$ برگرداند.

- `updateX(pos, val)`
  - `pos`: عددی صحیح در محدوده‌ی ۰ تا $N-1$.
  - `val`: مقدار جدید $X[\texttt{pos}]$.
  - این تابع باید بیشترین مقدار درآمد ممکن برای منصور پس از این به‌روزرسانی را در پیمانه‌ی 10^9+7 برگرداند.
- `updateY(pos, val)`
  - `pos`: عددی صحیح در محدوده‌ی ۰ تا $N-1$.
  - `val`: مقدار جدید $Y[\texttt{pos}]$.
  - این تابع باید بیشترین مقدار درآمد ممکن برای منصور پس از این به‌روزرسانی را در پیمانه‌ی 10^9+7 برگرداند.

می‌توانید فرض کنید که مقادیر اولیه، و همچنین مقادیر به‌روزرسانی شده‌ی $X[i]$ و $Y[i]$، همگی بین ۱ و 10^9 هستند. پس از فراخوانی تابع `init`، ارزیاب توابع `updateX` و `updateY` را چندین بار فراخوانی می‌کند. تعداد کل فراخوانی‌های این دو تابع M است.

زیرمسئله‌ها

زیرمسئله	امتیاز	N	M	محدودیت‌های دیگر
1	17	$1 \le N \le 10$	$M = 0$	$X[i], Y[i] \le 10$, $X[0] \times X[1] \times \cdots \times X[N-1] \le 1,000$
2	17	$1 \le N \le 1,000$	$0 \le M \le 1,000$	none
3	20	$1 \le N \le 500,000$	$0 \le M \le 100,000$	$X[i] \ge 2$ and `val` ≥ 2 for `init` and `updateX` correspondingly
4	23	$1 \le N \le 500,000$	$0 \le M \le 10,000$	none
5	23	$1 \le N \le 500,000$	$0 \le M \le 100,000$	none

ارزیاب نمونه

ارزیاب نمونه ورودی را از فایل `horses.in` در قالب زیر می‌خواند:

- خط ۱: `N`
- خط ۲: `X[0]` تا `X[N - 1]`
- خط ۳: `Y[0]` تا `Y[N - 1]`
- خط ۴: `M`
- خطوط ۵ تا $M+4$: سه عدد `type`، `pos` و سپس `val` (`type=1` به معنای فراخوانی `updateX` و `type=2` به معنای `updateY` است).

ارزیاب نمونه مقدار خروجی تابع `init` و پس از آن مقادیر خروجی توابع `updateX` و `updateY` را چاپ می‌کند.