نکته‌ها

هر مسئله یک بسته پیوست (attachment package) دارد که هم در CMS و هم برروی میزکار (desktop) شما موجود است.

برای مسئله‌های «فقط-خروجی»:

- بسته پیوست شامل ورودی‌های آزمایه (input test cases) و آزمایه‌های مثال (example test cases) هست. هر آزمایه یک زیرمسئله مجزاست.
- شما می‌توانید چندین فایل خروجی را به صورت یک فایل فشرده zip ارسال (submit) کنید. برای این منظور لازم است فایل‌های خروجی شما به صورت `??.out` نام‌گذاری شوند که `??` شماره آزمایه است (برای مثال `03.out`). شما می‌توانید چندین فایل را با دستور زیر به صورت zip درآورید: `zip output.zip *.out`
- شما می‌توانید برای مسئله‌های فقط-خروجی تا حداکثر ۱۰۰ ارسال داشته باشید. در هر ارسال شما می‌توانید فایل‌های خروجی برای هر زیرمجموعه‌ای از آزمایه‌ها را ارسال کنید.

برای سایر مسئله‌ها:

- بسته پیوست شامل ارزیاب‌های نمونه (sample graders)، پیاده‌سازی‌های نمونه (sample implementations)، آزمایه‌های مثال (example test cases)، و اسکریپت‌های کامپایل (compile scripts) است.
- شما لازم است دقیقا یک فایل ارسال کنید و می‌توانید حداکثر ۵۰ ارسال داشته باشید.
- ارسال‌های شما نباید از ورودی استاندار بخوانند، در خروجی استاندارد بنویسند، و یا با هر فایل دیگر ارتباط برقرار کنند. با این حال می‌تواند در خروجی خطای استاندارد (standard error) بنویسد.
- نام فایلی که شما باید ارسال کنید در بالای برگه صورت مسئله مشخص شده است. فایل شما باید تابع‌هایی که در صورت مسئله مشخص شده است را با مشخصات (signature) تعیین شده در پیاده‌سازی‌های نمونه (sample implementations) پیاده‌سازی کند.
- شما آزادید که توابع دیگری را نیز پیاده‌سازی کنید.
- هنگامی‌که برنامه خود را با ارزیاب نمونه (sample grader) ارزیابی می‌کنید، ورودی ارزیاب باید منطبق با الگوی ارائه شده در پیاده‌سازی نمونه باشد وگرنه ممکن است از ارزیاب نمونه رفتاری نامشخص رخ دهد.

قراردادها

صورت مسئله مشخصات تابع را با نوع‌های عمومی (generic type names) `bool`، `integer`، `int64` و `int[]` (آرایه) نشان می‌دهد.

در هر زبان برنامه‌نویسی مورد پشتیبانی، نوع‌های داده مناسب برای پیاده‌سازی به شرح زیر است:

زبان	bool	integer	int64	int[]	طول آرایه a
C++	`bool`	`int`	`long long`	`std::vector<int>`	`a.size()`
Pascal	`boolean`	`longint`	`int64`	`array of longint`	`Length(a)`
Java	`boolean`	`int`	`long`	`int[]`	`a.length`

محدودیت‌ها

مسئله	محدودیت زمانی	محدودیت حافظه
nowruz	فقط-خروجی	فقط-خروجی
wiring	1 ثانیه	256 مگابایت
train	2 ثانیه	256 مگابایت

نوروز

تنها چند روز تا نوروز (سال نوی ایرانی) باقی‌مانده است، و پدربزرگ خانواده‌اش را به باغش دعوت کرده‌است. در میان مهمانان k کودک هستند. پدربزرگ می‌خواهد برای سرگرم کردن کودکان بازی قایم-باشک برگزار کند.

باغ را می‌توان با یک جدول $m \times n$ از خانه‌های واحد نمایش داد. تعدادی از خانه‌ها (شاید صفر عدد) توسط سنگ اشغال شده‌اند، و سایر خانه‌ها *خالی* هستند. دو خانه *همسایه* گفته می‌شوند اگر یک ضلع مشترک داشته باشند. یعنی هر خانه حداکثر ۴ همسایه دارد: دو همسایه در جهت افقی و دو همسایه در جهت عمودی. پدربزرگ می‌خواهد باغ خود را به یک هزارتو تبدیل کند. به‌این منظور او می‌تواند بعضی از خانه‌های خالی را با کاشتن بوته اشغال کند. خانه‌هایی که پدربزرگ در آن بوته بکارد دیگر خالی نیستند.

یک هزارتو باید دارای این خاصیت باشد: برای هر جفت خانه خالی a و b از هزارتو، باید دقیقا یک *مسیر ساده* بین آن‌ها وجود داشته باشد. یک مسیر ساده بین خانه‌های a و b یک دنباله از خانه‌های خالی است که در آن اولین خانه a، و آخرین خانه b است، همه خانه‌ها متفاوت هستند، و هر دو خانه متوالی همسایه هستند.

یک کودک می‌تواند در یک خانه پنهان شود اگر و تنها اگر آن خانه خالی باشد و دقیقا یک همسایه خالی (خانه خالی یعنی خانه‌ای که شامل سنگ و بوته نیست) داشته باشد. همچنین هیچ دو کودکی نمی‌توانند در یک خانه پنهان شوند.

نقشه باغ در ورودی به شما داده می‌شود. شما باید به پدربزرگ کمک کنید تا هزارتویی بسازد که کودکان زیادی بتوانند در آن پنهان شوند.

جزئیات پیاده‌سازی

این یک مسئله فقط-خروجی با امتیازدهی جزئی (partial) است. به شما ۱۰ فایل ورودی داده می‌شود که هر کدام باغ پدربزرگ را توصیف می‌کنند. برای هر فایل ورودی شما باید یک فایل خروجی شامل نقشه هزارتو را ارسال کنید. برای هر فایل خروجی بر اساس کودکانی که می‌توانند در هزارتو پنهان شوند به شما امتیاز داده خواهد شد.

شما نباید هیچ کد منبعی (source code) برای این مسئله ارسال کنید.

قالب ورودی

هر فایل ورودی یک جدول که باغ را توصیف می‌کند و تعداد کودکان k که توسط پدربزرگ دعوت شده‌اند را نشان می‌دهد. قالب به این صورت است:

- سطر 1: m n k
- سطر $1+i$ (برای $1 \le i \le m$): سطر i جدول، که رشته‌ای به طول n است، شامل حروف زیر (بدون فاصله (space)) است:
  - '.': یک خانه خالی،

- '#': یک سنگ.

قالب خروجی

- سطر i (برای $1 \leq i \leq m$): سطر i از هزارتو (باغ، پس از کاشتن بوته‌ها). این سطر یک رشته به طول n است که شامل حروف زیر (بدون فاصله (space)) است:
  - '.': یک خانه خالی،
  - '#': یک سنگ،
  - 'X': یک بوته. (توجه کنید که حرف X باید به صورت حرف بزرگ نوشته شود.)

محدودیت‌ها

- $1 \leq m, n \leq 1024$

امتیازدهی

یک فایل خروجی *معتبر* به حساب می‌آید اگر شرایط زیر را داشته باشد:

- نقشه خروجی باید با نقشه ورودی منطبق باشد مگر این‌که برخی از نقاطی که با حروف '.' مشخص شده‌اند به حروف 'X' تبدیل شده باشد (که مشخص کننده خانه‌هایی است که در آن بوته کاشته شده است).
- نقشه خروجی باید مشخصات هزارتو را، همان‌طور که در صورت مسئله مشخص شده، داشته باشد.

اگر خروجی برای یک آزمایه معتبر نباشد، امتیاز شما برای آن آزمایه 0 خواهد بود. در غیر این صورت، شما $\min(10, 10 \cdot l/k)$ امتیاز خواهید گرفت که تا دو رقم اعشار به سمت پایین گرد خواهد شد. در اینجا، l تعداد کودکانی است که می‌توانند در هزارتوی شما پنهان شوند، و k عددی است که در ورودی به شما داده شده است.

شما 10 امتیاز خواهید گرفت اگر و فقط اگر به تعداد k یا بیشتر کودک بتوانند در هزارتوی شما پنهان شوند. برای هر آزمایه پاسخی وجود دارد که 10 امتیاز می‌گیرد.

توجه کنید که اگر پاسخ شما معتبر باشد ولی بر اساس فرمول بالا 0 امتیاز بگیرد، CMS عبارت `Wrong Answer` را به عنوان نتیجه نمایش می‌دهد.

مثال

ورودی زیر را در نظر بگیرید.

```
4 5 5
....#
.#..#
...#.
....#
```

در زیر یک خروجی معتبر ممکن آمده است.

```
.X.X#
.#..#
...#X
XX..#
```

از آن‌جایی که $l = 4$ کودک می‌توانند در این هزارتو پنهان شوند، این راه حل $10 \cdot 4/5 = 8$ امتیاز می‌گیرد. خانه‌هایی که کودکان می‌توانند در آن پنهان شوند با ○ در پایین نشان داده شده است.

```
○X○X#
.#.○#
...#X
XX.○#
```

سه خروجی زیر، معتبر نیستند:

```
.XXX#        ...X#        XXXX#
.#XX#        .#.X#        X#XX#
...#.        ...#X        ..X#X
XX..#        XXXX#        ..XX#
```

در خروجی سمت چپ، هیچ مسیر ساده‌ای بین خانه خالی در گوشه بالا چپ و خانه خالی در سمت راست‌ترین ستون وجود ندارد. در دو خروجی دیگر، برای هر جفت از خانه‌های خالی دو مسیر ساده مجزا بین آن‌ها وجود دارد.

سیم‌کشی

مریم یک مهندس برق است. او سیم‌کشی یک برج مخابراتی را طراحی می‌کند. در برج، تعدادی نقطه اتصال وجود دارد که در ارتفاع‌های متفاوت قرار گرفته‌اند. یک سیم می‌تواند برای اتصال هر دو نقطه به هم استفاده شود. هر نقطه اتصال می‌تواند به چند سیم متصل گردد. دو نوع نقطه اتصال وجود دارد: قرمز و آبی.

برای این مسئله، برج به صورت یک خط دیده می‌شود و نقاط اتصال به صورت نقاط آبی و قرمز با مختصات صحیح نامنفی بر روی این خط قرار دارند. طول یک سیم، فاصله دو نقطه اتصالی است که این سیم آن‌ها را به هم متصل می‌کند.

هدف شما کمک به مریم برای یافتن یک شِمای سیم‌کشی است به‌نحوی که:

- هر نقطه اتصال حداقل یک سیم به یک نقطه اتصال با رنگ متفاوت داشته باشد.
- مجموع طول سیم‌ها کمینه باشد.

جزئیات پیاده‌سازی

شما باید تابع زیر را پیاده‌سازی کنید:

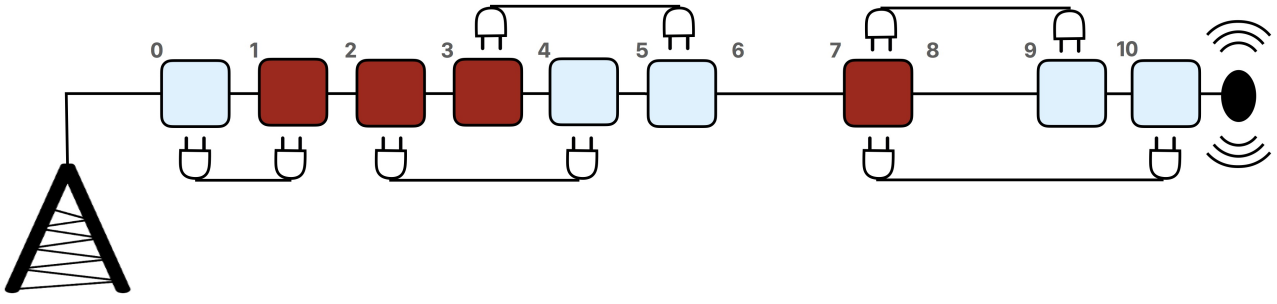
```
int64 min_total_length(int[] r, int[] b)
```

- r: آرایه‌ای به طول n شامل مکان نقطه‌های اتصال قرمز به ترتیب صعودی.
- b: آرایه‌ای به طول m شامل مکان نقطه‌های اتصال آبی به ترتیب صعودی.
- این تابع، باید کمینه مجموع طول سیم‌ها را در بین همه سیم‌کشی‌های معتبر برگرداند.
- توجه کنید نوع برگشتی این تابع `int64` است.

مثال

```
min_total_length([1, 2, 3, 7], [0, 4, 5, 9, 10])
```

تصویر زیر یک مثال را نشان می‌دهد.



- برج به صورت افقی نشان‌داده شده است.
- در نسخه پرینت‌شده سیاه و سفید این مسئله، نقاط قرمز به صورت تیره و نقاط آبی به صورت روشن نشان‌داده می‌شوند.
- ۴ نقطه اتصال قرمز وجود دارد که در مکان‌های ۱، ۲، ۳، و ۷ قرار دارند.
- ۵ نقطه اتصال آبی وجود دارد که در مکان‌های ۰، ۴، ۵، ۹، و ۱۰ قرار دارند.
- در این راه حل، مجموع طول سیم‌ها $1 + 2 + 2 + 2 + 3 = 10$ است که مقدار کمینه است. بنابراین تابع باید مقدار ۱۰ را برگرداند.
- توجه کنید دو سیم به یک نقطه اتصال در مکان ۷ متصل شده‌اند.

محدودیت‌ها

- $1 \leq n, m \leq 100\,000$.
- $0 \leq r[i] \leq 10^9$ (برای هر $0 \leq i \leq n - 1$).
- $0 \leq b[i] \leq 10^9$ (برای هر $0 \leq i \leq m - 1$).
- هر کدام از آرایه‌های r و b به صورت صعودی مرتب شده‌اند.
- همه $n + m$ عدد از آرایه‌های r و b متفاوت هستند.

زیرمسئله‌ها

1. (7 امتیاز) $n, m \leq 200$.
2. (13 امتیاز) همه نقاط اتصال قرمز، مکانی کوچک‌تر از نقاط اتصال آبی دارند.
3. (10 امتیاز) حداقل یک نقطه اتصال قرمز و یک نقطه اتصال آبی در بین هر ۷ نقطه اتصال متوالی وجود دارد.
4. (25 امتیاز) همه نقاط اتصال دارای مکانی در بازه $[1, n + m]$ هستند.
5. (45 امتیاز) بدون محدودیت.

ارزیاب نمونه

ارزیاب نمونه، ورودی را در قالب زیر می‌خواند:

- سطر 1: $n \;\; m$
- سطر 2: $r[0] \;\; r[1] \;\; \ldots \;\; r[n-1]$
- سطر 3: $b[0] \;\; b[1] \;\; \ldots \;\; b[m-1]$

ارزیاب نمونه، یک سطر شامل مقدار برگشتی `min_total_length` را چاپ می‌کند.

قطار اسباب‌بازی

آرزو و برادرش برزو دوقلو هستند. آن‌ها یک مجموعه قطار اسباب‌بازی جالب برای تولدشان هدیه گرفته‌اند و می‌خواهند یک سیستم راه‌آهن با n ایستگاه و m خط آهن *یک طرفه* بسازند. ایستگاه‌ها از 0 تا $n-1$ شماره‌گذاری شده‌اند. هر خط آهن از یک ایستگاه آغاز می‌شود و به همان ایستگاه یا یک ایستگاه متفاوت ختم می‌شود. از هر ایستگاه حداقل یک خط آهن آغاز می‌شود.

بعضی از ایستگاه‌ها، *ایستگاه شارژکننده* هستند. هنگامی‌که قطار به یک ایستگاه شارژکننده می‌رسد، به طور کامل شارژ می‌شود. قطار با شارژ کامل به اندازه کافی برای پیمودن n خط آهن متوالی شارژ دارد. یعنی پس از آن‌که شارژ شد، به محض ورود به $n+1$-امین خط آهن، شارژش تمام می‌شود.

در هر ایستگاه یک کلید وجود دارد که می‌تواند به هر کدام از خطوط آهنی که از آن ایستگاه آغاز می‌شوند، اشاره کند. هنگامی که یک قطار وارد یک ایستگاه می‌شود، از خط آهنی که کلید این ایستگاه به آن اشاره می‌کند خارج می‌شود.

دوقلوها می‌خواهند یک بازی با قطارشان انجام بدهند. آن‌ها همه ایستگاه‌ها را بین خودشان تقسیم کرده‌اند: هر ایستگاه یا متعلق به آرزوست و یا متعلق به برزو. تنها یک قطار وجود دارد. در آغاز بازی قطار در ایستگاه s قرار دارد و کاملا شارژ شده است. برای شروع بازی، صاحب ایستگاه s کلید ایستگاه s را به یکی از خطوط آهنی که از ایستگاه s آغاز می‌شود، اشاره می‌دهد. سپس قطار را روشن می‌کنند و قطار حرکت خود را در خطوط آهن آغاز می‌کند.

هنگامی‌که یک قطار برای نخستین بار وارد یک ایستگاه می‌شود، صاحب آن ایستگاه کلید آن ایستگاه را تنظیم می‌کند. هنگامی که کلید تنظیم شود، وضعیت کلید، یعنی خط آهنی که کلید به آن اشاره می‌کند، تا آخر بازی ثابت می‌ماند. بنابراین اگر قطار دوباره وارد ایستگاهی شود که قبلا آن را ملاقات کرده است، از همان خط آهنی که قبلا خارج شده است، دوباره خارج خواهد شد.

از آن‌جایی که تعداد محدودی ایستگاه وجود دارد، قطار در نهایت وارد یک *دور* خواهد شد. یک دور دنباله‌ای از ایستگاه‌های متفاوت $c[0], c[1], \cdots, c[k-1]$ است به‌طوری‌که قطار ایستگاه $c[i]$ (برای $0 \leq i < k-1$) را با خط آهنی که به سمت ایستگاه $c[i+1]$ می‌رود ترک می‌کند، و ایستگاه $c[k-1]$ را با خط آهنی که به سمت ایستگاه $c[0]$ می‌رود ترک می‌کند. توجه کنید یک دور ممکن است تنها شامل یک ایستگاه (یعنی $k=1$) باشد اگر قطار ایستگاه $c[0]$ را با استفاده از خط آهنی که به ایستگاه $c[0]$ بازمی‌گردد ترک کند.

آرزو برنده بازی خواهد بود اگر قطار به صورت نامتناهی به حرکت خود ادامه دهد، و برزو برنده بازی خواهد بود اگر شارژ قطار تمام شود. به عبارت دیگر، اگر حداقل یک ایستگاه شارژکننده در بین $c[0], c[1], \cdots, c[k-1]$ وجود داشته باشد، قطار دوباره شارژ می‌شود و دور بدون انتها ادامه پیدا می‌کند، و آرزو برنده می‌شود. در غیر این‌صورت، شارژ قطار (شاید بعد از چندین بار چرخیدن در دور) بالاخره تمام می‌شود و برزو برنده می‌شود.

به شما توصیف سیستم خط آهن داده شده است. آرزو و برزو می‌خواهند n بازی انجام دهند. در بازی s-ام، برای $0 \leq s \leq n-1$، قطار در ابتدا در ایستگاه s خواهد بود. شما باید برای هر بازی پیدا کنید که آیا یک استراتژی برای آرزو وجود دارد که برد آرزو را، علیرغم چگونگی بازی برزو، تضمین کند.

جزئیات پیاده‌سازی

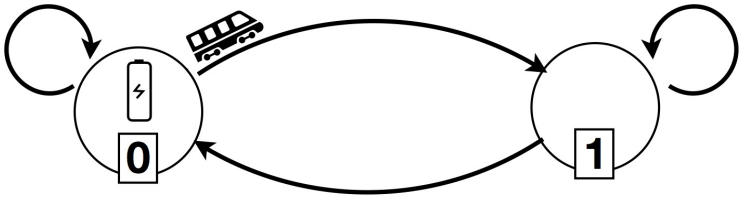
شما باید این تابع را پیاده‌سازی کنید:

```
int[] who_wins(int[] a, int[] r, int[] u, int[] v)
```

- a: آرایه‌ای به طول n. اگر آرزو صاحب ایستگاه i باشد, $a[i] = 1$. در غیر این صورت، برزو صاحب ایستگاه i است و $a[i] = 0$.
- r: آرایه‌ای به طول n. اگر ایستگاه i یک ایستگاه شارژکننده باشد، $r[i] = 1$. در غیر این صورت، $r[i] = 0$.
- u و v: آرایه‌هایی به طول m. برای هر $0 \leq i \leq m - 1$، یک خط آهن یک طرفه که از ایستگاه $u[i]$ آغاز می‌شود و در ایستگاه $v[i]$ خاتمه می‌یابد وجود دارد.
- این تابع باید آرایه w به طول n را باز گرداند. برای هر $0 \leq i \leq n - 1$، مقدار $w[i]$ باید 1 باشد اگر آرزو بتواند بازی که از ایستگاه i آغاز می‌شود را، علیرغم چگونگی بازی برزو، ببرد. در غیر این‌صورت, مقدار $w[i]$ باید 0 باشد.

مثال

```
who_wins([0, 1], [1, 0], [0, 0, 1, 1], [0, 1, 0, 1])
```



- دو ایستگاه وجود دارد. برزو صاحب ایستگاه ۰ است که یک ایستگاه شارژکننده است. آرزو صاحب ایستگاه ۱ است که ایستگاهی شارژکننده نیست.
- ۴ خط آهن $(0, 0)$، $(0, 1)$، $(1, 0)$، و $(1, 1)$ وجود دارد که (i, j) یک خط آهن یک طرفه که از ایستگاه i آغاز می‌شود و به ایستگاه j خاتمه می‌یابد را نشان می‌دهد.
- بازی را در نظر بگیرید که در ابتدا قطار در ایستگاه ۰ قرار دارد. اگر برزو کلید ایستگاه ۰ را به سمت خط آهن $(0, 0)$ تنظیم کند، قطار به صورت نامتناهی در این خط آهن دور خواهد زد (توجه کنید ایستگاه ۰ یک ایستگاه شارژکننده است). در این حالت آرزو برنده می‌شود. در غیر این صورت، اگر برزو کلید ایستگاه ۰ را به سمت خط آهن $(0, 1)$ تنظیم کند، آرزو می‌تواند کلید ایستگاه ۱ را به سمت خط آهن $(1, 0)$ تنظیم کند. در این صورت، قطار به صورت نامتناهی در بین دو ایستگاه دور خواهد زد، و از آن‌جایی که ایستگاه ۰ یک ایستگاه شارژکننده است و قطار متوقف نمی‌شود، دوباره آرزو برنده می‌شود. بنابراین آرزو می‌تواند بازی را ببرد، علیرغم هر حرکتی که برزو انجام بدهد.
- با استدلالی مشابه، در حالتی که بازی از ایستگاه ۱ آغاز می‌شود، آرزو می‌تواند، علیرغم هر بازی که برزو انجام دهد، برنده شود. بنابراین تابع باید $[1, 1]$ را برگرداند.

محدودیت‌ها

- $1 \leq n \leq 5000$.
- $n \leq m \leq 20\,000$.
- حداقل یک ایستگاه شارژکننده وجود دارد.

- از هر ایستگاه حداقل یک خط آهن خارج می‌شود.
- ممکن است برخی از خط‌های آهن از یک ایستگاه شروع شده و به همان ایستگاه ختم شوند (یعنی $u[i] = v[i]$).
- هر دو خط آهنی متفاوت هستند. به عبارت دیگر، هیچ دو اندیس i و j ($0 \leq i < j \leq m - 1$) وجود ندارند که برای آن‌ها داشته باشیم $u[i] = u[j]$ و $v[i] = v[j]$.
- $0 \leq u[i], v[i] \leq n - 1$ (برای هر $0 \leq i \leq m - 1$).

زیرمسئله‌ها

1. (5 امتیاز) برای هر $0 \leq i \leq m - 1$، داریم $v[i] = u[i]$ یا $v[i] = u[i] + 1$.
2. (10 امتیاز) $n \leq 15$.
3. (11 امتیاز) آرزو صاحب همه ایستگاه‌هاست.
4. (11 امتیاز) برزو صاحب همه ایستگاه‌هاست.
5. (12 امتیاز) دقیقا یک ایستگاه شارژکننده وجود دارد.
6. (51 امتیاز) بدون محدویت.

ارزیاب نمونه

ارزیاب نمونه ورودی را در قالب زیر می‌خواند:

- سطر 1: $n \;\; m$
- سطر 2: $a[0] \;\; a[1] \;\; \ldots \;\; a[n-1]$
- سطر 3: $r[0] \;\; r[1] \;\; \ldots \;\; r[n-1]$
- سطر $4 + i$ (برای $0 \leq i \leq m - 1$): $u[i] \;\; v[i]$

ارزیاب نمونه خروجی `who_wins` را در قالب زیر چاپ می‌کند:

- سطر 1: $w[0] \;\; w[1] \;\; \ldots \;\; w[n-1]$

نکته‌ها

هر مسئله یک بسته پیوست (attachment package) دارد که هم در CMS و هم برروی میزکار (desktop) شما موجود است.

- بسته پیوست شامل ارزیاب‌های نمونه (sample graders)، پیاده‌سازی‌های نمونه (sample implementations)، آزمایه‌های مثال (example test cases)، و اسکریپت‌های کامپایل (compile scripts) است.
- شما لازم است دقیقا یک فایل ارسال کنید و می‌توانید حداکثر ۵۰ ارسال داشته باشید.
- ارسال‌های شما نباید از ورودی استاندار بخوانند، در خروجی استاندارد بنویسند، و یا با هر فایل دیگر ارتباط برقرار کنند. با این حال می‌تواند در خروجی خطای استاندارد (standard error) بنویسد.
- نام فایلی که شما باید ارسال کنید در بالای برگه صورت مسئله مشخص شده است. فایل شما باید تابع‌هایی که در صورت مسئله مشخص شده است را با مشخصات (signature) تعیین شده در پیاده‌سازی‌های نمونه (sample implementations) پیاده‌سازی کند.
- شما آزادید که توابع دیگری را نیز پیاده‌سازی کنید.
- هنگامی‌که برنامه خود را با ارزیاب نمونه (sample grader) ارزیابی می‌کنید، ورودی شما برای ارزیاب باید منطبق با الگوی ارائه شده در پیاده‌سازی نمونه باشد وگرنه ممکن است از ارزیاب نمونه رفتاری نامشخص رخ دهد.

قراردادها

صورت مسئله مشخصات تابع را با نوع‌های عمومی (generic type names) `bool`، `int`، `int64`، و `int[]` (آرایه) نشان می‌دهد.

در هر زبان برنامه‌نویسی مورد پشتیبانی، نوع‌های داده مناسب برای پیاده‌سازی به شرح زیر است:

زبان	bool	int	int64	int[]	طول آرایه a
C++	`bool`	`int`	`long long`	`std::vector<int>`	`a.size()`
Pascal	`boolean`	`longint`	`int64`	`array of longint`	`Length(a)`
Java	`boolean`	`int`	`long`	`int[]`	`a.length`

محدودیت‌ها

مسئله	محدودیت زمانی	محدودیت حافظه
prize	۱ ثانیه	۱۰۲۴ مگابایت
simurgh	۳ ثانیه	۱۰۲۴ مگابایت
books	۲ ثانیه	۱۰۲۴ مگابایت

جایزه بزرگ

«جایزه بزرگ» یک مسابقه تلویزیونی مشهور است. شما یک شرکت‌کننده خوش‌شانس هستید که به دور نهایی راه یافته‌اید. شما در مقابل یک سطر از n جعبه ایستاده‌اید که از چپ به راست با 0 تا $n-1$ برچسب‌گذاری شده‌اند. هر جعبه جایزه‌ای دارد که تا وقتی جعبه باز نشده است نمی‌توان آن را مشاهده کرد. $v \geq 2$ *نوع* مختلف جایزه وجود دارد. این نوع جایزه‌ها از 1 تا v به ترتیب *نزولی* ارزششان شماره‌گذاری شده‌اند.

جایزه از نوع 1 گران‌ترین جایزه‌است: الماس. دقیقا یک الماس در میان جعبه‌ها وجود دارد. جایزه از نوع v ارزان‌ترین جایزه است: آب‌نبات چوبی. برای اینکه بازی هیجان‌انگیزتر شود، تعداد جایزه‌های ارزان خیلی بیشتر از تعداد جایزه‌های گران‌تر است. به صورت دقیق‌تر، برای هر t که $2 \leq t \leq v$ می‌دانیم که شرط روبرو برقرار است: اگر k جایزه از نوع $t-1$ داشته باشیم، تعداد جایزه‌های با نوع t *اکیدا بیشتر* از k^2 است.

هدف شما این است که الماس را ببرید. در پایان بازی شما باید یک جعبه را انتخاب کنید و برنده جایزه داخل آن جعبه خواهید شد. قبل از انتخاب این‌که چه جعبه‌ای را باز کنید، می‌توانید از رامبد، مجری این بازی تلویزیونی، چند سوال بپرسید. برای هر سوال، شما یک جعبه مانند i را انتخاب می‌کنید. در جواب، رامبد به شما آرایه a را خواهد داد که شامل دو عدد است. معنی آن‌ها به صورت زیر است:

- بین همه جعبه‌های سمت چپ جعبه i، دقیقا $a[0]$ جعبه جایزه‌ای گران‌تر از جعبه i دارند.
- بین همه جعبه‌های سمت راست جعبه i، دقیقا $a[1]$ جعبه جایزه‌ای گران‌تر از جعبه i دارند.

برای نمونه، فرض کنید $n = 8$. شما برای سوال خود جعبه $i = 2$ را انتخاب می‌کنید. رامبد در پاسخ، به شما می‌گوید که $a = [1, 2]$. معنی این پاسخ این است که:

- دقیقا یک جعبه از جعبه‌های 0 و 1 جایزه‌ای گران‌تر از جعبه شماره 2 دارد.
- دقیقا دو جعبه از جعبه‌های 3، 4، . . .، 7 جایزه‌ای گران‌تر از جعبه 2 دارند.

شما باید جعبه حاوی الماس را با تعداد کمی پرسش بیابید.

جزییات پیاده‌سازی

شما باید تابع زیر را پیاده‌سازی کنید:

```
int find_best(int n)
```

- این تابع دقیقا یکبار توسط ارزیاب (grader) فراخوانی می‌شود.
- n: تعداد جعبه‌ها.
- این تابع باید برچسب جعبه‌ای که حاوی الماس است را برگرداند. یعنی، عدد یکتای d ($0 \leq d \leq n-1$) که جعبه d حاوی جایزه از نوع 1 است.

تابع بالا می‌تواند تابع زیر را فراخوانی کند:

```
int[] ask(int i)
```

- i: برچسب جعبه‌ای که برای پرسش انتخاب کرده‌اید. مقدار i باید بین 0 و $n-1$ (شامل 0 و $n-1$) باشد.
- این تابع آرایه a شامل 2 عضو را بر می‌گرداند. اینجا، $a[0]$ تعداد جعبه‌های گران‌تر از جعبه i در جعبه‌های سمت چپ جعبه i است و $a[1]$ تعداد جعبه‌های گران‌تر از جعبه i در جعبه‌های سمت راست جعبه i است.

مثال

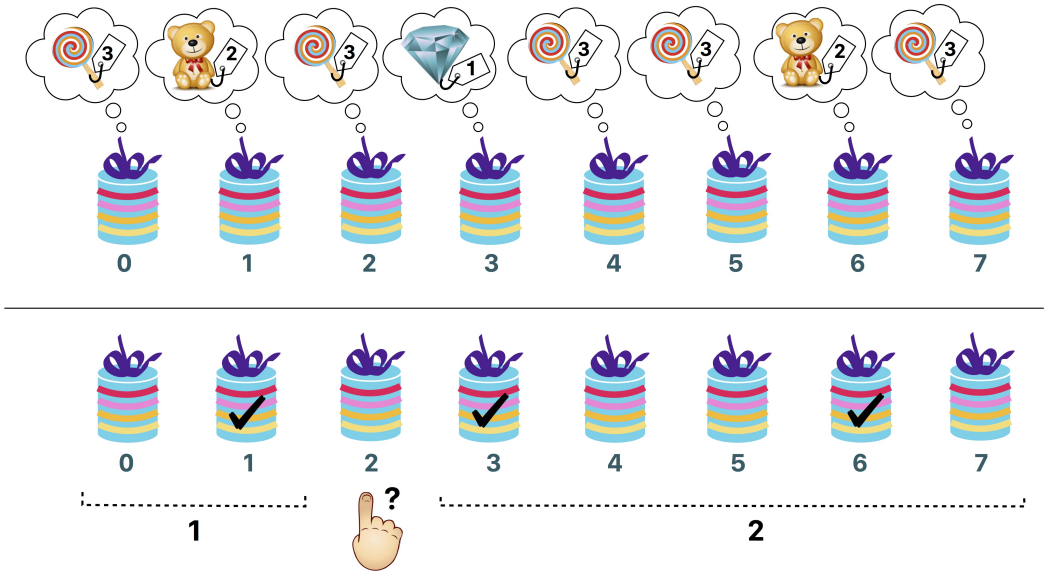
ارزیاب، تابع زیر را فراخوانی می‌کند.

```
find_best(8)
```

تعداد جعبه‌ها $n = 8$ است. فرض‌کنید که نوع جایزه‌ها $[3, 2, 3, 1, 3, 3, 2, 3]$ باشد. همه فراخوانی‌های ممکن تابع `ask` و مقدار برگشتی آن‌ها در زیر لیست شده است:

- `ask(0)` آرایه $[0, 3]$ را بر می‌گرداند.
- `ask(1)` آرایه $[0, 1]$ را بر می‌گرداند.
- `ask(2)` آرایه $[1, 2]$ را بر می‌گرداند.
- `ask(3)` آرایه $[0, 0]$ را بر می‌گرداند.
- `ask(4)` آرایه $[2, 1]$ را بر می‌گرداند.
- `ask(5)` آرایه $[2, 1]$ را بر می‌گرداند.
- `ask(6)` آرایه $[1, 0]$ را بر می‌گرداند.
- `ask(7)` آرایه $[3, 0]$ را بر می‌گرداند.

در این مثال، الماس در جعبه 3 است. بنابراین تابع `find_best` باید عدد 3 را برگرداند.



شکل بالا همین مثال را نمایش می‌دهد. بخش بالا ارزش جایزه‌های هر جعبه را نمایش می‌دهد. بخش پایین

پرسش `ask(2)` را نمایش می‌دهد. جعبه‌های علامت‌خورده حاوی جایزه‌های گران‌تر از جایزه درون جعبه 2 هستند.

محدودیت‌ها

- $3 \le n \le 200\,000$.
- نوع جایزه در هر جعبه عددی بین 1 تا v (شامل 1 و v) است.
- دقیقا یک جایزه نوع 1 وجود دارد.
- برای هر $2 \le t \le v$، اگر k جایزه با نوع $t-1$ وجود داشته‌باشد، تعداد جایزه‌های با نوع t *کمتر* از k^2 بیشتر است.

زیرمسئله‌ها و امتیازدهی

در بعضی از آزمایه‌ها (test cases) رفتار ارزیاب تطبیقی (adaptive) است. یعنی در این آزمایه‌ها ارزیاب دنباله معینی از جایزه‌ها ندارد. در عوض، پاسخ‌های ارزیاب ممکن است به سوالات پرسیده شده توسط راه حل شما بستگی داشته باشد. تضمین می‌شود که ارزیاب به گونه‌ای پاسخ می‌دهد که پس از هر پاسخ، حداقل یک دنباله از جایزه‌ها وجود داشته باشد که با پاسخ‌های تاکنون داده شده مطابقت داشته باشد .

1. (۲۰ امتیاز) دقیقا 1 الماس و $n-1$ آب‌نبات چوبی داریم (پس، $v=2$). شما می‌توانید تابع `ask` را حداکثر $10\,000$ بار فراخوانی کنید.
2. (۸۰ نمره) بدون محدودیت اضافی.

در زیرمسئله ۲ شما می‌توانید نمره جزئی (partial score) دریافت کنید. q را بیشترین تعداد فراخوانی تابع `ask` در میان همه آزمایه‌های (test cases) این زیرمسئله در نظر بگیرید. سپس، امتیاز شما برای این زیرمسئله مطابق جدول زیر محاسبه می‌شود:

تعداد سوالات	امتیاز
$10\,000 < q$	0 (در CMS با عبارت 'Wrong Answer' گزارش می‌شود)
$6000 < q \le 10\,000$	70
$5000 < q \le 6000$	$80 - (q - 5000)/100$
$q \le 5000$	80

ارزیاب نمونه

ارزیاب نمونه تطبیقی نیست. در عوض، آرایه معین p از نوع جایزه‌ها را از ورودی می‌خواند و استفاده می‌کند. برای هر $0 \le b \le n-1$، نوع جایزه در جعبه b برابر با $p[b]$ است. ارزیاب نمونه ورودی را در قالب زیر می‌خواند:

- سطر 1: n
- سطر 2: $p[0]\ \ p[1]\ \ \ldots\ \ p[n-1]$

ارزیاب نمونه یک سطر شامل مقدار برگشتی `find_best` و تعداد فراخوانی تابع `ask` را چاپ می‌کند.

سیمرغ

بنا بر افسانه‌ای قدیمی در شاهنامه، زال، قهرمان افسانه‌ای ایران به‌طور دیوانه‌واری دلداده رودابه شاهزاده کابل است. هنگامی‌که زال از رودابه درخواست ازدواج می‌کند، پدر رودابه چالشی را برای او طرح می‌کند.

در ایران، n شهر که از 0 تا $n-1$ برچسب‌گذاری شده، و m جاده دوطرفه که از 0 تا $m-1$ برچسب‌گذاری شده وجود دارد. هر جاده یک جفت شهر متفاوت را به هم متصل می‌کند. هر جفت شهر با حداکثر یک جاده به هم وصل شده‌اند. بعضی از جاده‌ها، *جاده سلطنتی* هستند که توسط افراد سلطنتی استفاده می‌شوند. وظیفه زال این است که مشخص کند کدام جاده‌ها سلطنتی هستند.

زال یک نقشه از همه شهرها و جاده‌های ایران دارد. او نمی‌داند که کدام‌یک از جاده‌ها سلطنتی هستند، اما می‌تواند از سیمرغ، پرنده پاک‌بین افسانه‌ای که محافظ زال است، کمک بگیرد. با این حال، سیمرغ نمی‌خواهد مجموعه جاده‌های سلطنتی را به طور مستقیم برای زال آشکار کند. به‌جای آن، به زال می‌گوید که مجموعه جاده‌های سلطنتی یک *مجموعه طلایی* است. یک مجموعه از جاده‌ها، یک مجموعه طلایی است اگر و تنها اگر دارای شرایط زیر باشد:

- دقیقا شامل $n-1$ جاده باشد، و
- برای هر جفت از شهرها، بتوان از شهر اول تنها با حرکت بر جاده‌های این مجموعه به شهر دوم رسید.

به علاوه، زال می‌تواند از سیمرغ سوال‌هایی بپرسد. برای هر سوال:

1. زال یک مجموعه طلایی از جاده‌ها را انتخاب می‌کند، و بعد
2. سیمرغ به زال می‌گوید که چندتا از جاده‌های مجموعه انتخاب شده توسط زال، جاده‌های سلطنتی هستند.

برنامه شما باید به زال کمک کند تا با پرسیدن حداکثر q سوال از سیمرغ، مجموعه جاده‌های سلطنتی را پیدا کند. ارزیاب نقش سیمرغ را بازی می‌کند.

جزئیات پیاده‌سازی

شما باید تابع زیر را پیاده‌سازی کنید:

```
int[] find_roads(int n, int[] u, int[] v)
```

- n: تعداد شهرها،
- u و v: آرایه‌هایی به طول m هستند. برای هر $0 \le i \le m-1$، $u[i]$ و $v[i]$ شهرهایی هستند که با جاده i به هم متصل هستند.
- این تابع باید یک آرایه‌ای به طول $n-1$، شامل برچسب جاده‌های سلطنتی را (به ترتیب دلخواه) برگرداند.

پاسخ شما می‌تواند حداکثر q فراخوانی از این تابع ارزیاب انجام دهد:

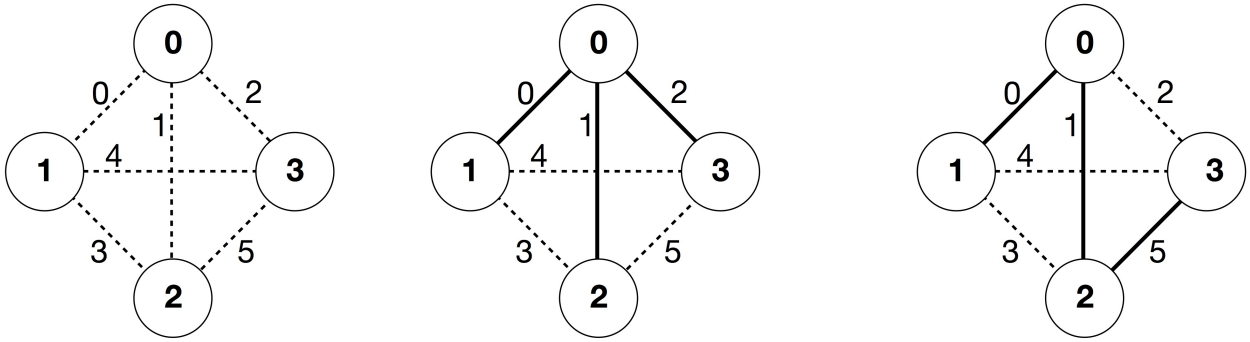
```
int count_common_roads(int[] r)
```

- r: آرایه‌ای به طول $n-1$ شامل برچسب‌های جاده‌ها در یک مجموعه طلایی (به ترتیب دلخواه).
- این تابع تعداد جاده‌های سلطنتی در r را برمی‌گرداند.

مثال

```
find_roads(4, [0, 0, 0, 1, 1, 2], [1, 2, 3, 2, 3, 3])
```

find_roads(...) count_common_roads([0, 1, 2]) = 2 count_common_roads([5, 1, 0]) = 3



در این مثال ۴ شهر و ۶ جاده وجود دارد. ما یک جاده که دو شهر a و b را به هم متصل می‌کند را با (a, b) نشان می‌دهیم. جاده‌ها از 0 تا 5 به این‌ترتیب برچسب‌گذاری شده‌اند: $(0, 1)$، $(0, 2)$، $(0, 3)$، $(1, 2)$، $(1, 3)$، و $(2, 3)$. هر مجموعه طلایی $n - 1 = 3$ جاده دارد.

فرض کنید جاده‌های سلطنتی جاده‌هایی هستند که با 0، 1، و 5 برچسب‌گذاری شده‌اند، یعنی جاده‌های $(0, 1)$، $(0, 2)$، و $(2, 3)$. سپس:

- `count_common_roads([0, 1, 2])` مقدار 2 را برمی‌گرداند. این پرسش مربوط به جاده‌های برچسب‌گذاری شده با 0، 1، و 2 یعنی جاده‌های $(0, 1)$، $(0, 2)$، و $(0, 3)$ است. دو تا از آن‌ها جاده‌های سلطنتی هستند.
- `count_common_roads([5, 1, 0])` مقدار 3 را برمی‌گرداند. این پرسش مربوط به مجموعه همه جاده‌های سلطنتی است.

تابع `find_roads` باید مقدار `[5, 1, 0]` را برگرداند، یا هر آرایه‌ای به طول ۳ که شامل این سه عضو است.

توجه کنید که فراخوانی‌های زیر مجاز نیستند:

- `count_common_roads([0, 1])`: در اینجا طول r برابر با 3 نیست.
- `count_common_roads([0, 1, 3])`: در اینجا r یک مجموعه طلایی را مشخص نمی‌کند، زیرا سفر از شهر 0 به شهر 3 تنها با استفاده از جاده‌های $(0, 1)$، $(0, 2)$، و $(1, 2)$ ممکن نیست.

محدودیت‌ها

- $2 \le n \le 500$
- $n - 1 \le m \le n(n-1)/2$
- $0 \le u[i], v[i] \le n - 1$ (برای هر $0 \le i \le m - 1$)
- برای هر $0 \le i \le m - 1$، جاده i دو شهر متفاوت را به هم متصل می‌کند (یعنی، $u[i] \neq v[i]$).
- حداکثر یک جاده بین هر دو شهر وجود دارد.
- حرکت بین هر دو شهر از طریق جاده‌ها امکان‌پذیر است.
- مجموعه همه جاده‌های سلطنتی یک مجموعه طلایی است.
- `find_roads` باید تابع `count_common_roads` را حداکثر q بار فراخوانی کند. در هر فراخوانی مجموعه جاده‌های مشخص شده توسط r باید یک مجموعه طلایی باشد.

زیرمسئله‌ها

1. (۱۳ امتیاز) $n \le 7$، $q = 30\,000$
2. (۱۷ امتیاز) $n \le 50$، $q = 30\,000$
3. (۲۱ امتیاز) $n \le 240$، $q = 30\,000$
4. (۱۹ امتیاز) $q = 12\,000$ و بین هر جفت شهر یک جاده وجود دارد
5. (۳۰ امتیاز) $q = 8000$

ارزیاب نمونه

ارزیاب نمونه ورودی را با قالب زیر می‌خواهند:

- سطر 1: $n \;\; m$
- سطر $2 + i$ (برای هر $0 \le i \le m - 1$): $u[i] \;\; v[i]$
- سطر $2 + m$: $s[0] \;\; s[1] \;\; \ldots \;\; s[n-2]$

در اینجا، $s[0]$، $s[1]$، . . .، $s[n-2]$ برچسب‌های جاده‌های سلطنتی هستند.

ارزیاب نمونه مقدار `YES` را به عنوان خروجی می‌دهد، اگر `find_roads` تابع `count_common_roads` را حداکثر $30\,000$ بار فراخوانی کند، و مجموعه صحیح جاده‌های سلطنتی را برگرداند. در غیر این‌صورت، `NO` را به عنوان خروجی می‌دهد.

توجه کنید که تابع `count_common_roads` در ارزیاب نمونه بررسی نمی‌کند که r همه خواص مجموعه طلایی را داشته باشد. در مقابل، تعداد برچسب‌های جاده‌های سلطنتی در آرایه r را می‌شمارد و برمی‌گرداند. اما، اگر برنامه‌ای که شما ارسال می‌کنید تابع `count_common_roads` را با مجموعه‌ای از برچسب‌ها فراخوانی کند که یک مجموعه طلایی را توصیف نمی‌کند، نتیجه امتیازدهی 'Wrong Answer' خواهد بود.

نکات فنی

تابع `count_common_roads` در C++ و Pascal به دلایل کارآیی از روش ارسال با ارجاع (pass by reference) استفاده می‌کند. شما همچنان می‌توانید این تابع را به روش متداول فراخوانی کنید. ارزیاب تضمین می‌کند که مقدار r را تغییر نخواهد داد.

کتاب‌های کهن

شهر تهران محل کتابخانه ملی ایران است. گنجینه اصلی این کتابخانه در یک سالن طولانی شامل یک ردیف از n میز، که از 0 تا $n-1$ از چپ به راست برچسب‌گذاری شده‌اند، قرارگرفته است. بر روی هر میز، یک کتاب خطی قدیمی برای نمایش قرار داده شده است. این کتاب‌ها بر اساس قدمتشان مرتب شده‌اند که جستجوی کتاب‌ها بر اساس عنوان را برای بازدیدکنندگان دشوار می‌کند. بنابراین، مدیر کتابخانه تصمیم گرفته است تا کتاب‌ها را بر اساس عنوان به ترتیب حروف الفبا مرتب کند.

آرین، کتابداری است که قرار است این کار را انجام بدهد. او لیست p به طول n، شامل اعداد صحیح متفاوت از 0 تا $n-1$ تهیه کرده است. این لیست تغییراتی که برای مرتب‌سازی کتاب‌ها به ترتیب حروف الفبا لازم است را مشخص می‌کند: برای هر $0 \leq i < n$، کتابی که هم‌اکنون بر روی میز i قرار دارد باید به میز $p[i]$ منتقل شود.

آرین مرتب‌سازی کتاب‌ها را در کنار میز s شروع می‌کند. او می‌خواهد پس از خاتمه کار به کنار همین میز بازگردد. از آن‌جایی که کتاب‌ها بسیار ارزشمند هستند، او نمی‌تواند هم‌زمان بیش از یک کتاب را حمل کند. در زمان مرتب‌سازی کتاب‌ها، آرین دنباله‌ای از عملیات را انجام می‌دهد. هر یک از این عملیات باید یکی از موارد زیر باشد:

- اگر او هیچ کتابی را حمل نکند و یک کتاب بر روی میزی که در کنار آن ایستاده است قرار داشته باشد، می‌تواند آن کتاب را بردارد.
- اگر او یک کتاب را حمل کند و کتاب دیگری بر روی میزی که در کنار آن ایستاده است قرار داشته باشد، می‌تواند کتابی را که حمل می‌کند با کتاب روی میز تعویض کند.
- اگر او کتابی را حمل کند و در کنار یک میز خالی ایستاده باشد، می‌تواند کتاب را بر روی میز خالی قرار دهد.
- او می‌تواند به کنار هر میزی برود. او در زمان حرکت می‌تواند یک کتاب را حمل کند.

برای هر $0 \leq i, j \leq n-1$، فاصله بین دو میز i و j دقیقا $|j-i|$ متر است. وظیفه شما محاسبه مقدار کمینه مجموع فاصله‌ای است که لازم است آرین برای مرتب‌سازی کتاب‌ها بپیماید.

جزئیات پیاده‌سازی

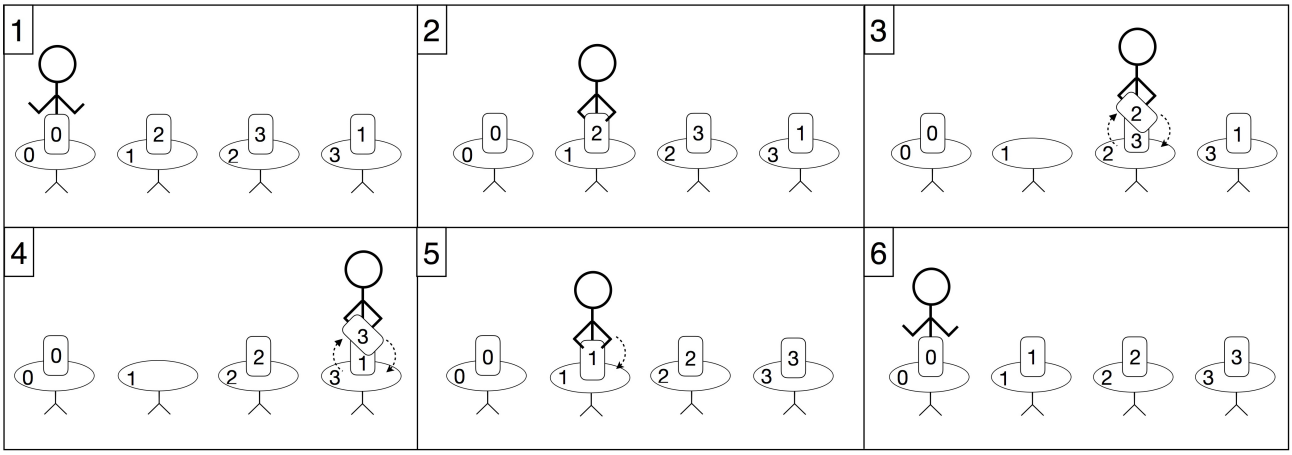
شما لازم است تابع زیر را پیاده‌سازی نمایید:

```
int64 minimum_walk(int[] p, int s)
```

- p آرایه‌ای به طول n است. کتابی که در ابتدا بر روی میز i قرار دارد، باید توسط آرین به میز $p[i]$ منتقل شود (برای هر $0 \leq i < n$).
- s برچسب میزی است که آرین در ابتدای کار در کنار آن قرار دارد و پس از مرتب‌سازی کتاب‌ها باید در کنار آن قرار داشته باشد.
- این تابع باید مقدار کمینه مجموع فاصله‌ای (بر حسب متر) را برگرداند که آرین برای مرتب‌سازی کتاب‌ها باید بپیماید.

مثال

```
minimum_walk([0, 2, 3, 1], 0)
```



در این مثال، $n = 4$ و آرین در ابتدا در کنار میز 0 ایستاده است. او کتاب‌ها را به این صورت مرتب می‌کند:

- او به کنار میز 1 رفته و کتابی را که بر روی آن قرار دارد برمی‌دارد. این کتاب باید بر روی میز 2 قرار بگیرد.
- سپس، او به کنار میز 2 رفته و کتابی را که حمل می‌کند با کتاب روی میز تعویض می‌کند. کتاب جدیدی که حمل می‌کند باید بر روی میز 3 قرار بگیرد.
- سپس، او به کنار میز 3 رفته و کتابی را که حمل می‌کند با کتاب روی میز تعویض می‌کند. کتاب جدیدی که حمل می‌کند، باید بر روی میز 1 قرار گیرد.
- سپس، او به کنار میز 1 رفته و کتابی را که حمل می‌کند بر روی میز قرار می‌دهد.
- در انتها به کنار میز 0 بازمی‌گردد.

توجه کنید، کتابی که بر روی میز 0 قرار دارد از ابتدا در محل صحیح، میز 0، قرار دارد. بنابراین، نیازی نیست که آرین آن کتاب را بردارد. مجموع فاصله‌ای که او در این راه حل می‌پیماید، 6 متر است. این یک پاسخ بهینه است. بنابراین، تابع باید مقدار 6 را برگرداند.

محدودیت‌ها

- $1 \leq n \leq 1\,000\,000$
- $0 \leq s \leq n - 1$
- آرایه p شامل n عدد صحیح متفاوت بین 0 تا $n - 1$ (شامل 0 و $n - 1$) است.

زیرمسئله‌ها

1. (۱۲ امتیاز) $n \leq 4$ و $s = 0$
2. (۱۰ امتیاز) $n \leq 1000$ و $s = 0$
3. (۲۸ امتیاز) $s = 0$
4. (۲۰ امتیاز) $n \leq 1000$
5. (۳۰ امتیاز) بدون محدودیت اضافی

ارزیاب نمونه

ارزیاب نمونه ورودی را با این قالب می‌خواند:

- سطر 1: $n \;\; s$
- سطر 2: $p[0] \;\; p[1] \;\; \ldots \;\; p[n-1]$

ارزیاب نمونه یک سطر شامل مقدار برگشتی `minimum_walk` را چاپ می‌کند.