شهر زیرزمینی تمساح‌ها (Crocodile's Underground City)

بنیاماس(Benjamas) یک باستان‌شناس است که پس از کشف شهر زیرزمینی تمساح‌ها برای نجات دادن جان خود در تلاش است. این شهر شامل N اتاق است. M راهروی دو طرفه نیز وجود دارد که هر کدام از راهروها دو اتاق متفاوت را به یکدیگر وصل می‌کنند. گذر از راهروهای متفاوت ممکن است زمان‌های متفاوتی طول بکشد. تنها K تا از این N اتاق، اتاق خروجی هستند که به بنیاماس اجازه‌ی فرار می‌دهند. بنیاماس اکنون در اتاق شماره‌ی صفر هست و می‌خواهد هر چه سریع‌تر خود را به یکی از اتاق‌های خروجی برساند.

نگهبان تمساح‌ها می‌خواهد مانع فرار بنیاماس شود. او از اتاق خودش درب‌های مخفی‌ای را کنترل می‌کند که با کمک آنها در هر لحظه تنها **یک راهرو** را می‌تواند مسدود کند. به‌این معنی‌که هر گاه او یک راهروی جدید را مسدود می‌کند، راهرویی که قبلاً مسدود شده باز می‌شود.

وضعیت بنیاماس را می‌توان به این شکل توضیح داد: هر زمانی که او تلاش می‌کند از یک اتاق خارج شود، نگهبان ممکن است یکی از راهروهای مجاور آن اتاق را مسدود کند. لذا، بنیاماس از طریق یکی از راهروهای مسدود نشده به اتاق دیگری می‌رود. وقتی بنیاماس وارد یک راهرو می‌شود، نگهبان نمی‌تواند تا زمانی که بنیامین به انتهای دیگر راهرو نرسیده است، آن راهرو را مسدود کند. در زمانی که بنیاماس به اتاق جدید رسید، نگهبان می‌تواند راهرو جدیدی مجاور اتاق جدید (حتی راهرویی که اخیراً عبور شده) را مسدود کند و ... .

بنیاماس مایل است که یک برنامه‌ی فرارِ آسان از قبل داشته باشد. به طور دقیق‌تر، او علاقه‌مند است که مجموعه دستورالعملی داشته باشد که به او بگوید که بعد از این که وارد یک اتاق شد چه کند. فرض کنید A یکی از اتاق‌ها باشد. بدیهی است که اگر A یک اتاق خروجی باشد نیازی به هیچ دستورالعملی نیست چرا که او می‌تواند از طریق همین اتاق از شهر فرار کند. در غیر این صورت، دستورالعمل مربوط به اتاق A باید به یکی از دو صورت زیر باشد:

- «اگر به اتاق A رسیدی، راهروی متصل به اتاق B را طی کن. ولی اگر این راهرو مسدود بود، راهروی متصل به اتاق C را طی کن.»
- «نگران اتاق A نباش. طبق برنامه‌ی فرار، هیچ‌گاه به این اتاق نخواهی رسید.»

توجه کنید که در بعضی حالات (برای مثال، اگر برنامه شما بنیاماس را داخل یک حلقه بیندازد) نگهبان ممکن است قادر به جلوگیری از رسیدن بنیاماس به اتاق‌های خروج شود. یک برنامه‌ی فرار، **خوب** است اگر تضمین کند که مستقل از نحوه‌ی کار نگهبان، بنیاماس را در زمانی متناهی به یک اتاق خروج برساند. برای یک برنامه‌ی فرار خوب، فرض کنید T کمترین زمانی باشد که برنامه تضمین می‌کند همواره بنیاماس (مستقل از عمل‌کرد نگهبان) در آن زمان به یک اتاق خروجی برسد. در این حالت می‌گوییم این برنامه‌ی فرارِ خوب، T واحد زمان طول می‌کشد.

تمساح

فارسی نسخه ۱٫۳

وظیفه شما

شما باید روال `travel_plan(`N, M, R, L, K, P`)` را بنویسید که پارامترهای زیر را بگیرد:

- N : تعداد اتاق‌ها. اتاق‌ها با شماره‌ی صفر تا $N-1$ شماره‌گذاری شده‌اند.
- M : تعداد راهروها. راهروها با شماره‌ی صفر تا $M-1$ شماره‌گذاری شده‌اند.
- R : یک آرایه‌ی دوبُعدی از اعداد صحیح که راهروها را نمایش می‌دهد. برای $0 \le i < M$، راهروی i دو اتاق متمایز $R[i][0]$ و $R[i][1]$ را به هم وصل می‌کند. هر دو اتاق با حداکثر یک راهرو به یکدیگر وصل می‌شوند.
- L : یک آرایه‌ی یک‌بُعدی از اعداد صحیح که شامل زمان گذر از هر راهرو است. برای $0 \le i < M$، زمان $1 \le L[i] < 1{,}000{,}000{,}000$ زمانیست که بنیاماس نیاز دارد که از اتاق $R[i][0]$ به اتاق $R[i][1]$ برسد.
- K : تعداد اتاق‌های خروجی. می‌توانید فرض کنید که $1 \le K < N$.
- P: یک آرایه‌ی یک‌بُعدی با K عدد متفاوت شامل شماره‌ی اتاق‌های خروجی. برای $0 \le i < K$، مقدار $P[i]$، شماره‌ی اتاق خروجی i-اُم است. اتاق شماره صفر هیچ‌گاه یک اتاق خروجی نخواهد بود.

روال شما می‌بایست کمترین زمان T که برای آن، یک برنامه‌ی فرار خوب وجود دارد را برگرداند.

می‌توانید فرض کنید که هر اتاقی که خروجی نیست حداقل دو راهرو متصل به خود دارد. هم‌چنین می‌توانید فرض کنید که برای هر ورودی یک برنامه فرار خوب با زمان $T \le 1{,}000{,}000{,}000$ موجود است.

مثال نمونه

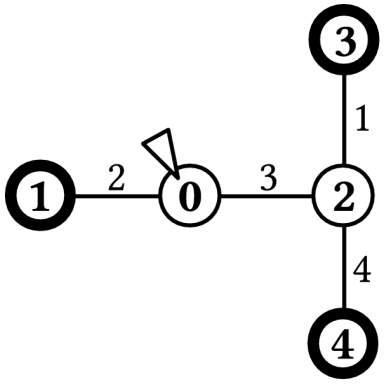
مثال ۱

حالتی را در نظر بگیرید که $N = 5$، $M = 4$، $K = 3$ و

R=
0 1
0 2
3 2
2 4

L=
2
3
1
4

P=
1
3
4



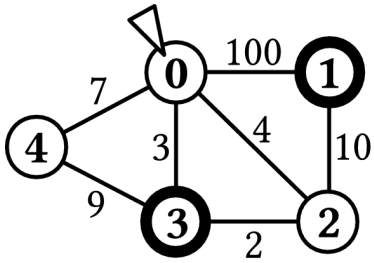
در شکل بالا، اتاق‌ها با دایره و راهروها با پاره‌خط نمایش داده شده‌اند. اتاق‌های خروجی با دایره‌های ضخیم نمایش داده شده‌اند. بنیاماس از اتاق صفر (که با مثلث مشخص شده) شروع می‌کند. یک برنامه‌ی فرار بهینه به این شکل است:

- هرگاه در اتاق صفر بودی، راهرویی را برو که به اتاق ۱ می‌رود. اگر آن راهرو مسدود بود، راهرویی را برو که به اتاق ۲ می‌رود.
- اگر در اتاق ۲ بودی، راهرویی را برو که به اتاق ۳ می‌رود. اگر آن راهرو مسدود بود، راهرویی را برو که به اتاق ۴ می‌رود.

در بدترین حالت، بنیاماس بعد از ۷ واحد زمانی به یکی از اتاق‌های خروجی می‌رسد. لذا `travel_plan` باید مقدار ۷ را برگرداند.

مثال ۲

حالتی را در نظر بگیرید که $N = ۵$، $M = ۷$ و $K = ۲$



R=
0 2
0 3
3 2
2 1
0 1
0 4
3 4

L=
4
3
2
10
100
7
9

P=
1
3

یک برنامه‌ی فرار بهینه به این صورت است:

- هرگاه در اتاق صفر بودی به راهرویی برو که به اتاق ۳ می‌رود و اگر آن راهرو مسدود بود به راهرویی برو که به اتاق ۲ می‌رود.
- هرگاه در اتاق ۲ بودی به راهرویی برو که به اتاق ۳ می‌رود و اگر آن راهرو مسدود بود به راهرویی برو که به اتاق ۱ می‌رود.
- نگران اتاق ۴ نباش. بر اساس این برنامه، هیچ‌گاه به آن نمی‌رسی.

بنیاماس بعد از حداکثر ۱۴ واحد زمان، حتماً به یکی از اتاق‌های خروجی می‌رسد. پس `travel_plan` باید ۱۴ را برگرداند.

زیرمسئله‌ها

زیرمسئله شماره یک (۴۶ امتیاز)

- $۳ \le N \le ۱٫۰۰۰$
- شهر زیرزمینی یک درخت است. یعنی، $M = N - ۱$ و بین هر دو اتاق یک مسیر وجود دارد که آن‌ها را به هم وصل می‌کند.
- هر اتاق خروجی دقیقا به یک اتاق دیگر وصل است.
- هر اتاق دیگر حداقل به ۳ اتاق دیگر وصل است.

زیرمسئله شماره دو (۴۳ امتیاز)

- $۳ \le N \le ۱٫۰۰۰$
- $۲ \le M \le ۱۰۰٫۰۰۰$

زیرمسئله شماره سه (۱۱ امتیاز)

- $۳ \le N \le ۱۰۰٫۰۰۰$
- $۲ \le M \le ۱٫۰۰۰٫۰۰۰$

جزئیات پیاده‌سازی

محدودیت‌ها

- **محدودیت زمانی CPU:** دو ثانیه
- **محدودیت حافظه:** ۲۵۶ مگابایت
- **توجه:** هیچ محدودیت صریحی روی اندازه‌ی حافظه‌ی پشته (stack) استفاده شده وجود ندارد. میزان حافظه‌ی استفاده شده در پشته به‌عنوان بخشی از حافظه‌ی مصرفی محاسبه می‌شود.

واسط API

- فولدرِ پیاده‌سازی: /crocodile
- فایل‌هایی که توسط شرکت‌کننده می‌بایست پیاده‌سازی شوند: crocodile.c یا crocodile.cpp یا crocodile.pas
- واسط شرکت‌کننده: crocodile.h یا crocodile.pas
- واسط مصحح: crocodile.h یا crocodilelib.pas
- مصحّحِ نمونه: grader.c یا grader.cpp یا grader.pas و crocodilelib.pas
- ورودی مصحّح نمونه: grader.in.1، grader.in.2 و ...
- **توجه:** مصحح نمونه، ورودی را به فرمت زیر می‌خواند:
  - **خط اوّل:** مقادیر N، M و در نهایت K.
  - **خط ۲ الی $M+1$ اُم:** برای $0 \le i < M$، خط $i+2$ شامل $R[i][0]$ و $R[i][1]$ و $L[i]$ می‌باشد که با یک فاصله از هم جدا شده‌اند.
  - **خط $M+2$ اُم:** یک لیست شامل K عدد صحیح $P[K-1], \cdots, P[1], P[0]$ که با یک فاصله از هم جدا شده‌اند.
  - **خط $M+3$:** جواب مورد انتظار.
- خروجی مورد انتظار برای ورودی مصحح نمونه: grader.expect.1، grader.expect.2 و ...

برای این مسئله، هر کدام از این فایل‌ها دقیقاً باید شامل عبارت «**Correct.**» باشند.