کلیدها

تیموتیِ معمار یک بازی فرار طراحی کرده است. در این بازی n اتاق وجود دارد که با شماره‌های 0 تا $n-1$ شماره‌گذاری شده‌اند. در ابتدا، داخل هر اتاق دقیقا یک کلید وجود دارد. هر کلید یک نوع دارد که عددی بین 0 تا $n-1$ (شامل آن‌ها) است. نوع کلیدی که داخل اتاق i ($0 \le i \le n-1$) قرار دارد از نوع $r[i]$ است. توجه داشته باشید که یک نوع کلید ممکن است داخل چند اتاق باشد. به عبارتی مقادیر $r[i]$ لزوما متمایز نیست.

همچنین m ارتباط **دو طرفه** در بازی وجود دارد که با شماره‌های 0 تا $m-1$ شماره‌گذاری شده‌اند. ارتباط j ($0 \le j \le m-1$) دو اتاق متمایز $u[j]$ و $v[j]$ را به هم متصل می‌کند. دو اتاق می‌توانند توسط چند ارتباط به هم متصل شوند.

این بازی یک بازیکن دارد که کلیدها را برمی‌دارد و با عبور از ارتباط‌ها بین اتاق‌ها حرکت می‌کند. زمانی می‌گوییم که بازیکن از ارتباط j **عبور** می‌کند که از این ارتباط برای عبور از اتاق $u[j]$ به اتاق $v[j]$ یا برعکس استفاده کند. بازیکن زمانی می‌تواند از ارتباط j عبور کند که کلیدی از نوع $c[j]$ داشته باشد.

هر زمان از بازی، بازیکن اگر داخل اتاق x باشد می‌تواند یکی از دو عملیات زیر را انجام دهد:

- کلید اتاق x که از نوع $r[x]$ است را بردارد. (مگر آنکه پیش از این آن را برداشته باشد).
- از ارتباط j عبور کند اگر $u[j] = x$ یا $v[j] = x$ و همچنین بازیکن پیش از این کلیدی از نوع $c[j]$ برداشته باشد. توجه داشته باشید که بازیکن **هرگز** کلیدی که برداشته است را رها نمی‌کند.

بازیکن بازی را از اتاق s بدون داشتن هیچ کلیدی **شروع** می‌کند. می‌گوییم اتاق t **قابل دسترس** است اگر بازیکن بتواند با شروع از اتاق s و انجام دنباله‌ای از عملیات‌های ذکر شده به اتاق t برسد.

برای اتاق i ($0 \le i \le n-1$)، $p[i]$ تعداد اتاق‌هایی است که با شروع از اتاق i قابل دسترس هستند. تیموتی می‌خواهد شماره‌ی اتاق‌هایی مانند i را بداند که $p[i]$ مقدار کمینه بین $0 \le i \le n-1$ را داشته باشد.

جزئیات پیاده‌سازی

شما باید تابع زیر را پیاده‌سازی کنید:

```
int[] find_reachable(int[] r, int[] u, int[] v, int[] c)
```

- r : آرایه‌ای به طول n . به ازای هر i ($0 \le i \le n-1$), کلید اتاق i از نوع $r[i]$ است.
- u, v : دو آرایه به طول m . به ازای هر j ($0 \le j \le m-1$), ارتباط j اتاق‌های $u[j]$ و $v[j]$ را به هم وصل می‌کند.
- c : یک آرایه به طول m . به ازای هر j ($0 \le j \le m-1$), نوع کلیدی که برای عبور از ارتباط j لازم است، $c[j]$ می‌باشد.
- تابع باید آرایه‌ای به طول n برگرداند این آرایه را a بنامید. به ازای $0 \le i \le n-1$، مقدار $a[i]$ باید 1 باشد اگر به ازای هر j که $0 \le j \le n-1$، داشته باشیم $p[i] \le p[j]$. در غیر اینصورت $a[i]$ باید 0 باشد.

مثال‌ها

مثال ۱

فراخوانی زیر را در نظر بگیرید:

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

اگر بازیکن از اتاق 0 بازی را شروع کند، می‌تواند دنباله زیر از عملیات‌ها را انجام دهد:

عملیات	اتاق کنونی
برداشتن کلید نوع 0	0
عبور از ارتباط 0 و رفتن به اتاق 1	0
برداشتن کلید نوع 1	1
عبور از ارتباط 2 و رفتن به اتاق 2	1
عبور از ارتباط 2 و رفتن به اتاق 1	2
عبور از ارتباط 3 و رفتن به اتاق 3	1

پس اتاق 3 با شروع از اتاق 0 قابل دسترس است. به طور مشابه می توان دنباله‌هایی ساخت که نشان دهد تمامی اتاق‌ها با شروع از اتاق 0 قابل دسترس هستند. که نشان می‌دهد $p[0] = 4$ جدول زیر تمامی اتاق‌های قابل دسترس به ازای همه اتاق‌های شروع را نشان می‌دهد:

i شروع از اتاق	اتاق‌های قابل دسترس	$p[i]$
0	$[0, 1, 2, 3]$	4
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[1, 2, 3]$	3

کمترین مقدار $p[i]$ در میان تمامی اتاق‌ها 2 است و این مقدار برای $i = 1$ و $i = 2$ وجود دارد. پس تابع باید $[0, 1, 1, 0]$ بازگرداند.

نمونه ۲

```
find_reachable([0, 1, 1, 2, 2, 1, 2],
               [0, 0, 1, 1, 2, 3, 3, 4, 4, 5],
               [1, 2, 2, 3, 3, 4, 5, 5, 6, 6],
               [0, 0, 1, 0, 0, 1, 2, 0, 2, 1])
```

جدول زیر اتاق‌های قابل دسترس را نشان می‌دهد:

شروع از اتاق i	اتاق‌های قابل دسترس	$p[i]$
0	$[0, 1, 2, 3, 4, 5, 6]$	7
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[3, 4, 5, 6]$	4
4	$[4, 6]$	2
5	$[3, 4, 5, 6]$	4
6	$[4, 6]$	2

کمترین مقدار $p[i]$ در میان تمامی اتاق‌ها 2 است و این مقدار برای $i \in \{1, 2, 4, 6\}$ وجود دارد. پس تابع باید $[0, 1, 1, 0, 1, 0, 1]$ بازگرداند.

نمونه ۳

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

جدول زیر اتاق‌های قابل دسترس را نشان می‌دهد:

شروع از اتاق i	اتاق‌های قابل دسترس	$p[i]$
0	$[0, 1]$	2
1	$[0, 1]$	2
2	$[2]$	1

کمترین مقدار $p[i]$ در میان تمامی اتاق‌ها 1 است و این مقدار تنها برای $i = 2$ وجود دارد. پس تابع باید $[0, 0, 1]$ بازگرداند.

محدودیت‌ها

- $2 \leq n \leq 300\,000$
- $1 \leq m \leq 300\,000$
- $0 \leq r[i] \leq n - 1$ به ازای $0 \leq i \leq n - 1$
- $0 \leq u[j], v[j] \leq n - 1$ و $u[j] \neq v[j]$ به ازای $0 \leq j \leq m - 1$
- $0 \leq c[j] \leq n - 1$ به ازای $0 \leq j \leq m - 1$

زیرمسئله‌ها

1. (۹ نمره) $c[j] = 0$ به ازای $0 \leq j \leq m - 1$ و $n, m \leq 200$

2. (۱۱ نمره) $n, m \leq 200$
3. (۱۷ نمره) $n, m \leq 2000$
4. (۳۰ نمره) $c[j] \leq 29$ (به ازای $0 \leq j \leq m-1$) و $r[i] \leq 29$ (به ازای $0 \leq i \leq n-1$)
5. (۳۳ نمره) بدون محدودیت اضافی.

ارزیاب نمونه

ارزیاب نمونه ورودی را در قالب زیر میخواند:

- خط 1: $n \;\; m$
- خط 2: $r[0] \;\; r[1] \;\; \ldots \;\; r[n-1]$
- خط $3+j$ ($0 \leq j \leq m-1$): $u[j] \;\; v[j] \;\; c[j]$

ارزیاب نمونه خروجی تابع `find_reachable` را در قالب زیر چاپ می‌کند:

- خط 1: $a[0] \;\; a[1] \;\; \ldots \;\; a[n-1]$