Teka-teki Sfinks

Sang Sfinks Agung punya teka-teki untuk Anda. Anda diberikan graf dengan N verteks. Verteksnya dinomori dari 0 hingga $N-1$. Ada M *edge* pada graf tersebut yang dinomori dari 0 hingga $M-1$. Setiap *edge* menghubungkan sepasang verteks berbeda dan bersifat dua arah. Lebih spesifiknya, untuk setiap j dari 0 hingga $M-1$ (inklusif), *edge* j menghubungkan verteks $X[j]$ dan $Y[j]$. Ada paling banyak satu *edge* yang menghubungkan pasangan verteks mana pun. Dua verteks dikatakan **bersebelahan** jika mereka terhubung oleh sebuah *edge*.

Sekuens verteks $v_0, v_1, \ldots, v_k$ (untuk $k \geq 0$) disebut sebagai **jalur** apabila setiap dua verteks v_l dan v_{l+1} (untuk setiap l yang memenuhi $0 \leq l < k$) bersebelahan pada graf. Kita anggap suatu jalur $v_0, v_1, \ldots, v_k$ **menghubungkan** verteks v_0 dan v_k. Pada graf yang diberikan, setiap pasangan verteks pasti terhubung oleh suatu jalur.

Ada $N+1$ warna, dinomori dari 0 hingga N. Warna N merupakan warna spesial yang dinamakan **warna Sfinks**. Setiap verteks diberi warna. Lebih spesifiknya, verteks i ($0 \leq i < N$) berwarna $C[i]$. Beberapa verteks bisa saja berwarna sama, dan mungkin saja ada warna yang tidak muncul pada verteks mana pun. Tidak ada verteks dengan warna Sfinks, yang berarti, $0 \leq C[i] < N$ ($0 \leq i < N$).

Jalur $v_0, v_1, \ldots, v_k$ (untuk $k \geq 0$) dikatakan **monokromatik** jika semua verteksnya berwarna sama, yakni ketika $C[v_l] = C[v_{l+1}]$ (untuk setiap l yang memenuhi $0 \leq l < k$). Selain itu, kita katakan verteks p dan q ($0 \leq p < N$, $0 \leq q < N$) berada di **komponen monokoromatik** yang sama jika dan hanya jika mereka terhubung oleh jalur monokromatik.

Anda tahu verteks dan *edge* dari graf, tetapi Anda tidak tahu warna dari setiap verteks. Anda ingin mencari tahu setiap warna verteks, dengan melakukan beberapa **eksperimen pewarnaan ulang**.

Pada sebuah eksperimen pewarnaan ulang, Anda bisa mewarnai ulang sebanyak mungkin verteks. Lebih spesifiknya, untuk melakukan eksperimen pewarnaan ulang, pertama-tama, pilih *array* E yang berukuran N, sehingga untuk setiap i ($0 \leq i < N$), $E[i]$ bernilai di antara -1 dan N **inklusif**. Kemudian, warna setiap verteks i menjadi $S[i]$, dengan nilai $S[i]$ adalah:

- $C[i]$, yaitu warna awal dari i, jika $E[i] = -1$, atau
- $E[i]$, jika tidak.

Perhatikan bahwa ini berarti Anda bisa menggunakan warna Sfinks dalam pewarnaan ulang Anda.

Pada akhirnya, Sang Sfinks Agung mengumumkan banyaknya komponen monokromatik pada graf, setelah menetapkan warna setiap verteks i menjadi $S[i]$ ($0 \leq i < N$). Pewarnaan barunya hanya berlaku untuk eksperimen pewarnaan ulang ini, sehingga **warna semua verteks kembali ke semula setelah eksperimennya selesai**.

Tugas Anda adalah mengidentifikasi warna semua verteks pada graf dengan melakukan paling banyak $2\,750$ eksperimen pewarnaan ulang. Anda juga bisa mendapatkan nilai parsial jika Anda menentukan dengan benar untuk setiap pasangan verteks bersebelahan, apakah mereka berwarna sama.

Detail Implementasi

Anda harus mengimplementasikan prosedur berikut.

```
std::vector<int> find_colours(int N,
    std::vector<int> X, std::vector<int> Y)
```

- N: banyaknya verteks pada graf.
- X, Y: *array* sepanjang M yang mendeskripsikan *edge*.
- Prosedur ini harus mengembalikan *array* G sepanjang N, yang menyatakan warna verteks pada graf.
- Prosedur ini dipanggil tepat sekali untuk setiap kasus uji.

Prosedur di atas dapat memanggil prosedur berikut untuk melakukan eksperimen pewarnaan ulang:

```
int perform_experiment(std::vector<int> E)
```

- E: *array* sepanjang N yang menjelaskan bagaimana verteks-verteksnya diwarnai ulang.
- Prosedur ini mengembalikan banyaknya komponen monokromatik setelah pewarnaan ulang verteks berdasarkan E.
- Prosedur ini bisa dipanggil paling banyak $2\,750$ kali.

Grader bersifat **tidak adaptif**, yang berarti, warna verteks-verteksnya sudah ditetapkan sebelum pemanggilan `find_colours`.

Batasan

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ untuk setiap j sehingga $0 \leq j < M$.
- $X[j] \neq X[k]$ atau $Y[j] \neq Y[k]$ untuk setiap j dan k sehingga $0 \leq j < k < M$.
- Setiap pasangan verteks terhubung oleh suatu jalur.

- $0 \leq C[i] < N$ untuk setiap i sehingga $0 \leq i < N$.

Subsoal

Subsoal	Nilai	Batasan Tambahan
1	3	$N = 2$
2	7	$N \leq 50$
3	33	Grafnya berbentuk jalur: $M = N - 1$ dan verteks j dan $j + 1$ bersebelahan ($0 \leq j < M$).
4	21	Grafnya lengkap: $M = \frac{N \cdot (N-1)}{2}$ dan dua verteks mana pun bersebelahan.
5	36	Tidak ada batasan tambahan.

Pada setiap subsoal, Anda bisa mendapatkan nilai parsial jika program Anda menentukan dengan benar untuk setiap pasangan verteks bersebelahan apakah mereka berwarna sama.

Lebih formalnya, Anda mendapatkan nilai penuh dari suatu subsoal jika pada semua kasus ujinya, *array* G yang dikembalikan oleh `find_colours` sama persis dengan *array* C (yaitu $G[i] = C[i]$ untuk semua i sehingga $0 \leq i < N$). Jika tidak, Anda mendapatkan 50% dari nilai untuk suatu subsoal jika syarat berikut berlaku untuk semua kasus ujinya:

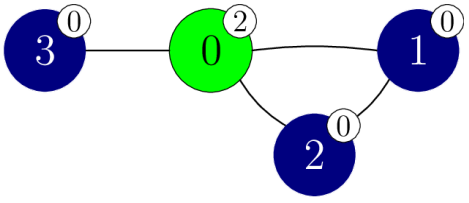
- $0 \leq G[i] < N$ untuk setiap i sehingga $0 \leq i < N$;
- Untuk setiap j sehingga $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ jika dan hanya jika $C[X[j]] = C[Y[j]]$.

Contoh

Perhatikan pemanggilan berikut.

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

Pada contoh ini, misalkan warna (yang belum diketahui) dari verteks adalah $C = [2, 0, 0, 0]$. Skenario ini ditunjukkan oleh ilustrasi berikut. Warna-warna direpresentasikan sebagai angka berlabel putih yang tertaut pada tiap verteks.



Prosedur bisa memanggil `perform_experiment` sebagai berikut.

```
perform_experiment([-1, -1, -1, -1])
```

Pada pemanggilan ini, tidak ada verteks yang diwarnai ulang, dan semua verteks tetap berwarna sama seperti semula.

Perhatikan verteks 1 dan verteks 2. Mereka berwarna 0 dan jalur $1, 2$ adalah jalur monokromatik. Oleh karena itu, verteks 1 dan 2 berada di komponen monokromatik yang sama.

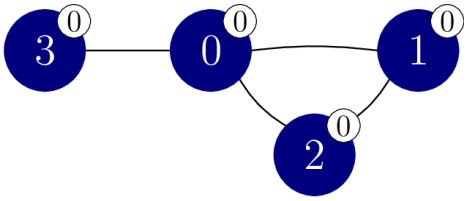
Perhatikan verteks 1 dan verteks 3. Walaupun kedua verteks berwarna 0, mereka berada di komponen monokromatik yang berbeda karena tidak terdapat jalur monokromatik yang menghubungkan mereka.

Secara keseluruhan, terdapat 3 komponen monokromatik, dengan verteks $\{0\}$, $\{1, 2\}$, and $\{3\}$. Maka dari itu, pemanggilan ini mengembalikan 3.

Sekarang, prosedur bisa memanggil `perform_experiment` sebagai berikut.

```
perform_experiment([0, -1, -1, -1])
```

Pada pemanggilan ini, hanya verteks 0 yang diwarnai ulang ke warna 0, yang memberikan pewarnaan seperti yang diilustrasikan sebagai berikut.

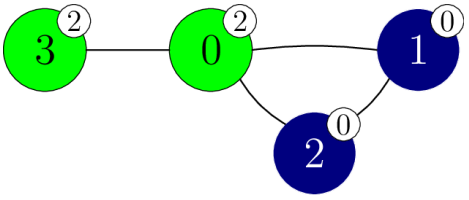


Pemanggilan ini mengembalikan 1, karena semua verteks berada di komponen monokromatik yang sama. Sekarang kita bisa menyimpulkan bahwa verteks 1, 2, dan 3 berwarna 0.

Prosedur bisa kemudian memanggil `perform_experiment` sebagai berikut.

```
perform_experiment([-1, -1, -1, 2])
```

Pada pemanggilan ini, verteks 3 diwarnai ulang ke warna 2, yang memberikan pewarnaan seperti yang diilustrasikan sebagai berikut.



Pemanggilan ini mengembalikan 2 karena terdapat 2 komponen monokromatik, masing-masing dengan verteks $\{0, 3\}$ dan $\{1, 2\}$. Kita bisa menyimpulkan bahwa verteks 0 berwarna 2.

Prosedur `find_colours` kemudian mengembalikan *array* $[2, 0, 0, 0]$. Karena $C = [2, 0, 0, 0]$, nilai penuh pun diberikan.

Perhatikan bahwa terdapat beberapa nilai kembali yang mendapatkan 50% dari nilai yang diberikan, sebagai contoh $[1, 2, 2, 2]$ atau $[1, 2, 2, 3]$.

Contoh *Grader*

Format masukan:

```
N  M
C[0]  C[1] ... C[N-1]
X[0]  Y[0]
X[1]  Y[1]
...
X[M-1]  Y[M-1]
```

Format keluaran:

```
L  Q
G[0]  G[1] ... G[L-1]
```

Di sini, L adalah panjang dari *array* G yang dikembalikan oleh `find_colours`, dan Q adalah banyaknya pemanggilan `perform_experiment`.