Das Rätsel der Sphinx

Die große Sphinx hat ein Rätsel für dich. Sie gibt dir einen Graphen mit N Knoten, nummeriert von 0 bis $N-1$. Der Graph hat M ungerichtete Kanten, nummeriert von 0 bis $M-1$. Jede Kante verbindet zwei unterschiedliche Knoten. Genauer, für jedes j von 0 bis $M-1$ (inklusive) verbindet Kante j die Knoten $X[j]$ und $Y[j]$. Es gibt höchstens eine Kante zwischen je zwei Knoten. Zwei Knoten heißen **benachbart**, wenn sie durch eine Kante verbunden sind.

Eine Folge von Knoten $v_0, v_1, \ldots, v_k$ (für $k \geq 0$) heißt **Pfad**, wenn zwei aufeinanderfolgende Knoten v_l und v_{l+1} (für jedes l mit $0 \leq l < k$) benachbart sind. Wir sagen, dass ein Pfad $v_0, v_1, \ldots, v_k$ die Knoten v_0 und v_k **verbindet**. Im gegebenen Graphen ist jedes Paar von Knoten durch einen Pfad verbunden.

Es gibt $N+1$ Farben, nummeriert von 0 bis N. Die Farbe N ist besonders und heißt **die Farbe der Sphinx**. Jedem Knoten wird eine Farbe zugewiesen: Der Knoten i ($0 \leq i < N$) hat die Farbe $C[i]$. Mehrere Knoten können die gleiche Farbe haben, und es kann Farben geben, die keinem Knoten zugewiesen sind. Kein Knoten hat die Farbe der Sphinx, das heißt, $0 \leq C[i] < N$ ($0 \leq i < N$).

Ein Pfad $v_0, v_1, \ldots, v_k$ (mit $k \geq 0$) heißt **einfarbig**, wenn alle seine Knoten die gleiche Farbe haben, das heißt $C[v_l] = C[v_{l+1}]$ für alle l mit $0 \leq l < k$. Außerdem sagen wir, dass die Knoten p und q ($0 \leq p < N$, $0 \leq q < N$) zur selben **einfarbigen Komponente** gehören, wenn und nur wenn sie durch einen einfarbigen Pfad verbunden sind.

Du kennst die Knoten und Kanten, aber du weißt nicht, welche Farbe die Knoten haben. Also willst du die Farben der Knoten herausfinden, indem du **Umfärbeexperimente** durchführst.

In einem Umfärbeexperiment darfst du beliebig viele Knoten umfärben. Um ein Umfärbeexperiment durchzuführen, wählst du zuerst ein Array E der Länge N, wobei für jedes i ($0 \leq i < N$) $E[i]$ zwischen -1 und N **inklusive** liegt. Dann wird die Farbe eines jeden Knoten i auf $S[i]$ geändert, wobei $S[i]$ wie folgt definiert ist:

- $C[i]$, also die ursprüngliche Farbe von i, wenn $E[i] = -1$, oder
- $E[i]$ andernfalls.

Das bedeutet, dass du auch die Farbe der Sphinx in deiner Umfärbung verwenden kannst.

Am Ende des Experiments verkündet die große Sphinx die Anzahl der einfarbigen Komponenten im Graphen, nachdem die Farbe jedes Knotens i auf $S[i]$ ($0 \leq i < N$) geändert wurde. Die neue

Färbung wird nur für dieses Umfärbeexperiment angewendet, dementsprechend werden **die Farben nach dem Experiment wieder auf den Originalzustand zurückgesetzt.**

Deine Aufgabe ist es, die Farben der Knoten des Graphen herauszufinden. Hierfür darfst du bis zu $2\,750$ Umfärbeexperimente durchführen. Du bekommst Teilpunkte, wenn du für jedes Paar benachbarter Knoten korrekt bestimmst, ob sie die gleiche Farbe haben.

Angaben zur Implementierung

Du sollst die folgende Funktion implementieren.

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

- N ist die Anzahl an Knoten im Graph.
- X und Y sind Arrays der Länge M, die die Kanten beschreiben.
- Diese Funktion soll ein Array G der Länge N zurückgeben, das die Farben der Knoten im Graph enthält.
- Diese Funktion wird in jedem Testfall genau einmal aufgerufen.

Die obige Funktion kann folgende Funktion aufrufen, um Umfärbeexperimente durchzuführen.

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

- E ist ein Array der Länge N, das angibt, wie die Knoten umgefärbt werden sollen.
- Diese Funktion gibt die Anzahl der einfarbigen Komponenten, nachdem die Knoten wie durch E angegeben umgefärbt wurden, zurück.
- Diese Funktion kann bis zu $2\,750$ Mal aufgerufen werden.

Der Grader ist **nicht adaptiv**, das bedeutet, dass die Farben der Knoten feststehen, bevor `find_colours` aufgerufen wird.

Beschränkungen

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ für alle j mit $0 \leq j < M$.
- $X[j] \neq X[k]$ oder $Y[j] \neq Y[k]$ für alle j und k mit $0 \leq j < k < M$.
- Jedes Paar von Knoten ist durch mindestens einen Pfad verbunden.
- $0 \leq C[i] < N$ für alle i mit $0 \leq i < N$.

Subtasks

Subtask	Punkte	Zusätzliche Beschränkungen
1	3	$N = 2$
2	7	$N \leq 50$
3	33	Der Graph ist ein Pfad: $M = N - 1$ und Knoten j und $j + 1$ sind benachbart ($0 \leq j < M$).
4	21	Der Graph ist vollständig: $M = \frac{N \cdot (N-1)}{2}$ und jedes Paar von Knoten ist benachbart.
5	36	Keine weiteren Beschränkungen.

In jedem Subtask kannst du Teilpunkte erhalten, wenn dein Programm für jedes Paar benachbarter Knoten korrekt bestimmt, ob sie die gleiche Farbe haben.

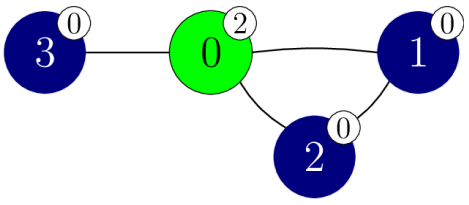
Genauer gesagt erhältst du für einen Subtask alle Punkte, wenn in allen seinen Testfällen das durch `find_colours` zurückgegebene Array G mit dem Array C übereinstimmt (also $G[i] = C[i]$ für alle i mit $0 \leq i < N$). Andernfalls erhältst du für einen Subtask 50% der Punkte, wenn in allen seinen Testfällen das von dir zurückgegebene Array G folgende Bedingungen erfüllt:

- $0 \leq G[i] < N$ für alle i mit $0 \leq i < N$;
- Für alle j mit $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ genau dann, wenn $C[X[j]] = C[Y[j]]$.

Beispiel

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

Der Graph, der durch diesen Aufruf angegeben wird, ist im Bild unten zu sehen. Die (unbekannten) Farben der Knoten sind $C = [2, 0, 0, 0]$, wie in den kleinen weißen Kreisen angegeben.



Nun wird `perform_experiment` einige Male aufgerufen:

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

In diesem Umfärbeexperiment behalten alle Knoten ihre Farben.

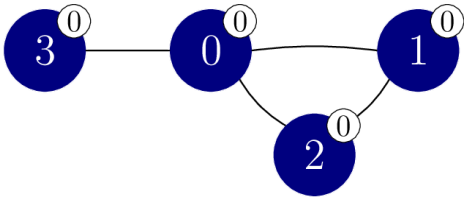
Knoten 1 und 2 haben beide die Farbe 0, und $1, 2$ ist ein einfarbiger Pfad. Die Knoten 1 und 2 gehören also zur selben einfarbigen Komponente.

Knoten 1 und 3 wiederum gehören zu unterschiedlichen einfarbigen Komponenten. Sie haben zwar beide die Farbe 0, sind aber nicht durch einen einfarbigen Pfad verbunden.

Insgesamt gibt es 3 einfarbige Komponenten mit den Knoten $\{0\}$, $\{1, 2\}$, and $\{3\}$. Dieser Aufruf von `perform_experiment` gibt also 3 zurück.

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

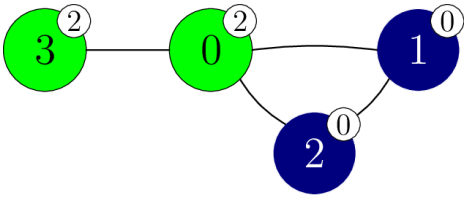
In diesem Experiment wird nur die Farbe von Knoten 0 geändert, nämlich auf 0. Insgesamt ergibt sich diese Färbung:



Dieser Aufruf gibt 1 zurück, denn alle Knoten gehören zur selben einfarbigen Komponente. Daraus lässt sich schließen, dass auch die Knoten 1, 2 und 3 die Farbe 0 haben.

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

In diesem Aufruf wird die Farbe von Knoten 3 auf 2 geändert. Insgesamt ergibt sich diese Färbung:



Dieser Aufruf gibt 2 zurück, denn es gibt 2 einfarbige Komponenten: $\{0, 3\}$ und $\{1, 2\}$.

Die Funktion `find_colours` gibt nun das Array $[2, 0, 0, 0]$ zurück. Dafür gibt es die volle Punktzahl, denn $C = [2, 0, 0, 0]$.

Für einige andere Rückgabewerte gäbe es 50% der Punkte, zum Beispiel für $[1, 2, 2, 2]$ oder $[1, 2, 2, 3]$.

Beispielgrader

Eingabeformat:

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

Ausgabeformat:

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

Hier ist L die Länge des Arrays G, das `find_colours` zurückgibt. Q ist die Anzahl der Aufrufe von `perform_experiment`.