El Acertijo de la Esfinge

La Gran Esfinge tiene un acertijo para ti. Se te da un grafo(graph) con N vértices. Los vértices están numerados del 0 al $N-1$. Hay M aristas(edges) en el grafo(graph) numeradas de 0 a $M-1$. Cada arista(edge) conecta un par de vértices distintos y es bidireccional. Específicamente, para cada j de 0 a $M-1$ (inclusive), la arista j conecta los vértices $X[j]$ y $Y[j]$ son adyacentes. Hay como máximo una arista(edge) conectando cualquier par de vértices. Dos vertices son llamados **adyacente**(**adjacent**) si estan conectados por una arista

Una secuencia de vértices $v_0, v_1, \ldots, v_k$ (for $k \geq 0$) se llama un **camino**(path) si cada dos vértices consecutivos v_l y v_{l+1} (para cada l tal que $0 \leq l < k$) son adyacentes. Decimos que un camino $v_0, v_1, \ldots, v_k$ **conecta** los vértices v_0 y v_k. En el grafo que se te da, cada par de vértices está conectado por algún camino.

Hay $N+1$ colores, numerados del 0 al N. El color N es especial y se llama el **color de la Esfinge**. Cada vértice está asignado a un color. Específicamente, el vértice i ($0 \leq i < N$) tiene el color $C[i]$. Múltiples vértices pueden tener el mismo color, y puede haber colores que no estén asignados a ningún vértice. Ningún vértice tiene el color de la Esfinge, es decir, $0 \leq C[i] < N$ ($0 \leq i < N$).

Un camino $v_0, v_1, \ldots, v_k$ (for $k \geq 0$) se llama **monocromático** si todos sus vértices tienen el mismo color, es decir, $C[v_l] = C[v_{l+1}]$ (for each l such that $0 \leq l < k$). Adicionalmente, decimos que los vértices p y q ($0 \leq p < N$, $0 \leq q < N$) están en el mismo **componente monocromático** si y solo si están conectados por un camino monocromático.

Conoces los vértices y las aristas, pero no sabes qué color tiene cada vértice. Quieres descubrir los colores de los vértices realizando **experimentos de recoloreo**.

En un experimento de recoloreo, puedes recolorear arbitrariamente muchos vértices. Específicamente, para realizar un experimento de recoloreo, primero eliges un arreglo E de tamaño N, donde para cada i ($0 \leq i < N$), $E[i]$ está entre -1 y N **inclusive**. Luego, el color de cada vértice i se convierte en $S[i]$, donde el valor de $S[i]$ es:

- $C[i]$, es decir, el color original de i, si $E[i] = -1$, o
- $E[i]$, de lo contrario.

Ten en cuenta que esto significa que puedes usar el color de la Esfinge en tu recoloreo.

Finalmente, la Gran Esfinge anuncia el número de componentes monocromáticas en el grafo, después de establecer el color de cada vértice i to $S[i]$ ($0 \leq i < N$). El nuevo recoloreo se aplica

solo para este experimento en particular, por lo que **los colores de todos los vértices vuelven a los originales después de que el experimento termine**.

Tu tarea es identificar los colores de los vértices en el grafo realizando como máximo $2\,750$ experimentos de recoloreo. También puedes obtener una puntuación parcial si determinas correctamente, para cada par de vértices adyacentes, si tienen el mismo color.

Detalles de Implementación

Debes implementar el siguiente procedimiento:

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

- N: el número de vértices en el grafo.
- X, Y: arrays de longitud M que describen las aristas(edges).
- Este procedimiento debe devolver un array G de longitud N, representando los colores de los vértices en el grafo.
- Este procedimiento se llama exactamente una vez por cada caso de prueba.

El procedimiento anterior puede hacer llamadas al siguiente procedimiento para realizar experimentos de recoloreo:

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

- E: un array de longitud N que especifica cómo deben recolorearse los vértices.
- Este procedimiento devuelve el número de componentes monocromáticas después de recolorear los vértices según E.
- Este procedimiento puede ser llamado como máximo $2\,750$ veces.

El calificador **no es adaptativo**, es decir, los colores de los vértices están fijos antes de que se llame a `find_colours`.

Restricciones

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ for each j such that $0 \leq j < M$.
- $X[j] \neq X[k]$ or $Y[j] \neq Y[k]$ for each j and k such that $0 \leq j < k < M$.
- Cada par de vértices está conectado por algún camino.
- $0 \leq C[i] < N$ for each i such that $0 \leq i < N$.

Subtareas

Subtarea	Puntuación	Restricciones adicionales
1	3	$N = 2$
2	7	$N \leq 50$
3	33	El grafo es un camino: $M = N - 1$ y los vértices j y $j + 1$ son adyacentes ($0 \leq j < M$).
4	21	El grafo es completo: $M = \frac{N \cdot (N-1)}{2}$ y cualquier par de vértices son adyacentes.
5	36	No hay restricciones adicionales.

En cada subtarea, puedes obtener una puntuación parcial si tu programa determina correctamente para cada par de vértices adyacentes si tienen el mismo color.

Más precisamente, obtienes la puntuación completa de una subtarea si en todos sus casos de prueba, el array G devuelto por `find_colours` es exactamente el mismo que el arreglo C (es decir, $G[i] = C[i]$ para todos los i tal que $0 \leq i < N$). De lo contrario, obtienes el 50% de la puntuación de una subtarea si se cumplen las siguientes condiciones en todos sus casos de prueba:

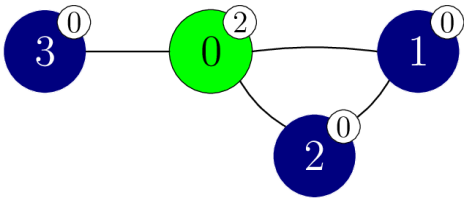
- $0 \leq G[i] < N$ for each i such that $0 \leq i < N$;
- For each j such that $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ if and only if $C[X[j]] = C[Y[j]]$.

Ejemplo

Considera la siguiente llamada.

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

Para este ejemplo, supongamos que los (ocultos) colores de los vértices están dados por $C = [2, 0, 0, 0]$. Este escenario se muestra en la siguiente figura. Los colores se representan adicionalmente con números en etiquetas blancas adheridas a cada vértice.



El procedimiento puede llamar a `perform_experiment` de la siguiente manera.

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

En esta llamada, ningún vértice se recolorea, ya que todos los vértices mantienen sus colores originales.

Considera el vértice 1 y el vértice 2. Ambos tienen el color 0 y el camino $1, 2$ es un camino monocromático. Como resultado, los vértices 1 y 2 están en el mismo componente monocromático.

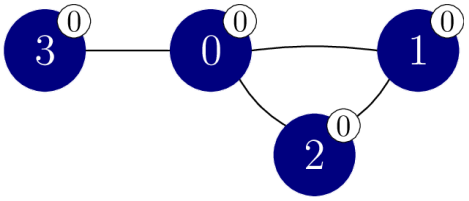
Considera el vértice 1 y el vértice 3. Aunque ambos tienen el color 0, están en diferentes componentes monocromáticos ya que no hay un camino monocromático que los conecte.

En total, hay 3 componentes monocromáticos, con los vértices $\{0\}$, $\{1, 2\}$, and $\{3\}$. Por lo tanto, esta llamada devuelve 3.

Ahora el procedimiento puede llamar a `perform_experiment` de la siguiente manera.

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

En esta llamada, solo el vértice 0 se recolorea al color 0, lo que resulta en la coloración mostrada en la siguiente figura.

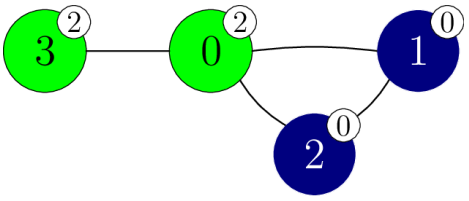


Esta llamada devuelve 1, ya que todos los vértices pertenecen a la misma componente monocromática. Ahora podemos deducir que los vértices 1, 2 y 3 tienen el color 0.

El procedimiento puede luego llamar a `perform_experiment` de la siguiente manera.

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

En esta llamada, el vértice 3 se recolorea al color 2, lo que resulta en la coloración mostrada en la siguiente figura.



Esta llamada devuelve 2, ya que hay 2 componentes monocromáticas, con los vértices $\{0, 3\}$ and $\{1, 2\}$ respectivamente. Podemos deducir que el vértice 0 tiene el color 2.

El procedimiento `find_colours` luego devuelve el array $[2, 0, 0, 0]$. Dado que $C = [2, 0, 0, 0]$, se otorga la puntuación completa.

Toma en cuenta que también hay múltiples valores de retorno, para los cuales se otorgaría el 50% de la puntuación, por ejemplo $[1, 2, 2, 2]$ o $[1, 2, 2, 3]$.

Sample Grader

Input format:

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

Output format:

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

Aqui, L es la longitud(length) del array G retornado(returned) by `find_colours`, y Q es el número de llamadas `perform_experiment`.