Adivinanza de la Esfinge

La Gran Esfinge tiene una adivinanza para ti. Tienes un grafo de N vértices. Los vértices están numerados desde 0 hasta $N-1$. Hay M aristas en el grafo, numeradas desde 0 hasta $M-1$. Las aristas son bidireccionales y conectan un par de vértices distintos. Específicamente, para cada j desde 0 hasta $M-1$ (inclusive) la arista j conecta los vértices $X[j]$ y $Y[j]$. Hay a lo más una arista conectando cualquier par de vértices. Dos vértices son **adyacentes** si están conectados por una arista.

Una secuencia de vértices $v_0, v_1, \ldots, v_k$ (para $k \geq 0$) es un **camino** si para cada par de 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 dado, todo par de vértices están conectado por algún camino.

Hay $N+1$ colores, numerados desde 0 hasta N. El color N es especial y es conocido como el **color de la Esfinge**. Cada vértice tiene un color asignado. Específicamente, el vértice i ($0 \leq i < N$) tiene color $C[i]$. Varios vértices pueden tener el mismo color, y pueden haber colores que no fueron 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 $v_0, v_1, \ldots, v_k$ (para $k \geq 0$) es **monocromático** si todos los vértices que pertenecen a él tienen el mismo color, es decir, $C[v_l] = C[v_{l+1}]$ (para cada l tal que $0 \leq l < k$). Adicionalmente, decimos que los vértices p y q ($0 \leq p < N$, $0 \leq q < N$) están en la misma **componente monocromática** si y solo si están conectados por un camino monocromático.

Conoces los vértices y las aristas, pero no sabes que color tiene cada vértice. Quieres encontrar el color de los vértices realizando **experimentos de recoloración**.

En un experimento de recoloración, puedes cambiar arbitrariamente el color de varios vértices. Específicamente, para realizar un experimento de recoloración primero eliges un arreglo E de tamaño N, donde para cada i ($0 \leq i < N$), $E[i]$ es un entero entre -1 and N **inclusive**.

Luego, el color de cada vértice i pasa a ser $S[i]$, donde el valor de $S[i]$ es:

- $C[i]$, es decir, el color original de i, si $E[i] = -1$, o
- $E[i]$, en otro caso.

Notar que puedes usar el color de la Esfinge en tu recoloración.

Finalmente, la Gran Esfinge anuncia el número de componentes monocromáticas que hay en el grafo después de asignar el color $S[i]$ a cada vértice i ($0 \leq i < N$). El nuevo coloreado es aplicado solo para este experimento en particular, de esta forma **los colores de todos los vértices vuelven a su color original después de que el experimento de recoloración finaliza**.

Tu tarea es identificar los colores de los vértices en el grafo realizando a lo más $2\,750$ experimentos de recoloración. Puedes recibir puntuación parcial si determinas correctamente para todo par de vértices adyacentes si tienen o no el mismo color.

Detalles de Implementación

Debes implementar la siguiente función.

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

- N: el número de vértices del grafo.
- X, Y: arreglos de tamaño M que describen las aristas.
- Esta función debe retornar un arreglo G de tamaño N que representa los colores de los vértices en el grafo.
- Esta función es llamada exactamente una vez por cada caso de prueba.

La función anterior puede realizar llamadas a la siguiente función para realizar los experimentos de recoloración:

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

- E: un arreglo de longitud N especificando cómo los vértices deben ser recoloreados.
- Esta función retorna el número de componentes monocromáticas después de recolorear los vértices de acuerdo a E.
- Esta función puede ser llamada a lo más $2\,750$ veces.

El evaluador **no es adaptativo**, es decir, los colores de los vértices son fijados antes de realizar una llamada a `find_colours`.

Restricciones

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ para cada j tal que $0 \leq j < M$.
- $X[j] \neq X[k]$ o $Y[j] \neq Y[k]$ para cada j y k tal que $0 \leq j < k < M$.
- Cada par de vértices están conectados por algún camino.
- $0 \leq C[i] < N$ para cada i tal que $0 \leq i < N$.

Subtareas

Subtareas	Puntos	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	Sin restricciones adicionales.

En cada subtarea, puedes obtener puntuación parcial si tu programa determina correctamente para todo par de vértices adyacentes si tienen o no el mismo color.

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

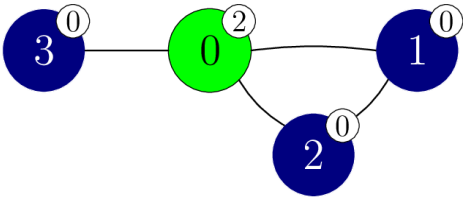
- $0 \leq G[i] < N$ para cada i tal que $0 \leq i < N$;
- Para cada j tal que $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ si y solo si $C[X[j]] = C[Y[j]]$.

Ejemplo

Considera la siguiente llamada.

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

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



La función puede llamar a `perform_experiment` de la siguiente forma.

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

En esta llamada, ningún vértice es recoloreado, así que todos los vértices mantienen su color original.

Considera los vértices 1 y 2. Ambos tiene color 0 y el camino $1, 2$ es un camino monocromático. Como resultado, los vértices 1 y 2 están en la misma componente monocromática.

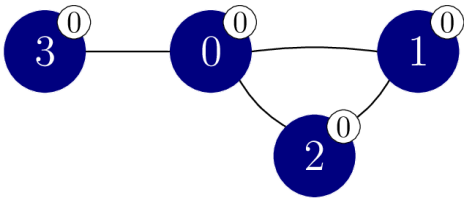
Considera los vértices 1 y 3. Aunque ambos tienen el color 0, están en diferentes componentes monocromáticas ya que no hay ningún camino monocromático conectándolos.

En general, hay 3 componentes monocromáticas, con los vértices $\{0\}$, $\{1, 2\}$, y $\{3\}$. Por eso, esta llamada retorna 3.

Ahora la función puede llamar a `perform_experiment` de la siguiente forma.

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

En esta llamanda, solo el vértice 0 es recoloreado al color 0, que resulta en el coloreado mostrado en la siguiente figura.

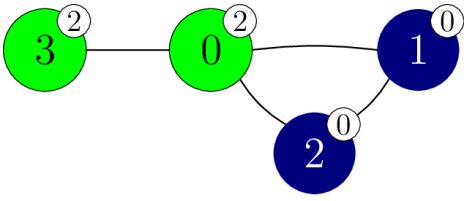


Esta llamada retorna 1, porque todos los vértices pertenecen a la misma componente monocromática. Ahora podemos deducir que los vértices 1, 2, y 3 tienen color 0.

Luego la función puede llamar a `perform_experiment` de la siguiente forma.

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

En esta llamada, el vértice 3 es recoloreado al color 2, que resulta en el coloreado de la siguiente figura.



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

La función `find_colours` luego retorna el arreglo $[2, 0, 0, 0]$. Como $C = [2, 0, 0, 0]$, se obtiene puntuación completa.

Nota además que pueden haber varios valores de retorno, para obtener el 50% de los puntos, por ejemplo $[1, 2, 2, 2]$ o $[1, 2, 2, 3]$.

Evaluador de prueba

Formato de entrada:

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

Formato de salida:

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

Aquí, L es el tamaño del arreglo G retornado por `find_colours`, y Q es el número de llamadas a `perform_experiment`.