El Acertijo de la Esfinge

La Gran Esfinge tiene un acertijo para ti. Te da un grafo de N vértices, enumerados del 0 al $N-1$. Existen M aristas en el grafo, enumeradas de 0 a $M-1$. Cada arista es bidireccional y conecta un par de vértices distintos. Específicamente, para cada j de 0 a $M-1$ (inclusive) la arista j conecta a los vértices $X[j]$ e $Y[j]$. Existe a lo más una arista conectando a cualquier par de vértices. Dos vértices son llamados **adyacentes** si están conectados por una arista.

Una secuencia de vértices $v_0, v_1, \ldots, v_k$ (para $k \geq 0$) es llamada un **camino** 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** a los vértices v_0 y v_k. En el grafo que se te entrega, cada par de vértices está conectado por algún camino.

Existen $N+1$ colores, enumerados del 0 al N. El color N es especial y es llamado el **color de la Esfinge**. Se asigna un color a cada vértice. Específicamente, el vértice i ($0 \leq i < N$) tiene el color $C[i]$. Múltiples vértices pueden tener el mismo color, y pueden existir colores que no son asignados a algún 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 llamado **monocromático** si todos sus vértices 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.

Tú conoces los vértices y las aristas, pero no sabes qué color tiene cada vértice. Quieres encontrar los colores de los vértices realizando **experimentos de recoloración**.

En un experimento de recoloración, puedes cambiar arbitrariamente el color de muchos 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 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.

Nota que esto significa que puedes usar el color de la Esfinge en tu recoloración.

Finalmente, la Gran Esfinge anuncia el número de componentes monocromáticas en el grafo después de asignar $S[i]$ al color de cada vértice i ($0 \leq i < N$). La nueva coloración es aplicada

únicamente para este experimento en particular, por lo que **los colores de todos los vértices vuelven a ser los colores originales después que termina el experimento de recoloración**.

Tu tarea es identificar los colores de los vértices en un grafo realizando a lo más $2\,750$ experimentos de recoloración. También puedes recibir un puntaje parcial si determinas para cada 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 en el grafo.
- X, Y: arreglos de largo M representando las aristas.
- Esta función debe retornar un arreglo G de tamaño N, representando los colores de los vértices en el grafo.
- Esta función es llamada exactamente una vez por cada caso de prueba.

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

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

- E: un arreglo de tamaño N especificando cómo se deben recolorear los vértices.
- 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 cualquier 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 tales que $0 \leq j < k < M$.
- Cada par de vértices está conectado por algún camino.
- $0 \leq C[i] < N$ para cada i tal que $0 \leq i < N$.

Subtareas

Subtarea	Puntaje	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 todos los vértices son adyacentes.
5	36	Sin restricciones adicionales.

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

Concretamente, obtienes el puntaje completo para una subtarea si, en todos los casos de prueba, el arreglo G retornado por `find_colours` es exactamente el mismo arreglo C (es decir $G[i] = C[i]$ para cada i tal que $0 \leq i < N$).

De lo contrario, obtienes 50% del puntaje de una subtarea si se cumplen las siguientes condiciones en todos sus casos de prueba:

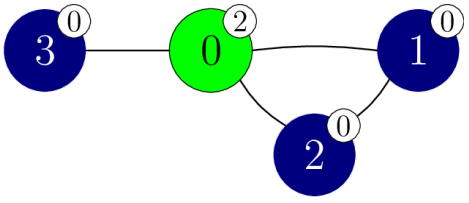
- $0 \leq G[i] < N$ para cada i tal que $0 \leq i < N$;
- Para cada j tal que $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ si y solo si $C[X[j]] = C[Y[j]]$.

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 están dados por $C = [2, 0, 0, 0]$. Este escenario se muestra en la siguiente figura. Los colores están representados adicionalmente por números en etiquetas blancas adjuntas a cada vértice.



La función puede llamar a `perform_experiment` de la siguiente manera:

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

En esta llamada, no se recolorea a ningún vértice, y todos los vértices mantienen su color original.

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

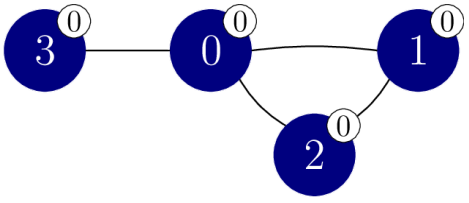
Considera los vértices 1 y 3. Aunque tienen el mismo color 0, están en componentes monocromáticas distintas, ya que no existe un camino monocromático entre ellos.

En general, existen 3 componentes monocromáticas, con vértices $\{0\}$, $\{1, 2\}$, y $\{3\}$. Entonces, esta llamada retorna 3.

Ahora la función puede llamar a `perform_experiment` de la siguiente manera:

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

En esta llamada, únicamente el vértice 0 es recoloreado al color 0, que resulta en la coloración mostrada en la siguiente figura:

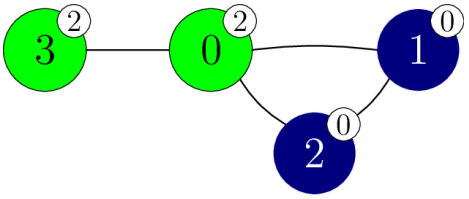


Esta llamada retorna 1, ya que todos los vértices pertenecen a la misma componente monocromática. Podemos entonces deducir que los vértices 1, 2 y 3 tienen el color 0.

La función puede entonces llamar a `perform_experiment` como sigue:

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

En esta llamada, el vértice 3 es recoloreado al color 2, que resulta en la coloración mostrada en la siguiente figura:



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

La función `find_colours` entonces retorna el arreglo $[2, 0, 0, 0]$. Dado que $C = [2, 0, 0, 0]$, se obtiene el puntaje completo.

Nota que existen múltiples valores de retorno para los cuales se otorga el 50% del puntaje. Por ejemplo, $[1, 2, 2, 2]$ o $[1, 2, 2, 3]$.

Evaluador de Ejemplo

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