Sphinx's Riddle

La Gran Esfinge tiene un acertijo para ti. Tienes un grafo con N vértices. Los vértices están enumerados de 0 a $N-1$. Hay M aristas en el grafo, enumeradas de 0 a $M-1$. Cada arista conecta a un par de vértices distintos y son bidireccionales. Específicamente, para todo j desde 0 hasta $M-1$ inclusive, la arista j conecta a los vértices $X[j]$ y $Y[j]$. Hay a lo sumo una arista conectando 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 todo 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 dado cada par de vértices está conectado por algún camino.

Hay $N+1$ colores enumerados de 0 a N. El color N es especial y es llamado **el color de la esfinge**. A cada vértice se le es asignado un color. Específicamente el vértice i ($0 \leq i < N$) tiene color $C[i]$. Múltiples vértices pueden tener el mismo color, y puede que hayan colores que no sean asignados a ningún vértice. Ningún vértice tiene el color de la esfinge, esto es, $0 \leq C[i] < N$ ($0 \leq i < N$).

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 todo 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 colores tienen los vértices. Quieres averiguar los colores de los vértices realizando **experimentos de recoloreo**.

En un experimento de recoloreo puedes recolorear arbitrariamente varios vértices. Específicamente para realizar un experimento de coloración primero escoges un arreglo E de tamaño N donde para todo i ($0 \leq i < N$), el color del vértice i se vuelve $S[i]$, donde el valor $S[i]$ es:

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

Nota que esto significa que puedes utilizar el color de la esfinge en tu recoloreo.

Al final la Gran Esfinge anuncia el número de componentes monocromáticas en el grafo luego de haber coloreado cada vértice i con el color $S[i]$ ($0 \leq i < N$). El nuevo coloreo es aplicado sólo para este experimento de coloreo, así que **los colores de los vértices vuelven a ser los originales después de que el experimento termina**.

Tu tarea es identificar los colores de los vértices en el grafo realizando a lo sumo $2\,750$ experimentos de recoloreo. También puedes recibir puntaje parcial si determinas correctamente para cada par de vértices adyacentes si tienen 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 longitud M que describen las aristas.
- Esta función debe retornar un arreglo G de longitud 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 de arriba realiza llamadas a la siguiente función para realizar los experimentos de recoloreo:

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

- E: un arreglo de longitud N que especifica como deben ser los vértices 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 sumo $2\,750$ veces.

El evaluador **no es adaptativo**, es decir, los colores de los vértices son establecidos antes de que la llamada a `find_colours` sea realizada.

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 todo j tal que $0 \leq j < M$.
- $X[j] \neq X[k]$ o $Y[j] \neq Y[k]$ para todo 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 todo 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 cualquier par de vértices son adyacentes.
5	36	Sin restricciones adicionales.

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

De manera más precisa, obtendrás el puntaje completo de una subtarea si en todos sus casos de prueba el arreglo G retornado por `find_colours` es exactamente el mismo arreglo C (es decir, $G[i] = C[i]$ para todo i tal que $0 \leq i < N$). De lo contrario, obtendrás el 50% del puntaje de una subtarea si las siguientes condiciones se cumplen en todos sus casos de prueba:

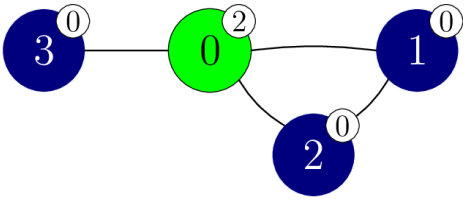
- $0 \leq G[i] < N$ para todo i tal que $0 \leq i < N$;
- Para todo j tal que $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ si y solo si $C[X[j]] = C[Y[j]]$.

Ejemplo

Considera la siguiente llamada.

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

Para este ejemplo, supón que los colores (escondidos) de los vértices son dados por $C = [2, 0, 0, 0]$. Este escenario es mostrado en la siguiente imagen. Los colores son representados adicionalmente por números en etiquetas blancas añadidas a cada vértice.



La función podría llamar a `perform_experiment` como sigue.

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

En esta llamada ningún vértice es recoloreado, por lo que cada vértice conserva su color original.

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

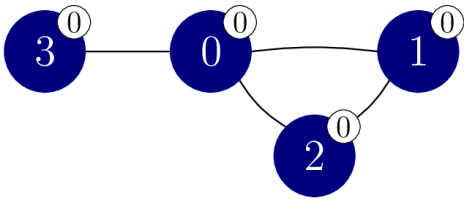
Ahora considera los vértices 1 y 3. Aunque ambos tienen el mismo color 0, ellos están en componentes monocromáticas diferentes dado que no hay un camino monocromático que los conecte.

Finalmente hay 3 componentes monocromáticas con los vértices $\{0\}$, $\{1, 2\}$, y $\{3\}$. Por lo tanto la función retorna 3.

Ahora la función podría llamar a `perform_experiment` como sigue.

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

En esta llamada sólo el vértice 0 es recoloreado de color 0, lo que resulta en el coloreo mostrado en la siguiente imagen.

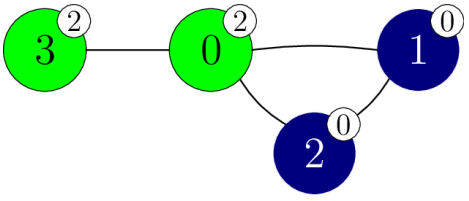


Esta llamada retorna 1 dado que todos los vértices pertenecen a la misma componente monocromática.

Esta función podría llamar después a `perform_experiment` como sigue.

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

En esta llamada el vértice 3 es recoloreado de color 2, dando como resultado el coloreo mostrado en la siguiente imagen.



Esta llamada retorna 2 dado 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` retornaría el arreglo $[2, 0, 0, 0]$. Puesto que $C = [2, 0, 0, 0]$, el puntaje completo es dado.

Nota que hay múltiples valores de retorno posibles para los cuales el 50% del puntaje sería dado, 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 la longitud del arreglo G retornado por `find_colours` y Q es el número de llamadas a `perform_experiment`.