Sphinx's Riddle

A Grande Esfinge tem um desafio para ti. É-te dado um grafo de N vértices. Os vértices estão numerados de 0 a $N-1$. Há M arestas no grafo, numerados de 0 a $M-1$. Cada aresta conecta um par de vértices distintos e é bidirecional. Especificamente, para cada j de 0 a $M-1$ (inclusive), a aresta j conecta os vértices $X[j]$ e $Y[j]$. Há no máximo uma aresta a conectar qualquer par de vértices. Dois vértices são **adjacentes** se etsão conectados por uma aresta.

Uma sequência de vértices $v_0, v_1, \ldots, v_k$ (for $k \geq 0$) é um **caminho** se cada dois vértices consecutivos v_l e v_{l+1} (para cada l tal que $0 \leq l < k$) são adjacentes. Dizemos que um caminho $v_0, v_1, \ldots, v_k$ **conecta** os vértices v_0 e v_k. No grafo que te é dado, cada par de vértices é conectado por algum caminho.

Existem $N+1$ cores, numeradas de 0 a N. A cor N é especial e é chamada de **cor da Esfinge**. A cada vértice é atribuída uma cor. Especificamente, o vértice i ($0 \leq i < N$) tem cor $C[i]$. Múltiplos vértices podem ter a mesma cor, e podem haver cores não atribuídas a nenhum vértice. Nenhum vértice tem a cor da Esfinge, isto é, $0 \leq C[i] < N$ ($0 \leq i < N$).

Um caminho $v_0, v_1, \ldots, v_k$ (for $k \geq 0$) é chamado **monocromático** se todos os seus vértices têm a mesma cor, i.e. $C[v_l] = C[v_{l+1}]$ (para cada l tal que $0 \leq l < k$). Adicionalmente, dizemos que os vértices p e q ($0 \leq p < N$, $0 \leq q < N$) estão na mesma **componente monocromática** se e só se são conectados por um caminho monocromático.

Tu sabes os vértices e as arestas, mas não sabes a cor de cada vértice. Queres descobrir as cores dos vértices, através da realização de **experiências de coloração**.

Numa experiência de coloração, podes recolorir quaisquer vértices. Especificamente, para realizar uma experiência de coloração primeiro escolhes um array E de tamanho N, onde para cada i ($0 \leq i < N$), $E[i]$ está entre -1 e N **inclusive**. Depois, a cor de cada vértice i passa a ser $S[i]$, onde o valor de $S[i]$ é:

- $C[i]$, isto é, a cor original de i, se $E[i] = -1$, ou
- $E[i]$, caso contrário.

Nota que isto significa que podes usar a cor da Esfinge na tua coloração.

Finalmente, a Grande Esfinge anuncia o número de componentes monocromáticas no grafo, após pintar cada vértice i da cor $S[i]$ ($0 \leq i < N$). A nova coloração é aplicada apenas para esta experiência de coloração em particular, portanto **a cor de todos os vértices volta a ser a original depois da experiência acabar**.

A tua tarefa é identificar as cores dos vértices do grafo ao realizar no máximo $2\,750$ experiências de coloração. Podes também receber pontuação parcial se determinares corretamente para cada par de vértices adjacentes, se eles têm a mesma cor ou não.

Detalhes de Implementação

Deves implementar a seguinte função.

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

- N: o número de vértices no grafo.
- X, Y: arrays de tamanho M descrevendo as arestas.
- Esta função deve devolver um array G de tamanho N, representado as cores dos vértices do grafo.
- Esta função é chamada exatamente uma vez para cada caso de teste.

A função acima pode fazer chamadas à seguinte função para realizar experiências de coloração:

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

- E: um array de tamanho N especificando como os vértices devem ser recoloridos.
- Esta função devolve o número de componentes monocromáticas após recolorir os vértices de acordo com E.
- Esta função pode ser chamada no máximo $2\,750$ vezes.

O avaliador **não é adaptativo**, isto é, as cores dos vértices são fixas antes de uma chamada a `find_colours` ser feita.

Constraints

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ para cada j tal que $0 \leq j < M$.
- $X[j] \neq X[k]$ ou $Y[j] \neq Y[k]$ para cada j e k tais que $0 \leq j < k < M$.
- Cada par de vértices é conectado por algum caminho.
- $0 \leq C[i] < N$ para cada i tal que $0 \leq i < N$.

Subtarefas

Subtarefa	Pontos	Restrições Adicionais
1	3	$N = 2$
2	7	$N \leq 50$
3	33	O grafo é um caminho: $M = N - 1$ e os vértices j e $j + 1$ são adjacentes ($0 \leq j < M$).
4	21	O grafo é completo: $M = \frac{N \cdot (N-1)}{2}$ e quaisquer dois vértices são adjacentes.
5	36	Sem restrições adicionais.

Em cada subtarefa, podes obter pontuação parcial se o teu programa determinar corretamente para todos os pares de vértices adjacentes, se eles têm a mesma cor ou não.

Mais precisamente, recebes a pontuação total de uma subtarefa se em todos os seus casos de teste, o array G devolvido por `find_colours` é exatamente o mesmo que o array C (i.e. $G[i] = C[i]$ para todos os i tais que $0 \leq i < N$). Caso contrário, recebes 50% da pontuação da subtarefa se as seguintes condições se verificarem em todos os casos de teste:

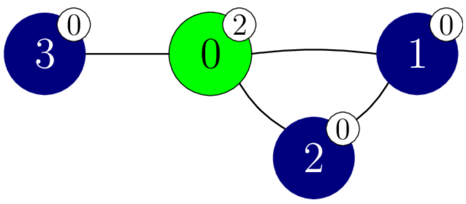
- $0 \leq G[i] < N$ para cada i tal que $0 \leq i < N$;
- Para cada j tal que $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ se e só se $C[X[j]] = C[Y[j]]$.

Exemplo

Considera a seguinte chamada.

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

Para este exemplo, supõe que as cores (escondidas) dos vértices são dadas por $C = [2, 0, 0, 0]$. Este cenário é mostrado na seguinte figura. As cores são adicionalmente representadas por números em etiquetas brancas colocadas em cada vértice.



A função pode chamar `perform_experiment` como se segue.

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

Nesta chamada, nenhum vértice é recolorido, visto que todos os vértices mantém as suas cores originais.

Considera o vértice 1 e o vértice 2. Ambos têm cor 0 e o caminho $1, 2$ é um caminho monocromático. Logo, os vértices 1 e 2 estão na mesma componente monocromática.

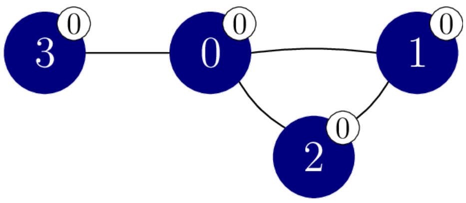
Considera o vértice 1 e o vértice 3. Apesar de ambos terem a cor 0, estão em componentes monocromáticas diferentes pois não há nenhum caminho monocromático a conecta-los.

Ao todo, existem 3 componentes monocromáticas, com vértices $\{0\}$, $\{1, 2\}$, e $\{3\}$. Logo, esta chamada devolve 3.

Agora a função pode chamar `perform_experiment` como se segue.

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

Nesta chamada, apenas o vértice 0 é recolorido para a cor 0, o que resulta na coloração mostrada na seguinte figura.

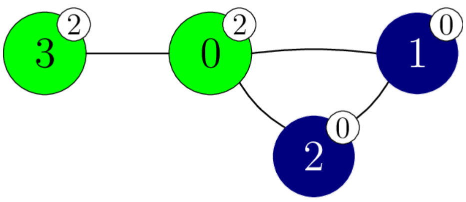


Esta chamada devolve 1, visto que todos os vértices pertencem à mesma componente monocromática. Podemos agora deduzir que os vértices 1, 2, e 3 têm cor 0.

A função pode então chamar `perform_experiment` como se segue.

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

Nesta chamada, o vértice 3 é recolorido para a cor 2, o que resulta na coloração mostrada na seguinte figura.



Esta chamada devolve 2, visto que existem 2 componentes monocromáticas, com vértices $\{0, 3\}$ e $\{1, 2\}$ respetivamente. Podemos deduzir que o vértice 0 tem cor 2.

A função `find_colours` devolve o array $[2, 0, 0, 0]$. Como $C = [2, 0, 0, 0]$, a pontuação total é dada.

Nota que podem existir múltiplos possíveis arrays de resposta para os quais 50% da pontuação seria dada, como por exemplo $[1, 2, 2, 2]$ or $[1, 2, 2, 3]$.

Avaliador Exemplo

Formato de input:

```
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 output:

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

Aqui, L é o tamanho do array G devolvido por `find_colours`, e Q é o número de chamadas a `perform_experiment`.