Sphinx's Riddle

La Grande Sfinge ha un enigma per te. Hai un grafo con N nodi, numerati da 0 a $N-1$, e M archi, numerati da 0 a $M-1$. Ogni arco collega una coppia di nodi distinti ed è bidirezionale. In particolare, per ogni j da 0 a $M-1$ (inclusi) l'arco j collega i nodi $X[j]$ e $Y[j]$. Ogni coppia di nodi è collegata da al massimo un arco. Due nodi sono chiamati **adiacenti** se sono collegati da un arco.

Una sequenza di nodi $v_0, v_1, \ldots, v_k$ (per $k \geq 0$) si chiama **percorso** se per ogni l ($0 \leq l < k$) i nodi v_l e v_{l+1} sono adiacenti. Diciamo che un percorso $v_0, v_1, \ldots, v_k$ **collega** i nodi v_0 e v_k. Nel grafo fornito, ogni coppia di nodi è collegata da un percorso.

Ci sono $N+1$ colori, numerati da 0 a N. Il colore N è speciale e viene chiamato **colore della Sfinge**. A ciascun nodo è assegnato un colore: il nodo i ($0 \leq i < N$) ha colore $C[i]$. Più nodi potrebbero avere lo stesso colore, e potrebbero esserci colori non assegnati ad alcun nodo. Nessun nodo ha il colore della Sfinge, cioè $0 \leq C[i] < N$ ($0 \leq i < N$).

Un percorso $v_0, v_1, \ldots, v_k$ (per $k \geq 0$) è chiamato **monocromatico** se tutti i suoi nodi hanno lo stesso colore, cioè $C[v_l] = C[v_{l+1}]$ (per ogni l tale che $0 \leq l < k$). Inoltre, diciamo che i nodi p, q ($0 \leq p < N$, $0 \leq q < N$) sono nella stessa **componente monocromatica** se e solo se sono collegati da un percorso monocromatico.

Conosci i nodi e gli archi, ma non sai i colori dei nodi. Vuoi scoprire i colori dei nodi, eseguendo **esperimenti di ricolorazione**. In un esperimento di ricolorazione, puoi ricolorare un numero arbitrario di nodi, scegliendo un array E di dimensione N, dove per ogni i ($0 \leq i < N$), $E[i]$ è compreso tra -1 e N **inclusi**. Quindi, il colore di ogni nodo i diventa $S[i]$, dove il valore di $S[i]$ è:

- $C[i]$, cioè il colore originale di i, se $E[i] = -1$;
- $E[i]$, altrimenti.

Tieni presente che questo significa che puoi usare il colore della Sfinge nella tua ricolorazione.

Dopo aver impostato il colore di ciascun nodo i su $S[i]$ ($0 \leq i < N$), la Grande Sfinge annuncia il numero di componenti monocromatiche nel grafo così ottenuto. La nuova colorazione viene applicata solo per questo particolare esperimento di ricolorazione, poi **i colori di tutti i nodi tornano quelli originali una volta terminato l'esperimento**.

Devi identificare i colori dei nodi nel grafo eseguendo al massimo $2\,750$ esperimenti di ricolorazione. Puoi anche ricevere un punteggio parziale se determini correttamente per ogni coppia di nodi adiacenti se hanno lo stesso colore.

Note di implementazione

Devi implementare la seguente funzione.

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

- N: numero di nodi nel grafo.
- X, Y: array di lunghezza M che descrivono gli archi.
- Questa funzione deve restituire un array G di lunghezza N, che rappresenta i colori dei nodi nel grafo.
- Questa funzione viene chiamata esattamente una volta per ogni caso di test.

La funzione sopra descritta può effettuare chiamate alla seguente funzione per eseguire esperimenti di ricolorazione:

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

- E: un array di lunghezza N che specifica come devono essere ricolorati i nodi.
- Questa funzione restituisce il numero di componenti monocromatiche dopo aver ricolorato i nodi secondo E.
- Questa funzione può essere chiamata al massimo $2\,750$ volte.

Il grader **non è adattivo**, cioè i colori dei nodi vengono fissati prima della chiamata a `find_colours`.

Assunzioni

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ per ogni j tale che $0 \leq j < M$.
- $X[j] \neq X[k]$ o $Y[j] \neq Y[k]$ per ogni j e k tali che $0 \leq j < k < M$.
- Ogni coppia di nodi è collegata da un percorso.
- $0 \leq C[i] < N$ per ogni i tale che $0 \leq i < N$.

Subtask

Subtask	Punteggio	Limitazioni aggiuntive
1	3	$N = 2$.
2	7	$N \leq 50$.
3	33	Il grafo è un cammino: $M = N - 1$ e i nodi j e $j + 1$ sono adiacenti ($0 \leq j < M$).
4	21	Il grafo è completo: $M = \frac{N \cdot (N-1)}{2}$ e due nodi qualsiasi sono adiacenti.
5	36	Nessuna limitazione aggiuntiva.

In ogni subtask puoi ottenere un punteggio parziale se il tuo programma determina correttamente per ogni coppia di nodi adiacenti se hanno lo stesso colore.

Più precisamente, ottieni l'intero punteggio di un subtask se in tutti i suoi casi di test, l'array G restituito da `find_colours` è esattamente lo stesso dell'array C (cioè $G[i] = C[i]$ per $0 \leq i < N$). Altrimenti, ottieni il 50% del punteggio per un subtask se sono soddisfatte le seguenti condizioni in tutti i suoi casi di test:

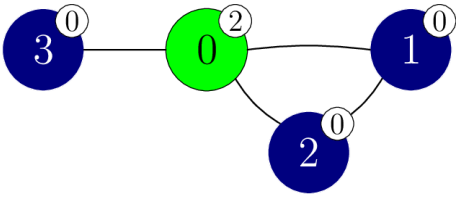
- $0 \leq G[i] < N$ per ogni i tale che $0 \leq i < N$;
- Per ogni j tale che $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ se e solo se $C[X[j]] = C[Y[j]]$.

Esempio

Consideriamo la seguente chiamata.

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

In questo esempio, supponiamo che i colori (nascosti) dei nodi siano dati da $C = [2, 0, 0, 0]$. Questo scenario è illustrato nella figura seguente. I colori sono rappresentati anche da numeri su etichette bianche attaccate a ciascun nodo.



La funzione può chiamare `perform_experiment` come segue.

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

In questa chiamata, nessun nodo viene ricolorato e tutti i nodi mantengono i loro colori originali.

Consideriamo il nodo 1 e il nodo 2. Entrambi hanno colore 0 e il percorso $1, 2$ è un percorso monocromatico. Di conseguenza, i nodi 1 e 2 si trovano nella stessa componente monocromatica.

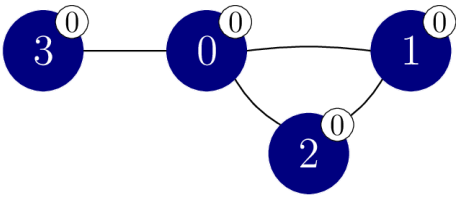
Consideriamo il nodo 1 e il nodo 3. Anche se entrambi hanno colore 0, sono in diverse componenti monocromatiche poiché non esiste un percorso monocromatico che li collega.

Puoi verificare che in totale ci sono 3 componenti monocromatiche, con nodi $\{0\}$, $\{1, 2\}$ e $\{3\}$. Quindi, questa chiamata restituisce 3.

Ora la funzione può chiamare `perform_experiment` come segue.

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

In questa chiamata, solo il nodo 0 viene ricolorato al colore 0, che dà come risultato la colorazione mostrata nella figura seguente.

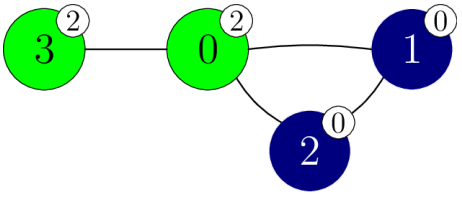


Questa chiamata restituisce 1, poiché tutti i nodi appartengono alla stessa componente monocromatica. Possiamo ora dedurre che i nodi 1, 2 e 3 hanno colore 0.

La funzione può quindi chiamare `perform_experiment` come segue.

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

In questa chiamata, il nodo 3 viene ricolorato al colore 2, che dà come risultato la colorazione mostrata nella figura seguente.



Questa chiamata restituisce 2, poiché ci sono 2 componenti monocromatiche, con nodi $\{0, 3\}$ e $\{1, 2\}$ rispettivamente. Possiamo ora dedurre che il nodo 0 ha colore 2.

La funzione `find_colours` restituisce quindi l'array $[2, 0, 0, 0]$. Poiché $C = [2, 0, 0, 0]$, ti viene assegnato il punteggio pieno.

Si noti che esistono anche valori restituiti da `find_colours` a cui verrebbe dato il 50% del punteggio, ad esempio $[1, 2, 2, 2]$ o $[1, 2, 2, 3]$.

Grader di esempio

Formato di 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 di output:

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

dove L è la lunghezza dell'array G restituito da `find_colours`, e Q è il numero di chiamate a `perform_experiment`.