Sphinx's Riddle

Marele Sfinx are o ghicitoare pentru tine. Ți se dă un graf cu N noduri, numerotate de la 0 la $N-1$.

Graful are M muchii, numerotate de la 0 la $M-1$. Fiecare muchie este bidirecțională și unește o pereche distinctă de noduri.

În particular, pentru fiecare j de la 0 la $M-1$ (inclusiv), muchia j unește nodurile $X[j]$ și $Y[j]$. Există cel mult o muchie care unește orice pereche de noduri. Două noduri se numesc **adiacente** dacă sunt unite de o muchie.

O secvență de noduri $v_0, v_1, \ldots, v_k$ (pentru $k \geq 0$) se numește un **drum** dacă fiecare două noduri consecutive v_l și v_{l+1} (pentru fiecare l, $0 \leq l < k$) sunt adiacente. Vom spune că un drum $v_0, v_1, \ldots, v_k$ **unește** nodurile v_0 și v_k. În graful pe care l-ai primit, fiecare pereche de noduri este conectată de cel puțin un drum.

Avem $N+1$ culori, numerotate de la 0 la N. Culoarea N este specială și se numește **culoarea Sfinxului**. Fiecărui nod îi este atribuită o culoare. În particular, nodul i ($0 \leq i < N$) are culoarea $C[i]$. Mai multe noduri pot avea aceeași culoare, și pot exista culori care nu sunt atribuite nici unui nod. Nici un nod nu are culoarea Sfinxului.

Un drum $v_0, v_1, \ldots, v_k$ (for $k \geq 0$) se numește **monocrom** dacă toate nodurile lui sunt de aceeași culoare, adică $C[v_l] = C[v_{l+1}]$ (pentru fiecare l, $0 \leq l < k$). Suplimentar, vom spune că nodurile p și q ($0 \leq p < N$, $0 \leq q < N$) sunt în aceeași **componentă monocromă** dacă și numai dacă ele sunt unite de un drum monocrom.

Cunoști nodurile și muchiile, dar nu cunoști ce culaore are fiecare nod. Vrei să determini culorile nodurilor, efectuând un **experiment de recolorare**.

În experimentul de recolorare, poți recolora arbitrar mai multe noduri. În particular, pentru a efectua un experiment de recolorare în primul rând alegi un tablou E de dimensiune N, unde pentru fiecare i ($0 \leq i < N$), $E[i]$ este între -1 și N **inclusiv**. Apoi, culoarea fiecărui nod i devine $S[i]$, unde valoarea $S[i]$ este:

- $C[i]$, care este culoarea originală a i, dacă $E[i] = -1$, sau
- $E[i]$, în caz contrar.

De notat că poți utiliza culoarea Sfinxului în experimentul de recolorare.

În final, Marele Sfinx anunță numărul de componente monocrome din graf, după setarea culorii pentru fiecare nod i cu $S[i]$ ($0 \leq i < N$). Noua colorare este aplicată doar pentru acest experiment de recolorare, astfel **culorile tuturor nodurilor redevin cele originale după ce se finalizează experimentul**.

Sarcina ta este să identifici culorile nodurilor în graf efectuând cel mult $2\,750$ experimente de recolorare. Poți obține un punctaj parțial dacă vei determina corect dacă fiecare pereche de noduri adiacente au aceeași culoare.

Detalii de implementare

Trebuie să implementați următoarea funcție:

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

- N: numărul de noduri din graf.
- X, Y: tablouri de dimensiune M care descriu muchiile.
- Această funcție trebuie să returneze un tablou G de lungime N, care reprezintă culorile nodurilor în graf.
- Această funcție este apelată exact o dată pentru fiecare test.

Pentru a efectua experimentele de recolorare, funcția de mai sus poate apela de mai multe ori următoarea funcție:

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

- E: un tablou de lungime N care specifică cum nodurile trebuie să fie recolorate.
- Această funcție returnează numărul de componente monocrome după recolorarea nodurilor conform E.
- Această funcție poate fi apelată de cel mult $2\,750$ ori.

Grader-ul **nu este adaptiv**, adică, culorile nodurilor sunt fixate înaintea apelului funcției `find_colours`.

Restricții

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ pentru fiecare j, $0 \leq j < M$.
- $X[j] \neq X[k]$ or $Y[j] \neq Y[k]$ pentru fiecare j și k, $0 \leq j < k < M$.
- Se garantează că între oricare două noduri există cel puțin un drum.
- $0 \leq C[i] < N$ pentru fiecare i, $0 \leq i < N$.

Subtask-uri

Subtask	Punctaj	Restricții Adiționale
1	3	$N = 2$
2	7	$N \leq 50$
3	33	Graful este un drum: $M = N - 1$ și nodurile j și $j + 1$ sunt adiacente ($0 \leq j < M$).
4	21	Graful este complet: $M = \frac{N \cdot (N-1)}{2}$ și oricare două noduri sunt adiacente.
5	36	Fără restricții adiționale.

Pentru fiecare subtask, puteți obține un punctaj parțial dacă programul vostru determină corect pentru fiecare pereche de noduri adiacente dacă ele au aceeași culoare.

Mai exact, primiți punctajul integral pentru un subtask dacă în toate testele lui, tabloul G returnat de `find_colours` este exact același ca și tabloul C (adică $G[i] = C[i]$ pentru toți i, $0 \leq i < N$). În caz contrar, veți primi 50% din punctajul pentru subtask dacă următoarele condiții se vor respecta în toate testele:

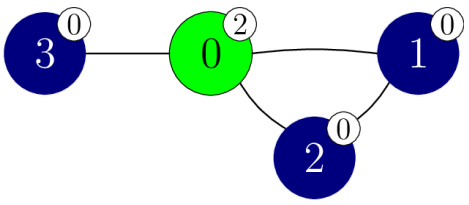
- $0 \leq G[i] < N$ pentru fiecare i astfel încât $0 \leq i < N$;
- Pentru fiecare j astfel încât $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ dacă și numai dacă $C[X[j]] = C[Y[j]]$.

Exemplu

Considerăm următorul apel.

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

Pentru acest exemplu, presupunem că culorile (ascunse) ale nodurilor sunt descrise de secvența $C = [2, 0, 0, 0]$. Acest scenariu este prezentat în următoarea figură. Culorile sunt adițional reprezentate prin numere în etichetele albe, atașate la fiecare nod.



Funcția va apela `perform_experiment` după cum urmează.

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

În acest apel, nici un nod nu este recolorat, deci toate nodurile își păstrează culorile originale.

Considerăm nodul 1 și nodul 2. Ambele noduri au culoarea 0 și drumul $1, 2$ este unul monocrom. În rezultat, nodurile 1 și 2 sunt în aceeași componentă monocromă.

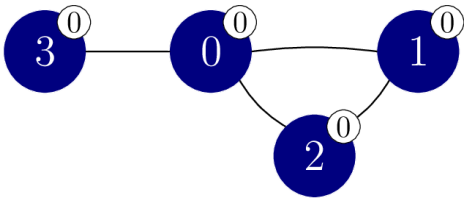
Considerăm nodul 1 și nodul 3. Chiar dacă ambele noduri au culoarea 0, ele se află în componente monocrome distincte deoarece nu există un drum monocrom care să le conecteze.

Per total, sunt 3 componente monocrome, cu nodurile $\{0\}$, $\{1, 2\}$, și $\{3\}$. Astfel, acest apel returnează 3.

Acum funcția poate apela `perform_experiment` după cum urmează.

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

În acest apel, doar nodul 0 este recolorat în culoarea 0, și are ca rezultat colorarea prezentată în următoarea figură.

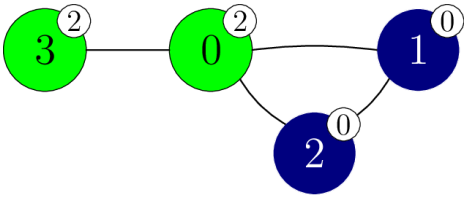


Acest apel returnează 1, deoarece toate nodurile aparțin aceleiași componente monocrome. Acum noi putem deduce că nodurile 1, 2, și 3 au culoarea 0.

Funcția poate apela `perform_experiment` după cum urmează.

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

În acest apel, nodul 3 este recolorat în culoarea 2, și are ca rezultat colorarea prezentată în următoarea figură.



Acest apel returnează 2, deoarece există 2 componente monocrome, având nodurile $\{0, 3\}$ și $\{1, 2\}$ respectiv. Putem deduce că nodul 0 are culoarea 2.

Funcția `find_colours` returnează tabloul $[2, 0, 0, 0]$. Deoarece $C = [2, 0, 0, 0]$, se acordă scorul integral.

De remarcat, că există de asemenea multiple valori returnate, pentru care se acordă 50% din scor, de exemplu $[1, 2, 2, 2]$ sau $[1, 2, 2, 3]$.

Grader local

Format intrare:

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

Format ieșire:

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

Aici, L este lungimea tabloului G returnată de `find_colours`, și Q este numărul de apeluri către `perform_experiment`.