Zagadka Sfinksa

Wielki Sfinks ma dla Ciebie zagadkę. Dany jest graf składający się z N wierzchołków. Wierzchołki są ponumerowane od 0 do $N-1$. W grafie znajduje się M krawędzi ponumerowanych od 0 do $M-1$. Każda krawędź jest dwukierunkowa i łączy parę różnych wierzchołków. Dla każdego j od 0 do $M-1$ (włącznie) krawędź j łączy wierzchołki $X[j]$ i $Y[j]$. Każdą parę wierzchołków łączy co najwyżej jedna krawędź. Dwa wierzchołki nazywane są **sąsiadującymi**, jeśli są połączone krawędzią.

Sekwencję wierzchołków $v_0, v_1, \ldots, v_k$ (dla $k \geq 0$) nazwiemy **ścieżką**, jeśli każde dwa kolejne wierzchołki v_l i v_{l+1} (dla każdego l takiego, że $0 \leq l < k$) są sąsiadujące. Mówimy, że ścieżka $v_0, v_1, \ldots, v_k$ **łączy** wierzchołki v_0 i v_k. Możesz założyć, że w otrzymanym przez Ciebie grafie, każda para wierzchołków jest połączona jakąś ścieżką.

Istnieje $N+1$ kolorów, ponumerowanych od 0 do N. Kolor N jest szczególny i nazywany jest **kolorem Sfinksa**. Każdemu wierzchołkowi przypisano jeden z kolorów. Dokładniej, i-ty wierzchołek ($0 \leq i < N$) ma kolor $C[i]$. Wiele wierzchołków może mieć ten sam kolor, a także mogą istnieć kolory nieprzypisane do żadnego wierzchołka. Żaden wierzchołek nie ma koloru Sfinksa, czyli $0 \leq C[i] < N$ ($0 \leq i < N$).

Ścieżkę $v_0, v_1, \ldots, v_k$ (dla $k \geq 0$) nazwiemy **monochromatyczną**, jeśli wszystkie jej wierzchołki mają ten sam kolor, tj. $C[v_l] = C[v_{l+1}]$ (dla każdego l takiego, że $0 \leq l < k$). Ponadto mówimy, że wierzchołki p i q ($0 \leq p < N$, $0 \leq q < N$) są w tej samej **monochromatycznej składowej**, wtedy i tylko wtedy, gdy są połączone monochromatyczną ścieżką.

Znasz wierzchołki i krawędzie, ale nie wiesz, jaki kolor ma każdy z wierzchołków. Chcesz poznać kolory wierzchołków, wykonując **przekolorowywujące eksperymenty**.

W przekolorowywującym eksperymencie możesz zmienić kolor dowolnej liczby wierzchołków. Aby wykonać taki eksperyment, najpierw wybierasz tablicę E o rozmiarze N, gdzie dla każdego i ($0 \leq i < N$), $E[i]$ mieści się w przedziale od -1 do N **włącznie**. Następnie kolor i-tego wierzchołka staje się $S[i]$, gdzie wartość $S[i]$ wynosi:

- $C[i]$, czyli oryginalny kolor i-tego wierzchołka, jeżeli $E[i] = -1$, lub
- $E[i]$, w przeciwnym wypadku.

Należy pamiętać, że oznacza to, iż podczas eksperymentu możesz wykorzystać kolor Sfinksa.

Na koniec eksperymentu Wielki Sfinks ogłasza liczbę monochromatycznych składowych w grafie, po ustawieniu koloru każdego wierzchołka i na $S[i]$ ($0 \leq i < N$). Nowe kolorowanie jest stosowane wyłącznie w tym konkretnym eksperymencie, a więc **kolory wszystkich wierzchołków powracają do oryginalnych po zakończeniu eksperymentu**.

Twoim zadaniem jest zidentyfikowanie kolorów wierzchołków w grafie, wykonując maksymalnie $2\,750$ eksperymentów. Możesz również otrzymać wynik częściowy, jeśli, dla każdej pary sąsiadujących wierzchołków, poprawnie określisz czy mają ten sam kolor.

Szczegóły implementacyjne

Powinieneś zaimplementować poniższą funkcję.

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

- N: liczba wierzchołków w grafie.
- X, Y: tablice o długości M opisujące krawędzie.
- Ta procedura powinna zwrócić tablicę G o długości N reprezentującą kolory wierzchołków w grafie.
- Ta procedura jest wywoływana dokładnie raz dla każdego testu.

Powyższa procedura może wywoływać poniższą procedurę, aby przeprowadzać przekolorowywujące eksperymenty:

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

- E: tablica o długości N określająca sposób zmiany kolorów wierzchołków.
- Ta procedura zwraca liczbę monochromatycznych składowych po przekolorowaniu wierzchołków zgodnie z E.
- Tę procedurę można wywołać maksymalnie $2\,750$ razy.

Sprawdzaczka **nie jest adaptywna**, to znaczy kolory wierzchołków są ustalane przed wywołaniem funkcji `find_colours`.

Ograniczenia

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ dla każdego j takiego, że $0 \leq j < M$.
- $X[j] \neq X[k]$ lub $Y[j] \neq Y[k]$ dla każdego j i k takiego, że $0 \leq j < k < M$.
- Każda para wierzchołków jest połączona pewną ścieżką.
- $0 \leq C[i] < N$ dla każdego i takiego, że $0 \leq i < N$.

Podzadania

Podzadanie	Punktacja	Dodatkowe ograniczenia
1	3	$N = 2$
2	7	$N \leq 50$
3	33	Graf jest ścieżką: $M = N - 1$ oraz wierzchołki j i $j + 1$ są sąsiadujące ($0 \leq j < M$).
4	21	Graf jest pełny: $M = \frac{N \cdot (N-1)}{2}$ oraz każda para wierzchołków jest sąsiadująca.
5	36	Brak dodatkowych ograniczeń.

W każdym podzadaniu możesz uzyskać ocenę częściową, jeśli Twój program dla każdej pary sąsiadujących wierzchołków prawidłowo określi, czy mają ten sam kolor.

Dokładniej rzecz biorąc, otrzymujesz pełną punktację za podzadanie, jeśli we wszystkich jego testach, tablica G zwrócona przez `find_colours` jest dokładnie taka sama jak tablica C (tj. $G[i] = C[i]$ dla każdego i takiego, że $0 \leq i < N$). W przeciwnym razie, otrzymujesz 50% punktów za podzadanie, jeśli spełnione są następujące warunki we wszystkich jego testach:

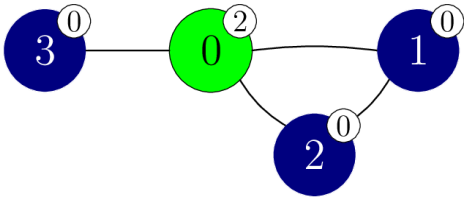
- $0 \leq G[i] < N$ dla każdego i takiego, że $0 \leq i < N$;
- Dla każdego j takiego, że $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ wtedy i tylko wtedy, gdy $C[X[j]] = C[Y[j]]$.

Przykład

Rozważmy poniższe wywołanie.

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

W tym przykładzie załóżmy, że (ukryte) kolory wierzchołków są dane przez $C = [2, 0, 0, 0]$. Scenariusz ten pokazano na poniższym rysunku. Kolory są dodatkowo reprezentowane przez liczby na białych etykietach przymocowanych do każdego wierzchołka.



Procedura może wywołać `perform_experiment` w następujący sposób.

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

W tym przypadku nie zmienia się kolor żadnego wierzchołka, ponieważ wszystkie wierzchołki zachowują swoje oryginalne kolory.

Rozważmy wierzchołek 1 i wierzchołek 2. Oba mają kolor 0, a ścieżka $1, 2$ jest ścieżką monochromatyczną. W rezultacie wierzchołki 1 i 2 znajdują się w tej samej monochromatycznej składowej.

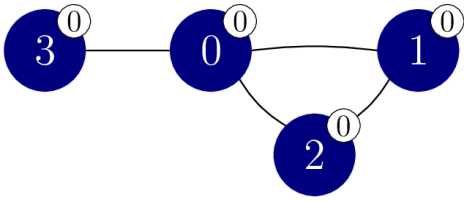
Rozważmy wierzchołek 1 i wierzchołek 3. Mimo, że oba mają kolor 0, to są w różnych monochromatycznych składowych, ponieważ nie łączy ich żadna monochromatyczna ścieżka.

W sumie są 3 monochromatyczne składowe, składające się z wierzchołków $\{0\}$, $\{1, 2\}$ i $\{3\}$. Zatem to wywołanie zwraca 3.

Teraz procedura może wywołać `perform_experiment` w następujący sposób.

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

W tym wywołaniu tylko wierzchołek 0 zostaje przekolorowany na kolor 0, co skutkuje kolorowaniem pokazanym na poniższym rysunku.

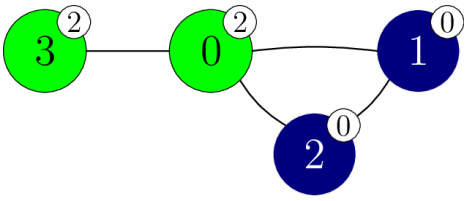


To wywołanie zwraca 1, ponieważ wszystkie wierzchołki należą do tej samej monochromatycznej składowej. Możemy teraz wywnioskować, że wierzchołki 1, 2 i 3 mają kolor 0.

Następnie procedura może wywołać `perform_experiment` w następujący sposób.

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

W tym wywołaniu wierzchołek 3 zostaje pokolorowany na kolor 2, co skutkuje kolorowaniem pokazanym na poniższym rysunku.



To wywołanie zwraca 2, ponieważ istnieją 2 monochromatyczne składowe, odpowiednio z wierzchołkami $\{0, 3\}$ i $\{1, 2\}$. Możemy wywnioskować, że wierzchołek 0 ma kolor 2.

Procedura `find_colours` zwraca następnie tablicę $[2, 0, 0, 0]$. Ponieważ $C = [2, 0, 0, 0]$, rozwiązanie to otrzymuje pełną punktację.

Należy pamiętać, że istnieje wiele różnych wartości, które można zwrócić, otrzymując przy tym 50% punktacji, na przykład $[1, 2, 2, 2]$ lub $[1, 2, 2, 3]$.

Przykładowy program oceniający

Format wejścia:

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

Format wyjścia:

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

L jest długością tablicy G zwróconej przez `find_colours`, a Q jest liczbą wywołań `perform_experiment`.