Sfinxens Gåta ≽^•ω•^≼

Den allsmäktiga Sfinxen (meow) har en gåta till dig. Du får en graf med N noder, numrerade från 0 till $N-1$. Grafen har M kanter, numrerade från 0 till $M-1$. Varje kant sammankopplar två olika noder och är dubbelriktad. Mer exakt, för varje j ($0 \leq j \leq M-1$), så sammankopplar kant j noderna $X[j]$ och $Y[j]$. Varje par av noder är sammankopplade av som mest en kant. Vi säger att två noder är **grannar** om de sammankopplas av en kant.

En sekvens av noder $v_0, v_1, \ldots, v_k$ ($k \geq 0$) kallas en väg om varje två intilliggande noder v_i och v_{i+1} ($0 \leq i < k$) är sammankopplade. Vi säger att en väg $v_0, v_1, \ldots, v_k$ **sammankopplar** noderna v_0 och v_k. I den givna grafen är varje par av noder sammankopplade av minst en väg.

Det finns $N+1$ stycken färger, numrerade från 0 till N. Färgen N är speciell och kallas **Sfinxens färg**. Varje nod har en färg. Mer exakt så har nod i ($0 \leq i < N$) färgen $C[i]$. Flera noder kan ha samma färg, och det kan finnas färger som inte används av någon nod alls. Det är däremot givet att ingen nod använder Sfinxens färg, det vill säga $0 \leq C[i] < N$ för alla $0 \leq i < N$.

En väg $v_0, v_1, \ldots, v_k$ ($k \geq 0$) kallas **monokromatisk** om alla noder i den vägen har samma färg, det vill säga $C[v_l] = C[v_{l+1}]$ ($0 \leq l < k$). Vi säger även att två noder p och q ($0 \leq p < N, 0 \leq q < N$) är i samma **monokromatiska komponent** om och endast om de är sammankopplade av en monokromatisk väg.

Du får givet noderna och kanterna, men du vet inte vilken färg som varje nod har. Ditt mål är att hitta färgerna genom att utföra **färgläggnings-experiment**.

I varje färgläggnings-experiment så kan du ändra färgen på hur många noder som helst. Mer exakt, när du utför ett färgläggnings-experiment, så väljer du först en array E av storlek N, där $-1 \leq E[i] \leq N$ ($0 \leq i < N$). Utifrån denna sätts färgen av nod i till 🎨$[i]$, där värdet av 🎨$[i]$ bestäms som följande:

- Om $E[i] = -1$ sätts 🎨$[i] = C[i]$, alltså behåller noden sin ursprungliga färg.
- Annars sätts 🎨$[i] = E[i]$, det vill säga ändrar du färgen till $E[i]$.

Notera att du alltså kan använda Sfinxens färg i din om-färgläggning.

Som svar berättar Sfinxen antalet monokromatiska komponenter i grafen efter att ha ändrat färgen av varje nod i till 🎨$[i]$ ($0 \leq i < N$). Omfärgläggningen appliceras endast för det här färgläggnings-experimentet, vilket innebär att **färgerna av alla noderna återgår till sin ursprungliga färg efter experimentet**.

Din uppgift är att hitta färgen av varje nod i grafen genom att utföra som mest 2750 färgläggnings-experiment. Du kan också få delpoäng om du korrekt identifierar alla par av noder som är grannar och har samma färg.

Implementationsdetaljer

SNÄLLA implementera följande funktion 😭🔥😔✊:

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

- N: antalet noder i grafen.
- X, Y: arrayer med längd M som beskriver kanterna.
- Denna funktionen ska returnerna en array G med längd N, som representerar nodernas färger.
- Denna funktionen anropas exakt en gång per testfall.

Ovanstående funktion kan anropa följande funktion för att utföra färgläggnings-experiment:

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

- E: en array av längd N som beskriver hur noderna ska färgas om.
- Denna funktionen returnerar antalet monokromatiska komponenter efter att noderna har färglagts utifrån E.
- Denna funktionen får anropas som mest $2\,750$ gånger.

Gradern är **inte adaptiv**, det vill säga är nodernas färger bestämda innan `find_colours` anropas.

Begränsningar

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ för varje j där $0 \leq j < M$.
- $X[j] \neq X[k]$ eller $Y[j] \neq Y[k]$ för varje j och k där $0 \leq j < k < M$.
- Varje par av noder är sammankopplade av någon väg.
- $0 \leq C[i] < N$ för varje i där $0 \leq i < N$.

Delpoäng

Grupp	Poäng	Ytterligare begränsningar
1	3	$N = 2$
2	7	$N \leq 50$
3	33	Grafen är en väg: $M = N - 1$ och noderna j och $j + 1$ är grannar ($0 \leq j < M$).
4	21	Grafen är komplett: $M = \frac{N \cdot (N-1)}{2}$, och därmed är varje par av noder grannar.
5	36	Inga ytterligare begränsningar.

I varje grupp kan du få delpoäng om ditt program kan avgöra för varje par av noder som är grannar ifall de har samma färg.

Mer exakt, du får full poäng för en grupp om arrayen G som returneras av `find_colours` är exakt samma som C ($G[i] = C[i]$ för alla i där $0 \leq i < N$) för alla testfall i gruppen.

Om du inte får full poäng kan du fortfarande få 50% av poängen för gruppen om följande stämmer i alla gruppens testfall:

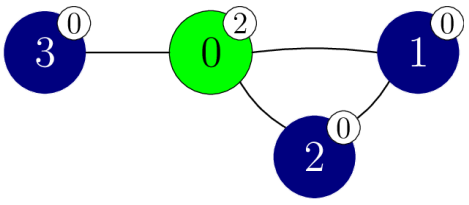
- $0 \leq G[i] < N$ för varje i där $0 \leq i < N$;
- För varje j där $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ om och endast om $C[X[j]] = C[Y[j]]$.

Exempel

Betrakta följande anrop:

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

I det här exemplet, anta att de (hemliga) färgerna av noderna är givna av $C = [2, 0, 0, 0]$. Detta fallet är ritat i figuren nedan. Färgerna beskrivs ytterligare av talen som står på de vita lapparna som sitter ihop med varje nod.



Därefter väljer vi att anropa `perform_experiment` på följande vis:

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

I detta anrop omfärgläggs inga noder, utan alla noder behåller sina ursprungliga färger.

Betrakta nod 1 och nod 2. Båda har färgen 0 och vägen $1, 2$ är en monokromatisk väg. Därmed är noderna 1 och 2 i samma monokromatiska komponent.

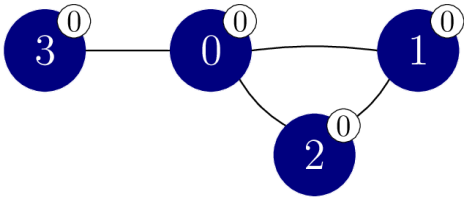
Betrakta nod 1 och nod 3. Trots att de båda har färgen 0, så är de i olika monokromatiska komponenter eftersom det inte finns någon monokromatisk väg som sammankopplar dom.

Sammanlagt finns det 3 monokromatiska komponenter med noderna $\{0\}$, $\{1, 2\}$, och $\{3\}$. Därmed returnerar detta anropet 3.

Därefter väljer vi att anropa `perform_experiment` på följande vis:

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

I detta anropet blir endast nod 0 omfärgad till färg 0, vilket leder till färgläggning som visas i nedanstående figur:

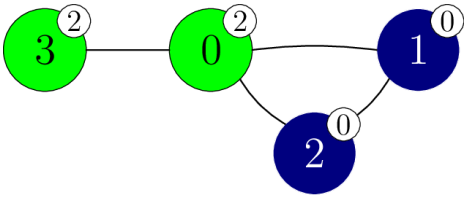


Detta anrop returnerar 1 eftersom alla noder tillhör samma monokromatiska komponent. Vi kan nu härleda att noderna 1, 2 och 3 har färg 0.

Vi väljer sedan att anropa `perform_experiment` på följande vis:

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

I detta anropet omfärgläggs nod 3 till färg 2, vilket leder till färgläggningen som visas i nedanstående figur.



Detta anrop returnerar 2, eftersom det finns 2 stycken monokromatiska komponenter, med noder $\{0, 3\}$ och $\{1, 2\}$ respektive. Vi kan därmed härleda att nod 0 har färg 2.

Funktionen `find_colours` returnerar sedan arrayen $[2, 0, 0, 0]$. Eftersom $C = [2, 0, 0, 0]$ får vi full poäng på testfallet.

Notera att det finns flera möjlig giltiga returvärden som ger 50% av poängen, exempelvis $[1, 2, 2, 2]$ eller $[1, 2, 2, 3]$.

Exempelgrader

Indataformat:

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

Utdataformat:

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

Där L är längden av array G som returnerats av `find_colours`, och Q är antalet anrop till `perform_experiment`.