Raadsel van de Sfinx

De Grote Sfinx heeft een raadsel voor je. Je krijgt een graaf met N knopen. De knopen zijn genummerd van 0 tot $N-1$. Er zijn M kanten in de graaf. Elke kant verbindt een paar verschillende knopen en is bidirectioneel. Twee knopen worden **aangrenzend** genoemd als ze door een kant met elkaar verbonden zijn. Om precies te zijn, voor elke j van 0 tot en met $M-1$, zijn $X[j]$ en $Y[j]$ aangrenzend. Er is hoogstens één kant die een specifiek paar knopen verbindt.

Een reeks knopen $v_0, v_1, \ldots, v_k$ (voor $k \geq 0$) wordt een **pad** genoemd als elk twee opeenvolgende knopen v_l en v_{l+1} (voor elke l zodat $0 \leq l < k$) aangrenzend zijn. We zeggen dat een pad $v_0, v_1, \ldots, v_k$ de knopen v_0 en v_k **verbindt**. In de graaf die je krijgt, is elk paar knopen met elkaar verbonden door een pad.

Er zijn $N+1$ kleuren, genummerd van 0 tot N. Kleur N is speciaal en wordt de **kleur van de Sfinx** genoemd. Aan elke knoop is een kleur toegewezen. Specifiek heeft knoop i ($0 \leq i < N$) kleur $C[i]$. Meerdere knopen kunnen dezelfde kleur hebben, en er kunnen kleuren zijn die aan geen enkele knoop zijn toegewezen. Geen enkele knoop heeft de kleur van de Sfinx, dat wil zeggen, $0 \leq C[i] < N$ ($0 \leq i < N$).

Een pad $v_0, v_1, \ldots, v_k$ (voor $k \geq 0$) wordt **monochromatisch** genoemd als alle knopen dezelfde kleur hebben, ofwel $C[v_l] = C[v_{l+1}]$ (voor elke l zodat $0 \leq l < k$). Bovendien zeggen we dat de knopen p en q ($0 \leq p < N$, $0 \leq q < N$) in dezelfde **monochromatische component** liggen dan en slechts dan als ze door een monochromatisch pad met elkaar verbonden zijn.

Je kent de knopen en kanten, maar je weet niet welke kleur elke knoop heeft. Je wilt de kleuren van de knopen achterhalen, door **herkleuringsexperimenten** uit te voeren.

In een herkleuringsexperiment, mag je een arbitrair aantal knopen een andere kleur geven. Om precies te zijn, om een herkleuringsexperiment uit te voeren kies je eerst een array E van grootte N, waar voor elke i ($0 \leq i < N$), $E[i]$ tussen -1 en N ligt **inclusief**. De kleur van elke knoop i wordt dan $S[i]$, waarbij de waarde van $S[i]$ betekent:

- $C[i]$, dus de originele kleur van i, als $E[i] = -1$, of
- $E[i]$, in elk ander geval.

Houd er rekening mee dat je de kleur van de Sfinx kunt gebruiken bij het herkleuren.

Als resultaat, kondigt de Grote Sfinx het aantal monochromatische componenten in de graaf aan nadat de kleur van elke knoop i is ingesteld op $S[i]$ ($0 \leq i < N$). De nieuwe kleuren worden alleen

toegepast voor dit specifieke herkleuringsexperiment, **dus de kleuren van alle knopen keren terug naar de oorspronkelijke kleuren nadat het experiment is afgelopen**.

Jouw taak is om de kleuren van de knopen in de graaf te achterhalen door maximaal $2\,750$ herkleuringsexperimenten uit te voeren. Je kunt ook een gedeeltelijke score behalen als je voor elk paar aangrenzende knopen correct bepaalt, of ze dezelfde kleur hebben.

Implementatiedetails

Je moet de volgende procedure implementeren.

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

- N: het aantal knopen in de graaf.
- X, Y: arrays met lengte M die de kanten beschrijven.
- Deze procedure moet een array G van lengte N returnen, die de kleuren van de knopen in de graaf aangeeft.
- Deze procedure wordt voor elke testcase precies één keer aangeroepen.

De bovenstaande procedure kan calls doen naar de volgende procedure om herkleuringsexperimenten uit te voeren:

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

- E: een array met lengte N die aangeeft hoe knopen opnieuw gekleurd moeten worden.
- Deze procedure returnt het aantal monochromatische componenten na het opnieuw kleuren van de knopen volgens E.
- Deze procedure kan maximaal $2\,750$ keer worden aangeroepen.

De grader is **niet adaptief**, dat wil zeggen dat de kleuren van de knopen worden vastgelegd voordat `find_colours` wordt aangeroepen.

Beperkingen

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ voor elke j zodat $0 \leq j < M$.
- $X[j] \neq X[k]$ of $Y[j] \neq Y[k]$ voor elke j en k zodat $0 \leq j < k < M$.
- Elk paar knopen is verbonden door een pad.
- $0 \leq C[i] < N$ voor elke i zodat $0 \leq i < N$.

Subtaken

Subtaak	Score	Extra beperkingen
1	3	$N = 2$
2	7	$N \leq 50$
3	33	De graaf is een pad: $M = N - 1$ en de knopen j en $j + 1$ zijn aangrenzend ($0 \leq j < M$).
4	21	De graaf is compleet: $M = \frac{N \cdot (N-1)}{2}$ en elke twee knopen zijn aangrenzend.
5	36	Geen extra beperkingen.

In elke subtaak kun je een gedeeltelijke score behalen als je programma correct bepaalt voor elk paar aangrenzende knopen of ze dezelfde kleur hebben.

Om precies te zijn, je krijgt de volledige score van een subtaak als in alle testgevallen, de array G gereturnt door `find_colours` exact hetzelfde is als de array C (dus $G[i] = C[i]$ voor alle i zodat $0 \leq i < N$). Anders, krijg je 50% van de score voor een subtaak als aan de volgende voorwaarden wordt voldaan in al zijn testgevallen:

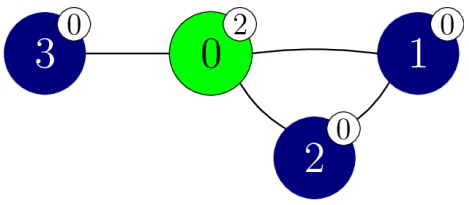
- $0 \leq G[i] < N$ voor elke i zodanig dat $0 \leq i < N$;
- Voor elke j zodanig dat $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ dan en slechts dan als $C[X[j]] = C[Y[j]]$.

Voorbeeld

Bekijk de volgende call.

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

Voor dit voorbeeld zijn de (verborgen) kleuren van de knopen gegeven door $C = [2, 0, 0, 0]$. Dit scenario wordt in de volgende afbeelding weergegeven. De kleuren worden ook weergegeven door middel van nummers op witte labels die aan elke knoop zijn bevestigd.



De procedure kan `perform_experiment` als volgt aanroepen.

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

Bij deze aanroep wordt geen enkel knoop opnieuw gekleurd, aangezien alle knopen hun oorspronkelijke kleuren behouden.

Kijk naar knoop 1 en knoop 2. Ze hebben beide kleur 0 en het pad $1, 2$ is een monochroom pad. Als gevolg hiervan bevinden de knopen 1 en 2 zich in dezelfde monochromatische component.

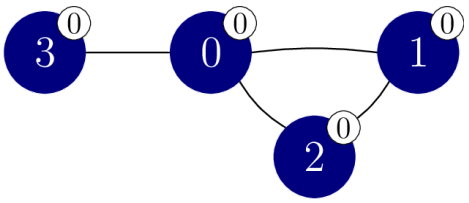
Kijk naar knoop 1 en knoop 3. Ondanks dat ze allebei de kleur 0 hebben, zitten ze in verschillende monochromatische componenten omdat er geen monochroom pad is dat ze met elkaar verbindt.

In totaal zijn er 3 monochromatische componenten, met knopen $\{0\}$, $\{1, 2\}$ en $\{3\}$. Deze aanroep returnt dus 3.

De procedure kan nu `perform_experiment` als volgt aanroepen.

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

In deze aanroep wordt alleen knoop 0 opnieuw gekleurd naar kleur 0, wat resulteert in de kleuren die in de onderstaande afbeelding zijn weergegeven.

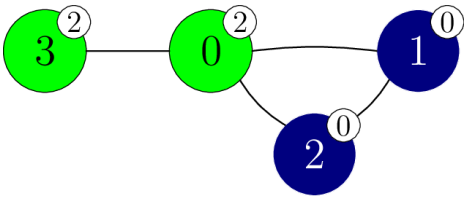


Deze aanroep returnt 1, omdat alle knopen in dezelfde monochromatische component liggen. We kunnen nu afleiden dat de knopen 1, 2 en 3 kleur 0 hebben.

De procedure kan dan `perform_experiment` als volgt aanroepen.

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

In deze oproep wordt knoop 3 opnieuw gekleurd naar kleur 2, wat resulteert in de kleuren die in de onderstaande afbeelding zijn weergegeven.



Deze aanroep returnt 2, omdat er 2 monochromatische componenten zijn, met respectievelijk knopen $\{0, 3\}$ en $\{1, 2\}$. We kunnen afleiden dat knoop 0 kleur 2 heeft.

De procedure `find_colours` returnt vervolgens de array $[2, 0, 0, 0]$. Omdat $C = [2, 0, 0, 0]$ wordt de volledige score gegeven.

Houd er rekening mee dat er ook meerdere returnwaarden zijn, waarvoor 50% van de score zou worden gegeven, bijvoorbeeld $[1, 2, 2, 2]$ of $[1, 2, 2, 3]$.

Voorbeeldgrader

Invoerformaat:

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

Uitvoerformaat:

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

Hierbij is L de lengte van de array G die wordt gereturnt door `find_colours`, en Q is het aantal aanroepen van `perform_experiment`.