L'énigme du Sphinx (Sphinx's Riddle)

Le Grand Sphinx a une énigme pour vous. Il vous donne un graphe à N sommets. Les sommets sont numérotés de 0 à $N-1$. Il y a M arêtes dans le graphe, numérotées de 0 à $M-1$. Chaque arête relie une paire de sommets distincts et est bidirectionnelle. Plus précisément, pour chaque j entre 0 et $M-1$ (tous deux inclus), l'arête j relie les sommets $X[j]$ et $Y[j]$. Pour chaque paire de sommets, il y a au plus une arête qui les relie. Deux sommets sont dits **adjacents** s'ils sont reliés par une arête.

Une suite de sommets $v_0, v_1, \ldots, v_k$ (pour $k \geq 0$) est appelée un **chemin** si tous les sommets consécutifs v_l et v_{l+1} (pour chaque l tel que $0 \leq l < k$) sont adjacents. On dit que le chemin $v_0, v_1, \ldots, v_k$ **connecte** les sommets v_0 et v_k. Dans le graphe fourni, chaque paire de sommets est connectée par un chemin.

Il y a $N+1$ couleurs, numérotées de 0 à N. La couleur N est spéciale et est appelée la **couleur du Sphinx**. Chaque sommet est coloré dans l'une de ces couleurs. Plus précisément, le sommet i ($0 \leq i < N$) est coloré dans la couleur $C[i]$. Plusieurs sommets peuvent partager la même couleur, et certaines couleurs peuvent apparaître sur aucun des sommets. Aucun sommet ne porte la couleur du Sphinx, autrement dit, $0 \leq C[i] < N$ ($0 \leq i < N$).

Un chemin $v_0, v_1, \ldots, v_k$ (for $k \geq 0$) est appelé **monochromatique** si tous ses sommets ont la même couleur, i.e. $C[v_l] = C[v_{l+1}]$ (pour chaque l tel que $0 \leq l < k$). De plus, on dit que deux sommets p et q ($0 \leq p < N$, $0 \leq q < N$) sont dans la même **composante monochromatique** si et seulement s'ils sont reliés par un chemin monochromatique.

Vous connaissez les sommets et les arêtes, mais vous ne connaissez pas en quelle couleur chaque sommet est coloré. Votre objectif est de trouver la couleur des sommets en effectuant des **expériences de recoloration**.

Lors d'une expérience de recoloration, vous pouvez recolorer un nombre arbitraire de sommets. Plus précisément, pour effectuer une expérience de recoloration, vous commencez par choisir un tableau E de longueur N, où, pour chaque i ($0 \leq i < N$), $E[i]$ est entre -1 et N **tous deux inclus**. La couleur de chaque sommet i devient ensuite $S[i]$, où la valeur de $S[i]$ est :

- $C[i]$, c'est à dire la couleur originale de i, si $E[i] = -1$, ou
- $E[i]$ sinon.

Notez que cela signifie que vous pouvez utiliser la couleur du Sphinx dans votre recoloration.

Enfin, le Grand Sphinx annonce le nombre de composantes monochromatiques du graphe, après avoir changé la couleur de chaque sommet i en $S[i]$ ($0 \leq i < N$). La nouvelle coloration est appliquée uniquement pour cette expérience de recoloration particulière, c'est à dire que **les sommets reviennent à leur couleur initiale après la fin de l'expérience.**.

Votre tâche est d'identifier la couleur de chacun des sommets du graphe en effectuant au plus $2\,750$ expériences de recoloration. Vous pouvez également recevoir un score partiel si vous déterminez correctement, pour chaque paire de sommets adjacents, s'ils partagent la même couleur.

Détails d'implémentation

Vous devez implémenter la fonction suivante :

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

- L'entier N représente le nombre de sommets du graphe.
- Les tableaux X, Y ont pour longueur M et décrivent les arêtes.
- Cette fonction doit renvoyer un tableau G de longueur N, qui représente la couleur de chacun des sommets du graphe.
- Cette fonction est appelée exactement une fois sur chaque test.

La fonction ci-dessus peut appeler la fonction suivante pour effectuer une expérience de recoloration :

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

- Le tableau E a pour longueur N et décrit comment les sommets doivent être recolorés.
- Cette fonction renvoie le nombre de composantes monochromatiques après avoir recoloré les sommets selon E.
- Cette fonction peut être appelée au plus $2\,750$ fois.

Le grader n'est **pas adaptatif**. Autrement dit, la couleur de chaque sommet est fixée avant l'appel à `find_colours`.

Contraintes

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ pour chaque j tel que $0 \leq j < M$.
- $X[j] \neq X[k]$ ou $Y[j] \neq Y[k]$ pour chaque j et k tels que $0 \leq j < k < M$.
- Chaque paire de sommets est connectée par un chemin.

- $0 \leq C[i] < N$ pour chaque i tel que $0 \leq i < N$.

Sous-tâches

Sous-tâche	Score	Contraintes supplémentaires
1	3	$N = 2$
2	7	$N \leq 50$
3	33	Le graphe est un chemin : $M = N - 1$ et les sommets j et $j + 1$ sont adjacents ($0 \leq j < M$).
4	21	Le graphe est complet : $M = \frac{N \cdot (N-1)}{2}$ et pour chaque paire de sommets, les deux sommets de la paire sont adjacents.
5	36	Aucune contrainte supplémentaire.

Dans chaque sous-tâche, vous pouvez obtenir un score partiel si votre programme détermine correctement si, pour chaque paire de sommets, les deux sommets de la paire sont de la même couleur.

Plus précisément, vous obtiendrez tous les points d'une sous-tâche si, sur chacun de ses tests, le tableau G renvoyé par `find_colours` est identique à C (i.e. $G[i] = C[i]$ pour chaque i tel que $0 \leq i < N$). Sinon, vous obtiendrez 50% des points d'une sous-tâche si la condition suivante est vérifiée sur chacun de ses tests :

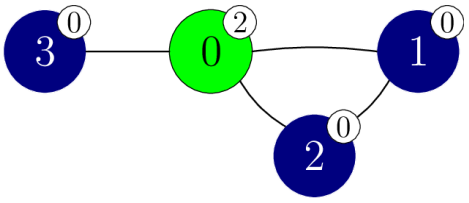
- $0 \leq G[i] < N$ pour chaque i tel que $0 \leq i < N$;
- Pour chaque j tel que $0 \leq j < M$:
  - $G[X[j]] = G[Y[j]]$ si et seulement si $C[X[j]] = C[Y[j]]$.

Exemple

Considérons l'appel de fonction suivant :

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

Sur cet exemple, supposons que les couleurs (cachées) des sommets soient données par $C = [2, 0, 0, 0]$. Ce scenario est illustré dans la figure suivante. Les couleurs sont également représentées par les nombres sur les étiquettes blanches de chaque sommet.



Cette fonction peut appeler `perform_experiment` de la manière suivante :

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

Dans cet appel, aucun sommet n'est recoloré, puisque tous les sommets gardent leur couleur initiale.

Considérons le sommet 1 et le sommet 2. Ils ont tous les deux la couleur 0, et le chemin $1, 2$ est un chemin monochromatique. Ainsi, les sommets 1 et 2 sont dans la même composante monochromatique.

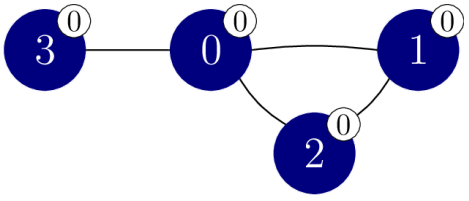
Considérons maintenant le sommet 1 et le sommet 3. Même s'ils ont tous les deux la couleur 0, ils sont dans des composantes monochromatiques différentes car il n'existe pas de chemin monochromatique qui les connecte.

En tout, il y a 3 composantes monochromatiques, constituées des sommets $\{0\}$, $\{1, 2\}$, et $\{3\}$. Ainsi, cet appel renvoie 3.

Maintenant, la fonction peut appeler `perform_experiment` de la manière suivante :

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

Dans cet appel, seul le sommet 0 est recoloré dans la couleur 0, ce qui résulte en la coloration illustrée dans la figure suivante :

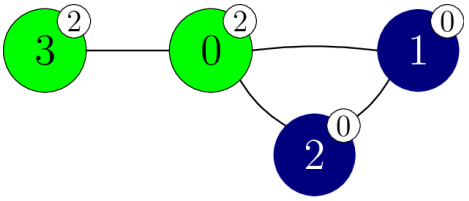


Cet appel renvoie 1, puisque tous les sommets appartiennent à la même composante monochromatique. On peut donc déduire que les sommets 1, 2, et 3 ont pour couleur 0.

La procédure peut ensuite appeler `perform_experiment` de la manière suivante :

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

Dans cet appel, le sommet 3 est recoloré dans la couleur 2, ce qui résulte en la coloration illustrée dans la figure suivante :



Cet appel renvoie 2, puisqu'il y a 2 composantes monochromatiques, constituées des sommets $\{0, 3\}$ et $\{1, 2\}$ respectivement. On peut donc déduire que le sommet 0 a pour couleur 2.

La fonction `find_colours` renvoie ensuite le tableau $[2, 0, 0, 0]$. Puisque $C = [2, 0, 0, 0]$, tous les points sont accordés.

Notez qu'il y a plusieurs valeurs de retour pour lesquelles 50% des points seront attribués, par exemple $[1, 2, 2, 2]$ ou $[1, 2, 2, 3]$.

Évaluateur d'exemple (grader)

Format d'entrée :

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

Format de sortie :

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

Ici, L est la longueur du tableau G renvoyé par `find_colours`, et Q est le nombre d'appels à `perform_experiment`.