Connexions de Super-arbres (supertrees)

Gardens by the Bay est un grand parc à Singapour. Dans le parc, il y a n tours, que l'on nomme des "super-arbres". Ces tours sont numérotées de 0 à $n-1$. On veut construire un ensemble de **zéro ou plus** ponts. Chaque pont connecte une paire de tours et peut être traversé dans **les deux** sens. Deux ponts distincts ne peuvent pas connecter la même paire de tours.

Un chemin de la tour x à la tour y est une suite de une tour ou plus telle que :

- le premier élément de la suite est x,
- le dernier élément de la suite est y,
- tous les éléments de la suite sont **distincts**, et
- chaque couple d'éléments (tours) consécutifs dans la suite est connecté par un pont.

Notez que par définition il y a exactement un chemin d'une tour à elle-même, et le nombre de chemins distincts de la tour i à la tour j est le même que le nombre de chemins distincts de la tour j à la tour i.

L'architecte responsable du projet souhaite que les ponts soient construits de manière à ce que pour tous $0 \leq i, j \leq n-1$ il y ait exactement $p[i][j]$ chemins distincts de la tour i à la tour j, où $0 \leq p[i][j] \leq 3$.

Construisez un ensemble de ponts qui satisfait les contraintes de l'architecte, ou déterminez que c'est impossible.

Détails d'implémentation

Vous devez implémenter la fonction suivante :

```
int construct(int[][] p)
```

- p : un tableau $n \times n$ représentant les contraintes de l'architecte.
- Si une construction est possible, cette fonction doit effectuer exactement un appel à `build` (voir ci-dessous) pour détailler la construction, après quoi elle doit renvoyer 1.
- Sinon, la fonction doit renvoyer 0 sans appeller `build`.
- Cette fonction est appelée exactement une fois.

La fonction `build` est définie ainsi :

```
void build(int[][] b)
```

- b : un tableau $n \times n$, où $b[i][j] = 1$ si il y a un pont entre la tour i et la tour j, ou $b[i][j] = 0$ sinon.
- Notez que le tableau doit satisfaire $b[i][j] = b[j][i]$ pour tous $0 \leq i, j \leq n - 1$ et $b[i][i] = 0$ pour tout $0 \leq i \leq n - 1$.

Exemples

Exemple 1

Considérons l'appel suivant :

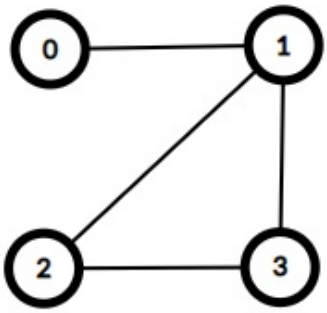
```
construct([[1, 1, 2, 2], [1, 1, 2, 2], [2, 2, 1, 2], [2, 2, 2, 1]])
```

Cela signifie qu'il doit y avoir exactement un chemin de la tour 0 à la tour 1. Pour toutes les autres paires de tours (x, y), telles que $0 \leq x < y \leq 3$, il doit y avoir exactement deux chemins de la tour x à la tour y.

Cela peut être réalisé avec 4 ponts, connectant les paires de tours $(0, 1)$, $(1, 2)$, $(1, 3)$ et $(2, 3)$.

Pour détailler cette solution, la fonction `construct` doit faire l'appel suivant :

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



Elle doit ensuite renvoyer 1.

Dans cet exemple, il y a plusieurs constructions qui respectent les contraintes, toutes seraient considérées comme valides.

Exemple 2

Considérons l'appel suivant :

```
construct([[1, 0], [0, 1]])
```

Cela signifie qu'il ne doit y avoir aucun moyen de se rendre d'une tour à l'autre. Cette contrainte peut être respectée en ne construisant aucun pont.

Par conséquent, la fonction `construct` doit faire l'appel suivant :

- `build([[0, 0], [0, 0]])`

Après quoi la fonction `construct` doit renvoyer 1.

Exemple 3

Considérons l'appel suivant :

```
construct([[1, 3], [3, 1]])
```

Cela signifie qu'il doit y avoir exactement 3 chemins de la tour 0 à la tour 1. Cet ensemble de contraintes ne peut pas être respecté. Par conséquent, la fonction `construct` doit renvoyer 0 sans appeler `build`.

Contraintes

- $1 \leq n \leq 1000$
- $p[i][i] = 1$ (pour tout $0 \leq i \leq n-1$)
- $p[i][j] = p[j][i]$ (pour tous $0 \leq i, j \leq n-1$)
- $0 \leq p[i][j] \leq 3$ (pour tous $0 \leq i, j \leq n-1$)

Sous-tâches

1. (11 points) $p[i][j] = 1$ (pour tous $0 \leq i, j \leq n-1$)
2. (10 points) $p[i][j] = 0$ ou 1 (pour tous $0 \leq i, j \leq n-1$)
3. (19 points) $p[i][j] = 0$ ou 2 (pour tous $i \neq j, 0 \leq i, j \leq n-1$)
4. (35 points) $0 \leq p[i][j] \leq 2$ (pour tous $0 \leq i, j \leq n-1$) et il y a au moins une construction qui satisfait les contraintes.
5. (21 points) $0 \leq p[i][j] \leq 2$ (pour tous $0 \leq i, j \leq n-1$)
6. (4 points) Pas de contraintes supplémentaires.

Évaluateur d'exemple

L'évaluateur d'exemple lit l'entrée au format suivant :

- ligne 1 : n
- ligne $2+i$ ($0 \leq i \leq n-1$) : $p[i][0]$ $p[i][1]$ $\ldots$ $p[i][n-1]$

La sortie de l'évaluateur d'exemple est au format suivant :

- ligne 1 : la valeur que renvoie `construct`.

Si la valeur renvoyée par `construct` est 1, l'évaluateur d'exemple affiche aussi :

- ligne $2 + i$ ($0 \leq i \leq n - 1$) : $b[i][0]$ $b[i][1]$ $\ldots$ $b[i][n-1]$