Plan d'aventure

Vous vous préparez à une aventure. Cette aventure est un voyage du sommet de départ au sommet d'arrivée, se déroulant sur un graphe acyclique orienté (DAG). Le DAG comporte n sommets numérotés de 0 à $n-1$ et m arcs. Le sommet de départ est le sommet 0 , et tous les sommets sont accessibles dès le départ.

Chaque arc de u_i à v_i possède deux attributs : l_i et r_i . Ces attributs indiquent les temps minimum et maximum requis pour le franchir en toute sécurité. Par conséquent, vous pouvez définir le temps prévu pour cet arc comme n'importe quel entier t_i dans l'intervalle $[l_i, r_i]$.

Beaucoup de vos amis se préparent également pour l'aventure. Chacun empruntera un itinéraire différent du début à la fin. Pour assurer la sécurité, vous espérez que les amis empruntant des itinéraires différents puissent se rencontrer simultanément à chaque sommet. Autrement dit, vous souhaitez définir le temps prévu pour chaque arc de telle sorte que pour chaque sommet u , tous les chemins du début à u aient le même temps total.

On dit qu'un graphe est sûr s'il satisfait à l'exigence. Il est garanti que le graphe initial est sûr.

Tâche 1

Vous souhaitez ajouter de nouveaux arcs au graphe pour le rendre plus intéressant. Vous effectuerez q opérations d'« ajout de nouveaux arcs ». Chaque opération fournit un nouvel arc (u_i, v_i, l_i, r_i) . Vous savez qu'après l'ajout de cet arc, le graphe reste un graphe acyclique orienté, mais vous n'êtes pas certain qu'il soit toujours fiable. Veuillez vérifier si le graphe est fiable après l'ajout de cet arc. Si c'est le cas, ajoutez l'arc au graphe. Sinon, ignorez cette opération.

Tâche 2

Après toutes les opérations, vous devez indiquer le temps prévu pour chaque arc (y compris ceux nouvellement ajoutées) pour prouver que vous êtes sûr que le nouveau graphe est sûr.

Détails de mise en œuvre

Vous devez mettre en œuvre deux procédures.

La première procédure que vous devez implémenter est `add_roads` :

```
boolean[] add_roads(int32 N, int32 M, int32 Q,
    int32[] U, int32[] V,
    int32[] L, int32[] R,
    int32[] U2, int32[] V2,
    int32[] L2, int32[] R2);
```

- N : le nombre de sommets ;
- M : le nombre d'arcs initial ;
- Q : le nombre d'opérations ;
- U , V : tableaux de longueur M , où $(U[i], V[i])$ représente la i -ième arc ;
- L , R : tableaux de longueur M , où $[L[i], R[i]]$ est l'intervalle de temps réalisable pour l'arc i ;
- $U2$, $V2$, $L2$, $R2$: tableaux de longueur Q , décrivant les nouveaux arcs ;
- Cette procédure est appelée exactement une fois pour chaque cas de test au début du programme.

La procédure doit renvoyer un vecteur de longueur Q , où le i -ème élément est `true` si le i -ème bord d'opération est ajouté, ou `false` sinon.

La deuxième procédure que vous devez implémenter est `assign_times` :

```
int32[] assign_times();
```

- Cette procédure est appelée exactement une fois pour chaque cas de test après l'appel de `add_roads` .

La procédure doit renvoyer un vecteur contenant le temps prévu t_i pour chaque arc présent dans le graphe final (dans n'importe quel ordre valide).

Contraintes

- $3 \leq n \leq 500$
- $n - 1 \leq m \leq 10^5$
- $0 \leq q \leq 500$
- $0 \leq u_i < v_i < n$
- $1 \leq l_i \leq r_i \leq 10^9$

Sous-tâches

1. Sous-tâche 1 (7 points) : $n \leq 3$
2. Sous-tâche 2 (21 points) : $q = 0$
3. Sous-tâche 3 (12 points) : $v_i = u_i + 1$
4. Sous-tâche 4 (11 points) : $l_i = r_i$

5. Sous-tâche 5 (24 points) : $n \leq 100, m \leq 100, q \leq 100$
6. Sous-tâche 6 (25 points) : Aucune contrainte supplémentaire

Exemples

Exemple 1 :

Considérez l'appel suivant.

```
add_roads(4, 4, 2,
      [0, 1, 0, 0],
      [1, 3, 3, 2],
      [1, 3, 9, 6],
      [5, 7, 14, 8],
      [2, 2],
      [3, 3],
      [7, 5],
      [11, 7]);
```

La procédure doit renvoyer `[false, true]`.

Considérez l'entrée suivante après l'appel de `add_roads`.

```
assign_times();
```

La procédure doit renvoyer `[5, 7, 12, 6, 6]`.

Grader

Le Grader lit l'entrée dans le format suivant :

- Ligne 1 : Trois entiers n , m et q
- m ;ignes suivantes : quatre entiers u_i, v_i, l_i, r_i décrivant chaque arc
- q lignes suivantes : quatre entiers u_i, v_i, l_i, r_i décrivant chaque opération