Lëtzebuerger Informatiksolympiad 2015

Finale

Problème I OFFICES DE TOURISME 50 points

Tâche

La fédération des offices du tourisme souhaite augmenter l'attractivité de la région par l'amélioration de l'offre en chemins pour randonnées en vélo au niveau de chaque office membre.

Sur le territoire de chaque office, la fédération a identifié un certain nombre de points d'intérêt dont elle souhaiterait qu'ils soient reliés par des liaisons directes pour randonnées en vélo. Chaque liaison directe (« arête » au sens d'un graphe) doit relier deux points d'intérêt et est praticable dans les deux sens.

La fédération souhaite qu'il soit possible de démarrer à n'importe lequel des points d'intérêt d'un office membre et de les visiter tous au moins une fois tout en parcourant chaque liaison directe une et une seule fois. En plus la fédération désire qu'il soit toujours possible de revenir vers le point de départ en fin de visite. Remarquons que cela est seulement possible si chaque point d'intérêt peut être atteint par un nombre pair de liaisons directes.

Écrire un programme qui détermine pour chaque office membre combien de liaisons directes doivent au minimum être ajoutées aux liaisons directes existantes pour remplir les conditions souhaitées, sachant qu'il est admis d'ajouter des liaisons directes qui relient deux points d'intérêt qui sont déjà reliés par une liaison directe existante.

Restrictions

Les restrictions suivantes sont à considérer :

- le nombre d'offices membres M : $1 \leq M \leq 100$, numérotés de 1 à M ;
- le nombre de points d'intérêt P : $2 \leq P \leq 1000$, numérotés de 1 à P ;
- le nombre de liaisons directes pour randonnée en vélo R : $2 \leq R \leq 5000$.

Entrée du programme

Principe

La première donnée est un nombre entier positif représentant M, le nombre d'offices membres de la fédération.

Pour chaque office membre on a :

- un couple de deux nombres entiers dont le premier désigne le nombre de points d'intérêt P et dont le deuxième désigne le nombre de liaisons directes pour randonnée en vélo R ;

- *R* couples de nombres entiers *A* et *B* désignant chacun une liaison directe existant déjà entre les deux points d'intérêt ayant comme numéro respectivement *A* et *B*. Comme *A* et *B* désignent des numéros de points d'intérêt, on sait que $1 \leq A, B \leq 1000$. On peut supposer pour chaque couple que le nombre *A* est différent du nombre *B* et que pour le même office membre tout couple de nombres *A* et *B* est unique.

Exemple

	Explications
3	On traitera trois offices membres
3 3	Il y a trois points d'intérêt et trois liaisons directes
1 2	Il existe une liaison directe entre les points d'intérêt 1 et 2
2 3	Il existe une liaison directe entre les points d'intérêt 2 et 3
1 3	Il existe une liaison directe entre les points d'intérêt 1 et 3
3 1	Il y a trois points d'intérêt et une liaison directe
1 2	Il existe une liaison directe entre les points d'intérêt 1 et 2
4 2	Il y a quatre points d'intérêt et deux liaisons directes
1 2	Il existe une liaison directe entre les points d'intérêt 1 et 2
3 4	Il existe une liaison directe entre les points d'intérêt 3 et 4

Sortie du programme

Pour chaque office membre le programme retourne :

- dans une première ligne le texte « Office membre » suivi du numéro de l'office membre et du symbole « : » ;
- dans une deuxième ligne le nombre de liaisons directes qui doivent au minimum être ajoutées.

Exemple (correspondant aux données de l'exemple ci-dessus)

```
Office membre 1:
0
Office membre 2:
2
Office membre 3:
2
```

Remettez le programme sous le nom TOURISME.*xxx*, avec *xxx*=PAS ou C(PP) ou JAVA.
Le programme doit être développé comme application console exécutable par
TOURISME < *input* > *output* (java -jar "TOURISME.jar" < *input* > *output*),
***input* et *output* désignant des fichiers de type txt dont le nom peut être quelconque.**

Problème II RESEAUX 50 points

Tâche

Dans une grande entreprise, un ingénieur est chargé d'interconnecter un très grand nombre d'ordinateurs et vous travaillez en tant que son assistant.

À chaque fois que l'ingénieur intègre un ordinateur dans un réseau, il vous informe qu'il vient de connecter un ordinateur *X* à un ordinateur *Y*. De plus, s'il a un doute, il vous demande de vérifier si un ordinateur *X* est bien relié à un ordinateur *Y*. Comme les moyens de communication entre lui et vous sont limités, vous ne lui fournissez la liste de ces informations qu'à la fin de son travail et ce d'une manière aussi succincte que possible.

Pour passer le temps et l'aider dans sa tâche, vous imaginez une solution qui permet de faire deux choses : saisir qu'un ordinateur vient d'être connecté à un autre et répondre à des requêtes pour savoir si un ordinateur est bien connecté à un autre. Comme vous êtes consciencieux, vous réalisez un tel programme de manière à ce qu'il traite plusieurs jeux de données en une exécution.

Votre travail consiste à écrire un programme qui affiche pour chaque jeu de données un nombre décimal synthétisant toutes les requêtes demandées par l'ingénieur. Le $i^{ème}$ bit de ce nombre représente la réponse à la $i^{ème}$ requête (1 si la réponse est oui et 0 autrement).

Entrée du programme

- L'ensemble des données débute par un nombre entier $J > 0$ représentant le nombre de jeux de données à traiter. J est inférieur à 100.
- Chaque jeu de données débute par une ligne vide suivie d'une ligne contenant un seul nombre entier $N > 0$ représentant le nombre d'ordinateurs à prendre en compte. N est inférieur à 10000.
- N est suivi d'un certain nombre de lignes (éventuellement 0) débutant par une lettre « c » ou « r » et suivie de deux nombres entiers n_1 et n_2 représentant chacun un ordinateur. n_1 et n_2 sont différents et compris entre 1 et N. La lettre « c » indique qu'une connexion a été mise en place entre les ordinateurs n_1 et n_2 tandis que la lettre « r » formule une requête pour savoir si une connexion existe bien entre n_1 et n_2.
- La réponse d'une requête se base toujours sur les connexions mises en place au moment de la requête !
- Si une connexion existe entre deux ordinateurs n_1 et n_2 elle l'est toujours dans les deux sens, c'est-à-dire de n_1 vers n_2 et de n_2 et n_1.
- Si une connexion existe entre deux ordinateurs n_1 et n_2 et qu'une connexion existe entre n_2 et n_3 alors une connexion est établie entre n_1 et n_3.

- Le nombre de requêtes fournissant une réponse positive (égale à 1) dans un jeu de données ne dépasse pas 32.

Sortie du programme

- Pour chaque jeu de données, le programme affiche un seul nombre décimal dont la représentation binaire fournit les réponses à toutes les requêtes. Ce nombre est suivi d'une ligne vide.
- Le $i^{ème}$ bit de ce nombre est à 1 si la $i^{ème}$ requête indique que les deux ordinateurs en cause sont bien interconnectés et 0 autrement.

Exemple d'exécution

Entrée

```
2
```
(Ce fichier d'entrée contient 2 jeux de données.)

```
200
c 1 2
r 1 2
c 100 200
r 200 1
c 100 1
r 200 1
r 200 2

6
c 5 2
r 1 2
```

Sortie

Le résultat livré par le programme pour le fichier de données précédent est

```
11

0
```

11 en binaire est 1011 : la première requête renvoie 1, la deuxième requête 0, la troisième 1 puisque à ce moment l'ordinateur 200 est connecté à l'ordinateur 1 en passant par l'ordinateur 100 et la dernière renvoie aussi 1 comme 200 est relié à 2 via les ordinateurs 100 et 1.

Remettez le programme sous le nom RESEAUX.*xxx*, avec *xxx*=PAS ou C(PP) ou JAVA.

Le programme doit être développé comme application console exécutable par RESEAUX < *input* > *output* (java -jar "RESEAUX.jar" < *input* > *output*), *input* et *output* désignant des fichiers de type txt dont le nom peut être quelconque.