Olympiades Régionales d'Informatique Franco-Australiennes

9 mars 2014

Durée : 4 heures

3 problèmes

Problème 1
Administration système réseau

Fichier d'entrée : *entrée standard*
Fichier de sortie : *sortie standard*

Limites de temps et de mémoire : 0.5 secondes, 128 MB

Vous êtes l'Administrateur Système Réseau (ASR) d'une petite école. Récemment embauché après le défaçage catastrophique du site web de l'école, vous êtes responsable de sa sécurisation.

L'administration de l'école a accepté de vous donner accès au flux des *méta-donées* des emails des étudiants (l'heure, l'expéditeur et le destinataire de chaque email envoyé à travers le réseau), bien que cela ait posé problème à l'ensemble des étudiants pour des raisons de respect de leur vie privée.

Afin de justifier à l'administration scolaire la permission qui vous a été donnée, vous devez écrire un programme montrant comment les méta-données pourraient être utiles dans le cas d'une *mule de Troie* sur le réseau. Vous tenez de votre formation d'expert en administration système réseau qu'une mule de Troie est un virus informatique semblable à un cheval de Troie, sauf qu'il ne se reproduit pas.

Une mule de Troie commence sur un ordinateur initial puis s'attache elle-même au premier email envoyé depuis cet ordinateur. Après cela, elle se supprime de l'ordinateur de l'expéditeur et infecte l'ordinateur du destinataire, puis attend le prochain envoi d'email, etc. Ainsi, il n'existe toujours qu'une seule copie du virus sur le réseau.

Votre programme doit être capable de répondre aux requêtes de la forme « si une mule de Troie était initialement sur l'ordinateur X, sur quel ordinateur serait-elle maintenant ? » Ces requêtes peuvent être entrecoupées de l'arrivée de nouvelles métadonnées.

Entrée

- La première ligne de l'entrée contient une deux entiers séparés par une espace, N et M, respectivement le nombre d'ordinateurs et le nombre de lignes restantes de l'entrée.
- Les M lignes suivantes ont chacune l'une des deux formes suivantes :
  1. `E` x y ($1 \leq x, y \leq N$ et $x \neq y$) si un nouvel email a été envoyé de l'ordinateur x à l'ordinateur y ;
  2. `Q` x ($1 \leq x \leq N$) : une requête demandant la position actuelle du virus s'il était initialement sur l'ordinateur x.

Sortie

Votre programme doit afficher une ligne pour chaque requête (chaque ligne de la forme `Q` x dans l'entrée) contenant un entier identifiant l'ordinateur que le virus infecterait actuellement s'il avait commencé sur l'ordinateur x.

(suite à la page suivante . . .)

Exemple d'entrée

```
5 8
E 1 2
E 3 4
Q 4
E 2 3
E 3 5
Q 1
E 5 2
Q 2
```

Exemple de sortie

```
4
5
2
```

Explications

Il y a 5 ordinateurs connectés au réseau de l'école.

1. Un email est envoyé de 1 à 2.
2. Un email est envoyé de 3 à 4.
3. On demande quel ordinateur serait infecté si la mule de Troie était initialement sur l'ordinateur 4. Comme aucun email n'a encore été envoyé depuis l'ordinateur 4, la mule de Troie devrait encore être sur l'ordinateur 4.
4. Un email est envoyé de 2 à 3.
5. Un email est envoyé de 3 à 5.
6. On demande quel ordinateur serait infecté si la mule de Troie était initialement sur l'ordinateur 1. La mule de Troie a été transmise de l'ordinateur 1 à l'ordinateur 2 (ligne 1), puis de l'ordinateur 2 au 3 (ligne 4) et enfin de l'ordinateur 3 au 5 (ligne 5). La mule de Troie devrait donc être sur l'ordinateur 5.
7. Un email est envoyé de 5 à 2.
8. On demande quel ordinateur serait infecté si la mule de Troie était initialement sur l'ordinateur 2. La mule de Troie est passée de l'ordinateur 2 à l'ordinateur 3 (ligne 4), puis de l'ordinateur 3 à l'ordinateur 5 (ligne 5) et pour finir est revenue de l'ordinateur 5 à l'ordinateur 2 : la mule de Troie devrait donc se trouver sur l'ordinateur 2.

Sous-tâches & Contraintes

Pour chaque sous-tâche, $1 \leq N \leq 100\,000$ et $1 \leq M \leq 100\,000$.

- Pour la première sous-tâche (10 points), $1 \leq N \leq 100$ et $1 \leq M \leq 100$.
- Pour la deuxième sous-tâche (25 points), $1 \leq N \leq 100$ et $1 \leq M \leq 50\,000$.
- Pour la troisième sous-tâche (30 points), aucun ordinateur n'envoie ni ne reçoit d'email après en avoir envoyé un. C'est-à-dire qu'une fois que l'identifiant d'un ordinateur apparaît comme un x dans une ligne commençant par un `E`, il n'apparaîtra plus dans aucune ligne commençant par un `E`.
- Pour la quatrième sous-tâche (35 points), il n'y a aucune contrainte supplémentaire.

Problème 2
Connexions

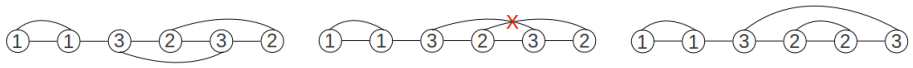
Fichier d'entrée : *entrée standard*
Fichier de sortie : *sortie standard*

Limites de temps et de mémoire : 1 seconde, 128 MB

Vous construisez un circuit électronique vital pour l'xPhone 6, un nouveau smartphone révolutionnaire, plus mince de 2.2mm et ayant des coins 5.4% moins arrondis que le désormais obsolète et inutile xPhone 5.

Le circuit électronique que vous construisez a la forme suivante :

- Il y a $2N$ nœuds sur un fil principal (un « bus »), reliés électriquement entre eux.
- Chaque nœud doit être connecté à exactement un autre nœud par un nouveau fil que vous devez placer. Les $2N$ nœuds portent chacun des numéros allant de 1 à N. Chaque numéro est porté par exactement deux nœuds et chaque nœud doit être relié à celui portant le même numéro.
- Le tout doit être placé sur un circuit imprimé plat : **les fils ne peuvent pas se croiser**. Chaque fil reliant deux nœuds doit être soit totalement au-dessus, soit totalement au-dessous du fil principal.



L'image de gauche montre une configuration correcte. L'image du milieu montre une configuration incorrecte. L'image de droite montre une configuration correcte dans laquelle tous les fils passent au-dessus du fil principal. Notez que pour les exemples de gauche et du milieu, il n'existe pas de configuration correcte dans laquelle les fils passent tous au-dessus du fil principal.

Vous devez écrire un programme qui détermine s'il est possible de construire un circuit respectant les contraintes données. Si oui, vous devez préciser pour chaque paire de nœuds si le fil avec lequel vous les connectez passe au-dessus ou au-dessous du fil principal.

Entrée

- La première ligne de l'entrée contient un entier N, le nombre de paires de nœuds.
- Les $2N$ lignes suivantes contiennent chacune un entier L_i compris entre 1 et N, le numéro porté par le $i^{\text{ème}}$ nœud. Chaque valeur possible pour L_i apparait deux fois.

Sortie

- S'il est impossible de connecter toutes les paires, votre programme doit afficher une ligne contenant `0`.
- Sinon, votre programme doit afficher $2N$ lignes décrivant une configuration possible. La $i^{\text{ème}}$ ligne doit contenir soit `1`, soit `2`. Il doit s'agir d'un `1` si le fil du $i^{\text{ème}}$ nœud passe au-dessus du fil principal et un `2` s'il passe au-dessous.

Exemple d'entrée 1

```
3
1
1
3
2
3
2
```

Exemple de sortie 1

```
1
1
2
1
2
1
```

Exemple d'entrée 2

```
3
1
1
3
2
2
3
```

Exemple de sortie 2

```
1
1
1
1
1
1
```

Explications

Le premier exemple correspond aux diagrammes de gauche et du milieu. Le second exemple correspond au diagramme de droite.

Sous-tâches & Contraintes

Pour chaque sous-tâche, $1 \leq N \leq 100\,000$.

- Pour la sous-tâche 1 (20 points), $1 \leq N \leq 100$ et soit il est impossible de respecter les contraintes, soit il est possible de faire passer tous les fils au-dessus du fil principal (c'est-à-dire que soit `0` est une sortie valide, soit $2N$ lignes contenant `1` est une sortie valide).
- Pour la sous-tâche 2 (20 points), $1 \leq N \leq 100$.
- Pour la sous-tâche 3 (20 points), $1 \leq N \leq 1\,000$.
- Pour la sous-tâche 4 (20 points), soit le circuit est impossible à réaliser, soit il est possible de faire passer tous les fils au-dessus du fil principal (c'est-à-dire que soit `0` est une sortie valide, soit $2N$ lignes contenant `1` est une sortie valide).
- Pour la sous-tâche 5 (20 points), il n'y a pas de contrainte supplémentaire.

Problème 3
Super Maria II : Un autre château

Limites de temps et de mémoire : 1 seconde, 128 Mo

La République Champignonesque était un havre de paix grâce aux faits de bravoure de notre humble électricienne, Maria, mais tout ceci changea avec l'invasion de son ennemi juré Wowser. Wowser kidnappa (une fois de plus) le prince Nectarine, en le cachant là où – pensait-il – Maria ne pourrait jamais le retrouver.

Mais la vaillante Maria, utilisant le signal du téléphone champignon de Nectarine, réussit à le localiser dans un château parmi N. Les châteaux sont alignés et numérotés de 0 à $N-1$, de gauche à droite.

Maria veut retrouver le prince au plus vite et a loué une calèche à laquelle elle atellera son ami Shoyi le dinosaure. À chaque fois que Maria veut sortir de la calèche ou changer de direction, elle doit donner un pourboire à Shoyi.

Maria commence au château 0, en direction de la droite. Lorsque Maria arrive au château i, elle peut faire l'une de ces trois actions :

- Se déplacer vers n'importe quel château dans sa direction, sans rien dépenser. C'est-à-dire, si elle se dirige vers la gauche, elle peut aller gratuitement vers tout château j tel que $j < i$. Si elle se dirige vers la droite, alors elle peut aller gratuitement vers tout château j tel que $j > i$.
- Payer à Shoyi C_1 pièces pour tourner la calèche dans la direction opposée.
- Payer à Shoyi C_2 pièces pour qu'il la laisse sortir, pour entrer dans le château i. Si Maria décide de faire cela, deux choses peuvent se produire :
  - Elle retrouve Nectarine, bat Wowser et sauve la République Champignonesque.
  - Elle trouve Grenouille, qui lui indique gentiment que le prince est dans un autre château et dans quelle direction (gauche ou droite) elle doit se diriger pour le trouver.

Maria veut minimiser la quantité totale de pièces qu'elle dépense pour retrouver le prince. Elle sort donc son ordinateur et vous envoie un e-mail, demandant votre aide pour écire un programme qui l'aidera à trouver le prince tout en minimisant la quantité totale de pièces qu'elle devra donner à Shoyi.

Entrée / Sortie

Ce problème n'a ni entrée ni sortie. À la place, vous devez utiliser les fonctions de "`carriage_lib.h`" pour obtenir des informations. Les fonctions disponibles sont expliquées en détail à la page suivante.

Bibliothèque

Votre programme doit interagir de cette façon avec les fonctions définies dans "`carriage_lib.h`" :

- Votre programme doit commencer par appeler `carriage_nb_cells()`, `carriage_reverse_cost()`, et `carriage_open_cost()`
  - `carriage_nb_cells()` retourne N, le nombre de châteaux parmi lesquels le prince est enfermé.
  - `carriage_reverse_cost()` retourne C_1, le nombre de pièces que Maria doit payer Shoyi pour changer de direction.
  - `carriage_open_cost()` retourne C_2, le nombre de pièces que Maria doit payer Shoyi pour arrêter la calèche et ouvrir une porte de château.
- Votre programme devra appeler `carriage_move(int k)` et `carriage_open()` pour trouver le château dans lequel le prince est enfermé.
  - `carriage_move(int k)` déplace la calèche d'une distance $|k|$ vers la gauche ou la droite, selon si k est positif ou négatif. Par exemple, appeler cette fonction avec l'argument -5 déplacera la calèche de Maria de 5 cases vers la gauche. Si la calèche allait vers la droite, Maria doit payer Shoyi C_1 pièces pour changer de direction. Si l'argument de cette fonction mène Maria à une position négative ou non-inférieure à N, votre programme s'arrêtera et n'obtiendra aucun points.
  - `carriage_open()` fait payer Maria C_2 pièces à Shoyi pour arrêter la calèche et aller ouvrir la porte du château. Si Maria se trouve au château où le prince Nectarine était caché, votre programme s'arrêtera et vous obtiendrez des points comme décrit dans la section « Score » de la page suivante. Sinon, cette fonction retournera 1 si le prince est dans un château à droite du château actuel, -1 dans le cas contraire.

Prototypes de la bibliothèque

Les signatures de ces fonctions en C et C++ sont les suivantes :

```
int carriage_nb_cells();
int carriage_reverse_cost();
int carriage_open_cost();
void carriage_move(int k);
int carriage_open();
```

Tests

Pour tester votre code sur votre propre machine, téléchargez `carriage_lib.h` dans le même dossier que votre code et ajoutez `#include "carriage_lib.h"` en haut de votre code.

L'exécutable prendra en entrée 4 entiers, séparés par des espaces, au premier appel à une fonction de la bibliothèque. Ces entiers sont N, C_1, C_2 et p (entre 0 et $N-1$) respectivement, où le château d'indice p est celui dans lequel est caché le prince Nectarine.

L'exécutable affichera une ligne sur la sortie standard : soit un message d'erreur à cause d'une entrée ou mouvement invalide, soit une indication de succès incluant le nombre de pièces dépensées.

Exemple d'entrée

```
10 3 4 7
```

Situation type

Appels de fonctions	Explications
`carriage_nb_cells()`	Retourne 10, le nombre de châteaux.
`carriage_reverse_cost()`	Retourne 3, le coût d'un changement de direction.
`carriage_open_cost()`	Retourne 4, le coût d'ouverture d'une porte de château.
`carriage_move(5)`	Se déplace gratuitement vers le château 5.
`carriage_open()`	Retourne 1, car le Prince est dans le château 7, à droite du château 5. Ceci a coûté 4 pièces.
`carriage_move(4)`	Se déplace gratuitement vers le château 9.
`carriage_open()`	Retourne -1, car le Prince est dans le château 7, à gauche du château 9. Ceci a coûté 4 pièces.
`carriage_move(-3)`	Se déplace vers le château 6. Ceci a coûté 3 pièces, car il a fallu changer de direction.
`carriage_open()`	Retourne 1, car le Prince est dans le château 7, à droite du château 6. Ceci a coûté 4 pièces.
`carriage_move(1)`	Se déplace vers le château 7. Ceci a coûté 3 pièces, car il a fallu changer de direction.
`carriage_open()`	Trouve le prince ! Ceci a coûté 4 pièces. Le programme affiche que le prince a été retrouvé en utilisant 22 pièces, puis se termine.

Score & Contraintes

- Dans tout les cas, N, C_1 et C_2 sont des entiers tels que $1 \leq N \leq 1,000,000$ et $0 \leq C_1, C_2 \leq 10,000$.
- Dans 30% des cas, $N \leq 1000$ et $C_1, C_2 \leq 1000$.
- Dans chacun des tests, votre score sera 100 si votre programme trouve le prince avec le nombre minimum de pièces. Sinon, si votre programme trouve le Prince, votre score sera compris entre 0 et 60, selon une fonction linéaire du nombre de pièces utilisées.
- Votre score total pour ce problème sera la moyenne de vos scores sur chacun des tests.
- Notez que le juge utilise une version modifiée de la bibliothèque qui forcera votre programme à utiliser le maximum de pièces possibles dans chacun des cas.