Olympiades Régionales d'Informatique Franco-Australiennes

11 mars 2012

Durée : 4 heures

3 problèmes

Règlement du concours

Le règlement du concours est défini par le comité d'organisation du FARIO. Seul ce comité est responsable de l'interprétation des règles et des sujets du concours, et est responsable de la clarification ou modification des règles ou des sujets du concours, si des modifications s'avéraient nécessaires.

Environment du concours

- Le concours dure quatre heures, et la participation se fait de chez soi, individuellement, de 9h à 13h le dimanche 11 mars 2012.
- Chaque participant ne peut utiliser qu'un seul ordinateur, ainsi que les compilateurs, interpréteurs, débuggers et environnements de développements associés.
- Les participants peuvent utiliser calculatrices, imprimantes, et tout matériel écrit (y compris des morceaux de code imprimés). Les participants peuvent également utiliser la documentation électronique du langage ainsi que les fichiers d'aide.
- Les participants ne peuvent **pas** utiliser de codes sources existants sous leur forme électronique, et ne doivent pas communiquer avec les autres participants.
- Les élèves ne doivent **pas** utiliser Internet pour d'autres raisons que l'accès au site du concours et l'envoi de messages aux organisateurs, ainsi que l'accès à la documentation sur le langage.

Restrictions sur les codes sources

- Les solutions ne seront acceptées qu'en C ou C++.
- Les programmes doivent être mono-tâche, et mono-thread. Toute soumission de code malicieux entraînera une disqualification immédiate.
- Le code source de chaque solution ne doit pas dépasser une taille de 40000 octets.

Soumettre ses solutions

- Le site de soumission sera accessible par un lien sur `http ://www.fario.org/`. Les détails de connexion seront les mêmes que ceux utilisés pour le site d'entraînement de l'AIOC.
- **Les participants doivent soumettre leurs solutions au fur et à mesure.** Le site du concours n'acceptera les soumissions que pendant la durée du concours (4h à partir du moment de la première connexion).
- Une fois les 4 heures passées, **les soumissions suivantes ne seront pas acceptées**. N'attendez pas les dernières minutes pour soumettres vos solutions !
- Lorsqu'une solution est soumise, elle sera mise dans une file d'attente pour son évaluation. Les résultats de l'exécution sur les fichiers d'exemples (tests publics) seront disponibles, tandis que les résultats complets d'une soumission peuvent être obtenus en utilisant un jeton (token). Les participants auront un nombre limité de tokens à utiliser par question, et cette limite sera affichée sur la page de soumission.

Si vous avez des questions concernant le déroulement du concours, ou sur les sujets du concours, envoyez votre question par mail à *fario@orac.amt.edu.au*. Nous répondrons à vos questions aussi vite que possible.

Bonne chance !

Problème 1
Master Chef

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

Limites de temps et de mémoire : 0.5 secondes, 64 Mo

Roderick était, dans le temps, le meilleur manipulateur d'interrupteurs sur terre. Avec une surprenante dextérité et à la vitesse de l'éclair, le talent unique de Rod pour manipuler plusieurs interrupteurs adjacents simultanément lui donnait un énorme avantage pour manipuler des interrupteurs. Cependant, en ces temps de récession, Rod a dû changer de voie. Hier, il lui a été proposé de participer à la fameuse émission de compétition culinaire *Master Chef*, mais il a besoin de votre aide pour savoir s'il a le talent nécessaire pour vraiment dominer la cuisine.

Roderick sera mis face à N ingrédients. Sa stratégie, pour cuisiner rapidement, consiste à sélectionner aléatoirement un ensemble d'aliments qui forment une sous-suite contigüe de la ligne des ingrédients (c'est à dire, tous les ingrédients entre le a^{e} et le b^{e} ingrédient inclus, pour une sélection aléatoire de a et b telle que $1 \leq a \leq b \leq N$). D'un geste rapide, grâce à ses mains énormes, Roderick prendra tous ces ingrédients et les mélangera dans sa casserole. Chaque ingrédient a une valeur gustative T (qui peut être négative, certains ingrédients sont vraiment mauvais), et la valeur gustative du plat final est la *moyenne* des valeurs gustatives des ingrédients sélectionnés. Par exemple, si dans l'exemple suivant, Roderick choisit les ingrédients de valeurs gustatives 3, -1 et 5, la valeur gustative du plat sera $(3 + (-1) + 5)/3 = 2\frac{1}{3}$.

6	3	-1	5

Comme Roderick ne peut pas prévoir à l'avance la sous-séquence qu'il sera amené à choisir le jour de la compétition, il vous demande d'écrire un programme qui calcule la valeur gustative moyenne du plat qu'il créera (c'est à dire la moyenne des valeurs gustatives de tous les repas qu'il pourrait créer). Dans l'exemple ci-dessus, avec 4 ingrédients, il y a 10 sélections d'ingrédients possibles, dont Roderick pourrait faire un plat. Les valeurs gustatives de ces plats seraient : 6, 3, -1, 5, 4.5, 1, 2, 2.6666..., 2.3333... et 3.25. La moyenne des valeurs gustatives de ces 10 plats possibles est 2.875. Comme Roderick n'aime pas trop les nombres, il vous demande de donner la valeur gustative finale moyenne, **arrondie vers 0 (c'est à dire tronquée)**. Dans l'exemple, la réponse à donner serait donc 2.

Entrée

- La première ligne de l'entrée contiendra un seul entier, N, le nombre d'ingrédients.
- La ligne suivante contiendra N entiers, représentant les valeurs gustatives des ingrédients, de gauche à droite.

Sortie

Vous devez afficher un seul entier sur la sortie : la valeur moyenne (tronquée) des valeurs gustatives de tous les plats possibles, où la valeur gustative d'un repas est la moyenne des valeurs gustatives de la sous-suite contigüe d'ingrédients qui le composent.

Exemple d'entrée 1

```
4
6 3 -1 5
```

Exemple d'entrée 2

```
8
-3 10 19 9 7 10 0 5
```

Exemple d'entrée 3

```
3
-10 -4 1
```

Exemple de sortie 1

```
2
```

Exemple de sortie 2

```
8
```

Exemple de sortie 3

```
-4
```

Explications

Le premier exemple est celui présenté précédemment. Dans le second cas, la valeur moyenne des 36 repas possibles est 8.12, qui est tronquée à 8. Dans le troisième cas, on tronque -4.30556 à -4.

Contraintes & Sous-tâches

Dans toutes les sous-tâches, la valeur gustative de chaque ingrédient sera un entier entre -1000 et 1000 inclus.

- Sous-tâche 1 (20 points) : $1 \leq N \leq 200$
- Sous-tâche 2 (20 points) : $1 \leq N \leq 2000$
- Sous-tâche 3 (60 points) : $1 \leq N \leq 200000$

Vous obtiendrez l'ensemble des points associés à une sous-tâche si votre programme retourne la réponse correcte pour chacun des tests de cette sous-tâche. Si votre programme échoue un quelconque test d'une sous-tâche, votre score sera de 0 pour cette sous-tâche.

Problème 2
Happy Feet

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

Limites de temps et de mémoire : 0.5 secondes, 64 Mo

Oubliez les vols spatiaux, les voyages en sous-marin et les randonnées dans le desert. Le nouveau tourisme à la mode consiste à observer les célèbres pingouins danceurs de l'Antartique. En tant que créateur d'une nouvelle entreprise vendant ses célèbres randonnées dans le continent de l'extrême sud, votre idée géniale consiste à déterminer la randonnée parfaite au milieu des colonies de pingouins, celle qui enchantera le plus les riches touristes.

Après de longues discussions avec des experts des pingouins de l'Antartique, vous avez mis au point une carte de tous les sentiers de la région qui peuvent être empruntés par un être humain. Chaque sentier relie un croisement à un autre croisement, et peut être parcouru dans les deux sens. Notez qu'un sentier peut relier un croisement à lui-même, et qu'il peut y avoir plusieurs sentiers entre deux croisements donnés. Les experts vous ont donné un décompte précis du nombre de pingouins qui peuvent être vus en train de danser le long de chacun des sentiers. Vous souhaitez trouver un trajet qui permette de voir le plus grand nombre de pingouins possible au cours de la randonnée. Ce trajet peut démarrer à n'importe quel croisement et se finir à n'importe quel croisement (y compris le croisement de départ). Il peut également repasser plusieurs fois par le même croisement. La seule condition est que **chaque sentier du trajet doit avoir un nombre de pingouins strictement plus grand que celui du sentier précédent**. En effet, un complexe psychologique étrange des touristes vous force à toujours aller au delà de leurs attentes. Ne pas réussir cela entraînerait une avalanche catastrophique de critiques négatives.

Etant donnée la carte des sentiers, écrivez un programme qui trouve le nombre maximum de pingouins qu'il est possible de voir sur un trajet de ce type.

Entrée

- La première ligne de l'entrée contient deux entiers, N et M, représentant respectivement le nombre de croisements et le nombres de sentiers.
- Les M lignes suivantes contiennent chacune trois entiers a_i, b_i, et p_i représentant un sentier. Le sentier relie les croisements a_i et b_i et on peut y voir p_i pingouins en train de danser. Les croisements sont numérotés de 1 à N, c'est-à-dire que $1 \leq a_i, b_i \leq N$.

Sortie

Le sortie doit consister en un unique entier, le nombre maximum de pingouins qu'il est possible de voir le long d'un chemin satisfaisant les conditions décrites.

Exemple d'entrée

```
5 9
1 2 4
2 3 2
1 3 1
1 4 3
4 3 4
4 3 5
3 5 6
4 5 6
5 5 7
```

Exemple de sortie

```
26
```

Explications

Dans l'exemple ci-dessus, le meilleur trajet commence au croisement 3 et se termine au croisement 5, en passant par les sentiers suivants :

Depuis	Vers	Pingouins vus
3	1	1
1	4	3
4	3	4
3	4	5
4	5	6
5	5	7

Sur ce trajet, on a un total de $1 + 3 + 4 + 5 + 6 + 7 = 26$ pingouins. Aucun trajet valide n'existe avec un plus grand nombre de pingouins. La bonne réponse est donc 26.

Sous-tâches & Contraintes

Dans chacune des sous-tâches, $1 \leq N \leq 1,000$ et chaque sentier aura entre 1 et $1,000,000$ (inclus) pingouins.

- Sous-tâche 1 (30 points) : $1 \leq M \leq 1,000$.
- Sous-tâche 2 (40 points) : $1 \leq M \leq 1,000,000$ et on vous garantit que deux sentiers différents auront toujours un nombre différent de pingouins (c'est-à-dire que les nombres de pingouins des sentiers sont distincts les uns des autres)
- Sous-tâche 3 (30 points) : $1 \leq M \leq 1,000,000$.

Vous obtiendrez l'ensemble des points associés à une sous-tâche si votre programme retourne la réponse correcte pour chacun des tests de cette sous-tâche. Si votre programme échoue un quelconque test d'une sous-tâche, votre score sera de 0 pour cette sous-tâche.

Problème 3
Blackout

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

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

Cette fois-ci, tout avait été prévu dans les moindres détails et chaque système dupliqué : deux clôtures électriques indépendantes, deux centres de contrôle, deux lignes électriques reliées à deux centrales différentes et enfin deux puces GPS placées sur chaque bête. Rien de sérieux ne pouvait se produire avec une telle sécurité et une telle redondance.

Et pourtant ce qui ne pouvait arriver s'est produit : une grève nationale affecte en ce moment toutes les centrales à énergie non-renouvelable du pays. Le gouvernement vient en effet d'annoncer son intention d'atteindre 100% d'énergie renouvelable d'ici cinq ans. Le blackout risque de s'éterniser et les générateurs de secours ne vont pas tenir plus de quelques heures. Après quoi, les Tyrannosaures Rex du parc flambant neuf seront libres d'aller explorer les zones résidentielles voisines.

Vous avez été sélectionné pour participer à la mise en place d'un plan d'urgence. Les paléontologistes ont récemment découvert que les T-Rex n'aiment pas la nourriture salée et ont conjecturé que le sel peut être utilisé comme répulsant pour T-Rex. Le parc a été conçu sous forme d'une grille $S \times S$ divisée en cases carées. Le plan consiste à utiliser des hélicoptères pour larguer de l'eau de mer sur certaines cases de terre de façon à ce que les T-Rex les évitent et ne sortent pas du parc. Votre travail consiste à écrire un programme qui détermine où larguer de l'eau de mer toutes les heures après le blackout afin de minimiser le nombre de T-Rex qui réussiront à s'évader.

Les cases du parc sont de deux types : des cases vierges dans lesquelles les T-Rex peuvent entrer et les cases obstacles dans lesquelles les T-Rex ne peuvent pas entrer. Les traqueurs GPS vous donneront la dernière position connue des T-Rex avant que les générateurs ne s'éteignent. Vous ne serez plus en mesure de suivre les déplacements des T-Rex par la suite mais vous savez qu'un T-Rex peut, à chaque heure, se déplacer sur l'une des 4 cases adjacentes à sa position précédente (si celle-ci n'est pas un obstacle et n'a pas été aspergée d'eau de mer). Toutes les heures, vous pouvez larguer de l'eau sur W cases pour empêcher de façon permanente les T-Rex d'y accéder.

Entrée

- La première ligne de l'entrée contient les trois entiers S, W et T. S est le nombre de lignes et de colonnes, W est le nombre de cases que vous pouvez asperger chaque heure et T est le nombre de T-Rex dans le parc.
- Chacune des S lignes suivantes contient S entiers séparés par des espaces, 0 représentant une case vierge et 1 une case d'obstacle.
- Les T lignes qui suivent contiennent chacune deux entiers C_i et R_i ($0 \leq C_i, R_i < S$) qui représentent la colonne et la ligne de la dernière position connue d'un T-Rex avant le blackout. Les lignes sont numérotées à partir de 0 depuis le haut et les colonnes à partir de 0 depuis la gauche (la position "0 0" désigne donc la case en haut à gauche).

Sortie

- La première ligne doit contenir un entier D qui représente le nombre total de cases aspergées. Vous ne pouvez pas larguer d'eau plus d'une fois sur chaque case (ce serait inutile). $D \leq S^2$.
- Chacune des D lignes qui suivent doit contenir deux entiers DC_i et DR_i ($0 \leq DC_i, DR_i < S$) qui représentent les coordonnées de la case sur laquelle l'hélicoptère doit larguer de l'eau de mer. Les coordonnées doivent être données dans l'ordre de larguage. Ainsi les W premières lignes correspondent aux cases aspergées la première heure, les W suivante la seconde heure, et ainsi de suite.

Exemple d'entrée

```
5 2 1
0 1 0 0 0
0 0 0 0 0
0 0 0 1 0
0 0 0 0 0
0 1 1 1 0
2 3
```

Exemple de sortie

```
3
1 3
2 2
4 3
```

Explication

Le parc est une grille 5×5 avec un seul T-Rex. Vous pouvez larguer de l'eau sur 2 cases chaque heure. La grille est représentée ci-dessous avec T la position initiale du T-Rex.

```
0 1 0 0 0
0 0 0 0 0
0 0 0 1 0
0 0 T 0 0
0 1 1 1 0
```

Une façon d'empêcher le T-Rex de s'enfuir du parc est de larguer de l'eau sur deux cases la première heure : à sa gauche en $(1, 3)$ et au dessus en $(2, 2)$. Le T-Rex ne pourra alors que rester dans la même case ou se déplacer vers la droite. La situation à la fin de la première heure est représentée ci-dessous, W représente les cases aspergées et T les positions possibles du T-Rex.

```
0 1 0 0 0
0 0 0 0 0
0 0 W 1 0
0 W T T 0
0 1 1 1 0
```

En lâchant ensuite de l'eau sur la case $(4, 3)$, on s'assure que le T-Rex ne pourra pas quitter la zone de deux cases dans laquelle il est confiné.

Contraintes & Sous-tâches

- Sous-tâche 1 (20 points) : $S = 5$, $W = 1$, $T = 1$.
- Sous-tâche 2 (20 points) : $S = 10$, $W = 2$, $T = 1$.
- Sous-tâche 3 (20 points) : $S = 15$, $W = 3$, $T = 2$.
- Sous-tâche 4 (20 points) : $S = 20$, $W = 4$, $T = 5$.
- Sous-tâche 5 (20 points) : $S = 30$, $W = 5$, $T = 10$.

Vous obtiendrez la totalité des points d'un test d'une sous-tâche si votre sortie permet de capturer autant ou plus de T-Rex que la sortie de référence, à savoir la meilleure sortie parmi tous les programmes des organisateurs. Autrement, votre score sera calculé ainsi :

$$\frac{100 \times [\text{Nombre de T-Rex capturés}]}{\text{Nombre de T-Rex capturés dans la solution de référence}}$$

Si votre programme écrit une sortie non valide, votre score sera de 0 pour ce test.

Le score total d'une sous-tâche est la **somme** des scores sur chacun de ses tests.