Implémentation de l'algorithme de Gale et Shapley

INF421

2015–2016

Énoncé

Nous vous demandons d'implémenter l'algorithme de Gale et Shapley sous forme d'une classe `StableMatching`. Cette classe doit être placée dans le fichier `StableMatching.java`. Elle doit implémenter l'interface `StableMatchingInterface`, dont la définition est située dans le fichier `StableMatchingInterface.java` que nous vous fournissons.

Dans l'archive qui vous est fournie, le fichier `StableMatching.java` est absent. À vous donc de le créer, de le remplir, et de le déposer. **Ne modifiez pas les autres fichiers.**

Pour que votre devoir soit considéré comme acceptable, nous exigeons deux choses :

1. **Votre code doit être accepté par le compilateur Java.** Aucune erreur de compilation ne doit être signalée. Pour compiler, vous pouvez utiliser Eclipse ou bien la commande `make`.
2. **Votre code doit satisfaire le jeu de tests fourni.** Nous fournissons un certain nombre de fichiers (`StableMatchingTest.java`, `Main.java`, etc.) dont le rôle est de soumettre votre code à une série de tests. Votre code doit passer avec succès l'ensemble de ces tests. Pour lancer les tests, vous pouvez exécuter le programme depuis Eclipse ou bien utiliser la commande `make test`.

Quelques détails

Le jeu de tests contient un certain nombre d'instances du problème des mariages stables. Ces instances sont de diverses tailles. Certaines sont engendrées de façon pseudo-aléatoire ; d'autres sont des cas particuliers fixés. Chacune de ces instances est soumise à votre algorithme. On vérifie alors que votre algorithme termine (une limite de quelques secondes est fixée), ne lance pas d'exception, et produit bien un couplage stable.

Si tout va bien, l'exécution du jeu de tests doit afficher une série de messages semblables à ceux-ci :

```
n = 25: running...
Elapsed time: 0 milliseconds
SUCCESS!

n = 26: running...
Elapsed time: 0 milliseconds
SUCCESS!
```

Si un test échoue, vous obtiendrez un message indiquant quelles données ont été fournies à votre code, quel résultat il a produit, et pourquoi ce résultat n'est pas acceptable. Par exemple, vous pourrez obtenir un message analogue à celui-ci :

```
n = 2: running...
Elapsed time: 0 milliseconds
FAILURE: NOT A STABLE MATCHING!
The pair formed by the man 1 and the woman 0 is unstable.
Indeed, man 1 prefers woman 0 to his bride 1
and woman 0 prefers man 1 to her groom 0.
The parameters of this test run were:
n = 2
menPrefs =
0 prefers: [0, 1]
1 prefers: [0, 1]
womenPrefs =
0 prefers: [1, 0]
1 prefers: [1, 0]
The result of this test run was:
bride =
[0, 1]
```

Les tests les plus simples concernent des instances de petite taille ; les tests les plus difficiles concernent des instances de grande taille.

L'exécution du jeu de tests produit de nombreux messages. Il peut éventuellement être utile de les enregistrer dans un fichier pour les étudier ensuite de façon plus approfondie. La commande `make test > log` redirige les messages vers un fichier nommé `log`. Si vous utilisez Eclipse, il faut ouvrir le menu `Project`, choisir la ligne `Properties`, choisir l'entrée `Run/Debug Settings`, sélectionner une « configuration de lancement » puis cliquer sur le bouton `Edit`, enfin choisir l'onglet `Common` et choisir d'envoyer la sortie standard (« *standard output* ») non pas vers la « console » mais vers un fichier de votre choix.

Par ailleurs, on peut souhaiter que l'exécution du jeu de tests s'arrête dès qu'un test échoue. Vous pouvez obtenir ce comportement en réglant la constante `STOP` dans le fichier `StableMatchingTest.java`.

Conseils généraux

Ne cédez pas à la tentation d'optimiser prématurément votre code. Votre code doit d'abord et surtout être clair et correct. Sa vitesse absolue n'est pas très importante, pourvu qu'il ait la bonne complexité asymptotique, c'est-à-dire $O(n^2)$.

Si besoin, insérez des assertions dans votre code, à l'aide de l'instruction `assert`. Le coût en sera minime, et vous pourrez ainsi détecter plus facilement certaines de vos erreurs.

Si vous utilisez Eclipse, il se peut que vous soyez obligé d'une part d'activer les assertions (parce que, par défaut, l'instruction `assert` n'a aucun effet), d'autre part de régler la taille du tas (parce que les tests que nous avons prévus demandent beaucoup de mémoire). Pour cela, éditez une « configuration de lancement » comme décrit plus haut, puis choisissez l'onglet `Arguments`, et insérez les mots `-ea -Xmx2G` dans le champ intitulé `VM arguments`.

Nous n'exigeons pas que votre code soit commenté, mais bien sûr il peut être utile pour vous d'insérer des commentaires qui expliquent le fonctionnement de l'algorithme.

Ce devoir à la maison (DM) est un travail individuel. S'il est bien sûr permis d'étudier à plusieurs le principe de l'algorithme de Gale et Shapley, nous vous demandons d'écrire seul votre implémentation. Les cas de plagiat seront détectés !