Université Paris Diderot — Outils Logiques
Licence — Année 2011-2012

Hashiwokakero

Projet de Programmation

Objectif

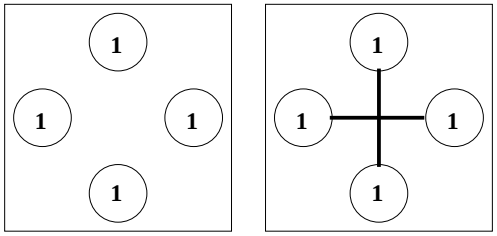
Il s'agit d'implémenter en Java le jeu *Hashiwokakero*[1], décrit ci-dessous, que nous allons abréger en *Hashi* par la suite.

Le jeu Hashi se joue sur une grille rectangulaire sans grandeur standard. On y retrouve des nombres de 1 à 4. Ils sont généralement encerclés et nommés "îles".

Le but du jeu est de relier les îles en créant une série de ponts entre elles, de telle sorte que :

- Tout pont débute et finit sur une île.
- Tous les ponts sont en ligne droite.
- Le nombre de ponts qui débutent ou finissent sur une île est le nombre indiqué sur l'île.

Voici représentés graphiquement un exemple très simple de grille de Hashi, et son unique solution.



En utilisant la terminologie de la théorie des graphes, on dirait qu'une solution est représentée par un graphe :

- *non orienté :* les arêtes (ponts) n'ont pas d'orientation.
- tel que le degré de chaque sommet (île) est l'entier qui étiquette le sommet en question.

Il y a une contrainte supplémentaire, plus difficilement exprimable en utilisant la terminologie de la théorie des graphes, à savoir : *toute arête relie deux sommets consécutifs, c.à.d. n'ayant pas d'autres sommets entre eux, sur une même ligne ou une même colonne de la grille.*

Réalisation du programme

Le programme lit une instance de Hashi à partir d'un fichier. La première ligne du fichier contient les dimensions de la grille, et chacune des lignes suivantes la description d'une île : abscisse, ordonnée, nombre de ponts. Par exemple, un fichier d'entrée décrivant l'instance de Hashi ci-dessus est :

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

Le programme affiche la grille décrite par le fichier d'entrée.

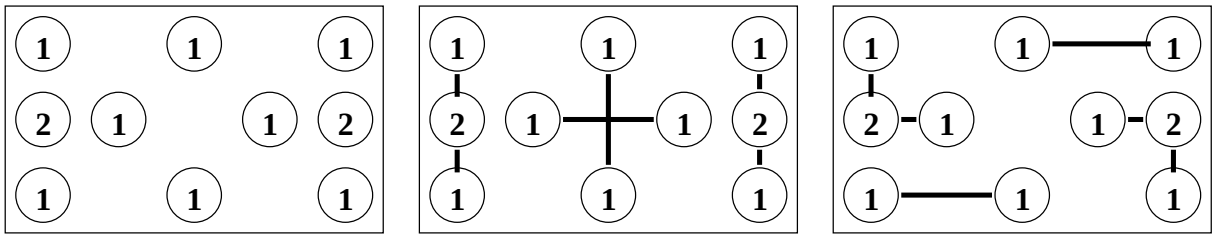
Ensuite, le programme calcule les solutions, s'il en existe, et les affiche. Après l'affichage de chaque solution, l'utilisateur doit pouvoir choisir d'interrompre l'exécution, ou bien de passer à la solution suivante. Une fois toutes les solutions affichées, le programme s'arrête.

1. "Construire des ponts", en japonais.

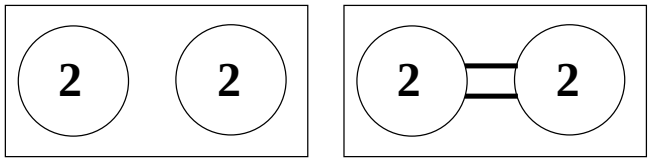
Extensions

Le jeu décrit ci-dessus est une variante du Hashi, simplifiée. Vous êtes censés implémenter au moins cette variante. Les extensions suivantes, optionnelles, peuvent être implémentées, pour améliorer votre note (elles sont indépendantes les unes des autres, et l'ordre proposé n'est pas significatif. Le noyau du jeu donné plus haut, plus les trois extensions ci-dessous, donnent le jeu de Hashi complet).

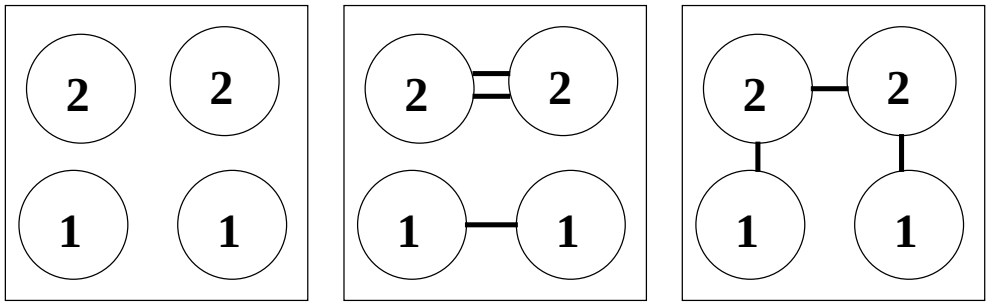
1. Aucun pont ne peut en croiser un autre. Dans l'exemple ci-dessus, la solution proposée ne respecte pas cette contrainte, et il n'y a aucune solution la respectant.
   Voici une instance de Hashi, et deux solutions, dont la deuxième respecte la condition de non croisement des ponts.



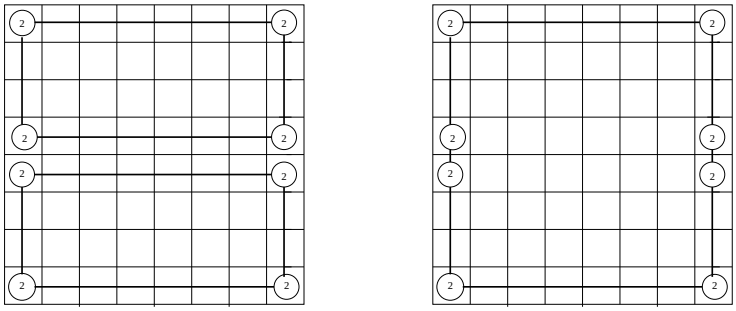
2. Les étiquettes varient entre 1 et 8, et deux îles peuvent être reliées par 1 ou 2 ponts (une solution est donc représentée par un *multi-graphe* dans lequel au plus deux arêtes peuvent relier deux sommets donnés).
   Voici une instance de Hashi qui n'a pas de solutions sans "ponts doubles" :



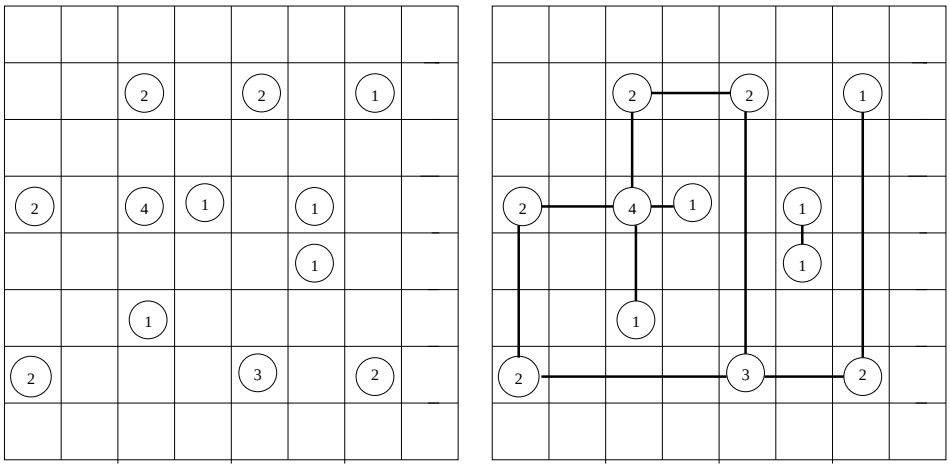
   et en voici une autre qui peut se résoudre avec ou sans ponts doubles :



3. Toute île doit être joignable à partir de n'importe quelle autre île (la solution doit être un graphe *connexe*).
   Voici une solution non connexe et une solution connexe d'une même instance de Hashi :



Voici une grille un peu plus compliquée, et son unique solution, non connexe.



Le logiciel `minisat` Pour de grandes grilles, les formules qui encodent les contraintes sont trop grandes pour pouvoir être manipulées à la main. Il faudra donc utiliser un logiciel adapté. On utilisera un logiciel de satisfaisabilité appelé `minisat`. Ce logiciel prend en entrée une formule en forme normale conjonctive et trouve une affectation qui la satisfait (s'il y en a une). Pour cela il suffit d'appeler `minisat formule reponse`, où `formule` doit être le nom d'un fichier qui contient la représentation d'une formule. Le programme écrira dans le fichier `reponse` une affectation qui satisfait la formule, s'il y en a.

Le format DIMACS Le format des fichiers d'entrée de `minisat` est un standard international pour la représentation de formules en forme normale conjonctive. Un fichier en format DIMACS commence par une ligne qui spécifie qu'il s'agit d'une forme normale conjonctive, qui dit combien de variables la formule contient, et de combien de clauses disjonctives elle est constituée. Par exemple :

```
p cnf 5 12
```

cela indique que le fichier contient une formule en forme normale conjonctive, avec 5 variables et 12 clauses. Ensuite, le fichier est composé de plusieurs lignes, une par clause. Chaque ligne contient des entiers positifs et/ou négatifs, et se termine par 0. Un entier positif i indique que la i-ème variable apparaît avec polarité positive dans la clause. Un entier négatif $-i$ indique que la i-ème variable apparaît avec polarité négative dans la clause. Par exemple :

```
2 4 -1 -3 0
```

cela représente la clause $x_2 \vee x_4 \vee \neg x_1 \vee \neg x_3$.

Exécution

Pour lancer `minisat`à partir d'un programme Java on utilisera la méthode `exec` de la classe `Execution` disponible sur le site web du cours. La méthode `exec` prends en entrée un tableau de `String`. Le premier

élément est le nom du programme à exécuter (dans notre cas `"minisat"`). Les éléments suivants sont les arguments (dans notre cas les noms des deux fichiers).

La lecture des fichiers pourra se faire à l'aide de la classe `Lecteur` disponible sur le site web du cours. Pour lire un fichier, il faut créer un objet de la classe `Lecteur`, qu'on peut voir comme un magnétophone contenant le fichier. A un instant donné, la tête de lecture se trouve au début d'une ligne du fichier (lors de la création de l'objet, elle se trouve au tout début du fichier). Un appel à la méthode `readLigne()` renvoie la ligne courante sous forme de chaîne de caractères, et déplace la tête de lecture sur la ligne suivante. La méthode `hasLigne()` permet de savoir si l'on est arrivé à la fin du fichier. Il n'y a pas de rembobinage.

L'écriture des fichiers pourra se faire à l'aide de la classe `Ecrivain` disponible sur le site web du cours. Pour écrire sur un fichier, il faut créer un objet de la classe `Ecrivain`, qu'on peut aussi voir comme un magnétophone. Le fichier est au début vide (si le fichier existait déjà il est effacé). Pour des raisons d'efficacité, le magnétophone garde le texte à écrire dans un tampon. A un instant donné, la tête d'écriture se trouve après la dernière ligne du tampon. Un appel à la méthode `writeln(String s)` écrit sur le tampon la chaîne `s` et passe à la ligne suivante. Pour écrire le contenu du tampon dans le fichier, on appelle la méthode `flush()`.

Les plus avertis pourront également utiliser les classes `FileWriter`, `Scanner`, `StringTokenizer`, `Process`, etc.

Organisation

Le projet devra être réalisé par groupe de 1 ou 2 personnes (**limites strictes**).

Le(s) programme(s) devront **impérativement** être écrits en langage `Java` version 5 ou plus, et être exécutables sur au moins une machine du SCRIPT.

Ce programme devra, au minimum :
- permettre à l'utilisateur d'entrer un nom de fichier,
- afficher si le fichier est bien formé,
- afficher, s'il en existe, la ou les solutions correspondantes à la grille décrite dans le fichier.

Un rapport devra être envoyé par email à votre chargé de TD au plus tard une semaine avant la date des soutenances. Ce rapport devra expliciter clairement l'ensemble des points les plus importants du projet. Il devra aussi contenir le code source du projet. Le contenu de ce rapport sera jugé et noté (autant être précis et concis).

Les soutenances auront lieu dans la semaine du 9 janvier 2012. Les dates exactes seront précisées ultérieurement ainsi que l'ouverture des inscriptions aux soutenances.

La soutenance est **obligatoire** (sauf pour les dispenses officielles), toute absence conduira le jury à délivrer la note 0. Chaque personne **devra** intervenir (les silences seront jugés très négativement). Pendant la soutenance, il sera nécessaire de donner une démonstration du projet à l'aide d'un ordinateur qui exécute le code. Cela pourra se faire sur un ordinateur du SCRIPT ou sur un ordinateur portable personnel.