

Projet de Programmation

Objectif

Il s'agit d'implémenter un programme qui va couvrir une grille rectangulaire dont certaines cases sont barrées, avec des dominos. Voici par exemple une grille de dimension 4 (largeur) sur 4 (hauteur). Dans une telle grille, les cases sont dénotées par leur numéro de colonne et leur numéro de ligne, la case (1,1) étant la case en bas à gauche. Sur l'exemple, les cases (1,1), (2,3), (3,2) et (4,4) sont barrées.

			×
	×		
		×	
×			

Les dominos couvrent toujours deux cases de la grille. On a un nombre illimité de dominos à sa disposition. Une case barrée ne peut pas être couverte par un domino, et toutes les cases non barrées doivent être couvertes par un domino. Il n'est pas autorisé que deux dominos se superposent, ni qu'un domino déborde sur les limites de la grille. Voici une solution possible sur l'exemple :

■	■	■	×
■	×	■	■
■	■	×	■
×	■	■	■

La fonctionnalité demandée

Votre programme permettra au moins de trouver une solution pour une grille donnée (ou de signaler qu'il n'y en a aucune), pour des dimensions et des cases barrées quelconques. Votre programme prendra en option -w le nombre de colonnes, et en option -h le nombre de lignes. L'option -e i,j indique que la case (i,j) est barrée, cette option peut être donnée plusieurs fois. Si votre programme s'appelle *monprojet* vous devez le lancer comme suit pour l'exécuter sur l'exemple :

```
monprojet -w 4 -h 4 -e 1,1 -e 2,3 -e 3,2 -e 4,4
```

Votre programme doit afficher la grille avec les cases barrées et les dominos placés selon la solution trouvée par votre programme.

Dans un deuxième temps, ajouter la possibilité pour l'utilisateur de taper sur la touche Espace, pour afficher une solution différente (s'il y en a une).

Indications pour le codage en logique propositionnelle

Un domino peut être placé sur la grille soit couché, soit debout. Si le domino est couché il couvre deux cases (i,j) et $(i+1,j)$, s'il est debout il couvre deux cases (i,j) et $(i,j+1)$. Dans les deux cas on appellera la case (i,j) l'*ancree* de ce domino. Sur la solution indiquée au-dessus, les positions suivantes sont des ancrées :

(1, 2), (1, 3), (2, 1), (2, 4), (3, 3), (4, 1)

Nous préconisons le codage suivant en logique propositionnelle. On choisit pour chaque case (i,j) de la grille deux variables propositionnelles :

- $A[i, j]$ dit si la case (i,j) est l'ancree d'un domino ;
- $O[i, j]$ dit, dans le cas où il y a un domino avec ancre (i,j) , l'orientation de ce domino (0=couché, 1=debout).

La solution indiquée dans l'introduction est donc représentée par l'affectation avec le support suivant :

$A[1, 2], A[1, 3], A[2, 1], A[2, 4], A[3, 3], A[4, 1]$
 $O[1, 3], O[4, 1]$

À vous maintenant de trouver toutes les formules propositionnelles nécessaires pour la représentation du problème.

Indications pour le codage en Java

Votre programme va construire une formule propositionnelle dans le format DIMACS présenté en cours, et lancer le SAT-solveur `minisat` pour chercher une solution.

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` prend 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 dans 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 (**limite stricte**).

Le(s) programme(s) devront être écrits en langage **Java** version 5 ou plus, et être exécutables sur au moins une machine du SCRIPT. Des langages de programmation autre que Java sont seulement admis sur demande individuelle et après autorisation par le responsable du cours.

Un rapport devra être envoyé par email à votre chargé de TP avant la soutenance. 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 en janvier 2013. La date limite pour l'envoi du rapport sera également dans le mois de janvier 2013, et avant la soutenance. 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.