51IF2IK3 – Examen

7 novembre 2008 – durée 2h – aucun document autorisé

On souhaite réaliser un programme pour résoudre des *carrés magiques*.
Un carré magique de taille n est composé d'une grille de taille $n \times n$ dans laquelle on place les n^2 premiers nombres entiers non nuls $(1, 2, 3, \ldots, n^2-1, n^2)$ de manière à ce que les sommes des nombres de chaque ligne et de chaque colonne de la grille soient égales. La figure 1 contient deux exemples de carré magique, un de taille 3 (chaque ligne et chaque colonne a une somme égale à 15) et un de taille 4 (chaque ligne et chaque colonne a une somme égale à 34).

4	9	2
3	5	7
8	1	6

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

FIG. 1 – Exemple de carrés magiques de taille 3 et 4.

Le **problème du carré magique** consiste, **étant donnée une grille partiellement remplie**, à trouver **tous** les carrés magiques qui contiennent les valeurs initiales de la grille. On va résoudre ce problème avec un algorithme de backtracking itératif.

On utilisera les classes `CarreM`, `Case` et `Choix` dont les documentations sont données en annexe.

1) Montrer que pour tout carré magique de taille n, la somme de chaque ligne et de chaque colonne est toujours égale à $\dfrac{n(n^2+1)}{2}$.
(On pourra utiliser sans la démontrer la formule donnant la somme des k premiers nombres entiers.)

Ecrire le constructeur de la classe `CarreM` qui prend en paramètre un entier, construit une grille vide de taille ce paramètre et initialise le champ `poids` en conséquence. On représentera une case vide par la valeur 0.

2) A partir de la grille ci-dessous, expliquer comment on peut obtenir directement la valeur de la case en haut à droite, puis les valeurs des deux autres cases vides.

.	9	.
8	1	6
.	5	7

En déduire un algorithme de prétraitement pour l'algorithme de backtracking.

3) L'algorithme de backtracking devra tester si une valeur `v` est possible ou non pour une case `c` :
Ecrire une méthode `estCompatible(Case c,int v)` de la classe `CarreM` qui renvoie `true` si et seulement si le choix de `v` pour `c` n'est pas incohérent avec les valeurs déjà existantes dans `grille` et avec le poids.

4) Décrire comment un algorithme de backtracking (itératif) pourrait résoudre le problème des carrés magiques à partir d'une grille partiellement remplie. **On ne demande pas une méthode Java mais un algorithme.**
Expliquer comment procède l'algorithme sur la grille suivante :

.	9	.
.	.	7
.	.	6

5) Implémenter l'algorithme de backtracking sous la forme d'une méthode `resoudre` de la classe `CarreM`.
On utilisera une pile Java (`java.util.Stack` dont une documentation partielle est donnée en annexe) pour stocker des objets de la classe `Choix`. On pourra bien sûr ajouter des méthodes à la classe `CarreM` comme dans les exemples vus en cours (même si ces méthodes n'apparaissent pas dans la documentation fournie en annexe de la classe `CarreM`).

6) Pour quelle(s) méthode(s), l'utilisation des exceptions peut être utile ? Expliquer votre choix.
Choisir une de ces méthodes et expliciter les changements à faire dans le code de sa définition et de ses appels.