SUDOKU

1 Partie TD

Question 1 • Modélisez sur papier le problème `sudoku(`$n \times n$`)` sous la forme d'un réseau de contraintes $N = \langle X, D, C \rangle$.

Question 2 • Donnez la taille de l'espace de recherche.

Question 3 • Déroulez l'algorithme du `backtracking` à partir de l'instanciation partielle présentée dans la figure 1.

4		2	3
	2		4
	3		
1			2

FIGURE 1 – Instanciation partielle du *sudoku*(4×4)

Question 4 • Appliquez la propagation des contraintes sur l'instanciation de la figure 1 et donnez la trace des valeurs supprimées.

Question 5 • Proposez sur papier une solution `Generate&Test` pour le problème `sudoku(`$n \times n$`)`.

Question 6 • Modifiez sur papier la version de l'algorithme `backtracking` (vu en cours) et l'algorithme `Generate&Test` de sorte pourvoir retourner l'ensemble des solutions du le problème `sudoku(`$n \times n$`)`.

2 Partie TP

Vous trouverez dans votre dépôt local :

— `"etud.fr.euniv_montpellier.iut.sudoku.generatetest.Sudoku.java"` : une solution `Generate&Test` à implémenter.

— `"etud.fr.univ_montpellier.iut.sudoku.backtrack.Sudoku.java"` : une solution `backtracking` à implémenter.

— `"etud.fr.univ_montpellier.iut.sudoku.ppc.Sudoku.java"` : une modélisation PPC du problème `sudoku`$(n \times n)$.

Question 7 • Codez le `Generate&Test`.

Question 8 • Codez le `backtracking`.

Question 9 • Comparez avec des tests unitaires le modèle PPC et les deux solutions `Generate&Test` et `backtracking`, quelles sont vos observations/conclusions ?

Question 10 • Révisez les trois solutions pour retourner l'ensemble des solutions de `sudoku`$(n \times n)$.

Question 11 • Comparez de nouveau avec des test unitaires les versions qui permettent de retourner la totalité des solutions.

3 Révisions des modèles en PPC

Nous allons passer maintenant tester la déclarativité de la PPC avec des révisions du modèle fourni. La figure 2 est l'une des plus difficiles instances du sudoku(9×9).

8								
		3	6					
	7			9		2		
	5				7			
				4	5	7		
			1				3	
		1					6	8
		8	5				1	
	9					4		

FIGURE 2 – Instance difficile du *sudoku*(9×9)

Question 12 • Révisez le modèle PPC qui est dans `Sudoku.java` pour résoudre l'instance de la figure 2.

Question 13 • Révisez votre code de sorte à pouvoir prendre en compte l'instance de la figure 2 aussi bien que celle de la figure 3.

	G			F	8	9	6	4	B	D	5			3	
6	C					4	E	2	7					5	9
			D			G	7	F	E			6			
		4	3	A							6	1	B		
7			5	8	F					B	E	9			G
8				9			4	D			3				2
C	1	3				6			G				F	4	5
9	D	B			G					F			7	A	6
G	B	A			2					7			5	6	D
5	6	F				A			2				8	7	4
D				6			9	5			G				F
3			C	B	5					A	4	G			1
		9	6	G							7	2	C		
			G			B	D	C	5			F			
4	3					8	2	G	F					1	7
	8			5	9	E	A	1	3	2	D			G	

FIGURE 3 – Instance du *sudoku*(16×16)

4 Greater Than Sudoku

Une des variantes du sudoku est le `Greater Than Sudoku` où une des instances est présentée dans la figure 4. En plus des contraintes du Sudoku classique, le GTSudoku ajoute dans la grille des symboles de comparaison ($\ll > \gg$ et $\ll < \gg$) indiquant une contrainte d'inégalité entre les nombres de cases adjacentes d'un même carré.

Question 14 • Révisez le modèle dans `Sudoku.java` de sorte à résoudre l'instance de la figure 4.

Question 15 • Déposez un jar executable sur la plateforme de test *CP – Dashboard* et comparez les résultats avec celles de vos collègues en temps réel. Le jar doit avoir uniquement l'affichage de `printStatistics()` sur la console.

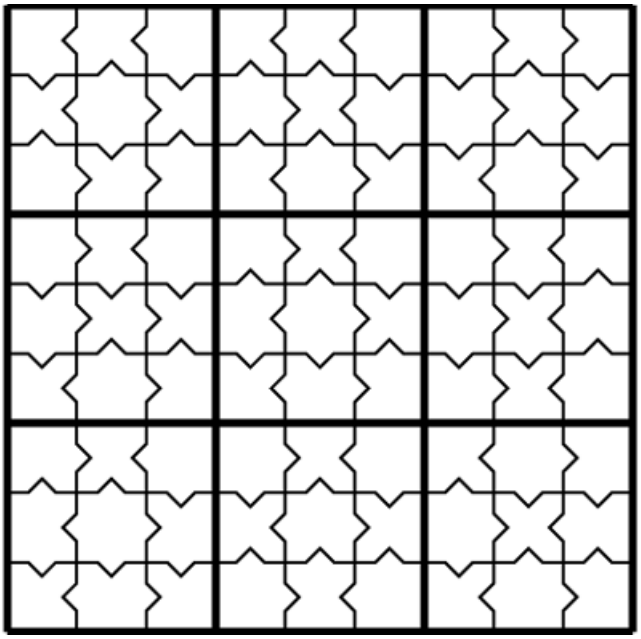

FIGURE 4 – Instance du $GT - sudoku(9 \times 9)$