Année 2019–2020

Projet informatique (PI4) – L2

Sudoku

François Laroussinie
francoisl@irif.fr

Résumé : L'objectif de ce projet est de réaliser un programme permettant de résoudre des sudoku en justifiant chaque coup.

1 introduction

Un Sudoku est un casse-tête composé d'une grille de 9 cases sur 9 où certaines cases contiennent un chiffre de $\{1, 2, \ldots, 9\}$. L'objectif est de compléter toutes les cases avec ces chiffres de manière à ce que chaque ligne, chaque colonne et chacun des 9 carrés internes (voir ci-dessous un exemple d'une grille et de sa solution) contiennent exactement l'ensemble $\{1, 2, \ldots, 9\}$. Un Sudoku doit avoir une solution unique.

On peut généraliser cette définition en disant qu'un Sudoku $n \times k$ est une grille $n \cdot k \times n \cdot k$ à remplir par les nombres $\{1, 2, 3, \ldots, n \cdot k\}$ (dans ce cas, il y a $n \cdot k$ rectangles internes de taille $n \times k$). Avec cette définition, un Sudoku « classique » est donc un Sudoku 3×3. On donne ci-dessous un exemple de Sudoku 2×3. C'est cette définition générale que l'on considérera dans la suite (NB : on suppose toujours qu'il existe une unique solution).

Exemples. On propose ici deux exemples : un Sudoku classique 3×3 et un Sudoku 2×3, avec leur solution.

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

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

1		3	4	5	6
5	6		2		4
3	4		6	1	2
	3			6	
4	5				
	1				5

1	2	3	4	5	6
5	6	1	2	3	4
3	4	5	6	1	2
2	3	4	5	6	1
4	5	6	1	2	3
6	1	2	3	4	5

On sait facilement concevoir des programmes pour résoudre les Sudoku : le plus simple est d'utiliser un algorithme de *backtracking* : étant donnée une case vide, on énumère la liste des valeurs possibles pour cette case (en fonction des contraintes existantes), puis on choisit une de ces valeurs et on recommence avec une autre case vide... lorsqu'il n'existe pas de valeur possible pour une case vide, on revient sur le choix précédent *etc.* Cette algorithme très simple marche bien sur les Sudoku de taille raisonnable.

Mais un humain ne procède pas ainsi ! En général, on va éviter de deviner des nombres, et plutôt chercher à *déduire* de nouvelles valeurs en fonction des valeurs présentes dans les cases, et ainsi on est sûr du choix fait : on peut le justifier.

Dans ce projet, il faudra implémenter ces deux algorithmes : celui par *backtracking*, et celui par déduction.

2 Travail à réaliser

On plante le décor. Dans un premier temps, il faudra définir les structures de données pour manipuler les Sudoku (les classes) et les méthodes associées. Sur la base du format de représentation d'une grille de Sudoku donné en annexe, on écrira des opérations de lecture/écriture d'une grille depuis/dans un fichier.

Base d'exemples. On construira un ensemble de grilles que l'on utilisera ensuite pour tester les algorithlmes (par exemple par génération aléatoire).

Algorithmes. On commencera par programmer l'algorithme de *backtracking*. Ensuite on s'intéressera à l'algorithme par déduction : celui-ci sera construit à partir de règles de déduction à trouver (et à compléter en fonction des résultats de votre algorithme sur les exemples). Et on comparera les algorithmes.

A Format de représentation

Dans un fichier, on représentera une grille de Sudoku par deux entiers n et k représentant sa taille, suivis par $(n \cdot k)^2$ valeurs (on utilisera le chiffre 0 pour représenter une case vide) séparées par des virgules de manière à ce que les $n \cdot k$ premières valeurs constituent la première ligne, les $n \cdot k$ suivantes la seconde ligne, *etc.* Par exemple, la première grille donnée sera codée par :

```
3 3
0, 0, 0, 0, 0, 9, 3, 0, 0, 1, 4, 5, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 8, 4,
7, 0, 0, 0, 0, 0, 0, 4, 0, 0, 2, 0, 0, 5, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 8, 0,
2, 0, 0, 5, 7, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 5, 0, 6, 0, 0, 6, 0, 4, 0,
9, 0, 0
```

Dans les fichiers représentant une grille, on pourra ajouter des lignes de commentaires en les préfixant par `%`.