TP Option informatique $n°3$ – Sudoku par Backtracking
Le 11/01/2024

Merci aux collègues du lycée Chateaubriand pour la mise en ligne d'un sujet ayant très fortement inspiré celui-ci.

On souhaite résoudre les sudokus par backtracking. Commençons par rappeler rapidement le principe du jeu. On considère une grille carré ayant n^2 cases de cotés (donc n^4 cases en tout). Le but est de remplir chaque case de la grille avec un nombre allant de 1 à n de telle sorte que :

- (L) : Un même nombre apparait une fois et une seule dans chaque ligne.
- (C) : Un même nombre apparait une fois et une seule dans chaque colonne.
- (B) : Un même nombre apparait une fois et une seule dans chacun des n^2 blocs.

On dispose généralement d'une grille partiellement remplie et on veut compléter les cases qui sont vides en respectant les règles (L), (C) et (B).

Un exemple avec $n = 3$.

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

On va repérer les cases de la grille de deux manières :

- Par un numéro allant de 0 à $n^4 - 1$. La case 0 est la case en haut à gauche, la case 1 celle qui est juste à droite et ainsi de suite, la case $n^4 - 1$ est la case en bas à droite.
- Par un couple (i, j) où i est le numéro de la ligne et j celui de la colonne. Les deux vont de 0 à $n^2 - 1$.

1. Écrire une fonction `numero : int → int*int → int` tel que `numero n (i,j)` retourne le numéro de la case située en ligne i, colonne j.

2. Écrire une fonction `coordonnes : int → int → int*int` tel que `coordonnes n a` retourne le couple (i,j) qui sont les coordonnées de la case numérotée `a`.

3. Écrire une fonction `meme_bloc : int → int*int → int*int → bool` tel que `meme_bloc n (i,j) (k,l)` retourne `true` si et seulement si les cases de coordonnées `(i,j)` et `(k,l)` sont dans le même bloc.

On souhaite maintenant tester si les contraintes (L), (C) et (B) sont vérifiées.

4. Écrire une fonction `condition : int → int → int → bool` telle que `condition n a b` retourne `true` si et seulement si l'une des contraintes (L), (C) ou (B) impose que le nombre de la case `a` soit différent du nombre de la case `b` (on doit donc retourner `false` pour `a=b`, mais `true` dans tout autre cas où les deux nombres sont sur une même ligne, une même colonne ou un même bloc).

On veut maintenant remplir la grille de sudoku. Une grille sera implémentée par le type suivant :

```
type grille = {taille : int; tab : int option array array};;
```

Le champ `taille` indique la taille n de la grille ($n = 3$ sur l'exemple), le champ `tab` est une matrice d'entiers avec n^2 lignes et n^2 colonnes, à ceci près que si une case n'a pas encore de nombre affecté cela sera codé par le fait que la case de la grille contient la valeur `None`.

Le but des questions suivantes est de compléter une grille initiale en une grille complète.

5. Écrire une fonction `verification : grille → int → int → bool` telle que pour toute grille, `verification grille a v` renvoie un booléen selon que l'on peut affecter à la case dont le numéro est `a` la valeur `Some v` sans enfreindre les contraintes (L), (C), (B).

6. Écrire une fonction `case_vide : grille → int` telle que `case_vide grille` retourne le numéro d'une case vide de la grille. On pourra, par exemple, retourner la case vide de plus petit numéro. Si la grille est entièrement remplie, la fonction devra provoquer l'exception `Not_found`.

Le principe de la fonction de backtracking est le suivant :

- La fonction trouve une case qui est encore vide ; s'il n'y en a plus, elle provoque l'exception `Not_found`.
- La fonction essaye de mettre toutes les valeurs possibles dans cette case (en utilisant la fonction `verification`) ; une fois qu'elle a mis la nouvelle valeur, elle se rappelle récursivement pour aller chercher la prochaine case vide.
- Quand on arrive à une impossibilité, elle revient en arrière en remettant la case concernée à `None`.

7. Écrire une fonction `sudoku : grille → unit` qui permet de résoudre un sudoku par backtracking. La fonction prend en argument une grille partiellement remplie, ne retourne rien, mais a pour effet de bord de modifier son argument en solution du Sudoku. S'il n'en existe pas, l'exception `Not_found` n'est pas rattrapée.

8. Tester la fonction en commençant par une grille vide. Tester ensuite avec la grille proposée en introduction. On pourra commencer par écrire une fonction qui remplit partiellement une grille.