Sudoku (Tableaux et récurrences)

Christophe Fiorio

L'objectif ici est de résoudre une grille de sudoku, par exemple celle-ci

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

Résoudre une grille de sudoku consiste à remplacer les 0 par un chiffre de 1 à 9 de telle sorte qu'il soit unique dans sa *ligne*, sa *colonne* et sa *région*. On appelle *région* les 9 sous-grilles de 3×3 encadrées.

Le principe de résolution est simple, et peut se résumer ainsi :

1. prendre la prochaine case à 0
2. si au moins une valeur est possible, on essaie une valeur et on continue de résoudre avec la prochaine case à 0
3. si on remplit toutes les cases à 0, le problème est résolu
4. sinon si on arrive à un blocage, on reprend au point 2. avec la prochaine valeur possible en ayant au préalable effacé toutes les valeurs inscrites depuis
5. si il n'y a plus de valeurs possibles, alors c'est un échec et il n'y a pas de solution

C'est exactement ainsi que l'on procède « à la main » en essayant d'être malin dans le choix des cases afin d'aller au plus vite à la solution.

La difficulté pour écrire un algorithme reprenant cette méthode est le point 4. Comment revenir en arrière simplement, sans avoir à mémoriser toutes les opérations pour pouvoir les annuler ?

Par la suite on supposera que chaque ligne est un tableau de 9 entiers, et donc que la grille est un tableau de lignes, c'est à dire un tableau de 9 tableaux de 9 entiers.

* Exercice 1 valeurs disponibles sur une ligne

Écrire une fonction qui pour une ligne donne la liste des valeurs disponibles pour cette ligne

* Exercice 2 valeurs disponibles sur une colonne

Écrire une fonction qui pour une colonne donne la liste des valeurs disponibles pour cette colonne

* Exercice 3 valeurs disponibles dans une région

Écrire une fonction qui pour une région donne la liste des valeurs disponibles pour cette région

** Exercice 4 valeurs possibles

En vous servant des 3 fonctions précédentes, écrire une fonction qui, pour une position donnée de la grille, donne la liste des valeurs possibles à cette position

* Exercice 5 positions à remplir

Écrire une fonction `listePositionsNonRemplies` qui renvoie la liste des positions de la grille pour lesquels il faut trouver une valeur.

*** Exercice 6 remplir grille

Soit la fonction suivante :

`grilleFinie` : $[[\text{int}]] \times [\text{int}] \times \text{int} \longrightarrow [[\text{int}]] \times \text{bool}$

qui à partir d'une grille, d'une liste des positions à remplir, de l'index à partir duquel les positions à remplir doivent être examinées [1], renvoie la grille complétée et `true` si elle a été entièrement remplie.

L'algorithme de résolution pourra s'écrire comme la fonction :

```
func sudoku(grille : [[int]])→ ([[int]] , bool)
    var lpos: [(int, int)] = listePositionsNonRemplies(grille)
    if len(listePositionsNonRemplies)> 0 then
        return grilleFinie(grille, lpos, 0)
    else
        return (grille, false)
    endif
endfunc
```

Algorithme 6 : Résolution de sudoku

Écrivez l'algorithme récursif de la fonction `grilleFinie`.

1. Les positions précédant l'index sont considérées comme n'appartenant pas à la liste : si on avait une vraie liste, on les aurait supprimées de la liste, ce qui n'est pas possible avec un tableau.