Séance 8: SUDOKU

Université Paris-Diderot

Le *sudoku* est un jeu en forme de grille défini en 1979 par l'Américain Howard Garns. Le but du jeu est de remplir la grille avec une série de chiffres tous différents, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou dans une même région. Les chiffres vont de 1 à 9, les régions étant alors des carrés de 3 × 3. Quelques chiffres sont déjà disposés dans la grille, ce qui autorise une résolution progressive du problème complet. Voici un exemple d'une telle grille et de sa solution :

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

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

On souhaite pouvoir vérifier automatiquement si une grille de sudoku donnée est une solution correcte. Pour cela, nous allons représenter la grille comme un tableau de tableaux d'entiers `S` de taille 9 × 9, de sorte que `S[i][j]` contient le chiffre inscrit dans la case située à l'intersection de la `i`ème ligne et de la `j`ème colonne. Les cases vides seront représentées par le chiffre 0.

Remarque:
Vous rédigerez les réponses aux exercices dans le fichier fourni `Exos.java`. Il est important de **tester** au fur et à mesure votre code, à l'aide des tests fournis dans le fichier `Tests.java`. Pour les lancer, vous devez donc compiler et exécuter le fichier `Tests.java`.

Si vous bloquez sur une fonction d'un exercice, vous pourrez quand même l'utiliser dans les exercices suivants en utilisant l'implémentation compilée dans le fichier `Correction.class`. Pour ce faire, il suffit lorsque vous appelez la fonction de précéder son nom par `Correction.`, par exemple vous écrirez `Correction.fonc(42)` au lieu de `fonc(42)` pour une fonction qui s'appelle `fonc` et qui prend en argument un `int`.

Exercice 1 (Utilitaires, ⋆⋆)

1. *Écrivez une fonction* `initBoolArray` *qui prend en entrée un entier* `n` *et un booléen* `b`*, et retourne un tableau de booléens de taille* `n` *où chaque case est initialisée avec* `b`.
2. *Écrivez une fonction* `equalsBoolArray` *qui prend en entrée deux tableaux de booléens, et renvoie* `true` *s'ils sont égaux (ils contiennent les mêmes valeurs dans le même ordre),* `false` *sinon.*
3. *Écrivez une fonction* `transpose` *qui prend en entrée une grille de sudoku et renvoie sa* transposée*, c'est-à-dire la même grille où les lignes et les colonnes ont été inversées. Par exemple,*

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

devient

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

□

Exercice 2 (Tous les chiffres, ⋆⋆)

Remarque:
Dans les questions qui suivent, essayez au maximum de vous servir des fonctions utilitaires définies dans l'exercice 1.

1. *Écrivez une fonction* `chiffresLigne` *qui prend en entrée une grille* `S` *et un entier* `i`*, et renvoie un tableau* `t` *de booléens tel que* `t[c]` *est égale à* `true` *si la* `i`*ème ligne de* `S` *contient le chiffre* `c`*, et* `false` *sinon. Par exemple, si* `S` *est la grille donnée au début du sujet, on aura*

```
chiffresLigne(S, 3) == {false, true, false, false, true, false,
    true, true, false, false}
```

2. *Écrivez une fonction* `chiffresColonne` *qui fait l'équivalent de* `chiffresLigne` *pour les colonnes. Par exemple, si* `S` *est la grille donnée au début du sujet, on aura*

```
chiffresColonne(S, 6) == {false, false, false, false, true, true,
    false, false, true, true}
```

3. *Écrivez une fonction* `chiffresCarre` *qui prend en entrée une grille* `S` *et deux entiers* `i`,`j`*, et fait l'équivalent des fonctions précédentes pour le carré de* 3×3 *qui contient la case en ligne* `i` *et en colonne* `j`*. Par exemple, si* `S` *est la grille donnée au début du sujet,* `i = 6` *et* `j = 7`*, on aura*

```
chiffresCarre(S, 6, 7) == {false, true, false, false, true, true,
    false, false, true, true}
```

□

Exercice 3 (Solution, ⋆⋆)

1. *À l'aide des fonctions des exercices 1 et 2, écrivez une fonction* `isSolution` *qui prend en entrée une grille de sudoku, et renvoie* `true` *si celle-ci est une solution correcte,* `false` *sinon. Par exemple, la grille donnée au début du sujet doit renvoyer* `false`*, et sa solution doit renvoyer* `true`*.*

□