

# TP1 : Programmation Par Contraintes

L'objectif de cette première étude est multiple :

- Réfléchir à une solution pour résoudre un problème combinatoire : `sudoku( $n \times n$ )` .
- Initiation à la PPC avec la bibliothèque et le solveur `choco`.
- Comparaison des deux approches.
- Test de la déclarativité des deux approches.

Vous trouverez dans votre dépôt local :

- `fr.univ_montpellier.fsd.sudoku.ppc.Sudoku.java` : une modélisation PPC du problème `sudoku( $n \times n$ )` .
- `fr.univ_montpellier.fsd.sudoku.ppc.HardSudoku.java` : une instance sudoku à modéliser.
- `fr.univ_montpellier.fsd.sudoku.ppc.GTSudoku.java` : une variante du problème à modéliser.
- `fr.univ_montpellier.fsd.sudoku.imp.Sudoku.java` : une solution Java à implémenter du problème.
- `fr.univ_montpellier.fsd.sudoku.App.java` : module pour comparer les solutions.

**Question 1** • Codez la fonction `solutionChecker()` de la classe `imp.Sudoku.java`.

**Question 2** • Codez une première version random de la fonction `generateSolution()` qui se trouve dans de la classe `imp.Sudoku.java`.

**Question 3** • Proposez un algorithme et son implémentation dans `findSolution()` de la classe `imp.Sudoku.java`.

**Question 4** • Comparez le modèle PPC avec votre solution dans `App.java`, quelles sont vos conclusions ?

**Question 5** • Proposez une solution Java pour résoudre le problème et comparez le avec le modèle PPC. Quelles sont vos conclusions ?

**Question 6** • Réviser les deux solutions (Java / PPC) pour retourner l'ensemble des solutions.

Nous allons passer maintenant à la modélisation PPC avec l'idée de voir comment on peut réviser un modèle sans difficulté en PPC. Voici une des plus difficiles instances du sudoku :

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

**Question 7** • Révissez le modèle PPC qui est dans `HardSudoku.java` pour résoudre l'instance donnée avant.

**Question 8** • Ci-dessous une instance  $16 \times 16$ . Proposez une façon simple pour pouvoir passer du modèle `HardSudoku.java` à celui de l'instance  $16 \times 16$  (`VeryHardSudoku.java`) en modifiant quelques lignes de code.

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

**Question 9** • Retournez l'ensemble des solutions de l'instance difficile.  
Une des variantes du sudoku est le **Greater Than Sudoku**.

**Question 10** • Révissez le modèle dans `GTSudoku.java` de sorte à résoudre l'instance suivante :

<	>		<	<		>	<
^	^	^	v	v	^	v	v
<			<	<	^	<	<
v	v	v	^	^	^	^	v
>	<		>	>		>	<
>	>		<	>		<	<
v	^	^	^	v	v	^	^
<	>		v	^	^	v	^
^	v	^				<	>
<	<		<	>			
>	<		>	>		<	<
^	^	^	^	v	^	v	v
>	>		v	v	>	>	>
v	v	v	v	v	^	^	^
>	<		>	<		>	>