TP Backtracking 2

Exercice 3 : Sudokus

Lors de la résolution, penser à ne pas modifier les cases dévoilées au départ !
On peut:

- pour une case de numéro r (compris entre 0 et 80), exprimer la ligne par r//9 et la colonne par r%9
- pour une case donnée [lig][col], exprimer le numéro de sa région: **3*(lig//3)+(col//3)**
- pour une case donnée [lig][col], exprimer le numéro (compris entre 0 et 80) de la case correspondante: 9*lig + col
- pour une région donnée (0,1,2,3,4,5,6,7,8), exprimer les coordonnées de la case située en haut et à gauche de cette région:

ligne=3*(region//3) et col=3*(region %3)

Exemple de grille de sudoku :

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

0, 9, 0, 0, 2, 0, 0, 1, 0
0, 0, 0, 1, 0, 6, 0, 0, 0
0, 4, 0, 0, 9, 0, 0, 8, 0

6, 0, 0, 0, 0, 0, 0, 0, 3
5, 7, 0, 3, 0, 1, 0, 6, 8
0, 3, 8, 0, 0, 0, 9, 5, 0

1. La fonction est_solution est inutile ici , pourquoi ?

2. Pour l'écriture de la fonction ajout_possible : on peut placer dans 3 listes différentes : les éléments de la ligne de la case traitée, ceux de la colonne de la case traitée, et ceux de la région de la case traitée et vérifier que ces 3 listes ont toutes bien des éléments différents en utilisant la fonction python count .

Exercice 2 : Carrés magiques

1. Spécifier et écrire la fonction `est_solution`
2. Générer des carrés aléatoires d'ordre N=3 jusqu'à obtenir un carré magique.
3. Afficher tous les carrés magiques d'ordre N=3 par une recherche exhaustive **itérative**.
4. En adoptant le schéma général de backtracking vu en cours, spécifier et écrire les fonctions `ajout_possible` et `placer` . Afficher ensuite tous les carrés magiques de taille N=3 puis 4 .

   Comment peut-on réduire l'espace de recherche ?

Exercice 3 : Problème des n reines

1. Écrire l'algorithme itératif de résolution du problème des 4 reines.
2. Généraliser à n reines grâce à un algorithme de backtracking :
   V0 : version qui affiche les solutions au fur et à mesure
   V1 : version qui renvoie la liste des solutions
3. Par des considération de symétries, comment peut-on réduire l'espace de recherche ?