Algorithmique:
Backtracking 1

Bonus :

l'exercice 1 de cette série (N-Reines) fait l'objet d'un bonus. Veuillez suivre attentivement les spécifications de l'énoncé. Ceci est un travail **individuel**.

Pour obtenir les points de bonus, vous devrez, pour le mardi 29 octobre 23h59 au plus tard, rendre les fichiers suivants (avec votre nom et votre prénom à la place de 'NomPrenom') :

- Un fichier pdf (*NomPrenom.pdf*) répondant à toutes les questions de l'exercice.
- Un fichier python (*NomPrenom.py*) de votre implémentation de l'algorithme. N'envoyez pas le bytecode (fichier .conpyc).

Les fichiers doivent être soumis sur Moodle dans le devoir 'Bonus 3'.

Le rapport doit être dactylographié au format pdf avec votre nom sur la première page.

Le script doit utiliser un version 3.x de Python et respecter strictement les spécifications pour l'input et l'output pour être pris en compte.

1 N-Reines

Dans la série 2, nous avons traité le problème des n-Reines avec une stratégie de type Greedy. Envisageons à présent le même problème avec une approche de type backtracking.

Dans le problème des N-Reines, nous avons un nombre n de reines à placer sur un échiquier carré de dimensions $n \times n$ sans qu'aucune reine n'en attaque une autre (une reine peut attaquer toute pièce se trouvant sur la même ligne, même colonne, ou mêmes diagonales qu'elle).

Notez bien que cela signifie que n peut prendre une valeur quelconque supérieure à 3, et pas nécessairement égale à 8 comme dans le cas d'un plateau d'échecs.

1.1 Implémentation (4 points)

Vous devez implémentez en Python une méthode de backtracking permettant de trouver toutes les solutions pour le problème des N-Reines. Pour ce faire, écrivez une fonction `solve(n)` qui renvoie la liste des solutions pour un échiquier de dimension n en utilisant un des code générique pour le backtracking vu en cours.

Pour l'implémentation de l'algorithme de backtracking, vous devez implémenter les trois fonctions `B(x,k,n)`, `T(x,k,n)` et `P(x,k,n)` correspondant au problème, comme vu en cours.

Le format de la solution renvoyée par la fonction `solve(n)` est une liste dont chaque élément est une liste de reines satisfaisant la contrainte du problème.

Par 'liste de reines' nous entendons une liste de numéros de colonnes, chaque entrée étant un numéro de ligne. Exemple : pour le problème des 4-reines, il y a 2 solutions, et l'algorithme doit répondre : `[[1, 3, 0, 2], [2, 0, 3, 1]]`. La première solution, `[1, 3, 0, 2]`, signifie que la reine de la ligne 0 est sur la colonne 1, le reine de la ligne 1 sur la colonne 3, la reine de la ligne 2 sur la colonne 0 et la reine de la ligne 3 sur la colonne 2.

Conseils :

- Le fichier *aidenreinesbt.py* disponible sur Moodle illustre une manière de coder les états et propose une fonction pour imprimer l'échiquier dans la console. Cependant, il vous faudra implémenter une manière d'obtenir les enfants d'un état, ainsi qu'une manière de contrôler si un état est solution et, enfin, d'implémenter l'algorithme de backtracking lui-même.
- La vérification des contraintes de lignes et de colonnes est facile ; pour ce qui est des diagonales, rappelez vous que deux éléments sont sur une même diagonale si et seulement si la valeur absolue de leur distance mutuelle horizontale est égale à la valeur absolue de leur distance mutuelle verticale.

1.2 Complexité (2 points)

À priori, quelle est la complexité en temps de l'algorithme backtracking tel qu'il a été présenté en classe, dans le pire des cas ? Argumentez votre réponse.

Et en pratique, quel est la complexité observée ? Justifiez !

1.3 Comparaison avec la méthode Greedy (3 point)

On rappelle que dans la méthode Greedy, les reines sont initialement placées aléatoirement et de façon à ce qu'aucune reine ne soit sur la même ligne ou colonne qu'aucune autre, puis sont déplacées de manière à satisfaire la contrainte du problème, en choisissant à chaque coup le mouvement réduisant le plus le nombre de conflits. Si l'algorithme reste coincé dans un minimum local, on réinitialise les reines aléatoirement en espérant trouver une solution la fois suivante.

Donnez votre intuition quant à la complexité en temps de l'algorithme Greedy. Quels sont les avantages et inconvénients de chaque méthode par rapport à l'autre (Greedy et Backtracking) ?

1.4 Astuce (1 point)

Etant donnée une solution au problème des n-Reines, quel(s) moyen(s) permet(tent) d'en trouver immédiatement d'autres à partir de cette dernière, sans avoir à relancer aucune recherche d'aucune sorte ?