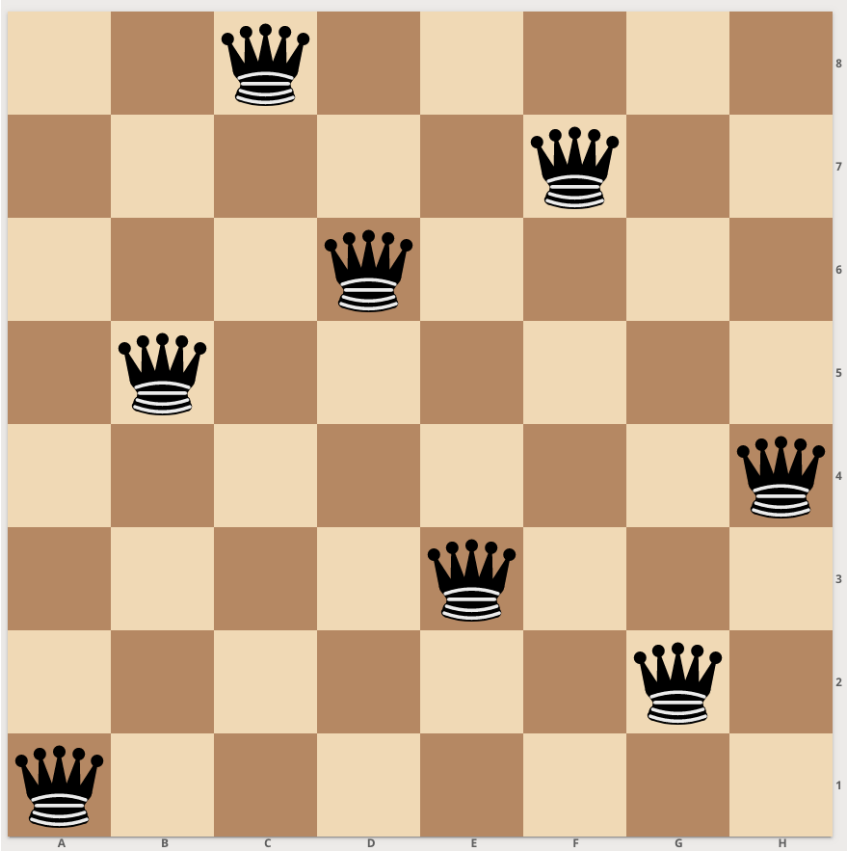
Projet : Les N reines

Contexte

Comme le dit la page wikipedia du problème des huit dames, 8 queens en anglais :

> Le but du **problème des huit dames** est de placer huit dames d'un jeu d'échecs sur un échiquier de 8 × 8 cases sans que les dames ne puissent se menacer mutuellement, conformément aux règles du jeu d'échecs (la couleur des pièces étant ignorée). Par conséquent, deux dames ne devraient jamais partager la même rangée, colonne, ou diagonale..
>
> Simple mais non trivial, ce problème sert souvent d'exemple pour illustrer des techniques de programmation.



But du jeu

Vous devez implémenter trois algorithmes différents ainsi que trois fonctions utilitaires.
Ces deux algorithmes ont les mêmes paramètres d'entrées, et renvoient la même chose (cf code). Il est conseillé de commencer par le premier algorithme.
Les paramètres d'entrées seront toujours un plateau vide (une matrice de taille N) et sa taille N.
Le plateau sera rempli de 0 pour signifier les cases vides. Vous le remplirez de 1 pour indiquer les reines.

Fonctions utilitaires

Il y a trois fonctions utilitaires à implémenter.

- Une fonction indiquant si aucune reine ne peut se frapper
- Une fonction indiquant si la solution a été trouvée ainsi que le nombre de reine.
- Une fonction d'affichage du plateau

Premier algorithme : la naïveté

Le premier algorithme de résolution doit rester simple, il existe plusieurs algorithmes dit "glouton" ou "greedy" en anglais pour résoudre un problème.

Un exemple l'algorithme est de placer une reine sur la première colonne du plateau, puis de boucler sur toute les cases de la seconde colonne pour voir où il est possible de passer la 2ème sans qu'elle ne puisse attaquer la 1ère. Puis répéter le même procédé jusqu'à ce que toutes les reines aient été placées. Si ce n'est pas possible, revenir au placement de la première reine.
Cet algorithme s'appelle "backtracking".

Ce premier algorithme ne sera pas optimal, mais doit renvoyer un plateau contenant les N reines demandées. Il conviendra pour un plateau petit, de 5*5, 8*8 par exemple. Au maximum 20*20.

Deuxième algorithme : le meilleur !

Ce deuxième algorithme nécessitera des recherches de votre part. Il devra fonctionner sur N reines, ou N pourrait être 100.

Lors de la soutenance, je vous demanderais de m'expliquer, voir de me représenter, cette solution. N'hésitez pas à fournir vos sources.

Troisième algorithme : toutes les combinaisons.

Ce troisième algorithme nécessitera aussi des recherches de votre part, il sera exhaustif et donnera toutes les combinaisons possibles, pour un plateau d'une taille donnée, réunissant les conditions.

Règles du jeu

En utilisant le code python fourni.
Vous devez implémenter les différentes fonctions.

Collaboration

- Vous pouvez être un ou deux, mais pas trois
- Vous pouvez vous aider les uns les autres, en particulier dans la recherche du meilleur algorithme. Sans copier évidemment.
- Vous pouvez vous échanger des fonctions/classes utilitaires non obligatoires. Mais si cela est fait, vous devrez le mentionner.

Respect du format

Tout manquement à ces règles sera sanctionné !

- Vous devez respecter le format d'entrée et de sortie
- Vous pouvez (devez) travailler dans d'autres fichiers
- Vous pouvez ajouter des tests, mais n'avez pas le droit de modifier les tests existants
- Vous devez coder en Python 3.6 ou supérieur
- Vous pouvez utiliser toutes les librairies intégrées au Python, à l'exception de pytest qui est autorisé, voir très fortement conseillé.

Liste des librairies intégrées à Python 3.6 :
https://docs.python.org/3.6/library/index.html
Si vous avez besoin d'une autre librairie, contactez moi et on en parlera.

Livraison

Vous devez me partager, au plus tôt, votre code dans un repository privé sur github (https://github.com/sophisur)
Afin que j'ai accès à tout l'historique de votre code. La version utilisée pour la correction sera celle de la branche principale le 6 janvier. Une soutenance, qui servira de pré-correction, le 7, après le DST, selon les conditions sanitaires. Vous permettant d'éviter des petits accidents.

Score final

Votre score final sera basé sur plusieurs éléments. Le barème est à titre indicatif, il vous sert de guide.

Est ce que tous les tests passent ? (14)

Tout simplement, est ce que tout fonctionne ? Plus ou moins bien ?

- Un algorithme : 5 points
- Deux algorithmes : 10 points
- Les trois fonctionnent : 14 points

Qualité du code : Le code respecte t'il les normes Python ? (4)

Ici je vais noter si vous respectez la norme PEP 8. Beaucoup d'IDE permettent de vérifier automatiquement si votre code respecte cette norme.
https://www.python.org/dev/peps/pep-0008/

Mais aussi la lisibilité du code, le nommage et la longueur de vos fonctions/méthodes...

La perfection ! La compétition ! (2)

Ici, je vais comparer la performance de votre algorithme, en temps, et sur la qualité du résultat.

- Le temps d'exécution des algorithmes
- Jusqu'à combien de reine je peux appeler sur votre meilleurs algorithme en gardant un temps raisonnable ?
- Ceux qui auront fait les meilleures fonctions utilitaires supplémentaires

Conseils

Une petite liste non exhaustive :

- Je vous conseille de coder en pair, au moins sur la mise en place du premier algorithme.
- N'attendez pas pour exécuter ! Dés que vous récupérez le code, avant même de commencer à coder. Allez y pas à pas, en exécutant au fur et à mesure. Le TDD est une bonne pratique https://fr.wikipedia.org/wiki/Test_driven_development
- Faites des recherches
- Posez moi des questions
- Mettez en place un environnement qui vous convienne, pour ma part, j'ai codé le projet sur Pycharm, en utilisant la librairie pytest.
- Faites des dessins