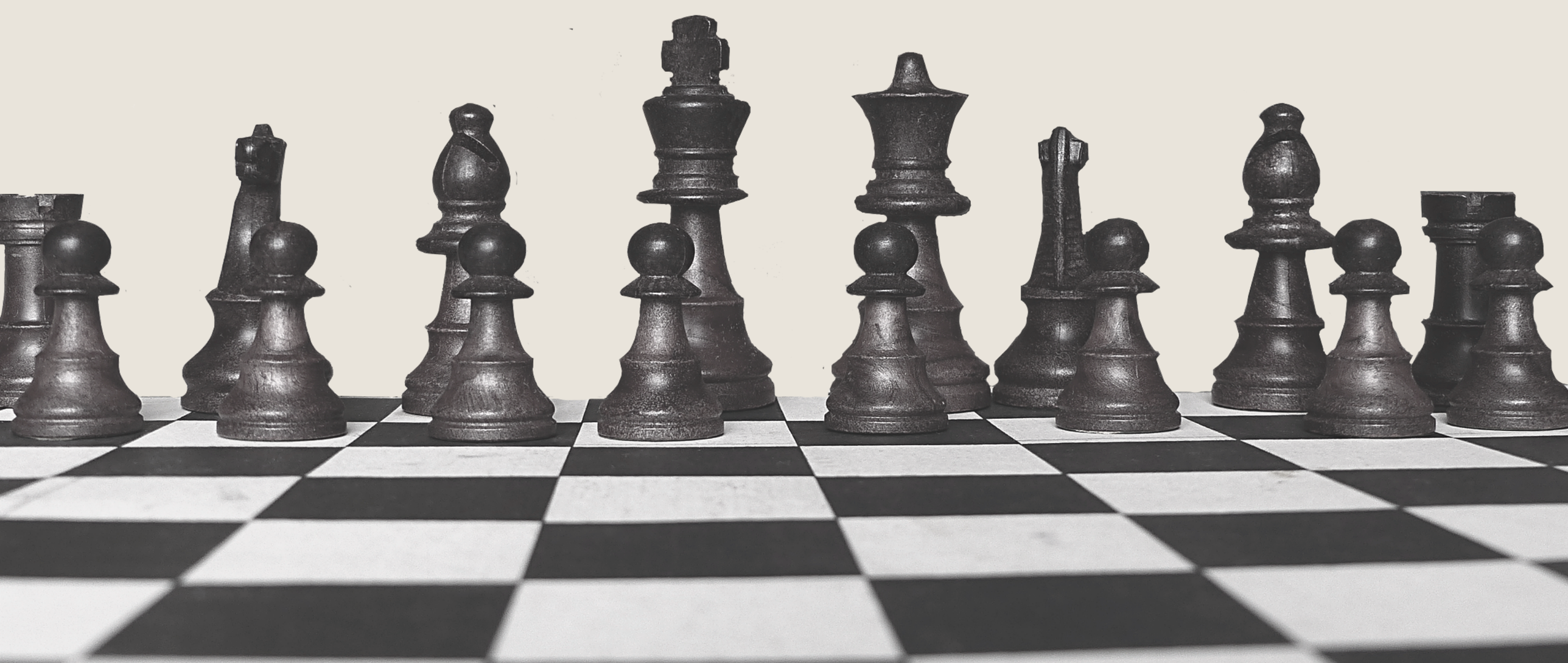
MARTIN Maxime STADNICHENKO Varvara

SAE2.02 : Exploration algorithmique

LE PROBLÈME DES HUIT REINES



SOMMAIRE 

INTRODUCTION

Le problème des huit reines est un défi intellectuel fascinant qui a une longue histoire et qui a été étudié par de nombreux mathématiciens et informaticiens célèbres au fil des ans.

PASSÉ

Le problème est connu depuis l'antiquité (~ l'époque médiévale): vous devez disposer les lettres de manière à ce qu'il n'y ait pas de lettres identiques sur une ligne ou une colonne, par exemple ici :

A	B	D	C
B	C	A	D
C	D	B	A
D	A	C	B

L'HISTOIRE

Le problème des huit reines est un célèbre casse-tête combinatoire imaginé en 1848 par le compositeur d'échecs Max Bezzel. L'objectif est de placer huit reines sur un échiquier de sorte qu'elles ne puissent pas s'attaquer les unes les autres, c'est-à-dire qu'aucune reine ne se trouve sur la même rangée, colonne ou diagonale qu'une autre. Depuis lors, de nombreux mathématiciens, dont Gauss, ont travaillé sur ce problème, tandis que des algorithmiciens et des programmeurs, tels que Dijkstra, ont conçu diverses approches pour trouver et calculer des solutions.

INTRODUCTION

LES SOLUTIONS

Initialement, le problème consistait à disposer huit dames sur un échiquier standard de 64 cases de manière à ce qu'aucune ne puisse être attaquée par une autre.

Cependant, le problème généralisé consiste à disposer des reines sur un champ rectangulaire arbitraire, notamment un échiquier carré de dimension n.

Pour un échiquier standard et 8 dames, il existe 92 configurations appropriées.

Mathématiquement, ce problème peut être formulé comme une exigence de remplir une matrice de dimension n avec des zéros et des uns, de sorte que la somme de tous les éléments soit égale à n, mais que la somme des éléments ne dépasse pas 1 dans n'importe quelle colonne, ligne ou diagonale de la matrice.

INTRODUCTION

NOTRE CHOIX

Nous avons choisi le problème des huit reines car nous en avons entendu parler depuis l'école.

Ce problème est utile pour développer des compétences en résolution de problèmes et en programmation, car il exige une analyse minutieuse et la mise en place de solutions créatives.

Le choix de cette tâche comme sujet d'étude est donc compréhensible, et les connaissances et compétences acquises à l'université offrent la possibilité de revisiter ce défi avec une perspective nouvelle et plus approfondie.

Il est également important de souligner que le problème des huit reines a des applications pratiques dans différents domaines, ce qui rend sa résolution pertinente et utile.

En effet, la résolution de ce problème peut avoir des implications concrètes dans des situations du monde réel, telles que la cryptographie, la planification des horaires et l'optimisation des réseaux de transport.

En somme, le problème des huit reines est un défi intellectuel stimulant et enrichissant qui peut aider à développer des compétences en résolution de problèmes et en programmation, tout en ayant des implications pratiques dans différents domaines.

ALGORITHMES

MCMENTALHISTHEBEST

L'algorithme McMentalHisTheBest utilise des listes Python pour stocker le plateau de jeu.

Tout d'abord, l'utilisateur choisit la taille du plateau.

Si la taille est inférieure ou égale à 3, le problème est insoluble et l'affichage ne sera pas correct.

Ensuite, la fonction "affichePossibilite" permet d'afficher le plateau graphiquement en fonction de sa taille pour aider l'utilisateur à visualiser le plateau de jeu.

La fonction "estPossible(plateau, ligne, colonne)" permet de vérifier si une reine peut être placée sur une case donnée sans menacer une autre reine déjà placée sur le plateau.

Pour cela, elle parcourt les diagonales et la colonne de la case testée pour vérifier si une autre reine est déjà présente. Si c'est le cas, elle renvoie "False". Sinon, elle renvoie "True".

La fonction "nQueen(plateau, ligne)" utilise la récursivité pour placer une reine sur chaque ligne du plateau.

Elle parcourt chaque colonne de la ligne en cours et place une reine si "estPossible == True". Ensuite, elle appelle récursivement "nQueen" pour la ligne suivante jusqu'à ce que toutes les lignes soient remplies. Si elle atteint la dernière ligne du plateau, elle affiche la solution en console.

ALGORITHMES 

PIZZADIÈTE

L'algorithme PizzaDiète utilise des dictionnaires Python pour stocker le plateau de jeu.

Tout d'abord, l'utilisateur est invité à saisir la taille du plateau de jeu. Si la taille est inférieure ou égale à 3, l'affichage sera incorrect car le problème est insolvable dans ces conditions.

La fonction "affichePlateau" est utilisée pour afficher le plateau de jeu dans la console en fonction de la taille choisie par l'utilisateur.

Elle prend en paramètre le plateau de jeu, qui est représenté sous forme d'un dictionnaire dont les clés sont les indices de ligne et les valeurs sont les indices de colonne où les reines sont placées.

La fonction "peutPoser" est utilisée pour déterminer si une reine peut être placée à une position donnée sur le plateau de jeu.

Elle prend en paramètre le plateau de jeu et une clé représentant la ligne et une valeur correspondant à la colonne.

Cette fonction parcourt les diagonales et la colonne de la reine que l'on souhaite analyser. Si elle croise une reine déjà placée, elle renvoie False, sinon elle renvoie True.

ALGORITHMES

PIZZADIÈTE

La fonction "Reine" est la fonction principale qui résout le problème des N-Reines en utilisant une approche récursive.

Elle prend en paramètre le plateau de jeu et la clé (ou autrement dit la ligne) du pion. Elle parcourt chaque ligne du plateau de jeu et place une reine sur chaque colonne possible si elle peut être placée sans menacer les reines déjà placées sur le plateau.

Si elle trouve une position où elle peut placer une reine, elle appelle récursivement la fonction "Reine" avec la clé suivante pour la ligne suivante. Si elle n'a pas trouvé de position valide, elle retourne False.

ALGORITHMES

DIGITALOVA

L'algorithme des N-Reines utilise la programmation orientée objet de Python pour stocker le plateau de jeu et créer une solution à chaque itération. Si la taille du plateau est inférieure ou égale à 3, le problème est insoluble et l'affichage ne sera pas correct.

L'algorithme utilise une classe Plateau pour créer et stocker le plateau de jeu. La classe Plateau contient plusieurs méthodes pour aider à créer et à manipuler le plateau. Tout d'abord, la méthode init permet de créer un plateau de jeu de taille donnée, avec des reines placées de manière aléatoire. La méthode str permet d'afficher le plateau de jeu sous forme graphique, avec des "Q" représentant les reines.

La méthode estMenacer calcule le nombre de reines qui sont menacées par d'autres reines, en parcourant le plateau de jeu pour voir si des reines sont sur le même axe (horizontalement, verticalement ou en diagonale). Le score de menace est calculé en additionnant le nombre de reines menacées sur chaque axe.

La méthode deplacerReine permet de déplacer une reine donnée sur une nouvelle colonne, pour trouver une nouvelle position qui ne menace pas d'autres reines. Cela est utilisé dans la méthode principale, RechercheSolution, pour itérer sur différentes positions de reines jusqu'à ce qu'une solution soit trouvée.

ALGORITHMES

DIGITALOVA

La classe RechercheSolution utilise le principe de la "température" pour trouver une solution. La température représente le nombre de déplacements restants nécessaires pour trouver une solution valide. La méthode getTemperature calcule la température actuelle du plateau de jeu en fonction du nombre de reines menacées.

La méthode principale, main, effectue la recherche de solution en déplaçant les reines et en calculant le score de menace pour chaque itération. Si le score de menace est de 0, une solution a été trouvée et l'algorithme s'arrête. Sinon, une reine est déplacée aléatoirement et le score de menace est recalculé pour voir si la nouvelle position est meilleure. Si la nouvelle position est meilleure, elle est gardée, sinon elle est rejetée avec une certaine probabilité pour éviter de se coincer dans un minimum local. Le processus est répété jusqu'à ce qu'une solution soit trouvée.

COMPARAISON 

Lorsqu'on parle de performance d'un programme informatique, on fait référence à sa capacité à résoudre un problème de manière efficace et rapide. Pour mesurer la performance d'un programme, on utilise souvent le temps de calcul, qui correspond au temps nécessaire à l'exécution du programme pour résoudre un problème donné.

Dans notre cas, nous avons comparé deux algorithmes différents pour résoudre le problème du placement des 8 dames aux échecs : PizzaDiète et McMentalHisTheBest. Pour mesurer leur performance, nous avons utilisé la bibliothèque "timeit" de Python, qui permet de mesurer le temps de calcul d'un programme.

Nous avons également noté que la performance d'un algorithme peut varier en fonction de la taille du plateau de jeu, c'est-à-dire du nombre de cases sur lequel les dames doivent être placées. Ainsi, il est important d'optimiser l'algorithme pour des tailles de plateau plus importantes.

En comparant les deux algorithmes, nous avons constaté que celui qui utilise les listes Python est plus performant que celui qui utilise les bibliothèques Python. Sur l'ordinateur de Barbara, l'algorithme qui utilise les listes a une durée d'exécution de 65,2 secondes, tandis que l'algorithme qui utilise les bibliothèques a une durée d'exécution de 69,8 secondes.

COMPARAISON

Cependant, le choix de l'algorithme dépend également de la préférence de l'utilisateur en termes de facilité d'utilisation et de lisibilité du code. En effet, un algorithme peut être plus rapide mais plus difficile à comprendre et à utiliser, tandis qu'un autre peut être moins rapide mais plus facile à utiliser et à modifier.

Enfin, nous avons présenté notre troisième algorithme, Digitalova, qui ne peut pas être comparé aux deux premiers car il ne renvoie qu'une seule solution à la fois. Cependant, son affichage est plus lisible, ce qui peut être un avantage dans certaines situations.

En résumé, le choix de l'algorithme dépend de plusieurs facteurs, tels que la taille du plateau de jeu, la performance souhaitée, la facilité d'utilisation et de modification, ainsi que la lisibilité du code. Il est important de prendre en compte tous ces facteurs pour choisir l'algorithme le plus adapté à sa situation.

CONCLUSION

Bien que le problème des 8 dames aux échecs ne soit pas une situation courante dans la vie quotidienne, l'algorithme Python utilisé pour le résoudre peut être appliqué à d'autres problèmes qui impliquent la résolution de problèmes complexes.

Par exemple, la résolution de problèmes de placement peut être appliquée dans la conception de circuits électroniques, la planification de la production, la disposition des magasins et des entrepôts, et la configuration des réseaux.

L'utilisation de l'algorithme de résolution des 8 dames pour ces problèmes peut aider à optimiser les résultats en minimisant les conflits et en maximisant l'efficacité.

De plus, la compréhension de la structure et de la logique de l'algorithme peut aider à améliorer les compétences en programmation et en résolution de problèmes, qui sont des compétences très utiles dans de nombreux domaines professionnels.

En fin de compte, la résolution de problèmes est une compétence précieuse dans la vie de tous les jours, et l'apprentissage de l'algorithme des 8 dames peut contribuer à développer cette compétence.