Rapport de SAE : Exploration algorithmique.

LEBELLEC Etienne, POUPIOT Timéo

Le problème du “*Tour du cavalier*”.

Présentation du problème :

Le problème du cavalier est un problème mathématique qui consiste à trouver un chemin permettant d’explorer toutes les cases d’un échiquier quelconque avec un cavalier. La principale difficulté est que le cavalier se déplace en L, c'est-à-dire de deux cases dans une direction (horizontalement ou verticalement) et une case dans une direction perpendiculaire.

Notre algorithme :

Notre algorithme est plutôt classique, d’abord il crée une grille avec les dimensions renseignées puis il va calculer pour toutes les cases, celles qui lui sont accessibles. Ensuite, ces résultats sont renseignés dans un dictionnaire : les clé sont les points (tuples) et les valeurs sont les points connectés (aussi des tuples) ce qui forme un graphe. Dans la grille que nous avions créés, chacune des cases est initialisées avec la valeur 0 : pour faire simple 0 = disponible et 1 = visitée. Le cavalier part du point indiqué et va sauter sur la première case disponible, sa case actuelle prend la valeur 1 ainsi que la case d’arrivée.

Grâce à notre système, la grille s’actualise en fonction du parcours, le plus simple était de sauvegarder le parcours dans un tableau, pour avoir une solution plus compréhensible. La fonction qui permet de calculer les solutions est une fonction récursive qui va utiliser le point actuel pour aller visiter les voisins non parcourus pour le moment.

Nous avons façonnés notre algorithme de cette façon afin de ne pas avoir une pile d'exécution immense qui pourrait ralentir notre programme, donc de notre point de vue il est relativement bien optimisé

Questions :

1. Solution pour la figure (3)

   [(0, 7), (2, 6), (4, 7), (6, 6), (7, 4), (6, 2), (7, 0), (5, 1), (7, 2), (6, 0), (4, 1), (2, 0), (0, 1), (1, 3), (0, 5), (1, 7), (3, 6), (5, 7), (7, 6), (6, 4), (5, 6), (7, 7), (6, 5), (7, 3), (6, 1), (4, 0), (2, 1), (0, 0), (1, 2), (0, 4), (1, 6), (3, 7), (4, 5), (2, 4), (0, 3), (1, 1), (3, 0), (4, 2), (5, 0), (7, 1), (6, 3), (7, 5), (6, 7), (4, 6), (2, 7), (0, 6), (2, 5), (4, 4), (3, 2), (5, 3), (3, 4), (1, 5), (2, 3), (0, 2), (1, 0), (3, 1), (5, 2), (3, 3), (5, 4), (3, 5), (1, 4), (2, 2), (4, 3), (5, 5)]

2. Solutions pour la figure (4)

   - [(5, 0), (3, 1), (1, 0), (0, 2), (1, 4), (3, 5), (5, 4), (4, 2), (3, 0), (5, 1), (4, 3), (5, 5), (3, 4), (1, 5), (0, 3), (1, 1), (3, 2), (4, 0), (5, 2), (4, 4), (2, 3), (0, 4), (2, 5), (1, 3), (0, 5), (2, 4), (4, 5), (5, 3), (4, 1), (2, 2), (0, 1), (2, 0), (1, 2), (0, 0), (2, 1), (3, 3)]
   - [(3, 2), (1, 1), (3, 0), (5, 1), (7, 0), (6, 2), (7, 4), (6, 6), (4, 7), (2, 6), (0, 7), (1, 5), (2, 7), (0, 6), (1, 4), (0, 2), (1, 0), (3, 1), (5, 0), (7, 1), (6, 3), (7, 5), (6, 7), (4, 6), (6, 5), (7, 7), (5, 6), (3, 7), (1, 6), (0, 4), (1, 2), (0, 0), (2, 1), (4, 0), (6, 1), (7, 3), (5, 4), (3, 5), (4, 3), (6, 4), (7, 6), (5, 7), (3, 6), (1, 7), (0, 5), (1, 3), (0, 1), (2, 0), (4, 1), (2, 2), (0, 3), (2, 4), (4, 5), (5, 3), (7, 2), (6, 0), (5, 2), (3, 3), (2, 5), (4, 4), (2, 3), (4, 2), (3, 4), (5, 5)]

3. Taille quelconque

   Le programme fonctionne pour les échiquiers de taille quelconque

L'algoritme trouvé sur internet :

(Algorithme généré avec chatGPT)

Cette algorithme crée un echiquier qui est en fait un tableau de n fois n pour un nombre n donné ou tout les éléments du tableau sont initialisé à -1. Puis il mets le case aux coordonnées 0,0 (soit tout en haut à gauche) à 0.

La boucle principale parcourt la liste de positions jusqu'à ce qu'elle soit vide. À chaque itération, elle obtient les coordonnées de la dernière position dans la liste et recherche la prochaine case valide à partir des mouvements possibles. Si une case valide est trouvée, elle met à jour la valeur de la case dans le plateau et empile les nouvelles coordonnées. Si une case valide n'est pas trouvée, elle revient en arrière en défilant la dernière position de la pile.

Une fois la boucle terminée, le plateau est affiché avec les valeurs mises à jour pour chaque case, représentant le nombre de déplacements nécessaires pour que le cavalier atteigne cette case. Chaque case est représentée par un nombre entier et affichée avec un padding pour un affichage plus clair.

Principale différence :

- Dans notre algorithme, à l'exécution il est demandé les dimensions de l'échiquier (lignes et colonnes), ainsi que la position du cavalier sur l'échiquier. Alors que dans l'algorithme trouvé sur internet, il n'est demandé que un nombre n qui sera la longueur et la largueur de l'échiquier.
- Dans notre algorithme, on crée, avant d'entrer dans le boucle principale, un dictionnaire contenant une liste des prochaines positions par rapport à une position initiale. L'algorithme trouvé, lui, stocke juste dans 2 listes les mouvements possibles du cavalier (en x, et en y).
- Dans notre algorithme, à la fin de l'exécution, si un chemin est trouvé, on le retourne sous forme de liste de positions, sinon on retourne une liste vide. Le programme trouvé, ne retourne rien, il affiche de manière stylisé, une grille avec le chemin qui commence par 0 et qui s'incrémente au fur et à mesure. S'il n'existe pas de chemin, l'algorithme fait le chemin le plus long qu'il puisse faire et met -1 sur les cases qu'il ne peut pas accéder.

Avantages et inconvénients de notre algorithme :

- ***Avantages :***
    - Notre échiquier est modulable, c'est à dire que l'on peut aussi bien prendre un échiquier classique en 8x8 qu'un échiquier avec un

nombre de colonnes différents du nombre de lignes. (ex : 6x7).

- On peut choisir la position initiale du cavalier tant qu'il reste sur le plateau.
- L'algorithme retourne une liste (vide ou non), ce qui permet de le réutiliser dans un autre programme.
- Le programme est décomposé en 4 fonctions pour une plus grande lisibilité.

- ***Inconvénients :***
  - L'algorithme devient lent à partir d'un échiquier de taille 11x11 (environ 4 secondes pour trouver la solution), et il est de plus en plus lent si on augmente encore le plateau.
  - Pour exécuter le programme il faut d'abord initialiser un graphe, une grille, donner les dimensions de l'échiquier et la position de départ du cavalier ce qui peut paraître long comparé au deuxième algorithme.
  - On considère que l'utilisateur ne fait pas de faute dans la saisie des valeurs (un cavalier en dehors de l'échiquier etc.)

Evolutions possibles pour notre algorithme :

- Nous pouvons surement améliorer l'optimisation du programme pour qu'il puisse s'exécuter rapidement sur de plus grand échiquier.
- Nous pouvons aussi faire un affichage console du chemin du cavalier une fois la recherche terminé.