Compte rendu SAE 2.02

Réalisé par Kilian Hodouin et Maxime Dardenne

[1.0 Présentation du problème 1](#_Toc159833889)

[2.0 Présentation de l’algorithme 1](#_Toc1081302239)

[2.1 Description du programme 2](#_Toc274156895)

[3.0 Comparaison de l’algorithme 2](#_Toc1895523802)

[3.1 Temps d’exécution et taille en mémoire de l’algorithme 1 3](#_Toc750699449)

[3.1.1 Temps d’exécution 3](#_Toc1648838204)

[3.1.2 Taille en mémoire 3](#_Toc2098393689)

[3.2 Temps d’exécution et taille en mémoire de l’algorithme 2 3](#_Toc894300472)

[3.1.1 Temps d’exécution 4](#_Toc375789131)

[3.1.2 Complexité temporelle 4](#_Toc1910429163)

[3.3 Conclusion générale 4](#_Toc1280710955)

[4.0 Proposition d’amélioration de l’algorithme 5](#_Toc1290386403)

[5.0 Conclusion 5](#_Toc1228731714)

# 1.0 Présentation du problème

Le problème du cavalier consiste à trouver une solution pour que le cavalier parcourt toutes les cases d’un échiquier en passant une seule et unique fois sur chaque case et tout en respectant les déplacements du cavalier.

# 2.0 Présentation de l’algorithme

Notre algorithme utilise les graphes.

Voici le principe :

1. Nous plaçons le cavalier sur la case initiale du parcours
2. À partir de cette case nous cherchons les voisins que nous n’avons pas encore visité en allant sur chacun des voisins.
3. Si jamais aucun voisin ne peut être visité alors nous sommes dans une impasse et nous retournons une case en arrière. Puis nous testons si une autre case est disponible.
4. Si tous les chemins empruntés mènent vers une impasse et qu’il n’en existe aucun autre alors le tour du cavalier est impossible
5. Si le programme trouve un chemin alors il retourne l’échiquier et sur chaque case le numéro correspondant à l’ordre de passage sur la case.

Avantages :

* L’algorithme parcours efficacement toutes les cases.

Inconvénients :

* L’algorithme peut être très lent sur un grand échiquier.

### 2.1 Description du programme

* L’algorithme 1 commence par demander la taille de l’échiquier puis le numéro de la case sur laquelle le cavalier commence son parcours.
* L’algorithme est fait en plusieurs fonction. Une pour les déplacements du cavalier. Une pour la recherche du parcours hamiltonien et l’enregistrement de l’ordre de passage sur les cases. Une qui permet de marquer les cases visitées ou non. Et enfin une qui affiche l’échiquier et l’ordre de passage du cavalier sur chaque case.
* La recherche du parcours analyse les cases non visitées sur lesquelles le cavalier peut faire son déplacement en L par rapport à sa position. Si la case sur laquelle il se déplace amène à une impasse et que toutes les cases n’ont pas encore été visitées alors le cavalier fait demi-tour et réessaye un autre parcours.

# 3.0 Comparaison de l’algorithme

Nous comparons notre algorithme avec un algorithme crée à l’aide de listes de listes.

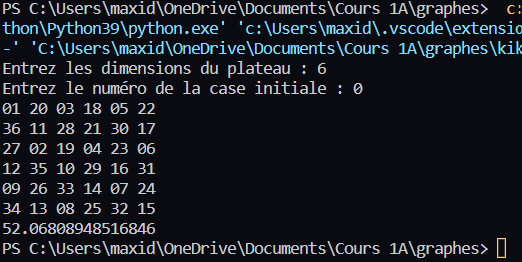
Les deux algorithmes seront comparés sur le temps d’exécution et la taille en mémoire du programme.

Nous allons appeler par la suite l’algorithme 1 celui réalisé à l’aide de graphes et algorithme 2 celui réalisé à l’aide de liste de liste

## 3.1 Temps d’exécution et taille en mémoire de l’algorithme 1

### 3.1.1 Temps d’exécution

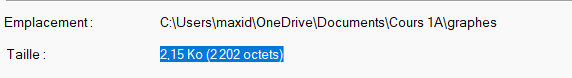
Voici une capture d’écran du temps d’exécution de l’algorithme 1 :



Sur l’avant dernière ligne on peut voir que le temps d’exécution est de 52 secondes environ, ce qui est très long

### 3.1.2 Taille en mémoire

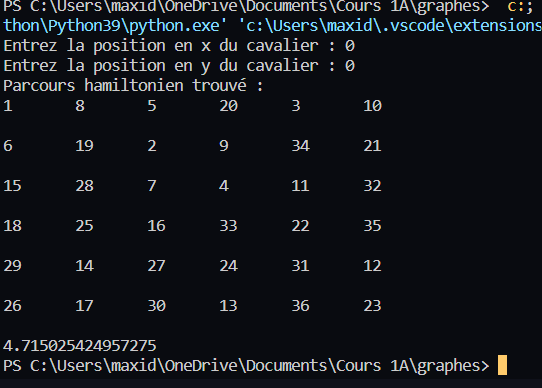
Voici une capture d’écran du volume du fichier de l’algorithme 1 :



## 3.2 Temps d’exécution et taille en mémoire de l’algorithme 2

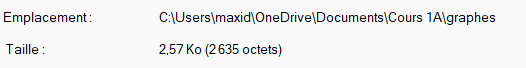
### 3.1.1 Temps d’exécution

Voici une capture d’écran du temps d’exécution de l’algorithme 2 :



### 3.1.2 Complexité temporelle

Voici une capture d’écran du volume du fichier de l’algorithme 2 :



## 3.3 Conclusion générale

Les deux programmes utilisés peuvent nous permettre de trouver un parcours hamiltonien pour le cavalier. Mais l’algorithme 2 est bien plus performant car sa vitesse d’exécution est rapide par rapport à l’algorithme 1. Néanmoins l’algorithme 1 fait en graphe permet mieux modéliser l’échiquier et de se le représenter facilement dans la tête.

# 4.0 Proposition d’amélioration de l’algorithme

Nous pouvons améliorer notre algorithme 1 en optimisant le programme et ainsi permettre de réduire le temps d’exécution.

# 5.0 Conclusion

On peut conclure ce compte rendu en disant que nos deux algorithmes sont fonctionnels mais que l’un est plus lent mais peu volumineux tandis que l’autre est rapide mais plus volumineux. Le programme en graphe est efficace pour pouvoir interpréter plus facilement les déplacements du cavalier. Malgré son temps d’exécution très long, c’est une méthode efficace pour répondre au problème du tour du cavalier.

Cette SAÉ nous a permis de proposer diverses solutions à un problème donné. La programmation de plusieurs programmes, leurs comparaisons et leur analyse étaient des étapes clés pour résoudre le problème. La communication et la mise en commun constante au sein de notre binôme nous a permis de réaliser rapidement et efficacement ce projet très intéressant à suivre.

Diaporama : https://www.canva.com/design/DAFfgFXB3ww/gmjvCVEALchr03ctAepF5Q/view?utm\_content=DAFfgFXB3ww&utm\_campaign=designshare&utm\_medium=link&utm\_source=publishsharelink