UT - Maubeuge
BUT – Informatique

Année universitaire 2024-2025
Groupe BUT 1 TPB1
Le 20 Janvier 2024

Compte Rendu - SAÉ 104
Comparaison d'approches algorithmiques

COLLOT Grégoire

Université Polytechnique
HAUTS-DE-FRANCE

Sommaire

Introduction

Présentation du sujet

Je suis étudiant en première année de BUT Informatique à l'IUT de Maubeuge.

Dans le cadre de la ressource SAÉ 102, j'ai réalisé une comparaison de différents algorithmes afin d'en déduire la méthode la plus adaptée à la tâche demandée.

La tâche principale qui nous a été demandée fût de résoudre un mastermind le plus rapidement possible et en moins de coup possible. Par ailleurs, le Mastermind fut un projet réalisé plus tôt dans l'année.

Ainsi, je me suis concentré d'abord sur le bon fonctionnement d'un premier algorithme afin de résoudre le mastermind. A la suite de cette réalisation, j'ai créé un affichage permettant de comparer les différentes versions du programme, j'ai ainsi pu voir facilement les bonnes directions à prendre.

Je vous remercie d'avance, pour le temps que vous m'accordez en lisant ce compte rendu,

Présentation des objectifs

J'ai eu trois objectifs à répondre.

Le premier étant de faire un algorithme de base afin de résoudre une partie de Mastermind.

Mettre en place une méthode de comparaison précise afin de pouvoir comparer les différentes méthodes implémentées.

Finir par une méthode appliquée la plus optimale possible déduite des comparaisons précédentes.

Présentation des attendus

Pour mon premier algorithme, l'objectif est de trouver la solution qu'importe le code secret de la partie de Mastermind. Ainsi, cela m'apportera une base solide pour améliorer la rapidité et l'efficacité de résolution des parties.

Ensuite, pour pouvoir comparer les différents algorithmes je dois prendre en compte la rapidité de résolution de la partie, son nombre de tour pour résoudre et enfin le temps pris entre chaque coup.

Ainsi il est nécessaire de créer une fonction permettant de lancer un nombre de parties précis pour chaque algorithme et de sauvegarder les données dans un fichier JSON afin de pouvoir mettre à jour les résultats de nos tests passés en fonction de nouveaux algorithmes créés et de faciliter la manipulation des données récoltées.

De plus, l'utilisation de la bibliothèque Matplot permet la réalisation de graphiques pour mieux visualiser les résultats.

Il serait idéal que je réalise des versions exhaustives avec différentes méthodes d'approche de résolution de l'algorithme afin de réussir à trouver la meilleure solution de résolution d'une partie de Mastermind.

Enfin, il est important que le dernier algorithme soit le plus performant, il faudra bien prendre en compte le temps de résolution moyen et le nombre de tour moyen en compte.

Benchmark des solutions

Description des méthodes

A travers ce projet, je suis partie sur trois méthodes différentes. Ces méthodes me sont venues suite aux discussions avec mon professeur d'algorithme monsieur Polet qui à su me guider ainsi qu'avec les résultats des différents comparatifs.

La première fut de trouver les couleurs présentes dans la combinaison afin d'éliminer toutes les combinaisons ne contenant pas ses couleurs et d'en déduire la bonne combinaison.

Ma deuxième fut d'enlever la détection de bonnes couleurs mais de directement comparer le résultat d'une combinaison envoyée aléatoirement et de retirer celles qui ne correspondent pas au résultat attendu.

Enfin, ma dernière fut de reprendre la deuxième méthode, mais de rajouter un guide afin d'envoyer la combinaison la plus probable pour les deux premiers tours afin d'optimiser le nombre de combinaisons à supprimer et de réduire le nombre de tours pour résoudre une partie.

Comparaison entre les méthodes

La première méthode utilise une logique classique qu'un joueur peut avoir dans une partie de Mastermind banale. Il pourra facilement déduire la solution à une partie au fur et à mesure des coups en déduisant les bonnes couleurs.

La deuxième méthode se repose sur le calcul de chaque combinaison possible et de la réduire à chaque tour en effectuant une comparaison des combinaisons valides.

Enfin la troisième est un compromis entre les deux autres méthodes, qui essaie de combiner logique, calcul et aléatoire à la solution.

Nous pouvons en déduire que la solution n°3 serait la plus performantes pour de la résolution de partie de Mastermind.

Explication de votre solution

Description textuelle des méthodes implémentées

Pour la première méthode, au premier tour, je génère toutes les combinaisons possibles avec la fonction récursive “combinaison”. Cette fonction par de la première couleur possible et va se rappeler tant qu’elle n’as pas fais toutes les autres couleurs présentes dans la liste de couleur avant de passer à la couleur suivante.
Résultant ainsi à un emboitement de couleur qui me formera la liste de toutes les combinaisons possibles.
Je finis par envoyer une combinaison de cinq couleurs identiques comme première combinaison de test.

Tant que je n’ai pas trouvé les 5 couleurs différentes je vais continuer à envoyer des combinaisons de couleur identique. Dès qu’une combinaison testée me renvoie une couleur bien ou mal placée, je parcours toute la liste des combinaisons possibles et supprime toutes combinaisons qui ne contiennent pas les bonnes combinaisons avec la fonction “removeCombinason”.
Si une combinaison testée ne me renvoie pas une couleur bien placée ou mal placée, dans ce cas je supprime toutes les combinaisons ayant cette couleur avec “removeCouleur” qui ne garde que les combinaisons ayant le même nombre de bien placée.

Enfin, une fois toutes les couleurs trouvées et toutes les combinaisons valides restantes, je renvoie une combinaison aléatoire de ma liste. Et à chaque fois, je compare le résultat obtenu avec celles qui sont encore possibles. Je finis par obtenir le résultat à la solution.

Pour la deuxième méthode, je crée aussi une liste contenant toutes les possibilités de combinaisons, mais je renvoie de suite une combinaison aléatoire de celle-ci.

Ainsi il me suffis à chaque tour de faire une comparaison en utilisant mon résultat précédent comme combinaison secrète temporaire et d’enlever toutes les combinaisons qui ne concordent pas au même nombre de bien et mal placé avec la fonction “removeComparaisonAll”, je renvoie alors une combinaison aléatoire de la liste des combinaisons possibles et répète l’action jusqu'à trouver la bonne combinaison.

Pour la dernière méthode, elle reprend la deuxième méthode, mais lors des deux premiers tours elle renvoie des combinaisons prédéfinies. Malheureusement, par manque de temps je n’ai pas pu pousser cette méthode comme je le souhaitais, néanmoins, j’ai pu réaliser des tests avec celle-ci.

Adaptation aux structures de données imposées par le sujet

Choix des tests

Pour les tests j'ai tout simplement réalisé une comparaison des temps moyen et du nombre de tours moyen pour résoudre une partie avec une combinaison secrète identique à tous les algorithmes.

De plus, j'ai aussi pris en compte la différence du temps d'exécution entre chaque tour qui permet de savoir à quel moment la méthode fait le plus de travail.

Ses tests nous permettent de comparer facilement la réussite de chacun des algorithmes et de définir quelle est la méthode la plus optimisée pour résoudre rapidement ou efficacement une partie.

Comparaison des performances et stratégies

Nous allons pouvoir comparer les différentes approches grâce au tableau suivant et aux trois graphiques réalisés avec la bibliothèque Matplot.

Les méthodes ont été nommées en fonction de l'ordre de leur réalisation.

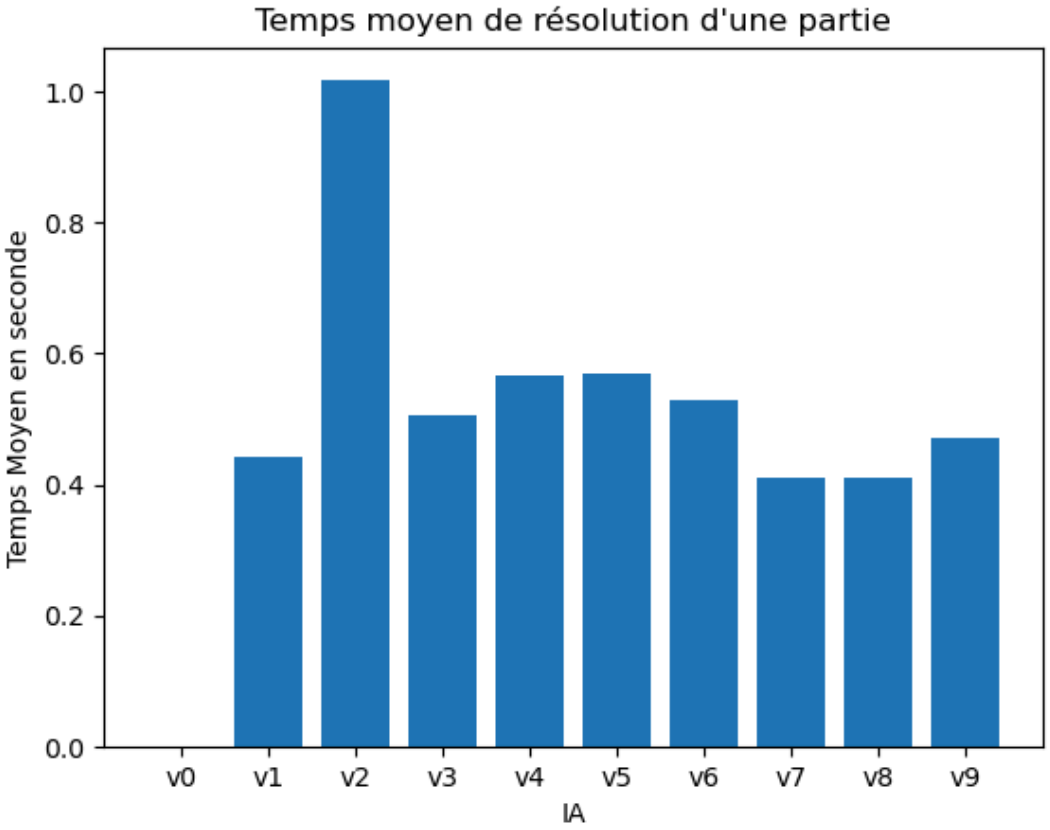
La v0 correspond à une méthode sans aucune logique, qui ne renvoie que 5 couleurs aléatoires.

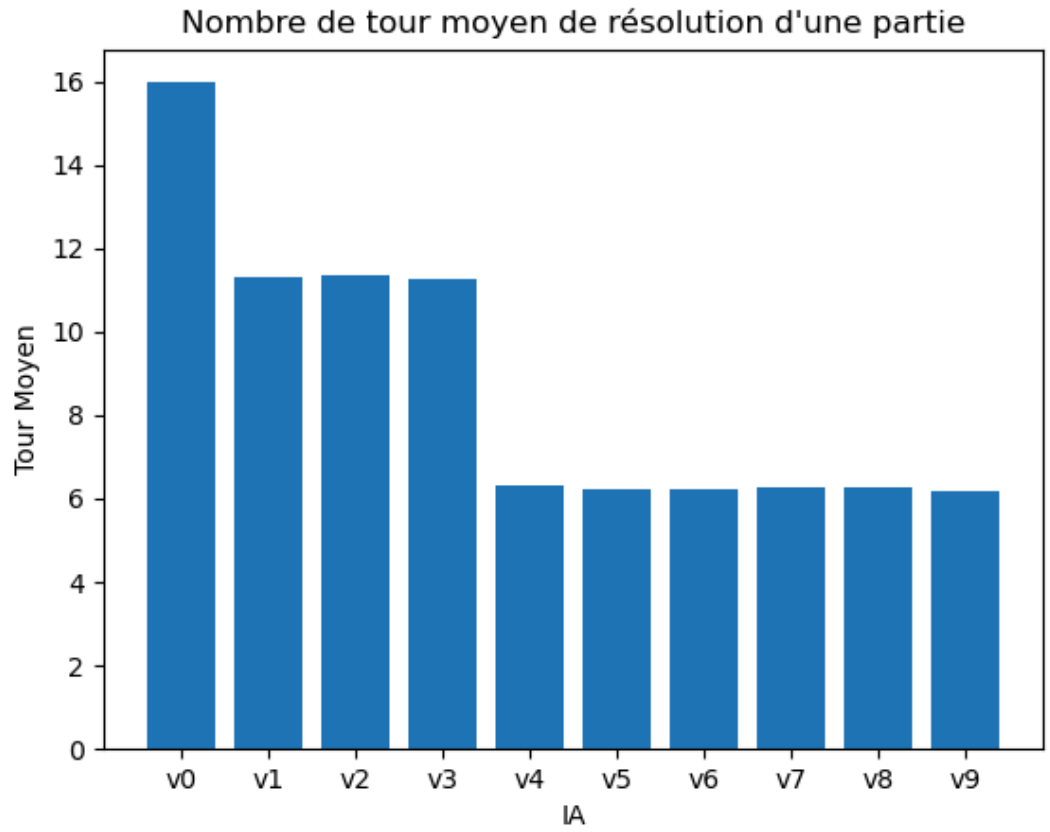
De v1 à v3, ce sont les algorithmes qui utilisent la méthode n°1.

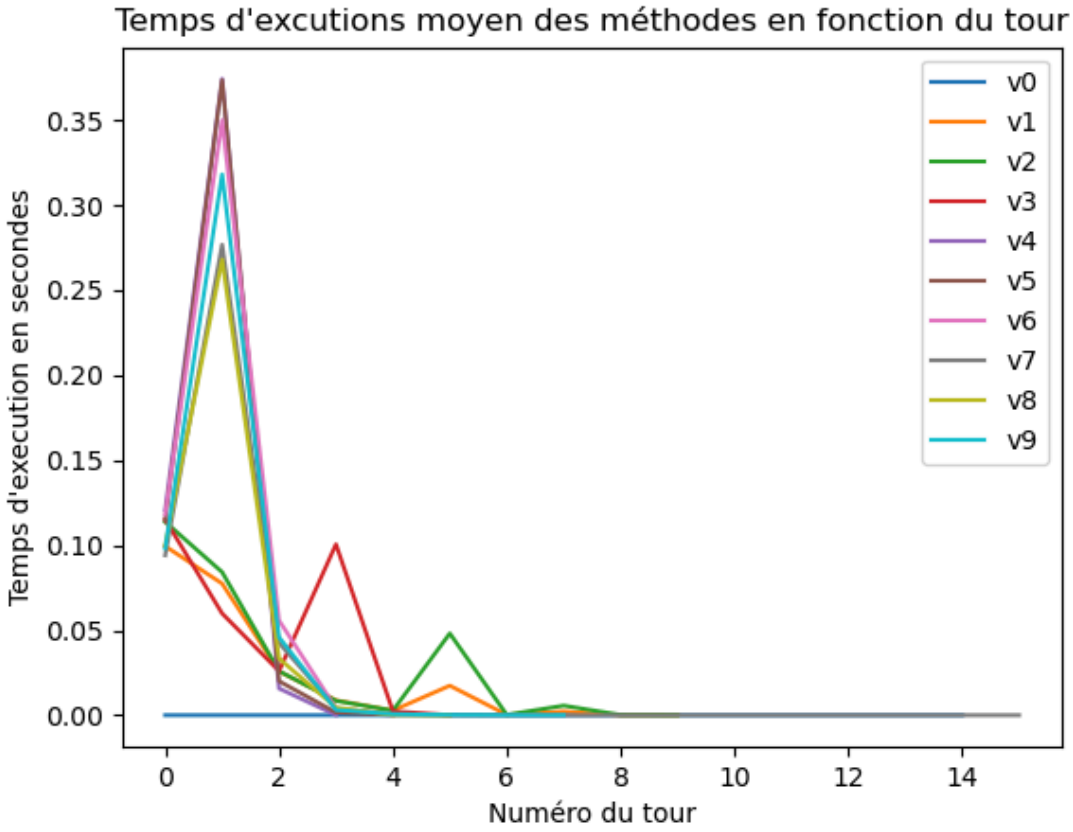
De v4 à v8, ce sont les algorithmes qui utilisent la méthode n°2.

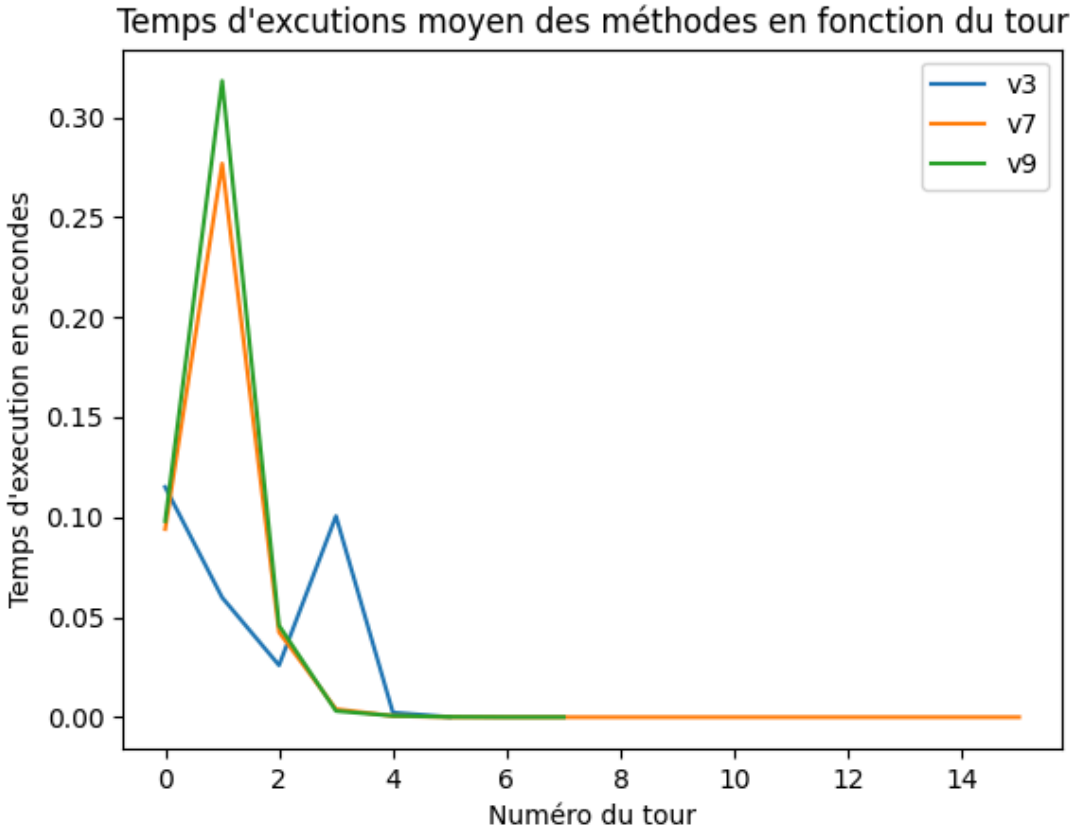
Et pour la v9, l'algorithme correspond à la méthode n°3.

Tableau comparatif des méthodes implémenté					
Méthodes	Temps Moyen	Nombre de coups moyen	Partie non résolue	Pourcentage de victoire	Nombre de test
v0	0.000	16.0	299	0.33%	300
v1	0.443	11.3	0	100%	300
v2	1.017	11.3	0	100%	300
v3	0.507	11.3	1	99.67%	300
v4	0.567	6.3	0	100%	300
v5	0.568	6.2	0	100%	300
v6	0.530	6.2	0	100%	300
v7	0.412	6.3	0	100%	300
v8	0.411	6.3	0	100%	300
v9	0.471	6.2	0	100%	300


Temps moyen de résolution d'une partie
Temps Moyen en seconde
1.0
0.8
0.6
0.4
0.2
0.0
v0
v1
v2
v3
v4
v5
v6
v7
v8
v9
IA


Nombre de tour moyen de résolution d'une partie
Tour Moyen
16
14
12
10
8
6
4
2
0
v0
v1
v2
v3
v4
v5
v6
v7
v8
v9
IA


Temps d'excutions moyen des méthodes en fonction du tour
Temps d'execution en secondes
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0
2
4
6
8
10
12
14
Numéro du tour
v0
v1
v2
v3
v4
v5
v6
v7
v8
v9


Temps d'excutions moyen des méthodes en fonction du tour
Temps d'execution en secondes
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0
2
4
6
8
10
12
14
Numéro du tour
v3
v7
v9

Commençons par la méthode “bête” la v0. On peut voir qu’elle n'a réussi à trouver la solution qu’une seule fois parmi les 300 tests réalisés. Nous ne pouvons pas du tout nous fier à cette méthode même si elle a un temps d'exécution extrêmement faible.

Continuons par la première méthode, nous pouvons remarquer qu'elle à beau avoir une moyenne de 11 tours de résolution, mais elle a aussi une partie où la solution n'a pas été trouvée.
Nous pouvons le voir au niveau du tableau pour la version v3.

Ainsi, nous pouvons de suite mettre de côté cette méthode, car elle n’est pas capable de résoudre tout type de combinaison. Par ailleurs, la combinaison qui n’a pas été résolue est la suivante: ["Gris", "Marron", "Vert", "Orange", "Rouge"].
Nous pouvons supposer que c’est du la forte présente de couleur qui sont en fin de liste.

L’algorithme à besoin d’utiliser ses 9 coups pour trouver toute les couleurs possibles, résultat à seulement 6 essais restant pour résoudre le bon ordre.

Intéressons-nous sur la deuxième méthode, nous pouvons voir qu’elle réussie à presque réduire du double le nombre de coups moyens par rapport à la première méthode. Nous passons d’environ 11 à 6 tours. Certes nous perdons du temps d'exécution, mais cela reste minime par rapport aux résultats finaux.
Nous pouvons donc en déduire que la recherche de bonnes couleurs n’est pas la méthode la plus performante.

Enfin, pour la méthode n°3 celle-ci n'a pas pu être approfondie du a un manque de temps. Elle se repose sur la méthode n°2 il n’y qu’une modification au premier et deuxième tour, on peut voir grâce au graphe n°3 que le gain de temps est négatif.

Les différentes versions d’implémentation des trois méthodes se diffèrent par de l'optimisation.

Par exemple pour la v2 et v3 la fonction permettant de déduire le nombre de pions bien et mal placé fut modifiée résultant en un temps d'exécution différent.

Et pour la v4 et v8, c’est le fait d’enlever toutes fonctions, variables et print() inutiles ainsi que de l’optimisation des fonctions de calcul de bien et mal placé, de déduction des combinaisons valides.

Nous venons de voir que la meilleure méthode implémentée dans ce projet est bien la deuxième méthode. Plus précisément la version 7 qui se base sur des propositions aléatoires de combinaison possible avec une réduction au fur et à mesure des possibles.

Analyse critique du travail

Points atteints et non atteints

A travers ce projet, j'ai atteint tous mes objectifs définis. J'ai réussi à créer une méthode de base par rapport à mon approche initiale, suivi par une fonction permettant de comparer facilement les différentes méthodes et enfin par la création d'une méthode plus optimisée et idéale pour la résolution de parties de Mastermind.

Analyse sur l'organisation du travail

Mon organisation du travail à été bien orienté grâce aux deux séances de TD (Travaux Dirigé) rattaché à ce projet. Durant la première séance et une séance de soutien, j'ai pu être guidé sur la création d'une fonction permettant de récupérer une liste de toutes les combinaisons de couleurs possibles pour une partie de MasterMind.

Ainsi avec cette base, j'ai pu me consacrer sur l'implémentation de ma méthode en fonction, permettant rapidement d'avancer dans mon travail et d'avoir une méthode fonctionnelle.

De plus, lors de la deuxième session de TD, j'avais des difficultés pour trouver une méthode de résolution lorsque ma solution a trouvé les bonnes couleurs. Je ne voyais pas comment différencier mes dernières combinaisons possibles pour trouver la bonne.
J'ai ainsi pu comprendre grâce à l'aide de mon professeur comment prendre la dernière combinaison donnée et comparer le résultat obtenu (pions bien et mal placé) afin de supprimer toutes les combinaisons qui ne correspondent pas à ce résultat.

Enfin, le fait de me consacrer sur une fonction de comparaison des différentes méthodes dès que j'ai eu une méthode fonctionnelle m'a permis de rapidement savoir si mes futurs algorithmes se dirigeaient vers une bonne direction grâce au gain ou perte de temps/tours.

Remarque : Je pense que j'aurai pu mieux m'organiser pour mon rapport en prenant des notes de ce que j'ai réalisé à chaque session de TP (Travaux Personnel) pour réaliser mon compte rendu plus rapidement.

Bilan global du projet

Nous venons de voir comment j'ai répondu aux différents objectifs de ce projet en utilisant une méthode de comparaison précise pour déterminer le meilleur algorithme de résolution de partie de Mastermind.

J'ai pu me rendre compte de l'importance d'avoir une méthode précise en tête afin de pouvoir l'implémenter, l'importance de chaque ligne de code résultant à un meilleur temps d'exécutions et de tours utilisés.

Ainsi, nous pouvons comparer ce projet à petite échelle face à des projets d'envergure retrouvés en entreprise qui vont être plus complexes et vont demander plus de ressources à l'exécution et pourquoi il est important de savoir trouver les meilleures méthodes de résolution de problème pour éviter les pertes de ressources de l'entreprise.

Enfin, j'ai dû être organisés et attentifs aux détails afin de créer une méthode de résolution de partie de Mastermind.

Je vous remercie pour le temps que vous m'avez accordé en lisant ce compte rendu,

COLLOT Grégoire