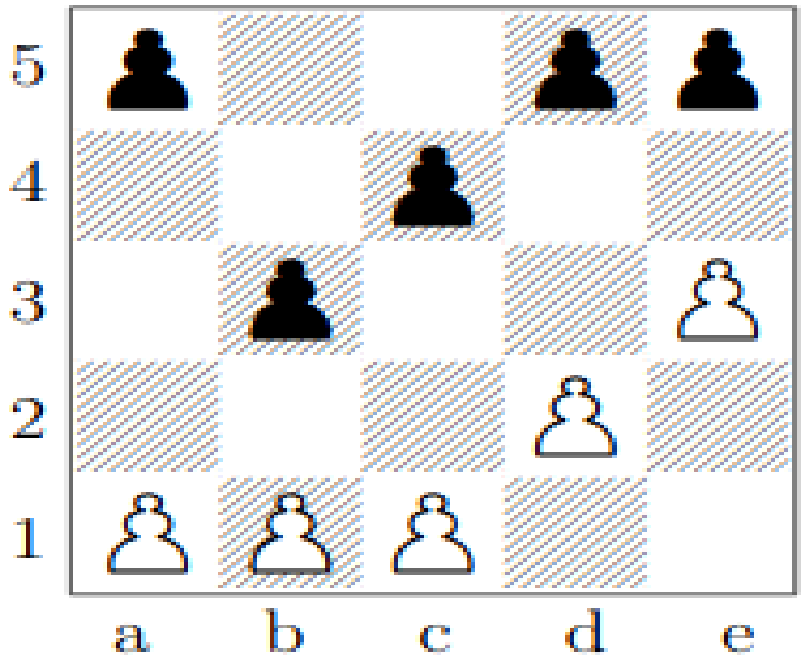
Rapport de TP

Algorithmes et Complexité

Hexpawn



Hamani Rayane
Nguyen Kévin

Master 1 d'Informatique
2020 - 2021

Introduction

Objectif

Le but du jeu est de mettre au point un programme qui joue selon une stratégie optimale, c'est-à-dire qui ne commet jamais d'erreur. S'il peut gagner, alors il joue le coup qu'il faut pour gagner. S'il ne peut pas gagner, alors il joue la meilleure défense possible (il essaie de retarder sa défaite). En effet, dans un jeu comme celui-ci (où il n'y a pas de match nul) on est forcément dans l'une de ces deux situations :

1. Soit le joueur qui commence peut, en jouant correctement, gagner quoi que fasse l'autre.
2. Soit le joueur qui commence, quoi qu'il fasse, va perdre si l'autre joue correctement.

Plus précisément, l'état du jeu (la configuration) est complètement décrit par la position de tous les pions, ainsi que par la couleur du joueur dont c'est le tour.

Question 1 : Calcul d'une configuration

configuration = -1 * (max(successeurs) + 1) si tous les successeur sont positifs.
= -1 * (max(successeurs) - 1) si tous les successeur sont nuls ou négatifs.
= -1 * (min(successeurs) - 1) dans tous les autres cas.

Le calcul de nos configurations est fait par la fonction eval dans notre code.

Question 2 : Version naïve

La principale difficulté de la version naïve était de construire une bonne structure à notre programme. Il y avait différentes façons de faire une fois que la partie était lancée. Nous avons décidé de transmettre l'entièreté des boards avec les coups joués. Ce n'est pas forcément la meilleure méthode mais c'est celle qui nous a paru la plus intuitive au début. En cours de route, nous avons eu quelques difficultés (segfaults, résultats proches

mais différents, etc...) liées à des détails que nous avons peut-être trop vite éludé. Cependant, du fait que notre programme était clair et bien amené, cela nous a permis d'avoir une meilleure vue sur ce qui n'allait pas.

Dans un même temps, contrairement à notre TP précédent (fait en Java), nous avons décidé de nous essayer au C++ afin d'améliorer les performances de notre programme. Ce fut pour nous deux la première fois que l'on code dans ce langage et le TP était d'autant plus intéressant qu'il nous a permis d'apprendre et expérimenter les différentes collections (vector, list, map, array, etc...) dans ce langage. Ce n'est peut-être pas important à relever puisque le choix du langage est libre et n'engage que le binôme qui l'utilise mais je tenais tout de même à le souligner puisque l'un des intérêts de la matière est d'observer à quel point un code bien pensé permet d'économiser des ressources et du temps.

Les tests 1 à 5 et le test 8 ont été passés en récursif sans mémorisation.

Question 3 : Version dynamique

La version dynamique est identique à la version naïve à l'exception près que l'on garde en mémoire les états du jeu, le joueur dont c'est le tour dans ces états, ainsi que le résultat qui interviendra à l'issue des meilleurs coups des joueurs dans ces états.

A chaque début de tour d'un joueur, on regarde si celui-ci n'a pas déjà rencontré l'état dans lequel le jeu est, auquel cas on retourne directement la configuration finale de cet état. S'il n'a pas encore rencontré cet état, on l'ajoutera dès que la partie se terminera. La configuration qui sera retenue sera celle qui avantage le plus le joueur dont c'est le tour.

La complexité temporelle de la version dynamique est considérablement réduite. Cela vient du fait qu'on épargne à notre unité de calcul de jouer et rejouer les mêmes coups à travers les états du jeu déjà connus. De cette façon, on s'assure également que chaque coup joué par un joueur dans un état, sera unique. Cela se fait légèrement au détriment de la complexité spatiale mais pour les exemples de test du TP, c'était négligeable. Les temps de calculs en ont grandement profité cependant ! (4,96s pour résoudre l'exercice)

Tous les tests de la plateforme ont été passés (8 / 8) avec la mémorisation.

Annexes

Le code de notre programme est inclus dans l'archive du rendu.

Pour compiler notre code en exécutable, vous pouvez utiliser la commande suivante :

```
g++  Hexpawn.cpp -g -o Hexpawn
```

Pour utiliser notre programme sur l'un des fichiers de test (ici le fichier Config5X5_15) dans l'archive, vous pouvez utilisez la commande suivante :

```
./Hexpawn < Tests/Config5X5_15
```