École de technologie supérieure LOG320 – Structures de données et algorithmes

Laboratoire Jeu plateau : Phase 1

Jeu de Tic Tac Toe

Durée : 2 séances

Introduction

Le jeu de Tic Tac Toe est bien connu et facile à jouer. Ses règles simples et l'espace de recherche limité en font un jeu idéal pour une première implémentation des algorithmes Minimax et Alpha-Beta car, entre autres, le débogage est plus simple étant donné la taille limitée de l'arbre de recherche. Ce laboratoire servira donc de point de départ pour le laboratoire de jeu de plateau qui demande une bonne compréhension de l'algorithme Alpha-Beta.

Les objectifs de ce laboratoire sont :

- Implémentation de l'algorithme minimax
- Implémentation de l'algorithme Alpha-Beta
- Mieux comprendre l'architecture pour l'implémentation d'un agent intelligent pour un jeu de plateau à deux joueurs.
- Apprendre à débugger des programmes complexes (l'arbre de recherche devient très vite gros) même pour un jeu aussi simple.

Travail à faire

Pour ce laboratoire, vous devez implémenter un agent intelligent capable de jouer au Tic Tac Toe. Normalement, étant donné la faible complexité du jeu et par conséquent, de son espace de recherche, votre agent ne devrait jamais perdre une partie. Dans le pire des cas, votre agent devrait toujours obtenir un match nul contre un joueur qui joue parfaitement.

Un modèle d'architecture est disponible sur Moodle. Les classes suivantes sont fournies :

- Mark.java : définition des pièces du jeu, soit le X et le O.
- Move.java : classe qui contient les informations relatives au mouvement, autrement dit, la position à laquelle une pièce est jouée.
- Board.java : classe qui contient les informations du plateau. Les signatures de deux méthodes sont fournies dans cette classe :
    - public void play(Move m, Mark mark) qui place la pièce mark à la case spécifiée par move
    - public int evaluate(Mark mark) qui évalue la position du point de vue du joueur Max. Elle doit retourner 100 pour une victoire, -100 pour une défaite et 0 pour un match nul.

  Il est particulièrement important de ne pas modifier les signatures de ces méthodes, elles seront utilisées pour tester votre implémentation. C'est la même consigne pour le nom de la classe et la signature du constructeur.

  De plus, cette classe devrait contenir votre méthode pour générer les coups possibles. Vous pouvez ajouter toutes les méthodes dont vous avez besoin, mais sans modifier les différentes définitions déjà présentes.

- CPUPlayer.java : classe qui contient les méthodes qui implémentent l'agent intelligent. Les signatures de trois méthodes sont fournies dans cette classe :
    - public int getNumOfExploredNodes() qui retourne le nombre de nœuds explorés durant une recherche MinMax ou Alpha-Beta (incluant les positions terminales). La variable numExploredNodes devrait être incrémentée à chaque appel de votre méthode Minimax ou Alpha-Beta. Idéalement, l'incrémentation est la première chose qui est faite dans ces deux méthodes.
    - ArrayList<Move> getNextMoveMinMax(Board board) retourne la liste de tous les coups qui ont la même valeur en utilisant l'algorithme Minimax. Le nombre de coups avec la même valeur dépend de la configuration des pièces sur le plateau.

- ArrayList<Move> getNextMoveAB(Board board) retourne la liste de tous les coups qui ont la même valeur en utilisant l'algorithme Alpha-Beta. Le nombre de coups avec la même valeur dépend de la configuration des pièces sur le plateau.

Le constructeur de cette classe reçoit en paramètre le joueur Max (X ou O). Tout comme pour la classe précédente, il est important de ne pas modifier les signatures des méthodes, mais vous pouvez ajouter d'autres méthodes au besoin. Au minimum, vous devrez ajouter votre implémentation de Minimax et Alpha-Beta (deux méthodes distinctes)

Vérification du programme

Lors de la remise, votre programme sera vérifié à l'aide d'un programme de test. Toutes les méthodes présentent dans le modèle d'architecture, y compris les constructeurs peuvent être testé. Si vous modifiez les signatures de ces méthodes, ou les noms de classes, vous pourriez être pénalisé jusqu'à obtenir 0 pour le laboratoire.

Votre programme devrait pouvoir se compiler avec la commande suivante :

```
javac Mark.java Move.java Board.java CPUPlayer.java Test.java
```

où Test.java est une classe avec un main qui peut appeler n'importe quelle méthode déclarée dans la version originale de Mark.java, Move.java, Board.java ou CPUPlayer.java.

Vous ne devriez pas avoir à ajouter de classes supplémentaires, mais vous pouvez le faire si vous voulez.

Rapport de laboratoire

Il n'y a pas de rapport de laboratoire à remettre pour ce travail.

Barème de correction

Cette première phase du laboratoire du jeu de plateau vaut 3 points sur les 15 du laboratoire complet.

L'entièreté des points est allouée à la bonne fonctionnalité des algorithmes Minimax et Alpha-Béta.

Le laboratoire est évalué avec des tests automatisés qui vont vérifier le nombre de nœuds explorés et si les coups sélectionnés sont les bons. Vous devez donc :

- Implémenter les algorithmes de bases pour Minimax et Alpha-Béta, rien de plus ;
- Que votre agent peut jouer les 'X' ou les 'O' ;
- Le nombre de nœuds explorés est le nombre d'appels à la fonction Minimax ou Alpha-Béta selon le cas ;
- Vous devez vous assurer que votre programme compile et que les signatures des méthodes sont les bonnes. Vous n'avez pas à fournir la classe Test mais vous devriez en faire une pour vous assurer dans la compilation ;
- Pour le générateur de coups, il est préférable d'explorer les cases du plateau de gauche à droite et de haut en bas comme le programme de test.