TP 4 – Récursivité et application à l'algorithme minimax (IA).

Q. Giorgi.

Recommandations

Lisez très attentivement le sujet

ATTENTION : La présentation, la lisibilité, ainsi que **le respect des consignes** seront prises en compte dans la notation.

Comme pour chaque TP, vous :

- Réaliserez un makefile pour la compilation des fichiers sources.
- Utiliserez les options de compilation -Wall -g
- Devrez rendre, selon les consignes de l'enseignant, un code source testé et qui se compile correctement !

Travail préparatoire, rappels et compléments de cours.

Lecture préparatoire :

- Lisez cet article afin de connaitre le jeux tic tac toe:

https://fr.wikipedia.org/wiki/Tic-tac-toe

- Lisez cet article afin de connaitre l'algorithme du minimax :

MinMaxL2.pdf (attention il ya un bug dans Tab1, à vous de le trouver…)

- Lisez et comprenez les codes source fournis.

- Exercice d'entrainement à réaliser avant le TP (exercice non noté, des indications seront données sur Chamilo pour ceux qui n'y arriveraient pas, il est très fortement recommandé de vous tester sur ces exercices qui normalement doivent vous prendre pas plus de 5 minutes)

Ecrire une fonction récursive permettant de calculer le factoriel d'un nombre positif.

Travail préparatoire à rendre en début de TP (ce travail sera évalué) :

1 - Ecrivez en C la fonction d'évaluation qui permet de savoir si une partie est nulle, gagnée ou perdue pour le joueur P1 (en fonction des commentaires)

2 - Reflechissez à, et indiquez la condition d'arrêt de la recursivité de minimax (sera implémenté dans la fonction fini)

Algorithme du minimax pour TIC TAC TOE

Vous devrez implémenter une intelligence artificielle par l'algorithme basée sur minimax.

Dans le cas de tic tac toe, voici comment cela se décline : Il vous faut rechercher tous les coups possibles à partir d'une position, et parcourir ainsi l'arbre des coups possibles, chaque niveau correspond aux coups possibles pour un utilisateur. plus vous avez de niveaux, plus vous anticipez les coups.

Avec tic tac toe, l'ensemble des coups possibles n'est pas très important, il est donc possible de parcourir tout l'arbre des coups possibles jusqu'à la fin de partie, vous n'aurez donc plus une « évaluation en cours de partie », mais bien la notion finale de gagné, perdu ou nul (contrairement à des jeux comme othello, le go ou les echecs, Même avec des ordinateurs puissants et beaucoup de mémoire, il est illusoire de vouloir explorer l'ensemble des coups possibles sur une grande profondeur.)

Quand cela sera votre tour de jouer (vous êtes Player1), vous aurez tendance à aller vers des branches de l'arbre vous favorisant (max : car la fonction d'évaluation est positive si P1 gagne), quand c'est le tour de l'IA (sera P2), il aura tendance à aller vers des branches vous défavorisant (min : car la fonction d'évaluation est négative si P1 perd).

Il n'est pas necessaire de stocker l'ensemble de l'arbre, un parcours en profondeur, et le stockage du score pour le niveau en cours suffit.

Vous trouverez sur Internet des optimisations :

- algorithmique : élagage alpha-beta.
- De code : negamax.

Il n'est pas demandé de les implémenter.

Question : Implémenter l'algorithme décrit ci-dessus en utilisant les fichiers sources fournis, et la compilation séparée comme indiquée par l'enseignant.