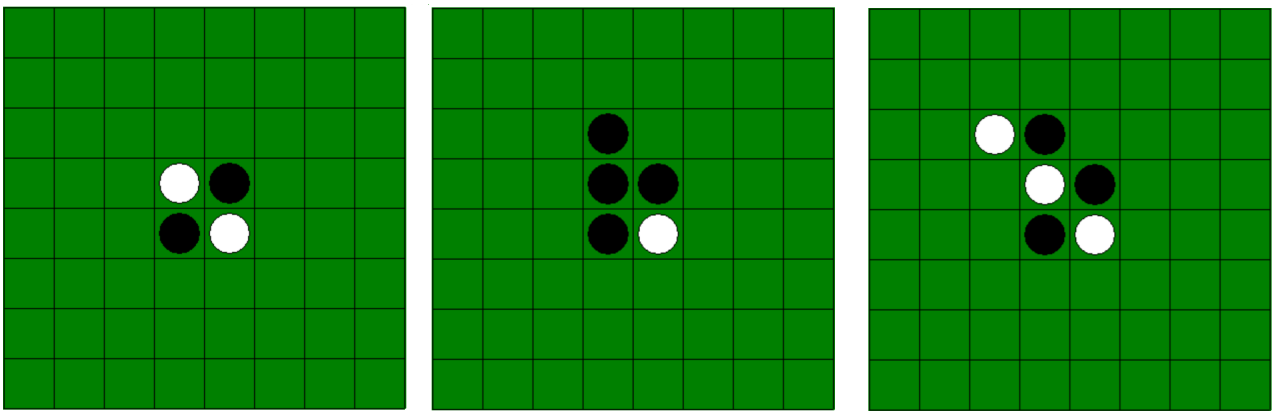
Mardi 2 avril 2024. MP 2023/2024.

T.P. d'informatique n° 23

Othello (ou Reversi) et algorithme minimax.

Othello (aussi appelé Reversi) est un jeu de société opposant deux joueurs. Il se joue sur une grille unicolore de 8 x 8 cases. Les joueurs disposent de 64 pions bicolores, noirs d'un côté et blancs de l'autre. En début de partie, quatre pions sont déjà placés au centre de la grille : deux noirs, en e4 et d5, et deux blancs, en d4 et e5. Chaque joueur (caractérisé par l'une des couleurs i.e. noir ou blanc) pose l'un après l'autre un pion avec la face visible de sa couleur sur la grille sur une case vide de son choix à condition de provoquer le retournement d'au moins un pion adverse ; le retournement de pions blancs est possible lorsque l'on pose un pion noir qui permet d'encadrer (en ligne, en colonne ou en diagonale) un certain nombre de pions blancs entre un pion noir et celui qui vient d'être posé : tous les pions blancs encadrés sont alors retournés et deviennent noirs et le principe est évidemment, de manière symétrique, le même en échangeant les deux couleurs. Le joueur J_1, commençant la partie, possède les pions noirs et son adversaire J_2 les pions blancs. Si l'un des deux joueurs ne peut pas poser de pion il passe son tour. Le jeu s'arrête quand les deux joueurs ne peuvent plus poser de pion (que la grille soit pleine ou non) ; on compte alors le nombre de pions et le joueur ayant le plus grand nombre de pions de sa couleur sur la grille a gagné. En cas d'égalité, le vainqueur est le dernier à avoir posé un pion.

Un exemple de début de partie :



On modélise simplement le jeu à l'aide d'une liste bidimensionnelle nommée *grille* de taille 8×8. Chaque pion noir (du joueur J_1) est noté 1 et chaque pion blanc (du joueur J_2) est noté 2 ; on utilise la numérotation python : les lignes et les colonnes sont numérotées de 0 à 7.

Question 1 : Commencer par télécharger sur le site **maths-laf.fr** le fichier **TP23ini.py** nécessaire pour ce TP et prendre connaissance des fonctions déjà écrites à partir de la ligne 121. Le fichier contient une interface graphique entièrement écrite dans la classe **Othello** et il n'est pas nécessaire de connaitre la programmation de cet objet pour avancer dans le TP ; la plus grande partie de ce que l'on ajoutera se fera dans le code hors de la classe (le code de la classe est compris entre les lignes 4 et 119). Après avoir analysé les trois fonctions écrites et particulièrement la fonction **coups**(*grille*, *couleur*) qui reçoit une grille et un entier *couleur* valant 1 ou 2 et retourne la liste des coups possibles avec tous les retournements associés, compléter la fonction nommée **fini**(*grille*) qui retourne le booléen **True** si aucun des deux joueurs ne peut jouer et **False** sinon.

Question 2 : Compléter la fonction nommée **jeuStupide**$(grille, couleur)$ en utilisant la fonction **coups**$(grille, couleur)$ en faisant choisir à l'I.A. l'un des coups possibles (par exemple, le premier) sous la forme $[i, j, r]$ où i et j sont les coordonnées du jeton à poser et r la liste des retournements associés s'il est possible de jouer et **None** sinon.

Question 3 : Compléter la fonction nommée **scoreNoir**$(grille)$ qui reçoit une grille et retourne le score sous la forme d'un entier relatif égal au nombre de jetons noirs moins le nombre de jetons blancs. Si le score est strictement positif, ce sont donc les noirs qui gagnent et s'il est strictement négatif, ce sont les blancs qui gagnent. *Vous n'avez pas à vous préoccuper de savoir quelle est la couleur du joueur et de l'I.A. car l'interface grahique s'en occupe déjà.*
Tester sur une partie contre l'« I.A. stupide ».

Question 4 : Écrire une fonction nommée **jouer**$(grille, couleur, coup)$ recevant l'argument *grille* correspondant à une position de jeu, l'argument $couleur \in [\![1, 2]\!]$ et une liste *coup* de la forme $[i, j, r]$ où i et j sont les coordonnées du jeton à poser et r la liste des retournements associés qui modifie la grille en posant un jeton de la couleur *couleur* en (i, j) et en retournant tous les jetons de la liste r. Écrire ensuite une fonction nommée **dejouer**$(grille, couleur, coup)$ qui annule toutes les modifications sur la grille de **jouer**$(grille, couleur, coup)$.

Question 5 : Écrire une fonction nommée **minimax**$(grille, couleur, prof)$ recevant l'argument *grille* correspondant à une position de jeu, l'argument $couleur \in [\![1, 2]\!]$ et l'entier $prof$ qui retourne un couple $(v, coup)$ où v est la valeur estimée par l'algorithme et *coup* est le meilleur coup permettant d'obtenir l'évaluation en question ;
si $prof \leqslant 0$, minimax se contente d'estimer le score et sinon il estime à l'aide d'appels récursifs le maximum (ou le minimum) des positions accessibles à l'aide de l'exploration du graphe biparti partiel de profondeur $prof$.

Question 6 : Écrire une fonction nommée **jeuIA**$(grille, couleur)$ recevant l'argument *grille* correspondant à une position de jeu et l'argument $couleur \in [\![1, 2]\!]$ et retournant le numéro de colonne du coup considéré optimal par la fonction **minimax**.
Pour tester la fonction, on n'oubliera pas de modifier la ligne 72 *du programme où l'on remplacera* **jeuStupide** *par* **jeuIA** *!*

Question 7 : [*Pour les plus rapides*] Si vous êtes bon joueur à Othello, nul doute que l'algorithme minimax écrit précédemment ne vous battra pas... Réfléchir au moyen d'améliorer l'estimation : le score ne suffit pas, il semble nécessaire de privilégier les positions sur le bord de la grille et même encore plus les 4 coins.

BIBLIOGRAPHIE :
- https ://www.ffothello.org/othello/regles-du-jeu/
- https ://fr.wikipedia.org/wiki/Othello_(jeu)