Programmation Impérative 2 **Licence informatique 2A**

Projet - 2015-2016 - Reversi 9 mai 2016

1 Descriptif

Le but du projet est de réaliser un programme en langage C implémentant le jeu Réversi qui est une variante du jeu d'Othello. Il s'agit d'un jeu à 2 joueurs, selon la page de la fédération française du jeu d'Othello, les règles sont définies comme suit:
Réversi, variante du jeu Othello, est un jeu de stratégie à deux joueurs : Noir et Blanc. Il se joue sur un plateau unicolore de 64 cases, 8 sur 8, appelé othellier. Ces joueurs disposent de 64 pions bicolores, noirs d'un côté et blancs de l'autre. Par commodité, chaque joueur a devant lui 32 pions, il ne peut jouer au maximum que ces 32 pions. Un pion est noir si sa face noire est visible et blanc si sa face blanche est sur le dessus. Les noirs commencent la partie.

Au début de la partie, les joueurs posent les 4 premiers pions sur les cases d4, d5, e4, e5 à tour de rôle (au choix parmi ces 4 cases). A son tour de jeu, le joueur doit poser un pion de sa couleur sur une case vide de l'othellier, adjacente à un pion adverse. Il doit également, en posant son pion, encadrer un ou plusieurs pions adverses entre le pion qu'il pose et un pion à sa couleur, déjà placé sur l'othellier. Il retourne alors de sa couleur le ou les pions qu'il vient d'encadrer. Les pions ne sont ni retirés de l'othellier, ni déplacés d'une case à l'autre. Un joueur a la possibilité de passer son tour (cette fonctionnalité pourra ne pas être offerte dans le programme proposé).

Le but du jeu est d'avoir plus de pions de sa couleur que l'adversaire à la fin de la partie. Celle-ci s'achève lorsque aucun des deux joueurs ne peut plus jouer de coup légal. Cela intervient généralement lorsque les 64 cases sont occupées, ou lorsque tous les pions sont sont de la même couleur après un retournement ou lorsqu'il n'est plus possible de retourner un pion.. Vous pourrez trouver plus d'information sur les règles, notamment la position de départ, les coups et la fin de partie auprès du lien suivant :
`http://www.ffothello.org/jeu/regles.php`

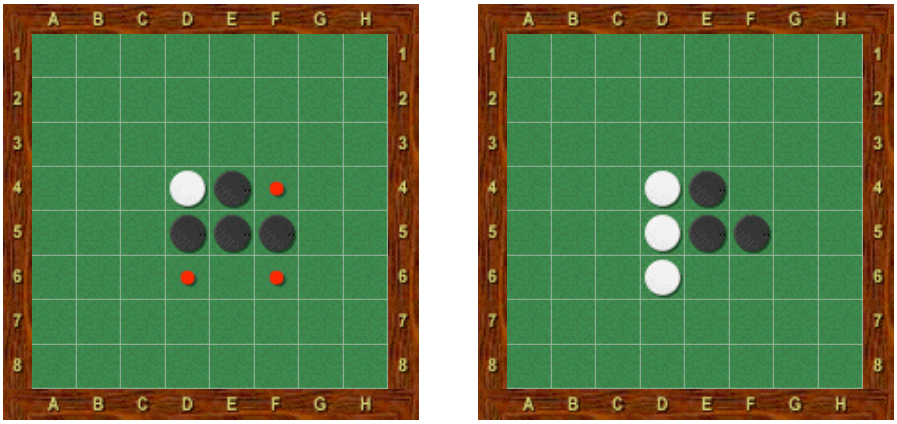

Figure 1: Illustration d'un coup lors d'une partie d'Othello/Réversi. Nous considérons la figure de gauche et c'est maintenant à Blanc de jouer. Il a trois coups possibles. En effet, il est obligatoire de retourner au moins un pion adverse à chaque coup. Blanc peut donc jouer en f4, f6 ou en d6.

2 Travail demandé

Vous devez programmer une version où un joueur joue contre l'ordinateur.

Pour réaliser une interface de jeu, on pourra se limiter à un affichage des plateaux successifs obtenus après

les coups de chaque joueur sur le terminal. Lorsque c'est au joueur humain de jouer, le programme lui demande de rentrer la colonne et la ligne où il souhaite jouer son coup (une vérification de la validité du coup devra être mise en œuvre), le plateau correspondant au coup joué sera alors affiché. Lorsque c'est à l'ordinateur, il affiche la ligne et colonne correspondant à son coup joué, le plateau associé est ensuite affiché. On pourra utiliser la convention suivante dans l'affichage: `.` (un point) représente une case vide, `B` représente un jeton du joueur humain Blanc, `N` un jeton de l'ordinateur Noir (on pourra proposer d'inverser Blanc et Noir si l'on souhaite). Exemple d'affichage d'un plateau :

```
   ----------------
1 | . . N N . . . .|
2 | . . . N B N . .|
3 | . B N N B B B N|
4 | N N N B B N B B|
5 | N N N B N N N B|
6 | N N N B B . B N|
7 | . . . N N . . .|
8 | . . . . N . . .|
   ----------------
    A B C D E F G H
```

Pour vous aider à réaliser votre programme, vous pourrez implanter les fonctionnalités suivantes :

- `position_gagnante`: qui renvoie 1 s'il y a une position gagnante dans le jeu, 0 sinon. On pourra utiliser un tableau à 2 dimensions pour représenter le plateau.
- `coup_ordinateur`: qui renvoie le coup que doit jouer l'ordinateur.
- `coup_valide`: qui vérifie si un coup joué par le joueur humain est valide.
- `evaluation_plateau`: qui renvoie la qualité d'une position pour un joueur, cette fonction est utilisée pour calculer la qualité de la position de l'ordinateur dans l'intelligence artificielle.
- `creer_arbre_position`: cette fonction crée un arbre de jeu utilisé par l'intelligence artificielle, votre implantation devra utiliser des pointeurs pour modéliser les fils.

Le programme laissera la possibilité au joueur humain de commencer la partie ou de laisser l'ordinateur commencer.

Vous pourrez développer votre programme par niveaux, selon la description suivante :

- Niveau 0: le programme permet de jouer contre l'ordinateur, mais l'ordinateur génère des coups aléatoirement sans intelligence artificielle.
- Niveau 1: l'intelligence artificielle est implantée et elle utilise un arbre de jeu de profondeur 2 (quand c'est au joueur de jouer, on regarde un coup en avance pour l'ordinateur, suivi d'un autre coup pour le joueur), et la fonction d'évaluation ne prend en compte qu'un nombre limité d'information sur les pions.
- Niveau 2: l'intelligence artificielle peut utiliser un arbre de profondeur plus importante qui pourra être indiqué sous la forme d'un paramètre du programme grâce à l'option `-prof` (`-prof 5` indique l'utilisation d'un arbre de profondeur 5). La fonction d'évaluation prend en compte les ensembles de 3 pions consécutifs et les ensembles de 2 pions consécutifs.
- Niveau 3: L'implémentation de l'intelligence artificielle est améliorée au niveau du parcours de l'arbre: les branches menant à des solutions potentiellement inintéressantes ne sont pas explorées (algorithme alpha-beta). D'autres fonctions d'évaluation peuvent être programmées (pondérer les ensembles de pions, pondérer des régions, etc.).
- Niveau 4: la gestion mémoire de l'arbre de jeu est optimisée: on n'utilise qu'un seul plateau pour modéliser les coups possibles et l'ensemble de l'arbre n'est pas stocké en entier en mémoire.

Vous pouvez rajouter au programme une option `-niveau` permettant à l'utilisateur d'indiquer le niveau souhaité.

Vous pouvez aussi ajouter des améliorations ou des fonctionnalités à votre programme en proposant des initiatives personnelles (autres fonctions d'évaluations, faire jouer 2 programmes l'un contre l'autre, ajouter une interface graphique, autres idées, ...).

Information importante : le code de votre programme devra être compatible avec la norme C `ANSI` (il suffit de rajouter l'option `-ansi` à la compilation.

3 Intelligence artificielle

L'intelligence artificielle est implantée à l'aide d'un arbre de jeu dont la définition est la suivante. La racine représente la situation de jeu actuelle et on considère que c'est à l'ordinateur de jouer. Les fils de la racine correspondent à tous les coups possibles à partir de la configuration actuelle (les coups possibles dépendent de la configuration). Ensuite, chacun des fils représente le joueur adverse (humain) et on calcule les différents coups possibles, ce qui donne un nouveau niveau de l'arbre. Ce niveau est encore développé pour modéliser l'ensemble des deuxièmes coups possibles de l'ordinateur. La profondeur des feuilles de l'arbre correspond alors au nombre de coups à l'avance que l'on peut considérer, on évalue les plateaux correspondant à chacune des feuilles de l'arbre à l'aide de la fonction évaluation. On effectue l'évaluation de bas en haut. Lorsque l'on est sur un nœud correspondant à l'ordinateur, on choisit le coup correspondant au nœud fils qui maximise la fonction d'évaluation et on renvoie la valeur de la fonction au niveau supérieur. Lorsque l'on s'intéresse aux solutions du joueur humain adverse, on prend le coup qui minimise la fonction d'évaluation. On remonte ainsi jusqu'à la racine et on choisit le prochain coup qui maximise la valeur de la fonction d'évaluation dans les coups à venir.

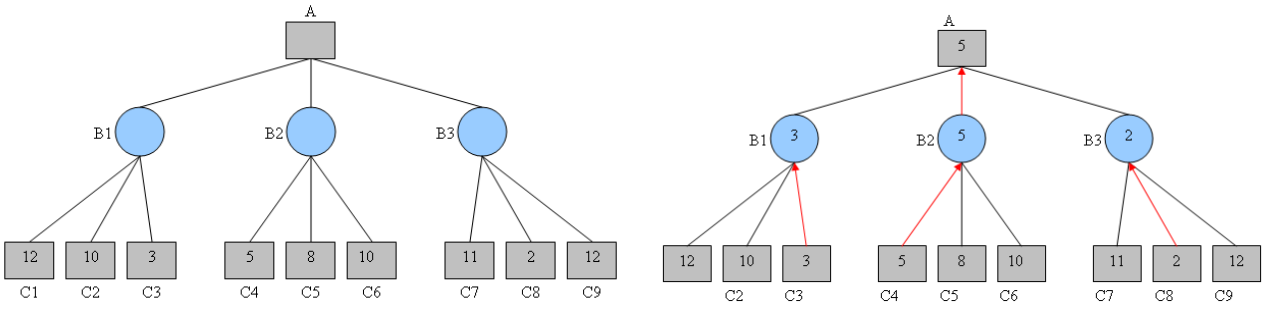

Figure 2: Illustration d'un arbre de jeu pour l'ordinateur: les nœuds gris représente un coup de l'ordinateur et les nœuds bleus un coup du joueur humain. Pour simplifier on ne considère que 3 coups possibles par niveau. A gauche, on observe la définition de l'arbre de jeu, à droite les coups et valeurs remontées à chaque niveau. Le niveau C correspond aux évaluations de du plateau de jeu après 2 coups de l'ordinateur et un 1 coup du joueur humain. Chaque Ci contient la valeur de la fonction d'évaluation pour le meilleur coup possible à ce niveau pour l'ordinateur. Le niveau B correspond au coup du joueur humain, on considère qu'il choisit la fonction la plus favorable pour lui, donc la moins favorable pour l'ordinateur, ce qui correspond au minimum des 3 cas possibles. Par exemple, en $B1$ on considère que le joueur humain va jouer le coup menant à $C3$ ayant pour valeur 3; en $B2$ ce sera $C4$ avec 5 et en $B3$ on remonte $C8$ avec pour valeur 2. Pour le niveau A, l'ordinateur choisit le coup le plus favorable pour lui parmi: 3 pour $B1$, 5 pour $B2$ et 2 pour $B3$. Le coup choisi sera donc celui menant à $B2$, ce qui correspond au maximum des 3 valeurs de la fonction d'évaluation remontées pour le niveau B.

Cet algorithme porte le nom de `minimax` puisque l'on prend à chaque niveau soit le maximum, soit le minimum de la fonction d'évaluation en fonction du joueur concerné. Dans l'implantation, à chaque nœud est associé un plateau de jeu et un coup correspondant.

La définition de la fonction d'évaluation est donc très importante pour améliorer la qualité de l'intelligence artificielle. Une solution est de compter le nombre de pions de chaque couleur ou de compteur ceux d'un joueur et de retrancher les pions du joueur adverse (attention à ne pas compter la même chose plusieurs fois). On peut

également pondérer l'importance des pions par rapport à la position sur le plateau de jeu. D'autres solutions sont bien sûr possibles.

Pour avoir quelques idées sur la fonction d'évaluation, vous pouvez visionner la vidéo suivante intitulée *L'odyssée d'Othello* : `https://vimeo.com/86877605`

4 Rendu

Le travail sera à déposer sur la page du cours sur la plateforme Claroline du cours pour le **9 mai 2016 minuit**, sous la forme d'une archive `.zip` ou `.tar.gz` organisée. Elle devra créer un dossier contenant votre nom et dans ce dossier vous mettrez le code source du programme et un fichier `README` ou `LISEZMOI` indiquant comment installer le programme, la liste des fichiers contenu dans l'archive et comment utiliser le programme. Vous joindrez également un rapport en LaTeXd'une dizaine de pages indiquant les fonctionnalités et solutions que vous avez retenues et les difficultés rencontrées ainsi que les pistes possibles pour améliorer le programme. Vous pouvez mettre quelques bout de code ou fonctions pertinentes, mais le rapport ne doit en aucun cas constitué d'une suite de lignes de code. Vous veillerez à mettre votre nom dans le nom de l'archive également.

Le travail pourra être effectué seul ou par binôme. Si le travail est effectué en binôme vous devrez préciser explicitement (dans un document annexe) comment le travail a été réparti, qui a écrit quoi, en quoi le fait de travailler en binôme vous a permis de faire un meilleur projet (on attend bien sûr plus de la part d'un projet en binôme que d'un projet rendu seul).

La qualité de la présentation des fichiers sources (clarté du code, commentaires, ...) ainsi que la modularité du programme seront pris en compte pour l'évaluation. Si vous avez utilisé des sources extérieures, il faut obligatoirement l'indiquer dans votre rapport (et dans vos commentaires s'il s'agit de parties de code).

Une soutenance avec Christophe Moulin sera programmée avant le **13 mai 2016**, le planning sera donné ultérieurement. Contact: `christophe.moulin@univ-st-etienne.fr`