# TP de « RepCo » : réalisation d'un jeu à deux joueurs à somme nulle et information complète

Page du cours : `http://homepages.loria.fr/JLieber/cours/repco/`

Ce sujet de TP est prévu pour des binômes et une durée de travail présentiel de 10 heures, par séances de 2 heures. Il fait partie du cours « RepCo » de la troisième année de la licence informatique de l'Université de Lorraine, à la FST (Nancy).

L'objectif de ce TP est la conception, réalisation et implantation d'un jeu à deux joueurs à somme nulle et information complète.

## 1 Avant le premier TP

Avant le premier TP, il faut se familiariser avec le jeu à deux joueurs que vous allez implanter. Ce jeu est :

— Soit un jeu de votre choix mais validé au préalable par votre intervenant en cours ou en TD (au moins une semaine avant le début des TP) : il faut que ce soit un jeu à deux joueurs à somme nulle et information complète et il faut également qu'il puisse être raisonnablement jouable (ce qui exclut des jeux comme les échecs et le go dont la complexité calculatoire est trop importante) ;

— Soit le jeu de Reversi [1], dont une description se trouve à la page `https://fr.wikipedia.org/wiki/Othello_(jeu)`. Vous paramétrerez le jeu pour que les dimensions du plateau puissent être changées facilement (par défaut : $8 \times 8$).

Se familiariser avec un jeu signifie *a minima* en comprendre les règles.

Par ailleurs, vous aurez soin de lire cet énoncé de TP en entier au préalable, pour comprendre les objectifs et faire les bons choix d'implantation.

Enfin, comme un rapport doit être rendu après le TP, il est conseillé de le constituer au fur et à mesure.

## 2 Contraintes dans la programmation

Il va sans dire que vous devez programmer de façon propre et documentée, en usant au mieux vos compétences en programmation, en particulier celles de la programmation orientée objets (principe d'encapsulation, polymorphisme, spécialisation des classes). Le langage de programmation préconisé est Java (dans la version que vous avez eue en cours). Si vous souhaitez utiliser un autre langage de programmation, demandez à votre intervenant en TP s'il valide votre choix.

Une contrainte dans la programmation qui est importante est la suivante : il importe que le jeu puisse jouer contre lui-même. Cela permettra, en particulier, de comparer deux fonctions d'évaluation $\texttt{éval}_0$.

Pour l'affichage, vous pouvez vous contenter d'un affichage textuel : la priorité de ce TP n'est pas dans l'affichage.

Le découpage en séance de TP qui suit est indicatif : si vous êtes en avance, n'hésitez pas à commencer la séance suivante, si vous êtes en retard, il faudra rattraper ce retard.

## 3 TP 1 : représentation des états de jeu et des transitions entre états

Représentez une classe abstraite pour un joueur et une sous-classe directe de cette classe pour le jeu que vous avez choisi.

Représentez une classe abstraite pour un état de jeu et une sous-classe directe pour le jeu choisi. Il est rappelé qu'un état de jeu est non seulement caractérisé par la position des pièces du jeu mais également par le joueur qui doit jouer à partir de cet état. Il est demandé que cette classe représentant un état de jeu pour le jeu chosi ait les méthodes

- ☐ pour accéder en lecture et en écriture à un état,
- ☐ pour tester l'égalité de deux états,

1. Ou « Othello », mais nous préférerons le terme Reversi, qui est libre de droit.

- ☐ pour faire un affichage (simple) et
- ☐ pour générer les successeurs d'un état de jeu (l'ensemble des états de jeu qu'on peut générer à partir de l'état courant).

**Remarque 1** Comme vous travaillez en binôme, vous pouvez faire une implantation « simple » de ces classes, avec les accès en lecture et en écriture bien spécifiés. Dans un deuxième temps, une meilleure implantation (prenant notamment moins de place mémoire et plus efficace en terme de manipulations) pourra être réalisée par un élément du binôme pendant que l'autre continuera le TP.

# 4 TP 2 : Mise en place du jeu à deux joueurs humains

Le but de cette séance est simple : faire en sorte que deux joueurs humains, prenant tour à tour le clavier, puisse jouer. Une fois cela fait, si vous n'êtes pas familier avec le jeu, faites quelques parties (pas plus d'une demi-heure) pour vous y familiariser et anticiper sur les problèmes de choix de la fonction $\texttt{éval}_0$.

**Remarque 2** Cette étape du TP est en principe très courte : il est plus que conseillé d'anticiper sur les étapes suivantes.

# 5 TP 3 : Implantation de l'algorithme minimax avec élagage $\alpha\beta$

Implantez et testez l'algorithme minimax pour le jeu choisi. Pour cela, vous devrez définir une fonction $\texttt{éval}_0$ simple (elle sera à améliorer dans les TP suivants).

Vous ferez d'abord une version sans l'élagage $\alpha\beta$ puis avec. Vous comparerez sur des exécutions vos temps de calcul (à indiquer dans votre rapport).

Vous testerez le programme en jouant contre lui.

# 6 TP 4 : Choix de la fonction $\texttt{éval}_0$

Implantez plusieurs fonctions $\texttt{éval}_0$ (au moins 3), qui ne soient pas clairement équivalentes (par exemple, remplacer une fonction $\texttt{éval}_0$ par une fonction $K \cdot \texttt{éval}_0$ — où $K > 0$ est une constante — donnera les mêmes exécutions du programme).

Écrivez une fonction permettant de comparer deux fonctions $\texttt{éval}_0$ qui donnera :

— `1` si la première est jugée meilleure que la deuxième ;

— `-1` si la deuxième est jugée meilleure que la première ;

— `0` dans les autres cas (pas de préférence détectée).

Pour cela, vous chosirez une profondeur (par exemple, $p = 2$) et vous ferez jouer le jeu contre lui-même avec les deux fonctions $\texttt{éval}_0$, une fois en faisant commencer un joueur, l'autre fois en faisant commencer l'autre joueur.

Indiquez dans le rapport ce que donne cette fonction sur les fonctions $\texttt{éval}_0$ implantées. S'il y a des couples de fonctions qu'on ne peut pas distinguer (valeur `0` retournée par la fonction de comparaison), essayez de changer la valeur de la profondeur pour essayer de les distinguer.

# 7 TP 5 : Amélioration de la fonction $\texttt{éval}_0$

Le but de cette séance est l'amélioration automatique de la fonction $\texttt{éval}_0$.

Pour cela, deux méthodes ont été présentées en cours (à la section 4.4). Choisissez une des deux méthodes et appliquez-la pour trouver une fonction $\texttt{éval}_0$ (en principe meilleure que celles définies au TP précédent).

Si vous voyez une troisième méthode adaptée au jeu choisi et que vous avez envie d'essayer, parlez-en avec votre intervenant en TP, pour qu'il la valide.

# 8 Après le dernier TP

Vous aurez une semaine pour envoyer un compte-rendu de votre TP à l'intervenant de votre TP, décrivant le travail effectué, l'organisation de votre travail (qui, du binôme, à fait quoi), les difficultés rencontrées et des perspectives de ce travail (qu'auriez-vous fait avec du temps supplémentaire). Vous joindrez à ce travail une archive contenant votre code (source et exécutable), avec un fichier explicatif (configuration, instruction d'exécution, etc.).