Mini-projet jeu: Oust

Laurent Fousse (`Laurent.Fousse@imag.fr`)

8 mars 2010

1 But du mini-projet

Le mini-projet consiste en la mise en œuvre en `C++` d'un jeu à deux joueurs (il s'agit de Oust dont les règles sont données plus loin) et à l'évaluation de différentes stratégies d'optimisation. Vous produirez deux réalisations concrètes au cours de ce mini-projet : un robot capable de jouer, ainsi qu'un rapport décrivant et justifiant vos choix techniques.

Quelques contraintes à respecter :
- Le langage de programmation est le `C++`.
- Le choix des structures de données est libre, mais celles de type liste ou vecteur devront respecter l'interface de la librairie standard (`STL`).

2 Règles du jeu *Oust*

Le jeu *Oust* se joue avec un plateau rectangulaire de n lignes par m colonnes type plateau de Go et des pierres noires et blanches que l'on joue sur les intersections des lignes. Dans notre cas n et m seront des paramètres. Le but du jeu est d'éliminer toutes les pierres adverses du plateau, et ce après que chaque joueur ait joué au moins une fois. On définit par ailleurs la notion de voisinage : deux pierres sont voisines si elles sont côte à côte sur la même ligne ou sur la même colonne (il n'y a pas de lien en diagonal).

On dit que deux pierres font partie du même *groupe* de pierres si elles ont la même couleur et qu'on peut passer de l'une à l'autre par des relations de voisinages toujours entre pierres de la même couleur. Les groupes de pierres sont donc connexes pour la relation de voisinage définie.

Voici les règles du jeu :

1. Le joueur ayant les pierres noires commence.
2. Tant que c'est son tour, et s'il peut jouer, un joueur doit effectuer un coup valide.
3. Il existe deux types de coups valides : les coups capturants, et les coups non capturants.
4. Un coup non-capturant se fait en plaçant une pierre sur une intersection libre du plateau, de telle sorte que la pierre posée ne soit pas voisine d'une pierre de même couleur. Un coup non-capturant marque la fin du tour d'un joueur.
5. Un coup capturant se fait en plaçant sa pierre au voisinage d'au moins une autre pierre de sa propre couleur. Un coup capturant est valide si le groupe dont fait partie la pierre posée est voisin d'au moins un groupe de pierres adverses, et si tous les groupes de pierres adverses voisins sont de taille strictement inférieure au groupe de la pierre nouvellement posée. On retire alors tous ces groupes de pierres adverses, et c'est encore au joueur qui vient de capturer de jouer.
6. Si après un coup capturant, il n'y a plus de pierre adverse en jeu, le jeu s'arrête sur la victoire du joueur qui vient de capturer.

3 Entrées-sorties de votre programme

Votre programme recevra en premier paramètre le temps en seconde de réflexion dont il dispose avant de rendre son coup. Ce paramètre sera un entier. Sur l'entrée standard, votre programme recevra une représentation de l'état du jeu dans lequel il doit jouer, ainsi que sa couleur, selon le format suivant :

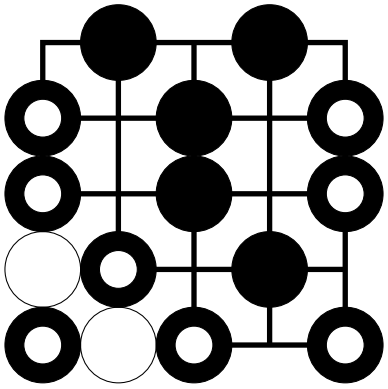


Fig. 1 – Coups non-capturants possibles pour noirs.

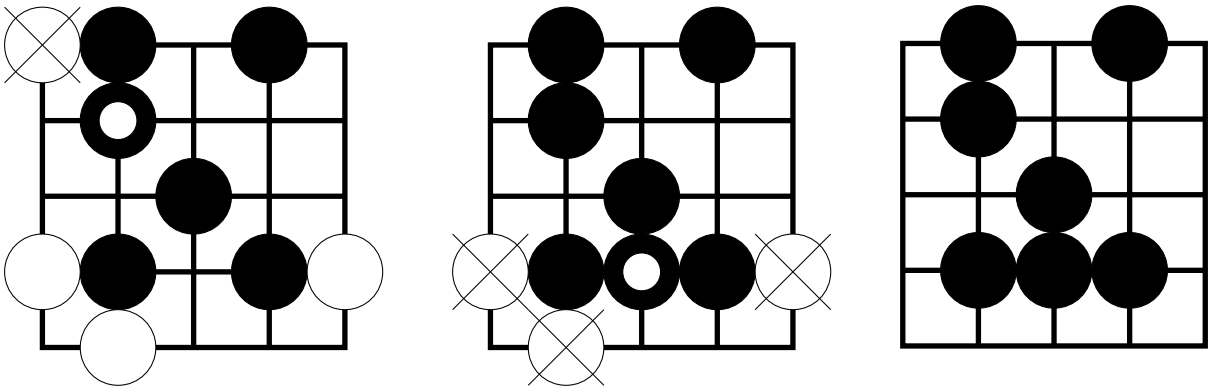


Fig. 2 – Séquence de coups capturants pour noir. À l'issue de cette séquence, noir a gagné.

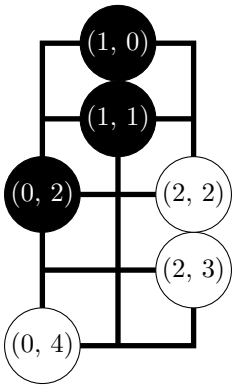
1. Sur la première ligne, deux entiers séparés par une espace et représentant respectivement le nombre de colonnes et de lignes du jeu. En particulier le plateau de jeu pourra ne pas être carré ! On garantit que chacune des dimensions sera inférieure à 30 lors du tournoi.
2. L'état du plateau est ensuite donné, ligne par ligne. Chaque ligne comportera le même nombre d'entiers (et égal au nombre de colonnes) séparés par des espaces. Chaque entier représente une case du plateau de jeu, et sa valeur sera :
   – 0 si la case est vide ;
   – 1 si la case est occupée par une pierre noire ;
   – 2 si la case est occupée par une pierre blanche.
3. À la suite du plateau sera ensuite indiqué par un entier seul sur une ligne la couleur pour laquelle votre programme doit jouer (1 pour noir et 2 pour blanc).

Votre programme devra répondre sur sa sortie standard les coordonnées où il pose sa pierre (indice de colonne, suivi de l'indice de ligne) en numérotant les lignes et les colonnes à partir de 0. Des coordonnées (positives) qui débordent du plateau sont interprétées comme passer. Attention, passer n'est un mouvement valide que lorsque aucun coup n'est possible pour vous (extrêmement rare).

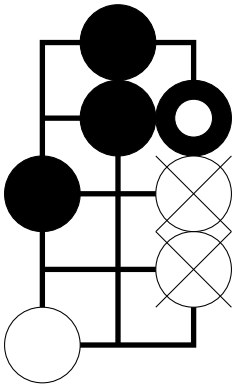
Par exemple l'entrée suivante :

```
3 5
0 1 0
0 1 0
1 0 2
0 0 2
2 0 0
1
```

correspond à ce plateau de jeu faisant apparaître les coordonnées des pierres déjà posées :



et c'est à noir de jouer. Une sortie valide de votre programme pourra alors être `2 1` ce qui correspond au coup suivant :



À noter que vous n'avez pas à tenir compte du fait que votre coup est capturant et que c'est encore à vous de jouer : vous ne jouez à chaque fois qu'un seul coup, et c'est au serveur lors du tournoi qu'il reviendra de détecter si votre coup est capturant, de mettre à jour le plateau et de demander au bon joueur de jouer le coup suivant.

4 Tournoi

Le bon respect de l'interface par votre programme permettra d'organiser un tournoi entre les robots des différents groupes. Par soucis d'équité, on fera jouer deux matchs (« aller » et « retour ») avec des couleurs différentes pour chaque paire d'adversaires. Le tournoi sera au format *round-robin*, ce qui signifie simplement que chacun rencontrera chacun des autres participants (en marquant 1 point par victoire).

On appliquera les règles suivantes lors du tournoi :

1. Le temps de réflexion par coup sera de 4 secondes.
2. Un coup invalide entraîne automatiquement la défaite du joueur fautif.
3. Un retard de réponse de plus de 4 secondes entraîne la défaite par abandon du joueur fautif.

Le temps de réflexion sera aussi conservé, et il sera tenu compte de retards trop fréquents dans la notation du tournoi.

Les matchs du tournoi se dérouleront par le réseau, en utilisant une petite infrastructure client-serveur. Vous n'aurez pas à vous soucier de la gestion des communications et un script de jeu vous sera fourni.

5 Rapport

Les comptes-rendus (environ 10 pages) devront être remis à la fin du tournoi ou envoyés par courriel. Ils doivent contenir :

- une brève description des structures de données choisies ;
- une analyse synthétique des expérimentations ainsi qu'un comparatif des optimisations employées ;
- une ou deux situations précises seront détaillées et analysées ;
- une annexe (préparée pendant le tournoi) présentera les résultats et vos commentaires des matches effectués.

6 Barème indicatif

Programmation (structures de données, α-β, gestion des entrées-sorties, de l'évaluation, de la profondeur, du temps, etc.)	7 points
Puissance du programme (profondeur maximale moyenne pour un temps donné, classement au tournoi, etc.)	5 points
Bonus pour optimisations (table de hachage ou parallélisation ou autre)	3 points
Qualité du rapport	5 points

Votre note pour le mini-projet ne sera pas une note de groupe. Les sources devront être séparées en plusieurs fichiers (une dizaine maximum) comportant chacun *un seul nom*, celui du responsable de ce fichier. Les 7 points de programmation seront attribués par personne au vu de ces fichiers.