TP 5 Algorithmique Avancée : Puissance 4

CERI - Licence 2 Informatique

Semestre 3

Nous nous proposons d'écrire un programme capable de jouer "intelligemment" au jeu puissance 4 [1] avec un adversaire humain. L'algorithme sera basé sur la méthode Min-Max vue en cours. Le plateau du jeu sera représenté par une matrice T de 6 lignes et 7 colonnes comme suit.

$T =$

i/j	0	1	2	3	4	5	6
5							
4							
3							
2							
1							
0							

Chaque joueur (ordinateur et humain) joue à tour de rôle. Un tirage au sort désigne celui qui commence. Les pions sont introduits en amont des colonnes du plateau et tombent donc verticalement. Les colonnes seront indicées $j = 0, 1, 2, 3, 4, 5, 6$ et les lignes $i = 0, 1, 2, 3, 4, 5$. La ligne 0 étant celle la plus basse, et la colonne 0 celle la plus à gauche. Les pions joués par l'ordinateur seront représentés par le chiffre 1, et ceux du joueur humain par -1. Un case vide de T contiendra 0. Cette configuration indique par exemple une victoire de l'ordinateur.

	0	1	2	3	4	5	6
5							
4							
3							
2	-1						
1	-1	-1					
0	1	1	1	1			

Chaque colonne j de T peut être vue comme une pile dans la laquelle tombe les pions. On notera $Col[j]$ l'indice de la ligne du dernier pion tombé dans la colonne j. Dans l'exemple ci-dessus, $Col[0] = 2$, $Col[1] = 1$, $Col[2] = 0$, $Col[3] = 0$. Pour les colonnes vides $4, 5, 6$, le contenu sera -1 : $Col[4] = Col[5] = Col[6] = -1$.

La réalisation d'un algorithme de type min-max pour ce jeu passe par le codage d'une classe nommée "Puissance4" dont les attributs et méthodes sont expliqués ci-dessous.

1. voir wiki pour les régles du jeu

1 Attributs

La classe Puissance4 contiendra (au moins) les trois attributs suivants :

- int T[6][7] ;
- int Col[7] ;
- int hmax ;

T et Col ont déja été expliqués en introduction. “hmax” est la profondeur à partir de laquelle l'état du jeu est évalué par une fonction d'évaluation à concevoir.

2 Méthodes

Les méthodes à développer sont les suivantes.

• “Puissance4(int h)” est le constructeur de la classe. Il permettra d'initialiser T à 0, Col à -1, et hmax à h.

• “void affichage()” est la méthode d'affichage du contenu de T.

• “jouer(int j,int joueur)” permet à “joueur” (1 : ordinateur, -1 : humain) de jouer sur la colonne j du plateau.

• “dejouer(int j)” retire le dernier pion joué dans la colonne j.

• “bool coupgagnant(joueur,bestMove)” renvoie vraie s'il est possible pour ”joueur” (1 : ordinateur, -1 :humain) de gagner la partie en un seul coup, tenant compte de l'état courant du plateau.

• “int JeuOrdi(int & bestMove, int niveau)” est appelé quand c'est au tour de l'ordinateur de jouer. Elle contient les instructions permettant d'évaluer les coups de l'ordinateur. Elle affecte à “bestMove” le “meilleur” coup de l'ordinateur, et renvoie la valeur de l'évaluation de ce coup. La méthode ne réalise pas le coup. La réalisation effective est faite dans le fichier principal (main.cpp). Selon les principes de l'algorithme min-max, le meilleur coup (bestMove) peut être obtenu en mettant en oeuvre le pseudo-code ci-dessous.

On y désigne par NUL la valeur 0 (match nul), par VORDI la valeur +1000 (évaluation arbitrairement grande représentant la victoire de l'ordinateur) et VHUMAIN la valeur -1000 (évaluation opposée de VORDI représentant la victoire de l'humain). L'argument “niveau” est la profondeur de l'arbre du min-max.

Algorithme : JeuOrdi(bestMove, niveau)

```
/* arg est un argument additionel passé à l'appel récursif à JeuHumain.    */
arg : valeur entière;
joueur = 1;
/* Si le plateau est rempli au moment où l'ordinateur doit jouer alors il
   s'agit d'un match nul.                                                   */
si plein() alors
    retourner(NUL);
fin
/* Si l'ordinateur a un coup gagnant, celui-ci est rangé dans bestMove, et
   la partie est terminée.                                                  */
si coupgagnant(joueur,bestMove) alors
    retourner(VORDI);
fin
/* Si ''niveau'' a atteint le maximum hmax alors évaluer l'état du jeu en
   faisant la somme des évaluations des coups réalisables par l'ordinateur.
   */
si niveau == hmax alors
    S = 0;
    pour j = 0, 1..., 6 faire
        si not plein(j) alors
            /* jouer dans la colonne j                                      */
            jouer(j,joueur);
            S = S + evaluation(joueur,j);
            /* Remettre la colonne j dans son état initial                  */
            dejouer(j);
        fin
    fin
    retourner(S);
fin
val = VHUMAIN;
/* Sinon pour chaque colonne j où il est possible de jouer,...              */
pour j = 0, 1..., 6 faire
    si not plein(j) alors
        jouer(j,joueur);
        /* évaluer les réponses possibles de l'adversaire si l'ordinateur joue
           dans la colonne j.                                               */
        res = JeuHumain(arg,niveau+1);
        dejouer(j);
        /* Stocker dans ''val'' la valeur maximale ''res'', correspondant au
           coup de l'ordinateur le plus favorable pour lui                  */
        si res > val alors
            val = res;
            bestMove = j;
        fin
    fin
fin
retourner(val);
```

• “int evaluation(int joueur, int j)” renvoie une valeur numérique représentant l'évaluation de la qualité d'un coup joué par “joueur” (1 : ordinateur, -1 : humain) sur la colonne j du plateau.

• “int JeuHumain(int & bestMove, int niveau)” est le symétrique de JeuOrdi. Il suffit, dans le pseudo-code précédent, d'interchanger VHUMAIN et VORDI, de remplacer joueur = 1 par joueur = -1, et (res > val) par (res < val).

3 Fonction d'évaluation

Une fonction d'évaluation est une fonction qui, pour un coup donné, renvoie une valeur numérique représentant la qualité de ce coup. Pour “puissance4”, cela revient à determiner, pour chaque colonne jouable, celle considérée comme la plus “optimale” pour remporter la partie. Plus cette valeur est élevée (positive), et plus le coup est censé être à l'avantage de l'ordinateur. Inversement, plus elle est faible (négative) plus le coup est favorable à l'humain.

Une fonction d'évaluation possible vous est proposée ci-dessous. D'autres sont envisageables. Vous pouvez en proposer d'autres, améliorant celle ci-dessous (ou complétement nouvelles), en détaillant lors du rendu, le principe de votre évaluation.

Considérons l'exemple ci-dessus représentant l'état du plateau à un instant donné du jeu. C'est à l'ordinateur de jouer.

			1			
		-1	-1	1	-1	
		1	-1	-1	1	
		1	-1	-1	1	

L'ordinateur devrait se rendre compte que le coup le plus efficace, menant à la victoire est la colonne 6. Dans ce cas “evaluation(1,6)” (i.e joueur 1 joue dans la colonne 6) conduira à une valeur au moins égale à +1000 signifiant la victoire de l'ordinateur. Mais remontons à l'instant où l'état du plateau était :

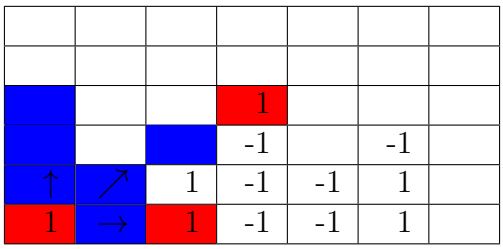
			1			
			-1		-1	
		1	-1	-1	1	
		1	-1	-1	1	

Ici la victoire ne se fait pas un coup. L'ordinateur doit estimer la colonne la plus judicieuse où jouer.

En algorithmique, on appelle “heuristique” une méthode qui, pour un problème, se calcule rapidement mais dont l'optimalité n'est pas garantie. Par exemple, une heuristique d'évaluation

consiste à evaluer pour chaque colonne jouable, les possibilités d'alignement de 4 pions.

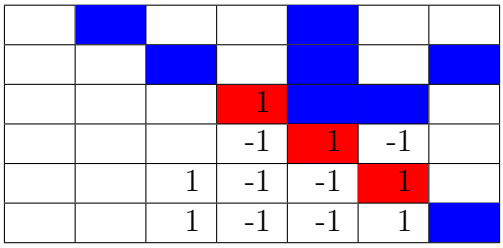
Evaluons par exemple le coup consistant à jouer dans la colonne 0. On s'intéresse, autour du coup joué, aux 4 directions possibles d'alignement de pions : horizontal, vertical et les deux diagonales. Pour chaque direction et chaque sens, si la case est occupée par 1, cela vaudra 5 points (valeur positive arbitraire), si la case est vide cela vaudra 1 point. Dès qu'un pion adverse est atteint le calcul s'arrête.



Ainsi en jouant en colonne 0, l'ordinateur pourrait aligner des pions dans trois directions et 3 sens indiqués par les fléches. Une case rouge indique une occupation par l'ordinateur. Une case bleue est une case vide éventuellement occupable par lui aux coups suivants.

- L'alignement possible vers le haut vaudra : $5 + 1 + 1 + 1 = 8$.
- L'alignement possible vers la droite vaudra : $5 + 1 + 5 = 11$.
- L'alignement possible dans le sens nord-est vaudra : $5 + 1 + 1 + 5 = 12$.

L'évaluation du jeu dans la colonne 0 sera la somme de ces valeurs, soit 31. Regardons maintenant la valeur du jeu en colonne 4 selon ce même procédé de calcul.



- L'alignement possible vers le haut vaudra : $5 + 1 + 1 + 1 = 8$
- L'alignement possible dans le sens nord-ouest vaudra : $5 + 5 + 1 + 1 = 12$
- L'alignement possible dans le sens nord-est vaudra : $5 + 1 + 1 = 7$
- L'alignement possible dans le sens sud-est vaudra : $5 + 5 + 1 = 11$

L'évaluation du jeu en colonne 4 vaut : 38. Celle-ci étant supérieure à la précédente devrait être préférée.

Le calcul exprime la possibilité d'aligner 4 pions dans l'une des 4 directions en fonction des pions déjà présents et des cases vides à proximité. En jouant en colonne 4, cette valeur signifie qu'il serait plus "facile" de gagner en complétant ultérieurement la diagonale. Mais cela ne garantit aucunement la victoire. C'est en cela que le calcul est heuristique et non optimal.

Au passage, examiner les 4 directions et les 8 sens possibles permet aussi de détecter l'alignement de 4 pions et donc la victoire de l'ordinateur ou de l'humain.

Soient direction1(joueur,j), direction2(joueur,j), direction3(joueur,j), direction4(joueur,j), des méthodes renvoyant le calcul du nombre de points cumulés dans les 4 directions. Noter que pour chaque direction les deux sens sont comptabilisés. Aussi, si dans une direction, le constat est fait de l'alignement de 4 pions de "joueur", ces méthodes renvoient +1000 si joueur = 1 (victoire de l'ordinateur) ou -1000 si joueur = -1 (victoire de l'humain). L'évaluation d'un coup de "joueur" dans la colonne j est la somme des valeurs calculées ci-dessus.

Algorithme : evaluation(joueur,j)
retourner(direction1(joueur,j)+direction2(joueur,j)+direction3(joueur,j)+direction4(joueur,j));

4 Main.cpp

Un fichier "main.cpp" permettra de lancer une partie de la façon suivante.

- La partie est représentée par un objet de la classe Puissance4 : $Puissance4\ partie(h)$ où "h" est la profondeur de recherche de l'algorithme min-max. L'argument peut être assimilé à un niveau de difficulté pour l'humain. Car plus la hauteur (h) de l'arbre min-max est grand, et plus l'ordinateur anticipe de coups et devient difficile à battre (si la fonction d'évaluation est bonne).
  .
- Un nombre aléatoire "piece" est choisi dans $\{0, 1\}$ pour savoir qui commence. Si $piece == 0$ alors l'ordinateur commence sinon l'humain.

- Les deux joueurs (ordinateur / humain) jouent à tour de rôle.

- L'ordinateur joue grâce à sa méthode "JeuOrdi(bestMove,niveau)" qui lui renvoie dans "bestMove" le coup à réaliser. Ce coup est ensuite réalisé.

- L'humain joue en indiquant le numéro de colonne dans laquelle il veut jouer.

- Le programme doit détecter la victoire de l'ordinateur, de l'humain ou un match nul.

5 Modalités des rendus

Vous devez rendre un fichier zip à votre nom comprenant les fichiers suivants :

- puissance4.h : contenant la déclaration de la classe Puissance4.h
- puissance4.cpp : contenant l'implémentation des méthodes de la classe.
- main.cpp : indiqué ci-dessus.
- puissance4.exe : l'exécutable issu de votre compilation

- Le TP est réalisable en groupe de 2 maximum (dans le zip sera au nom des deux personnes).
- Une présentation orale (pendant les séances TP) de vos réalisations, auprès de l'enseignant responsable de votre groupe, devra être faite.