1GI

Projet Résolution des problèmes

Le jeu de Puissance 4

Auteurs :
SETTAI Yassine
ASSAFI Issam

Encadrants :
Mme. ADDOU

07 mai 2018

Table des matières

Introduction

1.1 Contexte:

Ce projet s'inscrit dans le cadre du module « résolution de problèmes ». L'objectif est de programmer le jeu de stratégie Puissance 4 (Connect4 en anglais) en langage C/C++, et d'implémenter une intelligence artificielle IA sophistiquée en utilisant nos connaissances acquises pendant ce module.

1.2 Présentation du jeu:

Puissance 4 est un jeu de société à deux joueurs. Entre les deux joueurs se trouve une grille de 6 lignes et 7 colonnes, disposée verticalement. Chaque joueur, lors de son tour, doit mettre un de ses pions dans une colonne (et le pion tombe). Le premier joueur à avoir 4 de ses pions alignés (en ligne, colonne ou diagonale) remporte la partie.

1.3 Travail réalisé:

Dans un premier temps, nous présentant la version console et graphique.

Ensuite, on va présenter les différentes algorithmes, structures et fonctions utilisés pour rependre au travail demandé tout en décrivant leurs rôles et leurs fonctionnement globales.

1.4 Outils, frameworks et bibliothèques exploitées :

Cocos2d-x :

Cocos2d-x est le Framework open source #1 dans les différents Store (iOS, Android et Windows Phone).
Il nous offre plusieurs avantage, parmi :

- Permet de créer, d'un même code source, une interface graphique multiplateforme.
- Gestion plus stricte de la mémoire (Win32).
- Initialisation des données C++.

Modélisation du problème

2.1 Structuration du problème :

- **Espaces d'états :**

Un joueur est représenté par la classe Player, qui contient comme membre «int color » qui représente la couleur de son pion. Un joueur peut être humain ou encore une intelligence artificielle IA qui est représenté par une classe PlayerAI qui hérite de la classe Player.

color $\in \{1,2\}$

La grille de jeu est représentée comme une classe Board qui contient un tableau d'entiers de 6 lignes et 7 colonnes, les joueur Player1 qu'on le représente souvent par X et Player2 qu'on le représente souvent par O. On code par 0 un emplacement vide, 1 lorsqu'un pion du joueur 1 est présent, et 2 lorsqu'un pion du joueur 2 est présent.

$T[i][j] \in \{0,1,2\}$; avec $i \in [|0,5|]$ et $j \in [|0,6|]$;

L'espace d'états a donc comme membre la table T[6][7], qui contient les différents pions, le tour « turn » qui représente le tour courant et d'où on peut déduire qui doit jouer (turn%2+1).

Turn :

- **Objectif :**

Chaque joueur doit former une range ininterrompue de 4 couleur horizontale, vertical, ou diagonale. Le premier qui réalisera cette range gagne.

- **Etat initial :**

Dans l'état initial, on est dans le premier tour turn =0, et la grille est vide donc elle doit être initialiser à zero, ce qui correspond à T[6][7]={{0}}

- **Etat final :**

Elle correspond à un état où l'un des deux joueurs a réussi de compléter sa ranger consécutif des 4 pions.

- **Règles :**

On ne peut poser des pions que si la colonne est non plein.

R1 : jeter un pion sur la première colonne si elle est non plein.

R2 : jeter un pion sur la 2ème colonne si elle est non plein.

R3 : jeter un pion sur la 3ème colonne si elle est non plein.

R4 : jeter un pion sur la 4ème colonne si elle est non plein.

R5 : jeter un pion sur la 5ème colonne si elle est non plein.

R6 : jeter un pion sur la 6ème colonne si elle est non plein.

R7 : jeter un pion sur la 7ème colonne si elle est non plein.

- **Stratégie :**

La stratégie MinMax :

- Prévoir à l'avance les mouvements de l'adversaire.
- Prévoir un espace de recherche plus grand.
- Chercher le meilleur mouvement possible.

La stratégie AlphaBeta :

- Améliorer l'algorithme MinMax.
- Élaguer certaines branches explorées inutilement.

La fonction utilité :

- Max place les « O » alors que Min place les « X ». Pour un nœud :
- Si Max gagne, le score est de 512.
- Si Max perde, le score est de -512
- Si Match nul, le score est de 0

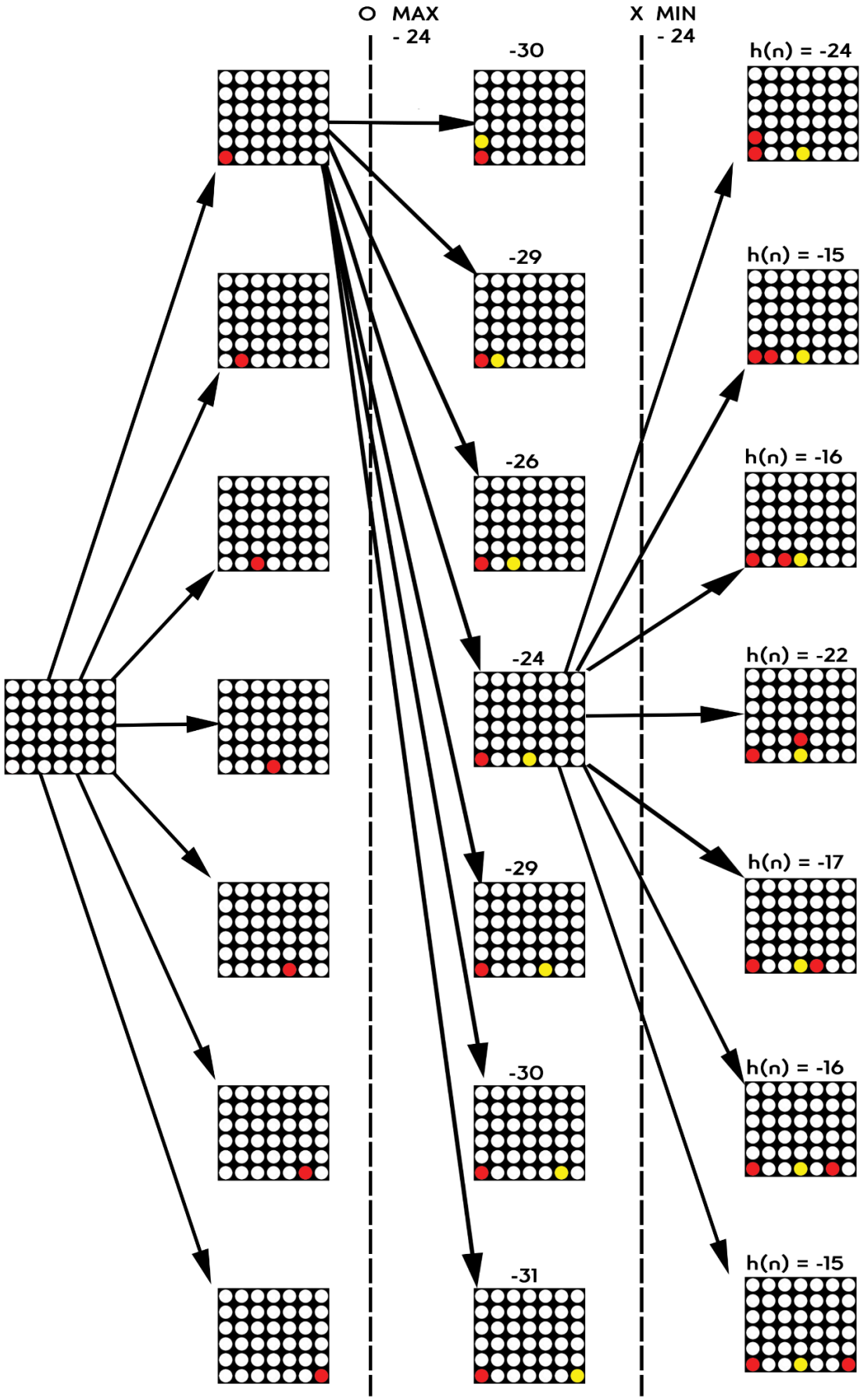
La fonction heuristique :

Et voici la fonction heuristique, si aucun des cas précédents ne s'applique après avoir atteint profondeur maximale. Pour chaque quatre nœuds connectés du plateau, on évalue selon le principe suivant :

- -50 s'il y a trois « O », et aucune « X ».
- -10 s'il y a deux « O », et aucune « X ».
- -1 s'il y a une « O », et aucune « X ».
- 0 s'il n'y a aucun pion, ou ils sont mixtes des « X « et des « O ».
- +1 s'il y a une « X », et aucune « O ».
- +10 s'il y a deux « X », et aucune « O ».
- +50 s'il y a trois « X », et aucune « O ».
- Et on ajoute un bonus de +16 si c'est le tour de Max, sinon -16.

2.2 Graphe de résolution Min-Max :

Le nombre des nœuds explorés 56 noeuds



Consonle output pour l'algorithme MinMax :

Evaluation = -19 (i,j) = (4,0), isMax? = false
Evaluation = -27 (i,j) = (4,0), isMax? = false
Evaluation = -28 (i,j) = (4,0), isMax? = false
Evaluation = -30 (i,j) = (4,0), isMax? = false
Evaluation = -20 (i,j) = (4,0), isMax? = false
Evaluation = -19 (i,j) = (4,0), isMax? = false
Evaluation = -18 (i,j) = (4,0), isMax? = false
On prend le min : Evaluation = -30 (i,j) = (0,0), isMax? = true

Evaluation = -27 (i,j) = (5,1), isMax? = false
Evaluation = -29 (i,j) = (5,1), isMax? = false
Evaluation = -19 (i,j) = (5,1), isMax? = false
Evaluation = -21 (i,j) = (5,1), isMax? = false
Evaluation = -20 (i,j) = (5,1), isMax? = false
Evaluation = -19 (i,j) = (5,1), isMax? = false
Evaluation = -18 (i,j) = (5,1), isMax? = false
On prend le min : Evaluation = -29 (i,j) = (0,0), isMax? = true

Evaluation = -26 (i,j) = (5,2), isMax? = false
Evaluation = -17 (i,j) = (5,2), isMax? = false
Evaluation = -22 (i,j) = (5,2), isMax? = false
Evaluation = -20 (i,j) = (5,2), isMax? = false
Evaluation = -19 (i,j) = (5,2), isMax? = false
Evaluation = -18 (i,j) = (5,2), isMax? = false
Evaluation = -17 (i,j) = (5,2), isMax? = false
On prend le min : Evaluation = -26 (i,j) = (0,0), isMax? = true

Evaluation = -24 (i,j) = (5,3), isMax? = false
Evaluation = -15 (i,j) = (5,3), isMax? = false
Evaluation = -16 (i,j) = (5,3), isMax? = false
Evaluation = -22 (i,j) = (5,3), isMax? = false
Evaluation = -17 (i,j) = (5,3), isMax? = false
Evaluation = -16 (i,j) = (5,3), isMax? = false
Evaluation = -15 (i,j) = (5,3), isMax? = false
On prend le min : Evaluation = -24 (i,j) = (0,0), isMax? = true

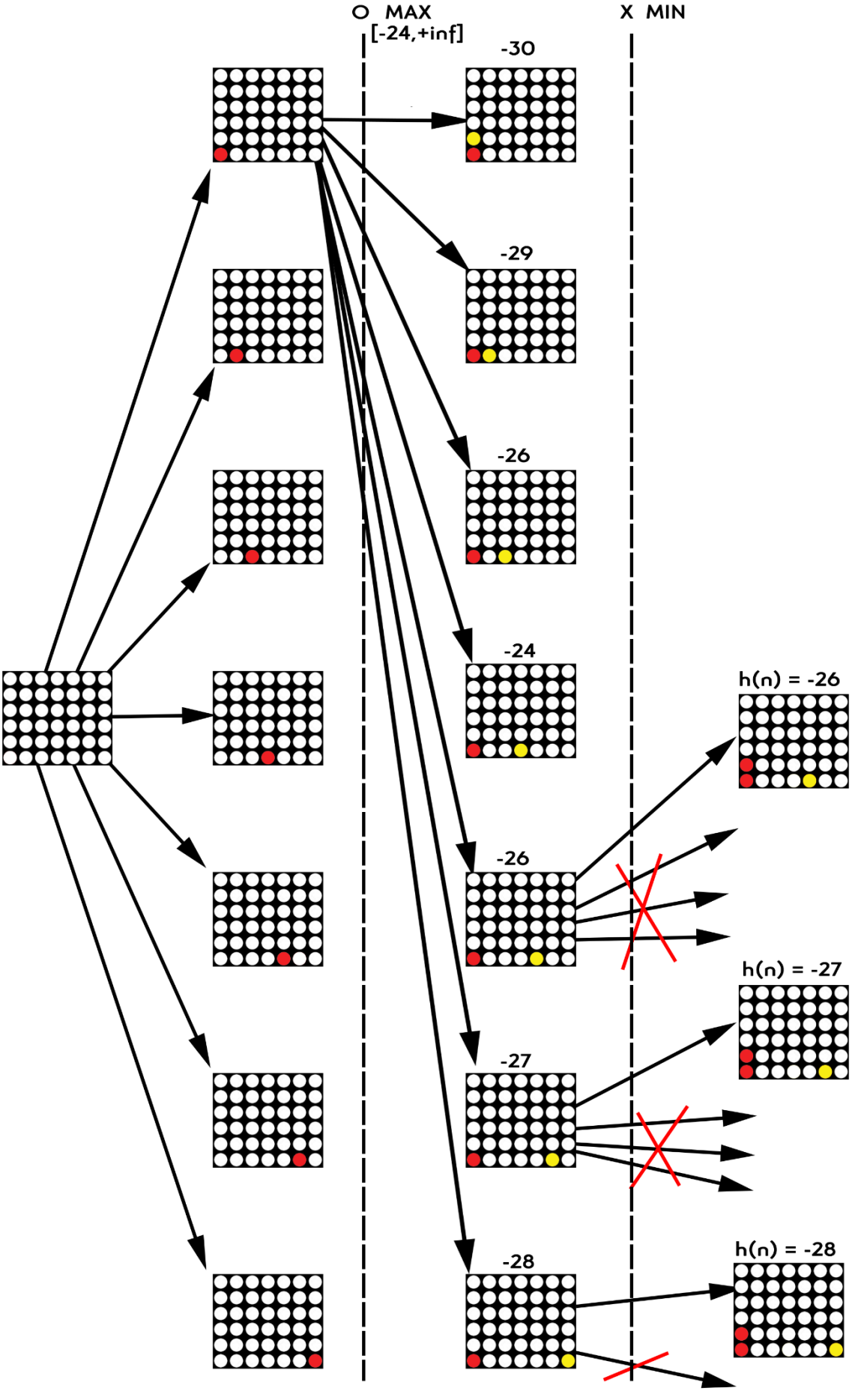
Evaluation = -26 (i,j) = (5,4), isMax? = false
Evaluation = -26 (i,j) = (5,4), isMax? = false
Evaluation = -27 (i,j) = (5,4), isMax? = false
Evaluation = -29 (i,j) = (5,4), isMax? = false
Evaluation = -22 (i,j) = (5,4), isMax? = false
Evaluation = -18 (i,j) = (5,4), isMax? = false
Evaluation = -17 (i,j) = (5,4), isMax? = false
On prend le min : Evaluation = -29 (i,j) = (0,0), isMax? = true

Evaluation = -27 (i,j) = (5,5), isMax? = false
Evaluation = -27 (i,j) = (5,5), isMax? = false
Evaluation = -28 (i,j) = (5,5), isMax? = false
Evaluation = -30 (i,j) = (5,5), isMax? = false
Evaluation = -20 (i,j) = (5,5), isMax? = false
Evaluation = -21 (i,j) = (5,5), isMax? = false
Evaluation = -18 (i,j) = (5,5), isMax? = false
On prend le min : Evaluation = -30 (i,j) = (0,0), isMax? = true

Evaluation = -28 (i,j) = (5,6), isMax? = false
Evaluation = -28 (i,j) = (5,6), isMax? = false
Evaluation = -29 (i,j) = (5,6), isMax? = false
Evaluation = -31 (i,j) = (5,6), isMax? = false
Evaluation = -21 (i,j) = (5,6), isMax? = false
Evaluation = -20 (i,j) = (5,6), isMax? = false
Evaluation = -20 (i,j) = (5,6), isMax? = false
On prend le min : Evaluation = -31 (i,j) = (0,0), isMax? = true

2.3 Graphe de résolution Alpha-Beta (Depth 2) :

Le nombre des nœuds explorées 38.



Consonle output pour l'algorithme AlphaBeta :

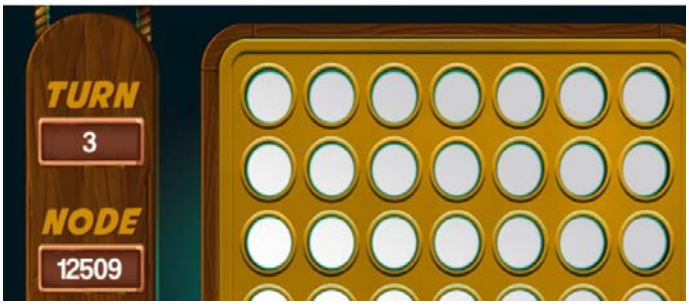
```
Evaluation = -19 and (i,j) = (4,0), isMax? = false
[alpha,beta]=[-100000000,100000000]
Evaluation = -27 and (i,j) = (4,0), isMax? = false
[alpha,beta]=[-100000000,-19]
Evaluation = -28 and (i,j) = (4,0), isMax? = false
[alpha,beta]=[-100000000,-27]
Evaluation = -30 and (i,j) = (4,0), isMax? = false
[alpha,beta]=[-100000000,-28]
Evaluation = -20 and (i,j) = (4,0), isMax? = false
[alpha,beta]=[-100000000,-30]
Evaluation = -19 and (i,j) = (4,0), isMax? = false
[alpha,beta]=[-100000000,-30]
Evaluation = -18 and (i,j) = (4,0), isMax? = false
[alpha,beta]=[-100000000,-30]
Evaluation = -30 and (i,j) = (0,0), isMax? = true

[alpha,beta]=[-100000000,100000000]
Evaluation = -27 and (i,j) = (5,1), isMax? = false
[alpha,beta]=[-30,100000000]
Evaluation = -29 and (i,j) = (5,1), isMax? = false
[alpha,beta]=[-30,-27]
Evaluation = -19 and (i,j) = (5,1), isMax? = false
[alpha,beta]=[-30,-29]
Evaluation = -21 and (i,j) = (5,1), isMax? = false
[alpha,beta]=[-30,-29]
Evaluation = -20 and (i,j) = (5,1), isMax? = false
[alpha,beta]=[-30,-29]
Evaluation = -19 and (i,j) = (5,1), isMax? = false
[alpha,beta]=[-30,-29]
Evaluation = -18 and (i,j) = (5,1), isMax? = false
[alpha,beta]=[-30,-29]
Evaluation = -29 and (i,j) = (0,0), isMax? = true

[alpha,beta]=[-30,100000000]
Evaluation = -26 and (i,j) = (5,2), isMax? = false
[alpha,beta]=[-29,100000000]
Evaluation = -17 and (i,j) = (5,2), isMax? = false
[alpha,beta]=[-29,-26]
Evaluation = -22 and (i,j) = (5,2), isMax? = false
[alpha,beta]=[-29,-26]
Evaluation = -20 and (i,j) = (5,2), isMax? = false
[alpha,beta]=[-29,-26]
Evaluation = -19 and (i,j) = (5,2), isMax? = false
[alpha,beta]=[-29,-26]
Evaluation = -18 and (i,j) = (5,2), isMax? = false
[alpha,beta]=[-29,-26]
Evaluation = -17 and (i,j) = (5,2), isMax? = false
[alpha,beta]=[-29,-26]
Evaluation = -26 and (i,j) = (0,0), isMax? = true

[alpha,beta]=[-29,100000000]
Evaluation = -24 and (i,j) = (5,3), isMax? = false
[alpha,beta]=[-26,100000000]
Evaluation = -15 and (i,j) = (5,3), isMax? = false
[alpha,beta]=[-26,-24]
Evaluation = -16 and (i,j) = (5,3), isMax? = false
[alpha,beta]=[-26,-24]
Evaluation = -22 and (i,j) = (5,3), isMax? = false
[alpha,beta]=[-26,-24]
Evaluation = -17 and (i,j) = (5,3), isMax? = false
[alpha,beta]=[-26,-24]
Evaluation = -16 and (i,j) = (5,3), isMax? = false
[alpha,beta]=[-26,-24]
Evaluation = -15 and (i,j) = (5,3), isMax? = false
[alpha,beta]=[-26,-24]
Evaluation = -24 and (i,j) = (0,0), isMax? = true

[alpha,beta]=[-26,100000000]
Evaluation = -26 and (i,j) = (5,4), isMax? = false
[alpha,beta]=[-24,100000000]
[alpha,beta]=[-24,-26]
Evaluation = -26 and (i,j) = (0,0), isMax? = true

[alpha,beta]=[-24,100000000]
Evaluation = -27 and (i,j) = (5,5), isMax? = false
```

[alpha,beta]=[-24,100000000]
[alpha,beta]=[-24,-27]
Evaluation = -27 and (i,j) = (0,0), isMax? = true

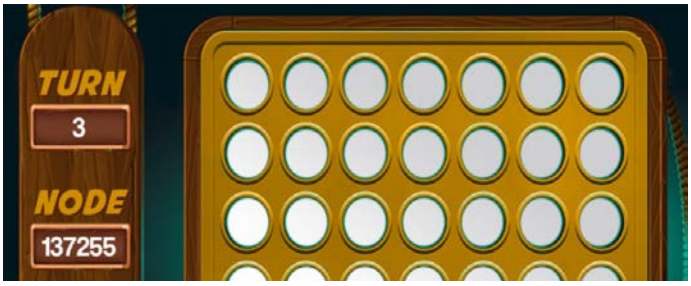
[alpha,beta]=[-24,100000000]
Evaluation = -28 and (i,j) = (5,6), isMax? = false
[alpha,beta]=[-24,100000000]
[alpha,beta]=[-24,-28]
Evaluation = -28 and (i,j) = (0,0), isMax? = true
[alpha,beta]=[-24,100000000]

Voici la comparaison enter l'algorithme MinMax et Alpha-Beta en termes de nœuds explorées :

L'algorithme MinMax :



L'algorithme Alpha-Beta :



Implémentation du Code

3.1 Structuration du problème :

- **Espaces d'états :**

Un joueur est implémenté par la classe **Player**, qui contient comme membre «int color » qui représente la couleur de son pion. Et comme méthode « int getPos() » qui prend la colonne joué par ce joueur. Pour l'ordinateur, une classe **PlayerAI** a été implémenter et qui hérite de la classe Player et elle a les méthodes spécialisées suivantes :

« int minMax(int state[6][7], int currentDepth, bool isMax) » : c'est l'implémentation de l'algorithme minMax qui permet de prévoir à l'avance les mouvements de l'adversaire et choisir le meilleur mouvement possible.

« int alphaBeta(int state[6][7], int currentDepth, bool isMax, int a, int b) » : c'est l'implémentation de l'algorithme alphaBeta qui permet d'améliorer l'algorithme MinMax en élaguant certaines branches explorées inutilement.

La grille de jeu est implémenté par la classe **Board** qui contient un tableau d'entiers de 6 lignes et 7 colonnes, les joueur Player1 (X) et Player2 (O). Elle contient les fonctions membres suivantes :

void play() : permet de tourner une partie entre deux joueurs.

bool makeMove() : permet au joueur de choisir une colonne et mettre un pion là-dedans.

emptyPlace(int) : permet de detecter la ligne

bool isWin(int, int, int) : permet de vérifier si un joueur à gagner une partie.

int countHorizontal(int i, int j, int pt) : calculé le nombre des pions connecté horizontalement à partir d'une position.

int countVertical(int i, int j, int pt) : calculé le nombre des pions connecté verticalement à partir d'une position.

int countDiagonal1(int i, int j, int pt) : calculé le nombre des pions connecté diagonal (/) à partir d'une position.

int countDiagonal2(int i, int j, int pt) : calculé le nombre des pions

connecté diagonal (\) à partir d'une position.

void editScore(int& score, int cp1, int cp2) : permet de changer le score donné comme référence en fonction de nombres des pions connectés du joueur 1 et 2.

int evalH (int i, int j, int &score, int pt) : évalue le score des 4 pion horizontalement commençant de la position (i, j) .

int evalV (int i, int j, int &score, int pt) : évalue le score des 4 pion verticalement commençant de la position (i, j) .

int evalD1(int i, int j, int &score, int pt) : évalue le score des 4 pion selon la première diagonale / commençant de la position (i, j) .

int evalD2(int i, int j, int &score, int pt) : évalue le score des 4 pion selon la deuxième diagonale \ commençant de la position (i, j) .

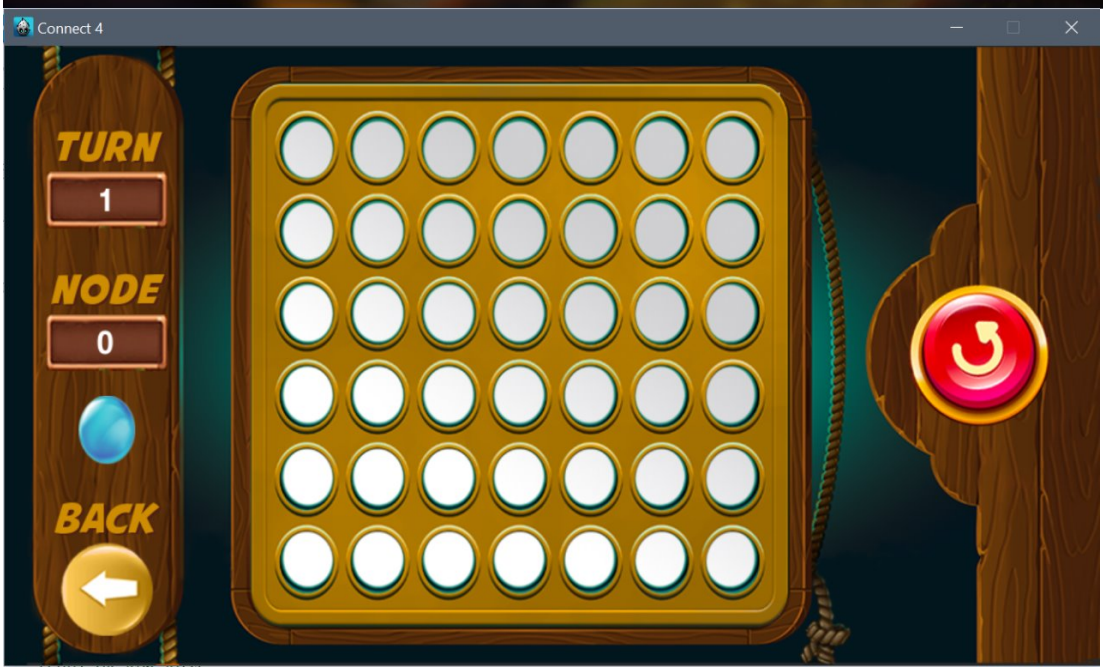
int heuristic(int pt) : permet d'évaluer l'état courant du jeu.

print() : afficher la table courante.

3.2 Simulation de programme :

L'exécution du programme est la suivante :


Connect 4
CONNECT FOUR
EASY
HARD


Connect 4
TURN
1
NODE
0
BACK

Conclusion :

Ce projet fut une très bonne expérience, vu qu'il a testé nos connaissances en résolution des problèmes acquises dans le 2er semestre. On a également pu améliorer nos compétences en modélisation et développement des différents algorithmes de recherche. Cela nous a permet de construire un AI que nous même ne pouvons pas vaincre.