Université d'Aix-Marseille - Polytech Marseille (Informatique)

Projet IA
Programmation du jeu morpion en C intégrant l'IA
(Travail individuel)

1. Organisation et Planning

Le projet va être réalisé individuellement. Vous devez implémenter en C le projet tel que décrit ci-dessous et rendre **avant le 21 janvier 2021 à minuit.**

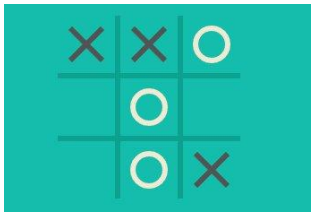
- le code source du projet et ;
- le rapport de 4 pages (maximum) expliquant comment vous avez réalisé ce projet (descriptif des algorithmes développés, les structures de données utilisées, calcul de complexité des algorithmes, organisation du programme...)

2. Enoncé du projet : implémentation d'un jeu de Morpion

2.1. Jeu du Morpion 1.0 de taille (3x3)

On se propose tout d'abord de programmer un jeu en console qui doit permettre à 2 utilisateurs (2 joueurs) de jouer tour à tour au clavier sur le morpion avec comme objectifs respectifs de gagner la partie.
Rappels sur le jeu du morpion :

- Le joueur qui joue en premier (joueur 1) peut mettre, lorsque c'est son tour, un cercle dans l'une des 9 cases du morpion ;
- Le joueur qui joue en second (joueur 2) peut mettre, à son tour, une croix dans l'une des cases ;
- Un joueur gagne la partie dès lors qu'il réussit à aligner 3 de ses symboles (cercles ou croix), sur une ligne, une colonne ou une diagonale.



Le programme se décompose en deux parties distinctes que nous détaillons :

- **Création du morpion et affichage du jeu vide :**

L'utilisateur a face à lui une grille (3 x 3) correspondant aux 9 cases du morpion. Initialement, la grille est vide. Pour coder le morpion, nous utiliserons un tableau à 2 dimensions (3 x 3) qui représente la grille de 9 cases :

```
typedef char morpion[3][3];
```

Chaque case de coordonnées (i, j) pourra contenir 3 valeurs différentes :
'-1' initialement, pour une case vide ;
'0' pour une case remplie par le joueur 1 ;
'1' pour une case remplie par le joueur 2 ;

- **Jeu tour à tour :**

Il s'agira ensuite de faire jouer tour à tour les 2 joueurs : le joueur 1 peut jouer 5 fois, tandis que le joueur 2 ne peut jouer que 4 fois, pour un total maximal de 9 tours. Voici une astuce pour récupérer :
Le joueur pour une variable i comprise entre 0 et 9 tours : joueur = i % 2;
Le joueur doit bien entendu rejouer tant qu'il n'a pas saisi un chiffre compris entre 1 et 9, et tant qu'il n'a pas saisi un chiffre correspondant à une case vide, c'est-à-dire une case qui ne soit pas déjà remplie par lui-même ou l'autre joueur.

Il vous est conseillé de définir deux fonctions importantes :

La fonction ***jouer*** qui prendra en paramètre le jeu de type **morpion**, un indice de ligne i et un indice de colonne j du jeu, et le joueur. Cette fonction renverra **1** si le joueur doit rejouer dans le cas où il a saisi une case non vide. Sinon cette fonction renverra **0**. Si la case saisie est vide, i.e. si la case saisie contient la valeur **-1**, alors elle sera affectée à la valeur de la variable joueur, i.e. '**0**' ou '**1**'.

La fonction ***gagnant*** qui prendra en paramètre le jeu de type **morpion**, et la variable joueur. Cette fonction renverra **1** si le joueur a gagné, i.e. s'il a réussi à aligner 3 cercles ou croix sur une ligne, une colonne ou une diagonale selon la valeur de joueur, i.e. respectivement '**0**' ou '**1**'. Sinon cette fonction renverra **0**. Si les 9 cases sont pleines ou s'il est impossible de gagner ou de perdre, alors la fonction renverra **-1**.

2.2. Jeu du Morpion 2.0 de taille (NxN) intégrant l'IA

- Modifiez votre programme pour résoudre le jeu de taille quelconque N. **N** est une valeur saisie par l'utilisateur avant de jouer au jeu.
- Appliquez l'intelligence artificielle afin qu'un joueur puisse jouer avec l'ordinateur. L'algorithme MinMax peut être utilisé comme logique de l'ordinateur (voir page suivante).
- Dotez votre jeu d'une jolie interface graphique. Vous pouvez utiliser la bibliothèque EZ-Draw ("Easy Draw") http://pageperso.lif.univ-mrs.fr/~edouard.thiel/ez-draw/index.html
  Ou encore la bibliothèque SDL (Simple DirectMedia) https://www.libsdl.org/index.php

Contact de l'enseignant :
ibtissem.daoudi@univ-amu.fr

L’algorithme min-max

Principe

L’algorithme min-max est particulièrement adapté aux jeux se jouant à tour de rôle, comme par exemple le morpion, parmi les plus simples. Le but de min-max est de trouver le meilleur coup à jouer à partir d’un état donné du jeu. Le principe consiste à construire un arbre de jeu : c’est un arbre qui représente toutes les évolutions possibles du jeu à partir de la configuration actuelle. Il amène l’ordinateur à passer en revue toutes les possibilités pour un nombre limité de coups et à leur assigner une valeur qui prend en compte les bénéfices pour le joueur et pour son adversaire. Le meilleur choix est alors celui qui minimise les pertes du joueur tout en supposant que l’adversaire cherche au contraire à les maximiser (le jeu est à somme nulle).

Comment ça Marche ?

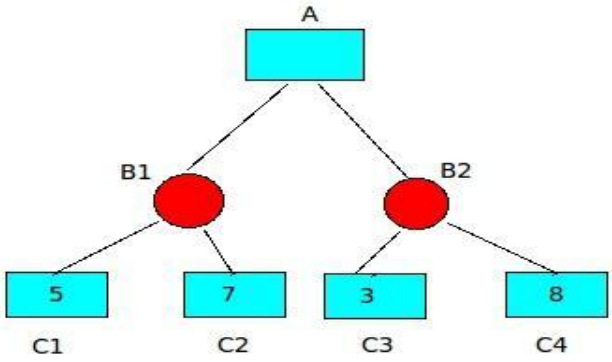


Figure 1. Un exemple de l’algorithme min-max.

1. La racine (le nœud A) représente l’état actuel du jeu, et la question est ici de savoir si le meilleur coup consiste en B1 ou B2. On va donc se demander ce que va jouer l’adversaire si l’on joue B1, puis B2, et choisir le coup qui nous avantagera le plus, c’est-à-dire donner un « poids » à B1 et B2.
2. En effet, si on joue B1, l’adversaire va choisir le coup qui nous avantage le moins entre C1 et C2, c’est-à-dire C1, qui à le poids le plus faible. Donc si on joue B1, on arrivera à la configuration C1, dont le poids est 5.
3. De même, si on joue B2, l’adversaire nous mènera en C3, dont le poids est 3.
4. On peut donc dire que le poids de B1 est 5, et celui de B2, 3. On choisira donc B1, et on recommence à partir de C1.

On va choisir le maximum des nœuds fils de A, sachant que chaque nœud fils de A est lui-même le minimum de ses nœuds fils, et que chaque nœud fils des nœuds fils de A est lui-même le maximum des ses propres nœuds fils, etc... Jusqu’à arriver aux feuilles qui symbolisent la fin de la partie.

Exemple avec un jeu du morpion

Supposons que nous arrivions à l’état A dans la figure 2. L’ordinateur (O) doit décider du prochain coup. En utilisant l’algorithme min-max, l’ordinateur créera 4 possibilités. Pour chaque possibilité, tous les mouvements sont simulés. Le score +10 est donné si l’humain est le gagnant, et -10 est pour l’ordinateur. Le score 0 signifie la condition égale (impossible de gagner ou de perdre). Le but

est de choisir le score le plus bas parmi les 4 possibilités afin que l'ordinateur puisse être le gagnant.

Remarque : ces valeurs sont arbitraires. On aurait pu mettre n'importe quelle autre valeur, le but étant de donner une valeur forte à une victoire de l'IA, une valeur faible pour une défaite et une valeur intermédiaire pour un résultat nul.

```
MinMax(morpion){
      Score <- +infini
      Coordonnées <- [0,0]

      Pour chaque case ij de morpion
            Si ij est vide alors
                  case <- 'O'
                  valeur <- max(morpion)
                  Si Score > valeur
                        Score <- valeur
                        Coordonnées <- ij
                  ij <- vide
      fin pour

      renvoyer Coordonnées
}
```

```
Max(morpion){
   Si gagnant(morpion, 'X')
      Renvoyer Score <- 10
   Sinon si gagnant(morpion, 'O')
      Renvoyer Score <- -10
   Sinon si gagnant est impossible
      Renvoyer Score  <- 0

   Score <- -infini
   Pour chaque case ij :
      Si ij est vide alors
         ij <-'X'
         valeur <- min(morpion)
         Si Score < valeur
            Score<-valeur
         ij <- vide
   fin pour

   renvoyer Score
}
```

```
Min(morpion){
   Si gagnant(morpion, 'X')
      Renvoyer Score <-10
   Sinon si gagnant(morpion, 'O')
      Renvoyer Score <- -10
   Sinon si gagnant est impossible
      Renvoyer Score  <-0

   Score <- +infini
   Pour chaque case ij :
      Si ij est vide alors
         ij<-'O'
         valeur <- max(morpion)
         Si Score > valeur
            Score <- valeur
         ij <- vide
   fin pour

   renvoyer Score
}
```

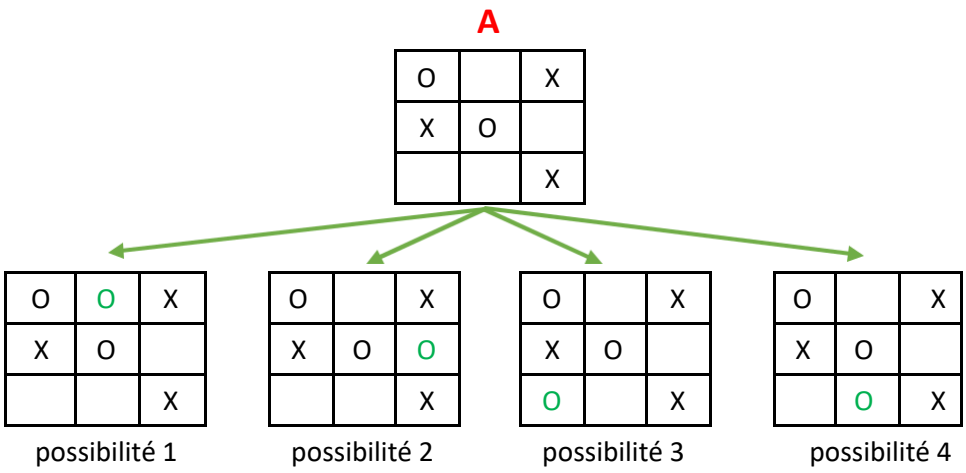
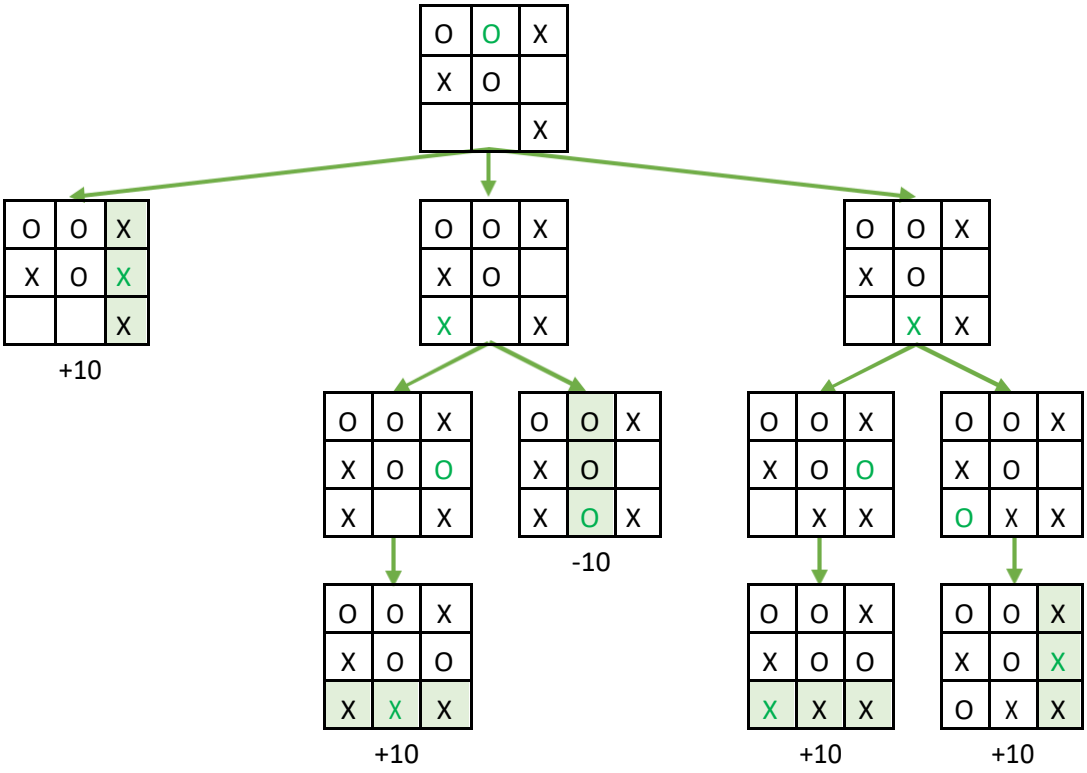
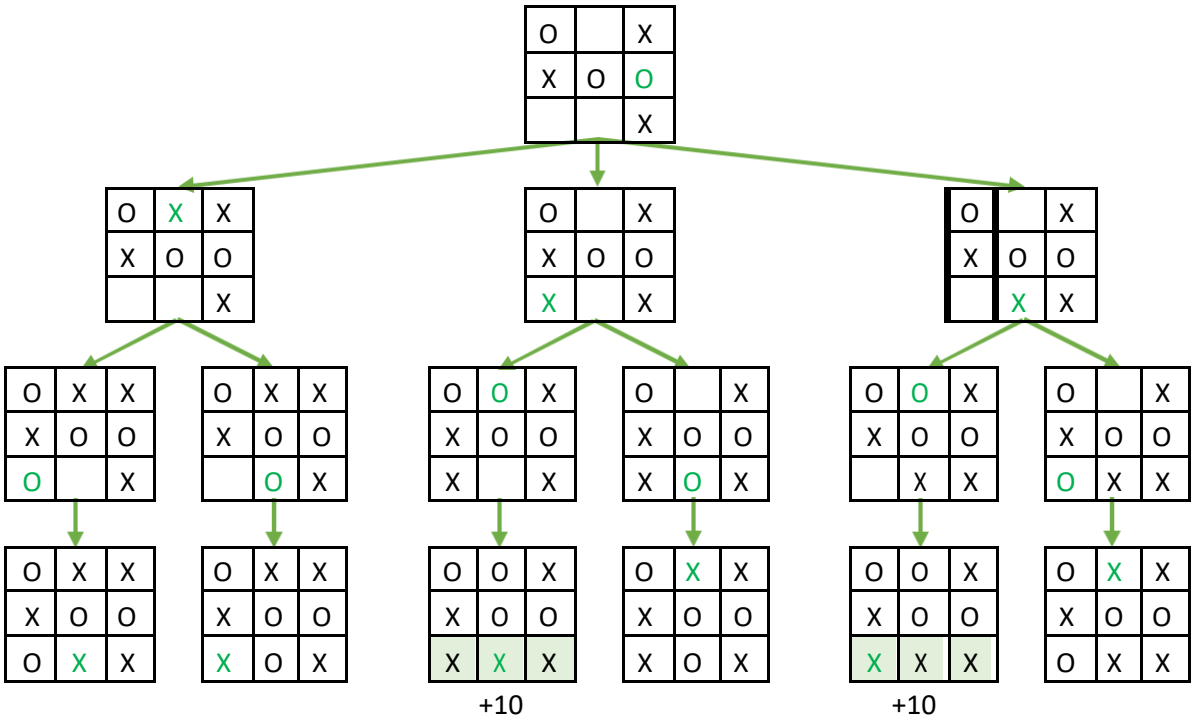


Figure 2. 4 possibilités pour le prochain coup.

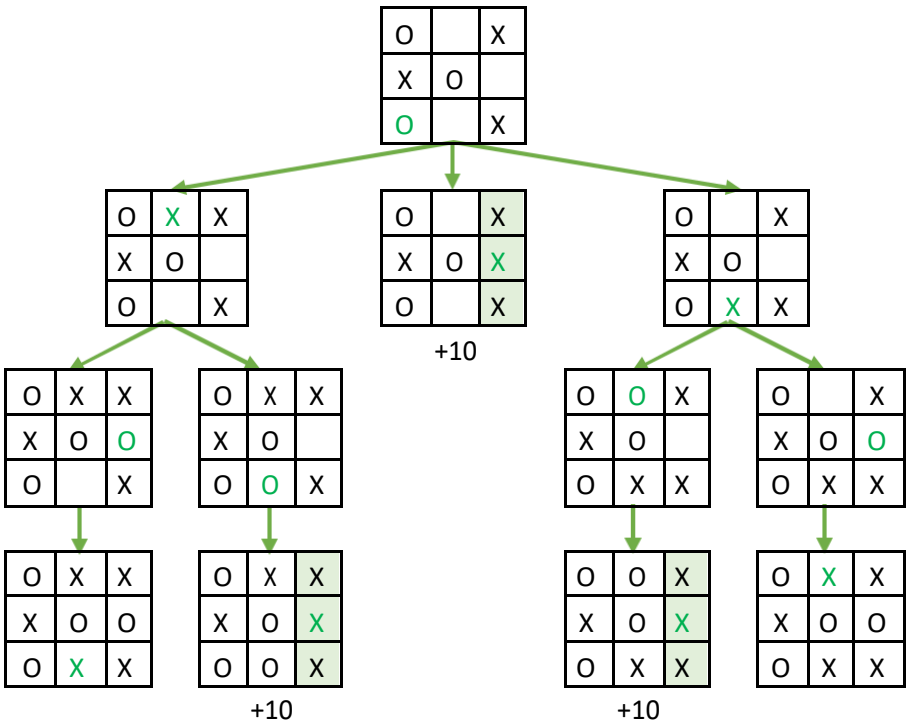
Possibilité 1



Possibilité 2



Possibilité 3



Possibilité 4