

Fil Rouge 2023

Minimax

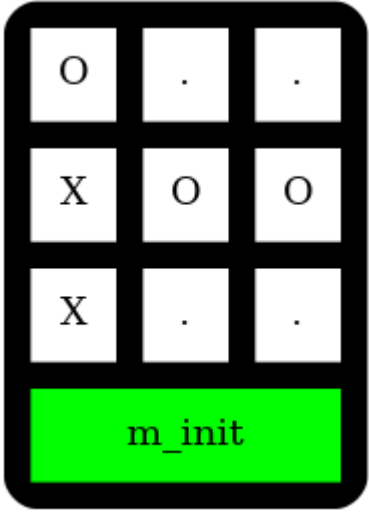
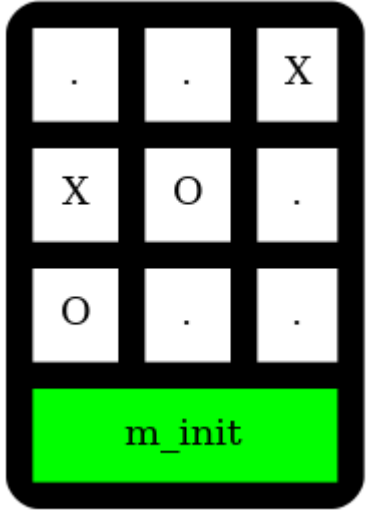
Organisation

- Ce Fil Rouge doit être réalisé en équipe. Chaque équipe doit être constituée au maximum de 4 étudiants, appartenant au même groupe TD. Il ne doit pas y avoir plus de 8 équipes par groupe TD : l'objectif est de pouvoir organiser au maximum 8 soutenances par plage de 4h afin d'accorder 30 minutes à chaque soutenance.
- Chaque équipe rendra une seule archive sur moodle, contenant :
 - Les fichiers sources du programme produit ;
 - Un fichier makefile permettant de compiler le programme
 - Un compte-rendu CR rédigé (avec introduction, développement, conclusion, perspectives et bibliographie) présentant l'organisation du programme et celle du groupe (qui a fait quoi, avec quel planning, quelles ont été les difficultés, etc.)
- Des soutenances seront organisées à la rentrée des congés de Noël. Elles auront lieu en groupe TD, avec un jury composé de deux enseignants. Chaque soutenance durera 30 minutes : 20 minutes de présentation, 10 minutes de questions.
- L'évaluation du fil rouge portera sur la qualité du code livré et de la livraison, sa compétitivité, la qualité du CR et la soutenance.

Programme 1 : **tttree**

Produire le programme **tttree** générant la représentation graphique d'un arbre de décision d'un tic-tac-toe à l'aide d'un **minimax basique** :

- La position initiale du tic-tac-toe sera déterminée depuis un argument en ligne de commande, au format FEN.
- Cet **UNIQUE** argument sera une chaîne de caractères comportant :
 - L'état de la grille formé à partir des caractères o,x,O,X et des chiffres 1 à 9
 - Un espace
 - Le joueur au trait (o,x)
- Par exemple :
 - Le morpion de la figure 1 se représente par la chaîne : "o2xoox2 x"
 - Le morpion de la figure 2 se représente par la chaîne : "2xxo1o2 o"

| | |
|---|--|
|  |  |
| Figure 1 (trait X) | Figure 2 (trait O) |

- Le fichier dot généré sera produit sur la sortie standard
 - Les noeuds MAX devront être affichés en vert, les noeuds MIN en rouge
- Les messages de débogage ou d'erreur éventuels seront fournis sur la sortie d'erreur
- Les valeurs remontées par le minimax devront être placées à côté de chaque noeud
- Le script suivant devra être utilisé pour tester votre programme :

```
#!/bin/bash

echo "Test de ttree pour le morpion 1o11o1oxx x"
./tttree "1o11o1oxx x" > g1.dot
dot g1.dot -T png -o g1.png

echo "Test de ttree pour le morpion x21o11xo o"
./tttree "x21o11xo o" > g2.dot
dot g2.dot -T png -o g2.png
```

Exemple de fichiers graphiques à produire pour les morpions d'exemple :

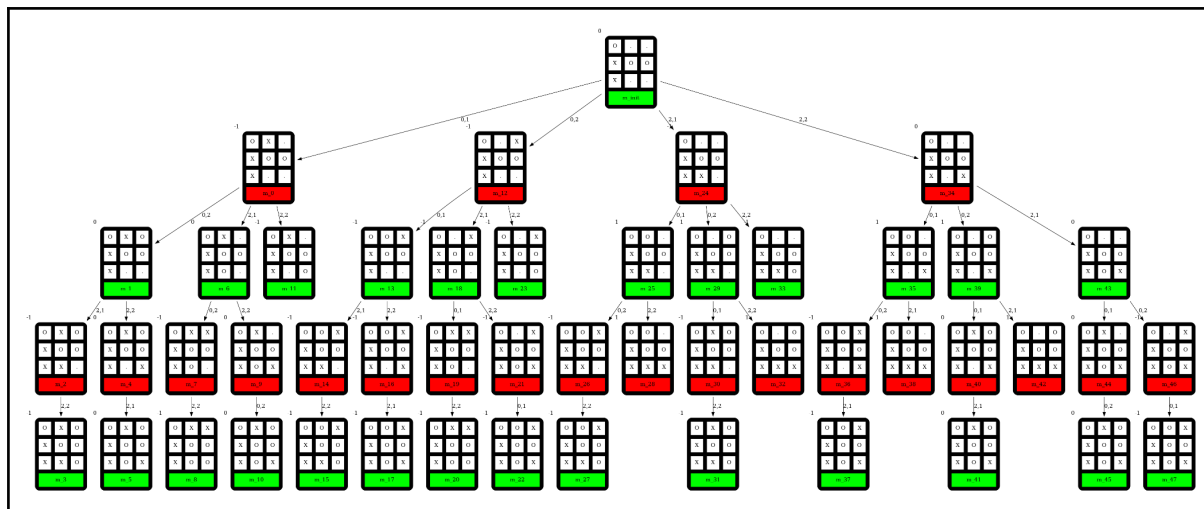


Figure 3 : arbre de décision pour le morpion de la figure 1 : [figure3.png](#)

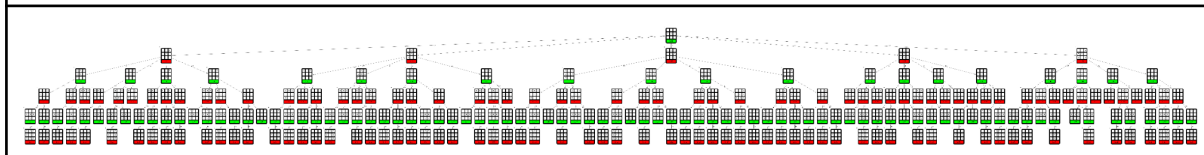


Figure 4 : arbre de décision pour le morpion de la figure 2 : [figure4.png](#)

Programme 2 : sm-refresh

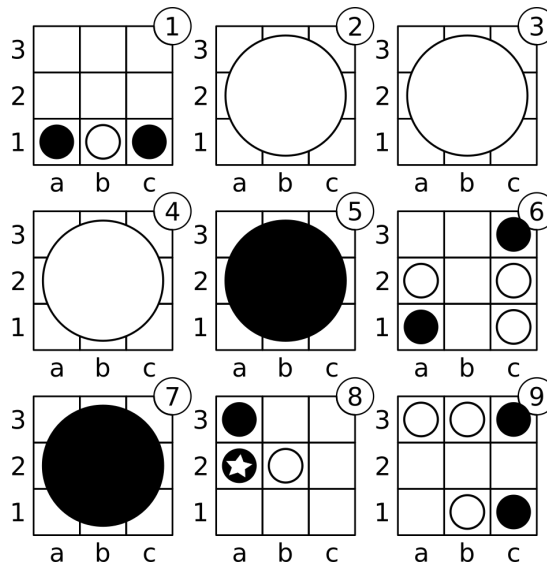
Produire un programme permettant de jouer au **super-morpion** à l'aide d'un **minimax basique** contre l'ordinateur

- La saisie des coups se fera sur l'entrée standard, en notation algébrique (comme aux échecs) : `<#grille>(1..9) espace <#colonne>(a..c) <#ligne>(1..3)`
 - Par exemple, **3 c3** pour jouer dans la case la plus en haut et à droite de la grille la plus en haut et à droite
- L'horizon de recherche du minimax sera défini à l'aide d'un paramètre fourni en ligne de commande
- L'affichage des positions devra se faire à la fois de manière pseudo-graphique sur la sortie standard et graphique dans un fichier nommé par défaut [g.png](#) à l'aide de graphviz
- La page HTML [refresh.html](#) pourra être utilisée pour visualiser les positions du super-morpion après les coups de chacun des joueurs
- Le programme devra être sensible aux variables d'environnement suivantes :
 - **DEBUG** pour activer un mode de débogage
 - **SMPATH** pour indiquer le chemin du fichier image à générer

Programme 3 : sm-bot

Développer un programme permettant de calculer le meilleur coup au jeu du super morpion, à partir d'une position saisie en ligne de commande, et de le renvoyer sur sa sortie standard

- Exploiter les principes de négamax et de l'élagage alpha-bêta
- Mettre en oeuvre une stratégie permettant de garantir une réponse dans un délai contraint
- La position initiale du super-morpion sera déterminée depuis un argument en ligne de commande, au format FEN.
- Ce **premier argument** sera une chaîne de caractères comportant :
 - 9 chaînes FEN comme précédemment (o,x,O,X,1..9)
 - concaténées pour décrire l'état de chaque grille
 - Un espace
 - Le dernier coup joué (00 si l'on représente le début de la partie) :
`<#grille>(0..9) <#case>(0..9)`
 - Un espace
 - Le joueur au trait (o,x)
- Un **second argument** sera fourni depuis un argument en ligne de commande du programme, représentant le temps restant, en secondes, pour terminer la partie
- Par exemple, la chaîne "6xoxOOOX2xo1ox1oXx2xo4oox4ox 84 o" représentera la position suivante, trait aux O :



- Le coup renvoyé par le programme devra être fourni au format suivant :
`<#grille>(1..9) <#case>(1..9)`
 - Par exemple, 65 pour jouer dans la case du milieu de la grille 6

Ce troisième programme sera utilisé pour organiser un tournoi de bots à la cadence de 15min+30s/coup, permettant de déterminer un classement qui contribuera à la note de livrable.