Résolution de jeux par une IA

Exercice 1 : Morpion

Règles du jeu

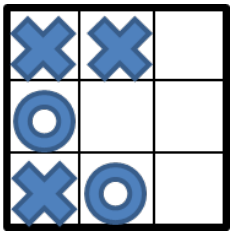
Morpion est un jeu séquentiel à information parfaite et complète dans lequel deux joueurs s'affrontent autour d'une grille de 3 lignes et 3 colonnes.

Tour à tour, les joueurs inscrivent dans une case libre le symbole qui leur correspond (une croix ou un cercle). Le joueur ayant les croix commence la partie.

La partie est terminée lorsque la grille est pleine ou si l'un des deux joueurs arrivent à aligner leur symbole sur une ligne, une colonne ou une diagonale.

Consignes

La partie a commencé et nous sommes arrivés à cette grille :



Quel est le prochain coup que le joueur qui a les cercles doit joueur ?
Pour répondre :

1. Développez l'arbre de jeu avec minimax jusqu'aux feuilles (-1 pour une défaite, 0 pour un match null, et 1 pour une victoire).
2. Quelles branches seraient coupées par alpha-bêta ?

Exercice 2 : Challenge Pyramide

Règles du jeu

Pyramide est un jeu séquentiel à information parfaite et complète dans lequel deux joueurs s'affrontent: le premier joueur avec des bloc de marbre noir et le second avec des blocs de marbre blanc. Au départ, les joueurs reçoivent 15 blocs chacun qu'ils doivent disposer sur un plateau afin de former une pyramide.

Le premier étage de la pyramide fait 4×4 blocs, le second 3×3, etc. Le but du jeu est de poser son bloc au sommet de la pyramide. Sur la base de la pyramide, on peut poser un bloc où l'on veut. Sur les étages suivants, on ne peut poser un bloc qu'au milieu d'un carré formé au niveau juste en-dessous. Un joueur perd s'il n'a plus de bloc dans sa réserve. Le jeu va donc consister à "économiser" des blocs dans sa réserve. Pour cela, il y a deux règles très simples:

- un joueur peut déplacer un bloc si aucun autre bloc ne repose sur elle et à condition de la replacer sur un niveau strictement supérieur;
- si un joueur forme un carré avec ses blocs (sur un même niveau), il peut alors retirer jusqu'à deux blocs du plateau pour les remettre dans sa réserve.

Consignes

En utilisant le matériel fourni, implémentez l'heuristique qui sera utilisée par l'algorithme ALPHABETA implémenté pour vous. Pour cela, une trame est fournie dans le code Python joint à ce TP. Il vous reste à trouver une heuristique rapide à calculer et permettant de mener votre Intelligence Artificielle à la victoire.

A la fin du TD (ou plus raisonnablement, après rendu de vos IA), les IA s'affronteront dans un tournoi lors duquel votre IA devra donner son prochain coup en moins de 10s, sans quoi elle passera son tour.

Matériel

Sources

Téléchargez l'archive Pyramide et décompressez-la dans le dossier de votre choix sur votre ordinateur. Le dossier contient plusieurs fichiers sources en Python.

Nous avons implémenté deux types de joueurs: un joueur humain, que l'on peut instancier avec la classe HumanPlayer, et une IA, que l'on peut instancier avec la classe IAPlayer.

Focalisez-vous sur le fichier aiplayer.py. Ce fichier définit la classe AIPlayer. Vous pouvez implémenter autant de méthodes que nécessaire dans cette classe, mais il faut bien comprendre que tout le reste ne doit pas être modifié, en particulier :

- le constructeur de la classe (la fonction ___init___) ne doit pas être changé,
- n'ajoutez pas de paramètres aux fonctions que l'on vous donne.

Ne changez pas non plus le nom du fichier aiplayer.py, à part au moment de le soumettre sur edunao.

La date limite de rendu de votre programme (aiplayer.py) est le 18 décembre. Il est attendu un programme par personne : par défaut, renommer le fichier aiplayer.py par nom_prenom.py.

Nous avons déjà implémenté pour vous l'algorithme alpha-bêta vu en cours. Vous devez vous concentrer sur la fonction `heuristic` de l'IA.

Cette fonction heuristique donne une note à un plateau, sous la forme d'un flottant. Plus la valeur est positive et grande, plus le plateau est favorable pour le joueur Max (celui pour qui on choisit le coup à jouer). Inversement, plus l'heuristique est négative et petite, plus le plateau est défavorable à Max. Pour calculer cette heuristique, la fonction reçoit le plateau (paramètre board). Le premier joueur se voit attribuer la valeur -1 (noir) et le second joueur se voit attribuer la valeur +1 (blanc). Pour accéder à cette valeur pour votre IA, et ainsi identifier qui est Max, utilisez la commande self.player.

Pour éventuellement permettre à alpha-bêta d'atteindre une plus grande profondeur, vous pouvez trier la liste des coups possibles comme vous le souhaitez en implémentant ce tri dans la fonction sortmoves. Ceci est purement facultatif.

N'oubliez pas de modifier la variable `name` en lui affectant vos nom/prénom. Ce nom sera affiché lorsque votre IA sera en train de jouer.

Avant de le remettre aux enseignants, renommez ce fichier: NOM_prenom.py, afin que les différents fichiers ne s'écrasent pas.

Vous n'avez pas besoin d'ouvrir un autre fichier que le fichier aiplayer.py.

Classe Board

La classe Board représente un plateau. Vous n'avez pas besoin de lire le code pour développer votre IA. Vous pouvez obtenir l'ensemble des éléments du plateau grâce à la propriété board.cells. Cette propriété est une liste dans laquelle le premier niveau (la base) est à l'indice 0, et le sommet à l'indice 3. Notez que chaque niveau level a exactement $4 - level$ lignes et $4 - level$ colonnes.

Vous pouvez afficher un plateau avec la commande print. Par exemple, pour la grille initiale, vous obtiendrez:

```
. . . .
      . . .
. . . .      . .
      . . .      .
. . . .      . .
```

```
          . . .
. . . .

remaining W:15  remaining B:15
```

Chaque point indique un emplacement vide. Le joueur blanc (W comme White) a 15 blocs, de même pour le joueur noir (B comme Black).

En cours de partie, un plateau peut ressembler à cela:

```
B W B W
          W B B
B W W W             B W
          W W B           .
B W B B               . .
          . . B
W W W W

remaining W:1   remaining B:4
```

Pour accéder au niveau *lev*, à la case qui est sur la ligne *line* et sur la colonne *col*, il suffit d'écrire:
board.cells[lev][line][col].

En effet, board.cells est une liste imbriquée en python, qui, pour l'exemple précédent, ressemble à ceci: $[[[-1, 1, -1, 1], [-1, 1, 1, 1], [-1, 1, -1, -1], [1, 1, 1, 1]], [[1, -1, -1], [1, 1, -1], [0, 0, -1]], [[-1, 1], [0, 0]], [[0]]]$.

Comment jouer ?

Vous pouvez exécuter le script main.py pour tester votre IA contre elle-même (attention, au début, l'heuristique fournie va renvoyer 0 donc le comportement de l'IA ne sera pas très correct). Vous pouvez remplacer une des instances de votre IA par HumanPlayer afin d'affronter votre IA.