A propos

Il y a deux fichiers principaux dans ce projet : runner.py et tictactoe.py. tictactoe.py contient toute la logique pour jouer le jeu, et pour faire des mouvements optimaux. runner.py a été implémenté pour vous, et contient tout le code pour faire fonctionner l'interface graphique du jeu. Une fois que vous aurez complété toutes les fonctions requises dans tictactoe.py, vous devriez être en mesure d'exécuter python runner.py pour jouer contre votre IA !

Ouvrons tictactoe.py pour comprendre ce qui est fourni. Tout d'abord, nous définissons trois variables : X, O, et EMPTY, pour représenter les mouvements possibles du plateau.

La fonction initial_state renvoie l'état initial du plateau. Pour ce problème, nous avons choisi de représenter le plateau comme une liste de trois listes (représentant les trois lignes du plateau), où chaque liste interne contient trois valeurs qui sont soit X, soit O, soit EMPTY. Ce qui suit est constitué de fonctions que nous vous laissons le soin d'implémenter !

Consignes

Complétez les implémentations des fonctions player, actions, result, winner, terminal, utility et minimax.

- La fonction player doit prendre en entrée l'état du plateau de jeu et retourner quel est le tour du joueur (X ou O).
  - Dans l'état initial du jeu, X joue le premier coup. Par la suite, le joueur alterne à chaque coup supplémentaire.
  - Toute valeur de retour est acceptable si un plateau terminal est fourni en entrée (c'est-à-dire si le jeu est déjà terminé).
- La fonction actions doit renvoyer un ensemble de toutes les actions possibles sur un plateau donné.
  - Chaque action doit être représentée par un tuple (i, j) où i correspond à la ligne du déplacement (0, 1 ou 2) et j correspond à la cellule de la ligne correspondant au déplacement (également 0, 1 ou 2).
  - Les déplacements possibles sont toutes les cases du tableau qui ne contiennent pas déjà un X ou un O.
  - Toute valeur de retour est acceptable si un tableau terminal est fourni en entrée.
- La fonction de résultat prend en entrée un tableau et une action, et doit renvoyer un nouvel état du tableau, sans modifier le tableau d'origine.
  - Si l'action n'est pas valide pour la carte, votre programme doit lever une exception.
  - L'état du plateau retourné doit être le plateau qui résulterait de la prise du plateau d'origine en entrée, et de la possibilité pour le joueur dont

c'est le tour d'effectuer son déplacement à la case indiquée par l'action en entrée.
    - Il est important de ne pas modifier le tableau d'origine, car Minimax devra prendre en compte un grand nombre d'états de tableau différents au cours de son calcul. Cela signifie que la simple mise à jour d'une cellule dans le tableau lui-même n'est pas une implémentation correcte de la fonction de résultat. Vous voudrez probablement faire une copie profonde du tableau avant de faire des changements.
- La fonction winner doit accepter un tableau en entrée, et retourner le gagnant du tableau s'il y en a un.
    - Si le joueur X a gagné la partie, votre fonction doit retourner X. Si le joueur O a gagné la partie, votre fonction doit retourner O.
    - Un joueur peut gagner la partie en jouant trois fois de suite horizontalement, verticalement ou en diagonale.
    - Vous pouvez supposer qu'il y aura au plus un gagnant (c'est-à-dire qu'aucun tableau n'aura jamais les deux joueurs avec trois coups dans une rangée, puisque ce serait un état de tableau invalide).
    - S'il n'y a pas de gagnant (soit parce que le jeu est en cours, soit parce qu'il s'est terminé par une égalité), la fonction doit renvoyer None.
- La fonction terminal doit accepter un plateau en entrée et renvoyer une valeur booléenne indiquant si le jeu est terminé.
    - Si le jeu est terminé, soit parce que quelqu'un a gagné le jeu, soit parce que toutes les cases ont été remplies sans que personne ne gagne, la fonction doit renvoyer True.
    - Dans le cas contraire, la fonction doit renvoyer False si le jeu est toujours en cours.
- La fonction d'utilité doit accepter un plateau terminal en entrée et renvoyer l'utilité du plateau.
    - Si X a gagné le jeu, l'utilité est de 1. Si O a gagné le jeu, l'utilité est de -1. Si le jeu s'est terminé par une égalité, l'utilité est de 0.
    - Vous pouvez supposer que l'utilité ne sera appelée sur un tableau que si terminal(tableau) est Vrai.
- La fonction minimax doit prendre un tableau en entrée, et retourner le mouvement optimal pour le joueur sur ce tableau.
    - Le coup retourné doit être l'action optimale (i, j) qui est l'une des actions autorisées sur le plateau. Si plusieurs mouvements sont également optimaux, n'importe lequel de ces mouvements est acceptable.
    - Si le tableau est un tableau terminal, la fonction minimax doit renvoyer None.

Pour toutes les fonctions qui acceptent un tableau en entrée, vous pouvez supposer qu'il s'agit d'un tableau valide (à savoir, qu'il s'agit d'une liste qui contient trois lignes, chacune avec trois valeurs X, O ou EMPTY). Vous ne devez pas modifier les déclarations de fonctions (l'ordre ou le nombre d'arguments de chaque fonction) fournies.

Une fois que toutes les fonctions sont correctement implémentées, vous devriez être en mesure d'exécuter python runner.py et de jouer contre votre IA. Et, puisque le Tic-Tac-Toe est un match nul si les deux parties jouent de manière optimale, vous ne devriez jamais pouvoir battre l'IA (bien que si vous ne jouez pas de manière optimale, elle pourrait vous battre !)