Séance 2

1) Implémentation d'un minimax générique

Minimax est un algorithme applicable à différents jeux à deux joueurs. Avant de continuez, assurez-vous d'avoir compris son principe (cf le site du projet ou Google). Pour l'implémenter il suffit de disposer de fonctions permettant, pour un état quelconque de jeu (un nœud de l'arbre), de :

- obtenir les états immédiatement accessibles (les nœuds enfants) afin de construire l'arbre du jeu
- évaluer l'état (attribuer un score au nœud) afin d'estimer l'avantage qu'a un joueur sur son adversaire cet état.

> Pour la suite du projet, on fixe la convention que la fonction d'évaluation renvoie un score d'autant plus grand que les blancs ont l'avantage (d'autant plus petit que les noirs ont l'avantage).

1) En supposant qu'un objet de type node possède des méthodes *get_children* et *evaluate* remplissant les fonctions précédentes, implémentez, dans le module **minimax/limited_depth.py** de votre package, une fonction minimax :

- prenant en entrée un état de jeu, un boolean signifiant si le but est de maximiser ou minimiser le score (en fonction du joueur dont c'est le tour de jouer), deux fonctions get_children et evaluate permettant de respectivement trouver les noueds enfants, et évaluer l'état, et enfin une profondeur maximale à explorer.

- renvoyant en sortie le score de l'état (qui est en fonction du joueur dont c'est le tour, le max ou le min des scores des nœuds enfants / états accessibles)

```
def minimax(state, maximize, get_children, evaluate, max_depth):
    #calculer le score ici à l'aide de get_children(state) et evaluate(state)
    return score
```

> Note : en python les fonctions sont des objets comme les autres. C'est pourquoi une fonction peut être donnée en argument d'entrée au même titre qu'un nombre ou un objet..
>
> Indication : la fonction minimax est implémentable en moins de 20 lignes de code python. Pendant qu'un de vous développe minimax.py, l'autre pourra passer directement à la question **1.3**

Une fonction de test pour minimax existe déjà dans le dossier scripts de votre package.
2) Testez votre implementation de minimax avec la commande :

```
python -m IN104_PROJECT_NOM1_NOM2.scripts.test_minimax
```

Si le test crash, c'est probablement qu'il y a une erreur dans votre fonction.

3) Completer la méthode *play* de la classe *MinimaxBrain* du module **minimaxBrain.py** de votre package. Vous devrez appliquer minimax à chaque état accessibles depuis l'état de jeu actuel et renvoyer le coup correspondant à celui avec le meilleur score. Vous ne considérerez pas la durée de temps timeLimit pour le moment.

> Note1: Vous pouvez pour l'instant ignorer les arguments de la methode __init__ de MinimaxBrain. Ils ne

vous seront utiles que par la suite.

Note2: pour appeler minimax, vous utiliserez les fonctions self.evaluate et self.get_children de votre objet MinimaxBrain. Ces fonctions sont définies dans le constructeur __init__ MinimaxBrain en fonction du type de jeu sur lequel va être utilisé l'IA.

2) Une IA (intelligente cette fois-ci) pour jeu de dames

Il ne vous reste plus grand-chose à faire pour obtenir une IA qui fasse un peu meux que de jouer au hasard !

4) Implémentez dans le module **evaluation_functions/checkers.py** une fonction *evaluate* prenant en entrée un objet GameState et renvoyant un score (selon la convention mentionnée plus haut). Vous pourrez vous contenter dans un premier temps de simplement renvoyer la différence du nombre de pièce entre joueur blanc et noir. En effet, **votre IA devra considérer qu'elle est le joueur Blanc** (le plateau est inversé par Game lorsqu'il est présenté au joueur noir). Vous pourrez vous aider pour de la section Checkers du wiki.

5) En suivant les indications données sur le wiki de aiarena, completez **scripts/human_vs_ai.py** afin de lancer une partie de dames entre vous et votre IA.

6) De même, créez **scripts/ai_vs_ai.py** afin de lancer une partie de dames entre deux instances de votre IA. Vous pourrez par exemple observer l'influence de la profondeur d'exploration de l'arbre sur le résultat d'une partie.

Vous pouvez modifier la profondeur de l'arbre de votre IA apres l'avoir instantiée :
mon_ia = MinimaxBrain(aiarena.checkers)
mon_ia.depth = 7

3) Adaptation pour le puissance4

7) Implémentez dans le module **evaluation_functions/connect4.py** une fonction *evaluate* prenant en entrée un objet GameState et renvoyant un score (selon la convention mentionnée plus haut). On rappelle que **votre IA devra considérer qu'elle est le joueur Blanc.** Vous pourrez vous aider pour de la section Puissance4 du wiki.

8) Adaptez vos scripts **human_vs_ai.py** et **ai_vs_ai.py** afin de pouvoir le donner en argument le type de jeu sur lequel les executer.

Par exemple :

```
python -m IN104_PROJECT_NOM1_NOM2.scripts.human_vs_ai connect4
```

devra lancer une partie de puissance4 entre vous et votre IA.

Afin de récuperer les arguments donnés a votre script dans le terminal, vous pourrez utiliser le package python argparse.