TP D'INFORMATIQUE N°7
IA pour Puissance 4

L'objectif de ce TP est de programmer un jeu de puissance 4 permettant, en entrant les coups dans la console, à deux joueurs humains de s'affronter, puis à un joueur humain d'affronter une IA.

1 Implémentation du jeu de Puissance 4

La base du jeu est donnée dans le fichier `puissance4.py` disponible sur cahier de prepa. La fonction `PvP` permet de lancer une partie entre deux joueurs humains. On remarquera qu'une configuration du jeu est représentée à la fois par une grille de dimensions $n \times m$, dont les cases valent 0 (case inoccupée), 1 (occupée par le joueur 1) ou 2 (occupée par le joueur 2), et (pour une raison d'efficacité) par une liste `hauteur` donnant pour chaque colonne le nombre de coups déjà joués dans cette colonne.

1. Quelle est la complexité de la fonction `jouer` ? Quelle serait sa complexité si on ne prenait pas `hauteur` en argument ?
2. Compléter la fonction `victoire` pour qu'elle renvoie `True` si le joueur numéro `p` a gagné sur `grille`, et `False` sinon.

2 Implémentation de l'algorithme min-max

On cherche à présent à écrire une fonction calculant automatiquement un coup pertinent dans une configuration donnée, en se basant sur l'algorithme Min-max.
Lors du parcours de l'arbre des coups possibles, pour obtenir la configuration fille d'une configuration donnée, on jouera le coup considéré avec la fonction `jouer`. Il faudra alors effacer ce coup une fois que le score de la configuration fille aura été calculé.

1. Écrire une fonction `enlever_coup` prenant en argument `grille`, `hauteur` et un entier `k`, et modifiant `grille` et `hauteur` pour enlever le dernier coup joué sur la colonne `k` (on suppose qu'il y a au moins un coup sur cette colonne). on pourra s'inspirer de la fonction `jouer`.
2. Il faut également disposer d'une heuristique évaluant une configuration. Écrire une fonction `eval` prenant en argument `grille`, `hauteur`, et un numéro `p` de joueur (1 ou 2), et renvoyant le score de cette configuration pour le joueur `p`. On calculera le score en sommant les quantités suivantes :
   - chaque série de 4 cases consécutives (sur une ligne, une colonne ou une diagonale) ne contenant aucun coup de `p` vaut $-x/(4-x)$, où x est le nombre de coups de l'adversaire dans cette série ;
   - symétriquement, chaque série de 4 cases consécutives ne contenant aucun coup de l'adversaire de `p` vaut $x/(4-x)$, où x est le nombre de coups de `p` dans cette série
3. Écrire une fonction `IA` prenant en argument `grille`, `hauteur`, `p` et une profondeur `prof`, et renvoyant un coup pour le joueur `p` en appliquant l'algorithme Min-max pour la profondeur `prof`.
4. Écrire une fonction `PvIA`, prenant en argument la profondeur à utiliser et permettant à un joueur humain d'affronter l'IA de la fonction précédente. On pourra s'inspirer de la fonction `PvP`.
5. Optimiser la fonction `PvIA` en mémoïsant les scores des configurations explorées.
6. Écrire une fonction `IAvIA` permettant à deux IA de s'affronter. Observer le résultat de parties entre IA utilisant des profondeurs différentes.
7. Déterminer une meilleure heuristique à utiliser dans l'IA.