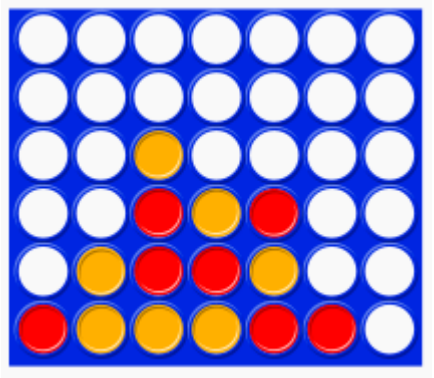
TP n=°7 : Heuristique

Informatique tronc commun 2ère année

1 Puissance 4

Le puissance 4 est un jeu d'alignement de pions. La situation initiale est une grille vide de 6 lignes par 7 colonnes. À son tour, un joueur place un jeton de sa couleur (rouge ou jaune) dans une colonne. Ce jeton tombe (par gravité) dans la première case non remplie de la colonne. Un joueur gagne la partie s'il aligne 4 jetons de sa couleur horizontalement, verticalement ou diagonalement.



Un plateau de jeu va être représenté par une liste de listes d'entiers.
Les cases vides seront représentées par un 0 et les cases pleines, par un 1 ou un 2 en fonction de la couleur du jeton.

```
[[0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0],
 [0, 0, 1, 0, 0, 0, 0],
 [0, 0, 2, 1, 2, 0, 0],
 [0, 1, 2, 2, 1, 0, 0],
 [2, 1, 1, 1, 2, 2, 0]]
```

1.1 Programmation informatique

Le nombre de positions étant très grand, on va trouver une solution acceptable afin de pouvoir de calculer un coup jouable. On va pour cela utiliser une heuristique et n'explorer les coups qu'à une profondeur max.

1) Écrire une fonction `coup_possible(plateau)` qui prend en argument une partie et renvoie une liste de toutes les colonnes dans laquelle on peut jouer.
Sur l'exemple, la fonction doit renvoyer [0,1,2,3,4,5,6]. Si le plateau est `plateau_rempl`, alors la fonction renverrait [0,1,3,4,5,6].

2) Écrire une fonction `jouer_coup(plateau,col,joueur)` qui prend en entrée `plateau` représentant la partie actuelle, `col`, un entier donnant la colonne dans laquelle on souhaite jouer et `joueur`, un entier qui est 1 ou 2 représentant la couleur du joueur qui joue. La fonction fait toutes les modifications sur la liste. La fonction ne renvoie pas une nouvelle liste, mais modifie la liste donnée en entrée.

3) Écrire de même une fonction `annuler_coup(plateau,col)` qui annule un coup joué.
On ne vérifiera pas que le jeton enlevé correspond au bon joueur.

Dans le fichier de TP est écrit une fonction `verif_gagnant(plateau,joueur)` qui permet de vérifier si le plateau contient une combinaison gagnante pour le joueur donné en entrée `joueur`. On pourra l'utiliser dans la suite du TP pour coder les différentes fonctions.

1.2 Stratégie

On cherche à créer des stratégies pour le puissance 4, la première stratégie qu'on va coder et celle de jouer au hasard :
Une fonction de stratégie sera une fonction `strat(plateau,joueur)` qui renvoie un indice de coup qu'il est possible de jouer. Elle permettra de définir une intelligence permettant au joueur de s'affronter.

4) Importer la fonction randrange du module random. Cette fonction prend deux entiers et `randrange(a,b)` renvoie un entier dans l'intervalle [a,b-1].
Écrire alors une fonction `strategie(plateau,joueur)` qui prend en entrée un plateau, et renvoie une colonne où il est possible de jouer. Cette colonne est choisi au hasard.

5) Dans le fichier est écrit la fonction `comparer(strat1,strat2)` qui prend en argument deux stratégies, simule une partie de puissance 4 et renvoie le numéro du gagnant (ou 0 si c'est un match nul).

Tester plusieurs parties où la fonction `strategie` précédente joue contre elle même.

Pour mettre en place une stratégie Min-Max, on a besoin d'une heuristique pour évaluer une grille.

On propose dans un premier temps l'heuristique suivante, qui consiste à attribuer un score à chaque pion posé, positivement pour le joueur 1 et négativement pour le joueur 2, selon la répartition ci-contre :

```
[[3, 4, 5, 7, 5, 4, 3],
 [4, 6, 8, 10, 8, 6, 4],
 [5, 8, 11, 13, 11, 8, 5],
 [5, 8, 11, 13, 11, 8, 5],
 [4, 6, 8, 10, 8, 6, 4],
 [3, 4, 5, 7, 5, 4, 3]]
```

Ce qui motive cette heuristique, c'est le nombre d'alignements possibles qu'une case peut faire partie. Un pion que je possède au milieu de la grille va me donner plus de chances de réaliser un alignement qu'un autre pion.

6) Écrire une fonction `heuristique(plateau)` qui calcule l'heuristique associée à une grille de puissance 4.

Une fonction `score_min_max(plateau,h,prof,joueur)` qui prend en argument une fonction de calcul d'heuristique `h`, un entier `prof` correspondant à une profondeur maximale et une partie de puissance 4 `plateau` et renvoie le score d'un plateau selon l'algorithme du Min-Max est écrite dans le fichier du TP.

Si un coup gagnant est détecté pendant l'exploration, le score de la grille obtenue en jouant ce coup sera fixé égal à 1000 (ou -1000 pour le joueur 2).

7) En utilisant la question précédente, écrire une fonction `coup_min_max(plateau,h,prof,joueur)` qui donne le coup à jouer si l'on suit l'algorithme min_max.

8) Écrire une fonction `strat(plateau,joueur)` qui va appelé la fonction `coup_min_max` avec une profondeur donnée à l'avance et h l'heuristique de la question 9). Tester cette stratégie avec une profondeur de 2,3,4.

1.3 Autre stratégie

On propose de travailler avec une autre heuristique : pour chaque bloc de quatre cases alignées, on attribue un score à ce bloc :

— si chaque case contient un pion de la même couleur, on donne le score +1000 ou -1000 selon la couleur ;
— si ce bloc est vide ou contient des pions de couleurs différentes, on donne le score 0 (aucun alignement gagnant ne peut se faire dans ce bloc) ;
— sinon, on attribue le score ±10, ±3 ou ±1 selon qu'il y a 3, 2 ou 1 pion d'une couleur (positif ou négatif selon sa couleur)

9) Écrire une fonction `strat2(plateau,joueur)` qui implémente cette heuristique avec min_max.

2 Jouer contre l'ordinateur

Pour finir, on souhaite implémenter un moyen de faire jouer l'utilisateur. Pour cela, on va implémenter une "stratégie" qui demande quel coup jouer à l'utilisateur (qui devra alors taper son choix dans la console)

10) Écrire une fonction `humain(jeu)` qui demande à l'utilisateur quel coup jouer (et le renvoie). On pourra utiliser la fonction input qui lit une valeur tapée par l'utilisateur dans la console, et la revoie sous la forme d'une chaîne de caractères. Attention à bien la reconvertir en entier