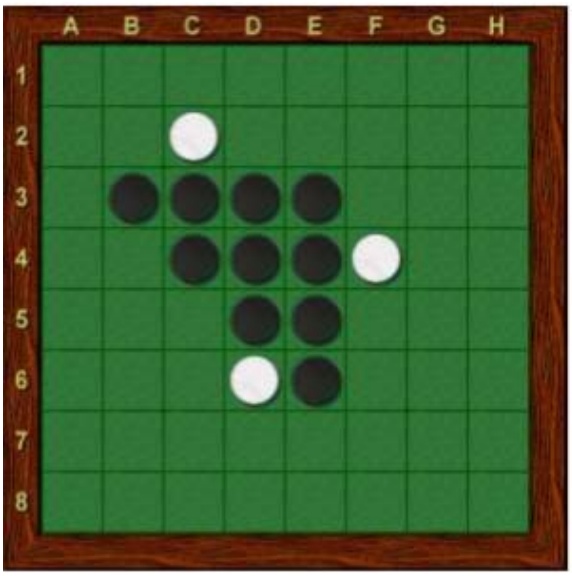
Othello : Console 2

Le jeu



Nous partons de la première version du jeu, et nous allons créer des fonctions qui vont faire jouer l'ordinateur contre lui-même.

Analyse des fonctions à réaliser

Plusieurs stratégies peuvent être mise en œuvre :

1. Dans le jeu, la conquête des coins est importante. Si c'est possible, la fonction sélectionnera un coin, sinon elle recherchera la case qui permet de retourner le maximum de pions adverses.
2. Est-ce que les bords, en dehors des coins sont importants ? Nous pourrons écrire une deuxième fonction qui si c'est possible, jouera en priorité un coin, sinon si c'est possible jouera une case sur un bord, sinon jouera la case qui permet de retourner le maximum de pions adverses.
3. Les livres de stratégie pour Othéllo disent aussi qu'en début de partie, on a intérêt à retourner le moins de pions possible, de façon à garder ses pions groupés au centre de l'othelier. Nous pouvons donc pendant les 3 premiers coups de celui qui applique cette stratégie, jouer les cases qui retournent le moins de pions adverses, puis ensuite jouer la stratégie mis en œuvre en 1.

Lorsque ces fonctions d'intelligence artificielle seront écrites nous ferons jouer un grand nombre de parties, d'une fonction contre une autre.

Nous afficherons le nombre de parties gagnées par l'une ou l'autre des IA ainsi que le pourcentage de gain.

Nous verrons aussi si le fait de commencer ou non apporte un avantage.

Modifier la boucle de jeux

Dans cette version JOUEUR représente l'ordinateur 1 : c'est toujours lui qui commence.

ORDINATEUR représente l'ordinateur 2.

```
#============================================================
NB_JEUX = 500
ordi1 = ordi2 = nul = 0
for i in range(NB_JEUX):
    init()
    while not fin:
        #y a t'il encore des coups possibles
        coupsPossiblesJoueur=trouverCoupsValides(plateau,JOUEUR)
        coupsPossiblesOrdinateur=trouverCoupsValides(plateau,ORDINATEUR)
        if coupsPossiblesJoueur==[] and coupsPossiblesOrdinateur == []:
            fin = True
            break
        if tour == JOUEUR:
            #l'ordinateur 1 a t'il encore des cases à jouer :
             #si oui il joue, sinon il passe son tour
            if coupsPossiblesJoueur!=[]:
                choix = coup_OrdinateurIA1(plateau, JOUEUR)
                #jouer le coup de l'ordinateur1
                jouer_Coup(plateau, JOUEUR, choix[0], choix[1])
            tour = ORDINATEUR
        else:
            #l'ordinateur 2 a t'il encore des cases à jouer
            if coupsPossiblesOrdinateur!=[]:
                choix = coup_OrdinateurIA1(plateau, ORDINATEUR)
                #jouer le coup de l'ordinateur2
                jouer_Coup(plateau, ORDINATEUR, choix[0], choix[1])
```

```
            tour = JOUEUR
    #fin de partie : modifier les scores
    scores = calculer_score(plateau)
    if scores['Joueur'] > scores['Ordinateur']:
        ordi1+=1
    elif scores['Joueur'] < scores['Ordinateur']:
        ordi2+=1
    else:
        nul+=1
#Les 500 parties sont jouées : calcul et affichage du résultat
print('Ordi 1: %s (%s%%)' % (ordi1, round(ordi1 / NB_JEUX * 100, 1)))
print('Ordi 2: %s (%s%%)' % (ordi2, round(ordi2 / NB_JEUX * 100, 1)))
print('Nul:   %s (%s%%)' % (nul, round(nul / NB_JEUX * 100, 1)))
```

Tout est en place pour faire jouer différentes fonctions IA, l'une contre l'autre. Ici, les deux joueurs jouent avec la même IA.

Première IA

a. Jouer les coins
b. Jouer la case qui retourne le plus de pions adverses.

```
#Recherche la case à jouer : si un coin est possible, on le prend
#sinon, on recherche la case qui retourne le plus de pions adverses
def coup_OrdinateurIA1(plateau, ordinateur):
    coupPossibles = trouverCoupsValides(plateau, ordinateur)
    #On mélange les coups possibles
    random.shuffle(coupPossibles)
    # Prendre un coin si possible
    for x, y in coupPossibles:
        if (x == 1 or x == NB_LIGNE) and (y == 1 or y == NB_COLONNE):
            return [x, y]
```

```
    #cherche à retourner le plus de pions possibles
    meilleurScore = -1
    for x, y in coupPossibles:
        plateau2 = np.copy(plateau)
        jouer_Coup(plateau2, ordinateur, x, y)
        scores = calculer_score(plateau2)
        if ordinateur == 1 :
            scoreOrdi = scores['Joueur']
        else:
            scoreOrdi = scores['Ordinateur']
        if scoreOrdi > meilleurScore:
            meilleurCoup = [x, y]
            meilleurScore = scoreOrdi
    return meilleurCoup
```

Faisons jouer nos 500 parties : IA1 contre IA1

Résultat : Premier essai

Ordi 1 : 239 (47.8%)

Ordi 2 : 241 (48.2%)

Nul: 20 (4.0%)

Deuxième essai :

Ordi 1 : 235 (47.0%)

Ordi 2 : 236 (47.2%)

Nul: 29 (5.8%)

Le résultat est conforme à ce que l'on pouvait attendre. Les joueurs se neutralisent.

Celui qui commence, à un petit désavantage.

Deuxième IA

a. Jouer les coins en priorité, si possible
b. Jouer une case d'un bord si possible
c. Jouer la case qui retourne le plus de pions adverses.

```
#Recherche la case à jouer : si un coin est possible, on le prend
#sinon on recherche un bord
#sinon, on recherche la case qui retourne le plus de pions adverses
def coup_OrdinateurAI2(plateau, ordinateur):
    coupPossibles = trouverCoupsValides(plateau, ordinateur)
    random.shuffle(coupPossibles)
     # Prendre un coin si possible
    for x, y in coupPossibles:
        if (x == 1 or x == NB_LIGNE) and (y == 1 or y == NB_COLONNE):
            return [x, y]
    #cherche une case sur un bord si possible
    for x, y in coupPossibles:
        if (x > 1 and x < NB_LIGNE and y == 1) or (x == 1 and y >1 and y <
NB_COLONNE):
            return [x, y]
        if (x > 1 and x < NB_LIGNE and y == NB_COLONNE) or (x == NB_LIGNE and y
>1 and y < NB_COLONNE):
            return [x, y]
    #cherche à retourner le plus de pions possibles
    meilleurScore = -1
    for x, y in coupPossibles:
        plateau2 = np.copy(plateau)
        jouer_Coup(plateau2, ordinateur, x, y)
        scores = calculer_score(plateau2)
        if ordinateur == 1 :
            scoreOrdi = scores['Joueur']
        else:
            scoreOrdi = scores['Ordinateur']
        if scoreOrdi > meilleurScore:
            meilleurCoup = [x, y]
```

```
            meilleurScore = scoreOrdi
    return meilleurCoup
```

Jouons IA2 contre IA2 : il suffit de changer le nom de la fonction dans

choix = coup_OrdinateurIA2(plateau, JOUEUR)

choix = coup_OrdinateurIA2(plateau, ORDINATEUR)

Résultat : Premier essai

Ordi 1: 209 (41.8%)

Ordi 2: 283 (56.6%)

Nul: 8 (1.6%)

Deuxième essai

Ordi 1: 203 (40.6%)

Ordi 2: 289 (57.8%)

Nul: 8 (1.6%)

Il semble qu'ici, le fait de commencer soit un net désavantage.

Jouons IA1 contre IA2 : IA1 commence.

choix = coup_OrdinateurIA1(plateau, JOUEUR)

choix = coup_OrdinateurIA2(plateau, ORDINATEUR)

Résultat : Premier essai

Ordi 1: 297 (59.4%)

Ordi 2: 187 (37.4%)

Nul: 16 (3.2%)

Deuxième essai

Ordi 1: 293 (58.6%)

Ordi 2: 198 (39.6%)

Nul: 9 (1.8%)

Jouons IA2 contre IA1 : IA2 commence.

choix = coup_OrdinateurIA2(plateau, JOUEUR)

choix = coup_OrdinateurIA1(plateau, ORDINATEUR)

Résultat : Premier essai

Ordi 1: 175 (35.0%)

Ordi 2: 313 (62.6%)

Nul: 12 (2.4%)

Deuxième essai

Ordi 1: 161 (32.2%)

Ordi 2: 331 (66.2%)

Nul: 8 (1.6%)

On peut déduire de ces résultats que le fait de prendre un bord, plutôt que de chercher à retourner le plus de pions adverses n'est pas une bonne stratégie.

La encore, le fait de commencer est un désavantage : le résultat de l'IA1 lorsqu'elle commence est moins bon que lorsqu'elle joue en deuxième.

Troisième IA

a. Pendant les 6 premiers tours (chaque joueur joue 3 fois), le joueur choisit la case qui retourne le moins de pions
b. Ensuite il joue l'IA1.

IA qui joue la case retournant le moins de pions

```
#Recherche la case à jouer:
#on recherche la case qui retourne le moins de pions adverses
def coup_OrdinateurIA4(plateau, ordinateur):
    coupPossibles = trouverCoupsValides(plateau, ordinateur)
    random.shuffle(coupPossibles)
    #Cherche à retourner le moins de pions possible
    mauvaisScore = 64
    for x, y in coupPossibles:
        plateau2 = np.copy(plateau)
        jouer_Coup(plateau2, ordinateur, x, y)
        scores = calculer_score(plateau2)
        if ordinateur == 1 :
            scoreOrdi = scores['Joueur']
```

```
        else:
            scoreOrdi = scores['Ordinateur']
        if scoreOrdi < mauvaisScore:
            mauvaisCoup = [x, y]
            mauvaisScore = scoreOrdi
    return mauvaisCoup
```

IA qui joue momentanément l'IA4, puis l'IA1

```
#Recherche la case à jouer:
#Pendant les 6 premiers coups, on joue la case qui retourne le moins de pions
#sinon, on revient à IA1
def coup_OrdinateurIA3(plateau, ordinateur):
    if nb_tour <6:
        return coup_OrdinateurIA4(plateau, ordinateur)
    return coup_OrdinateurIA1(plateau, ordinateur)
```

Nous devons rajouter dans notre boucle de jeu :

```
for i in range(NB_JEUX):
    init()
    nb_tour=-1
    while not fin:
        nb_tour+=1
        …
```

Jouons IA3 contre IA3

```
choix = coup_OrdinateurIA3(plateau, JOUEUR)
choix = coup_OrdinateurIA3(plateau, ORDINATEUR)
```

Résultat : Premier essai

Ordi 1: 223 (44.6%)

Ordi 2: 254 (50.8%)

Nul: 23 (4.6%)

Deuxième essai

Ordi 1: 220 (44.0%)

Ordi 2: 259 (51.8%)

Nul: 21 (4.2%)

Ici encore, commencer est un désavantage encore plus important que l'orsqu'on joue IA1 contre IA1.

Jouons IA3 contre IA1 : IA3 commence

choix = coup_OrdinateurIA3(plateau, JOUEUR)

choix = coup_OrdinateurIA1(plateau, ORDINATEUR)

Résultat : Premier essai

Ordi 1: 225 (45.0%)

Ordi 2: 255 (51.0%)

Nul: 20 (4.0%)

Deuxième essai

Ordi 1: 251 (50.2%)

Ordi 2: 226 (45.2%)

Nul: 23 (4.6%)

Troisième essai

Ordi 1: 250 (50.0%)

Ordi 2: 237 (47.4%)

Nul: 13 (2.6%)

Jouons IA1 contre IA3 : IA1 commence

choix = coup_OrdinateurIA1(plateau, JOUEUR)

choix = coup_OrdinateurIA3(plateau, ORDINATEUR)

Résultat : Premier essai

Ordi 1: 205 (41.0%)

Ordi 2: 283 (56.6%)

Nul: 12 (2.4%)

Deuxième essai

Ordi 1: 227 (45.4%)

Ordi 2: 256 (51.2%)

Nul: 17 (3.4%)

Troisième essai

Ordi 1: 198 (39.6%)

Ordi 2: 288 (57.6%)

Nul: 14 (2.8%)

Nous pouvons déduire de ces résultats, que commencer est toujours un désavantage.

Il semble aussi que cette troisième IA est plus performante que la première.