Informatique - DM n°1 - Pour le dimanche 4 mai, par mail

Jeu Hex

I. Règles du jeu

Le jeu Hex se joue sur un échiquier à cases hexagonales à n lignes, n colonnes.
À tour de rôle, chaque joueur (Bleu et Rouge) joue un pion sur l'une des cases libres (les cases n'ont donc que 3 valeurs possibles : vide, bleue ou rouge).
Bleu joue en premier.
Le premier joueur qui réussit à joindre ses deux camps (haut et bas de l'échiquier pour bleu, gauche et droite de l'échiquier pour rouge) a gagné.
On peut prouver qu'il n'y a jamais de partie nulle et que, si Bleu joue parfaitement, il gagne toujours.

II. Travail demandé

Vous devrez envoyer un fichier python ne contenant que des fonctions et des déclarations de variables, dont une fonction au moins aura pour en-tête :

```
def algo(grille, joueur):
```

C'est cette fonction qui sera appelée lors du combat entre les algorithmes.
Le paramètre `grille` est un tableau `numpy` de taille $n \times n$ et contenant des 0 (cases vides), des 1 (cases bleues) et des 2 (cases rouges).
Le paramètre `joueur` donne le numéro du joueur que vous jouez (1 pour bleu, 2 pour rouge).
La fonction doit renvoyer un couple (i, j) donnant les indices de ligne et de colonne de la case vide sur laquelle voue allez jouer.

III. Un exemple simple

Programmer une fonction qui renvoie aléatoirement un couple (i, j) d'une case vide.

IV. Un exemple plus raffiné

Voici un exemple possible d'algorithme (assez naïf) :

```
import numpy as np

voisins = [(-1,0), (-1,1), (0,1), (1,0), (1,-1), (0,-1)]

def nom(i,j,g):
    if g == 0:
        name = (i,j)
    elif g == 1:
```

```
        name = "B"+str((i,j))
    elif g == 2:
        name = "R"+str((i,j))
    return name

def construitLA(grille):
    LA = {"B0":["R0", "R1"],"B1":["R0", "R1"],"R0":["B0", "B1"],"R1":["B0", "B1"]}
    N = len(grille)
    for i in range(N):
        for j in range(N):
            name = nom(i,j,grille[i,j])
            LA[name] = []
            if i == 0:
                LA["B0"].append(name)
                LA[name].append("B0")
            if i == N-1:
                LA["B1"].append(name)
                LA[name].append("B1")
            if j == 0:
                LA["R0"].append(name)
                LA[name].append("R0")
            if j == N-1:
                LA["R1"].append(name)
                LA[name].append("R1")
            for di,dj in voisins :
                if di>0 or (dj>=0 and di>=0):
                    continue
                I = i+di
                J = j+dj
                if 0<=I<N and 0<=J<N:
                    nam2 = nom(I,J,grille[I,J])
                    LA[name].append(nam2)
                    LA[nam2].append(name)
    return N, LA

def fusion(LA,n1,n2):
    for v in LA[n2]:
        if v not in LA[n1] and v!=n1:
            LA[n1].append(v)
            LA[v] = [vv for vv in LA[v] if vv!=n2]
            LA[v].append(n1)
        else:
            LA[v] = [vv for vv in LA[v] if vv!=n2]
    LA[n1] = [vv for vv in LA[n1] if vv!=n2]
```

```
    del LA[n2]
    # print(LA)
    return LA

def fusionne(LA):
    cont = True
    while cont:
        for v in LA.keys():
            if (v not in ["B0", "B1", "R0", "R1"]) and (v[0]=="R" or v[0]=="B"):
                for vv in LA[v]:
                    if vv[0]==v[0]:
                        if vv not in ["B0", "B1", "R0", "R1"]:
                            LA = fusion(LA,v,vv)
                        else:
                            LA = fusion(LA,vv,v)
                        break
                else:
                    continue
                break
        else:
            cont = False
    return LA


def algo(grille, joueur):
    N, LA = construitLA(grille)
    LA = fusionne(LA)
    # print(LA)
    if joueur==1:
        cases = [(r[0],len(LA[r]),r) for r in LA["B0"] if isinstance(r,tuple)]
    else:
        cases = [(r[1],len(LA[r]),r) for r in LA["R0"] if isinstance(r,tuple)]
    cases.sort(reverse=True)
    return cases[0][2]
```