ISM MP, Informatique
Année 2024/2025

Corrigé du TP Informatique 14

Exercice 1

1. On saisit :

```
def minimax(tas,J_i):
    if tas==1:
        return score(J_i)
    else:
        aux=[]
        for k in range(1,4):
            reste=tas-k
            if reste>0:
                aux.append(minimax(reste,adversaire(J_i)))
        if J_i==0:
            return max(aux)
        else:
            return min(aux)
```

2. On saisit :

```
def nim_minimax(N):
    tas,fini,J_i=N,N==1,1
    while not fini:
        J_i=adversaire(J_i)
        print("J_"+str(J_i)+"=",tas)
        if J_i==0:
            valide=False
            while not valide:
                retrait=int(input("Retire="))
                valide=coup_valide(tas,retrait)
                if not valide:
                    print("Coup invalide")
        else:
            m,retrait=float('inf'),None
            for i in range(1,4):
                if tas-i>0:
                    aux=minimax(tas-i,0)
                    if aux<=m:
                        m,retrait=aux,i
            print("Retire= ",retrait)
        tas-=retrait
        fini=(tas==1)
    print("J_"+str(J_i)+" gagne")
```

Exercice 2

On saisit :

```
def minimax(tab,libres,J_i):
    align_o=alignement(tab,0)
    align_x=alignement(tab,1)
    # si feuille
    if align_o or align_x or len(libres)==0:
        if align_o:
            return pinf
        elif align_x:
            return minf
        else:
            return 0
    else:
        aux=[]
        for (i,j) in libres:
            tab_aux=deepcopy(tab)
            libres_aux=deepcopy(libres)
            tab_aux[i][j]=J_i
            libres_aux.remove((i,j))
            aux.append(minimax(tab_aux,libres_aux,adversaire(J_i)))
            if aux[-1]*(1-2*J_i)==pinf:
                return aux[-1]
        if J_i==0:
            return max(aux)
        else:
            return min(aux)
```

2. On saisit :

```
def morpion_minimax():
    tab=[[None]*3 for k in range(3)]
    libres=[(i,j) for i in range(3) for j in range(3)]
    nb,fini,J_i=0,False,1
    print("Partie de morpion")
    print("J_0 -> O")
    print("J_1 -> X")
    print()
    while not fini:
        J_i=adversaire(J_i)
        print('Coup=',nb)
        nb+=1
        aff(tab)
        if J_i==0:
            [...]
```

```
        else:
            m=pinf
            for (x,y) in libres:
                if m!=minf:
                    tab_aux=deepcopy(tab)
                    libres_aux=deepcopy(libres)
                    tab_aux[x][y]=1
                    libres_aux.remove((x,y))
                    aux=minimax(tab_aux,libres_aux,0)
                    if aux<=m:
                        m=aux
                        i,j=x,y
        tab[i][j]=J_i
        libres.remove((i,j))
        align_o=alignement(tab,0)
        align_x=alignement(tab,1)
        fini=align_o or align_x or nb==9
        print()
    if align_o or align_x:
        print("J_"+str(J_i)+" gagne")
    else:
        print("Match nul")
    aff(tab)
```

Exercice 3

1. On saisit :

```
def minimax_depth(tab,J_i,p):
    # recherche successeurs de tab pour J_i
    succ={}
    for pos in cases_libres(tab):
        L_retourne=retourne(tab,pos,J_i) # cases retournables pour J_i
        if L_retourne!=[]:
            succ[pos]=L_retourne
    if succ=={}: # si feuille de l'arbre
        return score(tab)
    elif p==0:
        return H(tab,poids)
    else:
        aux=[]
        for pos in succ:
            tab_aux=deepcopy(tab)
            joue(tab_aux,pos,J_i)
            aux.append(minimax_depth(tab_aux,adversaire(J_i),p-1))
        if J_i==0:
            return max(aux)
        else:
            return min(aux)
```

2. On saisit :

```
def othello_minimax_depth(p):
    tab=init_tab()
    J_i=1
    fini=False
    print("Partie d'Othello")
    print("J_0 -> O")
    print("J_1 -> X")
    print()
    while not fini:
        J_i=adversaire(J_i)
        aff(tab)
        if len(cases_jouables(tab,J_i))>0:
            if J_i==0:
                [...]
            else:
                print()
                m=pinf
                # construction liste cases jouables  pour J_i
                jouables=cases_jouables(tab,J_i)
                for pos_aux in jouables:
                    tab_aux=deepcopy(tab)
                    joue(tab_aux,pos_aux,J_i)
                    aux=minimax_depth(tab_aux,adversaire(J_i),p)
                    if aux<=m:
                        m=aux
                        pos=pos_aux
                print("\nLa machine joue ",pos)
            joue(tab,pos,J_i)
        elif len(cases_libres(tab))>0:
            print("\nJ_"+str(J_i)+" passe")
        J0_bloque=len(cases_jouables(tab,0))==0
        J1_bloque=len(cases_jouables(tab,1))==0
        fini=J0_bloque and J1_bloque
        print()
    n0,n1=comptage(tab)
    [...]
```

Exercice 4

On saisit :

```
def minimax_depth(tab,col,J_i,p):
    place(tab,col,J_i)
    if gagne(tab,col,J_i):
        return score(J_i)
    elif grille_pleine(tab):
        return 0
    elif p==0:
        return H(tab,poids)
    else:
        aux=[]
        for c in range(7):
            if coup_valide(tab,c):
                tab_aux=deepcopy(tab)
                aux.append(minimax_depth(tab_aux,c,adversaire(J_i),p-1))
        if J_i==0:
            return min(aux)
        else:
            return max(aux)
```

2. On saisit :

```
def puiss4_minimax(p):
    tab=init_tab()
    J_i=1
    fini=False
    print("Partie de puissance 4")
    print("J_0 -> O")
    print("J_1 -> X")
    while not fini:
        J_i=adversaire(J_i)
        aff(tab)
        print()
        if J_i==0:
            [...]
        else:
            m=pinf
            for col_aux in range(7):
                if coup_valide(tab,col_aux):
                    tab_aux=deepcopy(tab)
                    aux=minimax_depth(tab_aux,col_aux,J_i,p)
                    if aux<=m:
                        m=aux
                        col=col_aux
            print("La machine joue : col =",col)
        place(tab,col,J_i)
        fini=grille_pleine(tab) or gagne(tab,col,J_i)
```

```
    if gagne(tab,col,J_i):
        print("J_"+str(J_i)+" gagne")
    else:
        print("Match nul")
    aff(tab)
```