TD Informatique TSI2

Algorithms of games

Solution

EXERCICE 01

Graphe biparti ou non

Q1. A FAIRE

Q2 On colore les sommets $1, 2, 6, 8$ et 9 en gris. On voit que pour deux sommets reliés par une arête un est grisé et l'autre reste blanc.

Q3. On retape le code.

```
In [1]: from collections import deque
In [2]: def cycle_impair(graphe, sommet):
            niveaux = {s : None for s in graphe}
            niveaux[sommet] = 0
            file = deque()
            file.append(sommet)
            while len(file) > 0 :
                sommet = file.popleft()
                for s in graphe[sommet]:
                    if niveaux[s] is None :
                        niveaux[s] = niveaux[sommet] + 1
                        file.append(s)
                    elif niveaux[s] == niveaux[sommet]:
                        return True
            return False
```

```
In [3] : g= {0: [1,8,9], 1 : [0,3,4,7], 2: [3,5,7], 3 : [1,2,6,9],
4: [1,6,8,9], 5: [2,8,9], 6 : [3,4,7], 7 :[1,2,6,8],
8 : [0,4,5,7], 9 : [0,3,4,5]}
In [4] : cycle_impair(g,0)
Out[4] :  False
```

EXERCICE 02

Jeu de Nim avec un seul tas et quatre objets

Chaque joueur retire un ou deux objets du tas à tour de rôle.
Le dernier joueur qui retire des objets a gagné la partie.

Q1. Définir un graphe `G` sous forme d'un dictionnaire dont chaque clé, représentant un sommet, est un couple composé du numéro du joueur (1 ou 2) qui contrôle le sommet et du nombre d'objets restant dans le tas. Les valeurs associées aux clés sont des listes de couples où chaque couple représente un sommet après la prise du joueur qui contrôle le sommet représenté par la clé.

Le premier joueur est le joueur 1 et il y a 4 objets. Le sommet de départ est donc représenté par $(1, 4)$. Le joueur 1 peut retirer un ou deux objets et les deux sommets atteints sont contrôlés par le joueur 2. Alors `G = (1,4):[(2,3),(2,2)],...`

Réponse : On obtient

$\{(1,4) : [(2,2),(2,3)], (2,3) : [(1,1),(1,2)], (1,2) : [(2,0),(2,1)], (2,1) : [(1,0)], (1,0) : [], (2,0) : [], (1,1) : [(2,0)], (2,2) : [(1,0),(1,1)]\}$

Q2-a On veut retourner par une procédure le dictionnaire `G` demandé à la question **Q1**.
Pour cela, on commence par construire une procédure `nim(n,j,G)` récursive où `n` est le nombre de sommets (donc d'objets), `j` est le numéro d'un joueur (1 ou 2) et `G` le graphe que l'on initialise à `G = { }` ensuite. On remarque si `j` est un joueur alors `1+j%2` est l'autre joueur.
Ainsi dans le corps de la procèdure `nim(n,j,G)` :

```
# Pour la condition  n>1 , on tape :
G[(j,n)] = [(1+j%2,n-2),(1+j%2,n-1)]
nim(n-1,1+j%2,G)
nim(n-2,1+j%2,G)
# Pour la condition  n vaut 1 , on tape :
G[(j,n)] = [(1+j%2,n-1)]
nim(n-1,1+j%2,G)
# Sinon, on tape :
G[(j,n)] = []
```

Cela donne :

```
In [1] : def nim(n,j,G):
             if n > 1:
                 G[(j,n)] = [(1+j%2,n-2),(1+j%2,n-1)]
                 nim(n-1,1+j%2,G)
                 nim(n-2,1+j%2,G)
             elif n == 1:
                 G[(j,n)] = [(1+j%2,n-1)]
                 nim(n-1,1+j%2,G)
             else:
                 G[(j,n)] = []
```

Q2-b Tapons ensuite :

```
In [1] : def jeu(n):
            G={}
            nim(n,1,G)
            return G
```

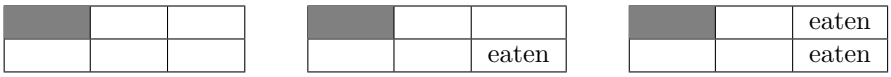
Retrouvons le graphe `G` pour $n = 4$ de la question **1**.

```
In [2] : jeu(4)
Out[2]:
{(1, 4): [(2, 2), (2, 3)],
 (2, 3): [(1, 1), (1, 2)],
 (1, 2): [(2, 0), (2, 1)],
 (2, 1): [(1, 0)],
 (1, 0): [],
 (2, 0): [],
 (1, 1): [(2, 0)],
 (2, 2): [(1, 0), (1, 1)]}
```

EXERCICE 03

Jeu de chomp

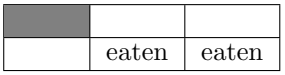
On considère le jeu de chomp avec une tablette composée de 2 lignes et 3 colonnes. Le carré $(1,1)$ est grisé et empoisonné. Les différents états de la tablette sont numérotés.



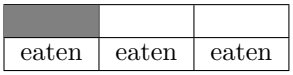
Etat 1

Etat 2

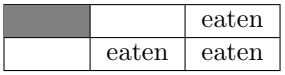
Etat 3



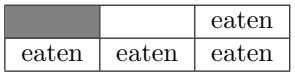
Etat 4



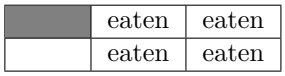
Etat 5



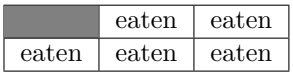
Etat 6



Etat 7



Etat 8



Etat 9

Pour simplifier le graphe et sauver les joueurs, on considère qu'un joueur ne peut pas manger le carré emposonné. Donc l'état 9 est un état final. Le joueur qui se trouve devant cet état a perdu la partie car il ne peut plus jouer.

Q1 Dessiner le graphe du jeu. On pourra faire deux colonnes. Un sommet est un couple avec pour premier élément le numéro du joueur et pour second élément létat de la tablette. Dans la première colonne, on mettra les huits sommets `(1,i)` contrôlés par J_1 les uns sous les autres en commençant par `(1,1)` et dans la deuxième colonne, on mettra les huits sommets `(2,i)` contrôlés par J_2 les uns sous les autres commençant par `(2,2)`. Puis on placera les différentes arêtes entre les deux colonnes.

Remarque : *attention à deux notations identiques pour deux types d'objets. Le sommet `(i,j)` désigne que le joueur numéro i est à l'étape j et le carré (i,j) désigne le carré de la tablette qui est à la ligne i et colonne j.*

Réponse : A FAIRE!

Q2. Alors on a :

$G = \{(1,1) : [(2,2),(2,3),(2,4),(2,5),(2,8)],$
$(1,3) : [(2,6),(2,7),(2,8)],$
$(1,4) : [(2,5),(2,6),(2,8)],$
$(1,5) : [(2,7),(2,9)],$
$(1,6) : [(2,7),(2,8)],$
$(1,7) : [(2,9)],$
$(1,8), [(2,9)],$
$(1,9) : [],$
$(2,2) : [(1,3),(1,4),(1,5),(1,8)],$
$(2,3) : [(1,6),(1,7),(1,8)],$
$(2,4) : [(1,5),(1,6),(1,8)],$
$(2,5) : [(1,7),(1,9)],$
$(2,6) : [(1,7),(1,8)],$
$(2,7) : [(1,9)],$
$(2,8) : [(1,9)],$
$(2,9) : [],$
$\}$

Q3. On suppose que J_1 depuis le sommet `(1,1)` mange le carré $(2,3)$ et atteint donc le sommet `(2,2)`. Au tour de J_2.

- J_2 se déplace en `(1,3)` alors J_1 se déplace en `(2,6)` et J_2 ne peut jouer que `(1,7)` ou `(1,8)` et J_1 conclut avec `(2,9)`.
- J_2 se déplace en `(1,4)` alors comme le cas précédent, J_1 se déplace en `(2,6)` et J_2 ne peut jouer que `(1,7)` ou `(1,8)` et J_1 conclut avec `(2,9)`.
- J_2 se déplace en `(1,5)` alors J_1 se déplace en `(2,9)` et conclut.
- J_2 se déplace en `(1,8)` alors J_1 se déplace en `(2,9)` et conclut.

Dans tous les cas, on a trouvé une stratégie gagnante pour J_1.