TP N°4 : Étude des jeux, Jeu de Nim

Dans le chapitre 5, nous aborderons quelques aspects de théorie des jeux. En voici un exemple de programmation avec le jeu de Nim.

Règles du jeu

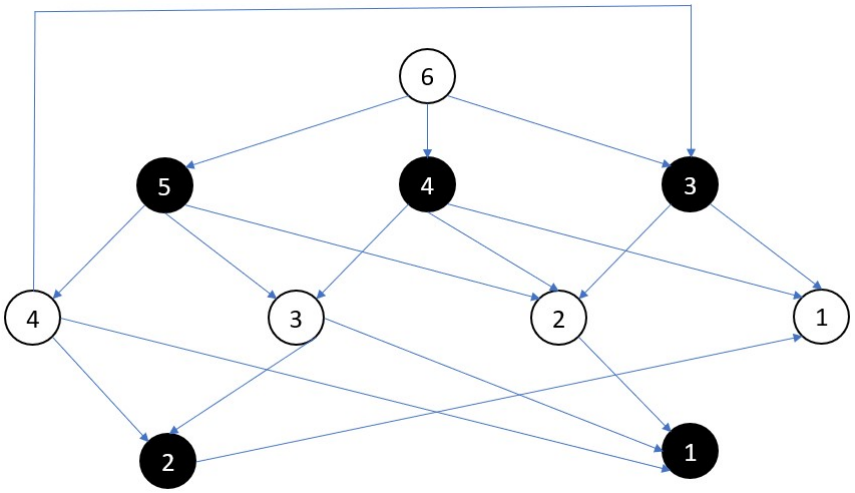
On dispose de N jetons identiques posés sur une table. Deux joueurs jouent tour après tour et on le droit de prendre entre 1 et trois jetons. Le joueur qui vide la table a perdu. Notons J_0 le joueur qui commence la partie, J_1 le second.

Remarque 5.1.

- *s'il reste 1 jeton sur la table, le joueur suivant a perdu, car il est obligé de prendre le jeton ;*
- *s'il reste i jetons avec $i \in \{2,3,4\}$ c'est le joueur suivant qui a gagné, il suffit qu'il prenne $i-1$ jeton(s).*

Représentation

On représente les possibilités d'une partie avec un graphe.



GRAPHE DE LA PARTIE POUR $N = 6$.

Les sommets en blanc représentent les états dans lesquels c'est le joueur J_0 qui joue. On dit qu'ils sont contrôlés par J_0. Les sommets en noir sont les sommets contrôlés par J_1. Les numéros de chaque sommet correspondent au nombre de jetons restants. Seuls le nombre de jetons sur la table et le nom ou l'indice du joueur qui doit jouer (qui contrôle le sommet) interviennent. Plusieurs déroulements de parties peuvent donc conduire à un même état, ce qui explique que nous ayons affaire à un graphe qui n'est pas un arbre.

Modélisation informatique

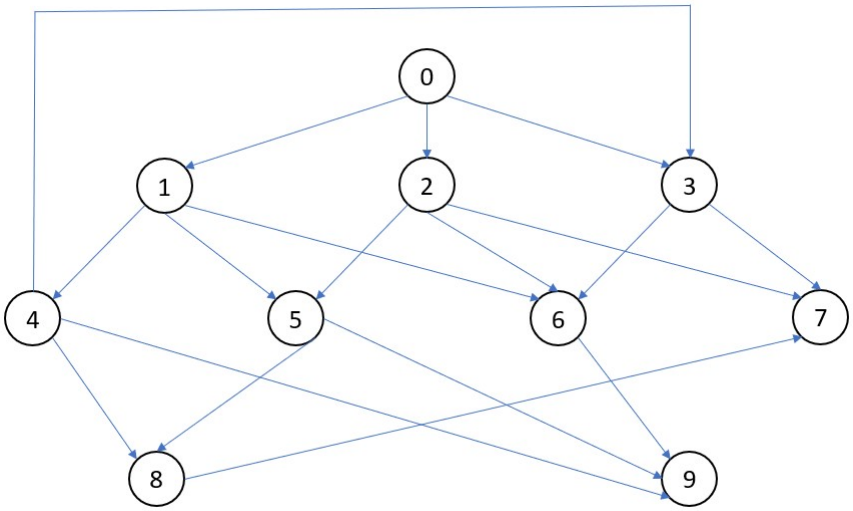
On se propose de construire le graphe des parties de Nim à deux joueurs J_0 et J_1 qui commencent à N jetons de la manière suivante :

- On étiquète chaque sommet de la manière suivante : il y a k jetons restants et c'est J_i qui doit jouer" est étiqueté `"k;i"`, cela jouera le rôle de clé dans le dictionnaire S des sommets. Les valeurs associés seront le numéro des sommets de 0 au nombre de sommets-1 (par ordre d'entrée dans le dictionnaire).
- On crée également la liste d'adjacence `A=[[i,j], ...]` (pour rappel, la liste `[i,j]` est dans A si une arête orientée va de i vers j).

Voici par exemple S et A dans le cas $N = 6$:

```
S={'6;0':0, '5;1':1, '4;1':2, '3;1':3, '4;0':4, '3;0':5, '2;0':6, '1;0':7, '2;1':8, '1;1':9}
A=[[0,1],[0,2],[0,3],[1,4],[1,5],[1,6],[2,5],[2,6],[2,7],[3,6],[3,7],[4,3],[4,8],[4,9],[5,8],[5,9],[6,9],[8,7]]
```

Ce qu'on peut retrouver avec l'illustration suivante ou chaque arête porte son numéro :



GRAPHE DE LA PARTIE POUR $N = 6$ AVEC LES NUMÉROS DES SOMMETS.

Remarque 5.2. *La numérotation des sommets est arbitraire, la matrice A n'est donc unique qu'à numérotation des sommets fixée. Il n'y a donc pas unicité du couple (S, A) pour une partie donnée.*

1 Créer une fonction `voisins('char')` qui prend une chaîne de caractère qui est l'étiquetage d'un sommet, et renvoie l'étiquetage des sommets atteignables depuis ce sommet. Par exemple `'5;0'` doit renvoyer la liste `['4;1','3;1','2;1']`, et `'3;1'` doit renvoyer `['2;0','1;0']`.

2 Créer une fonction `const_voisins(S,A,char,Lvoisins)` qui prend en argument S et A définis au préalable, char l'étiquetage d'un sommet et Lvoisins la liste des voisins (sommets atteints) de char, et qui rajoute (si nécessaire) les sommets atteints par le sommet char dans S et les arrêtes dans A. Par exemple, `'5;0'` vérifie si les trois chaines de la liste `['4;1','3;1','2;1']` (la liste des voisins Lvoisins) sont des clés, les rajoutent sinon, et créer les 3 arrêtes liées à ces sommets et les rajoutent à A. On supposera que `char` est une clé du dictionnaire. On renverra le couple (S,A).

3 Créer la fonction graphe(N) qui renvoie le couple (S,A) pour une partie de N jetons commencée par le joueur J_0. On pourra créer une liste M qui sera constitué des étiquetages des sommets déjà présent dans S et tel que L[S[char]]=char. Il faudra alors parcourir les indices de cette liste et utiliser `const_voisins` pour chacun des étiquetages.

Analyse d'une stratégie

Une des stratégies de ce jeu consiste lorsque le nombre de jetons restants n'est pas de la forme $4k+1$ de renvoyer autant de jetons de nécessaire pour qu'il soit de la forme $4k+1$, et ce jusqu'à la fin de la partie.

4 Créer une fonction `joueurA(char)` qui prend en argument l'étiquetage char et qui renvoie l'étiquetage après que le joueur joue. Si le nombre de jetons est de la forme 4k+1, on prendra un nombre aléatoire de jetons (entre 1 et 3).

5 Créer une fonction `joueurB(char)` qui effectue la stratégie de prendre un nombre aléatoire de jetons entre 1 et 3, sauf s'il reste assez peu de jetons pour gagner.

6 Créer une fonction `partie(Ljoueur,N)` qui prend une liste de fonction Ljoueur telle que L[k] est la fonction correspondant à la stratégie de joueur[k] et N le nombre de jetons. La fonction renvoie l'indice du vainqueur de la partie.

7 Tester et estimer la probabilité de gagner pour le joueur 1 si celui-ci joue la stratégie A et que le joueur 0 joue la stratégie B pour $N = 20$.

Remarque 5.3. *On peut généraliser à plusieurs joueurs, par exemple, en éliminant le joueur qui finit le tas de jetons et on recommence avec N jetons jusqu'à qu'il n'y ait plus que deux joueurs qui jouent. Si vous avez le temps, recommencer les questions avec cette configuration.*