VII

Jeux d'accessibilité à deux joueurs sur un graphe

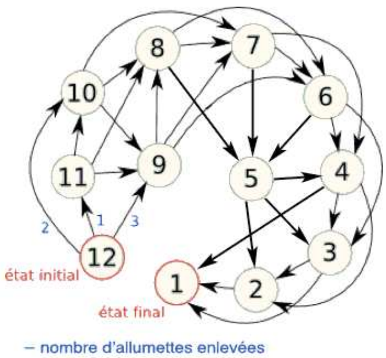


FIGURE VII.1 – Représentation en graphe du jeu de NIM inventé par Charles Leonard BOUTON (Source : interstices.info)

Charles Leonard Bouton (1869-1922) est un mathématicien américain. Originaire de St Louis dans le Missouri, il est titulaire d'un Master of Science de l'université de Washington et va ensuite à Leipzig où il prépare son doctorat sous la direction de Sophus Lie (1898). Il enseigne à l'université de Washington puis à Harvard. De 1900 à 1902 il est éditeur du Bulletin de la Société mathématique américaine. Il est connu pour avoir publié en 1901 une solution complète du jeu de Nim qui est un point de départ de la théorie des jeux combinatoires.

PLAN DU CHAPITRE

I Position du problème

I.1 Jeux à deux joueurs

Le but de ce chapitre est d'aborder les aspects algorithmiques des jeux ; on limitera notre étude aux jeux :

- à deux joueurs
- à alternance, i.e. les joueurs alternent à chaque coup
- à information complète i.e. sont toujours connus :
  - toutes les possibilités de coups du joueur et de ses adversaires (sont donc exclus la plupart des jeux de cartes où les cartes en main d'un joueur sont généralement dissimulées à ses adversaires)
  - les gains résultants des coups
  - les motivations de l'autre joueur
- sans mémoire, i.e. la décision prise par chaque joueur pour jouer un coup dépend exclusivement de l'état actuel du jeu et non des états antérieurs
- à somme nulle, c'est à dire que la somme des gains et des pertes de tous les joueurs vaut 0 ; par conséquent chaque gain pour un joueur constitue une perte pour l'autre.

I.2 Représentation des jeux par un graphe

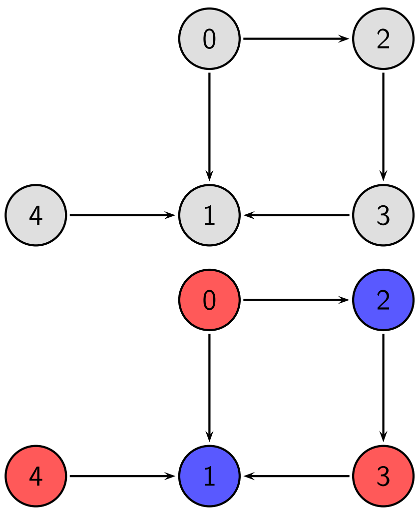
a - Graphes bipartis

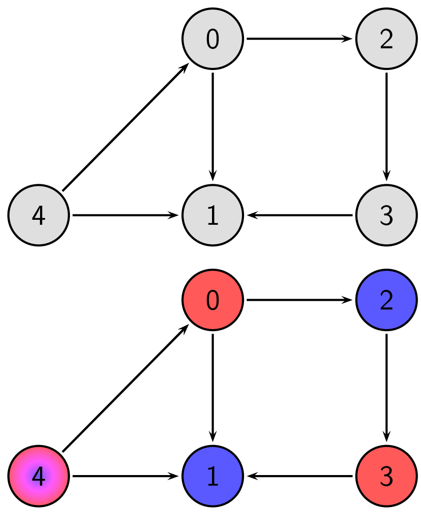
Il est possible de représenter simplement les parties de jeux **à alternance** à l' aide de graphes dits **bipartis**.

Définition I-1: GRAPHE BIPARTI

Un graphe $G = (V, A)$ est **biparti** si l'ensemble de ses sommets V peut être partitionné en deux sous-ensembles V_0 et V_1, c'est à dire $V = V_0 \cup V_1$ et $V_0 \cap V_1 = \varnothing$ tels que tout arc issu d'un sommet de V_0 ne peut pointer que vers un sommet de V_1 et inversement, ou encore chaque arc possède une extrémité dans V_0 et l'autre dans V_1.

Exercice de cours: **(I.2) - n° 1.** *Parmi les deux graphes ci-dessous, lequel est biparti ?*





b - Exemple du jeu de Nim

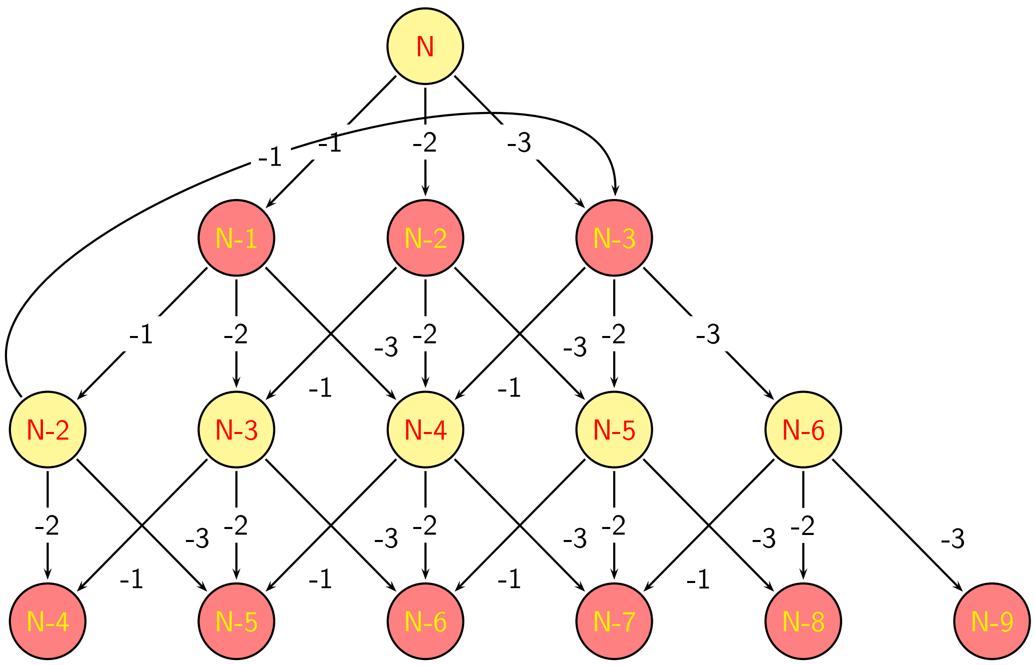
Une variante du jeu de Nim est par exemple de disposer N jetons identiques sur une table :

- Deux joueurs J_0 et J_1 jouent à tour de rôle ;
- chaque joueur prélève lorsque c'est son tour, soit 1, soit 2, soit 3 jetons ;
- le joueur qui retire le dernier jeton à perdu.

On va représenter ce jeu à deux joueurs à alternance par un graphe $G = (V, A) = (V_0, V_1, A)$:

- Les sommets V correspondent aux positions atteignables dans la partie, c'est à dire les différents états du jeu, i.e. **le nombre de jetons restants sur la table** ;
- les arcs A désignent toutes les possibilités de coups depuis un sommet vers un autre .

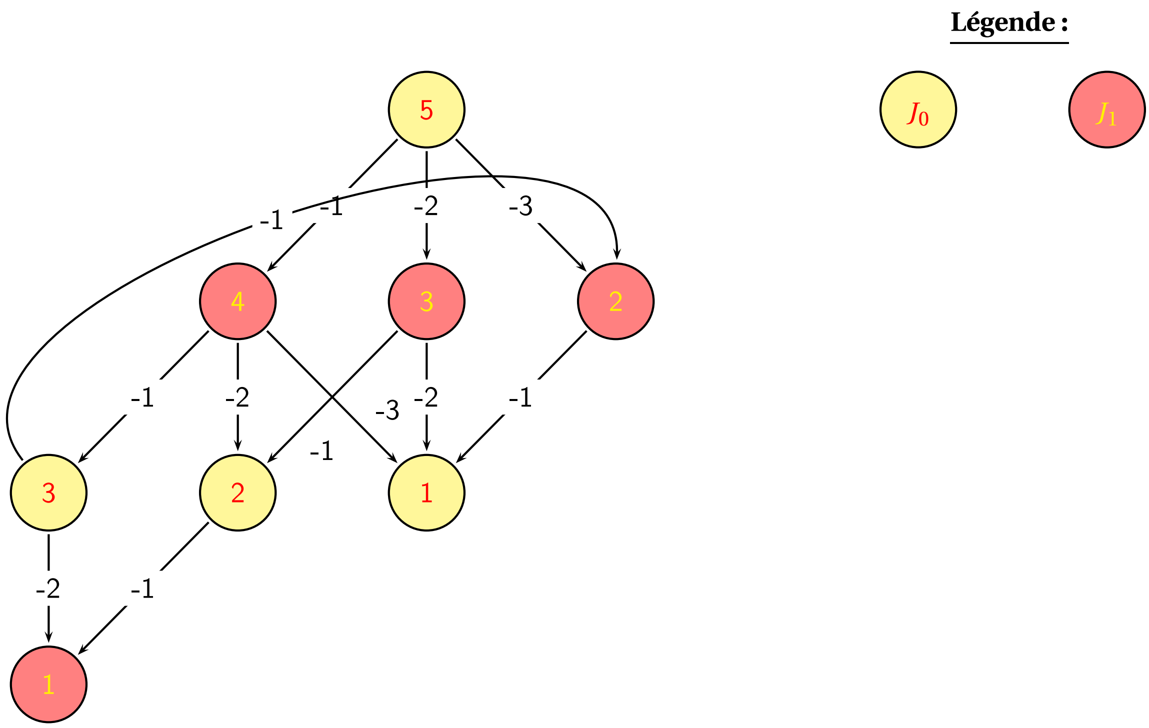
Par exemple le graphe suivant représente une partie partielle pour un nombre initial N de jetons (≥ 10) avec **le joueur J_0 qui joue en premier** :



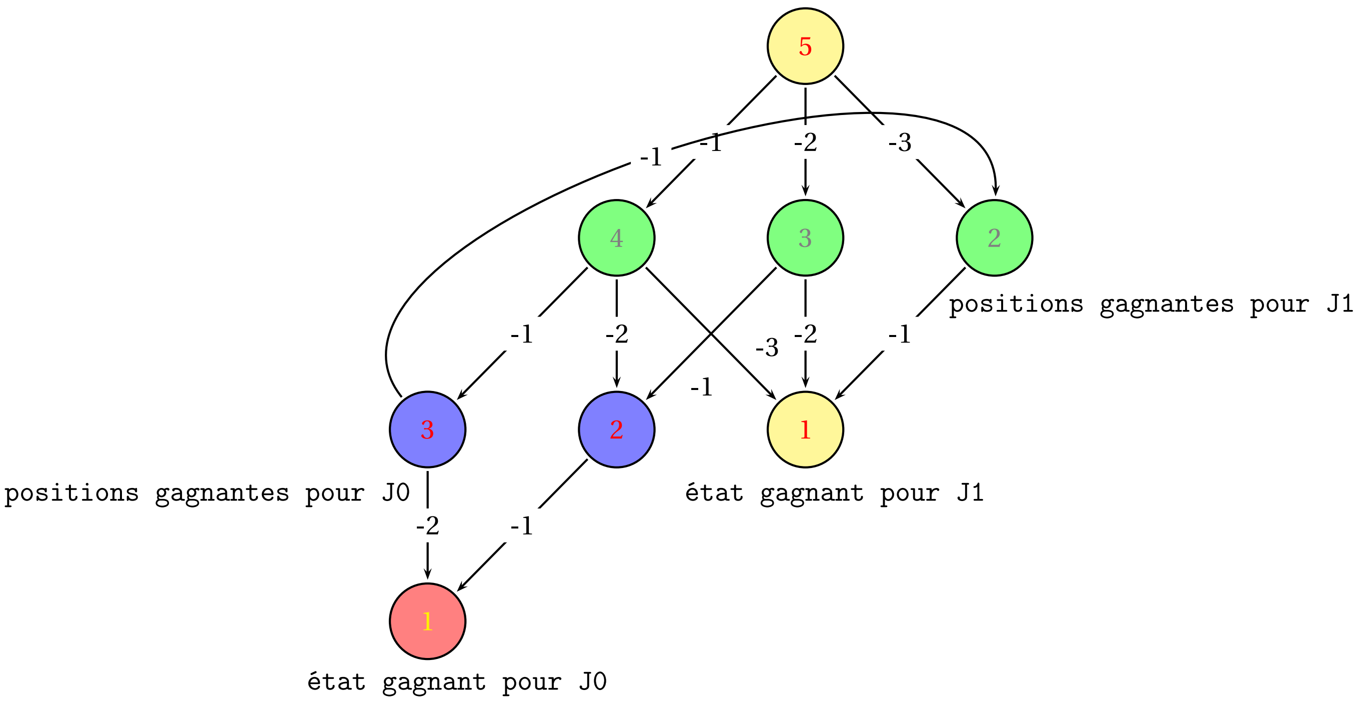
Propriété I-1: CONTRÔLE DES SOMMETS DANS LES JEUX À ALTERNANCE

Dans les jeux à deux joueurs (J_0, J_1) à alternance, un des deux sous-ensembles de V, par exemple V_0 comporte les sommets dits **contrôlés** par le joueur J_0 (rouge sur fond jaune dans le graphe ci-dessus), et l'autre V_1, les sommets **contrôlés** par le joueur J_1 (jaune sur fond rouge).

Les sommets terminaux correspondent à des situations de jeu à un seul jeton, et sont par conséquent des **états gagnants** pour le joueur ayant joué le dernier coup amenant à l'une de ces situations. Il est alors possible d'identifier pour un joueur J_i des positions qui conduiront **à coup sûr** à ces états ; on parle alors de **positions gagnantes pour le joueur J_i**. Par exemple pour $N = 5$:



Exercice de cours: **(I.2) - n° 2.** *A partir des états gagnants, identifier les positions gagnantes pour chacun des deux joueurs.*



OBJECTIFS : construire un algorithme permettant d'identifier les positions gagnantes et dégager des stratégies gagnantes pour l'un des joueurs.

I.3 Implémentation du graphe du jeu

On se propose d'écrire un code qui va implémenter le graphe du jeu de Nim avec deux joueurs J_0 et J_1. Les sommets vont être désignés par **un simple entier naturel correspondant au nombre de jetons restants**, mais pour assurer une identification plus facile, **ils seront intégrés dans un dictionnaire** :

- dont les clés auront le format suivant :

$$\boxed{'n^0 noeud - Jn^0 joueur'}$$

avec `n°noeud` désignant le noeud du graphe (nombre de jetons restants) et `n°joueur` **le numéro du joueur qui contrôle ce nœud** donc 0 ou 1 ici.

Par exemple : $'5 - J0'$ est la clé du nœud à 5 jetons qui est contrôlé par le joueur $J0$.

- et dont les valeurs seront données par la suite les entiers naturels (choix arbitraire).

Le code correspondant est le suivant :

Listing VII.1 –

```python
def graphe_NIM(N,V,A,j):
    #N: nombre de jetons
    #V: les sommets représentés par un dictionnaire
    #A: la liste d'adjacence sous forme d'une liste de listes
    def labelize(n,i): #fabrique les labels avec: n nombre de jetons du noeud,i numéro du joueur
            return "{0}-J{1}".format(n,i)
    el_0=labelize(N,j) #initialise le premier noeud

    if el_0 not in V:
        V[el_0] = len(V) #ajoute el_0 si non présent dans le dict. V avec sa valeur (ent.depuis 0)

    for i in [1,2,3]:
        if N-i>0:
            el_1=labelize(N-i,1-j)
            if el_1 not in V:
                V[el_1] = len(V)
            if [V[el_0],V[el_1]] not in A: #évite les doublons (sans gravité sinon)
                A.append([V[el_0],V[el_1]])
            graphe_NIM(N-i,V,A,1-j)
    return V,A
```

On lancera ce code avec un dictionnaire de sommets initialement vide $V = \{\}$, une liste d'adjacence également vide $A = []$, et enfin l'indice du joueur qui démarre le jeu, pour nous $j = 0$ par exemple.

Ce qui donne par exemple avec $N = 5$:

```
>>> N=5
>>> print("le graphe implémenté avec N={} est:".format(N),graphe_NIM(N,{},[],0)[0])

le graphe implémenté avec N=5 est : {'5-J0': 0, '4-J1': 1, '3-J0': 2, '2-J1': 3, '1-J0': 4,
'1-J1': 5, '2-J0': 6, '3-J1': 7}

>>> print("la liste d'adjacence correspondante est:",graphe_NIM(N,{},[],0)[1])

la liste d'adjacence correspondante est : [[0, 1], [1, 2], [2, 3], [3, 4], [2, 5], [1, 6],
[6, 5], [1, 4], [0, 7], [7, 6], [7, 4], [0, 3]]
```

II Gagner au jeu de Nim !

II.1 Stratégie - stratégie gagnante

On va définir dans un premier temps et de manière formelle les notions de **stratégie** et **stratégie gagnante** pour chaque joueur :

Définition II-1: STRATÉGIE

Si V_i est l'ensemble des sommets contrôlés par le joueur J_i, alors une **stratégie** pour ce joueur est une application $\varphi : V_i \rightarrow V_{1-i}$ telle que $\forall v \in V_i$ on a : $(v, \varphi(v)) \in A$
Par conséquent, on dira qu'un joueur **suit** la stratégie φ lors d'une partie $\mathscr{P}$ si :

$$\forall k \ v_k \in V_i \Longrightarrow \ v_{k+1} = \varphi(v_k)$$

Définition II-2: STRATÉGIE GAGNANTE - POSITION GAGNANTE

Une stratégie φ dite **gagnante** pour le joueur J_i est un ensemble de coups qu'il doit exécuter et qui le conduiront à la victoire **quelque soit les coups joués par** J_{1-i}.

Adopter une stratégie gagnante consiste donc pour J_i à se placer sur l'une des positions particulières du graphe appelée **position gagnante** qui le conduiront à la victoire **pour toute partie** s'il suit cette stratégie ; plus formellement :

un sommet $v \in V_i$ est dit **position gagnante** pour le joueur J_i s'il existe une stratégie gagnante pour J_i depuis ce sommet v.

II.2 Détermination des positions gagnantes

a - Notion d'attracteur

On va chercher maintenant à construire un outil permettant **d'exhiber les positions gagnantes pour le joueur** J_i jouant une des parties de l'ensemble $\mathscr{P}$ de toutes les parties.

On définit les éléments suivants :

- $\mathscr{F}_i$ un ensemble de sommets dits **terminaux** tel que les parties gagnantes pour J_i sont celles qui se terminent sur un sommet de $\mathscr{F}_i$. Par exemple, sur le graphe du jeu de Nim (en page 5), $\mathscr{F}_0$ est **le noeud 1 sur fond rouge**, et $\mathscr{F}_1$ est **le noeud 1 sur fond jaune**
- Ω_i l'ensemble des parties gagnantes pour J_i ; on a donc :

$$\Omega_i = \{(v_0, v_1, \ldots, v_t)\} \in \mathscr{P} \,/\, v_t \in \mathscr{F}_i$$

Ce type de jeu porte le nom de **jeu d'accessibilité**.

Considérons un jeu d'accessibilité modélisé par un graphe biparti $G = (V_0, V_1, A)$.

La condition de victoire pour le joueur J_0 est de finir sur l'un des sommets de $\mathscr{F}_0$; par exemple, dans le jeu de Nim, le joueur J_1 prendra alors le dernier jeton et perdra la partie.

Remarque II-1: GAIN DU JEU

Ainsi, pour une position donnée, le joueur J_0 **est à un coup de gagner avec certitude**,
- soit si c'est à lui de jouer et **qu'il peut choisir** une position qui le conduit à $\mathscr{F}_0$;
- soit si c'est au second joueur J_1 de jouer et **que tous les coups permis** le mènent à $\mathscr{F}_0$.

On définit alors l'ensemble $\mathscr{M}$ par la récurrence suivante :

$$\left[\begin{array}{l} \mathscr{M}_0 = \mathscr{F}_0 \\ \forall k \geq 0 \qquad \mathscr{M}_{k+1} = \mathscr{M}_k \cup \{v \in V_0 | \exists v' \in \mathscr{M}_k | (v, v') \in A\} \cup \{v \in V_1 | \forall v' \in V_0 | (v, v') \in A \implies v' \in \mathscr{M}_k\} \end{array}\right. \qquad (e)$$

qui définit l'ensemble $\mathscr{M}_{k+1}$ constitué :

- des sommets de $\mathscr{M}_k$
- des sommets contrôlés par le premier joueur J_0 pour lesquels il existe au moins un arc permettant de rejoindre un sommet de $\mathscr{M}_k$
- des sommets contrôlés par le second joueur J_1 dont **tous les arcs** aboutissent à des sommets de $\mathscr{M}_k$

Propriété II-1: RÉCURRENCE DE L'ATTRACTEUR

La récurrence (e) définit les sommets $v \in \mathscr{M}_k$ tels que **le joueur J_0 est à k coup(s) au plus de la victoire.**

Exercice de cours: **(II.2) - n° 3.** *Montrer la propriété* (e).

RÉPONSE :

- si $v \in \mathscr{M}_0 (= \mathscr{F}_0)$, il appartient à l'ensemble des positions gagnantes pour J_0, **lui assurant donc une victoire en 0 coup.**

- Supposons que $\mathcal{M}_k$ désigne les positions gagnantes pour J_0 en k coup(s) au plus, si $v \in \mathcal{M}_{k+1}$ alors compte tenu de la définition de $\mathcal{M}_{k+1}$:
  - soit $v \in \mathcal{M}_k$ alors par définition **c'est une position gagnante pour J_0 en k coup(s) au plus.**
  - soit $v \in \{v \in V_0 | \exists v' \in \mathcal{M}_k | (v, v') \in A\}$, il appartient alors à l'ensemble V_0 des sommets contrôlés par J_0 qui sont prédécesseurs d'un sommet contrôlé par J_1 appartenant à $\mathcal{M}_k$ et donc **qui assurent la victoire de J_0 en k coups au plus.**
  - soit $v \in \{v \in V_1 | \forall v' \in V_0 | (v, v') \in A \implies v' \in \mathcal{M}_k\}$, il appartient alors à l'ensemble des sommets contrôlés par J_1 dont tous les arcs aboutissent à des sommets contrôlés par J_0 et appartenant à $\mathcal{M}_k$, donc **qui assurent la victoire de J_0 en en k coups au plus.**

Cette suite $\mathcal{M}$ est croissante au sens de l'inclusion et compte tenu de V **fini**, elle est **stationnaire**, c'est à dire :

il existe un rang n_0 à partir duquel $\mathcal{M}_{n_0} = \mathcal{M}_{n_0+p}$

Définition II-3: ATTRACTEUR ET RANG

- **L'attracteur** de $\mathcal{F}_0$ pour le joueur J_0 est l'ensemble de sommets :

$$Attr(\mathcal{F}_0, J_0) = \mathcal{M}_{n_0} = \mathcal{M} = \bigcup_{k=0}^{+\infty} \mathcal{M}_k$$

- On définit de même pour tout sommet **son rang** avec :

$$rg(v) = \begin{cases} min\ \{k \ / \ v \in \mathcal{M}_k\} & \text{si } v \in Attr(\mathcal{F}_0, J_0) \\ +\infty & \text{si } v \notin Attr(\mathcal{F}_0, J_0) \end{cases}$$

Propriété II-2: RÔLE DE L'ATTRACTEUR

Si $G = (V_0, V_1, A)$ est un graphe de jeu d'accessibilité à deux joueurs à somme nulle (rappel : gain "à coup sûr" !, pas de match nul) pour lequel $\mathcal{F}_0$ est la position gagnante pour le joueur J_0 , alors :

- $\mathcal{M} = Attr(\mathcal{F}_0, J_0)$ est l'ensemble des positions gagnantes pour le joueur J_0 qui possède une stratégie gagnante pour tous ces sommets.
- $\complement\mathcal{M} = \complement Attr(\mathcal{F}_0, J_0)$ est l'ensemble des positions gagnantes pour le joueur J_1 qui possède une stratégie gagnante pour tous ces sommets.

b - Algorithme de calcul de l'attracteur

On va s'attacher maintenant à construire une fonction :

```
attracteur(V:dict,A:list,V0:list,F0:list)⟶ list
```

recevant en entrée :

- le dictionnaire des sommets V.
- la liste d'adjacence A du graphe.
- la partie $V0$ de V sous forme d'une liste, ensemble des sommets contrôlés par le joueur J_0.
- la liste $F0$ comportant la liste initiale des sommets gagnants pour $J0$.

renvoyant en sortie l'attracteur de $\mathscr{F}_0$, i.e. les positions gagnantes pour le joueur J_0 (ainsi que les stratégies gagnantes pour celui-ci.)

MISE EN PLACE DES "OUTILS" :

On commence par implémenter le dictionnaire des sommets et la liste d'adjacence avec :

```
V,A = graphe_NIM(N,{},[],0)}
```

Exercice de cours: **(II.2) - n° 4.** *Proposer les lignes de code permettant d'implémenter, d'une part la liste $V0$ des sommets contrôlés par le joueur $J0$, et d'autre part la liste initiale $F0$ des positions gagnantes pour $J0$. On pourra pour cela exploiter la reconnaissance des sous-chaînes $J0$ et $J1$ dans les clés du dictionnaire des sommets V.*

RÉPONSE :

```
V0=[v for el,v in V.items() if 'J0' in el]
F0=[v for el,v in V.items() if el=='1-J1']
```

Le principe de l'algorithme est le suivant :

- On initialise une liste vide `Attr=[]`, un dictionnaire vide des prédécesseurs `pred={}`, et un dictionnaire vide des degrés sortants des sommets du graphe `deg={}`.
- On initialise le dictionnaire des prédécesseurs des sommets avec des listes vides ; on initialise également à 0 toutes les valeurs du dictionnaire des degrés de sommet ; pour ces deux dictionnaires, les clés sont les indices des sommets.

Exercice de cours: **(II.2) - n° 5.** *Proposer une boucle python itérant sur V et permettant de réaliser ces deux dernières implémentations.*

RÉPONSE :

```
for v in V:
        pred[v] = []
        deg[v] = 0
```

- On remplit ensuite les dictionnaires des prédécesseurs `pred` et des degrés sortants de sommets `deg` à l'aide de la liste d'adjacence A, de telle façon que `pred[v]` contiennent la liste des prédécesseurs du sommet v et `deg[v]` son degré sortant.

Exercice de cours: **(II.2) - n° 6.** *Proposer une boucle python itérant sur A et permettant de réaliser ces deux dernières implémentations*

<u>RÉPONSE :</u>

```
for a in A:
        p,q = tuple(a)
        deg[p] += 1
        pred[q].append(p)
```

- CODE COMPLET : le coeur de l'algorithme va consister en un **parcours inversé** du graphe que l'on lancera **à partir de chacun des états gagnants par exemple pour** $J0$, c'est à dire inscrits dans $F0$. Le code est commenté et lancé pour $N=5$:

```
N=5
###### Implémentaion du graphe du jeu de Nim #######
V,A=graphe_NIM(N,{},[],0)

##### Liste des sommets contrôlés par J0
V0=[v for el,v in V.items() if 'J0' in el]

##### Liste des sommets positions gagnantes i.e. F0=M0
F0=[v for el,v in V.items() if el=='1-J1']

##### Algorithme de calcul de l'attracteur #####
def attracteurs(V,A,V0,F0):
    Attr=[] # initialisation de l'attracteur
    pred,deg={},{}
    for v in V: #initialisation dictionnaires de prédécesseurs et degrés sortants de sommets
        pred[v] = []
        deg[v] = 0
    for a in A: #construction dictionnaires de prédécesseurs et degrés sortants de sommets
        p,q = tuple(a)
        deg[p] += 1
        pred[q].append(p)

    def parcours_inv(v): #procédure de parcours inverse à partir du sommet v
        if v not in Attr: #si v n'est pas dans l'attracteur
            Attr.append(v) #on l'ajoute dans les positions gagnantes
            for u in pred[v]: #on itère sur tous les prédécesseurs du sommet v
                deg[u]-=1 #et pour ce sommet, on retire un degré sortant puisque traité!
                if u in V0 or deg[u]==0: #si u contrôlé par J0 ou de degré sortant nul (i.e. "isolé")
                    parcours_inv(u)#alors on relance la procédure de parcours avec u
        return None
    for x in F0: #A partir de tous les sommets présents dans $F0$:
        parcours_inv(x) #lancement parcours inverse du graphe pour construire l'attracteur
    return Attr

Attr=attracteurs(V.values(),A,V0,F0)
print(Attr)
```

La sortie console donne alors :

```
[5, 2, 6]
```

Ces numéros de sommets ne sont pas très "parlants" puisque non mentionnés sur le graphe ; pour pouvoir les identifier, il suffit d'inverser le dictionnaires des sommets en permutant les valeurs (entiers d'identification) avec les clés (bâties selon le schéma $'n^0 noeud - Jn^0 joueur'$) :

```python
###dictionnaire inversé des sommets ####
Vinv={val:cle for cle,val in V.items()}
for r in Attr:
    print(r,Vinv[r])
```

La sortie console donne alors :

```
5 1-J1
2 3-J0
6 2-J0
```

qui sont bien les positions gagnantes pour J_0.