Informatique. Attracteurs dans un jeu à un ou deux joueurs

On considère $G = (S, A)$ un graphe orienté admettant au moins deux sommets.

Pour tout sommet $s \in S$, on note $V[s]$ l'ensemble des successeurs de s, c'est-à-dire l'ensemble des sommets t tels que $(s, t) \in A$ (ensemble des arêtes).

On dit qu'un sommet t est accessible depuis s_0 ssi il existe un chemin de s_0 à t dans G.

On suppose que G n'a pas de puits, c'est-à-dire que pour tout $s \in S$, $V(s)$ n'est pas vide.

On suppose $S = S_1 \sqcup S_2$ (partition).

Soit $T \subset S$. *Remarque terminologique* : On dit que le quadruplet (G, S_1, S_2, T) est une arène.

Pour $s_0 \in S$, on considère le jeu à deux joueurs $J = (G, S_1, S_2, T, s_0)$.

Le principe du jeu est le suivant :

- au début de la partie, un jeton est placé sur le sommet s_0

- à chaque tour de jeu, **le joueur auquel appartient le sommet s où se trouve le jeton peut le déplacer sur un sommet de $V(s)$** (il s'agit du joueur 1 si $s \in S_1$, et du joueur 2 si $s \in S_1$).

Remarque : Dans ce jeu général, un même joueur peut parfois jouer plusieurs coups consécutifs.

- **si le jeton atteint un sommet de T , c'est le joueur 1 qui remporte la partie.** Sinon, si la partie est infinie sans que le jeton n'atteigne un sommet de T, et c'est le joueur 2 qui remporte la partie.

Formellement, une partie σ est une suite infinie de sommets $(s_i)_{i \in \mathbb{N}}$ telle que

$$\forall i \in \mathbb{N},\ s_{i+1} \in V(s_i)$$

En particulier, le premier sommet de la suite est le sommet s_0.

Une telle partie est gagnée par le joueur 1 ssi il existe $i \in \mathbb{N}$ tel que $s_i \in T$.

Une stratégie pour le joueur $j \in \{1, 2\}$ est une fonction $f : S_j \to S$ telle que pour tout $s \in S_j$, $f(s) \in V(s)$.

Soit $j \in \{1, 2\}$. On considère $f : S_j \to S$ une stratégie pour le joueur j.

Une f-partie est une partie σ telle que $s_{i+1} = f(s_i)$ pour tout i tel que $s_i \in S_j$.

Autrement dit, une f-partie est une partie où le joueur j suit la stratégie f.

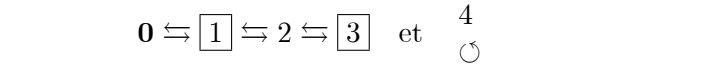
On dit que f est une stratégie gagnante pour le joueur j si toute f-partie est gagnée par le joueur j.

On dit que **le sommet $s_0 \in S$ est gagnant pour le joueur j s'il existe une stratégie gagnante pour j.**

Partie A. Préliminaires

Question 1.

On considère le graphe de suivant :



où $S_1 = \{1, 3\}$ et $S_2 = \{0, 2, 4\}$, et $A = \{(0, 1), (1, 0), (1, 2), (2, 1), (2, 3), (3, 2), (4, 4)\}$.

On suppose que $T = \{0\}$. Donner sans justification les sommets gagnants pour le joueur 1 et les sommets gagnants pour le joueur 2.

Question 2.

On note R_1 l'ensemble des sommets s_0 gagnants pour le joueur 1 et R_2 l'ensemble des sommets gagnants pour le joueur 2. Montrer que R_1 et R_2 forment **une partition** de S.

Pour la suite, on représente une arène (G, S_1, S_2, T) en PYTHON de la manière suivante :

- S est l'ensemble $[\![0, n-1]\!] = \{0, 1, 2, ..., n-1\}$

- un graphe $G = (S, A)$ est représenté par une liste `G` de listes d'adjacence telle que pour $s \in S$, `G[s]` est la liste contenant les sommets de $V(s)$

- S_1 et S_2 sont représentés par une liste booléenne `S` de taille n telle que pour $s \in S$, `S[s]` vaut $\begin{cases} \texttt{True} \text{ si } s \in S_1 \\ \texttt{False} \text{ si } s \in S_2 \end{cases}$

- T est représenté par la liste `T` de ses éléments.

Partie B. Jeu à un seul joueur

On suppose ici $S_1 = S$ et $S_2 = \emptyset$. Autrement dit, seul le joueur 1 joue.

On définit le graphe transposé de $G = (S, A)$ comme le graphe $G^T = (S, A^T)$, où

$$\forall (s,t) \in S^2, \quad (s,t) \in A^T \text{ ssi } (t,s) \in A$$

Autrement dit, le graphe est obtenu en inversant l'orientation des arêtes.

Question 3.

Montrer brièvement qu'un sommet $s \in S$ est gagnant pour 1 si et seulement si il existe un chemin de s à un sommet de T dans G.

Ainsi, un sommet est gagnant pour le joueur 1 ssi il est accessible depuis un sommet de T dans le graphe G^T.

Question 4.

Écrire une fonction `transpose(G)` qui prend en argument un graphe $G = (S, A)$ et renvoie le graphe G^T.

La fonction devra avoir une complexité en $O(n + m)$, où $n = \operatorname{card} S$ et $m = \operatorname{card} A$.

Question 5.

Écrire une fonction `parcours(G,T)` qui renvoie la liste des sommets gagnants dans l'arène (G, T).

La fonction renverra la liste des sommets gagnants.

Conseil : Utiliser un parcours en profondeur dans le graphe G^T pour construire (via une fonction auxiliaire récursive) la liste des sommets gagnants. On pourra aussi utiliser un tableau de marquage `visites` pour indiquer les sommets déjà traités.

Partie C. Cas général

On suppose pour la suite que S_1 et S_2 ne sont pas vides.

On définit par induction l'ensemble $X(i)$, pour $i \in \mathbb{N}$, par

- $X(0) = T$

- $X(i+1) = X(i) \cup \{s \in S_1 \mid V(s) \cap X(i) \neq \emptyset\} \ \cup \ \{s \in S_2 \mid V(s) \subset X(i)\}$.

Question 6.

Montrer par récurrence sur i que $s \in X(i)$ ssi il existe une stratégie f pour le joueur 1 telle que toute f-partie conduit à un sommet de T en au plus i étapes.

Question 7.

Montrer qu'il existe $k \leq n$ tel que $\forall i \geq k$, $X(i) = X(k)$.

On note désormais R_1 cet ensemble $X(k)$ appelé attracteur de T. On pose $R_2 = S \setminus R_1$.

Question 8.

(★) Montrer que tout sommet $s \in R_1$ est gagnant pour le joueur 1 et que tout sommet $s \in R_2$ est gagnant pour le joueur 2.

Question 9.

On veut définir une fonction `attracteur(G,S,T)` qui détermine l'ensemble des états gagnants pour le joueur 1.

a) Proposer une fonction utilisant un algorithme de style parcours en parcours en largeur qui construit successivement en n étapes les parties $X(0)$, $X(1)$, ... , $X(n)$.

Indication : Utiliser une liste booléenne `visites` pour garder la trace des éléments ajoutés à l'attracteur et `T.copy()` pour créer une copie de `T`.

b) Pour améliorer la complexité, on utilise un algorithmique de style parcours en profondeur donné par le code :

```
01.    def attracteur(G, S1, T):
02.         n = len(G)
03.         visites = [False] * n
04.         X = []
05.         GT = transpose(G)
06.         tabAttract = [0] * n
07.         pile = []
08.         for s in T :
09.              pile.append(s) ; visites[s] = True
10.         while pile != [] :
11.              s = pile.pop() ; X.append(s)
12.              for t in GT[s] :   tabAttract[t] += 1
13.              for t in GT[s] :
14.                   if S1[t] or (not S1[t] and tabAttract[t] == len(G[t]) :
15.                       if not visites[t] :  pile.append(t) ; visites[t] = True
16.         return X
```

Expliquer ce que contient le tableau `tabAttract` intervenant lignes 06, 12 et 14

Proposer une variante de ce programme utilisant une fonction auxiliaire récursive au lieu de la pile.

Question 10.

Donner la complexité des fonctions précédentes a) et b) en fonction de $n = \operatorname{card} S$ et $m = \operatorname{card} A$.