Jeux de la soustraction

INFORMATIQUE COMMUNE - TP n° 3.6 - Olivier Reynet

À la fin de ce chapitre, je sais :

- ☞ coder le calcul de attracteur pour un jeu d'accessibilité
- ☞ réutiliser les concepts de graphe et de parcours en largeur

A Jeu de la soustraction

Le jeu de la soustraction est un jeu à deux joueurs. Devant eux se trouvent n bâtonnets [1]. Les joueurs jouent l'un après l'autre et ont le droit de retirer **un**, **deux** ou **trois** bâtonnets. Le gagnant est celui qui tire le **dernier** [2] bâtonnet.

A1. Le jeu de la soustraction est-il un jeu d'accessibilité ? Pourquoi ?

A2. Jouer avec votre voisin en prenant sept bâtonnets.

A3. Combien de positions possibles existe-t-il pour cette partie à sept bâtonnets ?

A4. Construire sur le papier l'arène de ce jeu pour $n = 7$. On choisit la convention suivante : les sommets contrôlés par le premier joueur sont numérotés de 0 à n, ceux du second joueur de $n+1$ à $2n+1$.

A5. Calculer à la main l'attracteur du premier joueur et le reporter sur la figure précédente.

A6. Que fait le code suivant ?

```
def mystery_code(n):
  size = n + 1
  a = [[] for _ in range(2 * size)]
  for i in range(size):
    for j in range(1, 4):
      if i - j >= 0:
        a[i].append(size + i - j)
        a[i + size].append(i - j)
  return a
```

Un indice : pour $n = 7$, cette fonction renvoie :

```
[[], [8], [9, 8], [10, 9, 8], [11, 10, 9], [12, 11, 10], [13, 12, 11], [14, 13,
    12], [], [0], [1, 0], [2, 1, 0], [3, 2, 1], [4, 3, 2], [5, 4, 3], [6, 5, 4]]
```

---

1. On peut y jouer avec des pièces, des stylos ou des allumettes…
2. La variante misère en fait le perdant.

B Attracteur

Pour trouver l'attracteur $\mathcal{A}$ d'un joueur, il suffit de parcourir l'arène de jeu en partant de sa condition de gain et en inversant les arcs, c'est-à-dire en parcourant le graphe transposé. C'est l'algorithme que l'on développe dans cette section.

a Fonction utiles

B7. Écrire une fonction de signature `players_vertices(n : int)-> (list[int], list[int])` qui renvoie les listes des sommets du joueur 1 et du joueur 2. Par exemple, `players_vertices(7)` renvoie le tuple `([0, 1, 2, 3, 4, 5, 6, 7], [8, 9, 10, 11, 12, 13, 14, 15])`.

B8. Écrire une fonction de signature `gain_condition(n: int)-> (int, int)` qui renvoie les conditions de gain du joueur 1 et du joueur 2. Par exemple `gain_condition(7)` renvoie `(8,0)`.

B9. Écrire une fonction de signature `start_positions(n: int)` qui renvoie les positions de départ des joueurs 1 et 2 sur l'arène de jeu. Par exemple, `start_positions(7)` renvoie `(7,15)`.

B10. Écrire une fonction de signature `previous_positions(p: int, n:i nt)-> list[int]` dont les paramètres sont `p` un sommet du graphe et `n` le nombre de bâtonnets. Cette fonction renvoie la liste des sommets qui conduisent à `p` dans l'arène du jeu.

b Calcul de l'attracteur d'un joueur

L'algorithme 1 détaille comment parcourir l'arène de jeu transposée afin de trouver l'attracteur d'un joueur. La remarque fondamentale est la suivante : lorsqu'on découvre un nouveau sommet depuis l'attracteur :

- soit ce sommet est un sommet du joueur et on peut l'ajouter à l'attracteur sans risques,
- soit ce sommet est un sommet de l'adversaire. Dans ce cas, on ne peut l'ajouter à l'attracteur que si les arcs qui en sortent ne conduisent qu'à des sommets de l'attracteur (cas où l'adversaire est captif).

Le rang d'un sommet est mémorisé en même tant que le sommet : il permet de choisir une stratégie gagnante. Lorsque le rang d'un sommet diminue, on se rapproche de la victoire.

B11. Écrire une fonction de signature `build_attractor(n, player)` qui renvoie l'attracteur du joueur 1 si `player` vaut 0 et 2 sinon dans le cas du jeu de la soustraction. Cette fonction parcourt en largeur l'arène de jeu transposée. Elle renvoie une liste de tuples (sommet, rang) qui représente l'attracteur. Par exemple, `build_attractor(7,0)` renvoie la liste `[(9, 0), (1, 1), (2, 1), (3, 1), (13, 2), (5, 3), (6, 3), (7, 3), (17, 4)]`

B12. Écrire une fonction de signature `in_attractor(A, pos` qui renvoie la position dans la liste et le rang de la position si `pos` se trouve dans l'attracteur du joueur. Si ce n'est pas le cas, la fonction renvoie `(None, None)`.

B13. Écrire une fonction de signature `next_in_attractor(A, pos)` qui permet d'implémenter la stratégie gagnante. Si le joueur est en `pos`, elle renvoie le position jouable suivante dans l'attracteur ou `None` sinon.

Algorithme 1 Calcul de l'attracteur d'un joueur, jeu de la soustraction

```
1 : Fonction ATTRACTEUR(n)
2 :     V ← l'ensemble de sommets du joueur dont on calcule l'attracteur
3 :     C_g ← la condition de gain du joueur
4 :     s_1 et s_2 les positions de départ des joueurs
5 :     A ← ∅                                                    ▷ l'attracteur
6 :     d_in ← les degrés entrants des sommets l'arène de jeu transposé
7 :     r ← 0 le rang initiale (de la condition de gain)
8 :     F ← une file d'attente
9 :     ENFILER(F, (C_g, r))                  ▷ On va parcourir en largeur le graphe transposé
10 :    tant que F n'est pas vide répéter
11 :        u, r ← DÉFILER(F)
12 :        AJOUTER(A, (u, r))                                   ▷ augmentation de l'attracteur
13 :        si u est un sommet de départ alors                   ▷ Exploration achevée, early exit
14 :            renvoyer A          ▷ L'attracteur du joueur comporte nécessairement soit s_1 soit s_2
15 :        pour chaque sommet à v conduisant à u répéter
16 :            Décrémenter le degré de v dans d_in          ▷ On l'a découvert une fois depuis l'attracteur
17 :            si v appartient au joueur ou si on ne peut arriver à v que depuis l'attracteur alors
18 :                ENFILER(F, (v, r + 1))
```

C Programmation du jeu humain vs. ordinateur

On dispose d'un programme implémentant le jeu de la soustraction. L'humain joue en premier. L'ordinateur utilise les fonction précédentes qui se trouve dans une fichier (module) nommé `attractor.py`.

C14. Jouer et vérifier que vous ne pouvez que perdre lorsque n est pair.

C15. Jouer et vérifier que vous pouvez gagner lorsque n est impair.