DS 4 : Jeux de soustraction

On s'intéresse ici à une famille de jeux : les jeux de soustractions. Il s'agit d'une sous-famille des jeux combinatoires impartiaux, eux-mêmes une sous-famille des jeux d'accessibilité. Le sujet est entièrement à réaliser en OCaml. Les réponses doivent être justifiées et les codes commentés. On apportera une attention toute particulière à la qualité de la rédaction.

On notera les arêtes de graphes $u \to v$.

1 Jeux impartiaux

Un jeu impartial est un jeu à deux joueurs tour par tour pour lequel les coups possibles et les valeurs des positions ne dépendent que de la position et non du joueur dont c'est le tour. Soit 1 et 2 les deux joueurs. L'arène A d'un jeu impartial est de la forme

$$A = (G, S \times \{1\}, S \times \{2\}) \text{ avec } G = (S \times \{1;2\}, E) \text{ acyclique}$$

et tel que pour toute arête $(p,i) \to (q, 3-i) \in E$ on a $(p, 3-i) \to (q,i) \in E$.

Notons T l'ensemble des sommets sans arêtes sortantes de G. La valeur d'un élément (p,i) de T désigne le joueur ayant gagné et vaut $v((p,i)) = 3-i$. Cela signifie que le joueur ne pouvant plus jouer a perdu.

1. Les jeux suivants sont ils impartiaux ?
   a) Les échecs.
   b) Le pierre-feuille-ciseau.
   c) Le poker.

Du fait de la symétrie du jeu, on peut représenter un jeu impartial par son graphe simplifié $G' = (S, E')$ où $E' = \{p \to q \mid (p,1) \to (q,2) \in E\}$. Un élément de S est appelé **position** et une arête de E' est appelé **coup**.

Dans la suite, on considère les **jeux de soustractions**. Il s'agit de jeux composés de k piles d'allumettes et où chaque coup consiste à retirer un certain nombre d'allumettes. Les positions de ces jeux sont représentés par des vecteurs de $\mathbb{N}^k$. On considère en particulier les jeux suivants :

— **Le jeu des allumettes** : on dispose d'une pile contenant n allumettes, et un coup consiste à retirer 1, 2 ou 3 allumettes. On note Al_n ce jeu.

— **Le jeu de Nim** : on dispose d'une pile contenant n allumettes, et un coup consiste à retirer autant d'allumettes qu'on le souhaite, mais toujours au moins 1. On note N_n ce jeu.

— **Le jeu de Wythoff** : on dispose de deux piles de n et m allumettes, et un coup consiste soit à retirer des allumettes dans une seule des deux piles, soit à retirer autant des allumettes, au moins 1, dans chaque pile. On note $W_{n,m}$ ce jeu.

On présente le graphe simplifié du jeu N_5 figure 1

2. Représenter les graphes simplifiés de Al_7 et $W_{3,2}$.

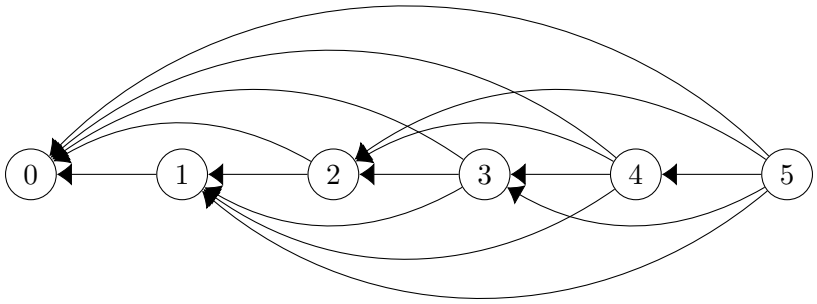

FIGURE 1 – Graphe simplifié de N_5

3. Donner une définition formelle de Al_n, N_n et $W_{n,m}$.

Soit $u = (u_1, \ldots, u_k)$ et $v = (v_1, \ldots, v_k)$ deux vecteurs de dimension k. On note, $u - v$ la soustraction terme à terme de u par v. Donc :

$$u - v = (u_1 - v_1, \ldots, u_k - v_k)$$

De même, on note $u + v$ l'addition terme à terme, et on compare les vecteurs terme à terme. Ainsi $u \leq v$ si et seulement si, pour tout i, $u_i \leq v_i$. On définit $u < v$ si $u \leq v$ et $u \neq v$. On écrit 0_k le vecteur nul de dimension k.

4. Écrire une fonction `soustraction : int array -> int array -> int array` qui calcul la soustraction de deux vecteurs de même dimension.
5. Écrire une fonction `positif : int array -> bool` qui prend en entrée un vecteur v et renvoie `true` si et seulement si il ne possède pas de composante strictement négative.

Un jeu de soustraction peut être entièrement représenté par son ensemble de règle R, potentiellement infinie, correspondant aux nombres d'allumettes pouvant être récupérée. Ainsi, si R est l'ensemble de règle d'un jeu à k piles d'allumettes, et que u et v sont des vecteurs de $\mathbb{N}^k$, $u \to v$ est un coup valide si et seulement si $u - v \in R$. Le jeu des allumettes est ainsi définie par l'ensemble de règle $R_A = \{1; 2; 3\}$.

6. Décrire R_N et R_W respectivement les ensembles de règles du jeu de Nim et du jeu de Wythoff.

On représente les positions d'un jeu de soustraction à k piles à l'aide d'un `int array` de dimension k. Un jeu de soustraction peut donc être représenté en OCaml par la donnée d'une **fonction de règle** `r : int array -> int array list` qui prend en entrée une position d'un jeu et renvoie la liste des positions accessibles.

7. Écrire une fonction `voisins_al : int array -> int array list` qui prend en entrée une position p d'un jeu d'allumette et renvoie la liste des positions accessibles à partir de p.
8. Écrire une fonction `voisins_nim : int array -> int array list` qui prend en entrée une position p d'un jeu de Nim et renvoie la liste des positions accessibles à partir de p.
9. Écrire une fonction `voisins_wyt : int array -> int array list` qui prend en entrée une position p d'un jeu de Wythoff et renvoie la liste des positions accessibles à partir de p.

Considérons un jeu de soustraction J à k piles tel que chaque pile a au plus n allumettes.

10. Déterminer une fonction bijective $\varphi_{n,k}$ des positions de J dans $\{0; \ldots; (n+1)^k - 1\}$.
11. Écrire une fonction `arr2int : int -> int array -> int` telle que `arr2int n pos` avec `pos` vecteur de dimension k, renvoie $\varphi_{n,k}(pos)$.
12. Écrire une fonction `int2arr : int -> int -> int -> int array` telle que `int2arr n k i` renvoie `pos`, `array` de dimension k tel que $\varphi_{n,k}(pos) = i$.
13. Prouver la correction de `arr2int` et `int2arr`.
14. Donner la complexité de `arr2int` et `int2arr`.

En utilisant la fonction $\varphi_{n,k}$, il nous est donc possible d'associer à chaque position une valeur stockée dans un tableau unidimensionnel.

15. Écrire une fonction `initialiser : int -> int -> 'a -> 'a array` tel que `initialiser n k x` renvoie un `array` de dimension $(n+1)^k$ intialisé à `x`.

Toutes nos positions initiales ne seront pas nécessairement sous la forme $(n, n, \ldots, n)$. Il peut donc être utile de récupérer le plus grand élément d'un vecteur.

16. Écrire une fonction `maxArr : int array -> array` qui renvoie le plus grand entier d'un `array` d'entiers.

On dit qu'une position p est gagnante si, à partir de l'état (p, i), le joueur i possède une stratégie gagnante. Sinon, elle est perdante. Les positions perdantes sont appelées $\mathcal{P}$-positions.

17. Justifier que si l'état (p, i) est gagnant pour i, alors $(p, 3-i)$ est gagnant pour $3-i$.
18. Soit une position perdante. Que peut-on dire de la nature de ses descendants ?
19. Même question pour les positions gagnantes.
20. Énumérer toutes les $\mathcal{P}-$positions d'un jeu de Wythoff avec au plus 5 allumettes dans chaque pile.
21. Écrire une fonction `gagnant : (int array -> int array list) -> int array -> bool` qui prend en entrée une fonction `r : int array -> list int array` décrivant un jeu de soustraction, une position p et qui répond `true` si et seulement si il s'agit d'une position gagnante. On suppose que les appels à `r` s'effectuent en $O(1)$ et on impose une complexité en $O(|E|)$, avec $|E|$ le nombre d'arêtes dans notre jeu.

À partir des positions gagnantes, il nous est alors possible de calculer une stratégie gagnante correspondante.

22. Exprimer une stratégie gagnante en fonction des $\mathcal{P}$-positions.
23. Écrire une fonction `strategie : (int array -> int array list) -> int array -> int array` qui prend en entrée une fonction `r : int array -> list int array` décrivant un jeu de soustraction, une position p non terminale et qui renvoie le meilleur coup à partir de p. Si il existe plusieurs meilleurs coups possibles, on renvoie n'importe lequel.

2 Somme de Jeux de Nim

Soit J et K deux jeux de soustractions de graphe simplifiés associés $G_J = (S_J, E_J)$ et $G_K = (S_K, E_K)$. On définit la somme de ces deux jeux, noté $J + K$ comme le jeu ayant comme graphe simplifié $G = (S, E)$ avec :

$$S = S_J \times S_K \quad E = \left\{(p,q) \to (p',q') \mid \left(p = p' \wedge q \to q' \in E_K\right) \vee \left(q = q' \wedge p \to p' \in E_J\right)\right\}$$

Si J se joue sur k piles et K sur k' piles, on note les positions de $J+K$ comme un vecteur sur $\mathbb{N}^{k+k'}$. Autrement dit, le jeu $J+K$ correspond au jeu ou chaque joueur peut jouer soit dans J soit dans K.

24. Réprésenter le graphe simplifié de $N_2 + N_2 + N_1$.
25. Quelles sont les positions terminales de $J+K$?

On désire à présent calculer la fonction de règle d'une somme de jeu.

26. Écrire une fonction `concatener : int array -> int array -> int array` qui concatène deux `int array`.
27. Écrire une fonction `extraire : int array -> int -> int -> int array` telle que `extraire a i j` renvoie un `int array` de valeur `[|a.(i);a.(i+1);...;a.(j)|]`.
28. Écrire une fonction `somme : (int array -> int array list) -> (int array -> int array list) -> int -> int -> int array -> int array list` qui prend en entrée `rj rk k1 k2` deux fonctions de règles `rj` et `rk` correspondant à des jeux J et K se jouant sur `k1` et `k2` piles, et renvoie une fonction de règle pour le jeu $J+K$.

On souhaiterait savoir quelle est la valeur de la somme d'une position.

29. Soit p une position de J et q une position de K. Que peut-on dire de la valeur de (p,q) dans $J+K$ si p ou q est une $\mathcal{P}-$position ?
30. Peut-on dire quelque chose si ni p ni q n'est une $\mathcal{P}-$position ?

On va maintenant s'intéresser à la somme de plusieurs jeux de Nim.

31. Énumérer les $\mathcal{P}-$positions du jeu $N_3 + N_4$.
32. Quelles sont les $\mathcal{P}-$positions du jeu $N_n + N_m$?

On va noter $N_{n_1+n_2+\ldots+n_k}$ le jeu $N_{n_1}+\ldots+N_{n_k}$. Pour pouvoir calculer la valeur des positions, d'un tel jeu, il nous faut considérer l'opérateur XOR bit à bit. On note cet opérateur $\oplus$. On définit $x \oplus y$ comme l'entier dont la représentation binaire est telle que chaque bit est le XOR des bits en même position de la représentation binaire de x et y. On notera $\overline{x}^b$ pour préciser que x est écrit en base b. Sans précision, on le considèrera comme écrit en décimal.
Ainsi, $\overline{10001}^2 \oplus \overline{111}^2 = \overline{10110}^2$.

33. Calculer $14 \oplus 5$.
34. Écrire une fonction `xor : int -> int -> int` qui calcule le XOR de deux entiers.

Soit $J = N_{n_1+\ldots+n_k}$ et, soit $x = (x_1, \ldots, x_k)$ une position d'un tel jeu.

35. Montrer que si $x_1 \oplus \ldots \oplus x_k \neq 0_k$, alors il existe une position $y = (y_1, \ldots, y_k)$ dans J accessible à partir de x telle que $y_1 \oplus \ldots \oplus y_k = 0_k$.
36. En déduire que les $\mathcal{P}-$positions de J sont les x tels que $x_1 \oplus \ldots \oplus x_k = 0_k$.
37. Écrire une fonction `gagnant_somme : int array -> bool` qui prend en entrée une position d'une somme de jeux de Nim et qui renvoie `true` si la position est gagnante.

On note qu'il nous est donc possible de résoudre efficacement une somme de jeux de Nim sans explorer entièrement le graphe.

38. Écrire une fonction `coup_optimal : int array -> (int*int)` qui prend en entrée une position d'une somme de jeu de Nim et qui renvoie un couple `(i,j)` correspondant au meilleur coup possible, avec i le numéro de la colonne dans laquelle prendre une allumette et j le nombre d'allumettes à prendre. Votre algorithme n'explorera pas l'arbre du jeu.
39. Donner sa complexité.
40. En déduire une fonction `destination_optimal : int array -> int array` qui donne la position optimale dans laquelle jouer pour une somme de jeux de Nim.

3 Somme de jeux de soustraction quelconques

Le but de cette partie est de remarquer que chaque jeu de soustraction est équivalent à un jeu de Nim. Autrement dit, on peut associer à chaque jeu de soustraction J un entier n tel que pour tout jeu de soustraction K, $J + K$ est une $\mathcal{P}-$position, si et seulement si $N_n + K$ est une $\mathcal{P}-$position.

Pour ce faire, on commence par introduire l'opérateur mathématique mex. Le mex d'un ensemble d'entiers E est le plus petit entier positif non inclus dans E.

41. Donner $\text{mex}\{3; 0; 2; 7; 1; 5\}$.
42. Que peut-on dire sur $\text{mex}(E)$ par rapport à $|E|$?
43. Écrire une fonction de complexité linéaire `mex : int list -> int` qui calcule le `mex` d'une liste d'entiers.

On définit la **valeur de Grundy** d'une position p comme $g(p)$ le mex des valeurs de Grundy de ses descendants.

44. Écrire `grundy : (int array -> int array list)-> int array -> int` qui prend en entrée la fonction de règle d'un jeu de soustraction et une position et calcul la valeur de Grundy de cette position.
45. Prouver que p est une $\mathcal{P}-$position si et seulement si $g(p) = 0$.

On va maintenant s'intéresser aux valeurs de Grundy d'une somme de jeu.

46. Soit p et q des positions des jeux J et K. Montrer que (p, q) est une position gagnante dans $J + K$ si et seulement si $g(p) = g(q)$.
47. Soit p et q des positions des jeux J et K. Prouver que $g((p, q)) = g(p) \oplus g(k)$. Autrement dit la valeur de Grundy de (p, q) dans $J + K$ correspond au XOR des valeurs de Grundy de p et q dans J et K.
48. Conclure.