Congrès Dédra-MATH-isons
MATH en JEANS 2012

Par

CLARINVAL Antoine
DE MEESTER Léonore
DROUMART Killian
GOFFIN Amory
HANOUL Chloé
JEMINE Corentin
LOBET Rémy
MAKHLOUFI Marine
THIRION Eve
VANDENBUSSCHE Sébastien
van der VALK Diederik

Encadrés par *Mme Sabine DE BLIECK*

Institut Saint-Laurent Marche-en-Famenne

Merci à Bérangère et Benjamin, et à leurs professeurs Thomas Brihaye et Stéphanie Bridoux, qui nous ont accompagnés dans notre recherche.

Si vous aussi avez toujours rêvé de gagner au Jeu de Nim, par exemple contre le Maître de Fort Boyard, c'est désormais possible. Intrigués par ce « *mathgame* », nous avons en effet décidé de nous aventurer dans cette énigme. Suite à de nombreuses recherches, nous avons découvert une stratégie gagnante pour différentes variantes de ce jeu. Père Fouras n'a qu'à bien se tenir. Grâce à nous, vous détiendrez les clés de Passe-Partout. Ne faites donc plus *NIM*porte quoi!

Notre aventure a débuté au cours de notre 5ème année secondaire, lors d'une activité organisée par notre professeur qui nous a proposé de voir les jeux sous un nouvel angle. En effet, contrairement au poker, certains jeux ne sont pas liés au hasard. C'est le cas du fameux jeu de Nim que nous avons découvert et revisité à la mode belge puisque les pions ont été remplacés par des frites !

Après plusieurs parties perdues contre notre professeur, les moins têtus d'entre nous ont enfin décidé d'abandonner définitivement et de se lancer dans la recherche de la « Formule Magique » pour gagner.

Afin d'élucider les mystères mathématiques de ces formules secrètes, nous avons divisé notre travail en plusieurs parties.

Tout d'abord, nous nous sommes intéressés aux origines possibles de ce jeu et à son histoire puisqu'il apparaît à la genèse des jeux dits « combinatoires ».

Ensuite, nous pouvions commencer le travail mathématique, le véritable objet de notre recherche. Dans un premier temps, le plus simple était d'élaborer intuitivement une stratégie pour **le jeu de Nim à un tas,** comme celui que nous avions testé un an auparavant.

Au travers de nos prospections trépidantes, il nous sembla intéressant d'aborder ce jeu sous l'angle de **l'arithmétique modulaire** et de la **théorie des graphes**.

Grâce à cela, nous avons pu dans un second temps trouver mais surtout justifier une stratégie gagnante pour le jeu de **Nim à plusieurs tas**. Bien évidemment, il nous pressait d'arriver au cours de mathématique pour annoncer notre découverte et la tester sur nos camarades ou notre professeur.

Par après, nous nous sommes étendus vers d'autres façons de jouer et nous avons repéré d'autres jeux très intéressants comme le jeu de **Moore** ou encore le jeu portant le nom du mathématicien Leonardo **Fibonacci**, basé sur sa suite rendue célèbre grâce au « Da Vinci Code ». Cela nous a permis d'apprendre une nouvelle forme de numération utilisant cette même suite et dont la paternité revient à un Belge, du nom d'Emile **Zeckendorf**.

Aussi, ce n'est pas un hasard si, tout au long de notre travail, nous avons choisi d'utiliser des frites plutôt que des pièces pour illustrer nos exemples. En effet, elles nous permettent non seulement d'exprimer notre unicité et une certaine convivialité dans le jeu mais elles montrent aussi une certaine particularité de notre pays que nous nous devions d'aborder.

Vous trouverez ci-joint tout le fruit de notre recherche ainsi que le lot de démonstrations s'y rapportant.

Bonne lecture et bon amusement !

Sommaire

I. Introduction

Le jeu de Nim est un jeu de stratégie à 2 joueurs. Il a été créé il y a si longtemps que nous n'en connaissons pas le réel inventeur. Il existait déjà en Chine au XVIème siècle sous le nom de *Fan-tan*, en Afrique sous le nom de *Tiouk-tiouk* mais nous avons gardé le nom Nim donné par un mathématicien anglais, Charles Léonard Bouton qui en fit une analyse détaillée et jeta ainsi les fondements de la théorie des jeux.
Dans la langue allemande, « Nimm » signifie « prendre ». Par ailleurs, une symétrie centrale amusante du mot NIM donne le mot WIN !
Le vieux classique d'Alain Resnais « L'année dernière à Marienbad » qui a remporté de nombreux prix en 1961 a fait connaître mondialement le jeu de Nim. Le héros y joue longuement à un jeu de Nim et a cette réplique célèbre « *Je puis perdre mais je gagne toujours* ». Cette version des jeux de Nim d'ailleurs été rebaptisée « Jeu de Marienbad».
De nombreuses versions du jeu de Nim ont été développées depuis et rentrent aujourd'hui dans la catégorie de ce que Aviezri Fraenkel (mathématicien israélien né en 1929) appelle les *Mathgames* : jeux qui intéressent essentiellement les mathématiciens pour le plaisir de la recherche d'une stratégie de contrôle.

I. 1. But du jeu de Nim

Le jeu de Nim se joue par deux à tour de rôle. Le plateau de jeu se présente sous la forme d'un certain nombre de pièces (cailloux, allumettes, pions … frites !) rangées en un ou plusieurs tas selon les versions et chacun des deux joueurs, tour à tour, y retire le nombre de pièces qu'il souhaite jusqu'à ce qu'il ne reste plus aucune pièce.
C'est un jeu de stratégie et non de hasard: ce sont des règles précises qui déterminent le cours du jeu. Il est impossible d'avoir un match nul, il y a toujours un vainqueur et un perdant.
Il existe plusieurs versions du jeu de Nim, qui diffèrent selon l'issue de la partie, le nombre de tas ou de pièces à prélever, que nous explorerons dans ce dossier. Chaque type de jeu de Nim possède une stratégie gagnante basée sur les positions de jeu. La recherche de ces stratégies gagnantes constitue l'enjeu principal de notre travail.

I. 2. Jeux combinatoires

Avant d'aborder spécifiquement le jeu de Nim, il convient d'évoquer plus généralement la théorie des jeux. Cette théorie qui se situe à la frontière de l'économie et des mathématiques, a pour objectif de proposer une aide à la décision et une recherche de stratégie par une approche mathématique.
Les jeux de Nim relèvent de la branche spécifique des **jeux combinatoires**.
Pour être un jeu combinatoire, le jeu doit satisfaire les conditions suivantes :

- Il y a exactement deux joueurs.
- L'ensemble de positions possibles du jeu est fini (le jeu se termine avec un nombre limité de déplacements, peu importe comment ils sont joués).
- Les changements qui permettent de passer d'une position du jeu à une autre sont régis par un ensemble de règles, qui peuvent être spécifiques à chacun des deux joueurs.
    - Si les règles ne font pas de distinction entre les joueurs, c'est-à-dire si les joueurs ont les mêmes options pour changer de position, le jeu est dit **impartial** ; si ce n'est pas le cas, il est dit **partisan.** Le jeu de nim est donc un jeu impartial (ce sont les mêmes règles pour les deux joueurs).
- Les joueurs jouent en stricte alternance (on ne passe pas son tour).
- Un jeu combinatoire est un jeu avec des informations parfaites : pas de mouvements simultanés ni de déplacements masqués. Chaque joueur a donc connaissance de la position en cours et peut déterminer celle à venir. Un jeu faisant intervenir le hasard comme le poker ne permet pas une information parfaite et ne rentre donc pas, pour cette raison, dans la catégorie des jeux combinatoires.
- Le jeu se finit lorsque la position dans laquelle se trouve le joueur qui doit jouer ne permet plus de déplacement.
    - Dans une version **normale** ou **directe** du jeu, ce joueur incapable de jouer est le perdant. Dans une version **misère** ou **indirecte**, le dernier joueur qui fait un déplacement perd.
- Le jeu doit avoir une fin et chaque coup doit rapprocher les joueurs de la fin du jeu. Si un jeu ne se finit jamais, il est déclaré comme jeu nul. Donc, pour éviter cela, un jeu combinatoire édicte des règles qui éliminent la posibilité de faire un jeu nul en imposant une condition de fin. Les échecs pouvant amener à un match nul (les joueurs ne peuvent plus déplacer leurs pions) ne participent donc pas aux jeux combinatoires.

I. 3. Stratégie gagnante

Dans un jeu combinatoire, les règles déterminent une stratégie gagnante. Autrement dit, si les joueurs savent bien jouer (ils connaissent une stratégie gagnante), l'issue de la partie est en fait simplement déterminée par la position initiale. Cette stratégie gagnante se présente sous la forme d'un algorithme à suivre et de positions de jeu à occuper pour être certain de gagner, quelle que soit la riposte de l'adversaire.
Dans son étude du jeu de Nim, Charles Bouton ([1]) observe ainsi que les positions de jeu se classent en positions perdantes et positions gagnantes.
Les positions perdantes seront nommées **P-positions** car ce sont des positions gagnantes pour le **P**revious player (le joueur qui vient juste de jouer) et les positions gagnantes seront nommées **N-positions** car elles sont gagnantes pour le **N**ext player (celui qui va jouer).
La recherche de stratégie vise à attribuer à chaque position de jeu le statut P ou le statut N.

II. Jeux de Nim à un tas

Nous allons examiner intuitivement plusieurs situations de jeu de Nim au départ d'un seul tas de 13 frites.
Le nombre de positions possibles de jeu est fini : il peut rester 13, 12, 11, 10… ou 1 frite(s) et la stratégie dépend du nombre de frites restantes. Nous allons voir comment gagner à chaque coup, c'est-à-dire découvrir les **P**-positions et **N**-positions en utilisant la méthode de la « backward induction » : nous partons de la fin du jeu et remontons le fil de la partie à rebours.

II. 1. Situation N°1 : aucune règle imposée !

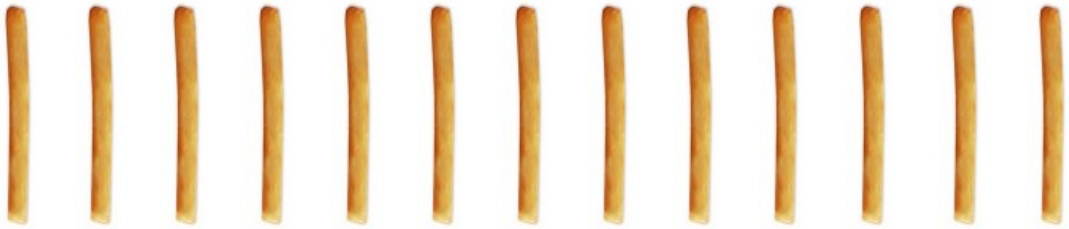
Si aucune règle n'est imposée sur le nombre de frites à prendre lors de chaque coup, le jeu ne démontre alors pas beaucoup d'intérêt ! En effet,

- Dans le **jeu direct**, le gagnant étant le joueur qui prend la dernière frite et le perdant, celui qui ne peut plus jouer, le cas est trivial car le premier joueur prendra le tas complet et aura gagné !
  Dans ce cas, la seule position perdante **P** est la situation où il ne reste plus de frite. Toutes les autres sont potentiellement gagnantes.
- Pour le **jeu inverse**, le perdant étant le joueur qui prend la dernière frite, le premier joueur gagnera aussi car il prendra toutes les frites sauf une, qu'il laissera à son adversaire et gagnera également.
  Dans ce cas, l'unique position perdante **P** est la situation où il reste exactement une frite.

[1] Voir section ***V. Le jeu de Nim à plusieurs tas***

II. 2. Situation N°2 : contrainte maximale en version directe

Cette fois, on impose une règle : on fixe à 3 le nombre maximum de frites à prélever à chaque coup.
Examinons ce qui se passe en jeu direct (le gagnant prend la dernière frite).
Voici une représentation du jeu :



13	12	11	10	9	8	7	6	5	4	3	2	1	0

Analysons les différentes positions de ce type de jeu, en les identifiant au nombre de frites restantes sur la table.

- La position **0** est forcément une position perdante **P** : le joueur suivant ne sait plus retirer de frite, il a perdu.
- Les positions **1**, **2** et **3** sont des positions gagnantes **N** : en un coup, elles peuvent amener l'adversaire à la position 0, perdante.
- **4** est une position perdante **P** : quoi que le joueur prenne, il placera son adversaire dans une position gagnante (1,2 et 3).
- **5**,**6** et **7** sont des positions gagnantes : le joueur a la possibilité de placer l'adversaire en position perdante (4).
- **8** est une position perdante **P** : quoi que le joueur prenne, il placera son adversaire dans une position gagnante (5 ,6,7)… Et ainsi de suite …

Les positions P sont en 0, 4,8, … et les positions N sont en 1, 2, 3, 5, 6, 7, …. Ce qui donne une configuration de ce type :

13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	P	N	N	N	P	N	N	N	P	N	N	N	P

La chaîne de caractère NNNP de longueur 4 se répète, et ce, quel que soit le nombre de frites au départ. Et les positions perdantes apparaissent avec une périodicité 4. Par ailleurs, il est aisé de remarquer qu'elles sont toutes divisibles par 4.

Dans ce type de configuration (13 frites au départ), il y a donc **un intérêt certain à commencer la partie** de façon à tout de suite mettre son adversaire dans une des positions perdantes décrites ci-dessus.

Une stratégie gagnante consiste donc à, premièrement, commencer la partie et ensuite placer à chaque coup son adversaire dans les positions 8, 4 puis 0. De cette façon, c'est la réussite assurée pour nous !

II. 3. Situation N°3: version misère

Que se passe-t-il si le perdant est celui qui prend la dernière frite ?

- La position **1** est par la règle du jeu une position perdante **P** .
- Les positions **2**, **3** et **4** sont des positions gagnantes **N** : en un coup, elles peuvent amener l'adversaire à la position 1, perdante.
- **5** est une position perdante **P** : quoi que le joueur prenne, il placera son adversaire dans une position gagnante (2, 3 et 4).
- **6**, **7**, **8** sont des positions gagnantes : le joueur a la possibilité de placer l'adversaire en position perdante (5).
- **9** est une position perdante **P**: quoi que le joueur prenne, il placera son adversaire dans une position gagnante (6, 7, 8)… Et ainsi de suite …

13	12	11	10	9	8	7	6	5	4	3	2	1	-
P	N	N	N	P	N	N	N	P	N	N	N	P	

La chaîne de caractère NNNP se décale d'une position. On voit alors qu'on n'a pas intérêt à commencer le jeu puisque la position initiale correspond à une position P.
Les positions 13, 9, 5 et 1 sont perdantes. Ce sont celles dont la division par 4 a pour reste 1.

II. 4. Généralisons

Maintenant que nous avons décrit la stratégie gagnante et trouvé les positions perdantes **P** pour cette situation particulière, nous allons analyser ce qu'il se passe si nous modifions certains paramètres du jeu comme le nombre total N de frites au départ ou le nombre maximum K de frites qu'on peut prendre par coup.

- **1er cas** : modification du nombre total de frites N (contrainte maximale K=3 en version directe)

Les positions perdantes **P** se répètent tous les multiples de 4. N étant indépendant de cela, notre stratégie ne va pas changer : nous allons jouer de sorte à maintenir notre adversaire dans ces P-positions. Il faudra cependant être attentif à ce que le nombre de frites au départ N ne soit pas un multiple de 4 et donc une position **P**. Le cas échéant, il faudrait laisser son adversaire commencer la partie et occuper d'entrée de jeu cette position perdante.

- **2ème cas** : modification de la contrainte maximale K (en version directe)

Si la règle impose cette fois un nombre maximum K fixé de frites à prendre à chaque coup, voici ce que nous observons :

- La position **0** est toujours évidemment une position perdante **P**.
- Mais cette fois, les positions **1**, **2** , **… K** sont des positions gagnantes **N** : en un coup, nous pouvons retirer la totalité des frites restantes et placer l'adversaire à la position 0, perdante.
- **K+1** est une position perdante **P** : le nombre maximum de frites à retirer étant K, le joueur laissera toujours au moins une frite et placera son adversaire dans une position gagnante correspondant aux positions 1 à K frites.
- **K+2**, …, **2K+1** sont des positions gagnantes **N**: le joueur ayant la possibilité de retirer entre 1 et K frites, il pourra remettre son adversaire dans la position perdante K+1.
- **2K+2=2(K+1)** est une position perdante **P**: le joueur n' ayant la possibilité que de retirer entre 1 et K frites, il laissera 2K+1 frites avec le retrait minimum permis et K+2 frites avec le retrait maximum de K frites, plaçant le joueur adverse dans l'une des positions gagnantes précédentes …

On observe donc que les positions perdantes **P** sont multiples de K+1 dans ce type de jeu.

- **3ème cas** : modification de la contrainte maximale K (en version misère)

En remontant dans le jeu à partir de la position **1** toujours perdante en version misère, on observe que les positions **K+2**, **2K+3**, **3K+4**, …, **1 + n.(K+1)** ($n \in \mathbb{N}$), … sont toutes des positions **P**, c'est-à-dire toutes celles donnant un reste égal à 1 dans la division par K+1.

III. Un peu de modulos

Les stratégies gagnantes de la section précédente utilisent les restes de la division du nombre restant de frites par K+1. C'est pourquoi des notions telles que la congruence et les modulos nous seront utiles.

III. 1. Définitions et notations

Soit *a* et *b* deux entiers et *n* un entier naturel non nul.
On dit que *a* est **congru** à *b* modulo *n* si et seulement si *a-b* est divisible par *n*.

On écrit alors $a \equiv b(n)$ ou $a \equiv b(mod\ n)$ ou encore $a \equiv b$ ***modulo*** n

Remarque : dans le cas contraire, on dit que « a est non congru à b modulo n ».

$$\forall a, b \in \mathbb{Z}, \forall n \in \mathbb{N}_0 , a \equiv b \ (\text{mod } n) \Leftrightarrow \exists k \in \mathbb{Z} : a = kn + b$$

- **Exemple :** $28 \equiv 13(mod\ 3)$ *car* $28=5.3 + 13$
  tout comme $28 \equiv 1\ (mod\ 3)$ *car* $28=9.3 + 1$

Cependant, si $a \equiv b(mod\ n)$, b n'est le reste de la division de a par n que si $0 \leq b < n$.

On peut alors définir l'opération mathématique suivante :

$$\forall a \in \mathbb{Z}, \forall n \in \mathbb{N}_0 , b=a \ (\text{mod } n) \Leftrightarrow \exists k \in \mathbb{Z} : a = kn + b \text{ et } 0 \leq b < n.$$

Dans ce cas, b est le reste de la division de a par n. On l'appelle « **résidu de a modulo n** ».

- **Exemples :**

  Si a est pair, $a(mod2)=0$ car $\exists k \in \mathbb{Z} : a = 2k$.
  Et si a est impair, $a(mod2)=1$ car $\exists k \in \mathbb{Z} : a = 2k +1$.

III. 2. Notion de classe

Les nombres entiers peuvent être classés selon leur congruence.
Par exemple, 4, 7, 25, 10, … sont tous congrus modulo 3.
Le reste de leur division euclidienne par 3 vaut 1.

1 = 4 (mod 3)= 7 (mod 3)= 25 (mod 3)= 10 (mod 3)

Ces entiers font partie d'un même ensemble, appelé **classe**. Ici, 4, 7, 25 et 10 en modulo 3 appartiennent à la classe de résidu égale à 1. On notera cette classe $\overline{1_3}$.

La définition d'une **classe** est donc : $\overline{c_m}$ ={c+k.m|k ∈ ℤ} tel que $\overline{c_m}$ désigne à la fois la classe (ensemble de nombres congrus modulo m) et son résidu.

Par ailleurs, il est à noter qu'il existe m classes en modulo m : $\overline{0_m}, \overline{1_m}, \overline{2_m}, \ldots, \overline{(m-1)_m}$ et respectivement m représentants de ces classes : 0, 1, …, m-1.

On définit alors l'ensemble des entiers représentant les classes de modulo m:

$$\mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\} = \{c \in \mathbb{N} \mid a=c+km,\ c<m,\ a\in\mathbb{Z},\ k\in\mathbb{Z}\}$$

III. 3. Opérations en modulos

On peut montrer que l'addition et la multiplication sont compatibles avec la congruence :

(1) (a+b) (mod m) = (a (mod m) + b (mod m))(mod m)
(2) (a.b) (mod m) = (a (mod m) . b (mod m))(modm)

Ceci permet de définir l'addition et la multiplication dans $\mathbb{Z}_m$:

$+ : \mathbb{Z}_m X \mathbb{Z}_m \to \mathbb{Z}_m : (\bar{a},\bar{b}) \to \bar{a}+\bar{b} = \overline{a+b}$	$. : \mathbb{Z}_m X \mathbb{Z}_m \to \mathbb{Z}_m : (\bar{a},\bar{b}) \to \bar{a}.\bar{b} = \overline{a.b}$
• Interne et partout définie $\overline{a+b} \in \mathbb{Z}_m, \forall \bar{a}, \bar{b} \in \mathbb{Z}_m$	• Interne et partout définie $\overline{a.b} \in \mathbb{Z}_m, \forall \bar{a}, \bar{b} \in \mathbb{Z}_m$
• Commutative $\forall \bar{a}, \bar{b} \in \mathbb{Z}_m : \bar{a}+\bar{b} = \bar{b} + \bar{a}$	• Commutative $\forall \bar{a},\bar{b} \in \mathbb{Z}_m : \bar{a}.\bar{b} = \bar{b}.\bar{a}$
• Associative $\forall \bar{a}, \bar{b}, \bar{c} \in \mathbb{Z}_m : (\bar{a}+\bar{b})+\bar{c} = \bar{a}+(\bar{b}+\bar{c})$	• Associative $\forall \bar{a}, \bar{b},\bar{c} \in \mathbb{Z}_m : (\bar{a}.\bar{b}).\bar{c} = \bar{a}.(\bar{b}.\bar{c})$
• Elément neutre $\forall \bar{a} \in \mathbb{Z}_m : \bar{a} + \bar{0} = \bar{0} + \bar{a} = \bar{a}$	• Elément neutre $\forall \bar{a} \in \mathbb{Z}_m : \bar{a}.\bar{1} = \bar{1}.\bar{a} = \bar{a}$
• Inverse $\forall \bar{a} \in \mathbb{Z}_m : \bar{a} + \overline{m-a} = \bar{0}$	• Distributive par rapport à + $\forall \bar{a}, \bar{b},\bar{c} \in \mathbb{Z}_m : (\bar{a}+\bar{b}).\bar{c} = \bar{a}.\bar{c} + \bar{b}.\bar{c}$

III. 4. Stratégie gagnante du Nim à un tas en termes de modulos

Nous avons découvert dans la section précédente la stratégie à adopter pour gagner à coup sûr au jeu de Nim à un tas.

Reprenons l'exemple du tas de 13 frites dans lequel on peut piocher avec une contrainte maximale de 3, joué dans la version directe.

Nous avons vu que les positions perdantes de jeu sont les positions 0, 4, 8, et 12. En modulo 4, toutes ces positions ont un résidu nul.

Pour mieux le constater, voici un tableau présentant en ligne 1 les positions possibles de jeu (les frites restantes) x_i, en ligne 2, le marquage des positions perdantes et gagnantes et en ligne 3, le résidu modulo 4 des x_i .

13	**12**	11	10	9	**8**	7	6	5	**4**	3	2	1	**0**
N	**P**	N	N	N	**P**	N	N	N	**P**	N	N	N	**P**
1	**0**	3	2	1	**0**	3	2	1	**0**	3	2	1	**0**

Toutes les positions appartenant à $\overline{0_4}$ sont les positions perdantes.

Nous pouvons ainsi généraliser la stratégie gagnante pour le jeu de Nim à un tas à contrainte maximale K.

- En version directe (le gagnant prend la dernière frite) : Les positions **P** appartiennent à $\overline{0_{K+1}}$
- En version misère (le perdant prend la dernière frite) : Les positions **P** appartiennent à $\overline{1_{K+1}}$

IV. Un peu de graphes

L'ensemble des configurations d'un jeu de Nim, même s'il est très grand, peut être représenté par un graphe fini. Nous allons voir que les positions P forment une structure graphique, nommée **noyau** par les mathématiciens et qu'à chaque position de jeu, peut être affecté un nombre, que nous appellerons **nimber**, donné par la **fonction de Grundy**.

IV. 1. Notion de graphe

Un graphe est une structure mathématique donnée sous la forme d'un couple (X,U) qui comprend un ensemble de **sommets** X, et un ensemble de paires de sommets, nommées arêtes, U.

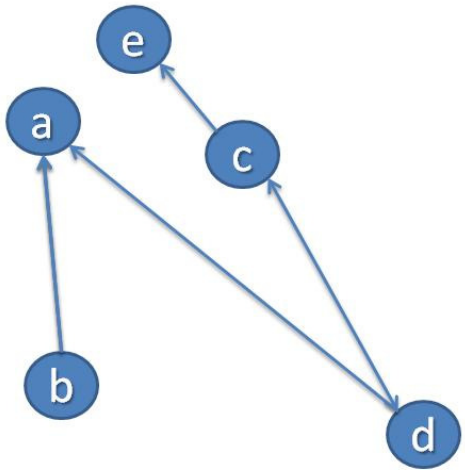
Un jeu tel que le jeu de Nim peut être représenté par un graphe orienté : l'ordre dans lequel on précise les paires de sommets de U est important, les arêtes se comportant alors comme des flèches, qu'on appelle **arcs**.

Dans le graphe associé à un jeu de Nim, les sommets représentent les positions de jeu et les arcs sortant d'un sommet et entrant dans un autre représentent les coups qui font passer d'une position à une autre.

Exemple de graphe orienté :

Nous pouvons décrire le graphe ci-contre comme suit :

G = ({a,b,c,d,e} ; {(b,a), (d,a), (d,c), (c,d), (c,e)}) c'est-à-dire X={a,b,c,d,e} et U={(b,a), (d,a), (d,c), (c,d),(c,e) }



Deux sommets x et y d'un graphe G=(X, U) sont **adjacents** si une arête ou un arc les relie directement. Autrement dit, si (x,y) Є U et/ou (y ,x) Є U.

Dans notre exemple, les sommets a et b sont adjacents. Par contre les sommets a et c ne le sont pas. On utilise une fonction multivoque de X dans X pour décrire le fait qu'un sommet est envoyé sur un autre sommet (par exemple, que b est envoyé sur a dans notre situation ou encore, que a est le successeur de b).

$$\Gamma : \quad X \rightarrow \quad X : x \rightarrow \Gamma(x) = \{y \in X |\ (x,y) \in U\}$$

Dans notre exemple, $\Gamma(d) = \{a,c\}$.

Un ensemble **intérieurement stable** d'un graphe est un ensemble de sommets de X non adjacents deux à deux.

Un ensemble **extérieurement stable** d'un graphe G = (X, U), (appelé aussi ensemble absorbant) est un sous-ensemble $E \subseteq X$ de sommets x tel que $\forall x \notin E, \exists\ y \in E, y \in \Gamma(x)$: il existe toujours au moins un arc reliant x extérieur à E à un sommet quelconque de E.

Dans notre exemple, {a,c, e} forme un ensemble intérieurement et extérieurement stable, mais {a, b} n'est ni l'un ni l'autre puisque a et b sont adjacents et que ni a ni b n' « absorbent » le sommet c.

IV. 2. Notion de noyau

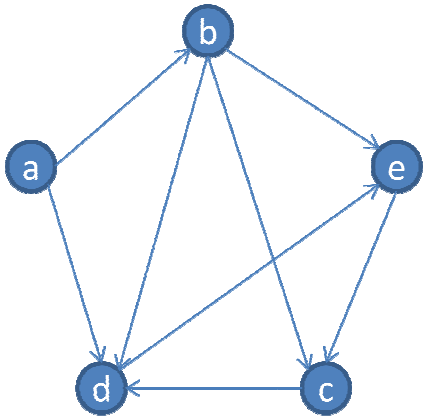
Une notion fondamentale de la théorie des jeux est ce qu'on appelle le **noyau**.

Un noyau est un sous-ensemble de sommets, à la fois extérieurement stable et intérieurement stable. Autrement dit, quel que soit le sommet du graphe hors du noyau, il existe toujours un arc qui relie ce sommet à un sommet du noyau (le noyau est extérieurement stable ou absorbant) et il n'existe pas d'arc reliant deux sommets du noyau entre eux (le noyau est intérieurement stable).

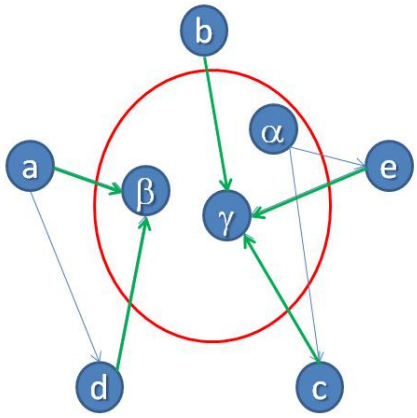
Définition : $K \subseteq X$ est un noyau $\Leftrightarrow$

1. $\forall x \in K : \Gamma(x) \cap K = \emptyset$
2. $\forall x \notin K : \Gamma(x) \cap K \neq \emptyset$

Exemple A



Exemple B



L'exemple A ne possède aucun noyau : {a, c} forme bien un ensemble stable intérieurement (aucun arc ne lie ces sommets entre eux) mais pas extérieurement car, si b et e sont bien absorbés par c ou par a, il n'y a pas d'arc qui envoie d sur l'un d'eux.

La partie entourée de l'exemple B $\{\alpha, \beta, \gamma\}$ représente bien le noyau du graphe : c'est un ensemble de sommets à l'intérieur duquel il n'y a aucun échange. Pour tout sommet étant extérieur à cet ensemble, il y a toujours au moins une flèche qui permet d'y revenir directement.

Le noyau détermine la stratégie gagnante et les positions P. Rappelez-vous que l'objectif est de maintenir l'adversaire dans une situation perdante. Rien de tel que le noyau pour y parvenir : si un joueur A se trouve dans une position appartenant au noyau, quoi qu'il fasse, il devra en sortir puisqu'aucun arc ne relie deux sommets du noyau entre eux (stabilité intérieure). Par ailleurs, le joueur B, en raison de la stabilité extérieure, aura toujours un coup qui ramènera le jeu à une configuration du noyau. Le joueur B pourra ainsi gagner la partie quoi que joue le joueur A.

IV. 3. Fonction de Grundy et nimbers

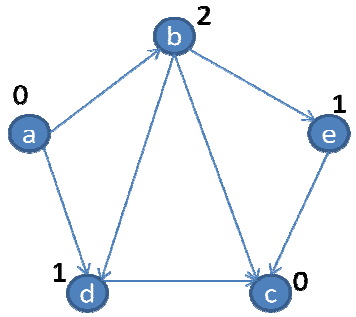
Trouver les positions P, ou trouver le noyau, peut parfois s'avérer difficile. Une fonction attribuant une valeur entière à chaque sommet va donc nous être utile. Ces valeurs entières sont nommées **nimbers** et si elles sont nulles, le sommet correspondant appartient au noyau.

C'est la **fonction de Grundy**, introduite par P. M. Grundy en 1939, qui attribue à chaque sommet son nimber. Il s'agit donc d'une fonction des sommets de G= (X,U) dans les naturels $\mathbb{N}$ tel que $x \in X$ est envoyé sur le plus petit naturel qui est différent de toutes les valeurs de la fonction de Grundy des successeurs du sommet x. Autrement dit :

$$g : X \to \mathbb{N} : \ x \to g(x)=\min(\mathbb{N} \setminus \{g(y) \mid y \in \Gamma(x)\})$$

Pour trouver la fonction de Grundy d'un graphe, il faut procéder de façon récursive en sachant, bien entendu, qu'un sommet sans successeur a pour nimber 0.

Recherchons la fonction de Grundy et les nimbers dans le graphe suivant :



$\Gamma(c)=\emptyset$, donc g(c)=0.

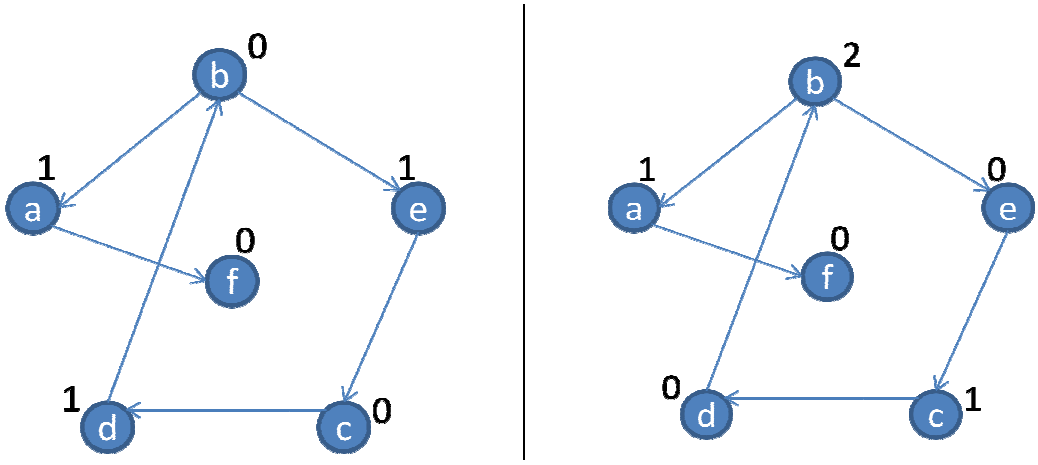
De là, g(e) =1 car il amène à 0

g(d)=1 car il amène en c.

g(b)=2 car il amène en c, d et en e.

g(a) amène en b et en d. Donc, on ne peut lui attribuer les nimbers 1 et 2. Mais 0 est plus petit, donc g(a)=0.

La fonction de Grundy n'est pas unique. L'exemple ci-dessous montre deux fonctions de Grundy possibles pour le même graphe:



Théorème ([2]): Si un graphe G=(X,U) possède une fonction de Grundy g et K⊆X est le noyau, ∀ x ∈ X, g(x)=0⇔x∈K.

Démonstration :

Soit G=(X,U) un graphe et K = {x ∈ X| g(x)=0}. Montrons que K est un noyau de G.

1. ∀x ∈ K : Γ (x)∩K =∅ :

Soit y ∈ Γ (x). Par définition, g(y)≠g(x)=0. Donc, y ∉ K.

2. ∀x ∉ K : Γ (x)∩K ≠∅

Par définition de la fonction de Grundy, ∀x ∈ X, si g(x)= n, alors ∀ p, 0≤p<n, ∃ y ∈ Γ (x), g(y)=p. En particulier, si n>0, ∃ y ∈ Γ (x), g(y)=0, c'est-à-dire y ∈ K. Donc, Γ (x)∩K ≠∅. ❑

Dans l'exemple précédent, {b, c, f} et {e, d, f} sont des noyaux.

IV. 4. Noyau et Nimbers dans le jeu de nim à un tas

Reprenons à présent nos jeux de Nim à un tas. Nous allons y illustrer les concepts de noyau et de fonction de Grundy.

Pour rappel, le noyau détermine à présent l'ensemble des positions perdantes P, qui ont toutes dès lors un nimber nul.

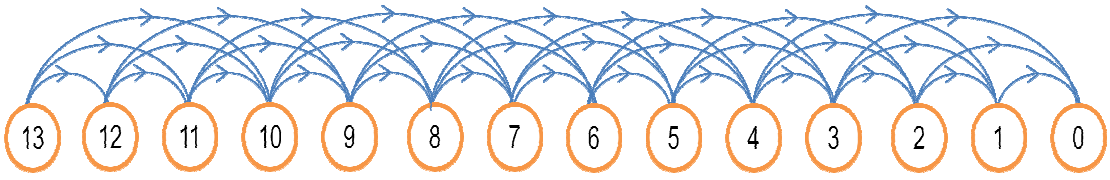
Pour jouer et gagner, la méthode est de tenter de placer son adversaire dans une position dont le nimber est 0, et dès que c'est fait, de le maintenir toujours dans des positions de nimber 0.

[2] Adaptation de [9]

Reprenons l'exemple où nous avions 13 frites, une contrainte de 3 frites à enlever au maximum, dans un jeu direct (le gagnant prend la dernière frite).

Nous avions identifié les positions P perdantes comme appartenant à $\overline{0_4}$. Le noyau sera donc composé de tous les multiples de 4, c'est-à-dire {0, 4,8,..}.

Tâchons de représenter la situation à l'aide d'un graphe orienté et découvrons la fonction de Grundy g(x) de façon récursive en commençant par la position 0, qui nous le savons est perdante et qui n'a pas de successeur. Son nimber est donc nul et voici ce que nous obtenons :



Nous avons donc :

- g(0)=0 car Γ (0) =Ø
- g(1) = 1 car Γ (1) ={0} et g(0)=0.
- g(2) = 2 car Γ (2) ={0,1} et g(0)=0, g(1)=1.
- g(3) = 3 car Γ (3) ={0,1,2} et g(0)=0, g(1)=1 et g(2)=2.
- g(4) = 0 car Γ (4) ={1,2,3} et g(x)>0 ∀x∈ Γ (4).
- g(5) = 1 car Γ (5) ={2,3,4} ...

En bref, on s'aperçoit que **g(x)=xmod4** dans le cas de ce type de jeu. On retrouve alors les positions P quand g(x)=xmod4=0.

Remarquez que le même jeu en version misère aurait eu comme position ultime le « 1 », laisser la dernière frite à l'adversaire sonnant la victoire du dernier joueur.

Ce qui aurait valu la fonction de grundy suivante :

- g(1)=0 car Γ (1) =Ø
- g(2) = 1 car Γ (2) ={1} et g(1)=0.
- g(3) = 2 car Γ (3) ={1,2} et g(2)=1, g(1)=0.
- g(4) = 3 car Γ (4) ={1,2,3} et g(1)=0, g(2)=1 et g(3)=2.
- g(5) = 0 car Γ (5) ={2,3,4} et g(x)>0 ∀x∈ Γ (5).
- g(6) = 1 car Γ (6) ={3,4,5} ...

Ce qui donne en version misère, x (x>0) étant le nombre de frites restant sur la table :

g(x)=(x-1)mod4

Nous pouvons alors généraliser la fonction de grundy pour le jeu de Nim à un tas à contrainte maximale K.

- En version directe (le gagnant prend la dernière frite) : g(x)=xmod(K+1)
- En version misère (le perdant prend la dernière frite) : g(x)=(x-1)mod(K+1)

IV. 5. Nimbers d'autres nims à un tas

Grâce à la fonction de Grundy et aux nimbers, nous pouvons nous laisser aller à toutes les fantaisies …

Par exemple, imposons une contrainte bornée m≤K≤M où m et M sont des naturels et jouons en version directe.

Pour fixer les idées, reprenons le tas de 13 frites, dans lequel on peut puiser entre 3 et 5 frites. Que deviennent les positions P, le noyau et la fonction de Grundy dans ce cas ?

13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	0	2	2	1	1	1	0	0	0

- g(0)=g(1)=g(2)=0 car Γ (0) = Γ (1) = Γ (2)= $\emptyset$. En effet, devant retirer au moins 3 frites, les positions 0, 1 et 2 ne nous permettent plus de jouer. Elles n'ont donc pas de successeur en terme de graphe.
- g(3) = 1 car Γ (3) ={0}
- g(4) = 1 car Γ (4) ={0,1}
- g(5) = 1 car Γ (5) ={0,1,2}
- g(6) = 2 car Γ (6) ={1,2,3}
- g(7) = 2 car Γ (7) ={2,3,4}
- g(8) = 0 car Γ (8) ={3,4,5}
- et on recommence …

On a une figure qui se répète avec une périodicité 8 (=5+3).

Les positions perdantes P appartiennent aux classes $\overline{0_8}$, $\overline{1_8}$, $\overline{2_8}$.

Le noyau de ce jeu à contrainte bornée m≤K≤M où m et M sont des naturels en version directe rassemble les positions **P** telles que **xmod(m+M) < m**, x étant le nombre de frites restant sur la table.

Remarque : Dans ce type de jeu, arriver aux positions 1 ou 2 revient à perdre : celui qui se retrouve dans ces positions ne peut plus jouer et a perdu.

Dans un autre exemple, imaginons que dans un jeu direct au départ de 13 frites, seulement certains retraits soient possibles. Imaginons ainsi que nous ne puissions retirer que 1, 2 ou 4 frites au jeu. Quels sont alors le noyau et les P-positions ?

13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	2	1	0	2	1	0	2	1	0	2	1	0

- g(0)=0 car Γ (0) =Ø
- g(1) = 1 car Γ (1) ={0}
- g(2) = 2 car Γ (2) ={0,1}
- g(3) = 0 car 0 ∉ {g(y)|y∈ Γ (3)}
- g(4) = 1 car Γ (4) ={0,2,3}
- g(5) = 2 car Γ (5) ={1,3,4}
- g(6) = 0 car Γ (6) ={2,4,5}
- et on recommence …

On remarque une répétition des 3 chiffres 0-1-2. Les positions perdantes P appartiennent à la classe $\overline{0_3}$.

Nos tribulations empiriques ne nous ont pas permis de trouver une loi qui permettrait de généraliser l'identification des P-positions ou le calcul des nimbers.

V. Le jeu de Nim à plusieurs tas

V. 1. De nouvelles règles

Dans les paragraphes précédents, nous avons abordé le jeu de Nim de façon simple en n'utilisant qu'une seule rangée de pions (frites). Cependant, la variante la plus répandue du jeu de Nim consiste à disposer les pions en plusieurs tas, pas forcément de même effectif. A tour de rôle, chaque adversaire prélève autant de pions qu'il le souhaite dans un tas de son choix, sans contrainte.

C'est cette version du jeu de Nim qui est mise à l'honneur dans le film «L'année dernière à Marienbad ». Le héros gagne toutes les parties et prononce cette phrase qui évoque la stratégie : « *Je puis perdre mais je gagne toujours* ».

Dans cette section, nous allons montrer qu'il existe une stratégie gagnante pour le jeu de Nim généralisé.

Pour cela, nous allons suivre le fil de l'histoire : le jeu de Nim à plusieurs tas est très important dans l'histoire des jeux combinatoires, puisque c'est le premier qui a été résolu. En effet, au début du vingtième siècle, et sans connaître les notions de noyaux ou de fonction de Grundy, Charles Bouton, de l'Université d'Harvard, a découvert une stratégie gagnante. C'est lui, le premier, qui a mis en exergue les statuts ***favorable*** et ***défavorable*** pour les positions de jeu (positions N et P) démontrant qu'un joueur dans une position favorable possédait toujours un coup qui lui permettait de gagner, alors que le joueur en situation défavorable, pouvait jouer n'importe quel coup et placer malgré tout, son adversaire dans une situation favorable. Sa découverte est connue sous le nom de ***Théorème de Bouton***.

Avant de démontrer ce théorème, nous aborderons les notions théoriques utiles à la preuve comme la décomposition binaire et la nim-somme que nous illustrerons dans un exemple de partie de jeu.

Nous complèterons la démonstration du Théorème de Bouton, par un théorème plus général sur la recherche de stratégie gagnante dans les jeux combinatoires, dont la paternité conjointe est attribuée à ***R. P. Sprague*** (1935) and ***P. M. Grundy*** (1939).

Enfin, nous appliquerons ce théorème dit de Sprague et Grundy à une version plus personnelle du Jeu de Nim généralisé, que nous avons (pompeusement) nommée « **Jeu de VanLoGo** ».

V. 2. Ecriture en système binaire

La démonstration du théorème de Bouton repose en partie sur l'écriture de nombres en base 2.

Écrire un nombre en base deux, c'est-à-dire utiliser le système binaire revient à décomposer ce nombre en une somme de puissances de 2 : 1, 2, 4, 8, 16, 32, ...

Le système que nous utilisons communément en mathématique est le système décimal (par correspondance naturelle avec les dix doigts de la main) : chaque nombre est décomposé suivant les puissances de dix et peut utiliser les 10 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.

Mais tout nombre peut se décomposer suivant une autre base.

Définition générale :

Dans un système de numération de base b ($b \in \mathbb{N}_0$) le nombre $\overline{a_n a_{n-1} \ldots a_2 a_1 a_0}^{b}$ représente $a_n.b^n + a_{n-1}.b^{n-1} + \ldots + a_2.b^2 + a_1.b^1 + a_0.b^0$

avec $\forall i \in N; 0 \le i \le n : 0 \le a_i < b$.

Le système binaire utilise la base 2 et chaque nombre fait donc apparaître une suite de symboles 1 et 0 ([3]).

Par exemple, le nombre n=103 (dans notre écriture usuelle) se décompose et s'écrit comme suit

- Dans le système décimal : n= **1**.10² + **0**.10¹ + **3**.10⁰= $\overline{103}^{10}$
- Dans le système binaire : n= **1**.2^6 + **1**. 2^5 + **0**.2^4 + **0**. 2^3+ **1**. 2^2 + **1**.2^1 + **1**. 2^0= $\overline{1100111}^{2}$

A l'inverse, le nombre écrit $\overline{1001101}^{2}$ dans le système binaire correspond à $\overline{77}^{10}$ dans le système décimal.

En quoi cette décomposition et écriture binaire intervient-elle dans la recherche d'une stratégie gagnante au jeu de Nim à plusieurs tas ?

Nous allons vous le montrer au départ d'un exemple de partie de jeu de Nim en disposant 17 frites en cinq tas comprenant respectivement 6, 4, 3, 1 et 3 frites. La première étape pour trouver une stratégie gagnante consiste à décomposer ces nombres dans le système binaire de la façon suivante :

=6 frites	=**1**.2^2+**1**.2^1+**0**.2^0	= $\overline{110}^{2}$
=4 frites	=**1**.2^2+**0**.2^1+**0**.2^0	= $\overline{100}^{2}$
=3 frites	=**0**.2^2+**1**.2^1+**1**.2^0	= $\overline{011}^{2}$
=1 frite	=**0**.2^2+**0**.2^1+**1**.2^0	= $\overline{001}^{2}$
=3 frites	=**0** .2^2+**1** .2^1+**1**.2^0	= $\overline{011}^{2}$

[3] Nous ne considèrerons ici que le cas utile et simple des nombres naturels.

Pour plus de lisibilité par la suite, nous pouvons également présenter ce résultat sous forme de matrice. Si bien qu'à chaque configuration de jeu peut correspondre une matrice ne comportant que les chiffres 0 et 1, que nous baptiserons « matrice-nim ».

Soit donc la matrice-nim associée à cette configuration du jeu :

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

V. 3. La nim-somme

Pour savoir comment gagner, Charles Bouton a utilisé un procédé de sommation particulier appelé la **nim-somme** ou encore **somme digitale**.

Définition : Notant $\oplus$ l'opérateur somme digitale (aucune convention en la matière n'existe), nous avons $\overline{a_n...a_1a_0}^2 \oplus \overline{b_n...b_1b_0}^2 = \overline{c_n...c_1c_0}^2$ où $\forall i \in N; i \le n : c_i = (a_i + b_i) \bmod 2$.

Ainsi, par exemple, calculons la nim somme de 17 et 9 : $\overline{10001}^2 \oplus \overline{01001}^2 = \overline{11000}^2$

La recherche de stratégie gagnante repose sur la nim-somme des termes de chaque colonne de la matrice-nim. Dans notre exemple, cela donne :

$$\begin{array}{rrr} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ \oplus\ 0 & 0 & 1 \\ \underline{0} & \underline{1} & \underline{1} \\ 0 & 1 & 1 \end{array}$$

Les P-positions (c'est-à-dire les positions défavorables) sont celles donnant lieu à une nim-somme nulle. Ici, nous avons 2 fois le chiffre 1 qui apparaît dans la nim-somme : cela correspond donc à une position N favorable. Pour placer notre adversaire en position P, il faut rendre la nim-somme nulle. Nous devons éliminer les « 1 » qui composent la somme actuelle, c'est-à-dire retirer $\overline{11}^2 = \overline{3}^{10}$ frites sans pour autant modifier les chiffres de la première colonne (déjà à 0). Nous pouvons y parvenir en retirant par exemple les frites de la 3ème rangée :

$$\begin{array}{rrr} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ \oplus\ 0 & 0 & 1 \\ \underline{0} & \underline{1} & \underline{1} \\ 0 & 1 & 1 \end{array}$$

J'enlève 3 frites dans la 3ème rangée.

$$\begin{array}{rrr} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ \oplus\ 0 & 0 & 1 \\ \underline{0} & \underline{1} & \underline{1} \\ 0 & 0 & 0 \end{array}$$

L'adversaire se trouve bloqué en position P. Nous allons montrer grâce au Théorème de Bouton, que quoi qu'il fasse, la position suivante sera une position N. Nous devons continuer à jouer de cette façon pour gagner, c'est-à-dire toujours laisser l'adversaire dans une configuration de nim-somme nulle (P-position du noyau), jusqu'à ce qu'il ne reste plus de frite.

Propriétés de la Nim-Somme :

Les propriétés de la Nim-somme découlent des propriétés de l'addition dans $\mathbb{Z}m$.

La Nim-somme est

- *Commutative* : $\overline{a_n...a_1a_0}^2 \oplus \overline{b_n...b_1b_0}^2 = \overline{b_n...b_1b_0}^2 \oplus \overline{a_n...a_1a_0}^2$
- *Associative* : $\left(\overline{a_n...a_1a_0}^2 \oplus \overline{b_n...b_1b_0}^2\right) \oplus \overline{c_n...c_1c_0}^2 = \overline{a_n...a_1a_0}^2 \oplus \left(\overline{b_n...b_1b_0}^2 \oplus \overline{c_n...c_1c_0}^2\right)$
- *Possède un neutre :* $\left(\overline{a_n...a_1a_0}^2 \oplus \overline{0}^2\right) = \overline{a_n...a_1a_0}^2 \oplus \left(\overline{0}^2 \oplus \overline{a_n...a_1a_0}^2\right)$
- *Chaque élément a un inverse :*

$$\forall \overline{a_n...a_1a_0}^2 \in \mathbb{N}, \exists\ \overline{a'_n...a'_1a'_0}^2 \in \mathbb{N} : \overline{a_n...a_1a_0}^2 \oplus \overline{a'_n...a'_1a'_0}^2 = \overline{0}^2$$

C'est-à-dire, $\forall i \in \{1,...,n\}\ \left(a_i = 0 \Rightarrow a'_i = 0\right) \wedge \left(a_i = 1 \Rightarrow a'_i = 1\right)$

V. 4. Généralisons

Soit **n** frites disposées en **k** rangées de frites, la i[ème] rangée contenant $\mathbf{n_i}$ frites ($1 \le i \le k$).

Ecrivons $n_1,...,n_k$ sous la forme d'une décomposition binaire : il existe $\alpha_{i,j} \in \{0;1\}$ tels que

Pour $1 \le i \le k$ $\quad n_i = \sum_{j=0}^{p} \alpha_{i,j} 2^j$, p étant la plus haute puissance de 2 de max(n_i).

Considérons alors la matrice-nim A à k lignes et p+1 colonnes :

$$A = \begin{pmatrix} \alpha_{1,p} & \cdots & \alpha_{1,0} \\ \alpha_{2,p} & \cdots & \alpha_{2,0} \\ \vdots & \ddots & \vdots \\ \alpha_{k,p} & \cdots & \alpha_{k,0} \end{pmatrix}$$

Pour $j \in [0;p]$, soit s_j la somme modulo 2 des termes de la colonne d'indice j :

$$s_j = \left(\sum_{i=1}^{k} \alpha_{i,j}\right) mod\ 2$$

Le calcul de la Nim-somme permet de montrer le statut N ou P de la position de jeu. Si cette nim-somme est nulle, c'est-à-dire si $s_j=0\ \ \forall j \in \{0,...,p\}$, ou encore si la position initiale est telle que $\sum_{i=1}^{k} \alpha_{i,j}$ est paire $\forall j \in \{0,...,p\}$, on doit laisser notre adversaire jouer le premier car une nim-somme nulle correspond à une P-position (perdante) du noyau, sinon on débute la partie.

La suite du jeu se déroulera de telle sorte que notre adversaire soit toujours placé dans une P-position perdante de somme-nulle jusqu'à ce qu'il ne reste plus de frite.

V. 5. Le théorème de Bouton

Nous allons à présent justifier la stratégie gagnante. Pourquoi la mise en œuvre du « *Je puis perdre mais je gagne toujours* » se traduit-elle en nim-somme nulle dans le jeu de Nim à plusieurs tas ? Dit autrement:

(a) Pourquoi, en opérant comme on vient de l'indiquer à partir d'une situation dont la nim-somme est non nulle, on finit par gagner ?

(b) Pourquoi, si on se trouve dans une position de nim-somme nulle, on perd toujours face à un adversaire qui connaît et applique la règle de la *Nim-somme* ?

C'est Charles Bouton, mathématicien américain, qui répondit à cette double question en 1902 au travers d'un théorème. Bouton remarqua qu'une position du jeu de Nim possédait toujours soit le statut favorable soit le statut défavorable pour le joueur qui allait jouer. C'est lui qui initia les P et N-positions et qui découvrit que les P-positions coïncidaient avec les nim-sommes nulles. Bien que Bouton ne connaisse ni les notions de noyau ni la fonction de Grundy (puisque Sprague et Grundy lui sont postérieurs), c'est en terme de noyau que nous allons démontrer son théorème, puisque le noyau regroupe l'ensemble des P-positions.

Théorème ([4]) : *Si K désigne l'ensemble des positions présentant une nim-somme nulle, alors K est un noyau.*

Adoptons les notations suivantes :

- X l'ensemble des positions possibles dans le jeu
- Γ(x) l'ensemble des positions possibles issues de x Є X.
- G = (X,U) où U={(x,y) | x Є X et y Є Γ(x) } le graphe sur X.

Par définition, si l'ensemble K est un noyau, il doit satisfaire deux conditions ([5]):

1. $\forall x \in K, \Gamma(x) \cap K = \varnothing$ *Je sors toujours du noyau*

   La première impose que si x est une position appartenant à K alors toutes les positions issues de x seront hors de K ; autrement dit, quelque soit le coup joué, la position sera hors de l'ensemble K.

   Ce qui revient en terme de nim-somme à dire : une position de nim-somme nulle est toujours suivie par une position de nim-somme non nulle, et ce, quelque soit le coup joué. Nous parlerons de situation paire.

2. $\forall x \in X \setminus K, \Gamma(x) \cap K \neq \varnothing$ *Je peux toujours revenir dans le noyau*

   La seconde condition impose que si x est une position hors de K alors il existe une position issue de x appartenant à K; autrement dit il existe un coup amenant une position de jeu appartenant à l'ensemble K.

   Ce qui revient en terme de nim-somme à dire : il est possible par prélèvement de transformer une position de nim-somme non nulle en une position de nim-somme nulle. Nous parlerons de situation impaire.

[4] Adaptation d'une démonstration tirée de [13]
[5] Se référer à la section IV.2

1. Soit à montrer que : $\forall x \in K, \Gamma(x) \cap K = \varnothing$

Hypothèse : Soit x ∈ K : x présente par définition une nim-somme nulle.

$$s_j = 0 \quad \forall j \in \{0,...p\}$$

Thèse : $\Gamma(x) \cap K = \varnothing$, c'est-à-dire $\forall y \in \Gamma(x), y \notin K$

Démonstration :

- Si $\Gamma(x) = \varnothing$, on a bien $\Gamma(x) \cap K = \varnothing$. Ceci correspond à la dernière position de la partie : il ne reste plus de frites dans la position x, le joueur suivant ne peut plus jouer ! Cette situation amène bien entendu une nim-somme nulle : 0⊕0⊕0⊕…⊕0=0 !
- Supposons $\Gamma(x) \neq \varnothing$

  Le joueur suivant peut jouer ($\Gamma(x) \neq \varnothing \Leftrightarrow \exists y \in X : y \in \Gamma(x)$) : supposons qu'il enlève **c** frites de la k^{ème} rangée.

  Dans la position x, notons n_k le nombre de frites de la rangée k.

  On doit avoir c ≤ n_k. Décomposons donc n_k et c en puissances de 2 :

$$n_k = \alpha_{k,p}2^p + ... + \alpha_{k,3}2^3 + \alpha_{k,2}2^2 + \alpha_{k,1}2^1 + \alpha_{k,0}2^0 = \sum_{j=0}^{p} \alpha_{k,j} 2^j$$

$$c = c_p 2^p + ... + c_3 2^3 + c_2 2^2 + c_1 2^1 + c_0 2^0 = \sum_{j=0}^{p} c_j 2^j$$

$$\text{avec } \alpha_{k,j} \in \{0,1\} \text{ et } c_j \in \{0,1\} \quad \forall j \in \{0,...p\}$$

  La nouvelle position obtenue à partir de x en enlevant c frites dans la k^{ème} rangée est telle que :

  - le nombre de frites dans les autres rangées n'est pas modifié.
  - le nombre de frites dans la k^{ème} rangée est $n'_k = n_k - c$

  On a $$n'_k = \sum_{j=0}^{p} (\alpha_{k,j} - c_j) 2^j$$

  Durant notre démonstration nous utiliserons le résultat suivant (= prop. I) :

$$2^0 + 2^1 + 2^2 + ... + 2^k < 2^{k+1} \quad \forall k \in \mathbb{N}$$

  Nous le montrons rapidement :

$$2^0 + 2^1 + 2^2 + ... + 2^k = 2.(2^0 + 2^1 + 2^2 + ... + 2^k) - (2^0 + 2^1 + 2^2 + ... + 2^k)$$
$$= 2^1 + 2^2 + ... + 2^{k+1} - (2^0 + 2^1 + 2^2 + ... + 2^k)$$
$$= 2^{k+1} - 2^0 = 2^{k+1} - 1 < 2^{k+1}.$$ ❑

Soit i, l'indice le plus petit tel que $c_i = 1$

Premier cas : supposons que $\alpha_{k,i} = 1$

On a alors $n'_k = \sum_{j>i}(\alpha_{k,j} - c_j)2^j + \mathbf{0x2^i} + \sum_{j<i}\alpha_{k,j}2^j$

Or, **(1)** $\sum_{j>i}(\alpha_{k,j} - c_j)2^j \geq 0$

En effet, en procédant par l'absurde,

Si nous avions $\sum_{j>i}(\alpha_{k,j} - c_j)2^j < 0$ c'est-à-dire $\sum_{j>i}(c_j - \alpha_{k,j})2^j > 0$

Nous aurions aussi, vu la prop. I : $\sum_{j>i}(c_j - \alpha_{k,j})2^j > 2^i > \sum_{j<i}\alpha_{k,j}2^j$

Ce qui amènerait : $\sum_{j>i}c_j 2^j > \sum_{j>i}\alpha_{k,j}2^j + \sum_{j<i}\alpha_{k,j}2^j$

Et donc : $c = \sum_{j>i}c_j 2^j + 2^i > \sum_{j>i}\alpha_{k,j}2^j + 2^i + \sum_{j<i}\alpha_{k,j}2^j = \sum_{j=0}^{p}\alpha_{k,j}2^j = n_k$

Ce qui est contraire aux conditions de jeu : on ne peut effectivement prendre plus de frites qu'il n'y a sur la table.

Ensuite, **(2)** $\sum_{j>i}(\alpha_{k,j} - c_j)2^j$ est un multiple de 2^{i+1} (car j>i).

Nous pouvons alors décomposer cette somme en puissances de 2. En effet,

$\sum_{j>i}(\alpha_{k,j} - c_j)2^j$ peut s'écrire $\sum_{j>i}\beta_j 2^j$ avec $\forall j > i, \beta_j = 0$ ou 1

Par conséquent, dans la $i^{ème}$ colonne, on a remplacé un et un seul coefficient valant 1 par 0. La nim-somme ne peut plus être nulle car $s_i=1$ (nous avons à présent un nombre impair de 1 dans cette colonne). Nous avons donc modifié au moins un « 0 » de la nim-somme.

Deuxième cas : supposons que $\alpha_{k,i} = 0$

On a alors $n'_k = \sum_{j>i}(\alpha_{k,j} - c_j)2^j \mathbf{- 1x2^i} + \sum_{j<i}\alpha_{k,j}2^j$

Comme précédemment, **(1)** $\sum_{j>i}(\alpha_{k,j} - c_j)2^j - 2^i \geq 0$

En effet, dans le cas contraire,

Si nous avions $\sum_{j>i}\left(\alpha_{k,j}-c_j\right)2^j-2^i<0$ c'est-à-dire $\sum_{j>i}\left(c_j-\alpha_{k,j}\right)2^j+2^i>0$

Nous aurions aussi, vu la prop. I : $\sum_{j>i}\left(c_j-\alpha_{k,j}\right)2^j+2^i\geq 2^i>\sum_{j<i}\alpha_{k,j}2^j$

Ce qui amènerait : $c=\sum_{j>i}c_j2^j+2^i>\sum_{j>i}\alpha_{k,j}2^j+\sum_{j<i}\alpha_{k,j}2^j=\sum_{j=0}^{p}\alpha_{k,j}2^j=n_k$

Ce qui est contraire aux conditions de jeu.

Ensuite, **(2)** $\sum_{j>i}\left(\alpha_{k,j}-c_j\right)2^j$ est un multiple de 2^{i+1} (car j>i).

Donc, dans la division euclidienne de $\sum_{j>i}\left(\alpha_{k,j}-c_j\right)2^j-2^i$ par 2^{i+1} le reste est nécessairement 2^i.

Nous en déduisons que

$\sum_{j>i}\left(\alpha_{k,j}-c_j\right)2^j-2^i$ peut s'écrire $\sum_{j>i}\beta_j2^j+1x2^i$ avec $\forall j>i,\beta_j=0$ ou 1.

Par conséquent, dans la i[ème] colonne, on a remplacé un et un seul coefficient valant 0 par 1. . La nim-somme ne peut plus être nulle car s_i=1 (nous avons à présent un nombre impair de 1 dans cette colonne).

Conclusion : Dans les 2 cas ci-dessus, le terme s_i qui valait *0*, vaut donc après transformation *1*. Ce qui signifie que l'on est bien sorti du noyau. ❑

Pour illustrer cette première partie de démonstration, prenons un tas de 16 frites et calculons-en la nim-somme :

	=5 frites	= $\overline{101}^2$
	=4 frites	= $\overline{100}^2$
	=3 frites	= $\overline{011}^2$
	=1 frite	= $\overline{001}^2$
	=3 frites	= $\overline{011}^2$
	Nim-somme	= $\overline{000}^2$

Le joueur qui s'apprête à jouer se trouve donc à l'intérieur du noyau et quoi qu'il fasse, le coup suivant fournira une position hors du noyau. Ainsi, voyons ce que ça donne si, par exemple, il retire 2 frites du premier tas :

	=3 frites	$= \overline{011}^2$
	=4 frites	$= \overline{100}^2$
	=3 frites	$= \overline{011}^2$
	=1 frite	$= \overline{001}^2$
	=3 frites	$= \overline{011}^2$
	Nim-somme	$= \overline{110}^2$

La nouvelle position présente un nimber non nul : le joueur qui va jouer se trouve en dehors du noyau et il aura intérêt à faire un coup qui ramène son adversaire en position de nimber nul.

C'est l'objet de la deuxième partie de la démonstration que nous démontrons ci-après.

2. Soit à montrer : $\forall x \in X \setminus K, \Gamma(x) \cap K \neq \varnothing$

Hypothèse : Soit x ∈ X\K : x présente par définition une nim-somme non nulle.

$$\exists j \in \{0,...p\} : s_j = 1$$

Thèse : $\Gamma(x) \cap K \neq \varnothing$, c'est-à-dire $\exists y \in \Gamma(x), y \in K$

Il faut monter que l'on peut trouver un coup qui transforme toutes les sommes s_i valant 1 en 0 sans pour autant modifier celles qui égalent déjà 0.

Soit g le plus grand indice tel que $s_g = 1$, et soit k une rangée telle que $\alpha_{k,g} = 1$ (il en existe au moins une).

Posons $J_1 \subset \{0,...,p\}$ la famille des indices i tels que $s_i = 1$ (g est le plus grand de ces indices) et $J_0 \subset \{0,...,p\}$ la famille des indices i tels que $s_i = 0$ (il est possible que $J_0 = \varnothing$).

Posons $N_1 = \sum_{j \in J_1} \alpha_{k,j} 2^j$ et $N_0 = \sum_{j \in J_0} a_{k,j} 2^j$.

Il va de soi que le nombre de frites dans la rangée k vaut

$$n_k = N_0 + N_1 = \sum_{j \in J_0} a_{j,k} 2^j + \sum_{j \in J_1} \alpha_{k,j} 2^j$$

Posons par ailleurs, $N'_1 = \sum_{j \in J_1} (1 - \alpha_{k,j}) 2^j$

Pour mieux comprendre ce qui se passe, décomposons :

$$N_1 = \alpha_{k,g} 2^g + \sum_{\substack{j<g \\ j \in J_1}} \alpha_{k,j} 2^j \qquad (1)$$

$$et\ N'_1 = (1 - \alpha_{k,g}) 2^g + \sum_{\substack{j<g \\ j \in J_1}} (1 - \alpha_{k,j}) 2^j \qquad (2)$$

Puisque $\alpha_{k,g} = 1$ et à fortiori, $1-\alpha_{k,g} = 0$, (1) et (2) peuvent simplement s'écrire :

$$N_1 = 2^g + \sum_{\substack{j<g \\ j \in J_1}} \alpha_{k,j} 2^j \qquad \text{et} \qquad N'_1 = \sum_{\substack{j<g \\ j \in J_1}} (1 - \alpha_{k,j}) 2^j$$

Par conséquent, $N_1 \geq 2^g$ et $N'_1 < 2^g$. En effet, par la prop. I,

$$2^g > \sum_{j=0}^{j-1} 2^j \geq \sum_{\substack{j<g \\ j \in J_1}} (1 - \alpha_{k,j}) 2^j = N'_1$$

Donc $N'_1 < N_1$. Il existe N'_2 tel que $N_1 = N'_2 + N'_1$.

Enlevons N'_2 frites dans la rangée k.

Il en reste : $n_k - N'_2 = N_0 + N_1 - N'_2 = N_0 + N'_1 = \sum_{j \in J_0} a_{k,j} 2^j + \sum_{j \in J_1} (1 - \alpha_{k,j}) 2^j$

Pour chaque indice j de J_0, on n'a changé aucun coefficient de la colonne : la somme s_j reste donc nulle. En revanche, pour chaque indice j de J_1, il y a eu permutation entre 0 et 1, dans la rangée k et uniquement dans la rangée k (chaque valeur $\alpha_{k,j}$ a été remplacée par $1-\alpha_{k,j}$).

La valeur des sommes s_j a donc été modifiée et est passée ainsi de 1 à 0. Celles qui valaient déjà 0 n'ont pas changé. Nous obtenons une nim-somme nulle. Nous sommes à présent dans une position appartenant à l'ensemble K, c'est-à-dire une position $y \in \Gamma(x), y \in K$.

Nous avons donc montré que les positions à nim-somme nulle, c'est-à-dire celles où $s_j = 0$ $\forall j \in N; 0 \leq j \leq p$ (pour rappel, ce qui signifie que la somme des termes de la colonne d'indice j est paire) correspondent au noyau dans le jeu de Nim. ❑

En résumé :

Toute position paire (à nim-somme nulle, donc perdante) devient nécessairement une position impaire (à nim-somme non nulle, gagnante) après un coup joué.

De toute position impaire (à nim-somme non nulle, gagnante) peut découler une position paire (à nim-somme nulle, donc perdante) après un coup joué. Ce qui confirme bien « *Je puis perdre* » (si je joue mal et je ne trouve pas le coup ramenant au noyau) mais « *je gagne toujours* » (mon adversaire ne peut que perdre si je joue correctement).

Nous pouvons alors finir la partie commencée en début de paragraphe : il nous restait 14 frites disposées comme suit et c'était au stratège à jouer.

=3 frites	$= \overline{011}^2$
=4 frites	$= \overline{100}^2$
=3 frites	$= \overline{011}^2$
=1 frite	$= \overline{001}^2$
=3 frites	$= \overline{011}^2$
Nim-somme	$= \overline{110}^2$

Le stratège veut faire apparaître une nim-somme nulle : il retire 2 frites dans le deuxième tas :

=3 frites	$= \overline{011}^2$
=2 frites	$= \overline{010}^2$
=3 frites	$= \overline{011}^2$
=1 frite	$= \overline{001}^2$
=3 frites	$= \overline{011}^2$
Nim-somme	$= \overline{000}^2$

L'adversaire supprime par exemple un tas entier de 3 frites (le premier) :

=2 frites	$= \overline{010}^2$
=3 frites	$= \overline{011}^2$
=1 frite	$= \overline{001}^2$
=3 frites	$= \overline{011}^2$
Nim-somme	$= \overline{011}^2$

Le joueur suivant supprime un tas de 3 frites également :

=2 frites	$= \overline{010}^2$
=3 frites	$= \overline{011}^2$
=1 frite	$= \overline{001}^2$
Nim-somme	$= \overline{000}^2$

L'adversaire prend une frite dans le tas de 3 :

=2 frites	$= \overline{010}^2$
=2 frites	$= \overline{010}^2$
=1 frite	$= \overline{001}^2$
Nim-somme	$= \overline{001}^2$

Le stratège prend le tas avec l'unique frite :

=2 frites	$= \overline{010}^2$
=2 frites	$= \overline{010}^2$
Nim-somme	$= \overline{000}^2$

L'adversaire prend une frite dans un tas de 2 :

=2 frites	$= \overline{010}^2$
=1 frite	$= \overline{001}^2$
Nim-somme	$= \overline{011}^2$

Le stratège prend une frite dans le tas de 2 :

=1 frite	$= \overline{001}^2$
=1 frite	$= \overline{001}^2$
Nim-somme	$= \overline{000}^2$

L'adversaire prend une frite et laisse la dernière au gagnant annoncé.

V. 6. Nimbers dans Nim à tas

L'utilité d'attribuer des nimbers aux positions de jeu se révèle vraiment dans un jeu comme le jeu de Nim à tas.

Bouton donne seulement deux statuts aux positions : les P, perdantes et les N, gagnantes. Or, on sait maintenant :

- qu'à tout jeu combinatoire, on peut associer un graphe G=(X,U) où (x,y) ∈ U est un arc du jeu si et seulement si y ∈ Γ (x), c-à-d s'il existe un coup qui permette de passer de la position x à la position y ;
- que les positions P perdantes forment un ensemble stable et absorbant appelé le noyau ;
- que la fonction de Grundy attribue à chaque position x une valeur entière, nulle si la position est de type P.

On pourrait alors imaginer le jeu de Nim à plusieurs tas comme une superposition de jeux de Nim à un tas, à chacun desquels un graphe et une fonction de Grundy sont attribués. La question se pose alors : peut-on retrouver la fonction de Grundy de ce « multi-graphe » au départ de celles des graphes qui le composent ?

Nous avons besoin de définir de nouvelles notions :

La **somme cartésienne** de graphes se définit comme suit.

Soit 2 graphes quelconques orientés $G_1 = (X_1, U_1)$ et $G_2 = (X_2, U_2)$. Alors la somme cartésienne (notée par $\hat{+}$) de G_1 et de G_2 est un graphe noté de la façon suivante:

$$G = G_1 \hat{+} G_2 = (X,U) \text{ où } \quad X=\{(x_1, x_2) \mid x_1 \in X_1 \text{ et } x_2 \in X_2 \}.$$

$$U=\{((y_1, y_2), (w_1, w_2)) \mid y_1 \text{ et } w_1 \in X_1$$

$$y_2 \text{ et } w_2 \in X_2 \quad \text{et}$$

$$\text{soit } y_1 = w_1 \quad \text{et } (y_2, w_2) \in U_2$$

$$\text{soit } y_2 = w_2 \quad \text{et } (y_1, w_1) \in U_1 \}$$

Autrement dit, la somme cartésienne des jeux consiste à jouer soit sur G_1, soit sur G_2 jusqu'à ce qu'il ne soit plus possible de jouer du tout. Bien entendu, cette définition peut être étendue à la somme cartésienne de plus de deux jeux.

Sprague et Grundy ont alors découvert (séparément mais simultanément !) un théorème intéressant qui permet de calculer la fonction de Grundy du graphe somme cartésienne.

Théorème de Sprague-Grundy :

Si g_i est la fonction de Grundy associée au graphe $G_i=(X_i, U_i)$, alors $G=(X, U) = G_1 \hat{+} G_2 \hat{+} \ldots \hat{+} G_n$ a pour fonction de Grundy la fonction g, telle que

$$\forall\, x=(x_1, x_2, \ldots, x_n) \in X,\ g(x)= g(x_1, x_2, \ldots, x_n) = g_1(x_1) \oplus g_2(x_2) \oplus \ldots \oplus g_n(x_n)\ .$$

où $\oplus$ représente l'opérateur somme digitale.

V. 7. Jeux de Nim « VanLoGo »

Nous allons illustrer les notions de somme cartésienne de graphes avec de nouvelles règles de jeu. Comme dans le jeu de Marienbad, les frites sont disposées par tas, les joueurs effectuent des prélèvements dans un tas de leur choix à tour de rôle mais le nombre de frites prélevées ne peut dépasser un nombre maximum K (contrainte). Cherchons donc à établir la fonction de Grundy pour différentes positions de jeu et identifions le noyau.

Prenons par exemple une contrainte K=3 et la disposition de frites suivantes et jouons en version directe (celui qui prend la dernière frite a gagné) :

=10 frites

=8 frites

=6 frites

Chaque tas de frites correspond à une position particulière dans un graphe et possède un nimber ($=n_i \bmod (k+1)$) tel que cela a été calculé dans la section *IV. « Noyau et nimbers dans le jeu de Nim à un tas »*. Ces nimbers sont réécris en binaire et additionnés suivant la règle de la somme digitale.

	nimber	$\overline{nimber}^2$
=10 frites	*10mod4=*2	$\overline{10}^2$
=8 frites	*8mod4=*0	$\overline{00}^2$
=6 frites	*6mod4=*2	$\overline{10}^2$
		$\overline{00}^2$

La somme digitale étant nulle, pour gagner, il faut laisser la main. Admettons que l'adversaire prélève le nombre maximum autorisé dan le tas 1 : 3 frites.

	nimber	$\overline{nimber}^2$
=7 frites	*7mod4=*3	$\overline{11}^2$
=8 frites	*8mod4=*0	$\overline{00}^2$
=6 frites	*6mod4=*2	$\overline{10}^2$
		$\overline{01}^2$

Il faut faire disparaître le « 1 » apparaissant dans la fonction de Grundy. Nous avons plusieurs possibilités : faire disparaître le deuxième « 1 » dans le $\overline{nimber}^2$ du tas 1 en y ôtant une frite ou faire apparaître un « 1 » dans le deuxième ou troisième tas. Nous choisissons cette dernière possibilité : retirons 3 frites du tas 2.

		$nimber$	$\overline{nimber}^2$
	=7 frites	7mod4=3	$\overline{11}^2$
	=5 frites	5mod4=1	$\overline{01}^2$
	=6 frites	6mod4=2	$\overline{10}^2$
			$\overline{00}^2$

Le joueur adverse retire une frite du tas 3 :

		$nimber$	$\overline{nimber}^2$
	=7 frites	7mod4=3	$\overline{11}^2$
	=5 frites	5mod4=1	$\overline{01}^2$
	=5 frites	5mod4=1	$\overline{01}^2$
			$\overline{11}^2$

Nous ôtons 3 frites dans le tas 2 :

		$nimber$	$\overline{nimber}^2$
	=7 frites	7mod4=3	$\overline{11}^2$
	=2 frites	2mod4=2	$\overline{10}^2$
	=5 frites	5mod4=1	$\overline{01}^2$
			$\overline{00}^2$

Il en retire 2 dans le tas 3

		$nimber$	$\overline{nimber}^2$
	=7 frites	7mod4=3	$\overline{11}^2$
	=2 frites	2mod4=2	$\overline{10}^2$
	=3 frites	3mod4=3	$\overline{11}^2$
			$\overline{10}^2$

Nous ôtons 2 frites dans le tas 1:

		$nimber$	$\overline{nimber}^2$
	=5 frites	$5 mod 4 = 1$	$\overline{01}^2$
	=2 frites	$2 mod 4 = 2$	$\overline{10}^2$
	=3 frites	$3 mod 4 = 3$	$\overline{11}^2$
			$\overline{00}^2$

Il retire le tas 3 entier :

		$nimber$	$\overline{nimber}^2$
	=5 frites	$5 mod 4 = 1$	$\overline{01}^2$
	=2 frites	$2 mod 4 = 2$	$\overline{10}^2$
			$\overline{11}^2$

Nous ôtons 3 frites dans le tas 1:

		$nimber$	$\overline{nimber}^2$
	=2 frites	$2 mod 4 = 2$	$\overline{10}^2$
	=2 frites	$2 mod 4 = 2$	$\overline{10}^2$
			$\overline{00}^2$

Il retire le tas 1 :

		$nimber$	$\overline{nimber}^2$
	=2 frites	$2 mod 4 = 2$	$\overline{10}^2$
			$\overline{10}^2$

Il nous reste à prendre les deux dernières frites : comme prévu, nous avons gagné !

En conclusion, au jeu de Nim à plusieurs tas et contrainte K, il suffit d'écrire le nimber en base 2 et faire la nim-somme de chaque tas pour ramener le joueur adverse à chaque fois dans le noyau. Ainsi on est assuré de la victoire !

VI. D'autres Nims

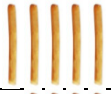
Il existe beaucoup d'autres versions de jeux qui s'apparentent au jeu de Nim dans lesquels les joueurs piochent tour à tour un certain nombre de pions et pour lesquels une stratégie gagnante existe, autrement dit, en jouant bien, un des joueurs a la possibilité de gagner, quoi que fasse son adversaire.

VI. 1. Le jeu de Moore

Le jeu de Moore, parfois aussi appelé Nim_k, a été développé et résolu par le mathématicien américain Eliakim H. Moore (1862-1932). Les règles du jeu de Moore sont identiques à celles du jeu de Marienbad classique à un détail près : on peut prendre autant de frites qu'on le souhaite dans maximum k tas (barquettes) ; k étant fixé par les joueurs. Ici, la barquette numéro i sera notée B_i.

Grâce à un exemple de partie, nous allons rechercher la fonction de Grundy associée au jeu de Moore. Nous avons choisi $X_0 = (3,5,4,2)$ et k=2, X_0 étant la position initiale du jeu.

Le tableau ci-dessous présente les 4 configurations et leurs fonctions de Grundy écrites comme dans le jeu de Marienbad en numération binaire.

	$f(B_1)=3$	$\overline{011}^2$
	$f(B_2)=5$	$\overline{101}^2$
	$f(B_3)=4$	$\overline{100}^2$
	$f(B_4)=2$	$\overline{010}^2$

Nous ne pouvons pas dans ce cas effectuer une Nim-somme classique car les joueurs peuvent modifier le nombre de frites dans plusieurs tas. Nous ne nous trouvons donc pas dans le cas d'une somme cartésienne de graphes et la fonction de Grundy associée au graphe ne peut être trouvée en effectuant une simple somme digitale.

Pour déterminer si une position X est perdante, on va pratiquer une somme sans retenue sur les colonnes, mais cette fois, écrite en base k + 1. Nous baptisons ce type de somme **Nim_k-somme**. Les positions du jeu amenant à une **Nim_k-somme nulle** seront les positions **P** perdantes du noyau.

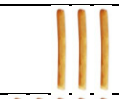
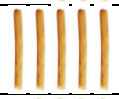
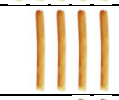
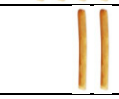
Ainsi, par exemple, dans un jeu de Nim_2, nous pouvons vérifier que la position (2, 3, 3, 1) est dans le noyau, donc perdante. En effet, en numération binaire, nous avons :

2	$\overline{10}^2$
3	$\overline{11}^2$
3	$\overline{11}^2$
1	$\overline{01}^2$

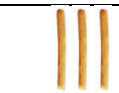
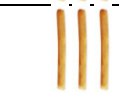
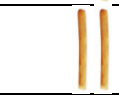
Nim_k-somme	$\overline{00}^3$

Cette position est donc bien une position perdante du noyau.

Revenons donc à notre exemple de départ où nous avons choisi comme configuration initiale X_0 = (3,5,4,2) dans un jeu de Nim_2 (k=2) et observons un exemple de partie où la stratégie gagnante est employée par un des joueurs :

	3	$\overline{011}^2$
	5	$\overline{101}^2$
	4	$\overline{100}^2$
	2	$\overline{010}^2$
	Nim_k-somme	$\overline{222}^3$

Le joueur connaissant la stratégie gagnante a donc intérêt à commencer la partie. Il va donc jouer retirant des frites de sorte à positionner son adversaire dans le noyau, c'est-à-dire dans un état de Nim_k-somme nulle. Pour cela, il doit faire apparaître dans chaque colonne un nombre de 1 multiple de 3. Comme il ne pourra pas faire apparaître de « 1 » dans la troisième colonne (cela reviendrait à ajouter des frites !), il est obligé de modifier les barquettes B_2 et B_3 : en ôtant au moins une frite dans chacune des barquettes, les « 1 » de la 3ème colonne disparaissent. Il faut alors s'arranger pour faire apparaître un seul 1 dans la colonne n°2 et un autre dans la colonne n°3. Cela peut se produire en retirant 2 frites dans B_2 (5 → 3) et 3 frites dans B_3 (4 → 1).

	3	$\overline{011}^2$
	3	$\overline{011}^2$
	1	$\overline{001}^2$
	2	$\overline{010}^2$
	Nim_k-somme	$\overline{000}^3$

Si le joueur suivant voulait maintenir dans le noyau (Nim_k-somme nulle) son adversaire, il devrait toucher à 3 barquettes et cela lui est interdit. Admettons qu'il décide de retirer 2 frites dans B_4.

	3	$\overline{011}^2$
	3	$\overline{011}^2$
	1	$\overline{001}^2$
	0	$\overline{000}^2$
	Nim_k-somme	$\overline{020}^3$

Le joueur stratège va revenir à une Nim_k-somme nulle en faisant disparaître les « 1 » de la 2ème colonne tout en maintenant la troisième colonne intacte : il retire 2 frites dans B_1 et B_2 ($3 \rightarrow 1$).

	1	$\overline{001}^2$
	1	$\overline{001}^2$
	1	$\overline{001}^2$
	0	$\overline{000}^2$
	Nim_k-somme	$\overline{000}^3$

La défaite du joueur suivant apparaît alors clairement : quoi qu'il fasse, qu'il prenne une frite dans une ou deux barquettes, au tour suivant, le gagnant aura la possibilité de prendre les frites restantes et il aura gagné ! Nous allons explorer les 2 cas de figure.

L'adversaire modifie une seule barquette

	0	$\overline{000}^2$
	1	$\overline{001}^2$
	1	$\overline{001}^2$
	0	$\overline{000}^2$
	Nim_k-somme	$\overline{002}^3$

Le stratège prend les deux dernières frites et gagne.

L'adversaire modifie deux barquettes

	0	$\overline{000}^2$
	0	$\overline{000}^2$
	1	$\overline{001}^2$
	0	$\overline{000}^2$
	Nim_k-somme	$\overline{001}^3$

Le stratège prend la dernière frite et gagne.

VI. 2. Le jeu de Fibonacci-Nim

Pour jouer à ce jeu, il est indispensable de connaître la célèbre suite de Fibonacci, définie par la formule de récurrence :

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_{n+2} = F_{n+1} + F_n \quad \forall n \in \mathbb{N} \end{cases}$$

La suite se développe comme suit : 0 – 1 – 1 – 2 – 3 – 5 – 8 – 13 – 21 – 34 – 55 - …

Fibonacci-Nim est donc une variante du jeu de Nim à un tas inventée par l'américain Robert E. Gaskell. Les joueurs fixent eux-mêmes le nombre de pions/frites sur le plateau et prélèvent tour à tour un certain nombre en respectant les règles suivantes :

1. Si un joueur prend k pions, le joueur suivant doit prendre entre 1 et 2k pions.
2. Le joueur qui commence doit obligatoirement laisser un pion au premier coup.
3. Le joueur qui prend le dernier pion gagne la partie.

Le nombre de pions prélevés est donc très variable et chaque position du jeu peut donc être définie par un couple ***(n, p)***, où **n** désigne le nombre de frites restantes à cet instant du jeu, et **p** le nombre maximal de jetons que le joueur peut prendre.

Tâchons de trouver les positions **P** et **N** dans un jeu. Bien entendu, belgitude oblige, nous ne parlerons plus de pions, mais de frites.

En analysant les règles, on s'aperçoit vite que

Obs 1. La position initiale n'est pas forcément gagnante ;

Obs 2. Toute position (n,p) avec $n \leq p$ est gagnante (il reste n frites et on est autorisé à les prendre toutes).

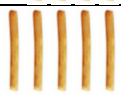
De cette dernière observation, on retire que :

Obs 3. le joueur n'a jamais intérêt à prendre un nombre de frite supérieur au tiers de frites restantes.

Effectivement, si le joueur 1 pioche $\frac{n}{3}$ frites (ou plus), il reste $\frac{2n}{3}$ (ou moins) avec une autorisation pour le joueur 2 de prélever les frites restantes (puisque $p \geq \frac{2n}{3}$).

En utilisant la « backward induction », nous allons tâcher d'identifier les positions gagnantes **N** et perdantes **P**.

Recherchons les **P** positions pour des tas de frites allant de n_0=2 à 21:

	n_0	**Positions P**
	2	2 est une P-position par définition, 1 est alors gagnant.
	3	3 est une P-position car je dois retirer au moins une frite, c'est-à-dire le tiers du tas (obs. 3), 1 et 2 sont gagnants.
	4	De 4, en retirant 1 seule frite, je place mon adversaire en P-position 3. 4 est donc gagnant.
	5	Pour placer l'adversaire en P-position 3, je dois retirer plus du tiers de frites. Je suis obligé d'envoyer mon adversaire en N-position 4. 5 est une P-position.
	6	Un seul retrait place l'adversaire en P-position 5 : 6 est une N-position.
	7	Un prélèvement de 2 frites restant inférieur au tiers du tas permet de placer l'adversaire en P-position 5 : 7 est une N-position.
	8	Pour placer l'adversaire en P-position 5, je dois retirer plus du tiers de frites (3>8/3). Je suis obligé d'envoyer mon adversaire en N-position 6 ou 7. 8 est une P-position.
	9	Un seul retrait place l'adversaire en P-position 8 : 9 est une N-position.
	10	Un retrait de 2 frites (<10/3) place l'adversaire en P-position 8: 10 est une N-position.
 ...	11	Pour arriver en P-position 8, il faut retirer 3 frites. Or, 3<11/3. Ce retrait apparait donc sensé pour autant qu'il soit autorisé, que p>2. n_0=11 est donc P-position.
 ...	12	Il ne serait pas judicieux d'envoyer l'adversaire en P-position 8, car, pour ce faire, il faudrait retirer 4 frites, soit le tiers du tas et donner l'occasion à l'adversaire de retirer la totalité du reste. Mais on peut envoyer l'adversaire en position intermédiaire 11 : il pourra au plus retirer 2 frites et sera donc incapable de nous envoyer en P-position 8.
	13	Par contre, il ne sert à rien d'envoyer l'adversaire de 13 vers 11 : car le coup suivant autoriserait 4 retraits et me placerait en P-position 8. Par ailleurs, l'envoi direct de 13 vers 8 n'est pas intelligent : (13-8)>13/3 ! De ce fait, 13 est une P-position.

	14	N-position par envoi sur 13.
	15	N-position par envoi sur 13.
	16	N-position par envoi sur 13, pour autant que la position précédente ne soit pas 17 (comme pour le cas 11).
	17	N-position par envoi sur 16.
	18	N-position par envoi sur 13, pour autant que la position précédente ne soit pas 19 ou 20.
	19	N-position par envoi sur 18.
	20	N-position par envoi sur 18.
	21	P-position : un envoi sur 13 impose un retrait égal au tiers des frites restantes : on perd !

On voit que si le retrait permettant de placer l'adversaire dans la position perdante la plus proche est supérieur ou égal au tiers du tas, la position est perdante. Par ailleurs, certaines positions ont un statut qui change selon le coup précédent (positions 11, 16 et 18).

Enfin, les P-positions absolues sont donc 2, 3, 5, 8, 13 , 21 … et bingo ! On retrouve la suite de Fibonacci.

En fait, le calcul de frites à retirer pour amener l'adversaire dans les positions P décrites précédemment utilise le fait que tout nombre s'écrit de façon unique comme somme de nombres de Fibonacci non consécutifs.

Théorème de Zeckendorf ou numérotation de Zeckendorf

Edouard Zeckendorf était un mathématicien belge qui a vécu de 1901 à 1983.

Voici son théorème :

« *Tout nombre N appartenant aux naturels strictement positifs peut s'écrire sous une et une seule forme d'une somme de nombres non consécutifs de la suite de Fibonacci d'indice plus grand que 1.* »

Ainsi, par exemple, grâce à ce théorème, il est possible d'écrire :

100 = 89+8+3

Ou encore 100 = 1.89+0.55+0.34+0.21+0.13+1.8+0.5+1.3+0.2+0.1

Ceci suggère ce qu'on a appelé la **numération de Zeckendorf**, une écriture positionnelle des naturels utilisant les nombre de Fibonacci comme base.

C'est-à-dire 100 = 1000010100**_Z**

Donc en généralisant, nous avons :

$$\forall n \in \mathbb{N}_0 : n = \sum_{i=1}^{k} \varepsilon_i(n).F_i$$

où $\varepsilon_i(n) \in \{0,1\}$ et $\varepsilon_k(n) = 1$,

avec $\varepsilon_i(n).\varepsilon_{i+1}(n) = 0$ (2 termes consécutifs ne sont pas permis).

Précision importante : Zeckendorf a utilisé la suite telle que définie ci-avant :

$$\begin{cases} F_1 = 1 \\ F_2 = 2 \\ F_{n+2} = F_{n+1} + F_n \quad \forall n \in \mathbb{N}_0 \end{cases}$$

Démonstration ([6])

- **Existence**

L'existence de la numération de Zeckendorf est évidente pour N = 1, 2 ou 3 : ce sont des nombres de Fibonacci !

Nous pouvons dès lors procéder par récurrence.

Supposons maintenant que tout naturel positif inférieur ou égal à N admette une numération de Zeckendorf. *(1)* Montrons que N+1 en admet une également.

Si N+1 appartient à la suite de Fibonacci, il existe d'office une numération pour N+1 !

Si N+1 n'appartient pas à la suite de Fibonacci, il existe p >0 tel que $F_p<N+1<F_{p+1}$.

Notons $N'=N+1- F_p$.

Comme $N' \leq N$, il existe une numération pour N' par *(1)*:

De plus, $N'=N+1- F_p<F_{p+1}-F_p = F_{p-1} + F_p-F_p= F_{p-1}$

Par conséquent, $N'<F_{p-1}$ de sorte que $N' = \sum_{i=1}^{k} \varepsilon_i(n).F_i$ avec k<p-1.

N+1 admet une décomposition en nombre de Fibonacci non consécutifs :

$$N+1= N'+F_p = \sum_{i=1}^{k} \varepsilon_i(n).F_i + F_p$$

- **Unicité**

Pour démontrer l'unicité, nous aurons besoin d'un lemme.

[6] Adaptation de [15]

Soit I une partie finie des naturels strictement positifs ne contenant pas 2 naturels consécutifs et soit n le plus grand nombre de cette partie I, alors

$$\sum_{i\in I} F_i < F_{n+1}$$

Nous allons procéder par récurrence :

Amorce : Cette relation est évidemment vraie pour n=1 : $F_1=1<F_2=2$

Pas récurrent : Supposons que le lemme soit vrai pour n.

Hérédité : Montrons qu'il est vrai pour n+1. On a $\sum_{i\in I} F_i = F_{n+1} + \sum_{i\in I\setminus\{n+1\}} F_i$

Or, en vertu du pas récurrent et comme $n \notin I$ (si $n+1 \in I, n \notin I$: I ne contient pas 2 termes consécutifs) :

$$\sum_{i\in I\setminus\{n+1\}} F_i < F_n$$

C'est-à-dire $\sum_{i\in I} F_i < F_{n+1} + F_n = F_{n+2}$ ❑

Nous utilisons ce dernier théorème pour démontrer l'unicité de la décomposition de Zeckendorf comme suit. Raisonnons par l'absurde et supposons que n possède deux décompositions de Zeckendorf différentes :

$$n = \sum_{i=1}^{k} \varepsilon_i(n).F_i \quad \text{(somme notée S)}$$

et $n = \sum_{i=1}^{k'} \varepsilon'_i(n).F_i$ (somme notée S') avec $\varepsilon_k(n) = \varepsilon'_{k'}(n) = 1$

Soustrayons de S et S' leurs termes communs, nous obtenons respectivement Σ et Σ', sommes égales mais formées de termes distincts. Appelons F_g le plus grand terme de Σ et $F_{g'}$ le plus grand terme de Σ'.

Par le lemme, nous avons :

(1) $\Sigma\ (=\Sigma') < F_{g+1}$

(2) $\Sigma'\ (=\Sigma) < F_{g'+1}$

Si $F_g \neq F_{g'}$, alors

- soit $F_g > F_{g'}$, et donc $F_g \geq F'_{g+1}$, ce qui amène $\Sigma(=\Sigma') \geq F_g \geq F_{g'+1}$, ce qui est contraire à (2).
- soit $F_{g'} > F_g$, et donc $F'_g \geq F_{g+1}$, ce qui amène $\Sigma' (=\Sigma) \geq F_g \geq F_{g'+1}$, ce qui est contraire à (1).

L'unicité de la décomposition_Z est ainsi démontrée par l'absurde. ❑

Stratégie gagnante

Nous avons déjà observé que les P-positions de ce jeu correspondent aux termes de la suite de Fibonacci. Les positions où il reste 1, 2, 3, 5, 8, 13, 21, … frites sur la table sont des situations perdantes.

La stratégie gagnante repose sur la décomposition de Zeckendorf (décomposition_Z).

$$\forall n \in \mathbb{N}_0 : n = \sum_{i=1}^{k} \varepsilon_i(n).F_i$$

où $\varepsilon_i(n) \in \{0,1\}$ et $\varepsilon_k(n) = 1$,

avec $\varepsilon_i(n).\varepsilon_{i+1}(n) = 0$ (2 termes consécutifs ne sont pas permis).

Soit n, le nombre de frites sur la table de jeu.

Nous écrivons sa décomposition_Z sous la forme : $n = \sum_{j=1}^{h} f_j$, où f_j est un terme de la décomposition_Z de n tel que

$$\forall j \in \{1,...,h\}, \exists i \in \{1,...,k\} : \varepsilon_i(n).F_i = f_j$$

et $f_j < f_{j+1} \; \forall j \in \{1,...,h-1\}$.

f_1 est donc le plus petit terme de cette décomposition_Z.

Pour illustrer ce propos, nous avons par exemple 30= 1 + 8 + 21= $f_1 + f_2 + f_3 = F_1 + F_6 + F_8$ avec $f_1 = F_1 = 1$; $f_2 = F_6 = 8$; $f_3 = F_8 = 21$.

On a les propriétés suivantes :

1. $f_1 \leq n$ (évident)
2. $f_1 = n \Leftrightarrow$ n est un terme de la suite de Fibonacci
3. si $f_1 < n$ alors , $f_1 < \frac{n}{3}$

En effet, puisque $f_1 < n$, il y a au moins deux termes dans la décomposition_Zeckendorf de n. Soit r le rang de f_1 dans la suite de Fibonacci. On a :

$$n \geq F_{r+2} + f_1 = F_{r+1} + F_r + f_1 > 3 f_1$$

4. Si le joueur (A) qui hérite d'un tas contenant n frites ($n \geq 1$) en retire f_1 frites (si les contraintes de jeu le permettent), alors
   - soit $f_1 = n$ et A gagne tout de suite ;
   - soit $f_1 < n$ (donc la décomposition_Z contient au moins deux termes et $n \geq 3+1=4$), si bien que le joueur suivant (B) ne peut retirer qu'un nombre K de frites tel que $1 \leq K < f_2$ (donc B ne peut tout retirer).

En effet, f_1 et f_2 n'étant pas consécutifs, $f_2 \geq F_{r+2} = F_{r+1} + f_1 > 2 f_1$: comme le joueur B peut retirer au plus 2 f_1, c'est que $K < f_2$; quant à $1 \leq K$, cela résulte des règles du jeu.

Si nous continuons la partie après que B ait retiré K frites et en notant n'=n- f_1-K le nombre de frites restantes, A recommence en retirant f'_1 frites (f'_1 étant le premier terme dans la décomposition_Z de n'), même si n'= f'_1.

Ceci est permis car $f'_1 \leq 2K$.

Ceci repose sur cette propriété :

Soit e un entier naturel non nul tel que le plus petit terme de sa décomposition_Z, noté f_e, soit ≥2 et K un entier tel que $1 \leq K < f_e$. Soit e'=e-K et $f_{e'}$ le plus petit terme de sa décomposition_Z, alors $f_{e'} \leq 2K$

Exemple : Prenons e=11=8+3 alors $f_e=3$

- soit K=1 et e'= 10 = 8+2 et $f_{e'} \leq 2K$
- soit K=2 et e'= 9 = 8+1 et $f_{e'} \leq 2K$
- mais si K= f_e =3, e'=8= $f_{e'} > 2K$ ou si K=0, e'=e, $f_e = f_{e'} > 2K$

En effet,
Posons $f_{e'}=F_i$ avec i≥2

- si i=2, $f_{e'}=1 \leq 2K$, puisque K≥1
- si i=3, $f_{e'}=2 \leq 2K$, puisque K≥1

On se place maintenant dans le cas où i≥4.
Raisonnons par l'absurde, c'est-à-dire supposons que $f_{e'}>2K$.

Alors, dans ce cas, la décomposition_Z de K ne peut contenir que des termes de la suite de Fibonnacci $\leq F_{i-2}$, car dans le cas contraire, la décomposition_Z de K contiendrait au moins un terme de cette suite $\geq F_{i-1}$, et donc on aurait $K \geq F_{i-1}$, ou encore $2K \geq 2\ F_{i-1}$ (1)

Or $2\ F_{i-1} > F_{i-1} + F_{i-2} = F_i = f_{e'}$. Par transitivité, l'inégalité (1) deviendrait $2K \geq f_{e'}$, ce qui est une contradiction. ❑

Donc, appliquant cette propriété en prenant e=n - f_1, donc $f_e = f_2$, et e'=n - f_1 – K, c'est-à-dire $f_{e'} = f'_1$, on a

$$f'_1 \leq 2K$$

Ainsi le joueur A peut retirer f'_1 frites du tas, cela, peu importe si ce f'_1 est égal ou non à n'. Et ainsi de suite …

Ainsi, le joueur (A) qui hérite d'un tas contenant n(≥1) frites en retire f_1 (c'est le plus petit terme de la décomposition_Z de n) frites (cela si les contraintes du jeu le permettent), alors ce joueur est certain de pouvoir gagner la partie (quoi que fasse l'autre joueur) : soit il gagne à ce coup, soit, lorsque vient son tour de jouer, il retire du tas un nombre de frites égal au plus petit terme de la décomposition_Z du nombre de frites constituant alors le tas (il pourra effectivement le faire).

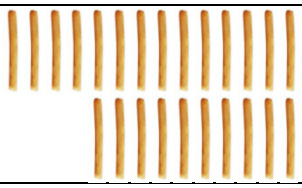
Comme le nombre de frites restant sur le tas diminue strictement à chaque retrait, le jeu ne peut "éternellement" continuer, donc au bout d'un nombre fini de retraits le nombre de frites restant sera un terme de la suite de Fibonacci et A pourra alors les retirer tous et donc gagnera.

Résumons-nous : Soit un tas de N frites au départ du jeu,

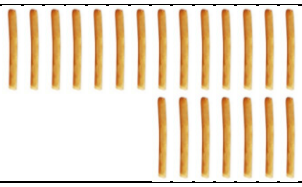
- Si N est un nombre de la suite de Fibonacci, je laisse la main (c'est une P-position),
- Si N n'est pas un nombre de la suite de Fibonacci, je prends la main et retire à chaque fois un nombre de frites égal au plus petit terme de la décomposition_Z de N et continue de la sorte en laissant mon adversaire jouer ce qu'il veut, puisqu'il sera alors incapable de gagner la partie, jusqu'à ce qu'il ne reste plus de frites dans le tas.

Illustrons cette stratégie avec un exemple.

Mettons en début de partie 24 frites sur la table et, 24 n'étant pas P-position, prenons la main en ramenant l'adversaire en P-position. Pour cela, nous devons écrire 24 en numération de Zeckendorf :

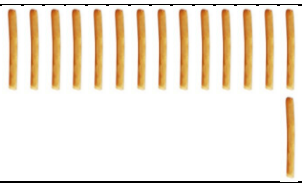
	24= 21 + 3	1 0 0 0 1 0 0_Z

Pour amener mon adversaire en position 21, je retire 3 frites : dans la décomposition_Z, on observe que le « 1 » de plus petit rang disparaît.

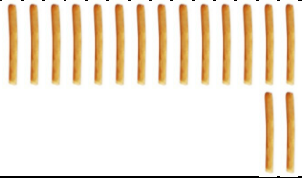
	21	1 0 0 0 0 0 0_Z

L'adversaire voudrait me ramener directement dans le noyau, il ne le pourrait pas. En effet, en vertu des règles du jeu, il ne peut retirer que 6 frites. Examinons tous les cas possibles pour illustrer le fait que le jeu de l'adversaire fera apparaître de nouveaux *1* dans la décomposition _Z.

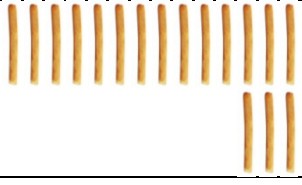
- L'adversaire retire 6 frites (c'est le maximum qui lui est autorisé par les règles du jeu) :

	15=13+2	1 0 0 0 1 0_Z

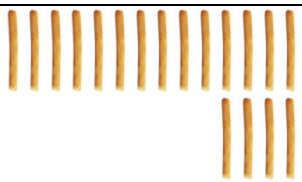
- L'adversaire retire 5 frites :

	16=13+3	1 0 0 1 0 0_Z

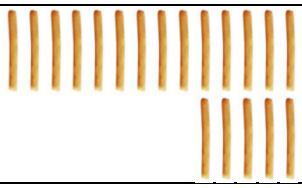
- L'adversaire retire 4 frites :

	17=13+3+1	1 0 0 1 0 1_Z

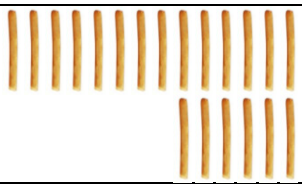
- L'adversaire retire 3 frites :

	18=13+5	1 0 1 0 0 0_Z

- L'adversaire retire 2 frites :

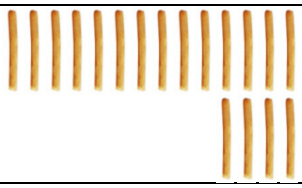
	19=13+5+1	1 0 1 0 0 1_Z

- Enfin, admettons qu'il retire 1 frite.

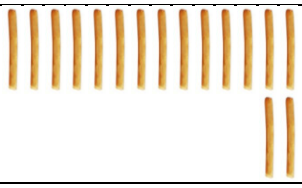
	20=13+5+2	1 0 1 0 1 0_Z

Dans chaque cas, de nouveaux « 1 » sont apparus dans la décomposition_Z!

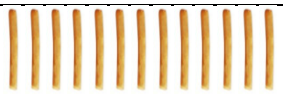
Comme il ne nous est pas possible de tester toutes les possibilités de jeu, nous continuerons la partie en supposant que l'adversaire a joué ce dernier coup. Je voudrais le ramener directement dans une P-position (sur un nombre de la suite de Fibonacci) mais il faudrait que je retire 7 frites. Or je ne peux en retirer que 2 maximum. Il est donc impossible de le ramener dans le noyau en 1 coup. Je vais alors retirer le « 1 » le plus à droite dans la décomposition_Z. Je retire donc 2 frites.

	18=13+5	1 0 1 0 0 0_Z

L'adversaire retire 2 frites.

	16=13+3	1 0 0 1 0 0_Z

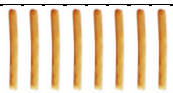
Je peux alors retirer 3 frites pour le placer en P-position 13.

	13	1 0 0 0 0 0_Z

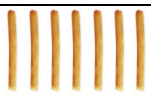
L'adversaire retire 4 frites.

	9=8+1	1 0 0 0 1_Z

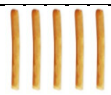
Je retire 1 frite.

	8	1 0 0 0 0_Z

L’adversaire retire 1 frite.

	7=5+2	1 0 1 0_Z

Je retire 2 frites.

	5	1 0 0 0_Z

L’adversaire est forcé de retirer 1 frite. S’il en retire plus, je prendrai toutes les frites restantes.

	4=3+1	1 0 1_Z

Je retire 1 frite

	3	1 0 0_Z

Si l’adversaire retire 1 frite, je prends les 2 restantes et inversement. Je gagne dans les 2 cas.

Sauf erreur de ma part, ma victoire était assurée dès que j’ai plongé mon adversaire dans le noyau. On voit ainsi que Fibonacci-Nim a une particularité : il n’est pas toujours possible d’entrer ou de sortir du noyau en un coup.

VI. 3. Encore bien d'autres Nims

Vous l'aurez compris, on peut imaginer à l'infini de nouvelles règles pour le jeu de Nim. Nous avons décidé de vous présenter les règles de jeux de Nim parmi les plus célèbres, sans toutefois en analyser la stratégie gagnante. Ce sera à vous de jouer, dans un prochain « Dédra-math-isons » ou « Math en Jeans » peut-être …

- **Jeu de Grundy** — Le jeu de Grundy se joue à une barquette (de frites !) unique et le seul coup autorisé est de séparer la barquette en deux parties de tailles distinctes. Le joueur ne pouvant plus la séparer a perdu.

- **Jeu de Nim Parité** — Le jeu de Nim Parité se joue à plusieurs barquettes. Les joueurs ne peuvent retirer qu'un nombre pair de frites dans une seule barquette. Le joueur prenant la dernière paire de frites a gagné. Les barquettes ne contenant plus qu'une frite sont écartées du jeu.

- **Jeu de Wythoff** — Le jeu de Wythoff se joue à deux barquettes. Les seuls coups autorisés sont : soit retirer un nombre quelconque de frites dans une des barquettes, soit retirer le même nombre de frites dans les deux barquettes. Le joueur prenant la dernière frite est le vainqueur.

- **Jeu de Nim "Pas tout »** — Le jeu de Nim « Pas tout » se joue à plusieurs barquettes. Les règles sont identiques à celles du jeu de Nim à plusieurs barquettes mais une contrainte est rajoutée : on ne peut pas prendre toutes les frites d'une barquette sauf s'il ne reste plus qu'une frite dans la barquette.

VII. Algorithmes de Nim

VII. 1. Introduction

Nous avons conçu un programme permettant de jouer au jeu de Nim et au jeu de Fibonacci contre une intelligence artificielle. Les deux jeux ont étés programmés en langage Python. Celui-ci offre l'avantage d'avoir une structure aisée. Nous tournerons notre focus vers la programmation de l'intelligence artificielle qui est toujours basée sur une stratégie gagnante. Ainsi, pour les deux algorithmes, on ne peut gagner sans appliquer cette même stratégie gagnante. D'autre part, et principalement dans le jeu de Fibonacci, les algorithmes ont été conçus de sorte à rendre le jeu le plus court possible pour autant que l'ordinateur ne se trouve pas dans une situation perdante.

VII. 2. Le jeu de Nim

Avant d'aller plus loin, nous aimerions noter une distinction importante entre l'informatique et les mathématiques. En mathématique, écrire **a=b** signifie que **a** et **b** ont une même valeur. En informatique, cette même écriture revient à dire que **a** *prend la valeur de* **b**. Par exemple, en mathématique, le système suivant serait impossible :

$$\begin{cases} a = 3 \\ a = 5 \end{cases}$$

Par contre, en informatique, ces deux lignes de code sont tout à fait valables :

```
a = 3
a = 5
```

La variable **a** prend d'abord la valeur 3 avant de prendre celle de 5. Cela veut donc également dire qu'en informatique, le signe « **=** » n'est pas commutatif. C'est important pour ce qui va suivre.

L'algorithme du jeu de Nim qui vous a été présenté initialement est de loin le plus simple. Pour rappel, la stratégie gagnante de ce jeu consiste, en version directe et pour une contrainte maximale égale à 3, à prendre si possible le reste de la division par 4 du nombre de frites restantes. Cela est visible dans l'algorithme [1]:

```
ai_removes = (remaining_matches)%4
 if  ai_removes == 0:
        ai_removes = random.randint(1,3)
    remaining_matches = remaining_matches - ai_removes
```

Comme en mathématique, on utilise des **variables**. Il aurait été possible de les nommer **a**, **b** ou même **x** mais il faut de nombreuses variables pour écrire un programme complet et donc il faut judicieusement les nommer. Dans cet extrait de code, il n'y en a que deux : **ai_removes** et **remaining_matches**, vous l'aurez compris, en français « l'ordinateur retire » et « allumettes restantes » (des frites dans note cas !).

Si vous avez saisi la stratégie gagnante du jeu de Nim, vous n'aurez aucun mal à déchiffrer la première ligne. On donne à la variable **ai_removes** la valeur calculée. Le **%** signifie modulo, en d'autres termes, le reste de la division. Donc, s'il reste 15 frites en jeu, **ai_removes** prendra la valeur 1. C'est le chemin le plus court vers la victoire. Si le reste de la division n'est pas zéro, l'ordinateur gagnera automatiquement. Donc, que se passe-t-il si celui-ci vaut zéro ? La ligne suivante l'indique :

```
    if  ai_removes == 0:
```

if introduit une condition au code, les deux points indiquent l'action à effectuer si cette condition s'avère valable. Le double « = » est en fait équivalent à notre « = » en mathématiques, c'est une simple question de notation.

```
        ai_removes = random.randint(1,3)
```

Vous l'aurez peut-être remarqué dans le bloc de code initial, cette ligne est décalée d'un espace de tabulation. Cela signifie qu'elle fait partie du bloc **if** mentionné avant. Toute ligne de code à sa suite fera partie de ce bloc si elle aussi commence par un espace de tabulation. Reprenons, si le reste de la division est bel et bien égal à zéro, l'ordinateur est dans une position perdante ! Quoi qu'il fasse, rien ne pourra le mener *directement* à une position de victoire. La meilleure solution est donc de jouer un coup au hasard, d'où la fonction **random**. Il ne s'agit plus d'une variable ici, c'est un appel de **fonction** qui selon ses **arguments** retournera différentes **données.** La fonction **random.randint()** - en Français, *nombre aléatoire entier* - avec les arguments **1** et **3** retourne un nombre aléatoire entier entre 1 et 3 inclus.

La dernière ligne est élémentaire. Elle montre simplement que l'ordinateur retire le nombre déterminé de frites.

VII. 3. Le jeu de Fibonacci-Nim

Cette première introduction au code était basique. L'algorithme de Fibonacci-Nim est plus complexe. Il faut savoir que la programmation d'une intelligence artificielle peut prendre des pages et des pages de texte et encore, dans le contexte présent, le niveau d'intelligence est très rudimentaire. Bref, voici encore une fois la partie intéressante du code du programme :

```
from math import*
import random

defget_highest_fibnumber():
    fibnumber_ID = 0
global fibnumber
    fibnumber,phi = 0,(sqrt(5)+1)/2
while fibnumber < remaining_matches:
        fibnumber_ID = fibnumber_ID + 1
       fibnumber = floor((1/sqrt(5))*(phi**(fibnumber_ID)+phi**(-fibnumber_ID)))
   fibnumber = floor((1/sqrt(5))*(phi**(fibnumber_ID-1)+phi**(-fibnumber_ID+1)))
```

```
def computer_turn(max1_matches):
global fibnumber,remaining_matches
    remove = 0
    max2_matches = floor((remaining_matches-1)/3)
    matches = remaining_matches
    max_matches = min(max1_matches,max2_matches)
while matches > 0:
get_highest_fibnumber(matches)
        matches = matches - fibnumber
if fibnumber <= max_matches:
            remove = fibnumber + remove
if remove > max_matches:
                remove = fibnumber
    save_fibnumber = fibnumber
    get_highest_fibnumber(remaining_matches)
    max3_matches = floor((remaining_matches - fibnumber - 1)/3)
if remove > max3_matches and (remaining_matches - remove) != fibnumber:
        remove = save_fibnumber
    if max1_matches >= remaining_matches:
        remove = remaining_matches
    if remove ==0:
        remove = random.randint(0,max_matches)
 if remove ==0:
            remove =1
remove_matches(remove)
```

Oui, le code est beaucoup plus long. Mais prenons la peine de l'analyser.

```
from mathimport*
```

Ici, on fait appel à un **module**supplémentaire car nous aurons besoin de quelques fonctions de mathématiques telles que **sqrt()** qui signifie *square root*, racine carrée. Pareil pour le module **random** qui nous permet de calculer un nombre aléatoire, ce module était déjà utilisé dans le premier algorithme.

```
def get_highest_fibnumber():
    fibnumber_ID = 0
    global fibnumber
    fibnumber,phi = 0,(sqrt(5)+1)/2
    while fibnumber < remaining_matches:
        fibnumber_ID = fibnumber_ID + 1
       fibnumber = floor((1/sqrt(5))*(phi**(fibnumber_ID)+phi**(-fibnumber_ID)))
   fibnumber = floor((1/sqrt(5))*(phi**(fibnumber_ID-1)+phi**(-fibnumber_ID+1)))
```

Ici est créée la fonction principale de l'AI. Elle fonctionne, comme une bonne partie de cet algorithme, par méthode de boucle. Nous utilisons la formule de Binet permettant de calculer n'importe quel nombre de la suite de Fibonacci en fonction de sa position dans celle-ci. Cette formule requiert le nombre d'or *phi* ($\varphi = \frac{\sqrt{5}+1}{2}$) ([7]):

$$F_n = \frac{1}{\sqrt{5}}(\varphi^{\,n} + \varphi^{\,-n})$$

Si vous avez saisi tout cela, comprendre cet extrait de code devient très facile. Tant que le terme de la suite de Fibonacci cherché est inférieur au nombre de frites restantes, le terme supérieur sera calculé. Vous noterez que la formule est écrite une seconde fois après la boucle, c'est normal car la boucle nous mène un terme trop loin, il faut donc revenir en arrière. La fonction **floor()** permet d'arrondir à l'unité. Indispensable car nous traitons de nombres irrationnels et il n'est pas rare d'avoir, par exemple, 14,000000000000000000001 comme résultat à la place de 14.

La fonction **get_highest_fibnumber()**est très utilisée puisque la stratégie gagnante de Fibonacci-Nim réside dans sa suite.

```
def computer_turn(max1_matches):
global fibnumber,remaining_matches
    remove = 0
    matches = remaining_matches
    max2_matches = floor((matches-1)/3)
    max_matches = min(max1_matches,max2_matches)
```

Afin de procéder au calcul de la valeur finale du nombre de frites retirées par l'ordinateur, beaucoup de *variables provisoires* seront utilisées. Ces variables ont un intérêt spécial mais peuvent alourdir le code ; à utiliser donc avec précaution. Ignorons la ligne commençant par **global** et voyons pourquoi on attribue à **remove** la valeur 0. **Remove** est la variable indiquant le nombre de frites définitivement retirées par l'ordinateur ou le joueur. Comme elle peut avoir pris une valeur différente auparavant, il est nécessaire de la remettre à zéro.

Maintenant, voyons le concept de *sauvegarde de variable*. Afin de procéder au calcul complet de l'algorithme, il faut utiliser le nombre de frites restantes en jeu (**remaining_matches**). Malheureusement, modifier cette variable – et c'est nécessaire dans les calculs – reviendrait à modifier le jeu en cours. C'est pour cela que l'on duplique la variable sous forme d'une *variable provisoire* -**matches** -dans le but de l'utiliser sans altérer son originale.

Dans la fonction complète, il y a 4 maxima utilisés. Ceux-ci permettent d'éviter que l'AI fasse un coup la mettant en désavantage ou tout simplement qu'elle triche en prenant plus de frites qu'autorisé. **Max1_matches** est la première d'entre elles. Cette variable indique le nombre maximum de frites à prendre que les règles autorisent. Si le joueur a retiré 17 frites au tour précédent, cette variable en vaudra le double, 34. **Max2_matches** indique à l'AI le nombre de frites à saisir à ne pas dépasser au risque de perdre au tour suivant. Dans le cas où il reste 90 frites en jeu, elle prend la valeur 29. En effet, si l'AI se saisit de 30 frites, rien ne m'empêche d'en saisir 60 au tour suivant et ainsi d'emporter la victoire.

[7] Voir Dédra-math-isons 2011 : « *Full metal numbers : à la suite des nombres métalliques* »

Comme dans aucun cas il ne faut dépasser l'un de ces deux maximums, nous n'en utiliserons qu'un seul, le plus petit des deux. D'où la nouvelle variable **max_matches** qui sera l'unique utilisée.

```
1.while matches > 0:
2.get_highest_fibnumber(matches)
3.        matches = matches - fibnumber
4.if fibnumber <= max_matches:
5.            remove = fibnumber + remove
6.if remove > max_matches:
7.                remove = fibnumber
```

Voici le noyau de l'algorithme, dans cette boucle qui peut paraître simple mais demande un peu de concentration pour en comprendre le fonctionnement. Remémorez-vous de la stratégie gagnante de Fibonacci-Nim avec un exemple concret.

Il y a74 frites en jeu, le joueur vient d'en retirer 7. **Matches** vaut 74 et **max_matches** vaut 14.

Idéalement, il faudrait placer l'adversaire en position 55 (nombre de la suite de Fibonacci) mais les règles du jeu ne nous le permettent pas. Or 74= 55 + 13 + 5 + 1 = 100101001_Z. Si nous ne pouvons pas placer le joueur directement en position « 55 », nous pouvons le placer dans une position intermédiaire en faisant « disparaître » un ou plusieurs « 1 » de l'écriture du nombre en base Fibonacci. C'est-à-dire, retirer un nombre de frites équivalent à un terme de la décomposition de Zeckendorf du nombre, qui respecterait les règles du jeu.

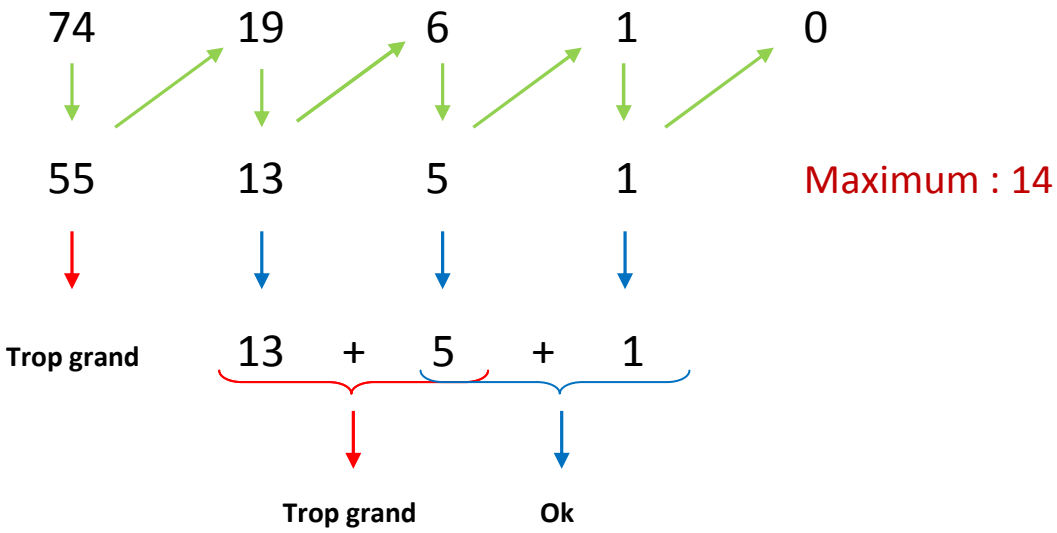
1. La condition est vérifiée
2. Le nombre retourné est 55, **fibnumber** prend cette valeur
3. **matches** devient 74 – 55 = 19
4. 55 est plus grand que 14 donc condition non respectée, retour au début de la boucle 

1. Ok
2. **fibnumber**devient 13
3. **matches** devient 19 – 13 = 6
4. 13 est bien plus petit que 14, condition valable
5. **remove** prend la valeur 13 + 0 = 13
6. 13 n'est pas plus grand que 14, retour au début 

1. Ok
2. **fibnumber** devient 5
3. **matches** devient 6 – 5 = 1
4. 5 est bien plus petit que 14 donc condition valable
5. **remove** prend la valeur 5 + 13 = 18
6. 18 est plus grand que 14, condition valable
7. **remove** retombe à 5, retour au début 

1. Ok
2. Le **fibnumber** de 1 est 1, il apparaît en effet deux fois dans la suite.
3. **matches** devient nul
4. Ok
5. **remove** prend la valeur 1 + 5 = 6
6. 6 est inférieur à 14, retour au début 

1. Condition non respectée : fin de la boucle. La valeur calculée est 6.



Bref, tout ça pour trouver un nombre : 6. Et c'est sans compter si il va être accepté ou non. Les lignes suivantes permettent de perfectionner l'algorithme pour le rendre infaillible.

```
  save_fibnumber = fibnumber
    get_highest_fibnumber(remaining_matches)
    max3_matches = floor((remaining_matches - fibnumber - 1)/3)
ifremove > max3_matches and (remaining_matches - remove) !=fibnumber:
        remove = save_fibnumber
```

Encore une fois nous utilisons une variable pour sauvegarder une valeur. **Save_fibnumber** enregistrera le dernier nombre de Fibonacci calculé, ici 1.

Nous recalculons ensuite le plus haut **fibnumber** du nombre initial de frites. Nous attribuons au dernier maximum utilisé, **max3_matches**, la valeur du tiers arrondi à l'inférieur de la différence entre le nombre initial de frites et le nouveau **fibnumber**. Dans notre exemple : $\frac{74-55-1}{3}$ qui vaut 6 (aucun rapport avec le 6 trouvé auparavant). Puisque l'AI a calculé plus tôt qu'elle devait retirer 6 frites, la condition qui suit n'est pas valable : 6 n'est pas plus grand que 6.

En fait, ces 5 lignes de codes alourdissent très fort l'algorithme et sont très peu utiles puisque elles n'interviennent en moyenne que dans un coup sur cent. A quoi servent-elles ? Hé bien si un joueur ne connaissant pas la stratégie gagnante joue de façon aléatoire, il peut arriver que l'AI joue par après un coup qui la mettrait en défaveur. Très peu de positions présentent ce cas mais en voici quand même une :

45 frites sont en jeu, le joueur en retire 12, il en reste donc 33.

L'ordinateur décompose 33 en nombres de Fibonacci : 33=21 + 8 + 3 + 1=1010101_Z. Il se doit de retirer 4 frites, pas 12 car le joueur pourrait alors gagner au tour suivant. Maintenant observez, si l'AI retire effectivement 4 frites, il va en rester 29 et le joueur suivant pourra en retirer 8, se plaçant ainsi en position gagnante ! Pour éviter cette situation, il vaut mieux jouer prudemment et ne retirer qu'une seule frite au lieu de 4.

```
    if max1_matches >= remaining_matches:
        remove = remaining_matches
    if remove ==0:
        remove = random.randint(0,max_matches)
if remove ==0:
            remove =1
remove_matches(remove)
```

Ces dernières lignes sont assez simples. D'abord, si l'ordinateur peut gagner en un coup - le joueur ayant retiré trop de frites avant - il prendra toutes les frites en jeu.

Ensuite, si l'AI est en position perdante, elle voudra ne retirer aucune frite. C'est pourquoi nous insérons une fonction **random.randint**. Vous remarquerez que le minimum est 0 et pas 1 quand le maximum est **max_matches**. La raison est que **max_matches** vaudra 0 quand il y aura moins de 3 frites en jeu.

Enfin si le nombre aléatoire calculé est 0, l'AI retirera 1 frite.

VII. 4. Les algorithmes complets

Si vous souhaitez observer les algorithmes, les voici dans leur intégralité :

Nim_Basique GUI

```
from tkinter import *
import random

root = Tk()

frite=PhotoImage(file="frite.gif")
canv=Canvas(root,width=1280,height=800)
root.title("Jeu de Nim Basique")
canv.pack()
root.resizable(False,False)
options_set=0
player_remove=StringVar()
player_turn=1
r1=Radiobutton(root,text="1",variable=player_remove,value="1")
r2=Radiobutton(root,text="2",variable=player_remove,value="2")
r3=Radiobutton(root,text="3",variable=player_remove,value="3")
r1.place(x=20,y=120)
r2.place(x=20,y=140)
r3.place(x=20,y=160)
r1.select()
r2.deselect()
r3.deselect()
label_computer_play= Label(root,text="",font=("Helvetica", 16))
label_computer_play.place(x=420,y=280)
label_computer_play2= Label(root,text="",font=("Helvetica", 16))
label_computer_play2.place(x=680,y=280)
label_computer_play3= Label(root,text="",font=("Helvetica", 16))
label_computer_play3.place(x=390,y=280)
label_victory= Label(root,font=("Helvetica", 16))
label_victory.place(x=350,y=340)

def prompt_options():
    global saisie,game_options,player_starts,pvp_mode,victory_mode

    label_computer_play.config(text="")
    label_victory.config(text="")
    button_player.config(state = DISABLED)
    button_start_game.config(state = DISABLED)

    game_options=Toplevel() # Fenêtre d'options
    game_options.title("Paramètres")
    game_options.focus_set()
    game_options.resizable(False,False)

    saisie=StringVar()  # Nombre d'allumettes initiales
    ent_matches_amount=Entry(game_options,textvariable=saisie)
    ent_matches_amount.grid(row =0,column=0)
    label_matches_amount= Label(game_options,text="Nombre de frites initiales")
    label_matches_amount.grid(row =0,column =1)

    close_options=Button(game_options,text="Valider",command=check_options)  #
Bouton Valider
    close_options.grid(row =5, column =1)

    player_starts=StringVar()
    chk1=Checkbutton(game_options,text="Tour du joueur 1 en
premier",variable=player_starts)
    chk1.select()
    chk1.grid(row =1,column =0)

    pvp_mode=StringVar()
    chk2=Checkbutton(game_options,text="Joueur contre joueur",variable=pvp_mode)
    chk2.deselect()
    chk2.grid(row =2,column =0)
```

```
    victory_mode=StringVar()
    chk3=Checkbutton(game_options,text="Mode misère",variable=victory_mode)
    chk3.grid(row =3,column =0)
    chk3.select()

def check_options():
    global error,remaining_matches,options_set
    error,remaining_matches="ok",0
    try:
        remaining_matches = int(saisie.get())
    except:
        error="pas un nombre"
    if remaining_matches<13 or remaining_matches>100:
        error="mauvais nombre"
    if error =="ok":    #Si aucune erreur n'est détectée, la fenêtre se ferme
        game_options.destroy()
        start_game()

def print_matches(print_range):
    if print_range< 0:
        print_range =0
    canv.delete(ALL)
    label_matches_amount= Label(root,text=print_range,font=("Helvetica",
16),width=30)
    label_matches_amount.place(x=20,y=450)
    if print_range > 30:
        spacing = 1170/print_range
    else:
        spacing = 39
    for i in range(print_range):
        x1 = 40 + i*spacing
        item = canv.create_image(x1, 560, image = frite)
    canv.create_line((15,480,45 + print_range*spacing,480),fill="black",width=1)
    canv.create_line((15,480,15,500),fill="black",width=1)
    canv.create_line((45 + print_range*spacing,480,45 +
print_range*spacing,500),fill="black",width=1)

def remove_player_matches():
    global remaining_matches,player_turn
    if remaining_matches>0 and int(pvp_mode.get()) == 0:
        remaining_matches = remaining_matches - int(player_remove.get())
        computer_turn()
    if int(pvp_mode.get()) == 1:
        if remaining_matches > 0:
            remaining_matches = remaining_matches - int(player_remove.get())
            if int(player_remove.get()) > 1:
                label_computer_play.config(text="frites retirées par Joueur")
            else:
                label_computer_play.config(text="frite retirée par Joueur")
            label_computer_play2.config(text=player_turn%2+1)
            label_computer_play3.config(text=int(player_remove.get()))
            player_turn = player_turn +1
        victorypresets()
    print_matches(remaining_matches)


def computer_turn():
    global remaining_matches
    victorypresets()
    if remaining_matches>1:
        if int(victory_mode.get()) == 1:
            computer_removes = (remaining_matches - 1)%4
        else:
            computer_removes = remaining_matches%4
        if computer_removes ==0:
            computer_removes =random.randint(1,3)
        remaining_matches = remaining_matches - computer_removes
        print_matches(remaining_matches)
        label_computer_play.config(text="L'ordinateur en a retiré")
        label_computer_play2.config(text=computer_removes)
```

```
def start_game():
    global player_turn
    button_player.config(state = NORMAL)
    print_matches(remaining_matches)
    if int(player_starts.get()) == 0 and int(pvp_mode.get()) == 0:
        computer_turn()
    if int(player_starts.get()) == 1 and int(pvp_mode.get()) == 1:
        player_turn = 0


def reset():
    button_start_game.config(state = NORMAL)
    label_computer_play.config(text="La partie est finie")
    label_computer_play2.config(text="")
    label_computer_play3.config(text="")

def victorypresets():
    global remaining_matches
    if int(victory_mode.get()) == 1 and int(pvp_mode.get()) == 0:
        if remaining_matches <1:
            button_player.config(state = DISABLED)
            label_victory.config(text="Perdu")
            reset()
        if remaining_matches ==1:
            remaining_matches =0
            label_victory.config(text="Gagné")
            reset()
            button_player.config(state = DISABLED)
    if int(victory_mode.get()) == 0 and int(pvp_mode.get()) == 0:
        if remaining_matches <1:
            button_player.config(state = DISABLED)
            label_victory.config(text="Gagné")
            reset()
        if remaining_matches >0 and remaining_matches <4:
            remaining_matches =0
            label_victory.config(text="Perdu")
            reset()
            button_player.config(state = DISABLED)
    if int(victory_mode.get()) == 1 and int(pvp_mode.get()) == 1:
        if remaining_matches < 1:
            button_player.config(state = DISABLED)
            if player_turn%2 == 0:
                label_victory.config(text="Victoire de joueur 1")
            else:
                label_victory.config(text="Victoire de joueur 2")
            reset()
    if int(victory_mode.get()) == 0 and int(pvp_mode.get()) == 1:
        if remaining_matches < 1:
            button_player.config(state = DISABLED)
            if player_turn%2 == 0:
                label_victory.config(text="Victoire de joueur 2")
            else:
                label_victory.config(text="Victoire de joueur 1")
            reset()

button_start_game= Button(root,text="Commencer une
partie",command=prompt_options)
button_start_game.place(x=15,y=20,height=40,width=150)

button_player= Button(root,text="Retirer",command=remove_player_matches)
button_player.place(x=60,y=140,height=25,width=80)



root.mainloop()
```

Fibonnaci_Nim CMD

```
from tkinter import *
from math import *
import random

root = Tk()

frite=PhotoImage(file="frite.gif")
canv=Canvas(root,width=1250,height=650)
root.title("Jeu de Fibonnaci-Nim")
canv.pack()
root.resizable(False,False)
options_set=0
player_remove=StringVar()
player_turn=1
player_remove=StringVar()
r1=Entry(textvariable=player_remove)
r1.place(x=20,y=120)
label_computer_play= Label(root,text="",font=("Helvetica", 16))
label_computer_play.place(x=420,y=280)
label_computer_play2= Label(root,text="",font=("Helvetica", 16))
label_computer_play2.place(x=680,y=280)
label_computer_play3= Label(root,text="",font=("Helvetica", 16))
label_computer_play3.place(x=390,y=280)
label_victory= Label(root,font=("Helvetica", 16))
label_victory.place(x=350,y=340)

def prompt_options():
    global saisie,game_options,player_starts,pvp_mode,victory_mode

    label_computer_play.config(text="")
    label_victory.config(text="")
    button_player.config(state = DISABLED)
    button_start_game.config(state = DISABLED)

    game_options=Toplevel() # Fenêtre d'options
    game_options.title("Paramètres")
    game_options.focus_set()
    game_options.resizable(False,False)

    saisie=StringVar()  # Nombre d'allumettes initiales
    ent_matches_amount=Entry(game_options,textvariable=saisie)
    ent_matches_amount.grid(row =0,column=0)
    label_matches_amount= Label(game_options,text="Nombre de frites initiales")
    label_matches_amount.grid(row =0,column =1)

    close_options=Button(game_options,text="Valider",command=check_options)  #
Bouton Valider
    close_options.grid(row =5, column =1)

    player_starts=StringVar()
    chk1=Checkbutton(game_options,text="Tour du joueur 1 en
premier",variable=player_starts)
    chk1.select()
    chk1.grid(row =1,column =0)

    pvp_mode=StringVar()
    chk2=Checkbutton(game_options,text="Joueur contre joueur",variable=pvp_mode)
    chk2.deselect()
    chk2.grid(row =2,column =0)

def check_options():
    global error,remaining_matches,options_set,max1_matches
    error,remaining_matches="ok",0
    try:
        remaining_matches = int(saisie.get())
    except:
        error="pas un nombre"
    if remaining_matches<8 or remaining_matches>1000:
        error="mauvais nombre"
```

```
    if error =="ok":    #Si aucune erreur n'est détectée, la fenêtre se ferme
        game_options.destroy()
        max1_matches = int(saisie.get()) - 1
        start_game()

def print_matches(print_range):
    if print_range< 0:
        print_range =0
    canv.delete(ALL)
    label_matches_amount= Label(root,text=print_range,font=("Helvetica",
16),width=30)
    label_matches_amount.place(x=20,y=450)
    if print_range > 30:
        spacing = 1170/print_range
    else:
        spacing = 39
    for i in range(print_range):
        x1 = 40 + i*spacing
        item = canv.create_image(x1, 560, image = frite)
    canv.create_line((15,480,45 + print_range*spacing,480),fill="black",width=1)
    canv.create_line((15,480,15,500),fill="black",width=1)
    canv.create_line((45 + print_range*spacing,480,45 +
print_range*spacing,500),fill="black",width=1)

def remove_player_matches():
    global remaining_matches,player_turn,max1_matches,error1
    error1,swap="ok",1
    try:
        swap = int(player_remove.get())
    except:
        error1="pas un nombre"
    if swap < 1:
        error1="mauvais nombre"
    if error1 =="ok":
        if swap <= max1_matches :
            if remaining_matches>0 and int(pvp_mode.get()) == 0:
                remaining_matches = remaining_matches - swap
                max1_matches = 2*swap
                computer_turn()
            if int(pvp_mode.get()) == 1:
                if remaining_matches > 0:
                    remaining_matches = remaining_matches -
int(player_remove.get())
                    max1_matches = 2*swap
                    if swap > 1:
                        label_computer_play.config(text="frites retirées par
Joueur")
                    else:
                        label_computer_play.config(text="frite retirée par
Joueur")
                    label_computer_play2.config(text=player_turn%2+1)
                    label_computer_play3.config(text=swap)
                    player_turn = player_turn +1
                victorypresets()
            print_matches(remaining_matches)

def get_highest_fibnumber(matches):
    global remaining_matches,fibnumber
    fibnumber,fibnumber_ID,phi = 0,0,(sqrt(5)+1)/2
    while fibnumber <= matches:
        fibnumber_ID = fibnumber_ID + 1
        fibnumber = floor((1/sqrt(5))*(phi**(fibnumber_ID)+phi**(-
fibnumber_ID)))
    fibnumber = floor((1/sqrt(5))*(phi**(fibnumber_ID-1)+phi**(-
fibnumber_ID+1)))

def computer_turn():
    global remaining_matches,max1_matches,fibnumber
    if remaining_matches <1:
        button_player.config(state = DISABLED)
        label_victory.config(text="Gagné")
```

```
        reset()
    else:
        computer_removes = 0
        max2_matches = floor((remaining_matches-1)/3)
        matches = remaining_matches
        max_matches = min(max1_matches,max2_matches)
        while matches > 0:
            get_highest_fibnumber(matches)
            matches = matches - fibnumber
            if fibnumber <= max_matches:
                computer_removes = fibnumber + computer_removes
                if computer_removes > max_matches:
                    computer_removes = fibnumber
        save_fibnumber = fibnumber
        get_highest_fibnumber(remaining_matches)
        max3_matches = floor((remaining_matches - fibnumber - 1)/3)
        if computer_removes > max3_matches and (remaining_matches -
computer_removes) != fibnumber:
            computer_removes = save_fibnumber
        if max1_matches >= remaining_matches:
            computer_removes = remaining_matches
        if computer_removes ==0:
            computer_removes = random.randint(0,max_matches)
            if computer_removes ==0:
                computer_removes =1
        remaining_matches = remaining_matches - computer_removes
        max1_matches = 2*computer_removes
        print_matches(remaining_matches)
        label_computer_play.config(text="L'ordinateur en a retiré")
        label_computer_play2.config(text=computer_removes)
        if remaining_matches <1:
            remaining_matches =0
            label_victory.config(text="Perdu")
            reset()
            button_player.config(state = DISABLED)

def start_game():
    global player_turn
    button_player.config(state = NORMAL)
    print_matches(remaining_matches)
    if int(player_starts.get()) == 0 and int(pvp_mode.get()) == 0:
        computer_turn()
    if int(player_starts.get()) == 1 and int(pvp_mode.get()) == 1:
        player_turn = 0


def reset():
    button_start_game.config(state = NORMAL)
    label_computer_play.config(text="La partie est finie")
    label_computer_play2.config(text="")
    label_computer_play3.config(text="")

def victorypresets():
    global remaining_matches
    if remaining_matches < 1:
        button_player.config(state = DISABLED)
        if player_turn%2 == 0:
            label_victory.config(text="Victoire de joueur 2")
        else:
            label_victory.config(text="Victoire de joueur 1")
        reset()

button_start_game= Button(root,text="Commencer une
partie",command=prompt_options)
button_start_game.place(x=15,y=20,height=40,width=150)

button_player= Button(root,text="Retirer",command=remove_player_matches)
button_player.place(x=60,y=140,height=25,width=80)
root.mainloop()
```

VIII. Conclusion

Dans ce document, nous avons voulu vous présenter des jeux dont l'issue était connue d'un des joueurs qui appliquerait judicieusement une stratégie gagnante.

Nous avons ainsi mis en évidence une série de positions dans lesquelles le gagnant devait enfermer son adversaire, et ce, en ayant recours à l'arithmétique modulaire ou à la théorie des graphes. Désormais, les jeux de Nim n'ont plus le moindre secret pour vous … A quoi bon alors encore vouloir jouer ensemble ? Connaissant par avance le gagnant de la partie, le jeu ne porte plus autant d'intérêt et s'amuser avec des jeux dont on connaît l'issue peut paraître une activité frivole.

Cependant, les jeux, à en lire les nombreux articles leur étant consacrés sur la toile, restent bien évidemment intéressants d'un point de vue mathématique. Ainsi, la recherche de stratégie peut parfois se révéler complexe et passionnante pour le mathématicien. Par exemple, le jeu d'échecs est un jeu combinatoire dont l'analyse des mouvements vers la victoire s'avère très riche. Par ailleurs, la théorie des jeux possède des connexions dans des domaines variés comme la logique, l'algorithmique, l'économie les sciences politiques ou encore la biologie. Cette théorie a pour objectif la recherche de la meilleure stratégie possible en analysant les situations actuelles et antérieures des « jeux » en utilisant des outils comme la théorie des graphes, les probabilités, la programmation linéaire.

Donc, ne gâchons pas notre double plaisir : nous avons partagé des idées mathématiques et vous pourrez toujours épater la galerie et défier vos amis qui se verront subjugués par votre talent, eux qui ignorent tout de la « formule magique ». Peu importe la situation, les règles utilisées, vous serez le grand vainqueur !

IX. Bibliographie

[1] BERGE Claude, ***Théorie générale des jeux à n personnes***, 1957, Gauthier-Villars, Paris

[2] BRIHAYE Thomas, ***Trains, sécurité, math et jeux***, 2008, Institut de Mathématique, Université de Mons-Hainaut

[3] CONWAY John H., ***On Numbers and Games***, 2de édition 2001, A K Peters, Ltd., Natick, Massachusetts

[4] DAVALAN Jean-Paul, ***Jeu de Fibonacci Nim***, version du 29/03/2005 (page consultée en octobre 2011), http://jeux-et-mathematiques.davalan.org/jeux/nim/fibonacci/index.html

[5] DELAHAYE Jean-Paul, ***Stratégies magiques au pays de Nim***, page créée le 21/06/2011, consultée en octobre 2011, http://interstices.info/jcms/i_61780/strategies-magiques-au-pays-de-nim

[6] FERGUSON Thomas S., ***Game theory***, 2005, Mathematics Department, UCLA

[7] FRAENKEL Aviezri S., ***Combinatorial Games: Selected Bibliographyn with a Succinct Gourmet Introduction***, 1994, Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot

[8] GORIN Baptiste, ***Jeux de Nim***, lycée Bellepierre de Saint Denis

[9] HARVIE Marie-Eve, ***Jeu de Fan Tan***, 2006, Université du Québec, Montréal

[10] INSTITUT SAINT-LAURENT, ***Modulus VS Cryptographix, L'arithmétique modulaire au service du chiffrement affine***, 2009, Congrès Dédramathisons UCL

[11] INSTITUT SAINT-LAURENT, ***Full metal numbers, à la suite du nombre d'or, les métalliques***, 2011, Congrès Dédramathisons UCL

[12] LABELLE Jacques, ***Théorie des graphes***, Modulo Editeur, 1981, p.46, 57-58, 105-111

[13] NOIRFALISE Robert, ***Le jeu de Marienbad***, IREM Clermont-Ferrand

[14] PANTALONI Vincent, ***Le jeu de Marienbad - Stratégie gagnante,*** 2009, Lycée Jean Zay d'Orléans

[15] PICHEREAU Alain, ***Suites de Fibonacci***, page consultée en octobre 2011, http://alain.pichereau.pagesperso-orange.fr/index.html

[16] TAVENEAUX Antoine, ***Etude de quelques jeux de Nim***, 2005, Université Paris Diderot, LIAFA

[17] WIKIPEDIA, ***Jeu de Nim***, mise à jour le 13 janvier 2012 (page consultée en octobre 2011), http://fr.wikipedia.org/wiki/Jeux_De_Nim

[18] WIKIPEDIA, ***Théorie des jeux combinatoires***, mise à jour le 6 septembre 2011 (page consultée en octobre 2011), http://fr.wikipedia.org/wiki/Th%C3%A9orie_des_jeux_combinatoires

[19] WIKIPEDIA, ***Jeu de Grundy***, mise à jour le 26/12/2011 (page consultée en janvier 2012), http://fr.wikipedia.org/wiki/Jeu_de_Grundy

[20] WIKIPEDIA, ***Jeu de Wythoff***, mise à jour le 12/12/2011 (page consultée en janvier 2012), http://fr.wikipedia.org/wiki/Jeu_de_Wythoff

[21] WIKIPEDIA, ***Zeckendorf's theorem***, mise à jour le 01/11/2011 (page consultée en novembre 2011), http://fr.wikipedia.org/wiki/Zeckendorf's_theorem