

# Baseball des couleurs - Une petite étude théorique

Christophe BAL : projetmbc@gmail.com

Version du 2017-01-21

## Mentions « légales »

Ce document est mis à disposition selon les termes de la licence Creative Commons « Attribution - Pas d'utilisation commerciale - Partage dans les mêmes conditions 4.0 International ».



---

## Table des matières

Baseball des couleurs - Une petite étude théorique	1
1 Origine du jeu	2
2 Les règles	2
3 Premier contact : la méthode « <i>une base à la fois</i> »	3
4 Soyons opportuniste : la méthode « <i>on avance au mieux</i> »	6
5 Annexe : nombre de configurations possibles	10

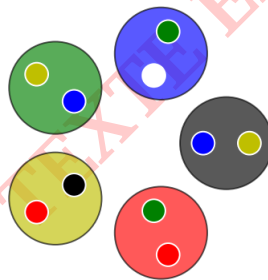
---

# 1 Origine du jeu

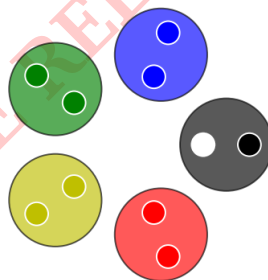
L'auteur de ces modestes lignes a pris connaissance du jeu « *le baseball des couleurs* » le jeudi 5 janvier 2017 lors d'une formation d'« *Informatique Débranchée* » proposée par l'Académie de Grenoble. Ce jeu est inspiré du « *jeu de l'orange* » proposé dans le livre « *L'informatique sans ordinateur - Programme d'activités d'éveil pour les élèves à partir de l'école primaire* » de Tim Bell, Ian H. Witten et Mike Fellows qui a été adapté à l'utilisation en classe par Robyn Adams et Jane McKenzie (Septembre 2009 - 2e éd. juin 2014) <sup>1</sup>.

## 2 Les règles

On considère cinq îlots colorés différemment que nous nommerons « *bases* ». Chacune de ces bases contient deux trous pour y accueillir deux jetons. A chaque base sauf une on associe deux jetons de même couleur, et pour une seule base on associe juste un jeton de même couleur. La partie commence en mettant au hasard deux jetons sur chaque base sauf une qui ne contiendra qu'un seul jeton. Voici un exemple où c'est la base noire qui n'est associée qu'à un seul jeton noir.



Le seul mouvement autorisé est le déplacement d'un jeton vers le trou libre à condition que le dit jeton et le trou soient dans des bases voisines. Dans l'exemple ci-dessus, on ne peut donc bouger que l'un des jetons des bases bleue et rouge. Le but du jeu est d'obtenir la configuration suivante où chaque jeton a retrouvé sa base (la position du trou dans la base noire est sans importance).



Si vous ne connaissez pas ce jeu nous vous conseillons, avant de lire la suite de ce document, d'essayer de trouver une méthode (il en existe plusieurs possibles et peut-être que vous trouverez un autre point de vue que celui que nous proposerons plus bas). Pour cela, il suffit d'expérimenter en vous fabriquant des bases et des jetons carrés, ce qui est très vite fait.

**Remarque :** dans ce jeu nous avons à tout moment accès à toutes les informations mais on pourrait très bien imaginer n'avoir accès qu'aux informations présentes sur trois bases voisines (concrètement, on mettrait les bases sur un plateau tournant avec au-dessus un cache ne laissant apparaître que trois bases). Il faut savoir qu'en informatique l'on peut avoir ce type de contrainte : par exemple, les routeurs qui dirigent les informations sur le réseau physique de l'internet n'ont pas une connaissance globale du réseau et pourtant ils arrivent à trouver d'assez bonnes solutions pour acheminer des données.

1. Voir <http://csunplugged.org/books/>.

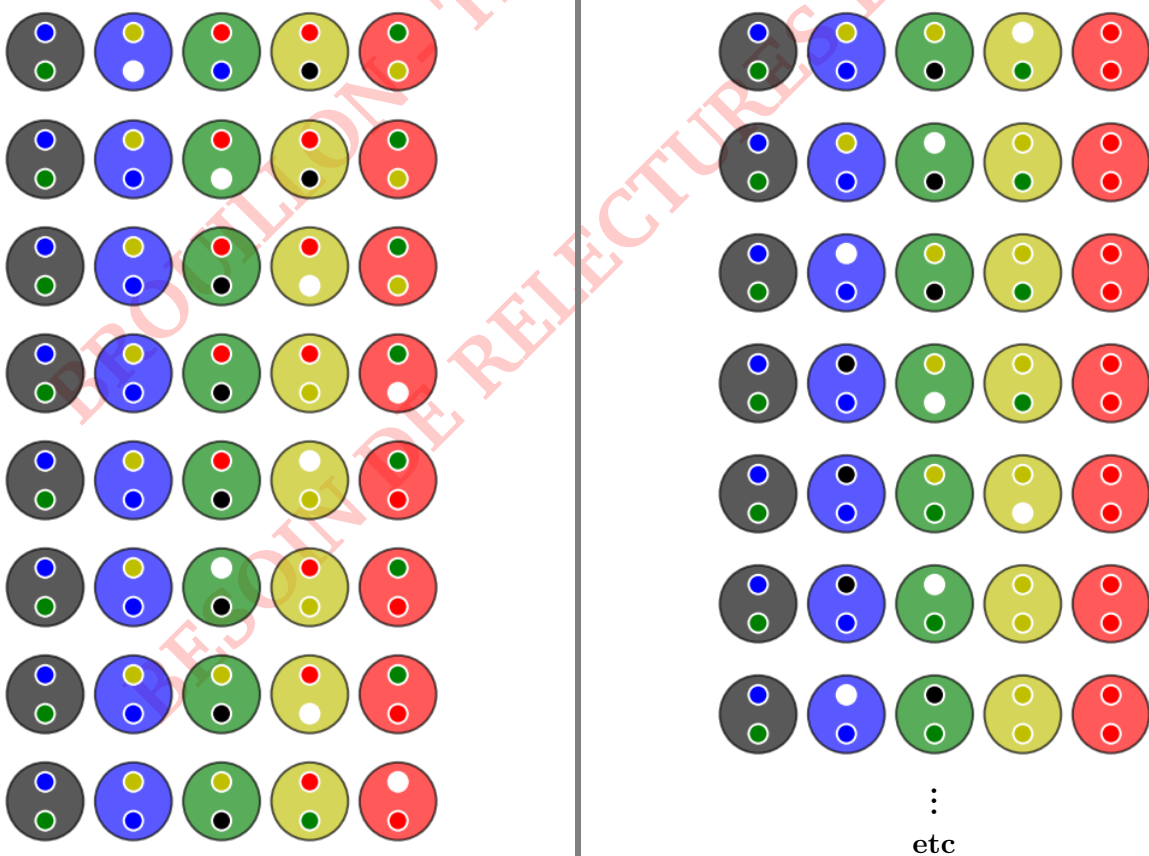
### 3 Premier contact : la méthode « une base à la fois »

Avant de chercher une solution optimale, commençons déjà par en trouver au moins une qui fonctionne<sup>2</sup>. Laissant de côté le problème de la solution la plus efficace, nous avons assez vite les deux idées simples suivantes.

1. Casser le cercle pour représenter les bases en ligne comme ci-dessous où il est important de bien mettre à gauche la base noire, celle qu'avec un seul jeton de même couleur une fois le problème résolu.
2. S'interdire tout mouvement de la base rouge, la plus à droite, à la base noire, la plus à gauche.



L'ajout de la contrainte va nous permettre de donner un algorithme simple à décrire mais aussi facile à valider. Par contre, nous devinons bien que nous nous interdisons de résoudre rapidement le problème. La méthode employée consiste à remplir la cinquième base, puis la quatrième... en cherchant juste à rapatrier les jetons manquants. Voici des premiers mouvements possibles où vous noterez au passage qu'une fois une base remplie à droite, celle-ci n'est plus jamais utilisée (le jeu évolue dans la colonne de gauche puis dans celle de droite).



Pour définir précisément comment fonctionne notre méthode, nous le faisons via l'algorithme suivant (l'écriture formelle employée est simple à comprendre).

<sup>2</sup> Les problèmes d'optimisation d'algorithme se traitent toujours dans un second temps. Cette règle s'applique aussi constamment lorsque l'on programme.

---

**Donnée :** une configuration en ligne quelconque de début de jeu

**Résultat :** une configuration en ligne où tous les jetons sont rentrés dans leur base

**Début**

Aucune base n'est isolée pour le moment (nous allons vite voir ce que cela signifie).

**Tant Que** la configuration contient un jeton qui n'est pas dans sa base :

$\mathcal{D}$  : la base non remplie la plus à droite.

$Coul_{\mathcal{D}}$  : la couleur de la base  $\mathcal{D}$ .

// Les deux jetons de couleur  $Coul_{\mathcal{D}}$  peuvent être dans la même base.

$j$  : un jeton de couleur  $Coul_{\mathcal{D}}$  le plus à droite possible en dehors de la base  $\mathcal{D}$ .

$\mathcal{J}$  : la base du jeton  $j$ .

// Deux contraintes.

$Ctr_1$  : ne pas passer par d'éventuelles bases isolées.

$Ctr_2$  : ne pas bouger l'autre jeton de couleur  $Coul_{\mathcal{D}}$  excepté si les deux jetons sont dans la même base.

En respectant les deux contraintes  $Ctr_1$  et  $Ctr_2$ ,

- si besoin, amener le trou dans la base à droite de la base  $\mathcal{J}$ ,
- puis utiliser le trou pour amener  $j$  dans la base  $\mathcal{D}$ .

**Si** la base  $\mathcal{D}$  est complète :

- └ Isoler la base  $\mathcal{D}$  (en la décalant un peu plus à droite par exemple).

---

Nous devons vérifier la validité de cet algorithme c'est à dire vérifier trois choses.

1. **NON AMBIGUÏTÉ** : les actions proposées doivent être sans ambiguïté.
2. **FINITUDE** : les actions à faire seront toujours en nombre fini.
3. **RÉSOLUTION** : une fois toutes les actions effectuées, nous devons obtenir une configuration où tous les jetons sont rentrés dans leur base.

Le contrat de « *non ambiguïté* » est rempli même si une certaine liberté est laissée pour déplacer le trou ou le jeton<sup>3</sup> à condition de ne pas visiter d'éventuelles bases isolées et sans bouger un éventuel jeton déjà bien placé dans la base la plus à droite non encore complète. En pratique, la « *non ambiguïté* » n'est jamais justifiée, par contre les deux derniers points doivent toujours faire l'objet d'une démonstration comme nous allons le faire tout de suite.

*Démonstration.* La preuve va s'appuyer sur deux résultats très simples dont on va donner des énoncés un peu formels mais avec des preuves visuelles très simples.

*Fait n°1 : on peut toujours déplacer le trou de la base où il est vers une base voisine  $\mathcal{V}$  en laissant fixe un jeton de son choix dans la base  $\mathcal{V}$ .*

En effet, considérons le cas suivant où l'on veut déplacer le trou vers la troisième base en ne touchant pas au jeton rouge (les choix faits ne nuisent pas à la généralité du raisonnement pour un déplacement vers la droite).



Il suffit de procéder comme suit (c'est évident).



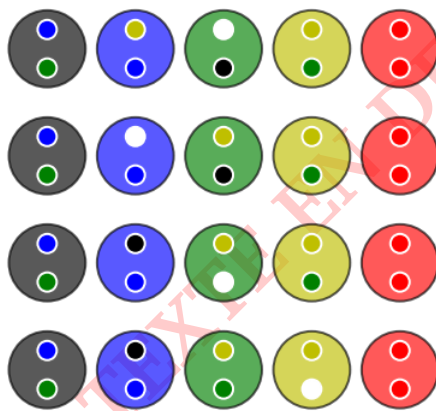
---

3. Nous avons ici un bel exemple d'algorithme « *non déterministe* » en ce sens qu'en lançant l'algorithme plusieurs fois sur la même configuration initiale, on ne passera pas forcément par les mêmes étapes intermédiaires pour résoudre le jeu.

Le cas où la base voisine est à gauche se traite de façon analogue (on peut aussi utiliser un argument de type « symétrie »).

*Fait n°2 : avec les notations de l'algorithme, on suppose le trou être dans la base  $\mathcal{J}_d$  à droite de la base  $\mathcal{J}$  du jeton  $j$  à déplacer, et que cette base  $\mathcal{J}_d$  n'est pas la base  $\mathcal{D}$  à remplir. Dans ce cas, on peut placer le jeton  $j$  dans la base  $\mathcal{J}_d$  et le trou dans la base à droite de  $\mathcal{J}_d$  en choisissant la place du trou.*

La preuve est bien plus simple à comprendre que l'affreux énoncé ci-dessus (on se demande bien qui a pu rédiger un truc pareil). Ci-dessous, le jeton à déplacer est le jaune dans la deuxième base bleue, et nous choisissons de ne pas toucher au jeton jaune de la quatrième base jaune (le lecteur notera la généralité de la méthode proposée).



*Finitude et résolution* : commençons pas démontrer que l'algorithme commence par remplir la base rouge la plus à droite.

Si la base rouge est déjà remplie, aucune action n'est requise et le résultat est vrai. Supposons donc que nous ayons au moins un jeton rouge dans l'une des quatre premières bases. Nous pouvons alors suivre les instructions de l'algorithme comme suit.

- Grâce au fait n°1, il est effectivement possible de placer le trou dans la base à droite de celle du jeton rouge à déplacer, et ceci sans faire bouger l'autre jeton rouge déjà dans la base rouge.
- Ensuite, le fait n°2 nous permet d'amener le jeton rouge à déplacer dans l'avant-dernière base jaune et le trou dans la base rouge, de nouveau sans faire bouger un éventuel jeton rouge déjà dans la base rouge.
- Il ne reste plus qu'à déplacer notre jeton rouge à la place du trou.

Si avant de faire ces opérations la base rouge contenait déjà un jeton rouge, nous avons rempli cette base, sinon l'algorithme nous fera remplir cette base dans un second temps (lors de la prochaine boucle **Tant Que**).

Une fois la base rouge remplie, celle-ci est isolée. Ceci implique que l'algorithme va travailler sur une version à quatre bases du jeu. Pour conclure, il suffit de reprendre le raisonnement ci-dessus non plus avec la base rouge mais avec la base jaune. Puis ensuite l'algorithme travaillera avec trois puis enfin deux bases et l'on raisonnera à chaque fois de la même façon. Ceci achève de montrer les propriétés de finitude et de résolution.  $\square$

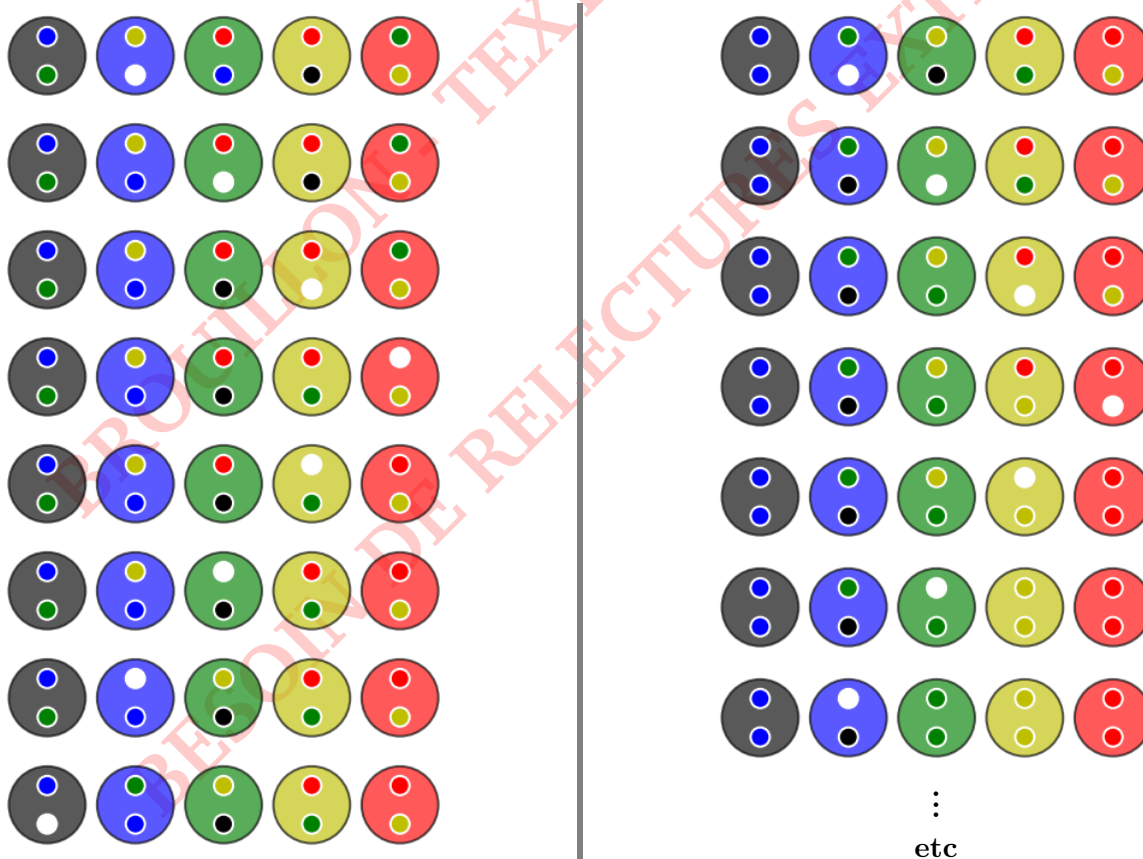
**Remarque :** vous noterez que la démonstration précédente est valable pour un nombre quelconque  $n \geq 2$  de bases (les plus tatillons pourront faire un raisonnement par récurrence).

## 4 Soyons opportuniste : la méthode « on avance au mieux »

La très grosse maladresse de la méthode « *une base à la fois* » est de ne pas chercher à profiter de chaque déplacement du trou pour arranger la situation. Nous allons juste considérer cette idée pour proposer une nouvelle méthode de résolution<sup>4</sup> qui elle aussi commence par mettre les bases en ligne de la base noire vers la rouge, et interdit tout mouvement de la base rouge à la base noire.

1. On ordonne les couleurs, pour cela nous dirons que lorsque l'on va de la gauche vers la droite, on passe de couleurs froides à des couleurs chaudes.
2. On commence par déplacer le trou vers la base rouge la plus chaude. Lors de ce déplacement, il faut déplacer à chaque fois le jeton le plus froid vers la gauche. Si les deux jetons de la base où va aller le trou sont de même chaleur, on choisit au hasard l'un des deux jetons.
3. Une fois arrivé à la base rouge, on bouge le trou vers la base noire la plus froide. Lors de ce déplacement, ce sont les jetons les plus chauds qui migrent vers la droite. Si l'on tombe sur deux jetons de même chaleur, on applique la même tactique que ci-dessus.
4. On continue les deux opérations 2 et 3 tant que toutes les bases ne sont pas complètes.

Voici le début de l'application de cette méthode.



Intuitivement, on sent bien que l'on ne va pas entrer dans une boucle infinie mais encore faut-il passer de l'intuition à une preuve irréfutable. Pour cela donnons d'abord une version plus formelle de notre nouvel algorithme de résolution (pour ordonner les couleurs nous allons les numéroter afin de simplifier les explications tout en raisonnant de façon très générale).

4. Cette méthode a été proposée par les personnes en charge de la formation d'« *Informatique Débranchée* » proposée par l'Académie de Grenoble le jeudi 5 janvier 2017. Voir la section « *Origine du jeu* ».

---

**Donnée :** une configuration en ligne quelconque de début de jeu

**Résultat :** une configuration en ligne où tous les jetons sont rentrés dans leur base

**Début**

On numérote de 0 à 4 les bases de la gauche vers la droite.

On associe chaque couleur au numéro de la base de la dite couleur.

//  $d$  donne la direction que doit suivre le trou.

// •  $d = 1$  pour un déplacement vers la droite.

// •  $d = -1$  pour un déplacement vers la gauche.

$d \leftarrow 1$

**Tant Que** la configuration contient un jeton qui n'est pas dans sa base :

$t$  : numéro de la base où est le trou.

**Si**  $(d, t) = (1, 4)$  :

$d \leftarrow (-1)$

**Sinon Si**  $(d, t) = (-1, 0)$  :

$d \leftarrow 1$

**Sinon:**

        //  $v$  est le numéro de la base voisine de celle du trou où l'on doit mettre ce dernier.

$v \leftarrow t + d$

$max$  : maximum des numéros des couleurs des jetons dans la base  $n^\circ v$ .

$min$  : minimum des numéros des couleurs des jetons dans la base  $n^\circ v$ .

**Si**  $d = 1$  :

            Déplacer un jeton de couleur  $min$  de la base  $n^\circ v$  vers la base  $n^\circ t$ .

**Sinon:**

            Déplacer un jeton de couleur  $max$  de la base  $n^\circ v$  vers la base  $n^\circ t$ .

---

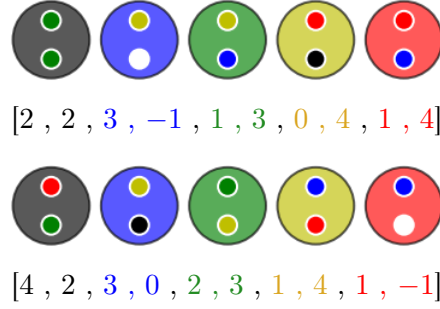
Les instructions sont clairement non ambiguës. Prouvons les propriétés de « *finitude* » et de « *résolution* ».

*Démonstration.* À chaque configuration  $\mathcal{C}$ , on associe une liste  $L(\mathcal{C})$  de nombres comme suit (des exemples visuels sont donnés un peu plus bas).

1. À chaque jeton on associe le numéro de sa couleur, et l'on considère le trou comme un jeton associé à l'entier  $(-1)$ .
2. Commençons par considérer les jetons de la base  $N^\circ 0$ .
  - (a) Si les deux jetons sont de la même valeur  $x$ , on définit la liste  $L(\mathcal{C}) = [x, x]$  notée en utilisant des crochets. Dans une liste, l'ordre d'écriture est important et il peut y avoir des répétitions.
  - (b) Si les deux jetons sont de valeurs différentes  $x$  et  $y$ , on définit la liste  $L(\mathcal{C}) = [x, y]$  si  $x < y$ , et  $L(\mathcal{C}) = [y, x]$  sinon.
3. Ensuite on passe aux jetons de la base  $n^\circ 1$ .
  - (a) Si les deux jetons sont de la même valeur  $r$ , on « *augmente* » la liste  $L(\mathcal{C})$  en lui adjoignant à droite la liste  $[r, r]$ . Ainsi si  $L(\mathcal{C}) = [x, y]$  à la fin de l'étape  $n^\circ 2$ , on obtient ici  $L(\mathcal{C}) = [x, y, r, r]$ .
  - (b) Si les deux jetons sont de valeurs différentes  $r$  et  $s$ , à la liste  $L(\mathcal{C})$  on adjoint à droite soit la liste  $[r, s]$  si  $r < s$ , soit  $[s, r]$  sinon. Ainsi si  $L(\mathcal{C}) = [x, y]$  à la fin de l'étape  $n^\circ 2$ , on obtient ici  $L(\mathcal{C}) = [x, y, r, s]$  ou  $L(\mathcal{C}) = [x, y, s, r]$  suivant les cas.
4. On fait de même avec la base  $n^\circ 2$ , puis la  $n^\circ 3$ , ... *etc.*

En termes plus courants, dans chaque base on ordonne les jetons de la valeur la plus petite à la valeur la plus grande, puis on « *accole dans une liste* » les couples « *ordonnés* » ainsi formés. Voici des exemples.





Les listes de nombres de même taille sont comparables comme le sont les mots d'un dictionnaire, on parle d'ordre lexicographique. Par exemple, nous avons :

- $[2, 2, 3, 1, 3, 0, 4, 1, 4] > [2, 2, 1, 3, 0, 4, 1, 3, 4]$  en comparant le 3 et le 1 aux troisièmes positions qui sont les premières valeurs à être différentes (par contre peu importe ce qu'il y a aux positions suivantes).
- $[1, 2, 2, 3, 3, 0, 4, 1, 4] < [2, 2, 3, 1, 3, 0, 4, 1, 4]$  en comparant le 1 et le 2 aux premières positions (sans se soucier des nombres qui suivent).

Nous n'allons pas comparer les deux listes  $L(\mathcal{C}_1)$  et  $L(\mathcal{C}_2)$  associées à deux configurations  $\mathcal{C}_1$  et  $\mathcal{C}_2$ . Nous allons considérer à la place les sous-listes  $SL(\mathcal{C}_1)$  et  $SL(\mathcal{C}_2)$  obtenues en retirant le seul  $(-1)$  présent dans les listes  $L(\mathcal{C}_1)$  et  $L(\mathcal{C}_2)$  respectivement (bien noter que les listes comparées via l'ordre lexicographique sont celles sans  $(-1)$  la valeur du trou).

Nous voilà armés pour faire notre démonstration sans encombre (mais pas sans réflexion).

*Fait n°1 : si le trou n'est pas dans la base n°0 alors les mouvements indiqués par l'algorithme amène le trou dans la base n°0.*

C'est évident à condition d'avoir bien noté que pour la configuration gagnante, le trou est dans la base n°0.

*Fait n°2 : la configuration gagnante est la seule telle que le trou soit dans la base n°0 et telle que sa sous-liste soit minimale (c'est à dire qu'elle inférieure ou égale à toutes les listes  $SL(\mathcal{C})$  possibles).*

Soit  $\mathcal{G}$  la configuration gagnante. Il est clair que  $SL(\mathcal{G}) = [0, 1, 1, 2, 2, \dots]$  et que le trou est dans la base n°0.

Soit  $\mathcal{C}$  une configuration non gagnante. Si  $SL(\mathcal{C}) \neq SL(\mathcal{G})$ , il est évident que  $SL(\mathcal{C}) > SL(\mathcal{G})$ . Sinon si  $SL(\mathcal{C}) = SL(\mathcal{G})$  c'est que le trou n'est pas dans la base n°0.

*Fait n°3 : lorsque l'un des mouvements demandés par l'algorithme est effectué, on passe d'une configuration  $\mathcal{C}_1$  à une configuration  $\mathcal{C}_2$ . On a alors la relation :  $SL(\mathcal{C}_2) \leq SL(\mathcal{C}_1)$ .*

Considérons le cas où le trou se déplace vers la droite. Notons  $L(\mathcal{C}_1) = [\dots, -1, x, y, z, \dots]$  où les points de suspension indiquent des éventuelles paires de valeurs. En particulier,  $(-1)$  et  $x$  sont les valeurs de deux jetons partageant la même base, tandis que  $y$  et  $z$  sont celles de deux autres jetons dans la base voisine à droite (notons que forcément  $y \leq z$ ). Nous avons deux situations.

1. **Cas 1 :**  $x \leq y$ .

Dans ce cas,  $L(\mathcal{C}_2) = [\dots, x, y, -1, z, \dots]$  d'où  $SL(\mathcal{C}_2) = SL(\mathcal{C}_1)$ .

2. **Cas 2 :**  $y < x$ .

Dans ce cas,  $L(\mathcal{C}_2) = [\dots, y, x, -1, z, \dots]$  d'où  $SL(\mathcal{C}_2) < SL(\mathcal{C}_1)$ .

Le cas d'un déplacement vers la gauche se traite de façon analogue.

*Fait n°4 : si  $\mathcal{C}$  est une configuration telle que  $SL(\mathcal{C})$  ne soit pas minimale, l'algorithme fera apparaître à un moment ou à un autre une configuration  $\mathcal{C}'$  telle que  $SL(\mathcal{C}') < SL(\mathcal{C})$  (en fait, le dit moment arrivera avant un éventuel aller-retour « complet » du trou depuis sa base dans la configuration  $\mathcal{C}$ ).*



Comme  $SL(\mathcal{C})$  n'est pas minimale, il existe au moins deux valeurs  $x$  et  $y$  telles que  $x > y$  avec  $y$  situé après  $x$  dans  $SL(\mathcal{C})$  lorsqu'on lit cette liste de gauche à droite. La condition  $x > y$  implique que  $x$  et  $y$  ne sont pas dans la même base. Nous avons :  $L(\mathcal{C}) = [\dots, x, \dots, y, \dots]$  où les points de suspension indiquent d'éventuelles valeurs.

Parmi tous les  $y$  possibles, on choisit celui qui est le plus à gauche possible dans  $SL(\mathcal{C})$ , c'est à dire celui qui est le plus prêt de  $x$  (dans ce cas,  $y$  est la plus petite valeur de sa base). Avec ce choix, les valeurs éventuelles  $w$  entre  $x$  et  $y$  dans  $SL(\mathcal{C})$  vérifient toutes  $y < x \leq w$ . Ceci permet donc de choisir  $x$  et  $y$  voisins dans  $SL(\mathcal{C})$  (avec des bases associés voisines différentes). Avec ce nouveau choix,  $L(\mathcal{C}) = [\dots, g, x, y, d, \dots]$  où les points de suspension indiquent des éventuelles paires de valeurs. Notons que seul  $g$  peut être égal à  $(-1)$ . Désignons par  $\mathcal{B}_x$  et  $\mathcal{B}_y$  les bases associées aux valeurs  $x$  et  $y$ .

Rappelons que tant que  $SL(\mathcal{C})$  n'est pas minimale, la configuration  $\mathcal{C}$  n'est pas une configuration gagnante et donc le trou se balade. Nous avons alors les situations suivantes.

1. **Cas 1 :**  $g = -1$

Si l'algorithme est dans une phase de déplacement vers la droite, alors comme dans la preuve du fait n°3, voir son cas 2, nous savons que la configuration suivante  $\mathcal{C}'$  vérifie  $SL(\mathcal{C}') < SL(\mathcal{C})$  ce qui prouve ici le fait n°4.

Sinon le trou va aller vers la base n°0. Si lors de ces déplacements, l'une des configurations  $\mathcal{C}'$  vérifie  $SL(\mathcal{C}') < SL(\mathcal{C})$  alors le fait n°4 sera validé. Sinon le trou reviendra en direction de la base  $\mathcal{B}_x$ . Si lors de ces déplacements, l'une des configurations  $\mathcal{C}'$  vérifie  $SL(\mathcal{C}') < SL(\mathcal{C})$  alors le fait n°4 sera validé. Sinon nous nous retrouvons dans le sous-cas traité ci-dessus où l'on avait un déplacement du trou vers la droite avec  $g = -1$ , et de nouveau nous obtenons le résultat souhaité.

2. **Cas 2 :**  $g \neq -1$  et le trou est à gauche de la base  $\mathcal{B}_x$ .

Si l'algorithme est dans une phase de déplacement vers la droite, alors le trou va prendre la place de  $g$  et l'on retombe dans le cas 1. Si le déplacement se fait vers la gauche, on peut appliquer exactement le même raisonnement que dans le deuxième sous-cas du cas 1 traité ci-dessus.

3. **Cas 3 :**  $g \neq -1$  et le trou est à droite de la base  $\mathcal{B}_y$ .

L'algorithme va amener le trou dans la base voisine de  $\mathcal{B}_y$  de sorte que l'on ait une configuration  $\mathcal{C}'$  telle que  $L(\mathcal{C}') = [\dots, g, x, y, d, -1, \alpha, \dots]$  avec  $SL(\mathcal{C}')$  non minimal.

Si lors de ces déplacements une configuration  $\mathcal{C}''$  vérifie  $SL(\mathcal{C}'') < SL(\mathcal{C})$ , le fait n°4 est bien entendu vérifié. Sinon  $L(\mathcal{C}')$  devient  $[\dots, g, x, -1, y, ?, ?, \dots]$ , puis  $[\dots, -1, g, y, x, ?, ?, \dots]$  où l'on utilise le fait que  $y < x$ . Pour cette dernière liste, la configuration  $\mathcal{C}'''$  vérifie  $SL(\mathcal{C}''') < SL(\mathcal{C})$  et c'est gagné!

*Finitude et résolution :* le fait n°1 nous permet de considérer le 1er moment où le trou est dans la base n°0. Si nous avons une configuration  $\mathcal{C}$  gagnante, nous nous arrêtons comme demandé et il n'y a rien à prouver.

Sinon la configuration  $\mathcal{C}$  est telle que  $SL(\mathcal{C})$  ne soit pas minimale d'après le fait n°2. Grâce au fait n°4, nous savons que nous arriverons ensuite à une configuration  $\mathcal{C}'$  telle que  $SL(\mathcal{C}') < SL(\mathcal{C})$ . Dès lors les faits n°3 et n°1 nous permettent d'affirmer que le trou va se retrouver dans base n°0 pour une configuration  $\mathcal{C}''$  telle que  $SL(\mathcal{C}'') \leq SL(\mathcal{C}') < SL(\mathcal{C})$ .

En résumé, si le trou se retrouve dans la base n°0 pour une configuration  $\mathcal{C}$  non gagnante, alors l'algorithme nous fera déplacer le trou jusqu'à le faire arriver de nouveau dans la base n°0 pour une nouvelle configuration  $\mathcal{C}''$  telle que  $SL(\mathcal{C}'') < SL(\mathcal{C})$ .

Pour conclure, il suffit de noter que le nombre de listes  $SL(\mathcal{C})$  est fini. Dès lors il est impossible d'avoir une suite strictement décroissante de listes du type  $SL(\mathcal{C})$ . Ceci signifie que l'algorithme va en un nombre fini d'étapes amener le trou dans la base n°0 pour une configuration gagnante.  $\square$

**Remarque :** la démonstration s'adapte sans problème à un nombre quelconque  $n \geq 2$  de bases.

## 5 Annexe : nombre de configurations possibles

On peut se demander combien de configurations de jeux sont possibles. Nous allons calculer ce nombre en cherchant directement une formule générale pour  $n \geq 2$  bases. Dans ce cas nous avons au total  $(2n - 1)$  jetons et  $n$  « couleurs » différentes.

1. Tout d'abord, nous avons  $n$  choix de bases où placer le trou.
2. Une fois le trou placé dans une base  $\mathcal{B}_0$ , nous avons  $n$  configurations possibles pour cette base  $\mathcal{B}_0$  car seule la « couleur » du jeton ajouté importe.
3. Maintenant que la base  $\mathcal{B}_0$  contient le trou et un jeton, que se passe-t-il pour les  $(2n - 2)$  jetons restants et les  $(n - 1)$  bases restantes  $\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_{n-1}$  ?

- (a) Pour la base  $\mathcal{B}_1$ , nous devons choisir deux jetons parmi  $(2n - 2)$  sans tenir compte de l'ordre du tirage. Ce nombre est noté  $\binom{2n-2}{2}$  ce qui se lit « 2 parmi  $(2n - 2)$  ». Nous pouvons ici le calculer directement.

En effet, nous avons  $(2n - 2)$  choix pour le premier jeton  $p$ , et ensuite  $(2n - 3)$  pour le deuxième jeton  $d$ . Si l'on tient compte de l'ordre cela fait  $(2n - 2)(2n - 3)$  choix possibles. Or tirer  $p$  puis  $d$ , ou  $d$  puis  $p$  ne change rien pour la configuration, donc le nombre de choix possibles est  $\frac{(2n - 2)(2n - 3)}{2}$ .

- (b) De façon analogue, nous avons ensuite  $\frac{(2n - 4)(2n - 5)}{2}$  pour la base  $\mathcal{B}_2$ .

- (c) ... etc.

Le nombre total  $C(n)$  de configurations est donc<sup>5</sup> :

$$C(n) = n \times n \times \frac{(2n - 2)(2n - 3)}{2} \times \frac{(2n - 4)(2n - 5)}{2} \times \dots \times \frac{4 \times 3}{2} \times \frac{2 \times 1}{2}$$

$$C(n) = \frac{n^2 \times (2n - 2)!}{2^{n-1}}$$

Dans la seconde formule, le  $2^{n-1}$  vient de ce que l'on divise par 2 uniquement pour les bases  $\mathcal{B}_1, \mathcal{B}_2, \dots$ , et  $\mathcal{B}_{n-1}$ . De plus, on utilise la factorielle d'un naturel non nul  $k$  définie par  $k! = 1 \times 2 \times 3 \times \dots \times k$ .

Ceci nous donne par exemple les valeurs suivantes (voir la remarque n°1 juste après).

- $C(2) = 4$  ce qui est calculable directement en imaginant les configurations possibles avec deux bases.
- $C(3) = 54$ ,  $C(4) = 1440$  et  $C(5) = 63\,000$ .
- $C(33) \approx 3,2 \times 10^{82}$  que l'on comparera à  $10^{80}$  qui est une estimation du nombre d'atomes dans l'univers<sup>6</sup>.

**Remarque n°1 :** les calculs ont été faits via le logiciel SageMath utilisable directement en ligne à l'adresse suivante <https://sagecell.sagemath.org>. Le code utilisé est le suivant.

---

```
def C(n):
    return n**2 * factorial(2*n - 2) / 2**(n - 1)

for k in range(2, 6):
    print C(k)

print C(33).n(10)
```

---

5. L'utilisation de points de suspension est un acte très critiquable mais il va nous permettre de faciliter la compréhension des formules manipulées.

6. Notons que si l'estimation du nombre d'atomes de l'univers est juste, alors il est tout simplement impossible « d'écrire » tous les naturels de 1 à  $C(33)$  en associant chaque naturel à un seul atome de l'univers. Vertigineux !

**Remarque n°2 :** un argument de symétrie montre facilement que  $C(n)$  est pair. Ceci se vérifie aussi à l'aide de la formule  $C(n) = \frac{n^2 \times (2n-2)!}{2^{n-1}}$ .

En effet, nous avons :

$$\begin{aligned} C(n) &= \frac{n^2 \times (2n-2)!}{2^{n-1}} \\ C(n) &= n^2 \times \frac{(2n-2) \times (2n-3) \times (2n-4) \times (2n-5) \times \cdots \times 4 \times 3 \times 2 \times 1}{2 \times 2 \times \cdots \times 2 \times 2} \\ C(n) &= n^2 \times (n-1) \times (2n-3) \times (n-2) \times (2n-5) \times \cdots \times 2 \times 3 \times 1 \end{aligned}$$

Comme  $n$  ou  $(n-1)$  est pair, nous retrouvons bien par le calcul que  $C(n)$  est pair.

BROUILLON - TEXTE EN DEVENIR  
BESOIIN DE RELECTURES EXTERNES