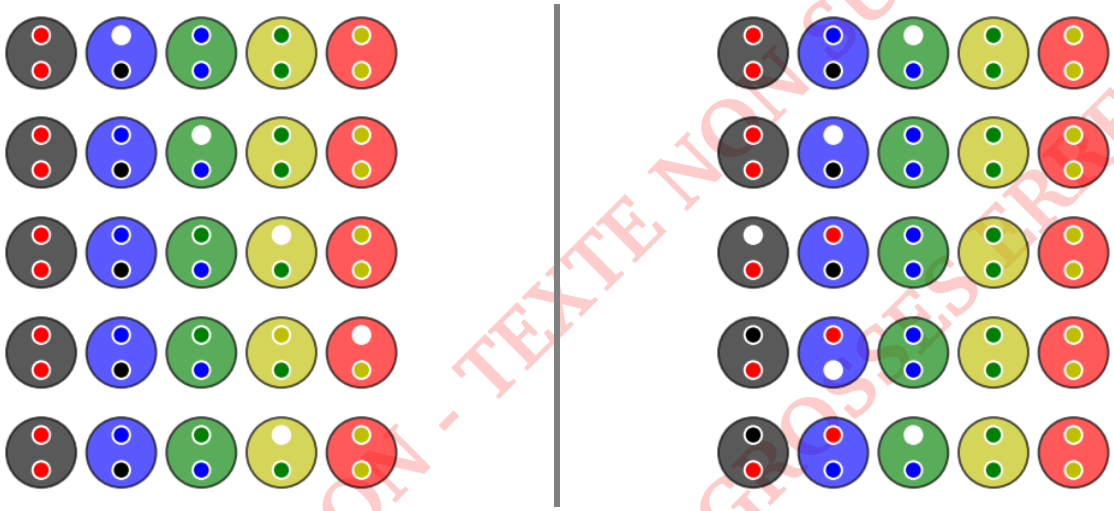
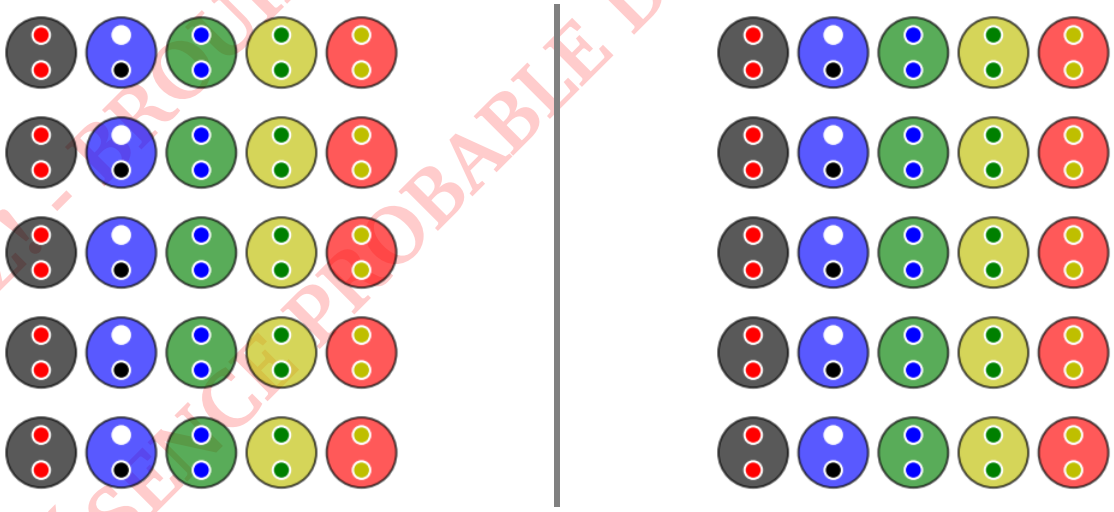
1 À la recherche d'une éventuelle solution optimale

1.1 Où tentons-nous d'aller ?

A CREUSER :
à cahque coup on va chercher à augmenter ou diminuer siuvant lsens de parcrousrs
par contre aux extrémités, on s'autoirsent à fair une boucle !
Il se trouve que la méthode « *on avance au mieux* » n'est pas la plus efficace comme le montre l'exemple suivant où six mouvements totalement inutiles sont effectués si bien que l'on n'a toujours pas gagné à la neuvième étape.



Or on peut gagner ici en seulement 9 coups ! Voici les mouvements à faire [1]. DESSIN FAUX !!!!!!!!!!!!!



Commençons par proposer une nouvelle modélisation du jeu qui soit bien plus fine que celles utilisées dans les deux sections précédentes. Que cherche-t-on à faire ? Trouver une solution peu coûteuse. Très bien ! Mais dans ce cas, comment évalue-t-on ce coup ? Nous choisissons de chercher à minimiser le nombre de déplacements du trou (donc toute opération autre que le déplacement d'un jeton ne sera pas comptabilisée). Dans le premier cas ci-dessus, le coût est strictement plus grand que 9, tandis que dans le second cas, il vaut exactement 9.

Soit une configuration $\mathcal{C}$

1. Notez au passage les enseignements que l'on peut tirer d'une configuration très, très particulière.

?????????????????????????????????????? notation en ligne car pratique mais ici on s'autorise le passage de la base tout à droite à celle tout à gauche ???
. Pour chaque jeton j, nous notons *deg* j son degré d'éloignement qui est égal par définition aux nombres de bases entre sa base comprise et la base de sa couleur. Par exemple, au début des exemples ci-dessus, tous les jetons ont un degré d'éloignement égal à un.

Nous pouvons alors proposer la troisième méthode suivante (notez la concision de l'algorithme). PAS BON PAS BON PAS BON PAS BON PAS BON PAS BON PAS BON PAS BON PAS BON PAS BON (voir juste parès !!!!

Donnée : une configuration quelconque de début de jeu
Résultat : une configuration où tous les jetons sont rentrées dans leur base

Début

- **Tant Que** *la configuration contient un jeton qui n'est pas dans sa base* **:**
  - j_1 , j_2 , j_3 et j_4 désignent les jetons déplaçables.
  - Pour k allant de 1 à 4, on note d_k est le dégré d'éloignement du jeton j_k s'il était déplacé.
  - Déplacer réellement un jeton j_k, éventuellement choisi au hasard, tel que d_k soit minimal.

Les instructions étant sans ambiguïté, nous allons pouvoir nous attaquer très sérieusement à la validation des propriétés de « *finitude* » et de « *résolution* ».

Démonstration. ???????????????????
CODER POUR VOIR SI CEL MARCHE
PAR EXEMPLe, faire gaffe au cas ou les quatre jetons déplaçables ont le même degré dans la config du moment
r r t n j j b b v v
r r j n t j b b v v
pas possible car boucle infinie possible □

1.2 Recherche de toutes les configurations possibles

1.3 Analyse de l'algorithme donnant toutes les configurations possibles

2 Annexe : calculer le nombre de configurations possibles

On peut se demander combien de configurations de jeux sont possibles. C'est ce que nous allons chercher à calculer. Soit $C(n)$ le nombre de configurations de jeux pour $n \geqslant 1$ base(s) (par exemple, $C(1) = 1$). Nous avons au total $(2n - 1)$ jetons et n « *couleurs* » différentes. Nous dirons que le jeton associé au trou est le jeton noir. Enfin, nous allons noter $\mathcal{B}_1$, $\mathcal{B}_2$, . . . , et $\mathcal{B}_n$ les bases rangées dans un ordre quelconque. Dans la suite, nous supposerons $n \geqslant 2$.

Commençons par examiner les configurations telles que la base $\mathcal{B}_1$ contienne le trou. Nous avons alors deux situations [2].

1. $\mathcal{B}_1$ *contient aussi le jeton noir.* Dans ce cas, nous devons compter le nombre de façons différentes de placer $(n-1)$ paires de jetons de même couleur sur $(n-1)$ bases, sachant que sur chaque base l'ordre des jetons n'est pas important. Nous n'allons pas chercher à calculer ce nombre. Nous le notons juste $P(n-1)$ (les hypothèses sur les couleurs sont similaires à celles du jeu du baseball des couleurs sauf que l'on n'a plus de trou, ni de jeton noir associé au trou). Indiquons que l'on a clairement $P(1) = 1$.

2. Le lecteur notera que ce qui compte dans les raisonnements de cette section, c'est la couleur tirée et non le jeton tiré. C'est une subtilité importante à noter. Dans une première version de ce document, l'auteur avait raisonné sur les jetons, ce qui lui avait fait obtenir et « *démontrer* » une formule fausse.

2. $\mathcal{B}_1$ *ne contient pas le jeton noir.* Notant c la couleur du jeton dans la base $\mathcal{B}_1$, sur les $(n-1)$ bases restantes, il reste à placer $(n-2)$ paires de jetons de même couleur, ainsi que deux jetons de couleurs différentes « *sans jumeau* », à savoir les jetons de couleurs respectives c et noire. Interprétant le jeton de couleur c comme un trou, nous retombons sur une configuration d'un jeu de baseball des couleurs mais avec $(n-1)$ jetons. Ceci nous fait donc $C(n-1)$ possibilités associées à la couleur c.

D'après ce qui précède, il y a donc $P(n-1)+(n-1)C(n-1)$ configurations possibles telles que la base $\mathcal{B}_1$ contienne le trou. Dans $(n-1)C(n-1)$, le « *fois* $(n-1)$ » vient de ce que l'on a $(n-1)$ choix possibles pour la couleur c.

En reprenant un raisonnement analogue au précédent, et en notant qu'il y a n choix de bases où placer le trou, nous arrivons à la relation $C(n) = n\,[P(n-1)+(n-1)C(n-1)]$, soit la formule suivante :

$$C(n) = nP(n-1) + n(n-1)C(n-1) \tag{1}$$

Nous allons reprendre un raisonnement similaire à ce qui a été fait ci-dessus pour trouver une relation de récurrence pour tenter d'évaluer $P(n)$ le nombre de façons de placer n paires monochromes de jetons sur n bases [3]. De nouveau, nous allons d'abord raisonner sur la base $\mathcal{B}_1$. Deux cas sont possibles.

1. $\mathcal{B}_1$ *contient deux jetons de la même couleur* c. Dans ce cas, il y a $P(n-1)$ possibilités de placer les autres jetons sur les bases restantes. En effet, on a enlevé une paire monochrome de couleur c et la base $\mathcal{B}_1$ de couleur d. Si $c=d$, l'affirmation est évidente, sinon il suffit d'associer la couleur de jeton d à la couleur de base c pour les bases et les jetons restants.
2. $\mathcal{B}_1$ *contient deux jetons de couleurs différentes* c_1 *et* c_2. Dans les jetons restants, il y a juste deux jetons sans « *jumeau* », à savoir ceux de couleurs c_1 et c_2. Interprétant c_1 comme étant la « *couleur* » du trou, et c_2 celle du jeton associé au trou, nous avons alors une configuration de type baseball des couleurs pour les jetons et les bases restants. Nous avons donc $C(n-1)$ possibilités dans ce cas.

Dans le premier cas, pour la base $\mathcal{B}_1$ il y a n choix de couleurs, ce qui nous fait $nP(n-1)$ configurations avec une première base monochrome. Dans le second cas, pour la base $\mathcal{B}_1$ nous avons n choix pour la première couleur, et $(n-1)$ pour la seconde (car l'on veut deux couleurs différentes). Comme l'ordre ne compte pas, car tirer c_1 puis c_2, ou bien tirer c_2 puis c_1 nous donne à chaque fois la même base complétée, nous avons donc $\frac{n(n-1)}{2}$ façons de remplir la première base avec deux couleurs différentes. Nous en déduisons que nous avons $\frac{n(n-1)}{2}C(n-1)$ configurations avec une première base non monochrome. Nous arrivons finalement à la relation suivante.

$$P(n) = nP(n-1) + \frac{n(n-1)}{2}C(n-1) \tag{2}$$

En résumé, nous avons démontré les deux relations de récurrence suivantes pour $n \in \mathbb{N}^* - \{1\}$ avec les conditions initiales $C(1) = P(1) = 1$:

$$\begin{cases} C(n) = nP(n-1) + n(n-1)C(n-1) \\ P(n) = nP(n-1) + \dfrac{n(n-1)}{2}C(n-1) \end{cases} \tag{3}$$

Ceci n'est pas très difficile à programmer. En utilisant le logiciel SageMath directement en ligne à l'adresse suivante `https://sagecell.sagemath.org`, il suffit d'insérer le code de type Python suivant.

```
def C(n):
    if n == 1:
        return 1

    return n * P(n - 1) + n*(n-1)*C(n - 1)
```

3. Il ya autant de couleurs que de bases.

```
def P(n):
    if n == 1:
        return 1

    return n * P(n - 1) + n*(n-1)/2*C(n - 1)

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

Ceci nous donne les résultats suivants.

- $C(2) = 4$ ce qui est calculable directement en imaginant les configurations possibles avec deux bases.
- $C(3) = 33$, $C(4) = 480$ et $C(5) = 11\,010$.

Par contre, si l'on veut par exemple calculer $C(36)$, il faut être un peu plus précautionneux car certains calculs sont effectués plusieurs fois. Le code suivant permet d'obtenir instantanément $C(36)$ (alors que celui qui précède est très lent). L'idée utilisée ici est de stocker tout ce qui est calculé et de regarder si une valeur à calculer a déjà été stockée en mémoire. Si c'est le cas, on récupère directement cette valeur, sinon on la calcule et on la stocke en mémoire en vue d'une éventuelle utilisation plus tard.

```
valsP, valsC = {}, {}

def C(n):
    if n == 1:
        return 1

    global valsC

    if n in valsC:
        return valsC[n]

    val      = n * P(n - 1) + n*(n-1)*C(n - 1)
    valsC[n] = val

    return val

def P(n):
    if n == 1:
        return 1

    global valsP

    if n in valsP:
        return valsP[n]

    val      = n * P(n - 1) + n*(n-1)/2*C(n - 1)
    valsP[n] = val

    return val

print C(36).n(10)
```

Ceci nous donne $C(36) \approx 4{,}2 \times 10^{82}$ que l'on comparera à 10^{80} qui est une estimation du nombre d'atomes dans l'univers [4].

4. 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 $C(36)$ en associant chaque naturel à un seul atome de l'univers. Vertigineux !