

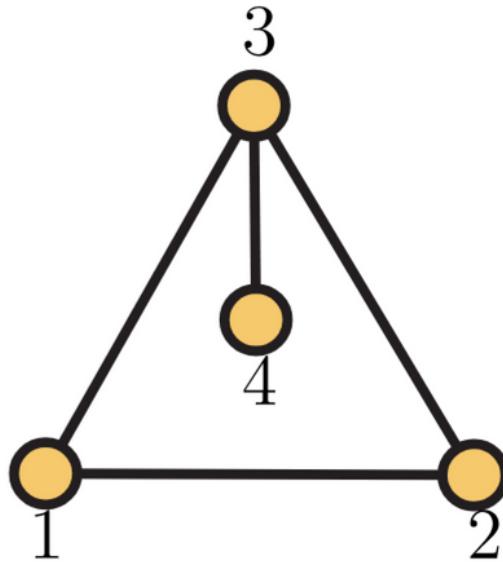
# Une méthode de dénombrement et de génération de graphes non-isomorphes

Marius Besognet--Lostia  
19869

# Graphe : définition

Ensemble  $(S, A)$  de sommets et d'arêtes

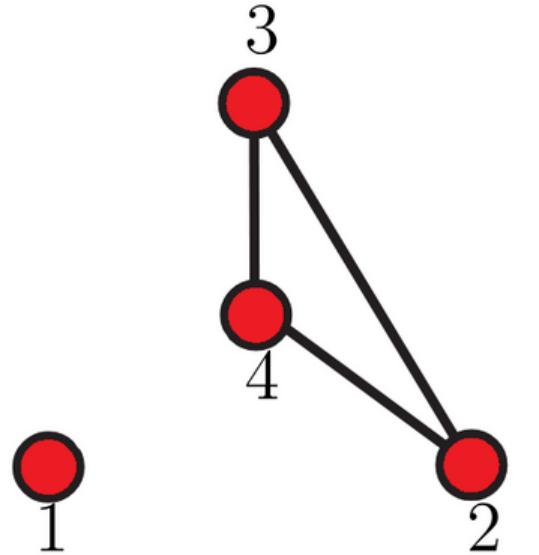
On les prendra ici non orientés, sans arêtes multiples



$$S = \{1, 2, 3, 4\}$$

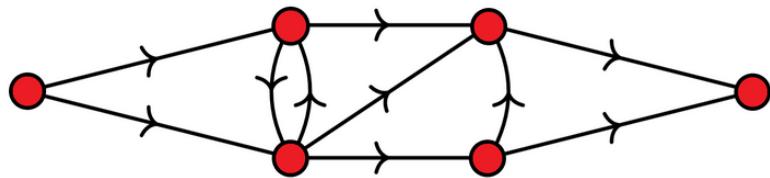
$$A = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{3, 4\}\}$$

Ils peuvent être non connexes

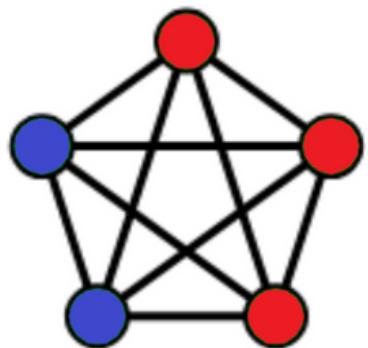


# Pourquoi dénombrer des graphes ?

Réseaux de neurones

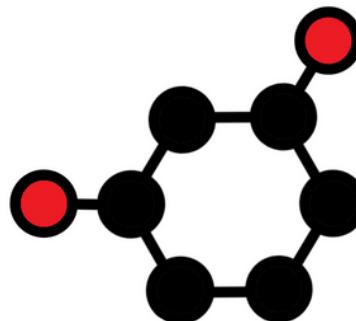
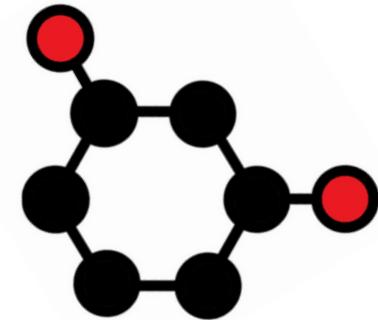
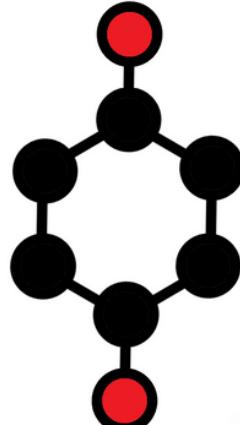


Théorie de Ramsey

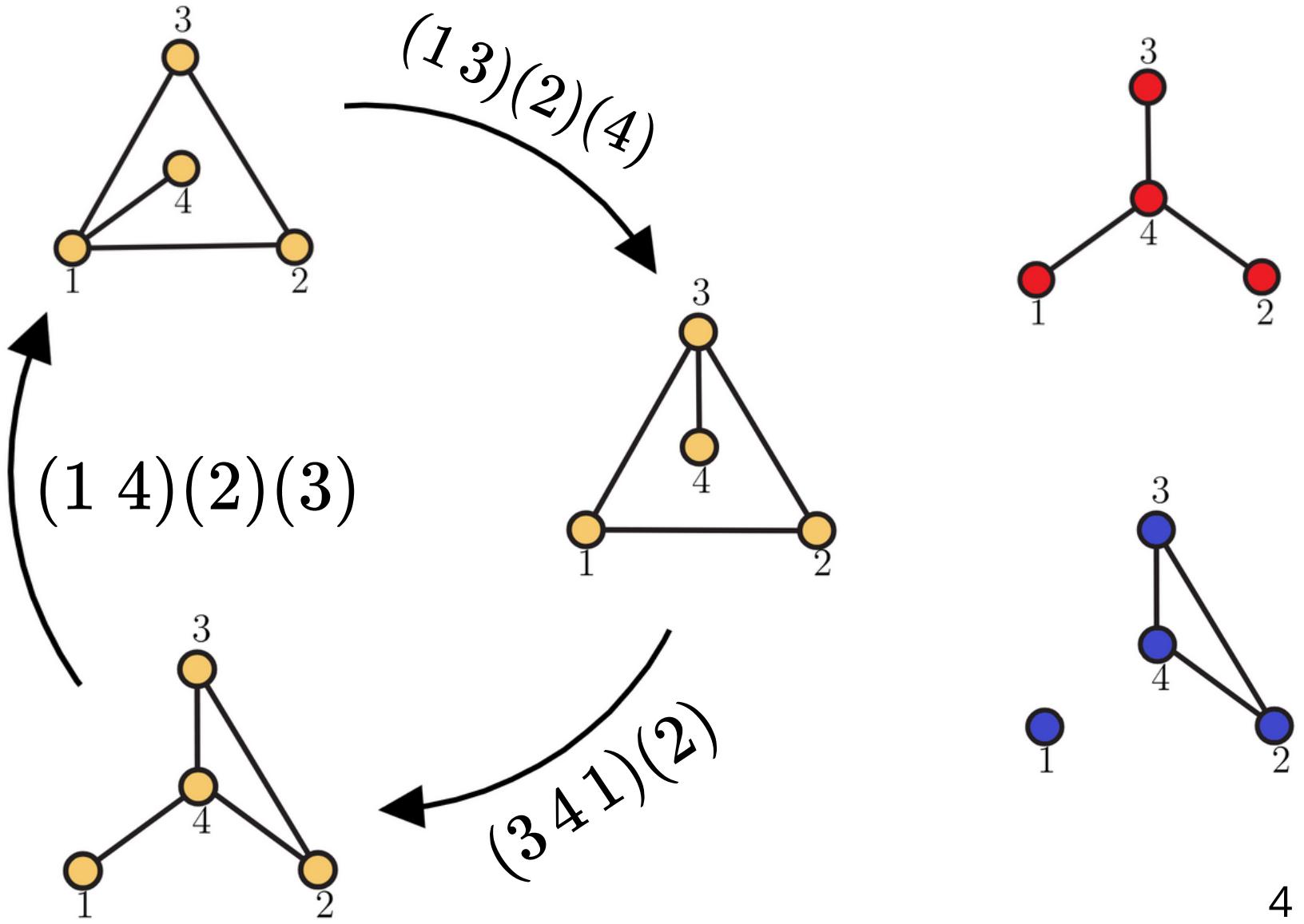


(Etude de  
coloriages de  
graphes  
complets)

Isomères de molécules



# Isomorphismes de graphes



# Comment les dénombrer ?

Dans le cas de sommets distinguables :

Pour un graphe à n sommets et k arêtes :

$$\binom{\binom{n}{2}}{k} \text{ possibilités}$$

Soit au total :

$$2^{\binom{n}{2}}$$

graphes à n sommets

Si les sommets sont indistinguables, il faut prendre en compte les isomorphismes possibles, on doit donc dénombrer les classes d'isomorphismes.

On a recours au théorème de dénombrement de George Pólya

## Actions de groupes :

Soit  $G$  un groupe et  $X$  un ensemble, on dit que  $G$  agit sur  $X$  en définissant :

$$G \times X \rightarrow X$$

$$(g, x) \mapsto g \cdot x$$

telle que, pour tout  $x \in X$  et pour tout  $g_1, g_2 \in G$ , on a

- (i)  $e \cdot x = x,$
- (ii)  $g_1 \cdot (g_2 \cdot x) = (g_1 * g_2) \cdot x.$

Dans notre exemple pour des graphes, le groupe de permutations agit naturellement sur les sommets.

## Orbites :

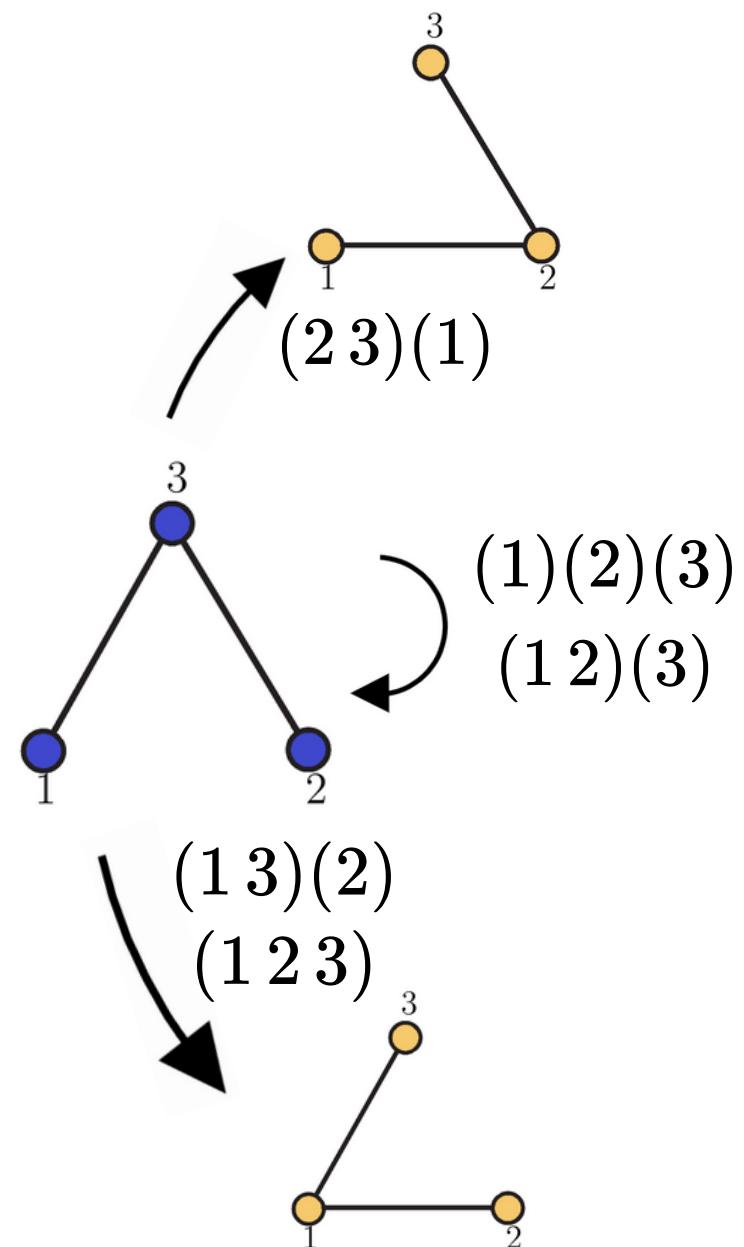
On définit l'orbite d'un élément  $x \in X$  comme :

$$Orb(x) = \{g \cdot x, g \in G\}$$

Les orbites définissent des classes d'équivalence qui partitionnent  $X$

On note  $X/G$  l'ensemble des classes d'équivalences

On cherche alors à déterminer le cardinal de  $X/G$



## Définitions intermédiaires :

Soit  $n \in \mathbb{N}$ , le nombre de sommets du graphe étudié.

Soit  $s \in \mathfrak{S}_3$ , qui agit sur les sommets du graphe, on peut l'écrire comme composition de cycles disjoints.

On définit alors le type de  $s$  comme :

$$type(s) = \lambda_s = (1^{a_1}, 2^{a_2}, \dots)$$

où  $a_i$  est le nombre de cycles de longueur  $i$  dans  $s$ .

$$(1)(2)(3) \quad \lambda_s = (1^3, 2^0, 3^0)$$

$$\mathfrak{S}_3 \quad \left. \begin{array}{c} (1\ 2)(3) \\ (1\ 3)(2) \\ (2\ 3)(1) \end{array} \right\} \lambda_s = (1^1, 2^1, 3^0)$$

$$(1\ 2\ 3) \quad \lambda_s = (1^0, 2^0, 3^1)$$

## Définitions intermédiaires :

On définit de plus :

$$\omega(\lambda) = |\{s \in S_n : \text{type}(s) = \lambda\}|$$

on a par ailleurs :

$$\omega(\lambda) = n! \prod_{1 \leq j \leq n} \frac{1}{j^{a_j} a_j!}$$

On écrit  $\lambda \vdash n$

$$(1)(2)(3) \quad \lambda_1 = (1^3, 2^0, 3^0) \quad \omega(\lambda_1) = 1$$

$$\left. \begin{array}{l} (1\ 2)(3) \\ (1\ 3)(2) \\ (2\ 3)(1) \end{array} \right\} \lambda_2 = (1^1, 2^1, 3^0) \quad \omega(\lambda_2) = 3$$

$$(1\ 2\ 3) \quad \lambda_3 = (1^0, 2^0, 3^1) \quad \omega(\lambda_3) = 1 \quad 9$$

## Le théorème de Pólya pour des graphes :

$$H(x) = \frac{1}{n!} \sum_{s \in \mathfrak{S}_n} \prod_j (1 + x^j)^{\alpha(j, s)}$$

Où  $H$  est le polynôme dont le coefficient de degré  $i$  est le nombre de graphes à  $n$  sommets et  $i$  arêtes

Et  $\alpha(j, s)$  le nombre d'orbites de longueur  $j$  induits par  $s$

Soit en termes de types  $\lambda$  :

$$H(x) = \frac{1}{n!} \sum_{\lambda \vdash n} \omega(\lambda) \prod_j (1 + x^j)^{\alpha(j, \lambda)}$$

Enumérons les graphes à 5 sommets :

$$H(x) = \frac{1}{n!} \sum_{\lambda \vdash n} \omega(\lambda) \prod_j (1 + x^j)^{\alpha(j, \lambda)}$$

La somme se fait sur les  $\lambda \vdash n$

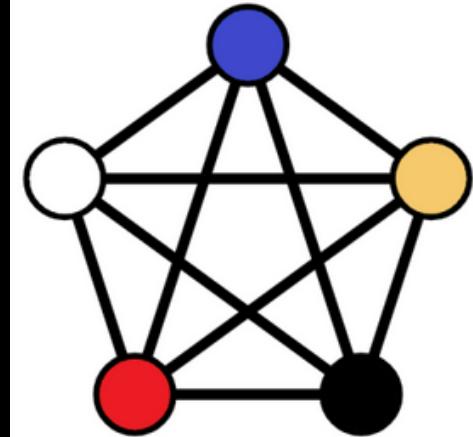
$$\{\lambda \vdash n\} = \{\lambda_s, s \in S_n\}$$

Pour  $n = 5$  les permutations associées sont des formes ci-contre :

Calculons par exemple le terme pour  
 $\lambda = (1^1, 2^0, 3^0, 4^1, 5^0)$

Qui correspond à un permutation de forme :

$$(1)(4)$$



(5),  
(1)(4),  
(1)(1)(3),  
(2)(3),  
(1)(1)(1)(2),  
(1)(2)(2),  
(1)(1)(1)(1)(1)

$$H(x) = \frac{1}{n!} \sum_{\lambda \vdash n} \omega(\lambda) \prod_j (1 + x^j)^{\alpha(j, \lambda)}$$

Le terme en  $\lambda = (1^1, 2^0, 3^0, 4^1, 5^0)$  est :  $\frac{5!}{1^1 1! 4^1 1!} (1+x^4)^2 (1+x^2)$

En effet, si on permute par exemple les sommets  $(1,2,3,4)$  en gardant fixe  $(5)$ , on a pour les arêtes :

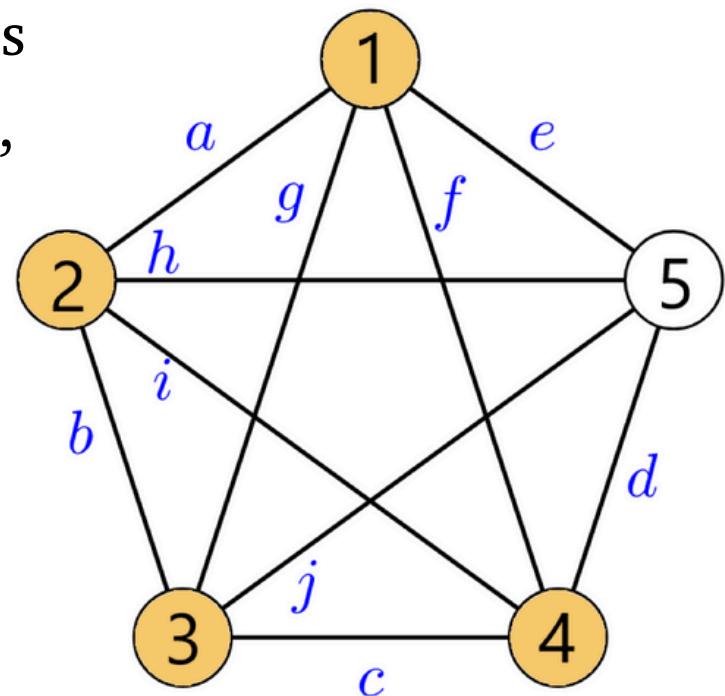
Deux cycles de longueur 4 :

$(e, h, j, d)$

$(a, b, c, f)$

Un cycle de longueur 2 :

$(g, i)$



En réitérant le procédé pour tous les  $\lambda \vdash n$ , on obtient à la fin :

$$H(x) = \frac{1}{5!} [24(1+x^5)^2 + 30(1+x^4)^2(1+x^2) + 20(1+x^3)^3(1+x) \\ + 20(1+x^6)(1+x^3)(1+x) + 15(1+x^2)^4(1+x)^2 + 10(1+x^2)^3(1+x)^4 + (1+x)^{10}]$$

$$H(x) = 1 + x + 2x^2 + 4x^3 + 6x^4 + 6x^5 + 6x^6 + 4x^7 + 2x^8 + x^9 + x^{10}$$

En évaluant en  $x = 1$  on trouve **34** classes d'isomorphismes pour les graphes à 5 sommets

Comment générer l'ensemble dénombré ?

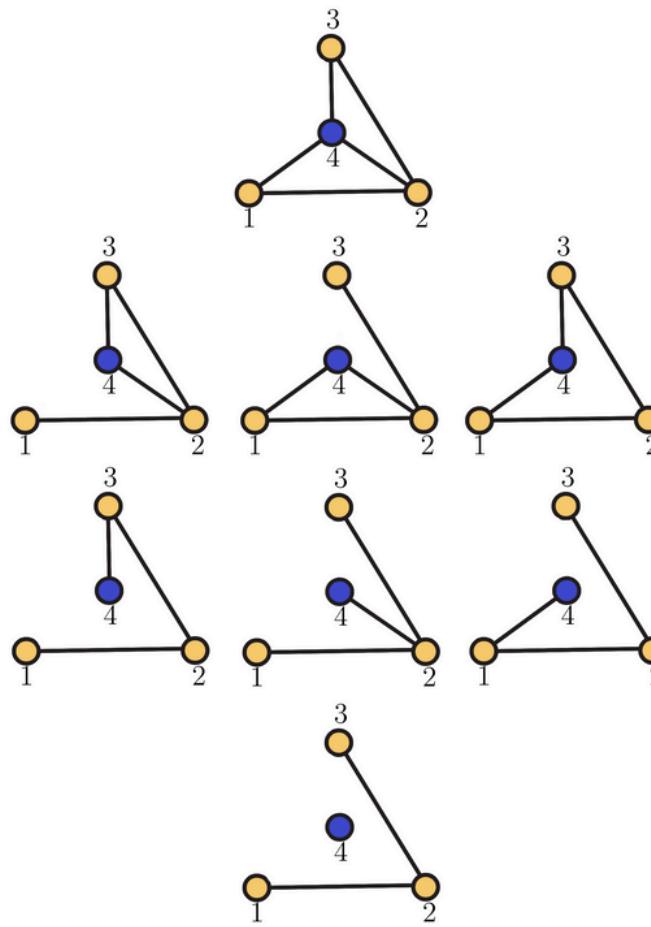
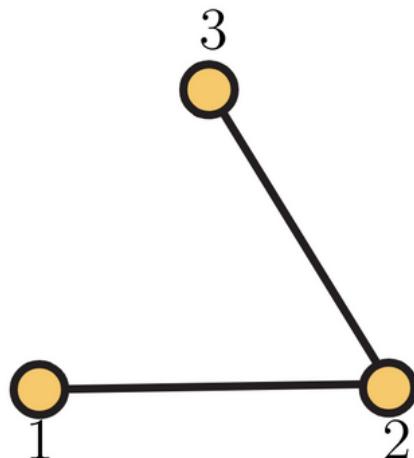
---

Comment générer un unique représentant de chaque orbite ?

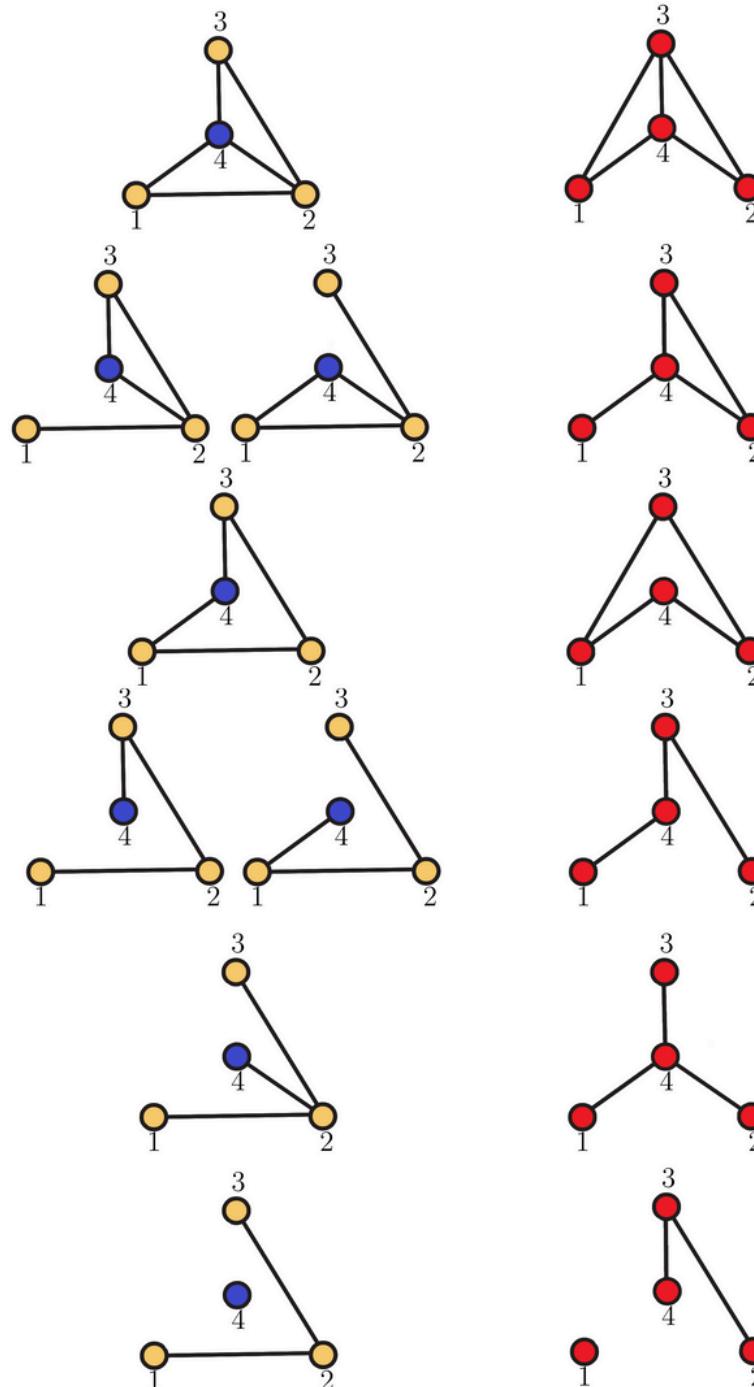
# Méthode de génération par construction de chemin canonique

On part d'une famille de représentants des orbites pour les graphes à n-1 sommets (une base)

Puis on augmente chaque représentant en l'augmentant de toutes les façons possibles



On transforme chaque graphe obtenu en l'élément canonique de son orbite :



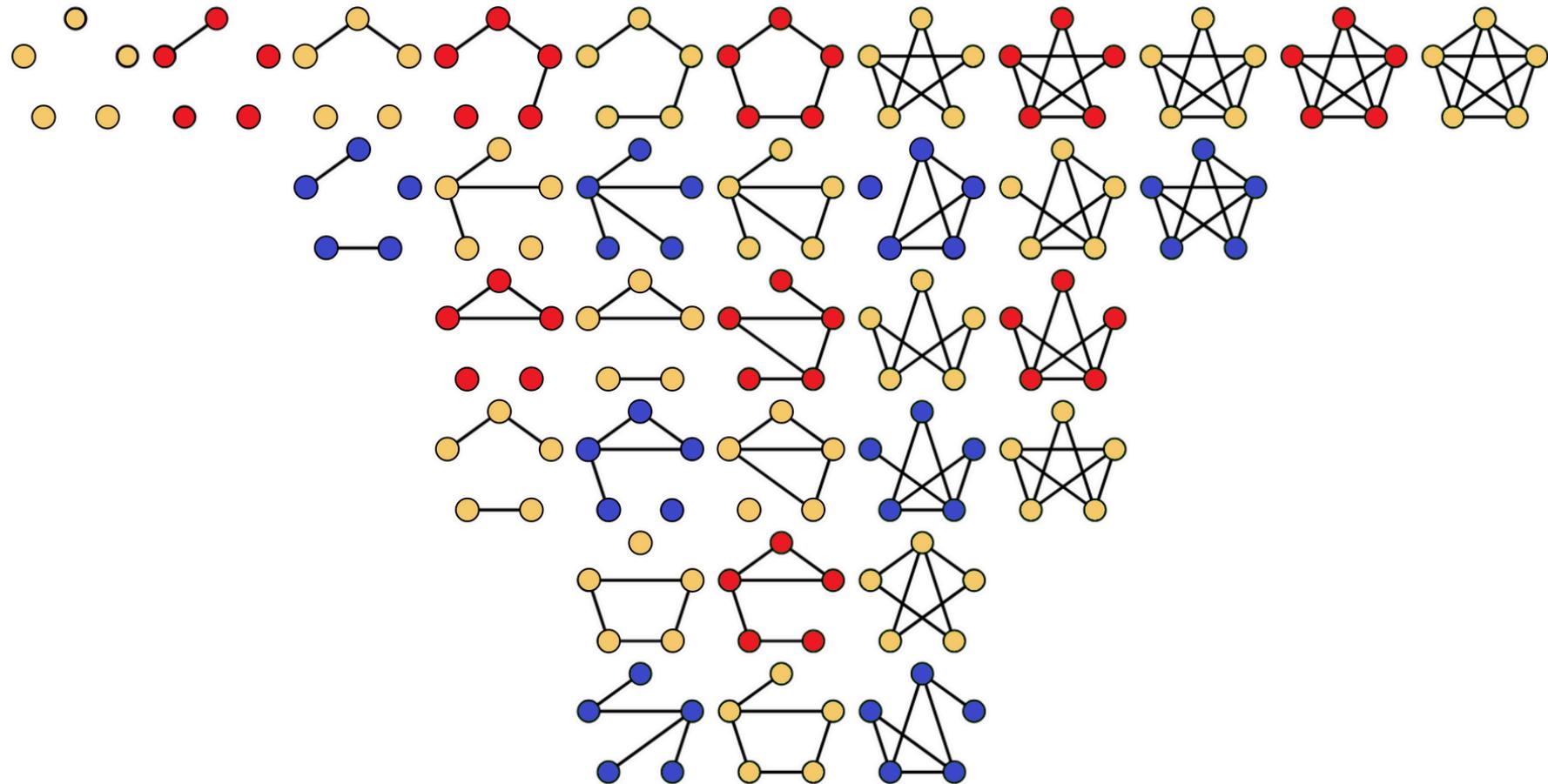
Il ne reste qu'à retirer les éléments redondants

# Résultats pour n=5

```
1 []
2 [(3, 4)]
3 [(2, 4), (3, 4)]
4 [(1, 4), (2, 4), (3, 4)]
5 [(0, 4), (1, 4), (2, 4), (3, 4)]
6 [(1, 4), (2, 3)]
7 [(0, 1), (2, 4), (3, 4)]
8 [(1, 4), (2, 3), (3, 4)]
9 [(2, 3), (2, 4), (3, 4)]
10 [(1, 4), (2, 3), (2, 4), (3, 4)]
11 [(0, 4), (1, 4), (2, 3), (2, 4), (3, 4)]
12 [(0, 4), (1, 3), (2, 3), (3, 4)]
13 [(1, 3), (1, 4), (2, 3), (2, 4)]
14 [(0, 4), (1, 2), (1, 3), (2, 4), (3, 4)]
15 [(0, 4), (1, 3), (2, 3), (2, 4), (3, 4)]
16 [(1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
17 [(0, 4), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
18 [(0, 3), (0, 4), (1, 3), (1, 4), (2, 3), (2, 4)]
19 [(0, 3), (0, 4), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
20 [(0, 3), (1, 2), (2, 4), (3, 4)]
21 [(0, 1), (2, 3), (2, 4), (3, 4)]
22 [(0, 1), (1, 4), (2, 3), (2, 4), (3, 4)]
23 [(0, 3), (0, 4), (1, 2), (1, 4), (2, 4), (3, 4)]
24 [(0, 3), (0, 4), (1, 2), (1, 4), (2, 3)]
25 [(0, 1), (0, 4), (1, 3), (2, 3), (2, 4), (3, 4)]
26 [(0, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
27 [(0, 3), (0, 4), (1, 2), (1, 4), (2, 3), (2, 4), (3, 4)]
28 [(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
29 [(0, 4), (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
30 [(0, 3), (0, 4), (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
31 [(0, 1), (0, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
32 [(0, 2), (0, 3), (0, 4), (1, 2), (1, 3), (1, 4), (2, 4), (3, 4)]
33 [(0, 2), (0, 3), (0, 4), (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
34 [(0, 1), (0, 2), (0, 3), (0, 4), (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
```

On retrouve bien les 34 classes d'isomorphismes prédictes par Pólya !

Soit de manière plus visuelle :



$$H(x) = 1 + x + 2x^2 + 4x^3 + 6x^4 + 6x^5 + 6x^6 + 6x^7 + 4x^8 + 2x^9 + x^{10}$$

# Annexes

Lemme de Burnside p.20

Indice de cycles, configuration p.21

Poids p.22

Lemme, constance sur les orbites p.23

Théorème de Pólya p.24-25

Calcul de  $\omega(\lambda)$  p.26

Cardinaux des orbites p.27-28

Théorème de Pólya pour les graphes p.29

Programme p.30-32

Références p.33

Soit  $G$  un groupe qui agit sur un ensemble  $X$ . Pour un élément  $x \in X$ , on définit le *stabilisateur de  $x$*  ainsi que *l'orbite de  $x$*  par

$$\begin{aligned}\text{Stab}_G(x) &:= \{g \in G : g \cdot x = x\}, \\ \text{Orb}_G(x) &:= \{g \cdot x : g \in G\}.\end{aligned}$$

Les orbites partitionnent  $X$

Si  $G$  est de cardinal fini, on a la formule d'orbite-stabilisateur :

$$|G| = |\text{Orb}_G(x)| |\text{Stab}_G(x)|$$

Lemme de Burnside :

*Soit  $G$  un groupe fini qui agit sur  $X$ . Alors*

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|,$$

où le membre de gauche représente le nombre d'orbites, et  $X^g := \{x \in X : g \cdot x = x\}$ .

*Démonstration.* Premièrement, on veut réécrire la somme de droite en fonction des éléments de  $X$ . Par la définition de  $X^g$  et de  $\text{Stab}_G(x)$ ,

$$\sum_{g \in G} |X^g| = \sum_{x \in X} |\text{Stab}_G(x)|. \quad (1)$$

Par la formule d'orbite-stabilisateur, on sait que l'on peut réécrire (1) en fonction des orbites sur  $X$ , et comme  $G$  est fini, (1.2) la formule d'orbite-stabilisateur s'applique et donc

$$\begin{aligned}|\text{Orb}_G(x)| &= \frac{|G|}{|\text{Stab}_G(x)|} \\ &\Downarrow \\ \sum_{x \in X} |\text{Stab}_G(x)| &= |G| \sum_{x \in X} \frac{1}{|\text{Orb}_G(x)|}.\end{aligned}$$

Finalement, comme les orbites de  $X$  ( $X/G$ ) partitionnent  $X$ , on décompose la somme sur toutes les classes d'équivalences

$$\sum_{x \in X} \frac{1}{|\text{Orb}_G(x)|} = \sum_{A \in X/G} \underbrace{\sum_{x \in A} \frac{1}{|A|}}_{\parallel 1} = |X/G|.$$

En combinant le tout,

$$\sum_{g \in G} |X^g| = |G| |X/G|,$$

et en divisant par le cardinal de  $G$  on obtient le résultat

Pour  $G$  un groupe de permutation, on définit l'*indice de cycles* par l'expression

$$\mathcal{Z}_G(x_1, x_2, \dots, x_n) := \frac{1}{|G|} \sum_{g \in G} x_1^{a_1} x_2^{a_2} \dots x_n^{a_n},$$

où le  $\text{type}(g) = (1^{a_1}, 2^{a_2}, \dots, n^{a_n})$ .

Soient  $X, Y$  deux ensembles finis, et  $G$  un groupe de permutation agissant sur  $X$

On dit que  $f_1, f_2$  deux fonctions de  $Y^X$  sont équivalentes

$$f_1 \sim f_2 \Leftrightarrow \exists g \in G, f_1(gx) = f_2(x),$$

et ce, pour tout  $x \in X$ . Ainsi,  $\sim$  définit une relation d'équivalence sur  $Y^X$ , et les classes d'équivalences  $C$  de  $Y^X / \sim$  sont appelées des *configurations*.

Soit  $R$  un anneau commutatif qui contient  $\mathbb{Q}$

La fonction *poids* est définie par

$$\begin{aligned} w : Y &\rightarrow R \\ y &\mapsto w(y). \end{aligned}$$

Aussi, on définit le *poids d'une fonction*  $f \in Y^X$  par

$$W(f) := \prod_{x \in X} w(f(x)).$$

Ces fonctions ne sont pas uniques, on fait le *choix* d'une fonction  $w$ , ce qui fixe la fonction  $W$ . Le poids d'une fonction est bien défini, en tant que fonction sur l'anneau commutatif  $R$ , étant donné que si  $f_1 \sim f_2$ , alors elles ont le même poids. En effet, il est facile de voir que

$$W(f_1) = \prod_{x \in X} w(f_1(x)) = \prod_{x \in X} w(f_1(gx)) = \prod_{x \in X} w(f_2(x)) = W(f_2).$$

On peut donc, sans ambiguïté, définir le *poids d'une configuration*,  $W(c)$ , par le poids d'un des représentants de la classe d'équivalence  $c \in C$ ,

$$W(c) := W(f).$$

En effet, on a mentionné que  $c$  forme une classe d'équivalence de la relation d'équivalence  $Y^X / \sim$ .

Soit  $X = X_1 \sqcup X_2 \sqcup \dots \sqcup X_k$ , et  $S \subset Y^X$  l'ensemble de fonctions suivant :

$\{f : X \rightarrow Y \mid f \text{ est constante sur } X_i \ \forall i\}$ . Notons que  $f$  n'a pas nécessairement la même valeur sur chaque  $X_i$ . Alors

$$\sum_{f \in S} W(f) = \prod_{i=1}^k \sum_{y \in Y} (w(y))^{|X_i|}. \quad (1)$$

Démonstration. On a dit que, pour  $f \in S$ ,  $f$  est constante sur chaque  $X_i$  individuellement. On peut alors réécrire  $f$  comme la composition de deux fonctions  $f = \phi \circ \psi$  avec

$$\begin{aligned} \psi : X &\rightarrow \{1, 2, \dots, k\} & \text{et} \\ x &\mapsto i \quad \text{t.q. } x \in X_i \end{aligned}$$

Notons que  $\psi$  est fixée, et que l'on a  $|Y|^k$  possibilités pour définir  $\phi$ . Ensuite, si on développe le produit dans l'énoncé, alors pour chaque  $i$ , on doit choisir un terme dans la somme qui a lieu sur  $Y$ , ce qui revient à construire une fonction

$$\phi : \{1, 2, \dots, k\} \rightarrow Y,$$

et donc un terme du développement du membre de droite de (1) peut s'écrire comme

$$w(\phi(1))^{|X_1|} w(\phi(2))^{|X_2|} \dots w(\phi(k))^{|X_k|} = \prod_{i=1}^k w(\phi(i))^{|X_i|}. \quad (2)$$

Rappelons-nous qu'il s'agit de la réécriture d'un seul terme. Et comme les  $f \in S$  sont de la forme  $f = \phi \circ \psi$ , alors

$$w(\phi(i))^{|X_i|} = \prod_{x \in X_i} w(\phi(\psi(x))) = \prod_{x \in X_i} w(f(x)).$$

On obtient donc que (1.5) devient

$$\begin{aligned} \prod_{i=1}^k \prod_{x \in X_i} w(f(x)) &= \left( \prod_{x \in X_1} w(f(x)) \right) \left( \prod_{x \in X_2} w(f(x)) \right) \dots \left( \prod_{x \in X_k} w(f(x)) \right) \\ &= \prod_{x \in X} w(f(x)) = W(f). \end{aligned}$$

Donc chaque terme du développement du membre de droite de (2) peut se réécrire comme  $W(f)$ . En procédant ainsi pour chaque terme, on obtiendra chaque fonction  $f \in S$ , et en sommant sur tout  $S$ , on obtiendra tous les termes du membre de droite de (2), ce qui démontre que

$$\sum_{f \in S} W(f) = \prod_{i=1}^k \sum_{y \in Y} w(y)^{|X_i|}.$$

Si  $C$  est un ensemble de configurations de  $Y^X$ , alors la fonction génératrice de configurations sera donnée par

$$F(C) := \sum_{c \in C} W(c).$$

### théorème de dénombrement de Pólya

Soit  $X, Y$  des ensembles finis,  $|X| = n$ , et  $G$  un groupe de permutation qui agit sur  $X$ . Si  $w$  est la fonction poids sur  $Y$ , alors la fonction génératrice de configuration est donnée par

$$F(C) = Z_G \left( \sum_{y \in Y} w(y), \sum_{y \in Y} w(y)^2, \dots, \sum_{y \in Y} w(y)^n \right).$$

*Démonstration.* Soit l'ensemble

$$S_w := \{f : X \rightarrow Y \mid W(f) = w\},$$

et notons

$$\psi_w(g) := \{f : X \rightarrow Y \mid W(f) = w \text{ et } f(x) = f(gx) \forall x\},$$

alors par le lemme de Burnside,  $S_w$  représentant le nombre d'orbites et avec la définition de  $\psi_w(g)$ , on obtient que

$$|S_w| = \frac{1}{|G|} \sum_{g \in G} |\psi_w(g)|. \quad (1)$$

Maintenant, comme toutes les configurations dans  $S_w$  ont un poids de  $w$ , en multipliant (1) par  $w$ , et en sommant sur tous les poids possibles, on obtient la fonction génératrice de configurations

$$\sum W(c) = \sum_w w |S_w| = \frac{1}{|G|} \sum_w \sum_{g \in G} w |\psi_w(g)|.$$

En notant  $\psi(g) := \{f : X \rightarrow Y \mid f = fg\}$ , on peut réécrire

$$\sum_w w |\psi_w(g)| = \sum_{f \in \psi(g)} W(f).$$

En interchangeant l'ordre des sommes, ce qui est permis étant donné qu'il s'agit de deux sommes finies, on obtient

$$\sum W(c) = \frac{1}{|G|} \sum_{g \in G} \sum_{f \in \psi(g)} W(f). \quad (2)$$

Pour poursuivre, il faut évaluer la somme sur  $\psi(g)$  de  $W(f)$ . Pour ce faire, rappelons-nous que tout  $g \in G$  permute  $X$  de façon cyclique, et disons que  $X$  se sépare donc en  $k$  cycles disjoints  $X_1, X_2, \dots, X_k$ . Pour une fonction  $f \in \psi(g)$ , on sait que

$$f(x) = f(gx) = f(g^2x) = \dots$$

Ce qui implique que  $f$  est constante sur chaque cycle  $X_i$  de  $X$ . Ainsi, le lemme p.23 s'applique et on obtient que

$$\sum_{f \in \psi(g)} W(f) = \prod_{i=1}^k \sum_{y \in Y} w(y)^{|X_i|}. \quad (3)$$

Comme il a déjà été mentionné, on sait qu'une permutation donne lieu à une partition, donc disons que  $\text{type}(g) = (1^{a_1}, 2^{a_2}, \dots, k^{a_k})$ . Alors, dans les nombres  $|X_1|, |X_2|, \dots, |X_k|$ , on a que  $i$  apparaît  $a_i$  fois pour chaque  $i \in \{1, 2, \dots, k\}$ . De cette façon, (3) devient

$$\sum_{f \in \psi(g)} W(f) = \left( \sum_{y \in Y} w(y) \right)^{a_1} \left( \sum_{y \in Y} w(y)^2 \right)^{a_2} \dots \left( \sum_{y \in Y} w(y)^k \right)^{a_k}. \quad (4)$$

Finalement, en remplaçant (4) dans (2), on obtient la fonction génératrice de configuration

$$\begin{aligned} \sum W(c) &= \frac{1}{|G|} \sum_{g \in G} \left( \left( \sum_{y \in Y} w(y) \right)^{a_1} \left( \sum_{y \in Y} w(y)^2 \right)^{a_2} \dots \left( \sum_{y \in Y} w(y)^k \right)^{a_k} \right) \\ &= Z_G \left( \sum_{y \in Y} w(y), \sum_{y \in Y} w(y)^2, \dots, \sum_{y \in Y} w(y)^k \right) \end{aligned}$$

ou sous forme close

$$F(C) = \frac{1}{|G|} \sum_{g \in G} \prod_{i=1}^k \left( \sum_{y \in Y} w(y)^i \right)^{a_i}$$

où  $\text{type}(g) = (1^{a_1}, 2^{a_2}, \dots, k^{a_k})$ .

## Calcul de $\omega(\lambda)$

Soit  $n \in \mathbb{N}$  et  $\lambda \vdash n$

$$\omega(\lambda) := |\{s \in S_n : \text{type}(s) = \lambda\}|$$

Pour  $n \in \mathbb{N}$  et  $\lambda \vdash n$ ,  $\lambda = (1^{a_1}, 2^{a_2}, \dots, k^{a_k})$  pour  $k \leq n$ , on a

$$\omega(\lambda) = n! \prod_{1 \leq j \leq n} \frac{1}{j^{a_j} a_j!}$$

$$\frac{n!}{\prod_{1 \leq j \leq n} a_j!}$$

est le nombre de façons de disposer de  $n$  éléments dans  $k$  boîtes. Or, de cette façon, on compte des éléments plusieurs fois, car on fait une distinction entre les boîtes contenant un même nombre d'éléments. Cependant,  $(1)(2), \dots, (k-1, k)$  est la même permutation que  $(2)(1), \dots, (k-1, k)$ . Il faut donc rediscuter par la taille des boîtes,  $j$ , ainsi que le nombre de fois qu'elles apparaissent,  $a_j$ .

# Calcul de $\alpha(j, s)$

On appellera des cycles de type I les cycles entre deux ensembles de sommets agissant sous différentes permutations. Les cycles de type II seront les cycles à l'intérieur d'un ensemble agissant sous une même permutation.

$$[a, b] := \text{ppcm}(a, b)$$

$$(a, b) := \text{pgcd}(a, b)$$

(Longueur des orbites type I)

Soit deux ensembles disjoints  $C_n$  et  $C_m$ , respectivement de cardinalité  $n$  et  $m$ , et que l'on permute le premier ensemble selon la permutation  $s_1 = (1, 2, \dots, n)$  et le deuxième par  $s_2 = (1, 2, \dots, m)$ , alors une arête allant de  $C_n$  vers  $C_m$  induira un cycle de longueur  $[n, m]$ .

*Démonstration.* SPDG, disons  $n < m$ . Soit une arête allant de  $n_i$  vers  $m_j$ ,  $1 \leq i \leq n$  et  $1 \leq j \leq m$ , en appliquant chacune des permutations  $n$  fois, alors  $n_i$  sera retourné à sa place, tandis que  $m_j$  sera en position  $j + n \pmod{m}$ . Il faut donc appliquer les permutations  $m$  fois pour que  $m_j$  retrouve son emplacement initial, mais cette fois c'est  $n_i$  qui sera en position  $i+m \pmod{n}$ . On procède ainsi jusqu'à ce que les deux soient à leur place, soit  $[n, m]$  fois.

On sait que les orbites seront de longueur  $[n, m]$ .

Sachant qu'il y a  $nm$  arêtes possibles entre ces deux ensembles, on a directement le nombre d'orbites

$$\frac{nm}{[n, m]} = (n, m).$$

# Calcul de $\alpha(j, s)$

(Nombre et longueur des orbites type II)

Soit un ensemble  $C_n$  comportant  $n$  sommets, avec  $n$  pair, et que l'on permute selon la permutation  $s_n = (1, 2, \dots, n)$ , alors il y aura  $n/2 - 1$  cycles de longueur  $n$  et 1 cycle de longueur  $n/2$ . Pour un ensemble  $C_m$  comportant  $m$  sommets, avec  $m$  impair, que l'on permute selon la permutation  $s_m = (1, 2, \dots, m)$ , alors il y aura  $(m - 1)/2$  cycles de longueur  $m$ .

*Démonstration.* D'abord, imaginons que nos sommets sont arrangés en polygone régulier. Commençons par le cas pair. Si on regarde ce qui se produit avec une arête entre deux sommets adjacents, disons le sommet 0 et 1, lorsque l'on permute elle fera le tour du polygone en engendrant un cycle de longueur  $n$ . En commençant du même sommet, si on saute un sommet et que l'on relie le sommet 0 au sommet 2, en permutant on engendrera un cycle de longueur  $n$  encore une fois. Il en sera de même jusqu'à ce qu'on relie deux sommets diamétralement opposés. Autrement dit, on peut sauter jusqu'à  $n/2 - 1$  sommets avant de rejoindre le sommet diamétralement opposé. Pour les sommets diamétralement opposés, en effectuant  $n/2$  fois la permutation, on retombera sur l'arête joignant les deux mêmes sommets, mais dans l'autre "sens". Or, comme nos arêtes sont non orientées, on a terminé ce cycle et il s'agit d'un cycle de longueur  $n/2$ . Donc, au total

28

$$\frac{n}{2} - 1 \text{ cycles de longueur } n \quad \text{et} \quad 1 \text{ cycle de longueur } \frac{n}{2}$$

Soit  $V$  un ensemble de sommets et soit  $E$  l'ensemble des arêtes sur le graphe, alors  $E^V$  représente l'attribution ou non d'une connexion entre deux sommets. En fait, pour respecter la définition d'un graphe, il faut modifier la notation un peu,

$$f : V \times V \longrightarrow E$$

$$(v_i, v_j) \longmapsto \begin{cases} 1 & \text{si } v_i \text{ et } v_j \text{ sont connectés,} \\ 0 & \text{sinon.} \end{cases}$$

Maintenant, comme on cherche une fonction génératrice, pour chaque arête dans l'ensemble  $E$ , donnons un poids de  $x^0$  si aucun lien n'existe, et  $x^1$  si un lien existe. Par souci de simplicité, notons  $E^{V \times V} =: E^V$ . Par le théorème de Pólya, on sait que notre fonction génératrice de graphe aura la forme suivante :

$$|E^V / S_n| = \frac{1}{|S_n|} \sum_{s \in S_n} \prod_{i=1}^k (1 + x^i)^{a_i},$$

pour  $\text{type}(s) = (1^{a_1}, \dots, k^{a_k})$ .

*Soit  $\Gamma = (V, E)$  un graphe à  $|V| = n$  sommets indistinguables, alors la fonction génératrice du nombre de graphes à isomorphisme près est donnée par*

$$H(x) = \frac{1}{n!} \sum_{s \in S_n} \prod_j \omega(\lambda_s) (1 + x^j)^{\alpha(j, s)},$$

où  $\lambda_s = (1^{a_1}, 2^{a_2} \dots)$  est la partition associée à la permutation  $s$ .

```

1 import itertools
2
3
4 # Renvoie un label approximant les orbites d'un graphe non orienté
5 # Chaque graphe de même orbite a le même label (mais pas injectif)
6 def label_orbite(graph, n):
7     labels = [1] * n
8     for r in range(2):
9         somme_vois = [0] * n
10        for i, j in graph:
11            somme_vois[i] += labels[j]
12            somme_vois[j] += labels[i]
13        for i in range(n):
14            labels[i] = somme_vois[i]
15    return labels
16
17
18 # Renvoie l'inverse d'une permutation
19 def inverse_permutation(perm):
20     n = len(perm)
21     inverse = [None] * n
22     for i in range(n):
23         inverse[perm[i]] = i
24     return inverse
25
26
27 # Renvoie une permutation qui trie les étiquettes
28 def permutation_triant_label(labels):
29     return inverse_permutation(sorted(range(len(labels)), key=lambda i: labels[i]))
30
31
32 # Renvoie le graphe où le noeud i devient perm[i]
33 # C'est à dire qu'on permute le graphe avec la permutation perm
34 def graphe_permute(perm, graph):
35     perm_graph = [(min(perm[i], perm[j]), max(perm[i], perm[j])) for (i, j) in graph]
36     perm_graph.sort()
37     return perm_graph

```

```

40 # Yield chaque permutation stabilisant le label, en considérant les transpositions
41 def label_stabilisateur(labels):
42     factors = (
43         itertools.permutations(block)
44         for _, block in itertools.groupby(range(len(labels)), key=lambda i: labels[i])
45     )
46     for subperms in itertools.product(*factors):
47         yield [i for subperm in subperms for i in subperm]
48
49
50 # Renvoie le graphe étiqueté canoniquement (selon les label triés, et le plus grand)
51 # Et la position du dernier label
52 def graphe_canonique(graph, n):
53     labels = label_orbite(graph, n)
54     sorting_perm = permutation_triant_label(labels)
55     graph = graphe_permute(sorting_perm, graph)
56     labels.sort()
57     return max(
58         (graphe_permute(perm, graph), perm[sorting_perm[n - 1]])
59         for perm in label_stabilisateur(labels)
60     )
61
62 # Renvoie la liste des permutations qui stabilisent le graphe
63 def stabilisateur_graphe(graph, n):
64     return [
65         perm
66         for perm in label_stabilisateur(label_orbite(graph, n))
67         if graphe_permute(perm, graph) == graph
68     ]
69
70
71 # Yield les sous ensembles de range(n+1) .
72 def power_set(n):
73     for r in range(n + 1):
74         for s in itertools.combinations(range(n), r):
75             yield list(s)
76
77
78 # renvoie l'ensemble permuté
79 def permute_ensemble(perm, s):
80     perm_s = [perm[i] for i in s]
81     perm_s.sort()
82     return perm_s

```

```

85 # Si s est canonique, renvoie la liste des permutations qui stabilisent s
86 # Sinon renvoie None
87 def stabilisateur_ensemble(s, group):
88     stabilizer = []
89     for perm in group:
90         perm_s = permute_ensemble(perm, s)
91         if perm_s < s:
92             return None
93         if perm_s == s:
94             stabilizer.append(perm)
95     return stabilizer
96
97 # Générateur des représentants de classes d'isomorphisme de graphes non orientés
98 def enumerer_graphes(n):
99     assert n >= 0
100    if n == 0:
101        yield []
102        return
103    for subgraph in enumerer_graphes(n - 1):
104        sub_stab = stabilisateur_graphe(subgraph, n - 1)
105        for neighbors in power_set(n - 1):
106            stab = stabilisateur_ensemble(neighbors, sub_stab)
107            if not stab:
108                continue
109            new_edges = [(min(i, n - 1), max(i, n - 1)) for i in neighbors]
110            graph, i_star = graphe_canonique(subgraph + new_edges, n)
111            if i_star == n - 1:
112                yield graph

```

## Références :

- [1] GEORGE PÓLYA : Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds : *Acta Mathematica*, 1937 ; Springer 1987 pour la traduction par R.C. Reads
- [2] REDFIELD, J. HOWARD : The Theory of Group-Reduced Distributions : 1927 *American Journal of Mathematics*, DOI : 10.2307/2370675
- [3] GAËTAN CHENEVIER : Cours d'Algèbre I à l'ENS : URL : [http://gaetan.chenevier.perso.math.cnrs.fr/ALG1/Algebre\\_ENS\\_Chenevier.pdf](http://gaetan.chenevier.perso.math.cnrs.fr/ALG1/Algebre_ENS_Chenevier.pdf)
- [4] J. HARRIS, J.HIRST, M.MOSSINGHOFF : Combinatorics and graph theory : Springer, DOI : [https://doi.org/10.1007/978-0-387-79711-3\\_2](https://doi.org/10.1007/978-0-387-79711-3_2)
- [5] ANTHONY RICHARD : Application du théorème de Pólya pour l'énumération d'une famille de graphes : URL : <https://corpus.ulaval.ca/server/api/core/bitstreams/3ff49b4e-3a3a-4297-82e5-ff0bee1af85c/content> pour les preuves
- [6] D. BABIĆ , D.J. KLEIN, J. VON KNOP & N. TRINAJSTIĆ : Combinatorial Enumeration in Chemistry : URL : <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=23c87484ba64df4ea6a803be6d4e49016f59b3d8>
- [7] BRENDAN D. MCKAY : Isomorph-free exhaustive generation : URL : <https://users.cecs.anu.edu.au/~bdm/papers/orderly.pdf>
- [8] BRENDAN D. MCKAY, GORDON F. ROYIE : Constructing the cubic graphs on up to 20 vertices : URL : <https://users.cecs.anu.edu.au/~bdm/papers/Gobstoppers.pdf>

Applied Combinatorics de Mitchel T. Keller, William T. Trotter pour la présentation des graphes  
Inspiration de Piet Mondrian pour le décor des diapositives