GIBERT Alexia
HALLEZ Manon

Année 2014/2015

MATh.en.JEANS

Compter les coloriages

Dans notre sujet, nous avons considéré des polygones convexes réguliers à n côtés et avons colorié chacun de leurs côtés. La question posée était de savoir de combien de façons possibles un tel polygone pouvait-il être colorié ?

Dans notre sujet, certaines règles ont été établies :
- Ne mettre qu'une couleur par côté ;
- Possibilité de pivoter le polygone vers la droite ou vers la gauche autant de fois que souhaité, ce qui a permis de réduire le nombre de coloriages différents d'un polygone pour un nombre de couleurs fixé ;
- Interdiction de retourner un polygone à la façon d'un reflet dans le miroir (symétrie axiale).

Sujet encadré par David Gréau, professeur de mathématiques, et proposé par François Ducrot, chercheur à la faculté des sciences d'Angers.

SOMMAIRE

I – Résultats obtenus par énumération

A) n côtés, une couleur

Tout d'abord, le cas le plus simple est évidemment un polygone avec un nombre de côtés quelconque, noté n, et une seule couleur pour le colorier. Il n'y a alors qu'une seule solution.

B) Énumération à l'aide d'un algorithme

Nous avons commencé par compter le nombre de combinaisons pour un cas donné manuellement, mais il y avait un nombre de possibilités très important, et donc un risque d'erreur élevé. C'est pourquoi nous nous sommes tournées vers une approche algorithmique.

1°) Objectif de notre algorithme

Nous avons crée un algorithme calculant le nombre de coloriages possibles en fonction du nombre de couleurs et du nombre de côtés. Il permet de déterminer les différentes combinaisons de coloriage pour n côtés et m couleurs.

L'inconvénient de cet algorithme, c'est qu'il ne va pas au delà d'un certain nombre de côtés et de couleurs car il dépasse vite les capacités de calcul d'Algobox. En effet, à partir d'un polygone avec 7 côtés notre algorithme ne fonctionne plus. Cela est lié au fait qu'avec n côtés et c couleurs, l'algorithme doit envisager c^n combinaisons.

Nous nous sommes ensuite basées sur les valeurs données par l'algorithme pour trouver des formules liant les différents nombres de possibilités.

2°) Explication de l'algorithme pas à pas

Les différentes couleurs sont remplacées par des nombres, par exemple si on a deux couleurs, rouge et bleu, elles sont désignées respectivement par les nombres 1 et 2. Dans un premier temps l'algorithme détermine donc toutes les listes de nombres possibles, puis dans un deuxième temps élimine celles qui sont identiques lorsqu'on tourne notre figure.

Exemple : cas d'un triangle colorié avec trois couleurs bleu, vert, rouge désignées respectivement par les chiffres 0, 1, 2.

Listes :	Éliminations combinaisons identiques :
012	012
021	021
102	~~102~~
120	~~120~~
210	~~210~~
201	~~201~~

Nous avons éliminé les combinaisons 102 et 210, car si on décale d'un ou deux crans, on retombe sur notre combinaison 021. Ainsi, ces trois polygones sont coloriés de la même manière. Nous avons le même cas avec les combinaisons 120 et 201 : si on fait un décalage d'un ou deux crans, on retombe sur la combinaison 012.
Dans ce cas là, il y a donc deux combinaisons possibles.
L'algorithme que nous avons utilisé commence par créer une liste de n éléments. Cette liste stocke la couleur de chaque côté (la couleur étant représentée par un nombre de 0 à c-1).
La fonction pow(c, n) représente ensuite le nombre de combinaisons possibles avec n côtés et c couleurs.
On compte ensuite le nombre de couleurs différentes présentes sur les côtés, et on les stocke dans C. Pour cela, la liste l1[i0] permet ici de déterminer si une couleur a déjà été comptée. (1)
On considère la liste comme une nombre écrit en base c (nombre de couleurs). Ensuite, on effectue une rotation sur la liste jusqu'à obtenir le plus grand nombre possible.
On réutilise l'écriture en base c, et on l'applique sur chaque combinaison déjà stockée dans L.
Si les deux nombres obtenus sont égaux, alors la combinaison testée est déjà présente dans la liste, et b vaut 0.
Si b vaut 1, c'est à dire si la combinaison n'est pas présente dans la liste, on la rajoute.
On crée ensuite la combinaison suivante.

Voir algorithme en Annexe

C) Les résultats obtenus

L'algorithme nous a permis d'obtenir les résultats suivants : (2)

Nombre de côtés du polygone / Nombre de couleurs	**3**	**4**	**5**	**6**
1	1	1	1	1
2	2	4	6	12
3	2	9	39	91
4		6	48	260
5			24	360
6				120

II – Conjecture permettant de compter n côtés et deux couleurs

Nous avons trouvé une conjecture nous permettant de compter tous les cas à n côtés et deux couleurs. Nous allons dans cette partie différencier les polygones possédant un nombre premier de côtés et les autres.

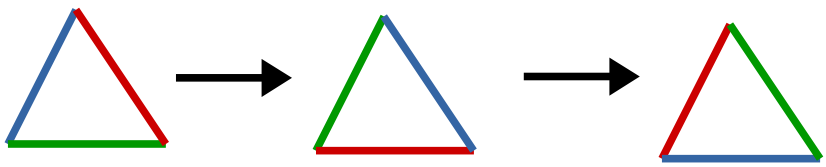
A) Définitions

Nous allons commencer par donner quelques définitions sur lesquelles s'appuiera notre conjecture.

Tout d'abord, un **décalage** c'est lorsqu'on part d'une combinaison initiale (par exemple, pour un carré et deux couleurs, on va partir de la combinaison B B R B), et qu'on la décale d'un certain nombre de crans (dans le cas de notre carré, si on décale de deux crans on obtient la combinaison R B B B).

Une **répétition**, c'est quand on décale d'un certain nombre de crans et qu'on obtient la même combinaison que celle prise initialement (par exemple, si notre carré a une combinaison initiale B R B R et qu'on décale de deux crans, on retrouve la même chose).

Par exemple, si on a un triangle (3 côtés), on a trois positions différentes. Nous considérons donc ces trois triangles comme identiques, et la combinaison Bleu – Rouge – Vert sera la même que la combinaison Vert – Bleu – Rouge et que la combinaison Rouge – Vert – Bleu. (3)



B) Cas des polygones avec un nombre de côtés premier

1°) Explication

Lorsque notre polygone possède un nombre n de côtés qui est un nombre premier, il s'agit du cas le plus simple puisque nous avons trouvé une formule marchant dans tous les cas.

Théorème : Soit n le nombre de côtés.
Le nombre total de combinaisons possibles est $(2^n-2)/n$

Démonstration :
Soit n le nombre de côtés.
On sait que pour chaque côté du polygone, il va y avoir deux possibilités (choix entre les deux couleurs). Pour commencer il y a donc 2^n possibilités pour

un polygone de n cotés.

Cependant, deux combinaisons sont à éliminer : en effet, si les deux couleurs sont bleu et rouge, cette formule va prendre en compte la combinaison tous les côtés sont bleus et la combinaison tous les côtés sont rouges. Or, notre polygone ne serait plus colorié que d'une seule couleur. On obtient ainsi 2^n-2 possibilités.

Enfin, on considère qu'on peut tourner notre polygone. Or, on sait qu'un polygone à n côtés peut avoir n positions différentes obtenues par n rotations. On doit donc finir par diviser par le nombre de rotations possibles. (4)

2°) Exemples et contre-exemple

Prenons tout d'abord l'exemple d'un triangle, donc n=3, colorié de deux couleurs différentes.
Le nombre de combinaisons étant donné par $(2^n-2)/n$, il y a donc $(2^3-2)/3 = 2$ possibilités pour un triangle. Cela correspond bien au résultat donné par l'algorithme.

Prenons ensuite l'exemple d'un pentagone, donc n=5, colorié de deux couleurs différentes.
Le nombre de combinaisons étant donné par $(2^n-2)/n$, il y a donc $(2^5-2)/5 = 6$ possibilités pour un pentagone. Cela correspond bien au résultat donné par l'algorithme.

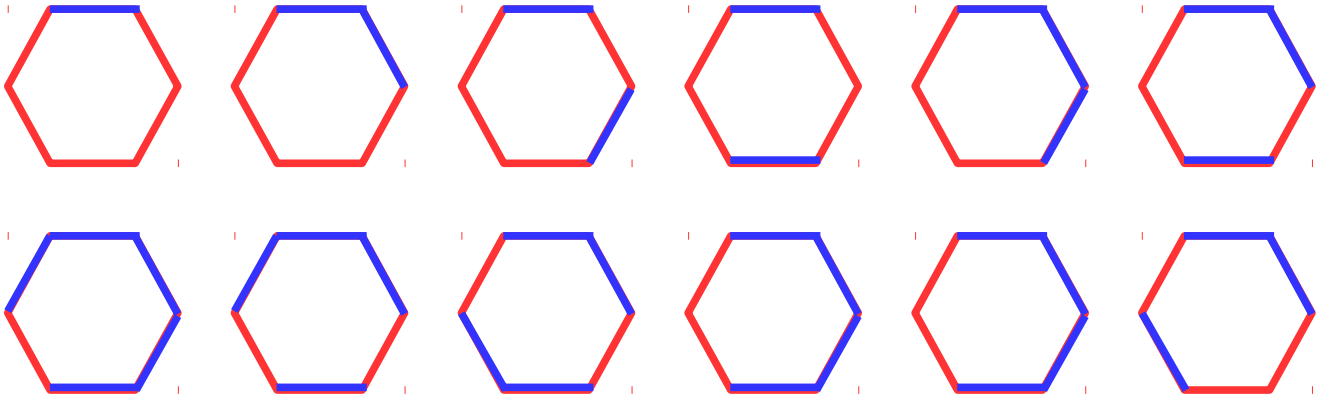
Par contre, si on prend l'exemple d'un hexagone, donc n=6, colorié de deux couleurs différentes, on obtient $(2^6-2)/6 = 31/3$.
Cette formule n'est donc pas valable pour les polygone avec un nombre de côtés qui n'est pas un nombre premier.

C)Cas des autres polygones

Pour les autres polygones, on utilise notre théorème, mais il va y avoir des répétitions au sein de l'enchaînement, ce qui signifie qu'on va obtenir plusieurs fois les mêmes combinaisons (étant donné qu'on peut tourner notre polygone). Nous avons donc voulu les recenser.

Par la suite, nous avons émis une conjecture qui va nous permettre de compter le nombre de répétitions en fonction du nombre de décalages.
Nous commençons par appliquer une partie de la formule obtenue précédemment : 2^n-2. Nous lui ajoutons ensuite le nombre de répétitions obtenu, puis divisons par le nombre de côtés n, ce qui va permettre d'éliminer les solutions qui se répètent lorsqu'on tourne la figure.

Nous allons prendre l'exemple d'un hexagone avec deux couleurs bleu et rouge.



On commence par appliquer la première partie de notre formule, soit $2^6-2 = 62$.

Nous repérons ensuite les différentes répétitions :

Nombre de décalages		
1	→ Pas de répétitions	
2	→ **BR / BR / BR** et inversement **(RB / RB / RB)**	+2 répétitions
3	→ **BRR / BRR** ; **BBR / BBR** ; **BRB / BRB** et leurs inverses	+6 répétitions
4	→ **BRBRBR** et inversement	+2 répétitions
5	→ Pas de répétitions	

On a ainsi un total de 10 répétitions, qu'on additionne à 62, puis on divise par 6 (nombre de côtés) : (62+10)/6 = 12, ce qui correspond bien au résultat que nous donnait l'algorithme.

Avec cette conjecture, on constate qu'il y a des répétitions quand le nombre de décalages n'est pas un nombre premier, car on peut diviser 6 (le nombre de côtés) par le nombre de décalages ou un de ses diviseurs. D'après le schéma précédent, on voit en effet que le nombre de côtés peut être découpé en parties égales (par exemple un hexagone peut être découpé en deux combinaisons de trois côtés), ce qui va créer des répétitions lorsque les différentes parties possèdent le même enchaînement de couleurs.
C'est pourquoi les polygones avec un nombre de côtés qui est un nombre

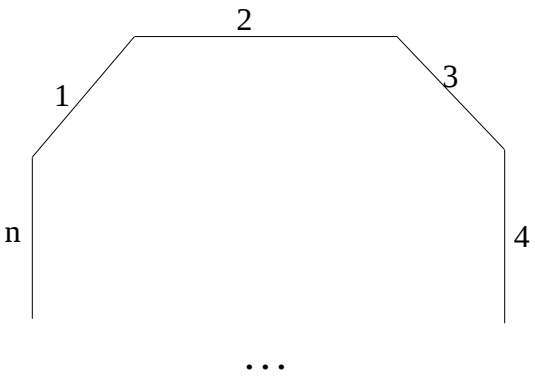
premier n'ont pas de répétitions. En effet, on ne peut donc pas découper le nombre de côtés en parties égales.
En conclusion, si n n'est pas premier alors toute configuration d'un polygone de n côtés et c couleurs aura nécessairement des répétitions (avec n un nombre non premier et c un nombre naturel). (5)

III – Théorème pour compter n côtés et n couleurs

Nous avons également trouvé un théorème lorsque nous avons le même nombre de côtés que de couleurs.

Théorème : Soit n le nombre de côtés,.
Le nombre de combinaisons pour n côtés et n couleurs est (n-1) !

Démonstration :
Nous avons un polygone à n côtés que nous devons colorier de n couleurs différentes. Nous commençons par numéroter les côtés de 1 à n comme sur la figure suivante :



...

Ensuite, on constate que pour le premier côté il y a n couleurs différentes possibles, pour le deuxième côté il y a n-1 couleurs possibles (puisqu'on ne peut pas mettre deux fois la même couleur) (6) et ainsi de suite jusqu'au n-ième côté pour lequel il ne reste plus qu'une seule possibilité.
On multiplie les nombres de possibilités de chaque côté entre eux, ce qui nous donne n ! possibilités.
Cependant, avec cette ce nombre de combinaisons obtenu on a encore les répétitions créées par les rotations de notre polygone. Nous divisons donc ensuite par le nombre de rotations possible, soit le nombre de côtés du polygone, ce qui nous donne la formule n !/n, qui vaut également (n-1) !. (7)
Cette formule marche dans tous les cas de polygones à n cotés coloriés avec n couleurs.

IV-Conclusion :

Nous avons donc réussi à trouver une manière de compter le nombre de

possibilités pour n cotés et n couleurs.
De même, nous sommes parvenues à compter tous les cas avec deux couleurs différentes, même si cela n'a pas été démontré.
On pourrait par la suite aborder le problème en trois dimensions ou tenter de trouver des formules pour un nombre plus important de couleurs.

Notes d'édition

(1) On s'assure ainsi que dans ce coloriage toutes les couleurs sont présentes. Si ce n'est pas le cas, le coloriage sera ignoré par la suite.

(2) Les auteurs utilisent dans la suite aussi le nombre de coloriage pour un « polygone » à deux côtés. Il est assez facile de voir qu'il n'y a qu'une seule façon de colorier les deux côtés avec une et avec deux couleurs.

(3) Cet exemple est mal placé, il montre simplement des colorations considérées comme identiques.

(4) Il faudrait expliquer ici, pourquoi en appliquant les rotations, nous n'obtenons pas de répétitions. Ce n'est vrai que si le nombre de cotés est premier. Une tentative d'explications est avancée à la fin du C).

(5) La méthode des répétitions exposée ici fonctionne. Il manque cependant une justification rigoureuse de celle-ci, qui n'est pas forcément aisée à formuler.

(6) Autrement, au moins une des couleurs ne serait pas présente dans le coloriage.

(7) Ici aussi, il faudrait justifier qu'aucune répétition n'apparait. C'est un exercice facile laissé à la sagacité du lecteur.

ANNEXE :

```
1   VARIABLES
2     L EST_DU_TYPE LISTE
3     l EST_DU_TYPE LISTE
4     l0 EST_DU_TYPE LISTE
5     l1 EST_DU_TYPE LISTE
6     n EST_DU_TYPE NOMBRE
7     N EST_DU_TYPE NOMBRE
8     c EST_DU_TYPE NOMBRE
9     C EST_DU_TYPE NOMBRE
10    k EST_DU_TYPE NOMBRE
11    k0 EST_DU_TYPE NOMBRE
12    b EST_DU_TYPE NOMBRE
13    i EST_DU_TYPE NOMBRE
14    i0 EST_DU_TYPE NOMBRE
15    i1 EST_DU_TYPE NOMBRE
16  DEBUT_ALGORITHME
17    N PREND_LA_VALEUR 0
18    //demander le nombre de cotés
19    AFFICHER "nombre de cotés : "
20    LIRE n
21    //demander le nombre de couleurs
22    AFFICHER "nombre de couleurs : "
23    LIRE c
24    //créer la première combinaison
25    POUR i ALLANT_DE 0 A n-1
26      DEBUT_POUR
27      l[i] PREND_LA_VALEUR 0
28      FIN_POUR
29    //generation+test de toutes les combinaisons
30    POUR i ALLANT_DE 1 A pow(c, n)
31      DEBUT_POUR
32      //on compte le nombre de couleurs dans le nombre
33      C PREND_LA_VALEUR 0
34      POUR i0 ALLANT_DE 0 A c-1
35        DEBUT_POUR
36        l1[i0] PREND_LA_VALEUR 0
37        FIN_POUR
38      POUR i0 ALLANT_DE 0 A n-1
39        DEBUT_POUR
40        SI (l1[l[i0]] == 0) ALORS
41          DEBUT_SI
42          l1[l[i0]] PREND_LA_VALEUR 1
43          C PREND_LA_VALEUR C+1
44          FIN_SI
45        FIN_POUR
46      //on tourne les chiffres afin d'obtenir le plus grand nombre possible
47      k PREND_LA_VALEUR -1
48      POUR i0 ALLANT_DE 0 A n-1
49        DEBUT_POUR
50        k0 PREND_LA_VALEUR 0
51        POUR i1 ALLANT_DE 0 A n-1
52          DEBUT_POUR
53          k0 PREND_LA_VALEUR k0+l[(i1+i0)%n]*pow(c, i1)
54          FIN_POUR
55        SI (k < k0) ALORS
56          DEBUT_SI
57          k PREND_LA_VALEUR k0
58          POUR i1 ALLANT_DE 0 A n-1
59            DEBUT_POUR
60            l0[i1] PREND_LA_VALEUR l[(i1+i0)%n]
61            FIN_POUR
62          FIN_SI
63        FIN_POUR
64      //On teste si le nombre obtenu existe dans ceux déjà générés
```

```
65      b PREND_LA_VALEUR 1
66      POUR i0 ALLANT_DE 0 A N-1
67        DEBUT_POUR
68        k0 PREND_LA_VALEUR 0
69        POUR i1 ALLANT_DE 0 A n-1
70          DEBUT_POUR
71          k0 PREND_LA_VALEUR k0 + L[i0*n+i1]*pow(c, i1)
72          FIN_POUR
73        //si les deux nombres sont égaux, b vaut 0
74        SI (k0 == k) ALORS
75          DEBUT_SI
76          b PREND_LA_VALEUR 0
77          FIN_SI
78        FIN_POUR
79      //si b vaut 1 et que le nombre de couleur est correct, alors on rajoute la
combinaison dans la liste
80      SI (b == 1 && C == c) ALORS
81        DEBUT_SI
82        POUR i0 ALLANT_DE 0 A n-1
83          DEBUT_POUR
84          L[N*n+i0] PREND_LA_VALEUR l0[i0]
85          AFFICHER l0[i0]
86          FIN_POUR
87        AFFICHER " "
88        N PREND_LA_VALEUR N+1
89        FIN_SI
90      //créer la combinaison suivante
91      l[0] PREND_LA_VALEUR l[0]+1
92      POUR i0 ALLANT_DE 0 A n-2
93        DEBUT_POUR
94        SI (l[i0] >= c) ALORS
95          DEBUT_SI
96          l[i0] PREND_LA_VALEUR 0
97          l[i0+1] PREND_LA_VALEUR l[i0+1]+1
98          FIN_SI
99        FIN_POUR
100     FIN_POUR
101   AFFICHER N
102   AFFICHER " combinaisons différentes"
103 FIN_ALGORITHME
```

Merci à Lionel Hemmerlé pour cet algorithme.