

Étude d'objets combinatoires - Applications à la bio-informatique

Rémi Vernay

▶ **To cite this version:**

Rémi Vernay. Étude d'objets combinatoires - Applications à la bio-informatique. Modélisation et simulation. Université de Bourgogne, 2011. Français. NNT : . tel-00618166

HAL Id: tel-00618166
https://theses.hal.science/tel-00618166

Submitted on 31 Aug 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Étude d'objets combinatoires
Applications à la bio-informatique

Thèse

(arrêté du 7 août 2006)

présentée et soutenue publiquement le

29 juin 2011

pour obtenir le grade de

Docteur de l'Université de Bourgogne

spécialité informatique

par

Rémi Vernay

Devant le jury ainsi composé :

• Pr Abderrafiâa Koukam	Université de Belfort-Montbéliard	*rapporteur*
• Pr Vlady Ravelomanana	Université Paris Diderot	*rapporteur*
• Pr Sarifuddin Madenda	Gunadarma University (Indonésie)	*examinateur*
• Pr Jean Pallo	Université de Bourgogne	*examinateur*
• Pr Vincent Vajnovszki	Université de Bourgogne	*directeur de thèse*
• Dr Jean-Luc Baril	Université de Bourgogne	*co-encadrant (HDR)*

Université de Bourgogne – UFR des Sciences et Techniques
École Doctorale Environnement Santé STIC – N°490
Laboratoire LE2I – UMR-CNRS 5158

À Joseph

Remerciements

Rédiger ces lignes n'est pas chose aisée, tant le nombre de personnes à remercier est grand, mais il est important de le faire : une thèse ne se réalise pas en restant seul, toutes les personnes qui sont parties prenantes pendant ces années méritent d'être remerciées ici. En revanche et pour pallier d'éventuels omissions, je n'ai volontairement inscrit aucun nom ou prénom. Je ne voudrais pas avoir à subir les foudres d'un proche oublié…

Je veux tout d'abord remercier mes encadrants. En effet, lorsque l'on cherche à débuter un doctorat, un conseil qui revient souvent est celui de s'attacher à bien choisir son directeur. Je confirme que c'est primordial : j'ai eu la chance de trouver des encadrants qui ont su être présents tout en me laissant une grande liberté de travail. Il convient également de remercier les rapporteurs et membres du jury qui ont accepté d'évaluer ce travail avec toute la rigueur nécessaire à cette tâche.

Je veux ensuite remercier ma petite sœur et tous les amis qui m'ont soutenu pendant ces années. On peut facilement se décourager, il devient alors important de ne pas se sentir seul. Être bien entouré permet de se redonner de la motivation et il faut en retrouver régulièrement pour ne pas craquer. Cela passe notamment par les moments de détente indispensables pour faire retomber une pression sans cesse croissante.

Enfin, je remercie toutes les personnes que j'ai pu rencontrer au cours de mon doctorat et qui m'ont permis d'avancer : les collègues enseignants de l'IUT informatique, les membres du laboratoire et bien sûr les collègues de bureau. Je n'oublie pas non plus toute l'équipe de l'Expérimentarium avec qui j'ai passé de grands moments de dur labeur et de détente.

Résumé

Cette thèse porte sur des classes d'objets combinatoires, qui modélisent des données en bio-informatique. Nous étudions notamment deux méthodes de mutation des gènes à l'intérieur du génome : la duplication et l'inversion.

Nous étudions d'une part le problème de la duplication-miroir complète avec perte aléatoire en termes de permutations à motifs exclus. Nous démontrons que la classe de permutations obtenue avec cette méthode après p duplications à partir de l'identité est la classe de permutations qui évite les permutations alternées de longueur $2^p + 1$. Nous énumérons également le nombre de duplications nécessaires et suffisantes pour obtenir une permutation quelconque de longueur n à partir de l'identité. Nous proposons également deux algorithmes efficaces permettant de reconstituer deux chemins différents entre l'identité et une permutation déterminée. Nous donnons enfin des résultats connexes sur d'autres classes proches.

La restriction de la relation d'ordre $\prec$ induite par le code de Gray réfléchi à l'ensemble des compositions et des compositions bornées induit de nouveaux codes de Gray pour ces ensembles. La relation d'ordre $\prec$ restreinte à l'ensemble des compositions bornées d'un intervalle fournit encore un code de Gray. L'ensemble des n-compositions bornées d'un intervalle généralise simultanément l'ensemble produit et l'ensemble des compositions d'un entier et donc la relation $\prec$ définit de façon unifiée tous ces codes de Gray. Nous réexprimons les codes de Gray de Walsh et Knuth pour les compositions (bornées) d'un entier à l'aide d'une unique relation d'ordre. Alors, le code de Gray de Walsh pour des classes de compositions et de permutations devient une sous-liste de celui de Knuth, lequel est à son tour une sous-liste du code de Gray réfléchi.

Mot clés : combinatoire, permutations, compositions d'entiers, codes de Gray, duplication, inversion, bio-informatique.

Abstract

This thesis considers classes of combinatorial objects that model data in bioinformatics. We have studied two methods of mutation of genes within the genome: duplication and inversion.

At first, we study the problem of the whole mirror duplication-random loss model in terms of pattern avoiding permutations. We prove that the class of permutations obtained with this method after p duplications from the identity is the class of permutations avoiding alternating permutations of length 2^p+1. We also enumerate the number of duplications that are necessary and sufficient to obtain any permutation of length n from the identity. We also suggest two efficient algorithms to reconstruct two different paths between the identity and a specified permutation. Finally, we give related results on other classes nearby.

The restriction of the order relation $\prec$ induced by the reflected Gray code for the sets of compositions and bounded compositions gives new Gray codes for these sets. The order relation $\prec$ restricted to the set of bounded compositions of an interval also yields a Gray code. The set of bounded n-compositions of an interval simultaneously generalizes product set and compositions of an integer, and so $\prec$ puts under a single roof all these Gray codes. We re-express Walsh's and Knuth's Gray codes for (bounded) compositions of an integer in terms of a unique order relation, and so Walsh's Gray code becomes a sublist of Knuth's code, which in turn is a sublist of the Reflected Gray Code.

Keywords: combinatorics, permutations, compositions of integers, Gray codes, duplications, inversions, bioinformatics.

Table des matières

Remerciements **7**

Résumé **9**

Abstract **11**

Introduction **15**

I. État de l'art **19**

1. Introduction à la combinatoire **21**

1.1. Trois notions fondamentales 21

1.2. Méthodes de génération et efficacité 23

2. Codes de Gray **29**

2.1. Historique 29

2.2. Forme 31

2.3. Applications 34

2.4. Codes de Gray et compositions d'entiers 36

3. Permutations **39**

3.1. Définitions 39

II. Notre étude **47**

4. Duplications complètes avec perte aléatoire **49**

4.1. Fondement biologique 49

4.2. Introduction 52
4.3. Notre contribution 55
4.4. Algorithmes 61
4.5. Autres modèles 70

5. Codes de Gray pour des classes de compositions et de permutations 77

5.1. Introduction 77
5.2. Compositions bornées d'un intervalle 80
5.3. Permutations restreintes 90
5.4. Algorithme 96

Conclusion 99

Annexe 101

Article 1 103
Article 2 111

Liste des tables 117

Liste des figures 119

Bibliographie 121

Introduction

La présente thèse est située entre l'*algorithmique* et la *combinatoire*, deux domaines de l'informatique aujourd'hui utilisés notamment en bio-informatique. Aussi nous étudions des classes d'objets combinatoires, qui modélisent des données en bio-informatique. Ce manuscrit se décompose en deux parties : un état de l'art (partie I page 20) et l'étude proprement dite (partie II page 48).

Dans l'état de l'art, nous rédigeons tout d'abord une introduction générale à la combinatoire (1 page 21). Cette introduction n'est bien naturellement pas complète mais permet de définir les trois notions fondamentales (1.1 page 21) que sont l'énumération, le listage et la génération exhaustive. Nous détaillons ensuite (1.2 page 23) quelques méthodes de génération et leur efficacité.

Nous donnons ensuite un condensé de l'histoire (2.1 page 29) des codes de Gray (2 page 29). Ledit code est décrit à partir des définitions de ses différentes formes (2.2 page 31) et des applications possibles de celui-ci (2.3 page 34). Un lien avec les compositions d'entier (2.4 page 36) permet enfin de préparer la suite de l'étude.

Un chapitre est consacré aux permutations (3 page 39), notamment pour donner plusieurs définitions utilisées dans la suite du manuscrit : permutation, miroir, classe, motif, permutation alternée, différentes statistiques, etc. Nous présentons également les différentes représentations des permutations couramment utilisées dans la littérature. En fin de chapitre, nous explicitons la stabilité des classes de permutations et la conséquence de celle-ci dans le cadre des classes de permutations à motifs exclus.

La partie consacrée à l'étude elle-même (partie II page 48) est à son tour séparée en deux chapitres : le premier concerne les duplications complètes avec perte aléatoire (4 page 49), le second présente différents codes de Gray pour des classes

restreintes de compositions et de permutations (5 page 77). Ces deux chapitres sont centrés sur deux méthodes de mutation des gènes modélisables en informatique : la duplication et l'inversion.

Nous abordons le problème de la duplication miroir complète avec perte aléatoire (WM-duplication) en termes de permutations à motifs exclus (définition 3.15 page 45). Afin de situer le contexte, nous rédigeons un bref *memorandum* sur le fondement biologique de cette étude en 4.1 (page 49).

Dans la section 4.2 (page 52) sont rappelés les résultats existants sur le sujet dans la littérature, présentés entre autres par Bouvel, Chauduri, Mansour, Rossin et Pergola.

La section 4.3 (page 55) présente le modèle de duplication puis il est démontré que la classe de permutations obtenue avec cette méthode après p WM-duplications à partir de l'identité est la classe de permutations qui évite les permutations alternées de longueur 2^p+1. Cette classe est également la classe des permutations avec au plus $2^{p-1}-1$ vallées. On trouve également l'énumération du nombre de WM-duplications nécessaires et suffisantes pour obtenir une permutation quelconque de longueur n à partir de l'identité.

Dans la section 4.4 (page 61) sont proposés deux algorithmes efficaces permettant de reconstituer deux chemins différents entre l'identité et la permutation recherchée en utilisant exclusivement des WM-duplications. Des résultats connexes sur d'autres modèles de duplication (4.5 page 70) sont également présentés en fin de chapitre.

La restriction de la relation d'ordre $\prec$ induite par le code de Gray réfléchi à l'ensemble des compositions et des compositions bornées induit de nouveaux codes de Gray pour ces ensembles. On voit que la relation d'ordre $\prec$ restreinte à l'ensemble des compositions bornées d'un intervalle fournit encore un code de Gray (5.2 page 80).

Nous montrons que l'ensemble des n-compositions bornées d'un intervalle généralise simultanément l'ensemble produit et l'ensemble des compositions d'un entier et donc la relation $\prec$ définit de façon unifiée tous ces codes de Gray. Les codes de Gray de Walsh et Knuth pour les compositions (bornées) d'un entier sont réexprimés à l'aide d'une unique relation d'ordre. Alors, le code de Gray de Walsh devient une sous-liste de celui de Knuth, lequel est à son tour une sous-liste du code de Gray réfléchi. À partir de cette relation d'ordre les codes de Gray de Knuth et Walsh sont

généralisés aux compositions bornées d'un intervalle $[k, \ell]$.

Ces résultats sont utilisés pour obtenir des codes de Gray pour les permutations avec un nombre d'inversions situé entre deux entiers ou avec un nombre pair (ou impair) d'inversions ou de cycles (5.3 page 90).

L'étude n'a pas permis de trouver un algorithme efficace offrant la possibilité de générer la liste des permutations avec un nombre borné d'inversions ou son pendant, à savoir les compositions $\boldsymbol{b}$-bornées d'un intervalle $[k, \ell]$. En revanche, un algorithme de rejet (5.4 page 96) en combinant les méthodes « successeurs » de Williamson et de Walsh fonctionne et permet d'obtenir dans certains cas la liste recherchée de manière efficace.

Nous terminons le manuscrit (page 99) par un aperçu des perspectives de travail futures basées sur les travaux présentés dans cette thèse.

Première partie

État de l’art

Chapitre 1

Introduction à la combinatoire

En informatique, il peut être opportun d'examiner un nombre fini voire dénombrable de possibilités afin de résoudre des conjectures ou de détecter des propriétés. En effet, de nombreux problèmes débutent par l'énumération de toutes les possibilités pouvant surgir sous certaines conditions. Ces possibilités ont généralement une structure combinatoire fondamentale qui peut être explorée afin d'obtenir un algorithme efficace pour générer quelques éléments (génération *aléatoire*) voire tous les éléments (génération *exhaustive*) ou encore un échantillonnage.

En général, la combinatoire permet d'étudier des objets discrets qui peuvent être décrits par un nombre fini de règles. Traditionnellement, les objets combinatoires sont munis d'une signification intuitive issue de problématiques concrètes. L'un des centres d'intérêt important en combinatoire est constitué de la recherche et de l'analyse des régularités dans les mots [68].

Avant de poursuivre, nous allons définir trois notions fondamentales en combinatoire (1.1).

1.1 Trois notions fondamentales

Nous allons adopter les définitions données par Donald Knuth dans l'un de ses ouvrages [64] « The art of computer programming ». Dans ce livre, il définit la *génération exhaustive* (définition 1.3 page 22) en la confrontant d'abord aux notions

d'*énumération* (définition 1.1) et de *listage* (définition 1.2).

Définition 1.1 : énumération

✎ Selon Knuth, l'*énumération* consiste à trouver le nombre d'éléments contenus dans une classe d'objets combinatoires. □

Par exemple, si vous souhaitez énumérer l'ensemble des permutations de taille 4, vous attendrez « 24 » comme réponse et non pas la liste détaillée $\{1234, 1432, \ldots\}$.

L'énumération est l'un des principaux objectifs de la combinatoire. Lorsque l'on manipule des classes d'objets, on cherche presque toujours à compter leurs éléments. Il est d'ailleurs généralement plus rapide d'énumérer plutôt que de lister les éléments, des fonctions mathématiques permettant souvent de trouver le résultat simplement en appliquant une formule à partir des paramètres de la classe ; c'est dans ce domaine qu'intervient pleinement la théorie des *fonctions génératrices*.

Définition 1.2 : listage

✎ Selon Knuth, il s'agit de *lister* tous les objets d'une classe combinatoire sous forme exhaustive, au sens où l'on devrait avoir au même moment sous les yeux la totalité des objets que l'on traite. □

Dans le cas des permutations de taille 10, on aurait par exemple besoin de voir les 3 628 800 permutations correspondantes. Cet exemple, qui utilise une taille d'objets faible (10), montre que cette définition est inutilisable en pratique, eu égard à la quantité d'éléments à manipuler dans une classe d'objets et de la taille de ceux-ci.

Bien que le but des travaux dans le passé en combinatoire était principalement l'énumération afin de déterminer le nombre des objets ayant une taille fixe dans une classe combinatoire, il est devenu clair qu'à l'aide d'un ordinateur il était désormais possible non seulement d'énumérer, mais également de lister, voire de générer des objets dans des classes combinatoires.

Définition 1.3 : génération exhaustive

✎ Selon Knuth, la *génération exhaustive* consiste à parcourir les éléments les uns après les autres, sans nécessairement avoir besoin de les avoir tous en même temps sous les yeux. □

On peut dire que la génération exhaustive revient à parcourir *tous* les éléments les uns après les autres ; on les veut *tous*, mais on les veut *un par un*.

En fait, les algorithmes de génération exhaustive permettent également d'énumérer ; il suffit de compter et éventuellement de sauvegarder chaque objet quand il est généré, voir par exemple [62, 79, 92].

De nombreuses questions pratiques exigent pour leur solution l'échantillonnage aléatoire d'une classe combinatoire ou une recherche exhaustive de tous les objets ayant une certaine propriété dans la classe. Par conséquent, il y a plusieurs raisons pour lesquelles il est intéressant de développer des algorithmes de génération des objets combinatoires. En particulier, il est utile d'avoir des algorithmes combinatoires qui peuvent lister, dans un ordre particulier, les objets d'une classe combinatoire, sans répétitions et sans omissions, que l'on appelle algorithmes de génération exhaustive (voir par exemple [4, 64, 79]). La génération exhaustive d'objets combinatoires peut actuellement être appliquée à de nombreuses disciplines, comme la biologie, la chimie, la médecine, la bio-informatique, etc.

1.2 Méthodes de génération et efficacité

Différentes méthodes algorithmiques de génération existent : les algorithmes récursifs (définition 1.4), la méthode ECO (définition 1.5 page 24), les algorithmes itératifs, dans un ordre particulier, etc. Ces méthodes peuvent être de différentes efficacités comme CAT (définition 1.6 page 25) et *loopless* (définition 1.7 page 25).

Définition 1.4 : algorithmes récursifs

✎ Un *algorithme récursif* utilise les listes d'objets de tailles inférieures pour générer les éléments de taille n. □

Par exemple, la liste $\mathscr{B}_n$ des mots binaires de taille n est définie récursivement par (λ est la liste vide) :

$$\mathscr{B}_n = \begin{cases} \lambda & \text{si } n = 0 \\ 0 \cdot \mathscr{B}_{n-1} \bigcup 1 \cdot \mathscr{B}_{n-1} & \text{si } n \geqslant 1 \end{cases}$$

Si les algorithmes récursifs ont souvent une mauvaise image, c'est que leurs coûts en temps processeur et en quantité de mémoire sont souvent prohibitifs. Il est pourtant parfois possible de trouver des algorithmes récursifs performants. C'est le cas de l'un des algorithmes de génération du code de Gray binaire réfléchi (CGBR) dont on

parlera dans le chapitre 2 (page 29). Cet algorithme (algorithme 1.1) ne modifie pas plus d'une valeur lors du passage d'un élément à l'autre.

```
void gen_rec(int n, int dir) {
   if (n==0)
      print(word);
   else {
      if (dir==0) {
         word[n]=0; gen_rec(n-1,0);
         word[n]=1; gen_rec(n-1,1);
      }
      else { // dir==1
         word[n]=1; gen_rec(n-1,0);
         word[n]=0; gen_rec(n-1,1);
      }
   }
}
```

Algorithme 1.1 – Génération du CGBR en C (méthode récursive CAT).

Plus récemment, on a donné une nouvelle méthode de génération, la méthode ECO (*Enumerating Combinatorial Objects*) [6] permettant d'énumérer ou de générer des classes d'objets combinatoires (mots de Dyck, permutations, etc. [4, 95]).

Définition 1.5 : méthode ECO

✎ La *méthode ECO* est une description récursive d'une classe d'objets combinatoires qui explique comment un objet de taille n peut être atteint à partir d'un seul et unique objet de taille inférieure. □

La technique d'énumération grâce aux arbres de génération a été introduite par West [102], puis reprise, généralisée et formalisée sous le nom de méthode ECO par Pinzani *et al.* [6]. Appliquée à une classe $\mathcal{C}$ d'objets, cette méthode décrit des règles de successions permettant de construire les objets de taille n à partir de ceux de tailles inférieures. Des étiquettes sont associées aux objets, de sorte à pouvoir coder les règles de succession.

La méthode ECO est un moyen très puissant pour définir et décrire des classes de permutations à motifs exclus (définition 3.15 page 45) en utilisant des règles de succession et des arbres de génération [5, 39, 54].

Pour certaines listes d'objets, il existe des algorithmes de génération possédant

une complexité constante, *i.e.*, indépendante de la taille des objets générés. Cette complexité peut être constante dans le « pire des cas » ou « en moyenne ». Lorsque l'on est dans le cas « en moyenne », le nombre global d'opérations effectuées pour générer la liste entière divisé par le nombre d'objets de la liste est indépendant de la taille des objets générés. Le but est que le nombre d'opérations soit proportionnel au nombre d'éléments de la classe générée.

Définition 1.6 : CAT

✎ Un algorithme de génération exhaustive pour des objets combinatoires qui demande seulement un temps moyen constant (c'est-à-dire indépendant de la taille des objets) par objet est appelé *algorithme CAT* (notion introduite par Ruskey), pour *Constant Amortized Time*. □

Une analyse simple montre que l'algorithme 1.1 (page 24) est un algorithme CAT. Comme on vient de le dire, cet algorithme CAT est performant bien que cette méthode soit récursive.

Si l'on cherche à optimiser le passage direct d'un élément au suivant, en utilisant une méthode permettant d'obtenir le « successeur » d'un objet, on obtient la définition 1.7 :

Définition 1.7 : *loopless*

✎ Un algorithme qui comprend une fonction permettant d'obtenir le successeur d'un objet d'une liste sans utiliser de boucle est dit *loopless* (ou *loopfree*). □

Les algorithmes sans boucle (appelés aussi *loopless*) sont particulièrement intéressants car le nombre d'opérations nécessaires pour générer l'élément suivant de la liste est borné par une valeur indépendante de la taille des objets de la classe. On peut même atteindre la notion de code de Gray, que l'on définira dans le chapitre 2 (page 29), si deux éléments consécutifs diffèrent en un nombre minimal de positions [52, 75, 103].

On peut trouver un algorithme *loopless* (algorithme 1.2 (page 26)) qui permet d'obtenir le code de Gray binaire réfléchi (CGBR) dans [18]. Cet algorithme permet d'obtenir la même liste que l'algorithme 1.1 (page 24).

```
void loopless() {
   print(word);
   i=t[0];
   word[i]=1-word[i];
   t[0]=1;
   t[i-1]=t[i];
   t[i]=i+1;
}
//----------------------

main() {
   for (j=0;j<=n;++j) {
      word[j]=0;
      t[j]=j+1;
   }

   i=0;
   while (i<=n)
      loopless();
}
```

Algorithme 1.2 – Génération du CGBR en C (méthode *loopless*.)

On observe depuis quelques temps une explosion des résultats sur la génération combinatoire pour diverses raisons : la croissance de la vitesse des ordinateurs, l'intérêt mathématique pour l'Hamiltonisme des graphes, l'émergence du calcul parallèle, les grandes bases de données, la bio-informatique, etc.

On trouve par exemple des algorithmes de génération pour certaines classes d'objets combinatoires : mots de Dyck, de Motzkin, de Schröder... Ces algorithmes ont la possibilité d'être basés sur les constructions récursives de listes [17, 93] ou la méthode ECO [38].

Cependant, afin qu'il soit possible de générer des objets, même de taille modérée, les méthodes de génération combinatoire doivent être extrêmement efficaces. Une approche habituelle a été d'essayer de générer des objets comme une liste dont les éléments successifs diffèrent légèrement. Un exemple classique est le bien connu code Gray [83], défini dans la chapitre 2 (page 29).

Le problème qui consiste à engendrer une classe d'objets combinatoires de telle sorte que deux objets successifs diffèrent d'une manière prédéfinie peut être formulé en terme de chemin Hamiltonien dans un graphe, ce qui est un problème NP-

complet.

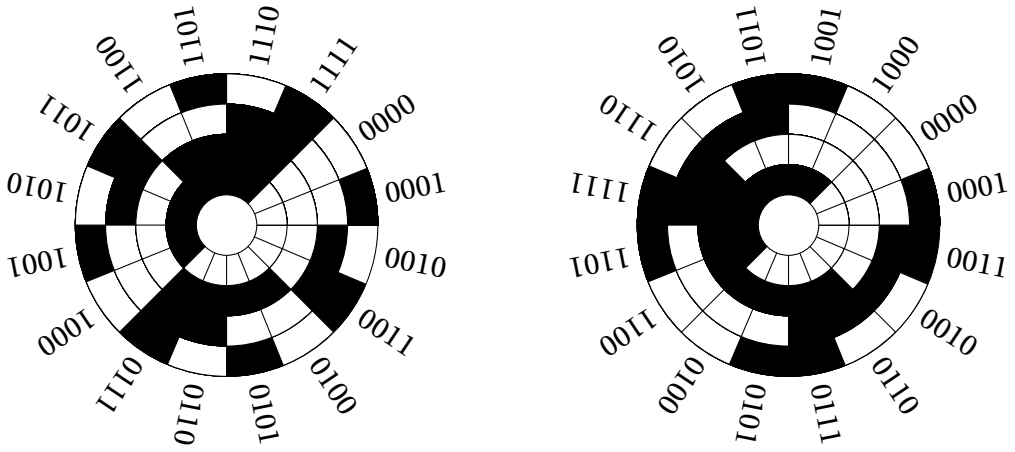
Chapitre 2

Codes de Gray

Dans cette partie nous allons aborder les codes de Gray. Les travaux de cette thèse sont fortement basés sur ces types de codes, c'est pourquoi nous allons les détailler ici. Nous évoquerons tout d'abord l'histoire (2.1) de ceux-ci. Ensuite dans la section 2.2 (page 31), nous parlerons de la forme de ces codes. Dans la section 2.3 (page 34) nous parlerons des applications possibles des codes de Gray et en 2.4 (page 36) nous nous intéresserons aux compositions d'entiers.

2.1 Historique

La première évocation d'un code correspondant à la définition actuelle d'un code de Gray date de 1872, lorsque le français Louis Agathon Gros rédigea un document [53] dans lequel il faisait le lien explicite entre ledit code et la résolution d'un casse-tête appelé le « baguenaudier ». On résout en effet ce jeu en appliquant une séquence sous forme de code de Gray ; c'est pourquoi on nomme parfois celui-ci : « code de Gros-Gray ». La première utilisation d'un tel code pour des applications industrielles date de 1878 où Émile Baudot [56] s'en servi dans l'un de ses modèles de télégraphe. On retrouve également l'idée du code binaire dans un brevet de George Stibitz [88] mais ce n'est qu'en 1953 que le brevet de Franck Gray [52], déposé en 1947, formalisera avec force de schémas électriques et représentations schématiques ce que deviendra le code de Gray tel qu'utilisé aujourd'hui.



(a) code lexicographique (b) code de Gray

Figure 2.1 – Représentations schématiques de roues codeuses : code lexicographique et code de Gray de taille 4.

La figure 2.1 expose deux représentations schématiques de roues codeuses. En (a) on trouve une roue utilisant le code lexicographique : on peut voir que le passage d'un élément à l'autre peut se faire avec 1, 2, 3 voire 4 modifications. La roue présentée en (b) utilise un code de Gray : le passage d'un élément à l'autre se fait avec une seule modification. Ce type de roue codeuse peut être utilisé dans les appareils mécaniques utilisant un moteur pas-à-pas.

Une représentation similaire peut se trouver également en électronique, en utilisant un chronogramme (figure 2.2).

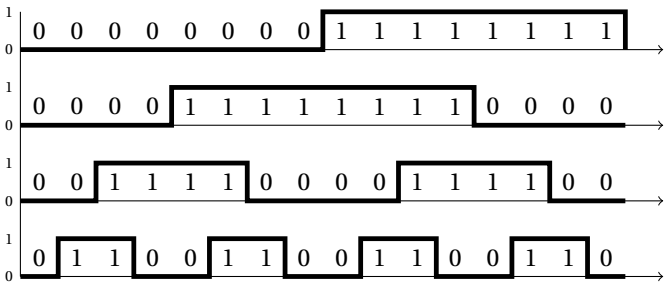


Figure 2.2 – Chronogramme : code de Gray de taille 4.

Ce code de Gray peut également se représenter de manière plus mathématique en théorie des graphes dans un hypercube. Dans la figure 2.3 (page 31), le CGBR est représenté par le chemin représenté en gras : on passe sur chaque élément une fois et une seule, avec une unique modification entre chaque élément. Ce cheminement forme un *chemin Hamiltonien*. On peut également suivre un autre chemin en utili-

sant les arêtes en pointillés ; les éléments sont les mêmes mais parcourus dans un ordre différent.

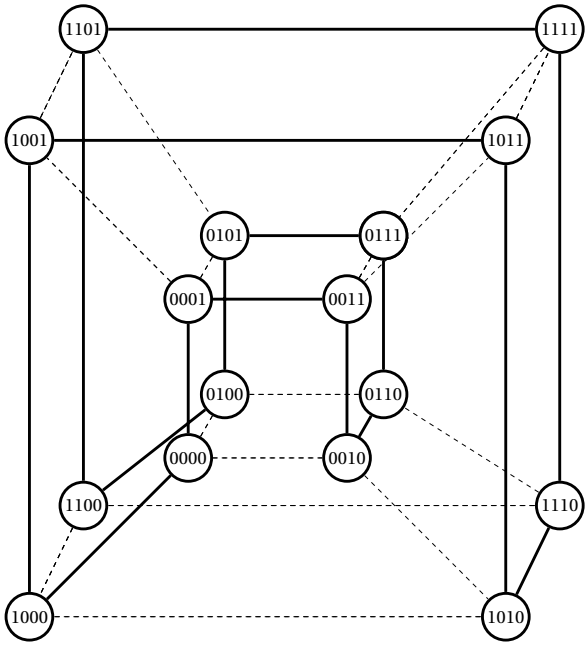


Figure 2.3 – Hypercube Q_4 : code de Gray de taille 4.

2.2 Forme

Un code de Gray se définit à partir de plusieurs caractéristiques. Avant de les détailler, rappelons d'abord la définition de la *distance de Hamming* dans le cas général (définition 2.1) et dans le cas particulier des mots binaires (définition 2.2 page 32).

Définition 2.1 : distance de Hamming

✎ Soit F un alphabet et D_n l'ensemble des mots de longueur n dans F. La *distance de Hamming* $d(\boldsymbol{a},\boldsymbol{b})$ entre deux éléments $\boldsymbol{a} = a_1 a_2 \dots a_n$ et $\boldsymbol{b} = b_1 b_2 \dots b_n$ de D_n est :
$d(\boldsymbol{a},\boldsymbol{b}) = \text{card}\,(\{i \ : \ a_i \neq b_i\})$ □

Dans le cas particulier des mots binaires, on peut simplifier la définition 2.1 pour

obtenir la définition 2.2 :

Définition 2.2 : distance de Hamming entre deux mots binaires

✎ Dans le cas de la *distance de Hamming entre deux mots binaires*, on a :
$\forall \boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^n, \ d(\boldsymbol{a}, \boldsymbol{b}) = \sum_{i=1}^{n} (a_i \oplus b_i)$
avec $\oplus$ correspondant au « ou exclusif ». □

Par exemple, les mots binaires 01011 et 01001 diffèrent en une seule position, la distance de Hamming entre ces deux mots est donc égale à 1.

On choisit d'utiliser la définition générale d'un code de Gray donnée dans [101] par Walsh (définition 2.3) :

Définition 2.3 : code de Gray

✎ Soit $\mathscr{G}_n$ une liste ordonnée d'un ensemble G_n de mots de longueur n. Si chaque paire d'éléments consécutifs de $\mathscr{G}_n$ a une distance de Hamming bornée par une constante indépendante de n, cette famille est appelée *code de Gray*. □

La distance de Hamming de deux mots consécutifs dans le code de Gray binaire réfléchi [18, 49, 52, 101] (ou CGBR) est de 1.

Il existe plusieurs méthodes pour construire un code de Gray. On peut notamment donner une définition récursive en terme de sous-listes (2.2.1), en donnant une relation d'ordre explicite (2.2.2 page 33) ou encore en utilisant la méthode ECO.

2.2.1 Définition en terme de sous-listes

Lorsque l'on définit un code de Gray en terme de sous-listes, la liste de taille n est formée d'une concaténation de sous-listes de taille inférieure (table 2.1 page 33 par exemple). $\overline{\mathscr{G}_{n-1}}$ représente la liste $\mathscr{G}_{n-1}$ renversée – *i.e.* lue de la fin vers le début – et λ est le mot vide.

$$\mathscr{G}_n = \begin{cases} \lambda & \text{si } n = 0 \\ 0 \cdot \mathscr{G}_{n-1} \bigcup 1 \cdot \overline{\mathscr{G}_{n-1}} & \text{si } n \geqslant 1 \end{cases}$$

$$\mathscr{G}_3 = \begin{matrix} 000 \\ 001 \\ 011 \\ 010 \\ 110 \\ 111 \\ 101 \\ 100 \end{matrix}$$

Table 2.1 – Code de Gray binaire réfléchi de longueur 3 : $\mathscr{G}_3$.

À noter que dans le cas du CGBR, le code est *cyclique*, c'est-à-dire que le dernier mot de la liste et le premier de celle-ci ont également une distance de Hamming de 1. Si on reprend l'hypercube de la figure 2.3 (page 31), cela se traduit par le fait que l'on revient au point de départ lorsque qu'on termine le cheminement entre les différents sommets (en suivant un cycle Hamiltonien).

D'autres méthodes ont également été utilisées pour obtenir des codes de Gray pour les principales classes de permutations à motifs exclus [8, 9, 10, 11, 12, 39]. Il peut aussi parfois être intéressant de prendre la restriction d'un code de Gray déjà existant pour obtenir des codes de Gray de classes plus petites. Enfin, on peut aussi définir explicitement la relation d'ordre sur la classe (2.2.2).

2.2.2 Définition fondée sur une relation d'ordre

Un code de Gray pour un ensemble d'objets combinatoires définit une relation d'ordre total sur cet ensemble : $\boldsymbol{x} \prec \boldsymbol{y}$ si et seulement si $\boldsymbol{y}$ se situe après $\boldsymbol{x}$ dans le code.

Dans le cas du CGBR, on a $\boldsymbol{x} \prec \boldsymbol{y}$ si

$$\sum_{j=1}^{i-1} x_j \text{ est pair et } x_i < y_i$$

ou bien si

$$\sum_{j=1}^{i-1} x_j \text{ est impair et } x_i > y_i$$

où i est la position la plus à gauche telle que $x_i \neq y_i$.

Toujours dans le cas du CGBR, il est intéressant de remarquer que la définition en terme de sous-listes (2.2.1 page 32) et celle fondée sur cette relation d'ordre (2.2.2 page 32) sont parfaitement équivalentes.

On pourra enfin remarquer qu'il existe des définitions encore plus restrictives de codes de Gray dans la littérature en ajoutant des contraintes à celles déjà nécessaires pour former un code de Gray. Par exemple, Knuth [63, page 10] donne un code de Gray pour les combinaisons d'entiers (2.4 page 36) sous forme de mots binaires où deux mots consécutifs ne diffèrent qu'en deux positions et où tous les éléments entre ces deux positions ne sont que des zéros (table 2.2).

0	0	0	1	1	1
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	0
0	1	0	1	1	0
0	1	0	1	0	1
0	1	0	0	1	1
0	1	1	**0**	0	1
0	1	1	0	1	0
0	1	1	1	0	0
1	0	1	1	0	0
1	0	1	0	**0**	1
1	0	1	0	1	0
1	0	0	1	1	0
1	0	0	1	0	1
1	0	0	0	1	1
1	1	**0**	**0**	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0

Table 2.2 – Un exemple de code de Gray plus restrictif proposé par Knuth.

Dans la table 2.2 on peut observer de manière simple qu'entre deux positions qui changent (éléments soulignés) on n'a que des zéros (éléments en gras).

2.3 Applications

Par application du code de Gray binaire [52], beaucoup de problèmes ont été résolus et d'autres améliorés [29, 30, 43, 47, 50, 52, 67]. Si une application de-

mande le parcours exhaustif d'une classe d'objets, alors les codes de Gray peuvent accélérer la tâche. En effet, puisque dans un code de Gray, deux éléments consécutifs diffèrent sur peu de positions, cette structure a des chances de pouvoir être générée efficacement, par exemple par un algorithme CAT (définition 1.6 page 25) ou *loopless* (définition 1.7 page 25).

Les premières applications des codes de Gray ont été trouvées en mécanique et en électronique. On s'en sert aujourd'hui largement en informatique. Dans les réseaux, il est utilisé notamment pour la détection et la correction d'erreurs, la distance de Hamming étant un excellent support. En bio-informatique, on utilise notamment la grande rapidité de génération des codes de Gray pour gérer les nombres de taille colossale employés dans ce domaine. On trouve également les codes de Gray dans le domaine de la cryptographie et dans le traitement d'images (compression, segmentation, etc.).

Par exemple, il existe des codes de Gray pour l'ensemble des permutations sur $\{1,2,\ldots,n\}$ (voir chapitre 3 (page 39)) telles que deux permutations consécutives ne diffèrent que par l'échange d'une paire d'éléments adjacents [57, 91].

L'algorithme dit de Johnson-Trotter est mis en place avec la contrainte de n'avoir qu'un seul échange (donc deux modifications) entre chaque permutation générée. Pour satisfaire cette exigence, il doit y avoir une donnée supplémentaire dans l'algorithme : la direction associée à chaque élément. Cette direction est soit à gauche soit à droite et est initialisée à gauche au début pour toutes les valeurs.

- On initialise la direction de chaque élément à « gauche » ;
- tant qu'il existe une valeur mobile (*i.e.* que l'entier juste à gauche ou juste à droite en fonction de la direction est plus petit et qu'on ne se trouve ni sur le bord gauche, ni sur le bord droit), faire :
  - trouver la plus grande valeur mobile k ;
  - échanger k avec l'entier adjacent (en fonction de la direction associés à k) ;
  - pour tout entier $l > k$, faire :
    - inverser la direction associée à la valeur l.

La figure 2.4 (page 36) montre un exemple de génération des permutations de taille 4.

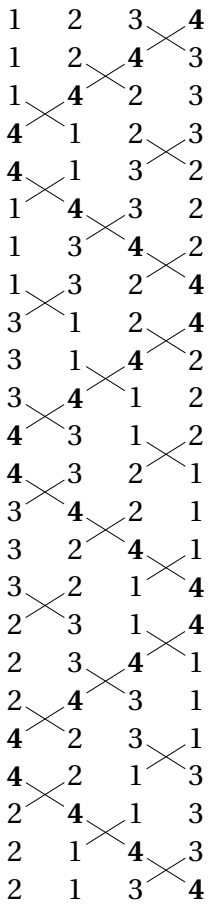


Figure 2.4 – Génération des permutations de taille 4 grâce à l'algorithme de Johson-Trotter.

On trouve également des codes de Gray pour les sous-ensembles de k-éléments d'un ensemble de n-éléments de telle façon que deux objets consécutifs diffèrent par exactement un élément [31, 80] (pour d'autres exemples, voir [69, 94]).

Les codes de Gray ont eu des applications dans divers domaines comme : le test de circuit [35], l'encodage de signaux [70], l'ordonnancement des documents, la compression de données [77], l'allocation de processeurs, les puzzles, le jeu du baguenaudier [53], le jeu des tours de Hanoï [47], etc.

2.4 Codes de Gray et compositions d'entiers

Dans le chapitre 5 (page 77) nous donnerons des codes de Gray pour différentes classes de compositions d'entiers. Nous allons donc définir quelques notions

avant de poursuivre.

Définition 2.4 : tuple et n-uplet d'entiers positifs ou nuls

✎ Un tuple de longueur n ou *n-uplet* $\boldsymbol{w}$ est : $w_1 w_2 \dots w_n$ où w_i est un entier positif ou nul pour tout $i \in \{1, 2, \dots, n\}$. □

Pour un entier n donné, $\mathbb{N}^n$ représente l'ensemble de tous les n-uplets.

Définition 2.5 : composition d'entier

✎ Une *n-composition d'un entier k* est un n-uplet $\boldsymbol{c}$ avec $c_1 + c_2 + \dots + c_n = k$. □

Knuth a donné[1] une définition d'un code de Gray pour l'ensemble des compositions de longueur n d'un entier k. Ce code de Gray a été défini récursivement par Nijenhuis et Wilf [75].

Pour $n = 2$, la liste est ainsi construite :

$$\begin{array}{cc} k & 0 \\ k-1 & 1 \\ k-2 & 2 \\ \dots & \dots \\ 2 & k-2 \\ 1 & k-1 \\ 0 & k \end{array}$$

Et pour $n > 2$, on a le cas général suivant :

$$\mathscr{L}(n,k) = \bigcup_{j=0}^{k} (-1)^j \left[\mathscr{L}\left(n-1, k-j\right) \cdot \{j\} \right]$$

Où $(-1)^j$ indique le sens de lecture de la liste $\mathscr{L}$: lorsque j est pair on lit la liste du début vers la fin, lorsque j est impair on la lit de la fin vers le début. La table 2.3 (page 38) montre un exemple de construction de ce code de Gray pour $\mathscr{L}(3,4)$.

Cette définition a ensuite été implémentée itérativement par Klingsberg [61].

[1] Réponse non publiée à une question de Nijenhuis et Wilf.

$\mathscr{L}(2,4)\cdot\{0\}$	4	0	**0**
	3	1	**0**
	2	2	**0**
	1	3	**0**
	0	4	**0**
$\overline{\mathscr{L}(2,3)}\cdot\{1\}$	0	3	**1**
	1	2	**1**
	2	1	**1**
	3	0	**1**
$\mathscr{L}(2,2)\cdot\{2\}$	2	0	**2**
	1	1	**2**
	0	2	**2**
$\overline{\mathscr{L}(2,1)}\cdot\{3\}$	0	1	**3**
	1	0	**3**
$\mathscr{L}(2,0)\cdot\{4\}$	0	0	**4**

Table 2.3 – Exemple de code de Gray pour $\mathscr{L}(3,4)$.

Définition 2.6

✎ Pour deux n-uplets $\boldsymbol{b} = b_1 b_2 \dots b_n$ et $\boldsymbol{c} = c_1 c_2 \dots c_n$, $\boldsymbol{c}$ est dit $\boldsymbol{b}$-*borné* si on a $0 \leqslant c_i \leqslant b_i$ pour tout i avec $1 \leqslant i \leqslant n$. □

Walsh [100] a donné un algorithme *loopless* pour générer un code de Gray pour les compositions $\boldsymbol{b}$-bornées d'un entier et a en particulier généralisé le code de Gray de Knuth cité précédemment. La fonction successeur de cet algorithme utilise deux *pivots* pour générer chaque objet en $O(1)$.

Chapitre 3

Permutations

3.1 Définitions

Dans cette partie, nous allons définir diverses notions à propos des permutations que nous aurons à utiliser dans la suite de ce manuscrit. Tout d'abord, rappelons la définition d'une permutation.

Définition 3.1 : permutation

✎ Une *permutation* de l'ensemble $\{1,2,\ldots,n\}$ est une bijection de $\{1,2,\ldots,n\}$ sur lui-même. L'entier n représente la *taille* ou la *longueur* d'une permutation. □

Notons S_n l'ensemble des permutations de longueur n et $\mathcal{S} = \bigcup_{n=0}^{\infty} S_n$.

De nombreuses représentations des permutations existent, dont la représentation linéaire (3.1.1), la représentation graphique (3.1.2 page 40) et la représentation en cycles (3.1.3 page 40).

3.1.1 Représentation linéaire

La *représentation linéaire* d'une permutation $\sigma \in S_n$ est la suivante : $\sigma_1\sigma_2\ldots\sigma_n$ où $\sigma_i = \sigma(i) \ \ \forall i \in \{1,2,\ldots,n\}$. σ_i est l'élément de *valeur* $\sigma(i)$ dans σ, i est appelé l'*indice* (ou la *position*) de cet élément.

3.1.2 Représentation graphique

La représentation graphique d'une permutation $\sigma \in S_n$ se fait à l'aide d'une grille carrée de coté n contenant un seul point par ligne et par colonne. Les colonnes représentent les indices i de la permutation et les lignes les valeurs σ_i correspondantes.

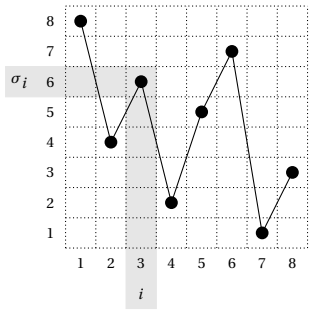


Figure 3.1 – Représentation graphique de la permutation $\sigma = 84625713$.

Cette représentation sera très utilisée dans cette thèse puisqu'elle permet de capter aisément les propriétés des motifs (définition 3.15 page 45).

3.1.3 Représentation en cycles

Un *cycle* dans une permutation $\sigma \in S_n$ est une séquence $(s_0 s_1 \dots s_{j-1})$ telle que $\sigma(s_i) = s_{(i+1) \bmod j}$ pour tout i, $0 \leqslant i \leqslant j-1$.

Toute permutation est un produit de cycles disjoints. Par exemple, la permutation $\sigma = 4251763 \in S_7$ est le produit de 4 cycles qui sont (41), (2), (573) et (6).

On dira qu'une permutation σ contient k cycles si elle se décompose en un produit de k cycles à supports disjoints.

Dans l'exemple suivant, $\sigma = 836174529 = (182364)(57)(9)$.

$$\begin{array}{c} 8\ \ 3\ \ 6\ \ 1\ \ 7\ \ 4\ \ 5\ \ 2\ \ 9 \\ \Downarrow \\ (182364)\quad (57)\quad (9) \end{array}$$

Les permutations de taille n ayant exactement k cycles sont comptées par les

nombres de Stirling de première espèce non signés [7, 51, 89, 90, 99] (A008275 [84]).

Définition 3.2 : permutation identité

✎ La permutation *identité* de taille n est la permutation $123\ldots n$. □

On note Id_n la permutation identité de longueur n.

Définition 3.3 : séquence dans une permutation

✎ Une *séquence* s de longueur k, dans une permutation $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ de longueur n est : $s = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_k}$ où $1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant n$ et $1 \leqslant k \leqslant n$. □

Dans la suite de ce manuscrit, sans autre précision, une séquence sera toujours supposée composée d'entiers positifs tous différents deux-à-deux.

Définition 3.4 : normalisation

✎ Soit s une séquence, on définit la *normalisation* de s notée $\mathrm{norm}(s)$ la séquence obtenue par le processus suivant : on prend le plus petit entier de celle-ci qu'on transforme en un 1, on prend le second plus petit entier qu'on transforme en un 2, etc. □

Par exemple $\mathrm{norm}(16984) = 13542$. Évidemment, le résultat de la normalisation d'une séquence s de longueur k est une permutation de S_k.

Définition 3.5 : isomorphisme

✎ On dira d'une séquence s qu'elle est *isomorphe* à une permutation σ si et seulement si $\mathrm{norm}(s) = \sigma$. □

Définition 3.6 : miroir

✎ Le *miroir* de $s = s_1 s_2 \ldots s_n$ est $\overline{s} = s_n s_{n-1} \ldots s_1$. □

Définition 3.7 : sous-chaîne

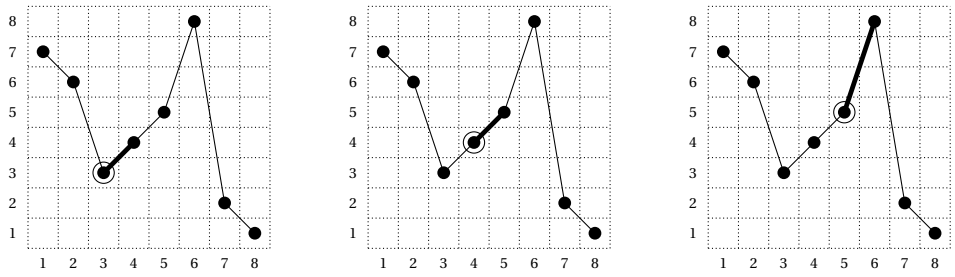
✎ Soit $s = s_1 s_2 \ldots s_n$ une séquence. On appelle *sous-chaîne* de s une séquence $s_{i_1} s_{i_2} \ldots s_{i_k}$ telle que l'ensemble $\{i_j, 1 \leqslant j \leqslant k\}$ est un intervalle, c'est-à-dire lorsque la séquence se trouve être composée d'éléments consécutifs de s. □

Par exemple, dans la permutation $\sigma = 7\mathbf{6345}821$, la séquence 6345 (représentée

en gras) est une sous-chaîne.

Définition 3.8 : montée et descente

✎ Une *montée* d'une séquence $s = s_1 s_2 \dots s_n$ est une position i ($1 \leqslant i \leqslant n-1$) telle que $s_i < s_{i+1}$. On définit de la même manière une *descente* avec $s_i > s_{i+1}$. □

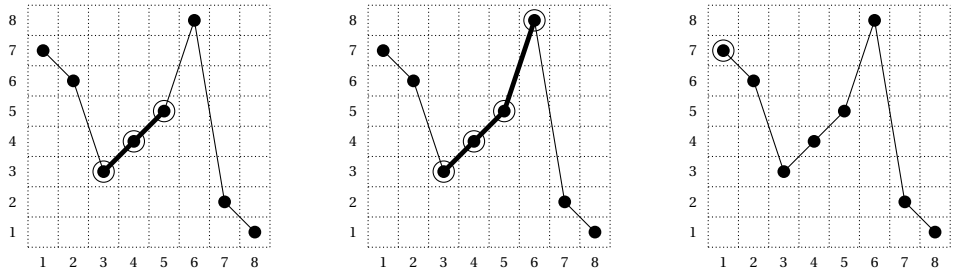


(a) (b) (c)

Figure 3.2 – Les montées dans la permutation $\sigma = 76345821$: (a) montée en 3, (b) montée en 4 et (c) montée en 5.

Définition 3.9 : *run-up* et *run-down*

✎ Un *run-up* (on trouve aussi parfois le terme français de « parcours montant » dans la littérature) est une sous-chaîne de s composée de montées successives. On le dira *maximal* si la prise en compte de l'élément précédent ou suivant dans s rompt la monotonie de la pente. On définit de la même manière un *run-down* à partir de descentes successives. □



(a) (b) (c)

Figure 3.3 – Différents *run-up* dans la permutation $\sigma = 76345821$: (a) 345 est un *run-up*, (b) 3458 et (c) 7 sont des *run-up* maximaux.

La figure 3.3 permet de voir la différence entre une montée (resp. descente) et un *run-up* (resp. *run-down*). Relevons avec attention qu'un *run-up* maximal peut-être de longueur 1 (figure 3.3 (c)).

Dans l'un de leurs articles, Canfield et Wilf [28] donnent plusieurs résultats à propos de l'énumération des permutations ayant un nombre déterminé de *run-up* et de *run-down* (de longueur $\geqslant 2$). Si on note $\psi_{n,k}$ le nombre de permutations de longueur n avec k *runs* (*run-up* ou *run-down*), on obtient la fonction récursive [28] suivante :

$$\psi_{n,k} = \begin{cases} 0 & \text{si } n < 2 \text{ ou } k \notin [1, n[\\ 2 & \text{si } n = 2 \text{ et } k = 1 \\ k \cdot \psi_{n-1,k} + 2 \cdot \psi_{n-1,k-1} + (n-k) \cdot \psi_{n-1,k-2} & \text{sinon.} \end{cases}$$

Définition 3.10 : vallée

✎ Une *vallée* d'une séquence s est une sous-chaîne formée d'un *run-down* suivi d'un *run-up*, chacun de longueur 2 au moins. Une vallée est donc une chaîne $s_k s_{k+1} \dots s_\ell$ $(1 \leqslant k < \ell \leqslant n)$, telle qu'il existe un j $(k < j < \ell)$ vérifiant $s_k > s_{k+1} > \dots > s_j$ et $s_j < s_{j+1} < \dots < s_\ell$. □

Une vallée est dite *maximale* si le *run-down* et le *run-up* qui la composent sont maximaux. On note $\mathrm{val}(s)$ le nombre de vallées maximales de s, c'est-à-dire la cardinalité de l'ensemble $\{j, s_j < \min\{s_{j-1}, s_{j+1}\}\}$.

Rieper et Zeleke [78] ont travaillé sur les permutations sans vallée et ont publié une formule récursive permettant d'énumérer les permutations avec un nombre de vallées déterminé.

Si on note $\nu_{n,k}$ le nombre de permutations de longueur n avec k vallées (non maximales), on obtient la fonction récursive [78] suivante :

$$\nu_{n,k} = \begin{cases} 0 & \text{si } n \leqslant 0 \\ 1 & \text{si } n = 1 \text{ et } k = 0 \\ 2 \cdot (k+1) \cdot \nu_{n-1,k} + (n-2k) \cdot \nu_{n-1,k-1} & \text{sinon.} \end{cases}$$

À partir des définitions précédentes, nous pouvons aisément constater qu'une permutation σ de taille n ne peut pas avoir plus de $\lfloor \frac{n-1}{2} \rfloor$ vallées maximales.

Définition 3.11 : permutation alternée

✎ Une permutation $\sigma \in S_n$ est dite *alternée* [1] si $\sigma_1 > \sigma_2 < \sigma_3 > \sigma_4 < \cdots$. □

On trouve également le terme de *down-up* pour désigner les permutations alternées dans la litterature [15]. Cette classe de permutations s'énumère avec les nombres d'Euler (A000111 [84]).

Remarquons qu'une permutation alternée, de part sa forme, contient exactement $\left\lfloor \frac{n-1}{2} \right\rfloor$ vallées maximales.

Définition 3.12 : inversion

✎ Soient $i, j \in \{1, 2, \ldots, n\}$ avec $i < j$, la paire (σ_i, σ_j) est une *inversion* pour $\sigma \in S_n$ si $\sigma_i > \sigma_j$. □

En pratique, deux éléments forment une inversion dans une permutation si l'élément de gauche de la paire est plus grand que le second.

Par exemple, dans la permutation $\sigma = 43152$, la paire $(3,5)$ n'est pas une inversion. En revanche, la paire $(4,2)$ est une inversion.

Définition 3.13 : nombre d'inversions

✎ Le *nombre d'inversions* [3, 46] d'une permutation σ de longueur n est le nombre de paires (σ_i, σ_j) avec $1 \leqslant i < j \leqslant n$ telles que $\sigma_i > \sigma_j$. □

On note le nombre d'inversions d'une permutation π : inv (π).

Par exemple, inv $(1426735) = 6$ car on a les paires suivantes :

$(6,5), (7,5), (4,3), (6,3), (7,3), (4,2)$.

Les permutations de taille n ayant exactement k inversions correspondent à la statistique Mahonienne [71, 72, 73] (A008302 [84]).

Définition 3.14 : major index

✎ Le *major index* [46, 71, 72, 73] d'une permutation σ est la somme des indices i où $\sigma_i > \sigma_{i+1}$. □

On note le major index d'une permutation σ : maj (σ).

Par exemple, maj $(1426735) = 7$ car on a les descentes 42 et 73 respectivement aux indices 2 et 5 d'où $2 + 5 = 7$.

Foata et Mac Mahon ont démontré [44, 45] qu'il y a une équidistribution entre le nombre d'inversions (définition 3.13) et le major index.

$$\sum_{\pi\in S_n} q^{\mathrm{inv}\,(\pi)} = \sum_{\pi\in S_n} q^{\mathrm{maj}\,(\pi)}$$

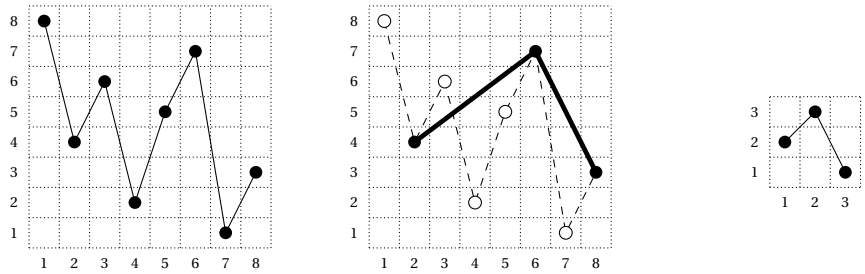
On pourra également trouver d'autres résultats sur les statistiques sur les permutations dans [86].

Définition 3.15 : motif

✎ Une permutation $\pi \in S_k$ de longueur k ($k \leqslant n$) est un *motif* de la permutation $\sigma \in S_n$ s'il existe dans σ une séquence $\sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_k}$ (avec $1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant n$) telle que $\sigma_{i_\ell} < \sigma_{i_m}$ chaque fois que $\pi_\ell < \pi_m$ ou de façon équivalente si $\mathrm{norm}\left(\sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_k}\right) = \pi$. □

Par exemple, dans la permutation $s = 615423$ on trouve le motif $\pi = 132$ à cinq reprises : 6**154**23, 6**15**4**2**3, 6**15**4**23**, 6**1**5**42**3, 6**15**42**3**.

On peut aisément visualiser les motifs (figure 3.4) en utilisant la représentation graphique des permutations.



(a) (b) (c)

Figure 3.4 – Mise en valeur du motif $\pi = 231$ dans la permutation $\sigma = 84625713$: en (a) la permutation de départ, en (b) la suppression des éléments représentés en blancs et en (c) la normalisation de la séquence obtenue.

Une permutation qui ne contient pas un motif est une permutation qui *évite* ce motif. Par exemple, 1423 contient les motifs 132, 123 et 312 mais évite le motif 321. Lorsque π est un motif de σ, nous le notons $\pi \prec \sigma$.

Définition 3.16 : stabilité

✎ Soit C un ensemble de permutations. On dit que C est *stable* pour la relation d'inclusion $\prec$ si, pour toute permutation $\sigma \in C$ et pour tout motif π de σ ($\pi \prec \sigma$), on a $\pi \in C$. □

Définition 3.17 : classe

✎ On appellera *classe* un ensemble C de permutations qui est stable. Dans ce cas, un ensemble C sera noté $\mathcal{C}$. Par exemple, l'ensemble $S(B)$ des permutations excluant les motifs situés dans B est stable et sera noté $\mathcal{S}(B)$. □

La classe $\mathcal{S}$ de toutes les permutations sera alors aussi notée $\mathcal{S}$, c'est-à-dire

$$\mathcal{S} = \bigcup_{n=0}^{\infty} S_n.$$

Bouvel et Pergola ont énoncé le théorème 3.1 [23] à propos de la stabilité :

Théorème 3.1

✎ Si $\mathcal{C}$ est une classe de permutations, alors $\mathcal{C}$ est aussi une classe de permutations excluant un ensemble B de motifs (*i.e.* $\mathcal{C} = \mathcal{S}(B)$) avec
$B = \{\sigma \notin \mathcal{C}, \forall \pi \prec \sigma \text{ avec } \pi \neq \sigma, \pi \in \mathcal{C}\}$.
On dit alors que $\mathcal{C}$ est la classe des permutations de *base* B. □

Les éléments de B sont en fait les éléments minimaux du complémentaire de la classe $\mathcal{C}$, relativement à la relation d'inclusion $\prec$.

Sur la figure 3.5, les points en noir représentent les permutations de $\mathcal{C}$, les points blancs représentent les permutations de B qui sont minimales pour la relation d'inclusion $\prec$. Si on retire une valeur d'une permutation de B et qu'on normalise la séquence obtenue, on obtiendra une permutation qui appartiendra à $\mathcal{C}$.

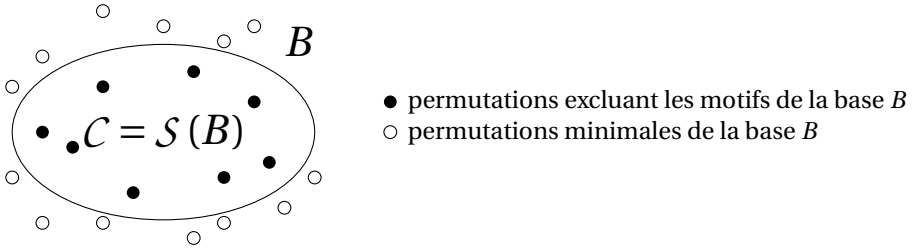


Figure 3.5 – Représentation de la stabilité de la relation d'inclusion $\prec$.

Deuxième partie

Notre étude

Chapitre 4

Duplications complètes avec perte aléatoire

Dans ce chapitre nous étudions le problème de la *duplication miroir complète avec perte aléatoire* (WM-duplication dans la suite du document) en termes de permutations à motifs exclus (définition 3.15 page 45). Afin de situer le contexte, nous parlons du fondement biologique de cette étude en 4.1. Dans la section 4.2 (page 52) nous rappelons les résultats existants dans la littérature. Dans la section 4.3 (page 55) nous détaillons le modèle de duplication présenté puis nous démontrons que la classe de permutations obtenue avec cette méthode après p WM-duplications à partir de l'identité est la classe de permutations qui évite les permutations alternées de longueur 2^p+1. Nous énumérons également le nombre de WM-duplications nécessaires et suffisantes pour obtenir une permutation quelconque de longueur n à partir de l'identité. Dans la section 4.4 (page 61) nous proposons deux algorithmes efficaces permettant de reconstituer deux chemins différents entre l'identité et la permutation recherchée en utilisant exclusivement des WM-duplications. Des résultats connexes sur d'autres modèles de duplication (4.5 page 70) sont également présentés en fin de chapitre.

4.1 Fondement biologique

Le but de cette section n'est évidemment pas d'expliquer en détails les différents processus biologiques existants, il s'agit simplement de donner un aperçu des raisons qui poussent les chercheurs à étudier le génome. Nous verrons qu'entrent en

jeu des quantités gigantesques d'éléments, ce qui explique l'approche combinatoire.

La génétique moderne se base sur un dogme principal : le modèle schématique de la conservation et de l'utilisation de l'information génétique. L'un des acteurs principaux de la transmission de l'information génétique est l'ADN. Il est le support stable et transmissible de l'information génétique qui définit les fonctions biologiques d'un organisme. L'ADN a la faculté de se répliquer ; il se réplique, puis se transcrit en ARN qui est éventuellement traduit en protéines. La transmission de l'information génétique persiste ensuite entre les parents et l'enfant : c'est le principe de l'hérédité.

Un *gène* désigne une unité d'information génétique transmise d'une génération d'une espèce à la suivante. L'ensemble des gènes d'un individu constitue son *génome*. Un gène est une unité d'information génétique à l'origine de la synthèse des protéines. Cette unité est généralement définie comme un enchaînement de portions d'ADN, situé à un endroit bien précis sur un chromosome et porteur d'une information génétique particulière. Les portions d'ADN sont alternativement porteuses ou non d'informations génétiques.

L'ADN, acronyme d'Acide DésoxyriboNucléique, est une longue molécule présente dans tous les organismes vivants. L'ADN est présent dans le noyau des cellules eucaryotes, dans les cellules procaryotes, les mitochondries ainsi que les chloroplastes. La structure de l'ADN se présente sous la forme d'une double hélice. Chacune de ces hélices est constituée d'un enchaînement d'unités de construction appelées nucléotides.

La figure 4.1 (page 51)[1] présente la décomposition schématique d'une cellule. Une cellule contient un noyau composé de chromosomes. C'est dans les chromosomes qu'on trouve l'ADN formant les gènes. Les paires de base sont les suivantes :

- A – Adénine ;
- T – Thymine ;
- C – Cytosine ;
- G – Guanine.

[1] Source : *NIH (USA), National Human Genome Research Institute, division of intramural research.*

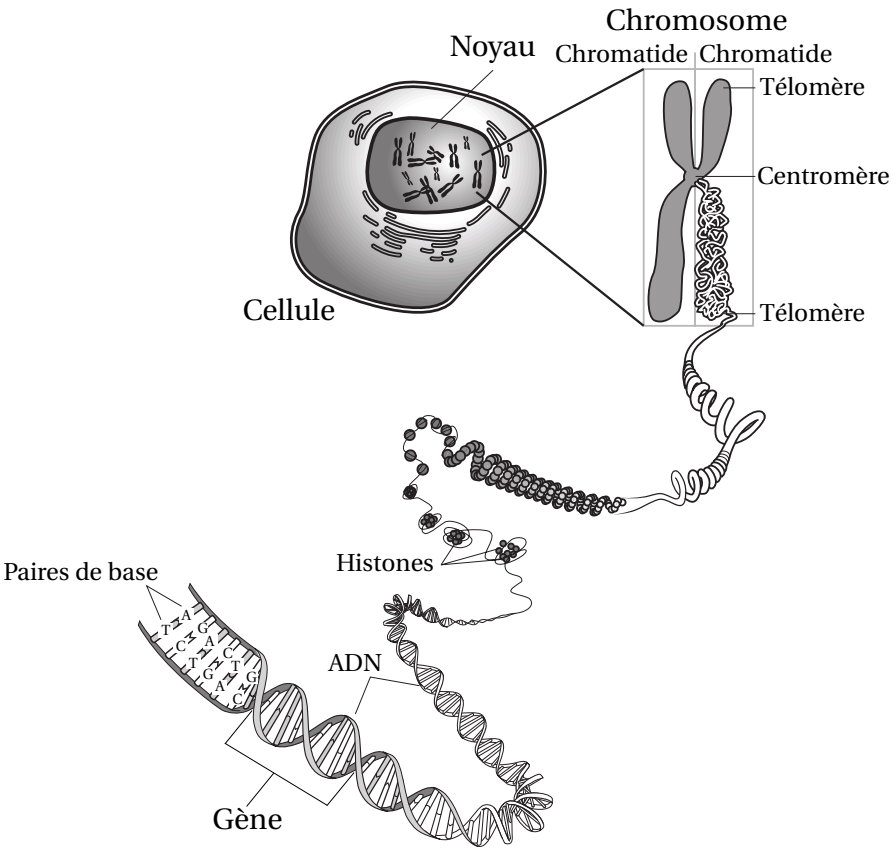


Figure 4.1 – De la cellule aux acides de base.

Les mécanismes mis en jeu sont complexes : il peut se produire des mutations à l'échelle des nucléotides ou des mutations plus globales à l'échelle des gènes basées sur la modification de l'ordre de ceux-ci dans le génome [20]. Les mutations au niveau de l'ADN sont typiquement l'insertion, la délétion et la substitution. Mathématiquement, on considère l'ADN comme les mots d'un alphabet de quatre lettres correspondant aux quatres acides de base : $\{A, T, C, G\}$.

Dans notre étude, nous travaillons à une échelle plus large, celle des gènes. Les mutations sur les gènes peuvent également être la délétion [25] et l'insertion [16] mais on en trouve d'autres, comme la duplication [14, 24, 32], la transposition [34, 65, 66], l'inversion [2, 36, 37, 42, 82, 97], etc. La complexité de ces processus est considérable, aussi se limite-t-on le plus souvent à l'étude d'une seule de ces mutations à la fois.

Un génome de n gènes est alors naturellement représenté par une permutation $\sigma \in S_n$: chaque gène correspondant à un numéro unique, entre 1 et n. Un entier dans la permutation représente donc un gène et la permutation représente l'ordre d'apparition des gènes dans le génome. On peut toujours supposer qu'un génome donné

correspond à la permutation identité $12\ldots n$; il suffit éventuellement de changer l'étiquette correspondant aux gènes. Les mutations possibles considérées dans le modèle d'évolution correspondent alors à des règles de transformation qui modifient l'ordre des éléments les uns par rapport aux autres dans la permutation. Les permutations qu'on peut alors obtenir par application de ces règles sont le résultat de « mutations » de la permutation de départ. Dans certains cas, on peut améliorer cette modélisation en utilisant plutôt des permutations signées lorsque l'on prend en compte le fait que les gènes ont un sens dans les génomes.

Dans le cadre des modèles de duplication étudiés dans ce manuscrit, la modélisation des génomes se fera à l'aide de permutations non signées.

4.2 Introduction

On dit d'un génome qu'il se duplique lorsqu'un segment de ce génome se copie juste après celui-ci et qu'une partie des gènes copiés disparaît afin de ne conserver qu'un exemplaire de chacun d'entre eux. Il existe différentes méthodes de duplication qu'on peut retrouver en détail dans plusieurs articles [26, 34, 55, 66, 81]. Dans un précédent article, Chaudhuri, Mihaescu et Rao [32] ont étudié la *duplication tandem par segment avec perte aléatoire*, (S_k-duplication dans la suite du document). Cette duplication procède de la manière suivante : on prend un segment de taille k du génome et on le recopie juste après la portion originale. En faisant cela, on se retrouve avec des gènes en double. On doit donc en retirer afin de n'en conserver qu'un seul de chaque : c'est la procédure de *perte aléatoire* (figure 4.2). Ce modèle provient de la biologie évolutionnaire où il se retrouve dans le génome mithocondrial de certains vertébrés.

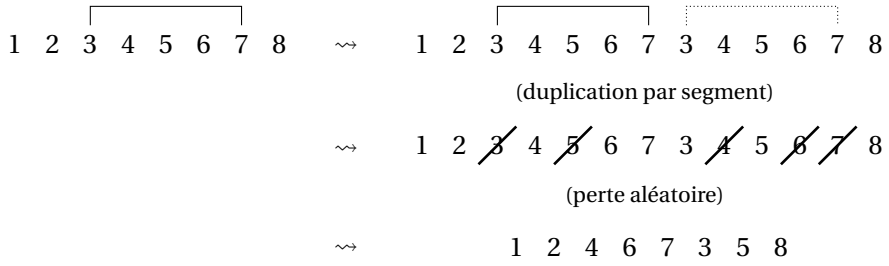


Figure 4.2 – Duplication par segment de longueur 5 avec perte aléatoire.

On parlera de *duplication complète* (W-duplication dans la suite de ce manuscrit) lorsque l'on duplique la totalité de la permutation.

Dans ce même article [32], les auteurs introduisent notamment la notion de *distance entre deux génomes* et produisent un algorithme qui permet de la déterminer efficacement dans certains cas.

La définition générale d'une distance mathématique (définition 4.1) est basée sur trois axiomes.

Définition 4.1 : distance

✎ Une *distance d* sur un ensemble S est une application $d : S \times S \to \mathbb{R}^+ : (s,t) \mapsto r$ telle que :

- $\forall s,t \in S : d(s,t) \geqslant 0$ et $d(s,t) = 0 \Leftrightarrow s = t$
- $\forall s,t \in S : d(s,t) = d(t,s)$
- $\forall s,t,u \in S : d(s,u) \leqslant d(s,t) + d(t,u)$ □

Reprenons maintenant dans la définition 4.2 la notion de la distance entre deux génomes définie dans [32]. Remarquons que dans le cas des études sur le génome, la distance entre deux génomes peut ne pas être symétrique. Il n'est en effet pas systématique que le coût pour passer d'une permutation $\pi \in S$ vers une permutation $\sigma \in S$ soit le même que pour passer de σ vers π.

Définition 4.2 : distance entre deux génomes

✎ Soient une permutation σ et un paramètre α, la *distance* est le coût minimal nécessaire pour passer de la permutation identité à la permutation σ. Le coût associé à chaque étape de duplication d'un segment de longueur k est égal à α^k et la distance se note alors $d_\alpha(\sigma)$. □

Le coût est inversement proportionnel à la probabilité de trouver un élément de la permutation dans le segment dupliqué ; dans le cas d'une duplication complète, le coût est constant et égal à 1. La distance entre l'identité et une permutation σ est donc égal au nombre d'étapes de duplications nécessaires pour obtenir σ à partir de l'identité.

Dans un autre article, Bouvel et Rossin [21, 24] étudient également cette duplication. Voici les principaux résultats obtenus par les auteurs dans le cas de chemins de W-duplications.

Le théorème 4.1 [32] provient d'une caractérisation implicite de Chaudhuri *et al.*

Théorème 4.1

✎ Soit σ une permutation de taille n. Dans le modèle de duplication complète avec perte aléatoire, $\lceil \log_2(\text{nombre de } \textit{run-up} \text{ maximaux de } \sigma) \rceil$ étapes de W-duplications sont nécessaires et suffisantes pour obtenir la permutation σ à partir de $12\dots n$. □

Le théorème 4.1 [32] peut être reformulé en théorème 4.2 [24, 32] pour utiliser les descentes au lieu du nombre de *run-up* maximaux.

Théorème 4.2

✎ Soit σ une permutation de taille n. Dans le modèle de duplication complète avec perte aléatoire, $\lceil \log_2(\text{nombre de descentes de } \sigma + 1) \rceil$ étapes de W-duplications sont nécessaires et suffisantes pour obtenir la permutation σ à partir de $12\dots n$. □

Cela signifie que les permutations obtenues après p étapes de W-duplications sont les permutations qui contiennent au plus $2^p - 1$ descentes.

Une permutation à d descentes et minimale pour ce critère est de taille au plus $2d$: par minimalité, elle ne contient jamais deux montées consécutives. Cela implique notamment que les motifs exclus qui caractérisent la classe de permutations que l'on peut obtenir en au plus p étapes de W-duplications forment une base finie comme énoncé dans le théorème 4.3 [24].

Théorème 4.3

✎ Les permutations que l'on peut obtenir en au plus p étapes dans le modèle de duplication complète avec perte aléatoire forment une classe de permutations. Cette classe est caractérisée par une base finie de motifs exclus : les permutations ayant exactement 2^p descentes et qui sont minimales pour ce critère. □

La décomposition d'une permutation $\sigma \in S_n$ en séquences non vides de descentes séparées par des montées isolées permet d'obtenir la caractérisation du théo-

rème 4.4 [24].

Théorème 4.4

✎ Une permutation $\sigma \in S_n$ est une permutation minimale à d descentes si et seulement si elle possède exactement d descentes et chacune de ses montées $\sigma_i\sigma_{i+1}$ est telle que $2 \leqslant i \leqslant n-2$ et $\sigma_{i-1}\sigma_i\sigma_{i+1}\sigma_{i+2}$ forme une occurrence soit du motif 2143, soit du motif 3142. □

Le résultat d'énumération obtenu par les permutations minimales à d descentes et de taille $2d$ est énoncé par le théorème 4.5 [24].

Théorème 4.5

✎ Les permutations minimales à d descentes et de taille $2d$ sont énumérées par les nombres de Catalan $c_d = \dfrac{1}{d+1}C_{2d}^{d}$. □

Mansour et Yan ont poursuivi l'étude de ces travaux et ont ajouté notamment le théorème 4.6 [74] pour les permutations minimales de longueur $2d-1$ avec d descentes.

Théorème 4.6

✎ Pour $d \geqslant 1$, le nombre de permutations minimales de longueur $2d-1$ avec d descentes est égal à $2^{d-3}(d-1)c_d$, avec c_d les nombres de Catalan tels que $c_d = \dfrac{1}{d+1}C_{2d}^{d}$. □

Les auteurs exhibent également dans leur article une fonction génératrice multivariée pour le nombre de permutations minimales de longueur n comptées par le nombre de descentes et par les valeurs des premier et second éléments de la permutation.

4.3 Notre contribution

Dans cette section nous travaillons sur la *duplication miroir avec perte aléatoire* : au lieu de copier un segment du génome comme dans la S_k-duplication, on recopie le miroir de ce segment puis on applique la procédure de perte aléatoire (figure 4.3 page 56 et figure 4.4 (page 56)). Ce modèle intervient dans la moitié des génomes en cytologie bactérienne ainsi que dans certaines études de chromosomes [33, 76]. Nous prouvons que la classe $\mathcal{C}(p)$ obtenue à partir de l'identité après p duplications miroir complètes (WM-duplications) est la classe de permutations

qui évite les permutations alternées de longueur $2^p + 1$.

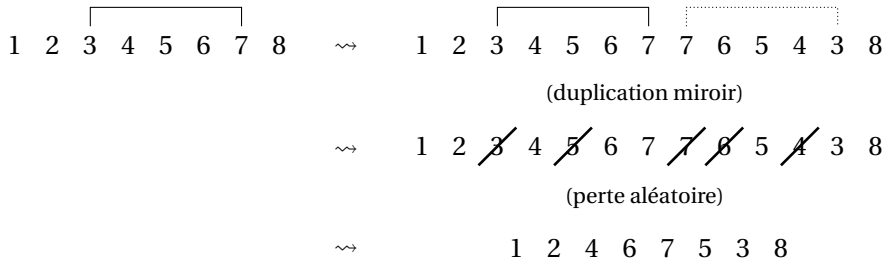


Figure 4.3 – Duplication miroir avec perte aléatoire de longueur 5.

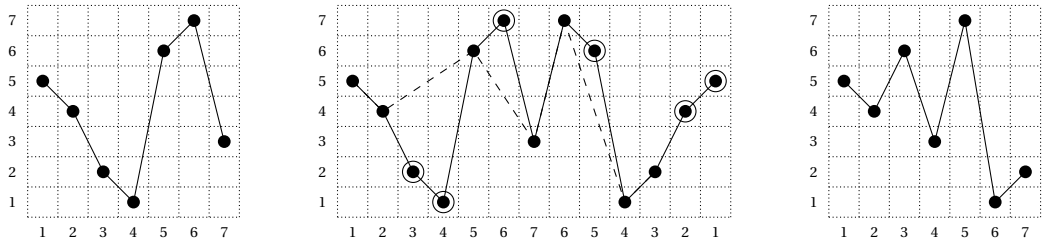


Figure 4.4 – Duplication miroir complète de la permutation 5421673. Les points entourés sont supprimés lors de la procédure de perte aléatoire.

On parlera de *duplication miroir complète* (WM-duplication dans la suite du manuscrit) lorsque l'on duplique la totalité de la permutation.

Nous énonçons deux lemmes techniques (lemme 4.1 et lemme 4.2 (page 57)) qui serviront par la suite.

Lemme 4.1

✎ Si σ et π sont deux permutations différentes telles que $\pi \prec \sigma$, alors σ contient au moins autant de vallées que π. □

Démonstration

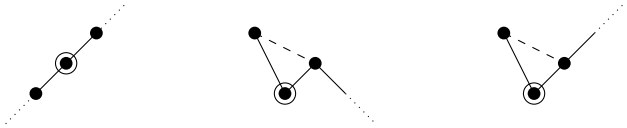
Nous vérifions cette hypothèse pour $\sigma \in S_n$ et $\pi \in S_{n-1}$ et une récurrence évidente complètera la démonstration. Prenons $\sigma \in S_n$ et $\pi \in S_{n-1}$ tels que $\pi \prec \sigma$, alors il existe une séquence obtenue à partir de σ en supprimant un entier de σ qui est isomorphe à π. Il y a essentiellement trois cas à distinguer (figure 4.5 page 57) :

(a) en supprimant la valeur entourée, on conserve le nombre de vallées ;

(b) en supprimant la valeur entourée, on retire une vallée de σ ;

(c) en supprimant la valeur entourée, on conserve le nombre de vallées.

Nous voyons bien que, quoi qu'il arrive, le fait de retirer une valeur ne peut pas entraîner la création de nouvelles vallées. ■



(a) (b) (c)

Figure 4.5 – Les 3 cas non-isomorphes de la démonstration du lemme 4.1 (page 56).

Lemme 4.2

✎ Une permutation obtenue à partir de l'identité après p WM-duplications contient au plus $2^{p-1}-1$ vallées. □

Démonstration

Nous établissons la preuve par récurrence. Le résultat est évident pour $p = 1$; en effet, une WM-duplication de l'identité ne peut pas créer de vallée puisqu'on obtient un *run-up* suivi d'un *run-down*. Maintenant, considérons que chaque permutation π obtenue à partir de l'identité après $(p-1)$ WM-duplications contient au plus $2^{p-2}-1$ vallées. Posons σ, une permutation obtenue à partir de l'identité après p WM-duplications. Alors σ est obtenue à partir d'une permutation π avec au plus $2^{p-2}-1$ vallées après exactement une WM-duplication. Par conséquent, σ peut s'écrire comme la concaténation de deux séquences de π et de $\overline{\pi}$: $\sigma = \tau\tau'$ où τ (resp. τ') est une séquence de π (resp. $\overline{\pi}$). Comme vu dans le lemme 4.1 (page 56), τ et τ' contiennent au plus $2^{p-2}-1$ vallées. Comme la concaténation de τ et τ' peut éventuellement créer une vallée entre les deux mais pas plus, σ contient au plus $2(2^{p-2}-1)+1 = 2^{p-1}-1$ vallées. Et ainsi récursivement pour toutes les valeurs de p. ■

Ces deux lemmes techniques étant énoncés, nous pouvons désormais énoncer notre premier résultat [14] par le théorème 4.7.

Théorème 4.7

✎ La classe $\mathcal{C}(p)$ des permutations obtenues après p WM-duplications à partir de l'identité est la classe des permutations avec au plus $2^{p-1}-1$ vallées. □

Démonstration

Considérant le lemme 4.2 que nous avons démontré auparavant, il reste simplement à prouver que chaque permutation σ avec au plus $2^{p-1}-1$ vallées peut être obtenue à partir de l'identité après p WM-duplications. Nous procédons par récur-

rence sur p. Soit σ une permutation avec k vallées tel que $2^{p-2}-1 < k \leqslant 2^{p-1}-1$. Alors σ peut s'écrire $\sigma = \tau\tau'$ où τ correspond au plus long préfixe de σ contenant exactement $2^{p-2}-1$ vallées et τ' le suffixe restant.

Nous décomposons la permutation $\tau = u_1 d_1 u_2 d_2 \dots u_\ell d_\ell$, où u_i et d_i sont respectivement des *run-up* et des *run-down* maximaux définis ainsi :

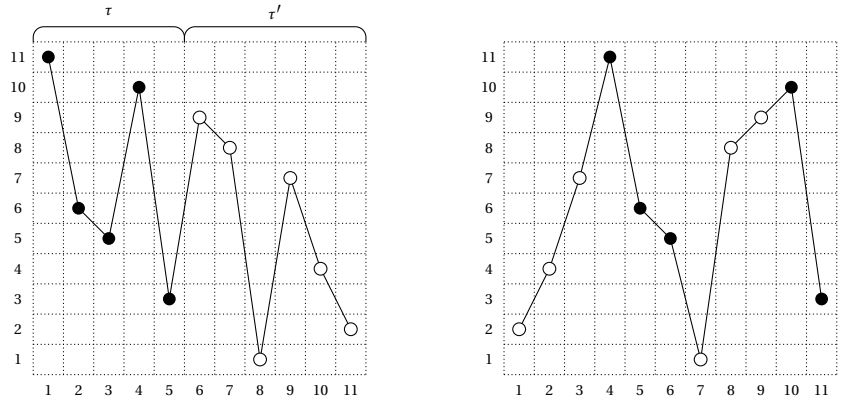
- u_1 le premier *run-up* maximal exceptée sa dernière valeur ;
- d_1 le *run-down* maximal juste après u_1 ;
- u_2 le *run-up* maximal juste après d_1 exceptée sa dernière valeur ;

et ainsi de suite… Notons que u_1 peut être vide. En revanche d_ℓ ne peut pas l'être. Par exemple, $\tau = 5421673$ se décompose comme suit : u_1 est vide, $d_1 = 5421$, $u_2 = 6$ et $d_2 = 73$.

Posons également $\tau' = u_{\ell+1} d_{\ell+1} \dots u_k d_k$ la décomposition similaire pour τ'. Maintenant procédons ainsi : nous trions en ordre décroissant les valeurs qui apparaissent dans d_ℓ et u_1 afin de créer un *run-down* D_ℓ ; nous trions en ordre croissant les valeurs dans u_ℓ et $d_{\ell+1}$ de manière à créer un *run-up* U_ℓ ; on obtient alors la séquence $S = U_\ell D_\ell$. Nous trions en ordre décroissant la séquence $D_{\ell-1}$, contenant les valeurs de $d_{\ell-1}$ et $u_{\ell+2}$, etc. À chaque étape j, nous insérons les séquences obtenues U_j et D_j au début de S. La permutation $S = \cdots U_{\ell-1} D_{\ell-1} U_\ell D_\ell$ obtenue à la fin du processus contient au plus $2^{p-2}-1$ vallées. Un exemple de construction de S permet de visualiser le processus (figure 4.6 page 59).

Par récurrence, nous obtenons donc S avec $(p-1)$ WM-duplications. Par construction, σ peut être obtenue par une WM-duplication de S, ce qui signifie que σ s'obtient avec p WM-duplications à partir de l'identité.

Remarquons que la permutation σ est décomposée en une succession de *run-up* et de *run-down* u_j, d_j, u_j et d_j. Cette décomposition sera utilisée pour construire un algorithme (4.4 page 61) déterminant l'un des plus courts chemins entre l'identité et une permutation σ. ■



$\sigma = 11\ 6\ 5\ 10\ 3\ 9\ 8\ 1\ 7\ 4\ 2$ $S = 2\ 4\ 7\ 11\ 6\ 5\ 1\ 8\ 9\ 10\ 3$

Figure 4.6 – Décomposition $\sigma = \tau\tau'$ et la permutation S obtenue après le processus décrit dans le théorème 4.7 (page 57). Nous avons $\tau = 11\ 6\ 5\ 10\ 3$, $\tau' = 9\ 8\ 1\ 7\ 4\ 2$, u_1 est vide, $d_1 = 11\ 6\ 5$, u_2 est vide, $d_2 = 10\ 3$, u_3 est vide, $d_3 = 9\ 8\ 1$, u_4 est vide, $d_4 = 7\ 4\ 2$ et $D_2 = 10\ 3$, $U_2 = 1\ 8\ 9$; $D_1 = 11\ 6\ 5$; $U_1 = 2\ 4\ 7$. D'où $S = U_1D_1U_2D_2 = 2\ 4\ 7\ 11\ 6\ 5\ 1\ 8\ 9\ 10\ 3$.

À partir du théorème 4.7 (page 57), nous pouvons déterminer dans le corollaire 4.1 le nombre d'étapes de WM-duplications nécessaires pour obtenir une permutation quelconque $\sigma \in S_n$ à partir de l'identité.

Corollaire 4.1

✎ Soient σ une permutation et $\mathrm{val}(\sigma)$ le nombre de vallées de celle-ci. Dans le modèle de WM-duplication, $\lceil \log_2(\mathrm{val}(\sigma)+1) \rceil + 1$ étapes sont nécessaires et suffisantes pour obtenir σ à partir de l'identité. □

Démonstration

Soit p un entier choisi de telle sorte que $2^{p-2} - 1 < \mathrm{val}(\sigma) \leqslant 2^{p-1} - 1$, c'est-à-dire que $p = \lceil \log_2(\mathrm{val}(\sigma)+1) \rceil + 1$. Nous savons d'après le théorème 4.7 (page 57) que $p = \lceil \log_2(\mathrm{val}(\sigma)+1) \rceil + 1$ étapes suffisent pour obtenir σ à partir de l'identité. Dans ce cas, nous avons nécessairement $\mathrm{val}(\sigma) \geqslant 2^{p-2}$; par conséquent, σ ne peut pas être obtenue à partir de l'identité en $(p-1)$ étapes seulement, une de plus est nécessaire. ■

Il est également possible de caractériser la classe des permutations qui sont obtenues après p duplications en termes de motifs exclus. Cette caractérisation énoncée dans le théorème 4.8 (page 60) utilise la notion de permutations alternées vue dans

la définition 3.11 (page 43).

Théorème 4.8

✎ La classe de permutations $\mathcal{C}(p)$ obtenue après p WM-duplications est également la classe de permutations qui évite les permutations alternées de longueur 2^p+1. □

Démonstration

La classe de permutations obtenue après p WM-duplications est stable pour la relation d'inclusion $\prec$; c'est-à-dire que si $\sigma \in \mathcal{C}(p)$ et $\pi \prec \sigma$ alors $\pi \in \mathcal{C}(p)$ (lemme 4.1 page 56). Donc $\mathcal{C}(p)$ est également une classe de permutations à motifs exclus $\mathcal{S}(B)$ où B est l'ensemble des permutations minimales σ (relativement à $\prec$) qui ne sont pas dans $\mathcal{C}(p)$. Une permutation minimale σ de ce type contient exactement 2^{p-1} vallées. En fait, $\sigma \notin \mathcal{C}(p)$ signifie que $\mathrm{val}(\sigma) \geqslant 2^{p-1}$ et toute permutation avec au moins $2^{p-1}+1$ vallées n'est pas minimale tant qu'elle contient au moins un motif π avec 2^{p-1} vallées. De plus, les configurations $\sigma_{i-1} < \sigma_i < \sigma_{i+1}$ et $\sigma_{i-1} > \sigma_i > \sigma_{i+1}$ ne peuvent pas se produire car si on supprime σ_i, on ne réduit pas le nombre de vallées. Nous avons donc nécessairement $\sigma_1 > \sigma_2$ et $\sigma_1 > \sigma_2 < \sigma_3 > \sigma_4 < \cdots$. Nous en déduisons donc que σ est une permutation alternée avec 2^{p-1} vallées et sa longueur est nécessairement 2^p+1. ■

Voir la table 4.1 un exemple de $\mathcal{C}(p) = \mathcal{S}(B)$ pour $p = 1$ et $p = 2$:

p	B
1	213,312
2	21435,31425,41325,32415,42315,21534,31524,51324, 32514,52314,41523,51423,42513,52413,43512,53412

Table 4.1 – Exemple de $\mathcal{C}(p) = \mathcal{S}(B)$ pour $p = 1$ et $p = 2$.

La table 4.1 présente la liste exhaustive des permutations de la base B pour $p = 1$ et $p = 2$. Pour $p = 3$ et $p = 4$ on a respectivement 7 936 et 209 865 342 976 permutations dans la base. En fait, les permutations alternées de longueur n sont comptées par les nombres d'Euler (séquence A000111 [84]).

De manière plus générale, nous obtenons également un résultat similaire (théorème 4.9 page 61) pour la classe de permutations avec au moins p vallées.

Notons que la fonction génératrice pour compter le nombre de permutations en

fonction de la statistique du nombre de vallées est donné dans [60] :

$$1-\frac{1}{y}+\frac{1}{y}\sqrt{y-1}\times\tan\left(x\sqrt{y-1}+\arctan\left(\frac{1}{\sqrt{y-1}}\right)\right)$$

À partir de cette fonction et de précédents travaux [59, 78], nous obtenons la fonction génératrice bivariée suivante pour cette classe :

$$\frac{1}{1-y}\left(1-\frac{1}{y}+\frac{1}{y}\sqrt{y-1}\times\tan\left(x\sqrt{y-1}+\arctan\left(\frac{1}{\sqrt{y-1}}\right)\right)\right)$$

où y est la variable correspondant au nombre maximal de vallées dans la permutation et x sa longueur.

On peut caractériser la classe des permutations avec au plus v vallées en terme de motifs exclus en utilisant de nouveau les permutations alternées.

Théorème 4.9

✎ La classe des permutations qui ont au plus v vallées est la classe de permutations qui évite les permutations alternées de longueur $2v+3$. □

Démonstration

La classe des permutations qui ont au plus v vallées est stable pour la relation d'inclusion $\prec$; c'est-à-dire que si $\sigma \in \mathcal{C}$ et $\pi \prec \sigma$ alors $\pi \in \mathcal{C}$ (lemme 4.1 page 56). Donc $\mathcal{C}$ est également une classe de permutations à motifs exclus $\mathcal{S}(B)$ où B est l'ensemble des permutations minimales σ (relativement à $\prec$) qui ne sont pas dans $\mathcal{C}$. Une permutation minimale σ de ce type contient exactement $v+1$ vallées. En fait, $\sigma \notin \mathcal{C}(p)$ signifie que $\mathrm{val}(\sigma) \geqslant v+1$ et toute permutation avec au moins $v+2$ vallées n'est pas minimale tant qu'elle contient au moins un motif π avec $v+1$ vallées. De plus, les configurations $\sigma_{i-1} < \sigma_i < \sigma_{i+1}$ et $\sigma_{i-1} > \sigma_i > \sigma_{i+1}$ ne peuvent pas se produire car si on supprime σ_i, on ne réduit pas le nombre de vallées. Nous avons donc nécessairement $\sigma_1 > \sigma_2$ et $\sigma_1 > \sigma_2 < \sigma_3 > \sigma_4 < \cdots$. Nous en déduisons donc que σ est une permutation alternée avec $v+1$ vallées et sa longueur est nécessairement $2v+3$. ■

4.4 Algorithmes

Il existe de nombreux chemins de WM-duplications permettant d'obtenir une permutation $\sigma \in S_n$ en partant de l'identité. Les deux algorithmes que nous pré-

sentons ici permettent de reconstituer de manière efficace deux chemins possibles qui conduisent à une permutation $\sigma \in S_n$ à partir de l'identité, en utilisant la WM-duplication. Dans les deux cas, il s'agit du plus court chemin possible.

L'algorithme présenté en 4.4.1 permet de reconstruire un plus court chemin entre l'identité et une permutation σ en « rebroussant chemin » à partir de la permutation σ jusqu'à l'identité. L'algorithme présenté en 4.4.2 (page 66) permet de reconstituer un autre plus court chemin en utilisant une construction faisant usage du code de Gray binaire réfléchi [52]. Nous étudions également la complexité de ces deux algorithmes.

4.4.1 Un chemin de $12\dots n$ vers $\sigma \in S_n$: algorithme 1

Voici un plus court chemin possible de l'identité vers $\sigma \in S_n$ en utilisant des WM-duplications. Décomposons la permutation σ de la manière suivante :

$$\sigma = u_1 d_1 u_2 d_2 \dots u_k d_k$$

où

- $u_1 = \sigma_1 \dots \sigma_{i_1}$ est le *run-up* maximal contenant la première valeur σ_1 ;
- d_k est le *run-down* maximal contenant σ_n ;
- d_1 est le *run-down* maximal contenant σ_{i_1+1} ;
- u_k est le *run-up* maximal juste avant d_k (autrement dit, la valeur juste avant d_k dans σ appartient à u_k) ;

Nous continuons ce processus en alternant les *run-up* et les *run-down* jusqu'à ce que la permutation σ soit entièrement décomposée en *run-up* et *run-down*. Pour i de 1 à $\ell = \left\lfloor \frac{k+1}{2} \right\rfloor$, nous appellons U_i (resp. D_i) la sous-chaîne croissante (resp. décroissante) formée du tri des valeurs de u_i et d_{k-i+1} (resp. d_i et u_{k-i+1}).

Posons $\pi = U_1 D_1 U_2 D_2 \dots U_\ell D_\ell$, la concaténation de toutes les sous-chaînes montantes et descendantes U_i et D_i (où D_ℓ peut être vide). Nous avons alors π qui contient au plus $(\ell - 1) = \left\lfloor \frac{k+1}{2} \right\rfloor - 1 = \left\lfloor \frac{\mathrm{val}(\sigma)+2}{2} \right\rfloor - 1 = \left\lfloor \frac{\mathrm{val}(\sigma)}{2} \right\rfloor$ vallées.

Il est bien entendu possible de détecter les *run-up* et *run-down* et de les trier en même temps, auquel cas cette étape aura une complexité en $O(n)$. En répétant cette procédure avec la permutation π nouvellement obtenue, le corollaire 4.1 (page 59) nous assure que la procédure BackPath permet de construire un plus court chemin partant de l'identité pour aller vers σ en $\left(\left\lceil \log_2(\mathrm{val}(\sigma)+1) \right\rceil + 1\right) O(n)$, c'est-à-dire

$O\big(n \cdot \log_2(n)\big)$ dans le pire cas possible.

Algorithme 1 La procédure BackPath permet de déterminer un chemin de WM-duplications pour obtenir σ en partant de $12\ldots n$.

Procedure BackPath

Tant que $\sigma \neq 12\ldots n$ **faire**

- $\pi \leftarrow$ *vide.*
- $\sigma = u_1 d_1 u_2 d_2 \ldots u_k d_k$ est la partition en *run-up* et *run-down.*

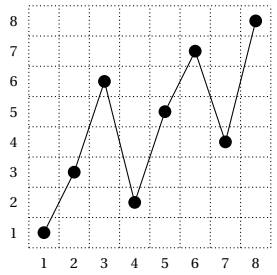
Pour $i \leftarrow 1$ **à** $\left\lfloor \frac{k+1}{2} \right\rfloor$ **faire**

- Trier en ordre croissant u_i et d_{k-i+1} et concaténer la sous-chaîne triée à la droite de π.
- Trier en ordre décroissant d_i and u_{k-i+1} et concaténer la sous-chaîne triée à la droite de π.

Fait

- $\sigma \leftarrow \pi$.

Fait



13625748

Figure 4.7 – Algorithme 1 : étape 0.

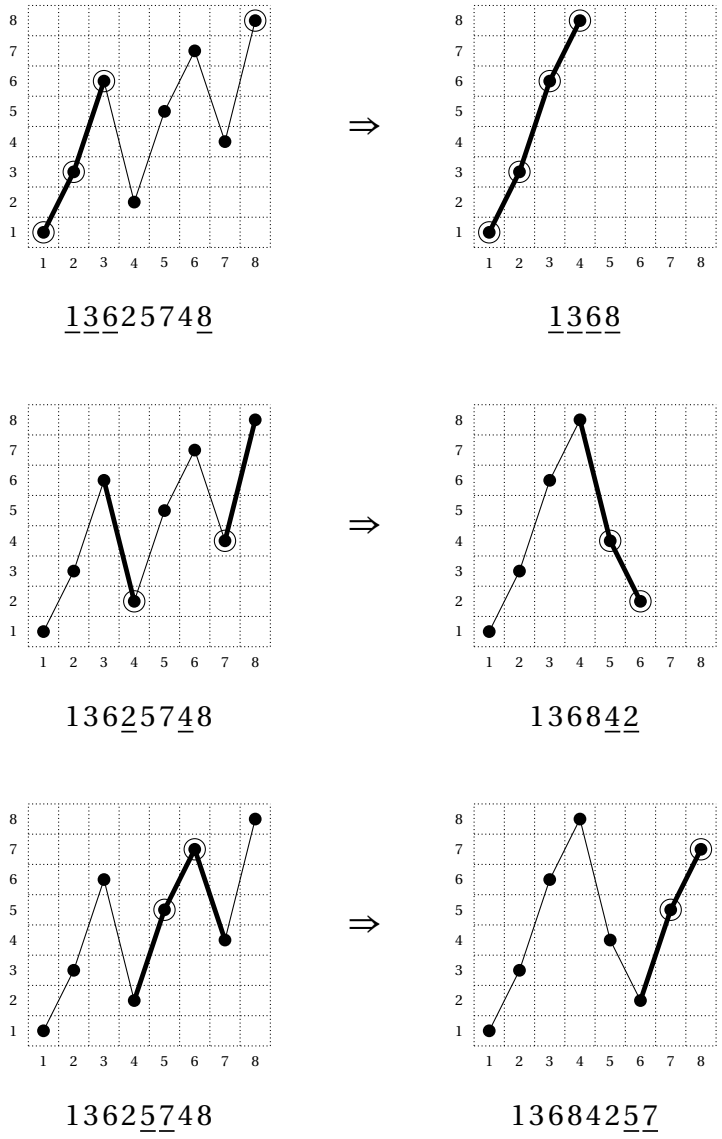
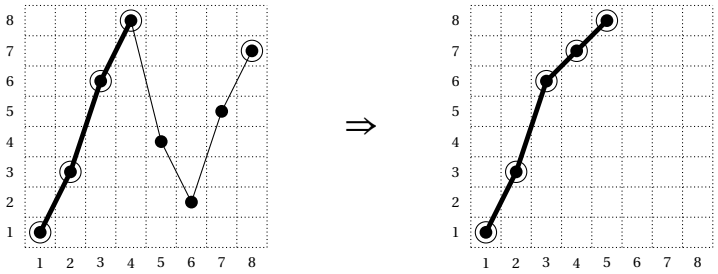
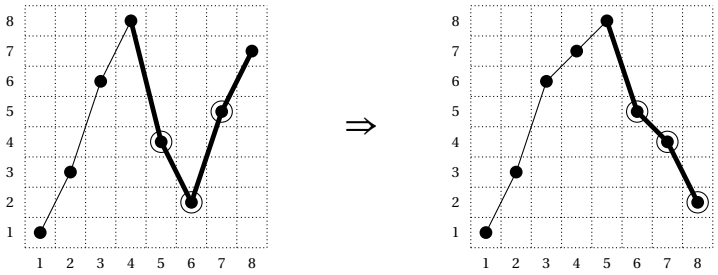


Figure 4.8 – Algorithme 1 : étape 1.

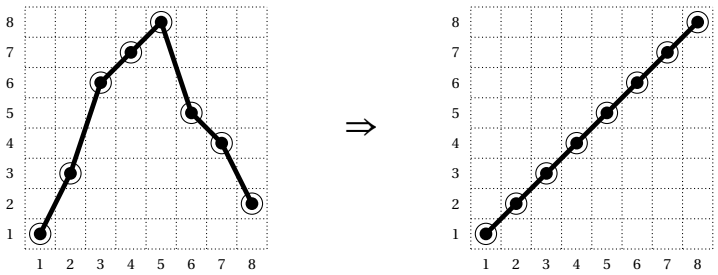


13684257 ⇒ 13678



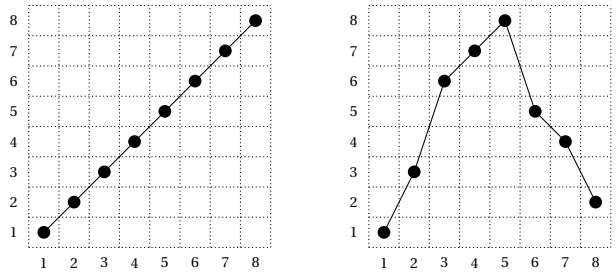
13684257 ⇒ 13678542

Figure 4.9 – Algorithme 1 : étape 2.

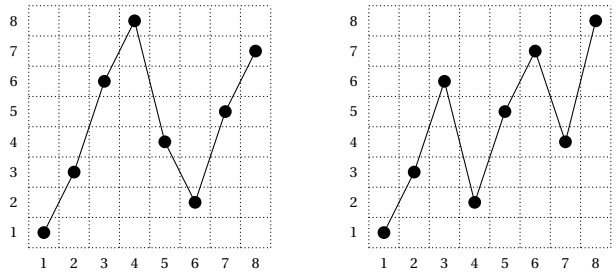


13678542 ⇒ 12345678

Figure 4.10 – Algorithme 1 : étape 3.



12345678 13678542



13684257 13625748

Figure 4.11 – Algorithme 1 : chemin final.

4.4.2 Un chemin de $12\ldots n$ vers $\sigma \in S_n$: algorithme 2

Nous allons ici trouver un autre plus court chemin. À la différence de l'algorithme précédent (4.4.1 page 62), dans celui-ci nous ferons usage du code de Gray binaire réfléchi. En effet, comme la démonstration du théorème 4.7 (page 57) le montre clairement, la reconstruction du chemin peut se faire par « pliage », une notion souvent associée aux codes de Gray.

Nous décomposons ici la permutation $\sigma = u_1 d_1 u_2 d_2 \ldots u_k d_k$ en *run-up* et *run-down* de la même manière que nous l'avons fait dans la démonstration du théorème 4.7 (page 57) :

- u_1 le premier *run-up* maximal exceptée sa dernière valeur ;
- d_1 le *run-down* maximal juste après u_1 ;
- u_2 le *run-up* maximal juste après d_1 exceptée sa dernière valeur ;

et ainsi de suite… Notons que u_1 peut être vide.

Nous allons maintenant étiqueter chaque *run-up* et chaque *run-down* avec le code de Gray binaire réfléchi [52]. Notons qu'il est possible de le faire en utilisant un algorithme sans boucle [18]. La figure 4.12 (page 68) présente un exemple d'étiquetage complet. La structure du code de Gray binaire réfléchi $\mathscr{B}_n$ est cruciale. Celui

que nous utilisons se définit comme suite : $\mathscr{B}_n = 0 \cdot \mathscr{B}_{n-1} \bigcup 1 \cdot \overline{\mathscr{B}_{n-1}}$ pour $n \geqslant 1$ avec $\mathscr{B}_0 = \lambda$ et où $\overline{\mathscr{B}_n}$ est la liste $\mathscr{B}_n$ dans l'ordre inverse.

Considérons que $\sigma = u_1 d_1 \dots u_\ell d_\ell u_{\ell+1} d_{\ell+1} \dots u_k d_k$. Tous les *run-up* et *run-down* de u_1 à d_ℓ ont une étiquette avec le dernier bit le moins significatif positionné à 0 et les autres *run-up* et *run-down* ont le même dernier bit positionné à 1. Comme dans le théorème 4.7 (page 57), nous reconstruisons la précédente permutation π à partir de σ en insérant en tant que D_ℓ (par la gauche) les valeurs triées par ordre décroissant de d_ℓ et $u_{\ell+1}$ et en tant que U_ℓ les valeurs en ordre croissant de u_ℓ et $d_{\ell+1}$, etc.

Lorsque l'on se trouve à une étape j de l'algorithme, nous procédons à une WM-duplication de π de la façon suivante :

(i) garder dans la première copie de π les éléments étiquetés 0 au $j^{\text{ème}}$ bit le moins significatif ;

(ii) conserver dans la copie miroir les éléments étiquetés 1 au $j^{\text{ème}}$ bit le moins significatif.

Un exemple est représenté dans la figure 4.12 (page 68). Cette étape s'effectue seulement en $O(n)$.

En répétant cette procédure avec la permutation π nouvellement obtenue, le corollaire 4.1 (page 59) nous assure que la procédure Path permet de construire un plus court chemin partant de l'identité pour aller vers σ en $\left(\lceil \log_2(\mathrm{val}(\sigma)+1) \rceil + 1\right) O(n)$, c'est-à-dire $O\left(n \cdot \log_2(n)\right)$ dans le pire cas possible.

Algorithme 2 La procédure Path permet de déterminer un chemin de WM-duplications pour obtenir σ en partant de $12\dots n$.

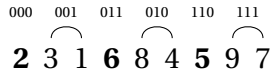
Procedure Path

- $\pi \leftarrow 12\dots n$.
- Décomposition de $\sigma = u_1 d_1 u_2 d_2 \dots u_k d_k$ en *run-up* et *run-down*.
- Étiquetage des *run-up* et *run-down* avec le code de Gray binaire réfléchi (algorithme sans boucle).

Pour $j = 1$ **à** $1 + \lfloor \log_2(k-1) \rfloor$ **faire**

- Effectuer une étape de WM-duplication sur π en conservant dans la première copie de π les éléments étiquetés 0 dans leur $j^{\text{ème}}$ bit le mois significatif et en conservant dans la copie miroir les éléments étiquetés 1 au même bit.

Fait

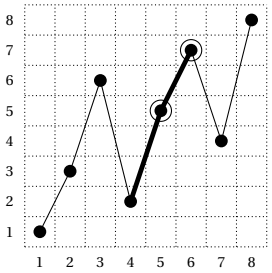


123456789 → 245897631 → 231679854 → 231684597

Figure 4.12 – Étiquetage de σ = 231684597 avec le code de Gray binaire réfléchi en fonction de ses *run-up* et *run-down*. Exemple d'un chemin de WM-duplications de l'identité vers σ.

Prenons un exemple de déroulement de l'algorithme 2. La figure 4.13 présente l'étiquetage de la permutation σ = 13625748 que l'on cherche à obtenir (les éléments 5 et 7 mis en valeur forment un exemple d'étiquetage de *run-up*). La figure 4.14 montre la construction de la permutation obtenue après la première étape de duplication ; les points en blancs correspondent aux 3$^{\text{ème}}$ bits positionnés à 1 dans l'étiquetage et sont placés à droite, les points en noirs correspondent aux 3$^{\text{ème}}$ bits positionnés à 0 dans l'étiquetage et sont conservés à gauche dans la nouvelle permutation. Le même processus est répété dans la figure 4.15 (page 69) puis dans la figure 4.16 (page 69) avec respectivement les 2$^{\text{ème}}$ et 1$^{\text{er}}$ bits. On peut enfin voir le chemin complet dans la figure 4.17 (page 69).

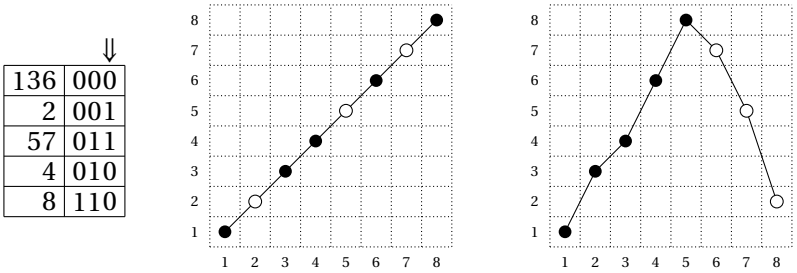
Une application sur Internet permet de tester cet algorithme :
http://jl.baril.u-bourgogne.fr/id2perm.php



136	000
2	001
57	011 ⇐
4	010
8	110

13625748 étiquetage

Figure 4.13 – Algorithme 2 : étape 0.



	⇓
136	000
2	001
57	011
4	010
8	110

3$^{\text{ème}}$ bit 12345678 13468752

Figure 4.14 – Algorithme 2 : étape 1.

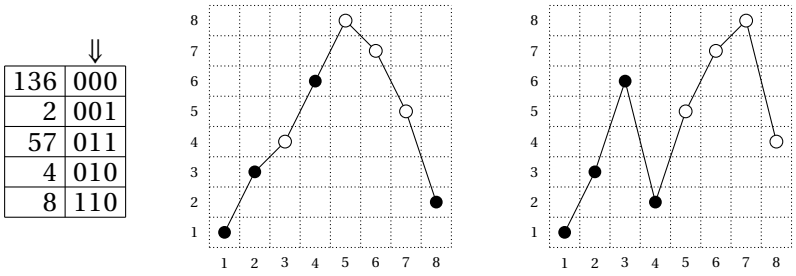


Figure 4.15 – Algorithme 2 : étape 2.

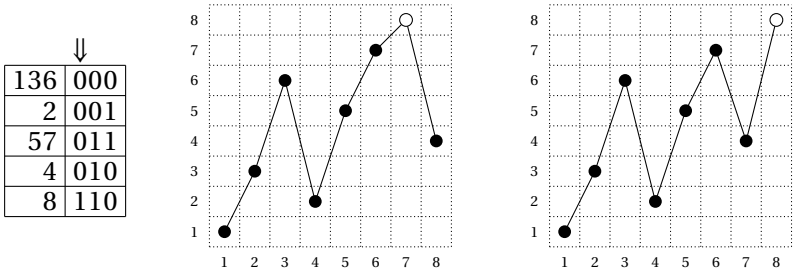


Figure 4.16 – Algorithme 2 : étape 3.

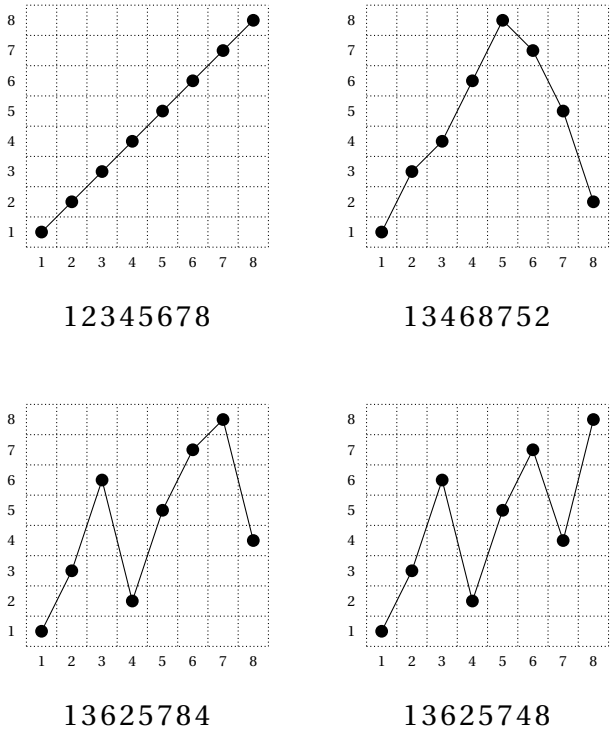


Figure 4.17 – Algorithme 2 : chemin final.

4.5 Autres modèles

Jusqu'ici, nous avons détaillé le processus total de WM-duplications. C'est-à-dire que chaque étape entre l'identité et la permutation σ recherchée est une WM-duplication. Nous allons ici présenter deux variantes. En 4.5.1 nous étudions le processus suivant : une WM-duplication de l'identité suivie de W-duplications. En 4.5.2 nous effectuons une W-duplication et/ou une WM-duplication de l'identité, suivie de W-duplications. Pour ces deux cas, nous déterminons une caractérisation de la classe de permutations obtenue après p duplications.

4.5.1 Une WM-duplication suivie de W-duplications

En 4.2 (page 52) nous avons observé qu'en faisant une WM-duplication de l'identité nous obtenions une permutation sans vallée. Par récurrence, nous pouvons affirmer que la classe de permutations obtenue après une duplication miroir de l'identité suivie par $(p-1)$ W-duplications est la classe de permutations qui ont au plus $2^{p-1}-1$ vallées.

Une nouvelle fois (théorème 4.10), nous pouvons caractériser les permutations obtenues après une WM-duplication de l'identité suivie par $(p-1)$ W-duplications en termes de motifs exclus en utilisant les permutations alternées.

Théorème 4.10

✎ La classe de permutations obtenue après une WM-duplication de l'identité suivie par $(p-1)$ W-duplications est la classe de permutations qui évite les permutations alternées de longueur 2^p+1. □

Démonstration

La preuve s'obtient de manière évidente à partir du théorème 4.7 (page 57) et du théorème 4.8 (page 60). ■

4.5.2 Une W(M)-duplication suivie de W-duplications

Ici nous allons étudier le cas d'une W-duplication ou une WM-duplication (que nous noterons W(M)-duplication) suivie de W-duplications (théorème 4.11 page 71).

La classe obtenue peut s'écrire comme l'union de deux autres classes.

Théorème 4.11

✎ La classe des permutations obtenue après une W-duplication ou une WM-duplication de l'identité suivie par p W-duplications est l'union de deux classes :

(i) la classe des permutations qui ont au plus $2^p - 1$ vallées ;

(ii) la classe des permutations avec au plus $2^{p+1} - 1$ descentes. □

Démonstration

Après une W-duplication ou une WM-duplication de l'identité, nous obtenons l'union de la classe de permutations sans vallées et de la classe de permutations avec au plus une descente. Une récurrence simple sur p nous permet d'obtenir le résultat en utilisant le théorème 4.7 (page 57) et le théorème 4.10 (page 70). ■

On peut également reprendre le théorème 4.11 pour caractériser la classe obtenue en termes de motifs exclus.

Corollaire 4.2

✎ La classe $\mathcal{C}'(p)$ des permutations obtenues après une W-duplication ou une WM-duplication de l'identité suivie de p W-duplications est la classe des permutations qui évite les permutations de longueur $3 \cdot 2^p + 1$ ayant exactement 2^p vallées et 2^{p+1} descentes. □

Démonstration

Posons $\sigma \notin \mathcal{C}'(p)$ une permutation minimale. Le fait que σ soit minimale entraîne de fait que σ ne contient aucun *run-up* de taille 3 ou plus. Par contradiction, si c'est le cas, alors en supprimant la seconde valeur du *run-up* la séquence obtenue sera isomorphe à une permutation qui n'est pas non plus dans $\mathcal{C}'(p)$. Pour les même raisons, σ ne peut pas commencer par une montée. La permutation σ a donc exactement 2^p vallées et 2^{p+1} descentes. 2^p montées plus 2^{p+1} descentes plus 1 valeur terminale, cela permet facilement de voir que les permutations sont de longueur $3 \cdot 2^p + 1$. ■

Par exemple, $\mathcal{C}'(0) = S(4132, 3142, 4312, 3241, 3214, 4231, 4213, 2143)$; nous avons déterminé la cardinalité de la base de $\mathcal{C}'(1)$ qui est 720. Dans la même idée, nous

obtenons le corollaire plus général suivant, toujours en termes de motifs exclus.

Corollaire 4.3

✎ La classe $\mathcal{C}''(p)$ des permutations qui ont au plus p vallées et au plus $2p+1$ descentes est la classe de permutations qui évite les permutations de longueur $3p+4$ ayant exactement $p+1$ vallées et $2p+2$ descentes. □

Remarquons que l'ensemble des permutations de longueur $3p+4$ ayant exactement $p+1$ vallées et $2p+2$ descentes est également l'ensemble des permutations avec la même longueur, un nombre de vallées identique et où chaque montée est immédiatement précédée par une descente (ce qui définit une vallée). Cet ensemble a été étudié par Shapiro *et al.* [48] (voir aussi la séquence A101280 [84]). Ils ont énuméré les permutations de longueur n avec k pics et avec comme propriété particulière que chaque montée est immédiatement suivie par une descente. La cardinalité de notre ensemble peut se déduire facilement en fixant les paramètres de la séquence de Shapiro *et al.* tels que $n = 3p+4$ et $k = p+1$.

Voir la table 4.2 pour les premières cardinalités de la classe $\mathcal{C}''(p)$:

p	cardinalité
0	8
1	720
2	230 144
3	179 266 560
4	277 662 253 056

Table 4.2 – Premières cardinalités de la classe $\mathcal{C}''(p)$.

4.5.3 Cas plus général

Dans cette partie, on peut choisir de faire indifféremment des W-duplications ou des WM-duplications.

Comme le montre la figure 4.18 (page 73), si l'on fait une série de W-duplications puis une WM-duplication, il est ensuite possible de choisir indifféremment de faire des W-duplications ou des WM-duplications : on obtiendra les mêmes permutations, mais avec des chemins différents.

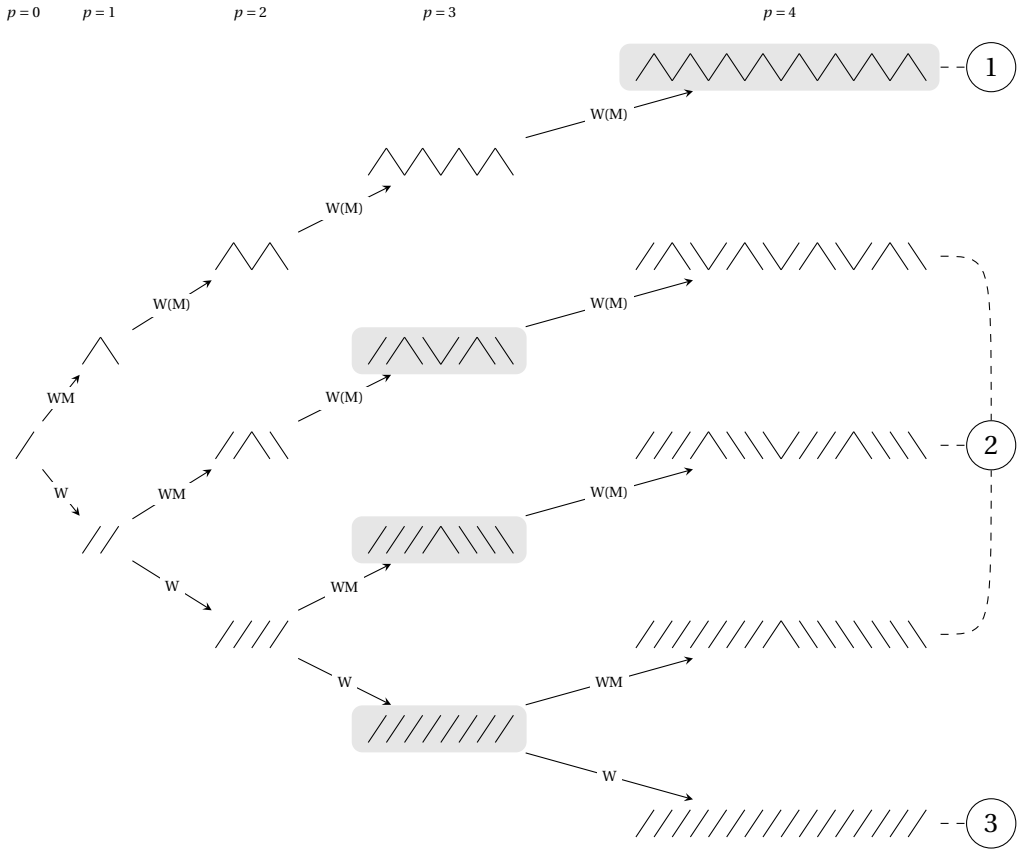


Figure 4.18 – Chemins de duplications.

Dans la figure 4.18, la branche (1) représente le chemin de duplications que nous avons étudié ici et totalement caractérisé ; il s'agit de la classe $\mathcal{C}(p)$ des permutations obtenues après p étapes étudiée dans ce chapitre. La branche (3) correspond au chemin de duplications de l'étude menée par Bouvel et Rossin [21, 24] et presque entièrement caractérisé ; on nomme $\mathcal{B}(p)$ la classe des permutations ainsi obtenues après p étapes. Entre ces deux branches, on trouve en (2) l'ensemble des chemins de duplications restants qui, eux, sont toujours à étudier ; on note $\mathcal{D}(p)$ la classe des permutations ainsi obtenues après p étapes. On peut aisément observer sur le schéma que les chemins permettant d'obtenir $\mathcal{D}(p)$ sont de la forme générale suivante :

- $x \times$ W-duplications ;
- $y \times$ WM-duplications $(y \geqslant 1)$.

Avec $p = x + y$, la longueur du chemin de duplication au niveau p.

Même si la caractérisation des branches (2) en termes de permutations à motifs exclus reste un problème ouvert pour le moment, on peut tout de même faire

la remarque 4.1 :

Remarque 4.1

✎ Les classes de permutations $\mathcal{D}(p)$ et $\mathcal{B}(p)$ obtenues après une W-duplications suivie de $p-1$ W- ou WM- duplications sont incluses dans la classe $\mathcal{C}(p+1)$ des permutations obtenues après $(p+1)$ WM-duplications. □

Formellement, on a $\mathcal{D}(p) \bigcup \mathcal{B}(p) \subset \mathcal{C}(p+1)$. Notons qu'il existe des permutations de $\mathcal{C}(p+1)$ qui ne sont pas dans $\mathcal{D}(p) \bigcup \mathcal{B}(p)$.

Sur la figure 4.18 (page 73) la remarque 4.1 est matérialisée par les schémas de permutations grisés : les permutations au niveau $p = 3$ trouvées à partir des chemins des branches (2) et (3) peuvent être obtenues au niveau $p = 4$ dans la branche (1).

Par exemple, prenons la permutation $\sigma = 3\,14\,2\,10\,4\,15\,5\,1\,7\,11\,9\,6\,8\,13\,12$:

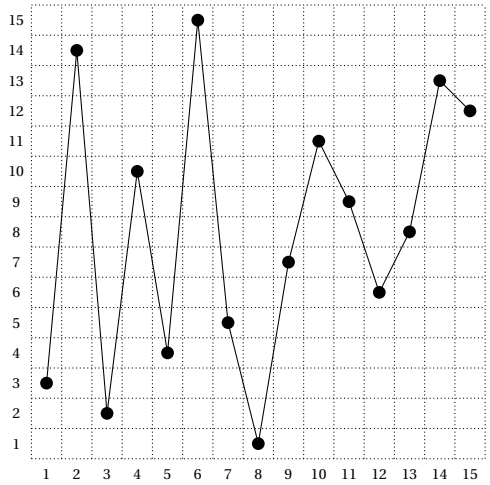


Figure 4.19 – Permutation $\sigma = 3\,14\,2\,10\,4\,15\,5\,1\,7\,11\,9\,6\,8\,13\,12$.

Dans la figure 4.19 on a $\sigma \in \mathcal{C}(4)$ mais $\sigma \notin \mathcal{D}(3)$ et $\sigma \notin \mathcal{B}(3)$.

Conséquence 4.1

✎ Soit σ une permutation avec k vallées. $\lceil \log_2(k+1) \rceil + 1$ étapes de WM-duplications sont suffisantes pour obtenir σ à partir de l'identité. □

Conséquence 4.2

✎ Soit σ une permutation avec k vallées. $\lceil \log_2(k+1) \rceil$ étapes de W- ou WM-duplications sont nécessaires pour obtenir σ à partir de l'identité. □

Ces deux conséquences nous permettent donc de dire que nos algorithmes présentés en 4.4 (page 61) permettent d'obtenir les permutations de $\mathcal{D}(p)$ en parcourant le plus court chemin possible modulo une étape supplémentaire dans certains cas. Le problème pour déterminer précisément $\mathcal{D}(p)$ reste ouvert, ainsi que l'élaboration d'un algorithme efficace pour tous les cas concernés.

Les différentes contributions présentées dans ce chapitre ont donné lieu à une présentation [13] lors de la conférence *Pattern Permutations* à Florence en 2009, ainsi qu'à un article [14] publié dans *Information Processing Letters* en 2010 (voir page 103).

Chapitre 5

Codes de Gray pour des classes restreintes de compositions et de permutations

Comme nous l'avons vu en 4.1 (page 49), la duplication n'est pas la seule méthode permettant d'obtenir un chemin entre deux génomes. Une autre opération existe, nommée l'*inversion* [2, 36, 82, 97]. Nous verrons en 5.3 (page 90) qu'il existe une bijection naturelle entre les permutations avec un nombre fixé d'inversions et les compositions bornées d'entiers. C'est pourquoi avant d'étudier ce cas précis, nous allons travailler sur les compositions d'entiers (5.2 page 80).

5.1 Introduction

La restriction d'une relation d'ordre $\prec$ (définie un peu plus tard dans cette section) à l'ensemble des compositions et des compositions bornées induit de nouveaux codes de Gray pour ces ensembles. Dans ce chapitre nous montrons que la relation d'ordre $\prec$ restreinte à l'ensemble *des compositions bornées d'un intervalle* fournit encore un code de Gray. L'ensemble des n-compositions bornées d'un intervalle généralise simultanément l'ensemble produit et l'ensemble des compositions d'un entier et donc la relation $\prec$ définit de façon unifiée tous ces codes de Gray.

Nous réexprimons les codes de Gray de Walsh et Knuth pour les compositions (bornées) d'un entier à l'aide d'une unique relation d'ordre. Alors, le code de Gray

de Walsh [100] devient une sous-liste de celui de Knuth, lequel est à son tour une sous-liste du code de Gray réfléchi [18, 52].

Basé sur cette relation d'ordre on généralise les codes de Gray de Knuth et Walsh aux compositions bornées d'un *intervalle* $[k, \ell]$. Nous appliquons ces résultats pour obtenir des codes de Gray pour les permutations avec un nombre d'inversions situé entre deux entiers ou avec un nombre pair (ou impair) d'inversions ou de cycles. De telles classes de permutations sont utilisées pour résoudre des problèmes difficiles de calculs, voir par exemple [19].

Définition 5.1 : miroir d'une liste

✎ Le *miroir* d'une liste $\mathscr{L}$, noté $\overline{\mathscr{L}}$, est la liste $\mathscr{L}$ lue de la fin vers le début. □

Pour un entier $m \in \mathbb{N}$, $[m]$ représente l'ensemble $\{0, 1, \ldots, m\}$ et pour un n-uplet $\boldsymbol{b} = (b_1, b_2, \ldots, b_n) \in \mathbb{N}^n$ on définit :

- $[\boldsymbol{b}]$ est l'ensemble produit $[b_1] \times [b_2] \times \cdots \times [b_n]$;
- $||\boldsymbol{b}||$ est la somme des composantes de $\boldsymbol{b}$, *i.e.* $||\boldsymbol{b}|| = \sum_{i=1}^{n} b_i$.

Le code de Gray binaire réfléchi pour l'ensemble produit $[\boldsymbol{b}]$, que l'on notera $\mathscr{G}_n(\boldsymbol{b})$, est l'extension naturelle du code de Gray binaire réfléchi appliqué à l'ensemble $[\boldsymbol{b}]$. $\mathscr{G}_n(\boldsymbol{b})$ a été défini récursivement par Er dans [41] par la relation ci-dessous et généré de façon *loopless* par Williamson dans [104, p. 112].

$$\mathscr{G}_n(\boldsymbol{b}) = \begin{cases} \lambda & \text{si} \quad n = 0, \\ 0\mathscr{G}_{n-1}(\boldsymbol{b}') \cup 1\overline{\mathscr{G}_{n-1}(\boldsymbol{b}')} \cup 2\mathscr{G}_{n-1}(\boldsymbol{b}') \cup \cdots \cup b_1\mathscr{G}'_{n-1}(\boldsymbol{b}') & \text{si} \quad n > 0. \end{cases}$$

où λ est la liste vide, $\boldsymbol{b}' = b_2 b_3 \ldots b_n$, $\overline{\mathscr{G}_{n-1}(\boldsymbol{b}')}$ est le miroir de $\mathscr{G}_{n-1}(\boldsymbol{b}')$ et $\mathscr{G}'_{n-1}(\boldsymbol{b}')$ est $\mathscr{G}_{n-1}(\boldsymbol{b}')$ ou $\overline{\mathscr{G}_{n-1}(\boldsymbol{b}')}$ suivant que b_1 est pair ou impair.

Dans $\mathscr{G}_n(\boldsymbol{b})$ deux tuples consécutifs diffèrent en une seule position et par $+1$ ou -1 sur cette position. On peut par exemple le voir sur la liste $\mathscr{G}_3(2\,1\,3)$ présentée dans la table 5.1 (page 79) (voir aussi la première colonne de la table 5.5 (page 90)).

000
001
002
003
013
012
011
010
110
111
112
113
103
102
101
100
200
201
202
203
213
212
211
210

Table 5.1 – Liste $\mathscr{G}_3$ (2 1 3).

On peut voir que la définition recursive précédente du code de Gray définit un ordre $\prec$ sur l'ensemble produit défini par : $\boldsymbol{x} = x_1 x_2 \dots x_n$ est inférieur à $\boldsymbol{y} = y_1 y_2 \dots y_n$ dans l'ordre $\prec$, et on le note par $\boldsymbol{x} \prec \boldsymbol{y}$, si

- $\sum_{j=1}^{i-1} x_j$ est pair et $x_i < y_i$, ou bien
- $\sum_{j=1}^{i-1} x_j$ est impair et $x_i > y_i$,

où i est la position la plus à gauche telle que $x_i \neq y_i$.

Définition 5.2 : *first* et *last*

✎ Pour $A \subset \mathbb{N}^n$, $first(A)$ et $last(A)$ sont respectivement le *premier* et le *dernier* tuple (dans l'ordre $\prec$) de l'ensemble A. □

Il n'existe donc pas de $\boldsymbol{x} \in A$ tel que $\boldsymbol{x} \prec first(A)$ ou $last(A) \prec \boldsymbol{x}$ ($\boldsymbol{x} \neq first(A)$ et $\boldsymbol{x} \neq last(A)$).

Définition 5.3 : successeur

✎ Pour $\boldsymbol{c} \in [\boldsymbol{b}]$, $\text{succ}^{\boldsymbol{b}}(\boldsymbol{c})$ est le *successeur* de $\boldsymbol{c}$ (dans l'ordre $\prec$) de l'ensemble produit $[\boldsymbol{b}]$. □

Si on a $\boldsymbol{c} \in [\boldsymbol{b}]$, on ne pourra donc pas trouver de $\boldsymbol{x} \in [\boldsymbol{b}]$ tel que $\boldsymbol{c} \prec \boldsymbol{x} \prec \text{succ}^{\boldsymbol{b}}(\boldsymbol{c})$ ($\boldsymbol{x} \neq \boldsymbol{c}$ et $\boldsymbol{x} \neq \text{succ}^{\boldsymbol{b}}(\boldsymbol{c})$).

Dans la suite et sauf si on l'explicite clairement d'une autre manière, le successeur d'un tuple, le premier et le dernier tuples dans un ensemble sont considérés avec la relation d'ordre $\prec$. Un ensemble ainsi considéré est donc supposé « ordonné » selon la relation d'ordre $\prec$.

On notera également $\mathscr{A}$ la *liste* ordonnée (dans l'ordre $\prec$) des tuples de l'ensemble A.

5.2 Compositions bornées d'un intervalle

On définit la notion de *n-composition $\boldsymbol{b}$-bornée d'un intervalle* $[k,\ell]$ de la manière suivante :

Définition 5.4

✎ Pour trois entiers $n \in \mathbb{N}$, $k \in \mathbb{Z}$ et $\ell \in \mathbb{N}$ avec $k \leqslant \ell$ et un n-uplet $\boldsymbol{b} = b_1 b_2 \dots b_n \in \mathbb{N}^n$, une *$n$-composition $\boldsymbol{b}$-bornée de l'intervalle* $[k,\ell]$ est un n-uplet $\boldsymbol{c} = c_1 c_2 \dots c_n \in \mathbb{N}^n$ tel que
- $k \leqslant \sum_{i=1}^{n} c_i \leqslant \ell$ et
- $0 \leqslant c_i \leqslant b_i$, pour $0 \leqslant i \leqslant n$. □

On note $W_{k,\ell}^{\boldsymbol{b}}$ l'ensemble des n-compositions $\boldsymbol{b}$-bornées de l'intervalle $[k,\ell]$. Bien evidement $W_{k,k}^{\boldsymbol{b}}$ est l'ensemble des compositions bornées de k et on a l'égalité sui-

vante : $W^{\boldsymbol{b}}_{k,\ell} = \bigcup_{i=k}^{\ell} W^{\boldsymbol{b}}_{i,i}$.

Pour $\boldsymbol{c} \in W^{\boldsymbol{b}}_{k,\ell}$, $\mathrm{succ}^{\boldsymbol{b}}_{k,\ell}(\boldsymbol{c})$ sera le successeur de $\boldsymbol{c}$ (en ordre $\prec$) dans l'ensemble $W^{\boldsymbol{b}}_{k,\ell}$.

Notons que dans la suite du chapitre nous omettrons les exposants $\boldsymbol{b}$ lorsque cela ne crée pas d'ambiguïté.

Remarquons que dans la définition précédente, k peut être un nombre négatif ; Dans ce cas $W_{k,\ell} = W_{0,\ell}$. De façon similaire, $W_{k,\ell} = W_{k,||\boldsymbol{b}||}$ si $\ell \geqslant ||\boldsymbol{b}||$.

Remarque 5.1

✎ On obtient comme cas particulier :

- $W_{0,||\boldsymbol{b}||}$ est l'ensemble produit $[\boldsymbol{b}]$; cet ensemble a été généré de façon *loopless* par Williamson dans [104, p. 112],
- $W_{k,k}$ est l'ensemble des n-compositions $\boldsymbol{b}$-bornées de k généré de façon *loopless* par Walsh dans [100],
- si $\boldsymbol{b} = [k]^n$, $W_{k,k}$ est l'ensemble des n-compositions de k sans restriction, défini formellement par Knuth, récursivement par Nijenhuis et Wilf [75] et implémenté itérativement par Klingsberg [61]. □

Sur la table 5.2 (page 82), on retrouve : en (1) l'ensemble produit $W_{0,||\boldsymbol{b}||}$; en (2) l'ensemble des 3-compositions de k sans restriction, *i.e.* $W_{k,k}$ avec $\boldsymbol{b} = 333$; en (3), (4) et (5) les ensembles des 3-compositions $\boldsymbol{b}$-bornées de k avec $\boldsymbol{b} = 312$ et $k = \{4,5,6\}$; et en (6) le cas général, *i.e.* $W_{k,\ell}$ avec $\boldsymbol{b} = 312$.

①	②	③	④	⑤	⑥
$\boldsymbol{b} = 333$	$\boldsymbol{b} = 333$	$\boldsymbol{b} = 312$	$\boldsymbol{b} = 312$	$\boldsymbol{b} = 312$	$\boldsymbol{b} = 312$
$k = 0$	$k = 4$	$k = 4$	$k = 5$	$k = 6$	$k = 4$
$\ell = 9$	$\ell = 4$	$\ell = 4$	$\ell = 5$	$\ell = 6$	$\ell = 6$
000					
001					
002					
003					
013	013				
012					
011					
010					
020					
021					
022	022				
023					
033					
032					
031	031				
030					
130	130				
131					
132					
133					
123					
122					
121	121				
120					
110					
111					
112	112	112			112
113					
103	103				
102					
101					
100					
200					
201					
202	202	202			202
203					
213					
212			212		212
211	211	211			211
210					
220	220	220			
221					
222					
223					
233					
232					
231					
230					
330					
331					
332					
333					
323					
322					
321					
320					
310	310	310			310
311			311		311
312				312	312
313					
303					
302			302		302
301	301	301			301
300					

Table 5.2 – Exemples de 3-compositions en fonction des contraintes fixées.

Il peut se produire que $first(W_{k,\ell})$ soit égal à $first(W_{k+1,\ell+1})$. Ce cas apparaît seulement si $00\ldots0$ appartient à la fois à $W_{k,\ell}$ et à $W_{k+1,\ell+1}$, et donc si $k+1 \leqslant 0$.

De manière similaire, il peut arriver que $last(W_{k,\ell})$ soit égal à $last(W_{k+1,\ell+1})$ et ce cas apparaît seulement si $last([\boldsymbol{b}])$ appartient à la fois à $W_{k,\ell}$ et à $W_{k+1,\ell+1}$.

Formellement, on obtient le lemme 5.1.

Lemme 5.1

✎ On a :

1. $first(W_{k+1,\ell+1})$ et $first(W_{k,\ell})$ sont soit égaux tous les deux à $00\ldots0$ soit ils diffèrent en une seule position et par 1 sur cette position ;
2. $last(W_{k+1,\ell+1})$ et $last(W_{k,\ell})$ sont soit égaux tous les deux à $last([\boldsymbol{b}])$ soit ils diffèrent en une seule position et par 1 sur cette position ;
3. si $\boldsymbol{s} = last(W_{k+1,\ell+1})$ diffère de $\boldsymbol{t} = last(W_{k,l})$ sur la position i, alors on a $s_j = t_j \in \{0, b_j\}$ pour tout $i \neq j$;
4. $last(W_{k,\ell}) \in \{last(W_{k,k}), last(W_{\ell,\ell}), last([\boldsymbol{b}])\}$. □

Démonstration

Point 1. $first(W_{k,\ell})$ est le plus petit (lexicographiquement) tuple $\boldsymbol{c}$ appartenant à $[\boldsymbol{b}]$ avec $||\boldsymbol{c}|| = \max\{0, k\}$ et $first(W_{k+1,\ell+1})$ le plus petit (lexicographiquement) tuple $\boldsymbol{d}$ appartenant à $[\boldsymbol{b}]$ avec $||\boldsymbol{d}|| = \max\{0, k+1\}$ et le résultat se vérifie.

On peut observer ce point sur l'exemple de la table 5.2 (page 82) : les premiers tuples des colonnes (3), (4) et (5) sont 112, 212 et 312. On a bien une seule position différente entre chaque et la valeur modifiée change seulement de 1.

Point 2. Supposons que $last(W_{k,\ell}) \neq last([\boldsymbol{b}])$ (c'est à dire que, $last([\boldsymbol{b}]) \notin W_{k,\ell}$). Le cas $last(W_{k+1,\ell+1}) \neq last([\boldsymbol{b}])$ (c'est à dire, $last([\boldsymbol{b}]) \notin W_{k+1,\ell+1}$) est similaire.

• Si $0 \leqslant \ell < b_1$, alors

$$last(W_{k,\ell}) = \ell 0\ldots0$$

et

$$last(W_{k+1,\ell+1}) = (\ell+1)0\ldots0$$

et l'assertion est vérifiée.

• Si $\ell \geqslant b_1$ et b_1 est impair, alors

$$last\big(W_{k,\ell}\big) = b_1 \cdot first\big(W_{k-b_1,\ell-b_1}\big)$$

et

$$last\big(W_{k+1,\ell+1}\big) = b_1 \cdot first\Big(W^{\boldsymbol{b}'}_{k+1-b_1,\ell+1-b_1}\Big)$$

avec $\boldsymbol{b}' = b_2 b_3 \dots b_n$.

Mais $first\Big(W^{\boldsymbol{b}'}_{k-b_1,\ell-b_1}\Big) \neq 00\dots0$, sinon $last\big(W_{k,\ell}\big) = last([\boldsymbol{b}])$ et grâce au premier point de ce lemme l'assertion se vérifie.

• Si $k \geqslant b_1$ et b_1 est pair, alors

$$last\big(W_{k,\ell}\big) = b_1 \cdot last\Big(W^{\boldsymbol{b}'}_{k-b_1,\ell-b_1}\Big)$$

et

$$last\big(W_{k+1,\ell+1}\big) = b_1 \cdot last\Big(W^{\boldsymbol{b}'}_{k+1-b_1,\ell+1-b_1}\Big).$$

Mais $last\Big(W^{\boldsymbol{b}'}_{k-b_1,\ell-b_1}\Big) \neq last\big([\boldsymbol{b}']\big)$, sinon $last\big(W_{k,\ell}\big) = last([\boldsymbol{b}])$ et une récurrence sur n complète la preuve.

On peut également observer ce point sur l'exemple de la table 5.2 (page 82) : les derniers tuples des colonnes (3), (4) et (5) sont 301, 302 et 312. On a bien une seule position différente entre chaque et la valeur modifiée change seulement de 1.

Points 3 et 4. Ces deux points sont des conséquences de la preuve du point 2. ■

Si $\boldsymbol{c} \in W_{k,\ell}$ et $\mathrm{succ}(\boldsymbol{c}) \in W_{k,\ell}$, alors $\mathrm{succ}_{k,\ell}(\boldsymbol{c}) = \mathrm{succ}(\boldsymbol{c})$; sinon, la proposition 5.1 (page 86) dit que $\mathrm{succ}_{k,\ell}(\boldsymbol{c}) = \mathrm{succ}_{p,p}(\boldsymbol{c})$ avec $p = ||\boldsymbol{c}||$. Avant de prouver cette proposition on présente un lemme technique.

Lemme 5.2

✎ Soient $\boldsymbol{c}, \boldsymbol{c}' \in W_{k,\ell}$ avec $\boldsymbol{c}' = \mathrm{succ}_{k,\ell}(\boldsymbol{c})$ et u la position la plus à gauche où $c'_i \neq c_i$. Alors $c'_u = c_u + 1$ ou $c'_u = c_u - 1$. □

Démonstration

Soit $I \subset \{1, 2, \dots n\}$ l'ensemble des indices i avec $c'_i \neq c_i$. Alors u est l'élément minimal de I et supposons que $c'_u = c_u + p$ pour un $p > 1$; le cas où $c'_u = c_u - p$ se traite de façon similaire.

• Si $c'_i > c_i$ pour tout $i \in I$, alors le tuple $\boldsymbol{c}''$ défini par

$$c''_i = \begin{cases} c_i & \text{si} \quad i \neq u, \\ c_i + 1 & \text{si} \quad i = u, \end{cases}$$

appartient à $W_{k,\ell}$ et est tel que $\boldsymbol{c} \prec \boldsymbol{c}'' \prec \boldsymbol{c}'$ (effectivement, dans ce cas $\boldsymbol{c}'' = \mathrm{succ}(\boldsymbol{c})$). Ceci est en contradiction avec $\boldsymbol{c}' = \mathrm{succ}_{k,\ell}(\boldsymbol{c})$.

• S'il existe un $v \in I$, $v > u$, tel que $c'_v = c_v - r$ pour un $r \geqslant 1$, alors le tuple $\boldsymbol{c}''$ défini par

$$c''_i = \begin{cases} c_i & \text{si} \quad i \neq u, v, \\ c_i + (p-1) & \text{si} \quad i = u, \\ c_i - (r-1) & \text{si} \quad i = v, \end{cases}$$

appartient à $W_{k,\ell}$ et est tel que $\boldsymbol{c} \prec \boldsymbol{c}'' \prec \boldsymbol{c}'$, ce qui fournit encore une contradiction. ■

…
2021022
2021021
2021020
2021010
2021011
2021012
2021002
2021001
2021000
2022000
2022001
2022002
2022012
2022011
2022010
2022020
…

Table 5.3 – Un extrait de l'ensemble ordonné $W_{3,9}$ avec $\boldsymbol{b} = 3124532$.

Prenons en exemple (table 5.3) un extrait de l'ensemble ordonné $W_{3,9}$ avec $\boldsymbol{b} =$

3124532 : on constate que sur la position la plus à gauche qui diffère d'une ligne à l'autre (chiffres soulignés), la valeur ne change que de 1, en plus ou en moins.

Proposition 5.1

✎ Soit $\boldsymbol{c}, \boldsymbol{c}' \in W_{k,\ell}$ avec $\boldsymbol{c}' = \mathrm{succ}_{k,\ell}(c)$. Alors l'un des deux points ci-dessous est vérifié :

- $\boldsymbol{c}' = \mathrm{succ}(\boldsymbol{c})$ et alors $\boldsymbol{c}$ et $\boldsymbol{c}'$ diffèrent en une seule position et par $+1$ ou -1 sur cette position ;
- $||\boldsymbol{c}|| = ||\boldsymbol{c}'||$ et $\boldsymbol{c}$ et $\boldsymbol{c}'$ diffèrent en deux positions et par $+1$ ou -1 sur ces positions. □

Démonstration

Soit u la position la plus à gauche où $\boldsymbol{c}$ diffère de $\boldsymbol{c}'$. Par le lemme 5.2 (page 84), $c'_u = c_u + \alpha$, avec $\alpha \in \{-1, 1\}$.

Si $\beta = \sum_{i=1}^{u} c_i$ est impair, alors

$$\boldsymbol{c} = c_1 c_2 \dots c_u \mathit{first}\left(W^{\boldsymbol{b}'}_{k-\beta,\ell-\beta}\right)$$

et

$$\boldsymbol{c}' = c_1 c_2 \dots (c_u + \alpha) \mathit{first}\left(W^{\boldsymbol{b}'}_{k+\alpha-\beta,\ell+\alpha-\beta}\right).$$

avec $\boldsymbol{b}' = b_{u+1} b_{u+2} \dots b_n$.

Dans ce cas, grâce au premier point du lemme 5.1 (page 83), on a :

- $\mathit{first}\left(W^{\boldsymbol{b}'}_{k-\beta,\ell-\beta}\right) = \mathit{first}\left(W^{\boldsymbol{b}'}_{k+\alpha-\beta,\ell+\alpha-\beta}\right) = 00\dots0$, et alors $\boldsymbol{c}' = \mathrm{succ}(\boldsymbol{c})$ ou
- $\mathit{first}\left(W^{\boldsymbol{b}'}_{k-\beta,\ell-\beta}\right)$ diffère de $\mathit{first}\left(W^{\boldsymbol{b}'}_{k+\alpha-\beta,\ell+\alpha-\beta}\right)$ en une seule position et par $-\alpha$ sur cette position.

Si $\beta = \sum_{i=1}^{u} c_i$ est pair, alors

$$\boldsymbol{c} = c_1 c_2 \dots c_u \mathit{last}\left(W^{\boldsymbol{b}'}_{k-\beta,\ell-\beta}\right)$$

et

$$\boldsymbol{c}' = c_1 c_2 \dots (c_u + \alpha)\, \mathit{last}\left(W^{\boldsymbol{b}'}_{k+\alpha-\beta,\ell+\alpha-\beta}\right).$$

Maintenant, en appliquant le second point du lemme 5.1 (page 83) le résultat se vérifie. ■

...
2020300
2020400
2020401
2020410
2020500
2021400
2021300
2021301
2021310
...

Table 5.4 – Un extrait de l'ensemble ordonné $W_{3,9}$ avec $\boldsymbol{b} = 3124532$.

L'exemple de la table 5.4 représente un extrait de l'ensemble ordonné $W_{3,9}$ avec $\boldsymbol{b} = 3124532$: on constate que s'il y a deux positions qui diffèrent d'une ligne à l'autre, c'est bien systématiquement sur deux positions adjacentes et l'une est augmentée de 1, l'autre diminuée d'autant.

Une conséquence de la proposition précédente et de sa preuve est la suivante :

Corollaire 5.1

✎ Soient $\boldsymbol{c} \in W_{k,\ell}$ et $\boldsymbol{c}' = \mathrm{succ}_{k,\ell}(\boldsymbol{c})$:

1. si $\mathrm{succ}(\boldsymbol{c}) \in W_{k,\ell}$, alors $\boldsymbol{c}' = \mathrm{succ}(\boldsymbol{c})$;
2. si $\mathrm{succ}(\boldsymbol{c}) \notin W_{k,\ell}$, alors $\boldsymbol{c}' = \mathrm{succ}_{p,p}(\boldsymbol{c})$ avec $p = ||\boldsymbol{c}||$. Dans ce cas $p = k$ ou $p = \ell$ et si u et v, $u < v$, sont les deux positions où $\boldsymbol{c}$ et $\boldsymbol{c}'$ diffèrent, alors $c_i = c'_i \in \{0, b_i\}$ pour tout $i > u$, $i \neq v$.

□

En combinant la proposition 5.1 (page 86) et le corollaire 5.1, on obtient le théorème 5.1 :

Théorème 5.1

✎ La liste $\mathscr{W}_{k,\ell}$ des tuples dans l'ensemble $W_{k,\ell}$ listé dans l'ordre $\prec$ est un code de Gray pour cet ensemble. En particulier, $\mathscr{W}_{k,k}$ est un code de Gray pour les compositions $\boldsymbol{b}$-bornées de l'entier k. □

On pourra aisément observer cela sur la table 5.2 (page 82).

Un autre code de Gray pour $W_{k,\ell}$ est donné par le corollaire 5.2 :

Corollaire 5.2

✎ Pour $k \leqslant \ell \leqslant ||\boldsymbol{b}||$, la liste

$$\mathcal{W}_{k,k}, \overline{\mathcal{W}_{k+1,k+1}}, \mathcal{W}_{k+2,k+2}, \ldots, \mathcal{W}'_{\ell,\ell}$$

est un code de Gray pour l'ensemble $W_{k,\ell}$, où $\mathcal{W}'_{\ell,\ell}$ est $\mathcal{W}_{\ell,\ell}$ ou $\overline{\mathcal{W}_{\ell,\ell}}$ suivant que $\ell - k + 1$ est impair ou pair. □

Démonstration

Par le théorème 5.1 (page 87), pour tout i, $k \leqslant i \leqslant \ell$, $\mathcal{W}_{i,i}$ est un code de Gray pour $W_{i,i}$. Le dernier tuple de $W_{i,i}$ est $last\big(W_{i,i}\big)$ et le premier tuple de $\overline{W_{i+1,i+1}}$ est $last\big(W_{i+1,i+1}\big)$. Et par le lemme 5.1 (page 83) (2) ils diffèrent en une seule position et par 1 sur cette position. De façon similaire, par le lemme 5.1 (page 83) (1), le dernier tuple de $\overline{W_{i,i}}$ et le premier tuple de $W_{i+1,i+1}$ diffèrent de la même manière. ■

Par exemple, pour $\boldsymbol{b} = 44 \in \mathbb{N}^2$, nous avons les listes en code de Gray suivantes pour l'ensemble $W_{3,4}$:

- $\mathcal{W}_{3,4} = 03, 04, 13, 12, 21, 22, 31, 30, 40$ et
- $\mathcal{W}_{3,3}, \overline{\mathcal{W}_{4,4}} = 03, 12, 21, 30, 40, 31, 22, 13, 04$.

La première liste comporte 3 transitions de taille 2 ; dans la seconde liste nous avons 7 transitions de taille 2. Cependant, la liste $04, 03, 13, 12, 22, 21, 31, 30, 40$ pour le même ensemble est plus restrictive puisqu'il n'y a aucune transition de taille 2, on pourrait donc la considérer comme « plus optimale ». Malgré tout, l'existence de cette configuration optimale dans le cas général demeure un problème ouvert.

Comme mentionné précédemment, l'ensemble $W_{k,\ell}$ généralise l'ensemble produit, les compositions sans restriction et les compositions bornées d'un entier. La remarque 5.2 (page 89) dit que cela reste vrai dans le contexte où les ensembles sont

ordonnés.

Remarque 5.2

✎ Comme cas particuliers on a :
- $\mathscr{W}_{0,||\boldsymbol{b}||}$ est le code de Gray réfléchi, $\mathscr{G}_n(\boldsymbol{b})$, pour l'ensemble produit $[\boldsymbol{b}]$ défini par Er [41] et généré de façon *loopless* par Williamson dans [104, p. 112] ;
- si $\boldsymbol{b} = [k]^n$, alors $\mathscr{W}_{k,k}$ est le code de Gray de Knuth [61, 75, 104] pour les n-compositions de k sans restriction ;
- $\mathscr{W}_{k,k}$ devient le code de Gray de Walsh pour les n-compositions $\boldsymbol{b}$-bornées de k, définies et générées de façon *loopless* dans [100]. □

Pour deux listes $\mathscr{A}$ et $\mathscr{B}$, $\mathscr{A} \subset \mathscr{B}$ signifie que $\mathscr{A}$ est une sous-liste de $\mathscr{B}$ (éventuellement non connexe) ; dans ce cas la sous-liste correspondante vérifie $A \subset B$.

Avec ces notations on obtient la remarque 5.3 :

Remarque 5.3

✎ $W^{\boldsymbol{b}}_{u,v} \subset W^{\boldsymbol{c}}_{k,\ell}$ si $[u,v]$ est un sous intervalle de $[k,\ell]$ et $\boldsymbol{b}$ est plus petit (au sens large) que $\boldsymbol{c}$ (chaque composante de $\boldsymbol{b}$ est plus petite au sens large que la composante correspondante dans $\boldsymbol{c}$). □

En particulier, $\mathscr{W}^{\boldsymbol{b}}_{k,k} \subset \mathscr{W}^{\boldsymbol{b}}_{k,\ell} \subset \mathscr{W}^{\boldsymbol{b}}_{0,||\boldsymbol{b}||} = \mathscr{G}_n(\boldsymbol{b})$.

$\mathscr{G}_n(\boldsymbol{b})$	$\mathscr{W}_{2,4}$		
	$\mathscr{W}_{2,3}$		$\mathscr{W}_{4,4}$
	$\mathscr{W}_{2,2}$	$\mathscr{W}_{3,3}$	
000			
001			
002	✓		
003		✓	
013			✓
012		✓	
011	✓		
010			
110	✓		
111		✓	
112			✓
113			
103			✓
102		✓	
101	✓		
100			
200	✓		
201		✓	
202			✓
203			
213			
212			
211			✓
210		✓	

Table 5.5 – Les ensembles $[\boldsymbol{b}]$, $W_{2,2}$, $W_{3,3}$, $W_{2,3}$, $W_{4,4}$, $W_{2,4}$ listés dans l'ordre $\prec$ pour $n = 3$ et $\boldsymbol{b} = 2\,1\,3 \in \mathbb{N}^3$.

5.3 Permutations restreintes

Il y a une correspondance naturelle (voir la table 5.6 (page 92)) entre l'ensemble produit $[0] \times [1] \times \cdots \times [n-1]$ et l'ensemble S_n des permutations de longueur n. Soient $\mathscr{G}_n([0] \times [1] \times \cdots \times [n-1])$ le code de Gray défini précédemment pour l'ensemble

produit $[0] \times [1] \times \cdots \times [n-1]$ et $\mathscr{S}_n$ sa *liste image* à travers cette correspondance. Dans le lemme 5.4 (page 93) il est dit que $\mathscr{S}_n$ est un code de Gray pour S_n, lequel est effectivement le code de Gray bien connu de Johnson-Trotter [57, 87, 91] pour les permutations (voir 2.3 (page 34)).

De plus, la proposition 5.2 (page 94) et la proposition 5.3 (page 95) nous disent que la restriction de $\mathscr{S}_n$ à certaines classes particulières de permutations fournit encore des codes de Gray.

Le tableau $\boldsymbol{t} = t_1 t_2 \dots t_n \in [0] \times [1] \times \cdots \times [n-1]$ est la *table d'inversion* (voir [85, p. 20]) d'une permutation $\tau \in S_n$ si, pour tout i avec $1 \leqslant i \leqslant n$,

$$t_i = \text{le nombre d'éléments de } \tau \text{ plus petits que } i \text{ et à sa droite.}$$

De manière évidente, $||\boldsymbol{t}||$ est le nombre d'inversions dans la permutation τ, noté $\operatorname{inv} \tau$; c'est aussi le nombre de transpositions adjacentes nécessaires pour classer la permutation τ.

Pour chaque $n \in \mathbb{N}$, la fonction

$$\phi : [0] \times [1] \times \cdots \times [n-1] \to S_n$$

définie par $\tau = \phi(\boldsymbol{t})$ où $\boldsymbol{t}$ est la table d'inversions de τ, est une bijection de

$$[0] \times [1] \times \cdots \times [n-1]$$

vers S_n.

La table 5.6 (page 92) tirée de [100, p. 331] permet de voir clairement la correspondance entre les compositions $\boldsymbol{b}$-bornées (avec $\boldsymbol{b} = 01234$) de l'entier 5 d'une part et les permutations de longueur 5 avec 5 inversions d'autre part.

01004	52134
00230	43125
00203	35124
01103	25314
01220	34215
01130	42315
00104	51324
00032	41523
01202	32514
01022	24513
00023	15423
00122	14532
01112	23541
00212	31542
00131	41352
01211	32451
01121	24351
00113	15342
01013	25143
00221	34152
01031	42153
00014	51243

Table 5.6 – Correspondance entre les 5-compositions $\boldsymbol{b}$-bornées par $\boldsymbol{b}$ = 01234 de l'entier 5 et les permutations de longueur 5 avec 5 inversions.

Si deux permutations diffèrent par une transposition adjacente, alors leurs tables d'inversions diffèrent en une position et par $+1$ ou -1 sur cette position. Réciproquement, il est facile de voir que si deux tables d'inversions diffèrent en une position et par $+1$ ou -1 sur cette position, alors leurs permutations correspondantes diffèrent par une transposition de deux éléments, lesquels ne sont pas nécessairement adjacents. Par exemple, $\boldsymbol{s}$ = 011032 diffère de $\boldsymbol{t}$ = 001032 en une seule position et 253614 = $\phi(\boldsymbol{s})$ diffère de 153624 = $\phi(\boldsymbol{t})$ par une transposition non adjacente. Cependant, avec des contraintes additionnelles, la propriété d'adjacence peut être

préservée.

Lemme 5.3

✎ Soient $\boldsymbol{s}, \boldsymbol{t} \in [0] \times [1] \times [2] \times [n-1]$. S'il y a un $i \in \{1, 2, \ldots, n\}$ avec :
- $t_i = s_i + 1$ et $s_j = t_j$ pour tout $j \neq i$,
- $s_j = t_j \in \{0, j-1\}$ pour tout $j > i$,

alors $\sigma = \phi(\boldsymbol{s})$ et $\tau = \phi(\boldsymbol{t})$ diffèrent par une transposition adjacente. □

Démonstration

Si $i = n$, alors n n'est pas sur la position la plus à gauche dans σ et τ est obtenue de σ en transposant n et l'élément juste à sa gauche.

Si $i \neq n$, alors n est l'élément le plus à gauche ou le plus à droite de σ suivant que s_n vaut $n-1$ ou 0 et généralement, n'importe quel j avec $j > i$ est l'élément le plus à gauche ou le plus à droite de la permutation obtenue de σ en supprimant tous les éléments plus grands que j. Alors, σ est de la forme

$$\sigma = \underbrace{\sigma_1\sigma_2\ldots\sigma_u}_{>i}\underbrace{\sigma_{u+1}\sigma_{u+2}\ldots\sigma_{u+i}}_{\leqslant i}\underbrace{\sigma_{u+i+1}\sigma_{u+i+2}\ldots\sigma_n}_{>i}$$

avec $\sigma_{u+1}\sigma_{u+2}\ldots\sigma_{u+i}$ une permutation dans S_i avec la table d'inversions $s_1 s_2 \ldots s_i$. Puisque $s_i \neq i-1$, i n'est pas l'élément le plus à gauche de $\sigma_{u+1}\sigma_{u+2}\cdots\sigma_{u+i}$.
Et, comme dans le cas $i = n$, τ est obtenue de σ en transposant i avec l'élément juste à sa gauche. ■

Nous disons que deux permutations diffèrent en une *transposition adjacente* si on peut obtenir l'une à partir de l'autre en transposant deux éléments adjacents.

Par le lemme 5.3 et le lemme 5.1 (page 83) (3) on a :

Lemme 5.4

✎ Si $\boldsymbol{s}$ et $\boldsymbol{t}$ sont deux tuples successifs dans $\mathscr{G}_n([0] \times [1] \times \cdots \times [n-1])$ alors les permutations $\phi(\boldsymbol{s})$ et $\phi(\boldsymbol{t})$ dans S_n diffèrent par une transposition adjacente et donc $\mathscr{S}_n$ est un code de Gray pour S_n. □

Effectivement $\mathscr{S}_n$ est donc le code de Gray classique de Johnson-Trotter pour

l'ensemble des permutations de longueur n [57, 87, 91] (voir également 2.3 (page 34)).

Proposition 5.2

✎ La restriction de $\mathscr{S}_n$ à l'ensemble des permutations avec un nombre d'inversions situé entre deux entiers est un code de Gray où deux permutations consécutives diffèrent par une ou deux transpositions adjacentes. □

Démonstration

Soient σ et τ deux permutations consécutives dans la restriction de $\mathscr{S}_n$ à l'ensemble des permutations avec un nombre d'inversions situé entre k et ℓ inclus. Soient $\boldsymbol{s}$ et $\boldsymbol{t}$ les tuples correspondants dans $\boldsymbol{b} = [0] \times [1] \times \cdots \times [n-1]$ avec $\phi(\boldsymbol{s}) = \sigma$ et $\phi(\boldsymbol{t}) = \tau$. Par définition de $\mathscr{S}_n$, $\boldsymbol{s}$ et $\boldsymbol{t}$ sont consécutifs dans $\mathscr{W}^{\boldsymbol{b}}_{k,\ell}$. Alors, par la proposition 5.1 (page 86) on a soit : $\boldsymbol{t} = \text{succ}^{\boldsymbol{b}}(\boldsymbol{s})$ et dans ce cas, par le lemme 5.4 (page 93), σ diffère de τ par une transposition adjacente ; soit $\boldsymbol{s}$ et $\boldsymbol{t}$ diffèrent en deux positions, notées u et v avec $u < v$ et par $+1$ et -1 sur ces deux positions. Dans ce dernier cas et suivant le corollaire 5.1 (page 87), $s_i = t_i \in \{0, i-1\}$ pour tout $i > u$ et $i \neq v$. Nous montrerons que σ et τ diffèrent en deux transpositions adjacentes.

Soient σ' et π' les permutations de S_{v-1} avec, comme tableaux de transpositions : $s_1 s_2 \dots s_{v-1}$ et $t_1 t_2 \dots t_{v-1}$. Par le lemme 5.3 (page 93), σ' et π' diffèrent par une transposition adjacente. Maintenant, les permutations σ'' et π'' dans S_v avec les tableaux de transpositions $s_1 s_2 \dots s_{v-1} s_v$ et $t_1 t_2 \dots t_{v-1} t_v$ diffèrent par deux transpositions adjacentes. En effet, σ'' est obtenue de σ' en insérant v dans la $s_v{}^{\text{ème}}$ position considérée de la droite vers la gauche et τ'' est la permutation obtenue à partir de τ' en insérant v dans la $t_v{}^{\text{ème}}$ position de la droite vers la gauche et la position la plus à droite étant la position 0. Maintenant, puisque $s_i = t_i \in \{0, i-1\}$ pour $i > v$, il résulte que σ et τ diffèrent comme σ'' et τ'', c'est à dire en deux transpositions adjacentes. ■

La proposition 5.2 dit que deux permutations consécutives dans la restriction de $\mathscr{S}_n$ à l'ensemble des permutations avec un nombre d'inversions compris entre deux entiers donnés différent en au plus quatre positions. Remarquons qu'en général il n'y a pas de code de Gray plus restrictif pour cet ensemble. En effet, le graphe de la figure 5.1 (page 95) n'est pas Hamiltonien.

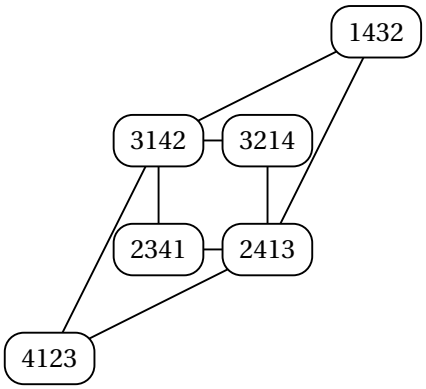


Figure 5.1 – Le graphe avec S_4 comme ensemble des sommets.

Une permutation est dite *paire* (resp. *impaire*) si elle a un nombre pair (resp. impair) d'inversions. L'ensemble des permutations paires forme un sous-groupe de S_n noté A_n ; on l'appelle le groupe alterné et sa cardinalité est $\frac{n!}{2}$.

Proposition 5.3

✎ La restriction de $\mathscr{S}_n$ aux ensembles suivants fournit un code de Gray où deux permutations consécutives diffèrent par deux transpositions adjacentes :

1. l'ensemble des permutations paires ;
2. l'ensemble des permutations impaires ;
3. l'ensemble des permutations avec un nombre pair de cycles ;
4. l'ensemble des permutations avec un nombre impair de cycles. □

Démonstration

Pour les points 1 et 2 la preuve est basée sur la remarque suivante : si $\boldsymbol{s}$ est le successeur de $\boldsymbol{r}$ et $\boldsymbol{t}$ celui de $\boldsymbol{s}$, dans la liste $\mathscr{G}_n([0] \times [1] \times \cdots \times [n-1])$, alors $||\boldsymbol{r}||$ et $||\boldsymbol{t}||$ ont la même parité. Les permutations $\phi(\boldsymbol{r})$ et $\phi(\boldsymbol{t})$ diffèrent par deux transpositions adjacentes et ont la même parité.

Pour les points 3 et 4 la preuve est similaire à celle des points 1 et 2 et est basée sur la remarque suivante : une transposition (pas nécessairement adjacente) dans une permutation regroupe deux cycles en un seul cycle ou scinde un cycle en deux cycles. ■

5.4 Algorithme

Notre étude n'a pas permis de trouver un algorithme efficace offrant la possibilité de générer la liste des permutations avec un nombre borné d'inversions ou son pendant, à savoir les compositions $\boldsymbol{b}$-bornées d'un intervalle $[k,\ell]$. En revanche, nous pouvons utiliser un algorithme de rejet en combinant les méthodes « successeurs » de Williamson [104] et de Walsh [100].

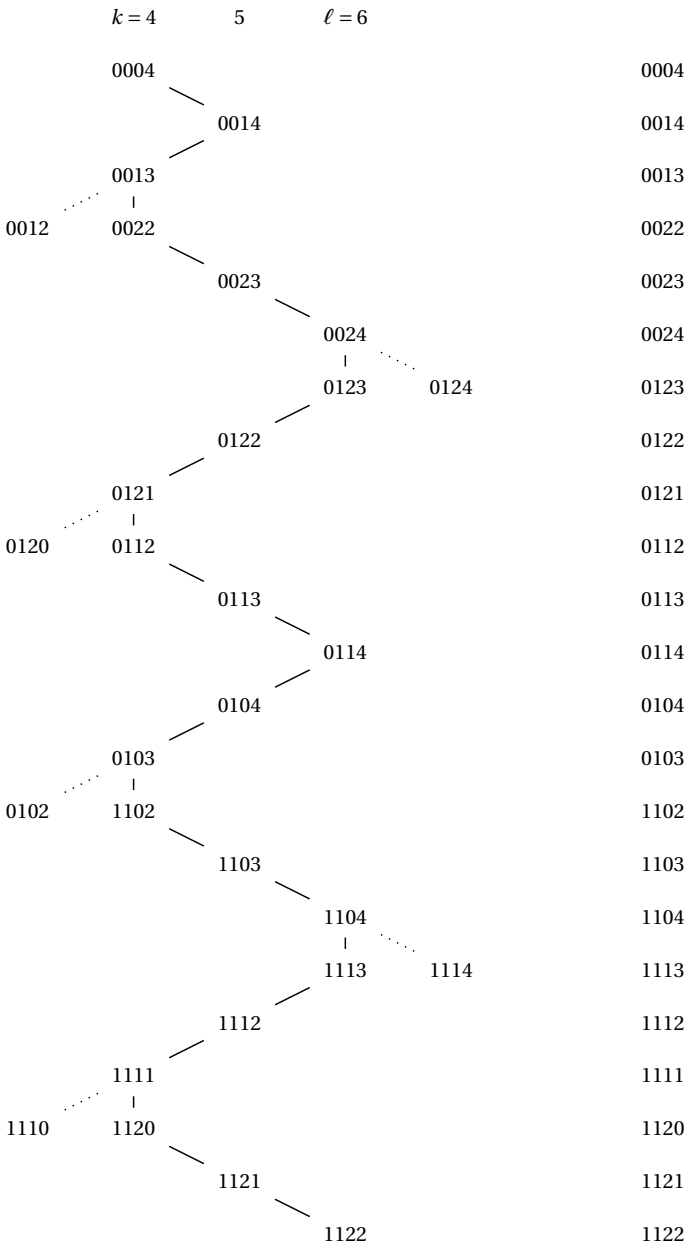


Figure 5.2 – Principe de l'algorithme de rejet.

La figure 5.2 expose les premiers tuples de la génération des 4-compositions de l'intervalle $[4,6]$. Sur ce schéma, sont représentées par une diagonale les générations

utilisant la fonction « successeur » de Williamson et par une verticale celle de Walsh. Le principe de l'algorithme est simple :

- on génére un successeur avec l'algorithme de Williamson ;
- on vérifie si le mot généré ne sort pas de l'intervalle $[k, \ell]$;
- s'il sort des limites, on génére un nouveau mot, cette fois avec l'algorithme de Walsh ;
- on recommence avec l'algorithme de Williamson jusqu'à obtention de la liste complète.

Sur la figure 5.2 (page 96), les rejets sont matérialisés par une diagonale en pointillés. Il est aisé de constater que cet algorithme n'est pas efficace *a priori*, puisque l'on génère des éléments inutiles. En revanche, l'algorithme devient plus acceptable si l'on travaille sur des intervalles $[k, \ell]$ importants. On le voit facilement sur le schéma, c'est lorsque l'on s'approche des bornes que l'on est susceptible de rejeter des valeurs. Plus les bornes sont éloignées l'une de l'autre, moins on aura de rejets.

L'élaboration d'un algorithme de génération efficace dans le cas général reste un problème ouvert.

Les différentes contributions présentées dans ce chapitre ont donné lieu à une présentation [96] lors de la conférence *French Combinatorial Conference* à Orsay en 2010, ainsi qu'à un article [97] publié dans *Information Processing Letters* en 2011 (voir page 111).

Conclusion

Pour clore ce manuscrit, nous allons dégager les perspectives de travail futures basées sur les travaux menés au cours de cette thèse.

Le chapitre 4 (page 49) dans lequel nous avons étudié la duplication miroir complète avec perte aléatoire laisse à penser que l'on pourrait caractériser de manière plus fine la classe de permutations présentée dans la remarque 4.1 (page 74) qui pourraient, dans certains cas, être obtenues en une étape de moins que ne le permet notre travail actuel.

De façon plus globale, on pourrait également s'interroger sur la possibilité d'obtenir les mêmes permutations par des chemins différents ; en effet, la bio-informatique entretenant des liens étroits avec la biologie, il est tout à fait plausible d'imaginer que des contraintes naturelles viennent s'ajouter aux contraintes mathématiques évoquées dans ce chapitre.

Il reste également à trouver un algorithme efficace permettant d'obtenir le code de Gray présenté dans la théorème 5.1 (page 87). Nous avons vu qu'un algorithme de rejet existe et qu'il fonctionne, mais aussi qu'il n'est pas efficace dans tous les cas.

De façon encore plus générale, il serait intéressant à terme d'envisager l'application successive de différentes méthodes de mutation des gènes, *i.e.* imaginer qu'il serait possible d'envisager qu'un génome mute par une duplication, puis mute une nouvelle fois par délétion, puis encore une fois par inversion, etc. Malheureusement, comme nous l'avions évoqué en 4.1 (page 49), ces phénomènes sont complexes et particulièrement coûteux en temps et en capacité de calculs. Malgré tout, il est clair que cette approche serait beaucoup plus fidèle à la réalité biologique que de seulement envisager les enjeux de manière purement mathématique.

Annexe

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Whole mirror duplication-random loss model and pattern avoiding permutations

Jean-Luc Baril *, Rémi Vernay

LE2I UMR-CNRS 5158, Université de Bourgogne, B.P. 47 870, 21078 Dijon Cedex, France

ARTICLE INFO

Article history:
Received 29 June 2009
Received in revised form 10 February 2010
Accepted 20 April 2010
Available online 21 April 2010
Communicated by A.A. Bertossi

Keywords:
Algorithms
Combinatorial problems
Pattern avoiding permutation
Whole duplication-random loss model
Genome
Generating algorithm
Binary reflected Gray code

ABSTRACT

In this paper we study the problem of the whole mirror duplication-random loss model in terms of pattern avoiding permutations. We prove that the class of permutations obtained with this model after a given number p of duplications of the identity is the class of permutations avoiding the alternating permutations of length $2^p + 1$. We also compute the number of duplications necessary and sufficient to obtain any permutation of length n. We provide two efficient algorithms to reconstitute a possible scenario of whole mirror duplications from identity to any permutation of length n. One of them uses the well-known binary reflected Gray code (Gray, 1953) [10]. Other relative models are also considered.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction and notation

The well-known genome duplication consists in copying a part of the original genome inserted into itself, followed by the loss of one copy of each of the duplicated genes (see [2,9,11,14,17] for an explanation of different methods of duplication). From a formal point of view, a genome of n genes is represented by a permutation of length n. In a previous article, Chaudhuri et al. [7] investigated a variant called the tandem duplication-random loss model: the duplicated part (of size K) of the genome is inserted immediately after the original portion, followed by the loss procedure. This model comes from evolutionary biology where it has been applied to the vertebrate mitochondrial genomes. Chaudhuri et al. introduce a notion of distance between two genomes and they provide an algorithm to compute it efficiently for certain regions of the parameter space. Bouvel and Rossin [5] have also studied this model. They proved that the class of permutations obtained from the identity after p steps (of width K) is also a class of pattern avoiding permutations. More particularly, they investigate the restricted case of a *whole duplication* (*W-duplication* for short): the whole duplication consists in copying entirely the permutation on its right and the loss procedure consists to delete one of the two copies of each gene. Here, we give an example of the process of a W-duplication followed by the loss procedure on the permutation 123456:

$$123456 \rightsquigarrow 123456\underbrace{123456}_{\text{duplication}}$$
$$\rightsquigarrow \underbrace{12\not3\not4\not5\not6\not1\not2\not3\not4\not5\not6}_{\text{loss procedure}}$$
$$\rightsquigarrow 124635.$$

So, they prove that the permutations obtained after p W-duplications is the class of permutations avoiding all minimal permutations with 2^p descents, minimal in the sense of pattern-involvement relation on permutations. Moreover, they computed the number of duplication-loss steps

* Corresponding author.
E-mail addresses: barjl@u-bourgogne.fr (J.-L. Baril), remi.vernay@u-bourgogne.fr (R. Vernay).

0020-0190/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ipl.2010.04.016

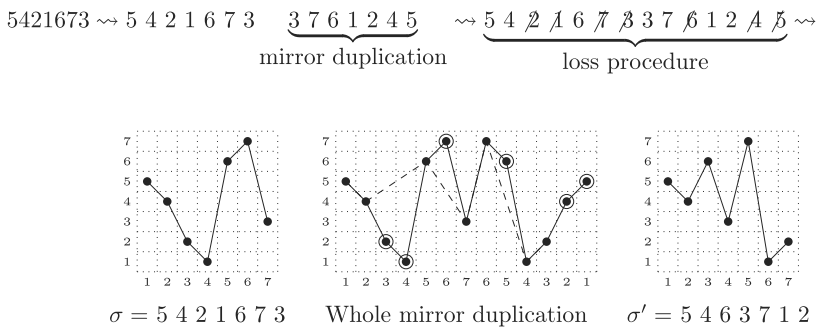


Fig. 1. A WM-duplication of the permutation 5421673. The encircled points are deleted by the loss procedure.

of width K necessary and sufficient to obtain any permutation. More recently, Bouvel and Pergola [4] showed a local and simpler characterization and several properties of the set of minimal permutations with 2^p descents.

In this article, we focus on the *whole mirror duplication-random loss model* (*WM-duplication* for short): it consists in copying the mirror of the permutation on its right followed by the loss procedure. This model very likely occurs in one half of eubacterial genomes, and possibly in most chromosomes [8,15]. Fig. 1 illustrates a *WM-duplication* for the permutation $\sigma = 5421673$:

$$5421673 \rightsquigarrow 5\,4\,2\,1\,6\,7\,3\;\underbrace{3\,7\,6\,1\,2\,4\,5}_{\text{mirror duplication}}$$
$$\rightsquigarrow \underbrace{5\,4\,\not{2}\,\not{1}\,6\,\not{7}\,\not{3}\,3\,7\,\not{6}\,1\,2\,\not{4}\,\not{5}}_{\text{loss procedure}}$$
$$\rightsquigarrow 5463712.$$

Let $\mathcal{S}_n$ denote the set of n-length *sequences* $s = s_1 s_2 \dots s_n$ of positive integers. The *mirror* of s is $\bar{s} = s_n s_{n-1} \dots s_1$. A *subsequence* of s is a sequence $s_{i_1} s_{i_2} \dots s_{i_m}$ for $1 \leqslant i_1 < i_2 < \dots < i_m \leqslant n$. A subsequence is called a *substring* when the set $\{i_j,\ 1 \leqslant j \leqslant m\}$ is an interval, *i.e.* when the subsequence appears as consecutive elements in s. An *ascent* (resp. a *descent*) of s is any position i ($1 \leqslant i \leqslant n-1$) with $s_i < s_{i+1}$ (resp. $s_i > s_{i+1}$). A *run up* (resp. *a run down*) of s is a substring (of length at least one) in which the elements are in increasing order (resp. decreasing order). More generally, a run up (or run down) is said to be *maximal* when it cannot be extended in a longer run up (resp. run down) in the sequence. We refer the reader to Rodney and Wilf [6] for results concerning the enumeration of permutations that have a given number of runs up and down. For instance, if $s = 5467312$ then position 4 is a descent, the substring 46 is a run up, and 467 is a maximal run up. Note that 5 is also a maximal run up. A *valley* of s is a substring which is a run down followed by a run up each of the length of at least two, *i.e.* a substring $s_k s_{k+1} \dots s_\ell$, $1 \leqslant k < \ell \leqslant n$, such that there is j ($k < j < \ell$) verifying $s_k > s_{k+1} > \dots > s_j$ and $s_j < s_{j+1} < \dots < s_\ell$. A valley is *maximal* if the substring is maximal for this property. In the above example, the substrings 5467 and 7312 are the only maximal valleys of s. We denote by $\mathrm{val}(s)$ the number of maximal valleys in s, *i.e.* the cardinality of the set $\{j, s_j < \min\{s_{j-1}, s_{j+1}\}\}$. A recursive formula enumerating permutations with a given number of valleys can be found in [16].

On the other hand, a sequence s of length n is a *permutation* whenever each s_i is a distinct member of the $[n] = \{1, 2, \dots, n\}$. In the sequel, permutations will be denoted by Greek letters: $\sigma, \pi, \tau, \dots$. Let S_n be the set of all permutations of length n ($n \geqslant 1$). In relation to the previous definition, any permutation σ contains at most $\lfloor \frac{n-1}{2} \rfloor$ valleys. A permutation $\sigma \in S_n$ is *alternating* if $\sigma_1 > \sigma_2 < \sigma_3 > \sigma_4 < \sigma_5 > \cdots$. In the literature [1], alternating permutations are also called *down-up* permutations and are enumerated by the Euler numbers (A000111 [19]). For instance, the permutation 324165 is alternating. Notice that an alternating permutation of length n contains exactly $\lfloor \frac{n-1}{2} \rfloor$ valleys.

A permutation π of length k, $k \leqslant n$, is a *pattern* of a permutation $\sigma \in S_n$ if there is a subsequence of σ which is order-isomorphic to π; *i.e.*, if there is a subsequence $\sigma_{i_1} \sigma_{i_2} \dots \sigma_{i_k}$ of σ (with $1 \leqslant i_1 < i_2 < \dots < i_k \leqslant n$) such that $\sigma_{i_\ell} < \sigma_{i_m}$ whenever $\pi_\ell < \pi_m$. We write $\pi \prec \sigma$ to denote that π is a pattern of σ. A permutation σ that does not contain π as a pattern is said to *avoid* π. For example, $\sigma = 1423$ contains the patterns 132, 312 and 123; but σ avoids the pattern 321. The class of all permutations avoiding the patterns $\pi_1, \pi_2, \dots, \pi_k$ is denoted $S(\pi_1, \pi_2, \dots, \pi_k)$, and $S_n(\pi_1, \pi_2, \dots, \pi_k)$ denotes the set of permutations of length n avoiding $\pi_1, \pi_2, \dots$ and π_k. We also say that $S(\pi_1, \pi_2, \dots, \pi_k)$ is a class of pattern-avoiding permutations of basis $\{\pi_1, \pi_2, \dots, \pi_k\}$. A class $\mathcal{C}$ of permutations is *stable* for $\prec$ if, for any $\sigma \in \mathcal{C}$, for any $\pi \prec \sigma$, then we also have $\pi \in \mathcal{C}$. We now formulate a remark that is crucial for the present study.

Remark 1. If a class $\mathcal{C}$ of permutations is stable for $\prec$ then $\mathcal{C}$ is also a class of pattern avoiding permutations of basis $\mathcal{B} = \{\sigma \notin \mathcal{C}, \forall \pi \prec \sigma \text{ with } \pi \neq \sigma, \pi \in \mathcal{C}\}$.

The paper is organized as follow. In Section 2, we prove that the class of permutations obtained from the identity after a given number p of whole mirror duplications is the class of permutations with at most $2^{p-1} - 1$ valleys. This is the class of permutations avoiding the alternating permutations of length $2^p + 1$. Moreover, we obtain the length of a shortest path between any permutation and the identity. In Section 3, we yield two algorithms (and their

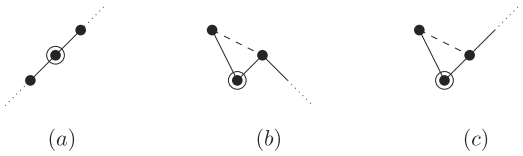


Fig. 2. The three non-isomorphic configurations in the proof of Lemma 1.

complexity) which construct such a shortest path. One of them uses an efficient algorithm for generating the well-known binary reflected Gray code [3,10]. In Section 4, we give some results about other models of duplications using W- and WM-duplications.

2. Pattern avoiding permutations and the mirror duplication-random loss model

In this section we study the WM-duplication random loss model in terms of pattern avoiding permutations. We establish that the class $\mathcal{C}(p)$ obtained from the identity after p WM-duplications is exactly the class of permutations avoiding all alternating permutations of length 2^p+1.

Lemma 1. *Let σ and π be two different permutations such that $\pi \prec \sigma$. Then σ contains at least many valleys as π does.*

Proof. It is sufficient to check the result for $\sigma \in S_n$ and $\pi \in S_{n-1}$ since a straightforward induction will complete the proof. Let $\sigma \in S_n$ and $\pi \in S_{n-1}$ such that $\pi \prec \sigma$, then π is order-isomorphic to a subsequence of σ obtained from σ by deleting only one entry of σ. We distinguish three non-isomorphic configurations illustrated in Fig. 2. More precisely, if we delete the encircled value in configuration (b), this reduces the number of valleys of σ. This does not occur for (a) and (c) where the number of valleys remains invariant after deletion of the encircled value. In all cases, a deletion of value in σ can not increase the number of valleys. □

Lemma 2. *A permutation obtained from the identity after a given number p of WM-duplications contains at most $2^{p-1}-1$ valleys.*

Proof. We obtain the proof by induction. The result holds for $p=1$; indeed a mirror duplication of the identity cannot create a valley. Now, let us assume that each permutation π obtained from the identity after $(p-1)$ mirror duplications contains at most $2^{p-2}-1$ valleys. Let σ be a permutation obtained from the identity after p mirror duplications. Then σ is obtained from a permutation π with $2^{p-2}-1$ valleys after exactly one mirror duplication. Therefore σ can be written as the concatenation of two subsequences of π and $\bar{\pi}$: *i.e.* $\sigma = \tau\tau'$ where τ (resp. τ') is a subsequence of π (resp. $\bar{\pi}$). According to Lemma 1, τ and τ' contains at most $2^{p-2}-1$ valleys. As the concatenation of τ and τ' can (eventually) create one valley between them, σ contains at most $2\cdot(2^{p-2}-1)+1 = 2^{p-1}-1$ valleys which achieves the induction. □

Theorem 1. *The class $\mathcal{C}(p)$ of permutations obtained from the identity after a given number p of mirror duplications is the class of permutations with at most $2^{p-1}-1$ valleys.*

Proof. After considering Lemma 2, it suffices to prove that any permutation σ with at most $2^{p-1}-1$ valleys can be obtained from the identity after p mirror duplications. We proceed by induction on p. Indeed, let σ be a permutation with k valleys, $2^{p-2}-1 < k \leqslant 2^{p-1}-1$. Then σ can be written $\sigma = \tau\tau'$ where τ corresponds to the longest prefix of σ containing exactly $2^{p-2}-1$ valleys and τ' the remaining suffix. We decompose the permutation $\tau = u_1d_1u_2d_2\dots u_\ell d_\ell$, where u_i and d_i are respectively runs up and down defined as follows: u_1 is the first run up excepted its top value; d_1 the run down just after u_1; u_2 the run up just after d_1 excepted its top value, and so on. Notice that u_1 can be empty which does not occur for d_ℓ. Remark that we have $\ell = 2^{p-2}$ and thus $k < 2\ell$. For example, $\tau = 5421673$ has the decomposition: u_1 is empty, $d_1 = 5421$, $u_2 = 6$ and $d_2 = 73$. Let also $u_{\ell+1}d_{\ell+1}\dots u_kd_k\dots u_{2\ell}d_{2\ell}$ be the similar decomposition for τ'. In this decomposition, the runs u_i and d_i are empty for $i > k$. Now let us perform the following process: we sort in decreasing order the values that appear in d_ℓ or $u_{\ell+1}$ which creates a run down D_ℓ; we sort in increasing order the values in u_ℓ or $d_{\ell+1}$ which creates a run up U_ℓ; we construct the sequence $S = U_\ell D_\ell$. We sort in a decreasing order sequence $D_{\ell-1}$ the values in $d_{\ell-1}$ or $u_{\ell+2}$, and so on. At each step j, we insert the obtained ordered sequence U_j and D_j at the beginning of S. The permutation $S = U_1D_1\dots U_{\ell-1}D_{\ell-1}U_\ell D_\ell$ obtained at the end of this process contains at most $2^{p-2}-1$ valleys. See Fig. 3 for an example of construction of S. Thus, by induction, S can be obtained from the identity after $(p-1)$ mirror duplications. By construction σ is reached by one mirror duplication of S. Indeed, from any U_i (resp. D_i), $1 \leqslant i \leqslant \ell$, we can reconstitute the corresponding u_i and $d_{2\ell-i+1}$ (resp. d_i and $u_{2\ell-i+1}$). Therefore σ can be constructed from the identity with p mirror duplications.

Notice that the permutation σ is decomposed in a partition into runs up and down u_j and d_j. We will use this decomposition in Section 3.2 to reconstitute a path of WM-duplications from the identity to σ. □

Corollary 1. *Let σ be a permutation and* $\mathrm{val}(\sigma)$ *the number of its valleys. In the mirror duplication model, $\lceil \log_2(\mathrm{val}(\sigma)+1)\rceil + 1$ steps are necessary and sufficient in order to obtain σ from the identity permutation.*

Proof. Let p be the integer such that $2^{p-2}-1 < \mathrm{val}(\sigma) \leqslant 2^{p-1}-1$, *i.e.* $p = \lceil \log_2(\mathrm{val}(\sigma)+1)\rceil + 1$. According to Theorem 1, $p = \lceil \log_2(\mathrm{val}(\sigma)+1)\rceil + 1$ steps are sufficient to obtain σ from the identity. It is also necessary since σ contains at least 2^{p-2} valleys which means that σ cannot be obtained from the identity in $(p-1)$ steps at most. □

In the next part we will provide algorithms in order to reconstitute a shortest path between the identity and a given permutation.

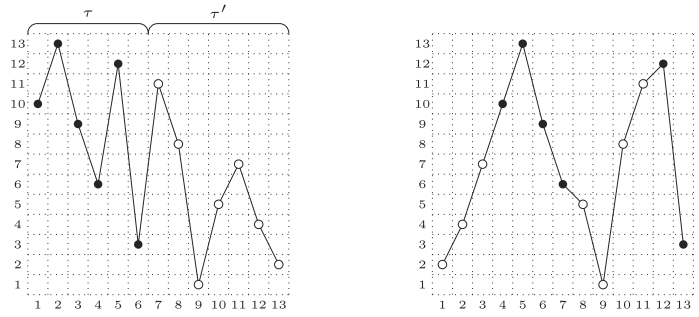


Fig. 3. Decomposition $\sigma = \tau\tau'$ and the permutation S obtained after the process in the proof of Theorem 1. We have $\tau = 10\ 13\ 9\ 6\ 12\ 3$, $\tau' = 11\ 8\ 1\ 5\ 7\ 4\ 2$, $u_1 = 10$, $d_1 = 13\ 9\ 6$, u_2 is empty, $d_2 = 12\ 3$, u_3 is empty, $d_3 = 11\ 8\ 1$, $u_4 = 5$, $d_4 = 7\ 4\ 2$ and, $D_2 = 12\ 3$, $U_2 = 1\ 8\ 11$; $D_1 = 13\ 9\ 6\ 5$; $U_1 = 2\ 4\ 7\ 10$. Thus $S = U_1D_1U_2D_2 = 2\ 4\ 7\ 10\ 13\ 9\ 6\ 5\ 1\ 8\ 11\ 12\ 3$.

Theorem 2. *The class $\mathcal{C}(p)$ of permutations obtained after a given number p of WM-duplications is the class of permutations avoiding the alternating permutations of length $2^p + 1$.*

Proof. Indeed, the permutations obtained after p mirror duplications is stable for the relation $\prec$, *i.e.* if $\sigma \in \mathcal{C}(p)$ and $\pi \prec \sigma$ then $\pi \in \mathcal{C}(p)$ (see Lemma 1). Thus (see Remark 1), $\mathcal{C}(p)$ is also a class of pattern avoiding permutations $S(B)$ where B is the set of minimal (relatively to $\prec$) permutations σ that are not in $\mathcal{C}(p)$. Such a minimal permutation σ contains exactly 2^{p-1} valleys. Indeed, $\sigma \notin \mathcal{C}(p)$ means $\mathrm{val}(\sigma) \geqslant 2^{p-1}$ and any permutation with at least $2^{p-1} + 1$ valleys is not minimal since it contains a pattern $\pi \notin \mathcal{C}(p)$ with 2^{p-1} valleys. Moreover, the configuration $\sigma_{i-1} < \sigma_i < \sigma_{i+1}$ and $\sigma_{i-1} > \sigma_i > \sigma_{i+1}$ can not occur since if we delete σ_i we do not decrease the number of valleys. Also we necessarily have $\sigma_1 > \sigma_2$. Thus we deduce that σ is an alternating permutation with 2^{p-1} valleys and its length is $2^p + 1$. □

For example, $\mathcal{C}(1) = S(213, 312)$ and $\mathcal{C}(2) = S(21435, 31425, 41325, 32415, 42315, 21534, 31524, 51324, 32514, 52314, 41523, 51423, 42513, 52413, 43512, 53412)$.

We also obtain a more general result (see Theorem 3) for the class of permutations having at most p valleys. Notice that, with [16,12,13], a bivariate generating function for this class is:

$$\frac{1}{1-y}\left(1 - \frac{1}{y} + \frac{1}{y}\sqrt{y-1} \times \tan\left(x\sqrt{y-1} + \arctan\left(\frac{1}{\sqrt{y-1}}\right)\right)\right),$$

where the coefficient of $x^n y^k$ is the number of permutations of length n with at most k valleys.

Theorem 3. *The class of permutations having at most p valleys is the class of permutations avoiding the alternating permutations of length $2p + 3$.*

Proof. The proof is obtained *mutatis mutandis*. □

3. Algorithmic considerations

In this section, we present two algorithms that provide a possible scenario of whole mirror duplications and losses for any $\sigma \in S_n$. Both construct a shortest path of WM-duplications from identity to σ. The first algorithm reconstitutes the path in reverse order, *i.e.*, we start from σ and at each step we compute the predecessor of the current permutation until the identity. The second algorithm computes the successor of the current permutation by starting from the identity until σ. These two algorithms give (generally) two different paths. The second method is interesting in the sense where it uses the special structure of the reflected binary Gray code [10]. Moreover, in the case where we want to get the first (resp. last) elements of the path, Algorithm 2 (resp. Algorithm 1) will be more efficient. Finally, we discuss the complexity of these algorithms.

Notice that if we have a shortest path $\mathcal{P}$ of WM-duplications from identity to σ, we can easily obtain a shortest path of WM-duplications from σ^{-1} to the identity by applying σ^{-1} on each permutation of $\mathcal{P}$.

3.1. A path from $12\ldots n$ to $\sigma \in S_n$: Algorithm 1

Here we explain how we can establish an algorithm in order to construct a scenario of WM-duplications from identity to $\sigma \in S_n$ by reconstructing this path from σ, *i.e.* at each step we find the predecessor of the current permutation and the process finishes when the identity is reached. We partition the permutation σ as follow: $\sigma = u'_1 d'_1 u'_2 d'_2 \ldots u'_k d'_k$ where $u'_1 = \sigma_1 \ldots \sigma_{i_1}$ is the maximal run up containing the first entry σ_1; d'_k is the maximal run down containing σ_n; d'_1 is the maximal run down containing σ_{i_1+1}; u'_k is the maximal run up just before d'_k, *i.e.* the value just before d'_k in σ belongs to u'_k; we continue this process by alternating runs up and down until the permutation σ is entirely partitioned. Notice that this decomposition is not the same as we have done in the proof of Theorem 1. For i from 1 to $\ell = \lfloor \frac{k+1}{2} \rfloor$, we define by U_i (resp. D_i) the increasing (resp. decreasing) order sorting of u'_i and d'_{k-i+1} (resp. d'_i and u'_{k-i+1}). Let $\pi = U_1 D_1 U_2 D_2 \ldots U_\ell D_\ell$ be the concatenation of all sorted

Algorithm 1 The procedure BackPath producing a scenario of WM-duplications from $12\dots n$ to $\sigma \in S_n$.

procedure BackPath
while $\sigma \neq 12\dots n$ **do**
- $\pi \leftarrow$ *empty*
- Let $\sigma = u'_1 d'_1 u'_2 d'_2 \dots u'_k d'_k$ be the partition into runs up and runs down

for $i \leftarrow 1$ **to** $\lfloor \frac{k+1}{2} \rfloor$ **do**
- Sort in increasing order u'_i and d'_{k-i+1} and append the sorted substring on the right of π
- Sort in decreasing order d'_i and u'_{k-i+1} and append the sorted substring on the right of π

end for
- $\sigma \leftarrow \pi$

end while

substrings U_i and D_i where D_ℓ can be empty. Therefore π contains at most $(\ell - 1) = \lfloor \frac{k+1}{2} \rfloor - 1 = \lfloor \frac{\mathrm{val}(\sigma)+2}{2} \rfloor - 1 = \lfloor \frac{\mathrm{val}(\sigma)}{2} \rfloor$ valleys. This step requires $\mathcal{O}(n)$ computations since we can simultaneously detect and sort the runs up and down. By iterating this process with this new permutation π, Corollary 1 ensures that the procedure BackPath (see Algorithm 1) constructs a path from identity to σ in $(\lceil \log_2(\mathrm{val}(\sigma)+1) \rceil + 1)\mathcal{O}(n)$, *i.e.* $\mathcal{O}(n \cdot \log_2(n))$ in the worst case. For instance, this process applied to the permutation $\sigma = \mathbf{5}421\mathbf{6}73$ gives the path $\sigma \leftarrow \mathbf{357}6421 \leftarrow 1234567$ (the runs up are in boldface).

3.2. A path from $12\dots n$ to $\sigma \in S_n$: Algorithm 2

In this section, we construct a path of WM-duplications from $12\dots n$ to the permutation $\sigma \in S_n$, *i.e.* we start from the identity and at each step we find the successor of the current permutation; the process finishes when σ is reached. We decompose the permutation $\sigma = u_1 d_1 u_2 d_2 \dots u_k d_k \dots u_{2\ell} d_{2\ell}$ in runs up and down in the same way as we have done in the proof of Theorem 1. We recall this decomposition: u_i and d_i are respectively runs up and down defined as follows: u_1 is the first run up excepted its top value; d_1 the run down just after u_1; u_2 the run up just after d_1 excepted its top value, and so on. Notice that u_1 can be empty: such a decomposition is given as example in the proof of Theorem 1.

We now label each run in this decomposition with the reflected binary Gray code [10]: we can do this using a loopless algorithm introduced by Bitner, Ehrlich, and Reingold [3]. For instance, Fig. 4 shows such a labeling. The structure of the binary reflected Gray code B_n is crucial: $B_n = 0 \cdot B_{n-1} \cup 1 \cdot \overline{B_{n-1}}$ anchored by $B_1 = \{0, 1\}$ and where $\overline{B_n}$ is the list B_n considered in the reverse order. The first runs from u_1 to d_ℓ have 0 as their least significant bit, then those have 1. According to the proof of Theorem 1, we reconstitute the previous permutation π of σ by concatenating (on the left) the sorting in decreasing order D_ℓ of d_ℓ and $u_{\ell+1}$, the sorting in increasing order U_ℓ of u_ℓ and $d_{\ell+1}$, and so on.

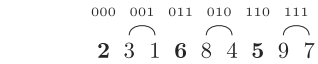
Conversely, at the step j of our algorithm we perform a WM-duplication of π by: (i) keeping in the first copy of π the elements labeled 0 on the jth least significant bit, and, (ii) keeping in the mirror the elements labeled 1 on the jth least significant bit (see Fig. 4 for an example). This step requires only $\mathcal{O}(n)$ computations.

Algorithm 2 The procedure Path producing a scenario of WM-duplications from $12\dots n$ to $\sigma \in S_n$.

procedure Path
- $\pi = 12\dots n$
- Partition $\sigma = u_1 d_1 u_2 d_2 \dots u_k d_k$ into runs up and runs down
- Label the runs up and runs down with the reflected binary Gray code (loopless algorithm [3])

for $j = 1$ **to** $1 + \lfloor \log_2(k-1) \rfloor$ **do**
- Make a WM-duplication step on π that keeps in the first copy of π exactly the elements whose label has 0 in its jth least significant bit.

end for



$$123456789 \rightarrow 245897631 \rightarrow 231679854 \rightarrow 231684597$$

Fig. 4. Labeling of $\sigma = 231684597$ with the binary reflected Gray code according to the runs up and down. Example of a path of WM-duplications from the identity permutation to σ.

By iterating this process, Corollary 1 ensures that the procedure Path (see Algorithm 2) constructs a path from the identity to σ in $(\lceil \log_2(\mathrm{val}(\sigma) + 1) \rceil + 1)\mathcal{O}(n)$, *i.e.* $\mathcal{O}(n \cdot \log_2(n))$ in the worst case. Remark that Algorithm 2 is more efficient than Algorithm 1 in the case where we want to get the first permutations of the path. An implementation of our algorithm is available on http://www.u-bourgogne.fr/jl.baril/id2perm.php.

4. Other models of duplication

In this section, we investigate two variants of the whole mirror duplication: 1) we do one WM-duplication of the identity followed by several W-duplications and, 2) we do one WM- or/and W-duplication of the identity followed by several W-duplications. For these two cases we provide a characterization of the class of permutations obtained after p duplications.

4.1. One WM-duplication followed by several W-duplications

We have seen in the first section that one WM-duplication of the identity gives the class of permutations without valley. Thus, we obtain by induction that the class of permutations obtained after one WM-duplication of the identity followed by $(p-1)$ W-duplications is included in the class of permutations with at most $2^{p-1} - 1$ valleys.

Theorem 4. *The class of permutations obtained after one WM-duplication of the identity followed by $(p-1)$ W-duplications is the class of permutations with at most $2^{p-1} - 1$ valleys.*

Proof. Theorem 1 induces that the class of permutations obtained after one WM-duplication of the identity is the class of permutations without valley. Thus the result is true for $p = 1$. Now we proceed by induction. Let us assume that the class of permutations obtained after one WM-duplication of the identity followed by $(p-1)$ W-duplications is the class of permutations with at most $2^{p-1} - 1$ valleys. Let σ be a permutation in this last class. If we apply one

W-duplication on σ, we obtain a permutation π with at most $2(2^{p-1}-1)+1 = 2^p - 1$ valleys. Now it suffices to prove that any permutation π with at most $2^p - 1$ valleys can be obtained after one W-duplication of a permutation σ with at most $2^{p-1} - 1$ valleys. Indeed, let π be a permutation with k valleys, $2^{p-1} - 1 < k \leqslant 2^p - 1$. As we have done in the proof of Theorem 2, we write $\pi = \tau\tau'$ where τ corresponds to the longest prefix of π containing exactly $2^{p-1} - 1$ valleys and τ' the remaining suffix. We decompose the permutation $\tau = u_1 d_1 u_2 d_2 \dots u_\ell d_\ell$, where u_i and d_i are respectively runs up and down defined as follows: u_1 is the first run up excepted its top value; d_1 is the run down just after u_1; u_2 is the run up just after d_1 excepted its top value, and so on. We necessarily have $\ell = 2^{p-1}$ and thus $k < 2\ell$. Let also $u_{\ell+1} d_{\ell+1} \dots u_k d_k \dots u_{2\ell} d_{2\ell}$ be the similar decomposition for τ'. Now let us perform the following process: we sort in increasing order the values that appear in u_1 or $u_{\ell+1}$ which creates a run up U_1; we sort in decreasing order the values in d_1 or $d_{\ell+1}$ which creates a run down D_1; we construct the sequence $S = U_1 D_1$. We sort in an increasing order sequence U_2 the values in u_2 or $u_{\ell+2}$, and so on. At each step j, we insert the obtained ordered sequences U_j and D_j on the right of S. The permutation $S = U_1 D_1 \dots U_{\ell-1} D_{\ell-1} U_\ell D_\ell$ obtained at the end of this process contains at most $2^{p-1} - 1$ valleys. By construction π is reached by one W-duplication of S since we can reconstitute u_i and $u_{\ell+i}$ from U_i (resp. d_i and $d_{\ell+i}$ from D_i). We conclude by induction. □

The following corollary is directly deduced from Theorem 4 with the same proof as for Theorem 2.

Corollary 2. *The class of permutations obtained after one WM-duplication of the identity followed by $(p-1)$ W-duplications is the class of permutations avoiding the alternating permutations of length $2^p + 1$.*

4.2. One W- or WM-duplication followed by several W-duplications

In this part, we perform one W- or WM-duplication of the identity followed by several W-duplications. Let us recall that an ascent (resp. a descent) of an n-length permutation σ is a position i, $1 \leqslant i \leqslant n-1$, such that $\sigma(i) < \sigma(i+1)$ (resp. $\sigma(i) > \sigma(i+1)$).

Theorem 5. *The class of permutations obtained after one W- or WM-duplication of the identity followed by p W-duplications is the union of the two classes*: (i) *permutations having at most $2^p - 1$ valleys and,* (ii) *permutations with at most $2^{p+1} - 1$ descents.*

Proof. After one W- or WM-duplication of the identity we obtain the union of the class of permutations without valley with the class of permutations with at most one descent. A straightforward induction on p allows us to obtain the result using Theorems 1 and 4. □

Corollary 3. *The class $\mathcal{C}'(p)$ of permutations obtained after one W- or WM-duplication of the identity followed by p W-duplications is the class of permutations avoiding the permutations of length $3 \cdot 2^p + 1$ having exactly 2^p valleys and 2^{p+1} descents.*

Proof. Let σ be a minimal permutation which is not in $\mathcal{C}'(p)$. The minimality of σ induces that σ does not contain any run up of size at least three. By contradiction, if it is the case then by deleting the second value of the run up, the obtained sequence is isomorphic to a permutation which is also not in $\mathcal{C}'(p)$. Also σ cannot begin by an ascent for the same reasons. The permutation σ necessarily has 2^p valleys and 2^{p+1} descents. Indeed, if σ contains at least $2^{p+1} + 1$ descents, it contains a run down of length at least three which contradicts the fact that σ is minimal. A simple calculation allows us to show that σ is necessarily of length $3 \cdot 2^p + 1$. □

For example, $\mathcal{C}'(0) = S(4132, 3142, 4312, 3241, 3214, 4231, 4213, 2143)$; we have computed the cardinality of the basis of $\mathcal{C}'(1)$, which is 720. In the same way, we have the more general corollary:

Corollary 4. *The class of permutations having at most p valleys and at most $2p+1$ descents is the class of permutations avoiding the permutations of length $3p+4$ having exactly $p+1$ valleys and $2p+2$ descents.*

Notice that the set of permutations of length $3p+4$ having exactly $p+1$ valleys and $2p+2$ descents is also the set of permutations with the same length, the same number of valleys and where every ascent is immediately preceded by a descent (which defines a valley). This set was studied by Shapiro et al. [18] (see also the sequence A101280 in [19]). Indeed, they enumerate the permutations of length n having k peaks and with the additional property that every ascent is immediately followed by a descent. Thus, the cardinality of our set is obtained when $n = 3p+4$ and $k = p+1$. For instance, the first cardinalities for $p = 0, 1, 2, 3, 4$ are $8, 720, 230144, 179266560, 277662253056$.

5. Further research directions

In this paper, the whole mirror duplication model is studied. However, many relative open problems still need to be considered. For example, let σ_1 and σ_2 be two permutations in S_n. Can one exhibit an efficient algorithm to compute the permutation π which minimizes the sum $d(\pi, \sigma_1) + d(\pi, \sigma_2)$ where $d(\pi, \sigma)$ is the minimum of WM-duplication steps required to transform π into σ? Can one characterize the class of avoiding permutations corresponding to the permutations obtained from the identity after one W-duplication followed by p WM-duplications? More generally, can we characterize the permutations obtained after p steps of either W- or WM-duplications? Can one obtain similar results with the tandem mirror duplication model: the mirror of the duplicated part (of size K) is inserted immediately after the original portion and we apply the loss procedure.

Acknowledgements

We would like to thank the anonymous referee and Vincent Vajnovszki for their constructive remarks which have greatly improved this paper.

References

[1] B. Bauslaugh, F. Ruskey, Generating alternating permutations lexicographically, BIT Numerical Mathematics 30 (1) (1990) 17–26.
[2] S. Bérard, A. Bergeron, C. Chauve, C. Paul, Perfect sorting by reversals is not always difficult, IEEE ACM Transactions on Computational Biology and Bioinformatics 4 (1) (2007) 4–16.
[3] J.R. Bitner, G. Ehrlich, E.M. Reingold, Efficient generation of the binary reflected Gray code and its applications, Communications of the ACM 19 (9) (1976) 517–521.
[4] M. Bouvel, E. Pergola, Posets and permutations in the duplication-loss model, Pure Math. Appl. 19 (2–3) (2008) 71–80.
[5] M. Bouvel, D. Rossin, A variant of the tandem duplication-random loss model of genome rearrangement, Theoretical Computer Science 410 (8–10) (2009) 847–858.
[6] E.R. Canfield, H.S. Wilf, Counting permutations by their alternating runs, Journal of Combinatorial Theory 115 (2008) 213–225.
[7] K. Chaudhuri, K. Chen, R. Mihaescu, S. Rao, On the tandem duplication-random loss model of genome rearrangement, in: Proceedings of the Seventeenth Annual ACM–SIAM Symposium on Discrete Algorithm, ACM, New York, NY, USA, 2006, pp. 564–570.
[8] H.D. Chen, W.L. Fan, S.G. Kong, H. Lee, B. Zheng, N. Zhou, Inverse symmetry in genomes and whole-genome inverse duplication, in: International Bioinformatics Workshop (IBW2008), Yunnan University, Kunming, Yunnan, China, 2008.
[9] M.C. Chen, R.C.T. Lee, Sorting by transpositions based on the first increasing substring concept, in: BIBE'04, Proceedings of the 4th IEEE Symposium on Bioinformatics and Bioengineering, IEEE Computer Society, Washington, DC, USA, 2004, p. 553.
[10] F. Gray, Pulse code communication, U.S. Patent, 2,632,058, 1953.
[11] M.T. Hallett, J. Lagergren, New algorithms for the duplication-loss model, in: Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, ACM, New York, NY, USA, 2000, pp. 138–146.
[12] S. Kitaev, Introduction to partially ordered patterns, Discrete Applied Mathematics 155 (2007) 929–944.
[13] S. Kitaev, A. Pyatkin, On avoidance of V- and Λ-patterns in permutations, Ars Combinatoria (2010), in press.
[14] A. Labarre, New bounds and tractable instances for the transposition distance, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 3 (4) (2006) 380–394.
[15] R. Nussinov, Some indications for inverse DNA duplication, Journal of Theoretical Biology 95 (4) (1982) 783.
[16] R. Rieper, M. Zeleke, Valleyless sequences, ArXiv:math/0005180, 2000.
[17] D. Sankoff, Gene and genome duplication, Current Opinion in Genetics and Development 11 (2001) 681–684.
[18] L.W. Shapiro, W.-J. Woan, S. Getu, Runs, slides and moments, SIAM J. Algebraic and Discrete Methods 4 (461) (1983).
[19] N.J.A. Sloane, The on-line encyclopedia of integer sequences, http://www.research.att.com/~njas/sequences/.

Correspondance des références de cet article avec la bibliographie générale :

[1] → [15]
[2] → [27]
[3] → [18]
[4] → [22]
[5] → [24]
[6] → [28]
[7] → [32]
[8] → [33]
[9] → [34]
[10] → [52]
[11] → [55]
[12] → [59]
[13] → [60]
[14] → [66]
[15] → [76]
[16] → [78]
[17] → [81]
[18] → [48]
[19] → [84]

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Restricted compositions and permutations: From old to new Gray codes

V. Vajnovszki *, R. Vernay

LE2I UMR-CNRS 5158, Université de Bourgogne, B.P. 47 870, 21078 Dijon Cedex, France

ARTICLE INFO

Article history:
Received 13 December 2010
Received in revised form 30 March 2011
Accepted 31 March 2011
Available online 8 April 2011
Communicated by J. Torán

Keywords:
Combinatorial problems
Gray codes
(Bounded) compositions of an integer
Permutations
Inversion table

ABSTRACT

Any Gray code for a set of combinatorial objects defines a total order relation on this set: $\boldsymbol{x}$ is less than $\boldsymbol{y}$ if and only if $\boldsymbol{y}$ occurs after $\boldsymbol{x}$ in the Gray code list. Let $\prec$ denote the order relation induced by the classical Gray code for the product set (the natural extension of the Binary Reflected Gray Code to k-ary tuples). The restriction of $\prec$ to the set of compositions and bounded compositions gives known Gray codes for those sets. Here we show that $\prec$ restricted to the set of bounded *compositions of an interval* yields still a Gray code. An n-composition of an interval is an n-tuple of integers whose sum lies between two integers; and the set of bounded n-compositions of an interval simultaneously generalizes product set and compositions of an integer, and so $\prec$ put under a single roof all these Gray codes.

As a byproduct we obtain Gray codes for permutations with a number of inversions lying between two integers, and with even/odd number of inversions or cycles. Such particular classes of permutations are used to solve some computational difficult problems.

© 2011 Published by Elsevier B.V.

1. Introduction

Roughly speaking, a Gray code is a listing of the objects in a combinatorial family so that successive objects differ 'in some pre-specified small way' [5]. Here we adhere to the definition given in [14]: a Gray code for a combinatorial family is a listing of the objects in the family so that successive objects differ by a number of changes bounded independently of the object-size. Alternative more restrictive definitions for Gray codes exist in literature; they are obtained by imposing additional constraints. For instance, in [8, Section 7.2.1.3, p. 10] is given an example of a Gray code for combinations in binary word representation where two consecutive words differ in two positions and all values between them are zeros. And in [12] is defined a Gray code for generalized Dyck words where two consecutive words differ in two positions which are either adjacent or separated by a zero.

For a given integer n, $\mathbb{N}^n$ denotes the set of all integer n-tuples and we adopt the convention that lower case bold letters represent such n-tuples; e.g., $\boldsymbol{w} = w_1 w_2 \dots w_n$.

An n-composition of an integer k is a tuple $\boldsymbol{c}$ with $c_1 + c_2 + \cdots + c_n = k$. Knuth gave[1] a definition of a Gray code for the set of n-compositions of an integer which is defined recursively by Wilf [15] and implemented iteratively by Klingsberg [7]. For two n-tuples $\boldsymbol{b}$ and $\boldsymbol{c}$, $\boldsymbol{c}$ is said *$\boldsymbol{b}$-bounded* if $0 \leqslant c_i \leqslant b_i$, for all i, $1 \leqslant i \leqslant n$. Walsh [13] gave a loopless algorithm for generating a Gray code for bounded compositions of an integer, and in particular for Knuth's Gray code.

A *$\boldsymbol{b}$-bounded n-composition of the integer interval* $[k, \ell]$ is a $\boldsymbol{b}$-bounded tuple $\boldsymbol{c}$ with $k \leqslant c_1 + c_2 + \cdots + c_n \leqslant \ell$. In particular, when $k = \ell$ we retrieve the notion of bounded n-composition of k, and when $k = \ell = b_1 = b_2 = \cdots = b_n$ that of classical composition; and when $k = 0$ and $\ell = b_1 + b_2 + \cdots + b_n$ we retrieve the product set $[b_1] \times [b_2] \times \cdots \times [b_n]$. So, bounded compositions of an interval simultaneously generalize product set and (bounded) compositions of an integer.

* Corresponding author.
E-mail addresses: vvajnov@u-bourgogne.fr (V. Vajnovszki), remi.vernay@u-bourgogne.fr (R. Vernay).

0020-0190/$ – see front matter © 2011 Published by Elsevier B.V.
doi:10.1016/j.ipl.2011.03.022

[1] Unpublished answer to a question of Nijenhuis and Wilf.

In this paper we re-express Walsh's and Knuth's Gray codes for (bounded) compositions of an integer in terms of a unique order relation, and so Walsh's Gray code becomes a sublist of Knuth's one, which in turn is a sublist of the *Reflected Gray Code*. Based on this order relation we generalize Knuth's and Walsh's Gray codes to bounded compositions of *an interval* $[k, \ell]$. We apply these results to obtain Gray codes for permutations with a number of inversions ranging between two integers, and with an even/odd number of inversions or cycles. Such particular classes of permutations are used to solve some computational difficult problems, see for instance [2].

2. Notations and definitions

For an integer $m \in \mathbb{N}$, $[m]$ denotes the set $\{0, 1, \ldots, m\}$ and for an n-tuple $\boldsymbol{b} \in \mathbb{N}^n$:

- $[\boldsymbol{b}]$ is the product set $[b_1] \times [b_2] \times \cdots \times [b_n]$, and
- $\|\boldsymbol{b}\|$ is the componentwise sum of $\boldsymbol{b}$, i.e., $\|\boldsymbol{b}\| = \sum_{i=1}^{n} b_i$.

The Reflected Gray Code for the product set $[\boldsymbol{b}]$, denoted here by $\mathcal{G}_n(\boldsymbol{b})$, is the natural extension of the Binary Reflected Gray Code [4] to this set. $\mathcal{G}_n(\boldsymbol{b})$ was defined recursively by Er in [1] by the relation below and generated looplessly by Williamson in [16, p. 112]

$$\mathcal{G}_n(\boldsymbol{b}) = \begin{cases} \emptyset \quad \text{if } n = 0, \\ 0\mathcal{G}_{n-1}(\boldsymbol{b}'), 1\overline{\mathcal{G}_{n-1}(\boldsymbol{b}')}, 2\mathcal{G}_{n-1}(\boldsymbol{b}'), \ldots, \\ \quad b_1\mathcal{G}'_{n-1}(\boldsymbol{b}') \quad \text{if } n > 0, \end{cases} \tag{1}$$

where $\boldsymbol{b}' = b_2 b_3 \ldots b_n$, $\overline{\mathcal{G}_{n-1}(\boldsymbol{b}')}$ is the reverse of $\mathcal{G}_{n-1}(\boldsymbol{b}')$ and $\mathcal{G}'_{n-1}(\boldsymbol{b}')$ is $\mathcal{G}_{n-1}(\boldsymbol{b}')$ or $\overline{\mathcal{G}_{n-1}(\boldsymbol{b}')}$ according as b_1 is even or odd. In $\mathcal{G}_n(\boldsymbol{b})$ two consecutive tuples differ in a single position and by $+1$ or -1 in this position, see the first column of Table 1 for the list $\mathcal{G}_3(2\,1\,3)$.

The *Reflected Gray Code Order* $\prec$ on $\mathbb{N}^n$ is defined as: $\boldsymbol{x} = x_1 x_2 \ldots x_n$ is less than $\boldsymbol{y} = y_1 y_2 \ldots y_n$ in $\prec$ order, and we denote it by $\boldsymbol{x} \prec \boldsymbol{y}$, if either

- $\sum_{j=1}^{i-1} x_j$ is even and $x_i < y_i$, or
- $\sum_{j=1}^{i-1} x_j$ is odd and $x_i > y_i$,

where i is the leftmost position with $x_i \neq y_i$. It is easy to see that $\mathcal{G}_n(\boldsymbol{b})$ lists tuples in $[\boldsymbol{b}]$ in $\prec$ order.

In the following we introduce the notions of successor, and the first and last tuple in a set; unless explicitly specified otherwise, they are considered with respect to $\prec$ order:

- for $A \subset \mathbb{N}^n$, first(A) and last(A) stand for the first and last tuple (in $\prec$ order) in the set A,
- for $\boldsymbol{c} \in [\boldsymbol{b}]$, succ$^{\boldsymbol{b}}(\boldsymbol{c})$ is the successor of $\boldsymbol{c}$ (in $\prec$ order) in the product set $[\boldsymbol{b}]$,
- for a set A of tuples, $\mathcal{A}$ is the *ordered list* of tuples in A, listed in $\prec$ order.

Table 1
The sets $[\boldsymbol{b}]$, $W_{2,2}$, $W_{3,3}$, $W_{2,3}$, $W_{4,4}$, $W_{2,4}$ listed in $\prec$ order for $n = 3$ and $\boldsymbol{b} = 2\,1\,3 \in \mathbb{N}^3$.

$\mathcal{G}_n(\boldsymbol{b})$	$\mathcal{W}_{2,4}$		
	$\mathcal{W}_{2,3}$		$\mathcal{W}_{4,4}$
	$\mathcal{W}_{2,2}$	$\mathcal{W}_{3,3}$	
000			
001			
002	✓		
003		✓	
013			✓
012		✓	
011	✓		
010			
110	✓		
111		✓	
112			✓
113			
103			✓
102		✓	
101	✓		
100			
200	✓		
201		✓	
202			✓
203			
213			
212			
211			✓
210		✓	

3. Bounded compositions of an interval

Definition 1. For three integers $n \in \mathbb{N}$, $k \in \mathbb{Z}$ and $\ell \in \mathbb{N}$ with $k \leqslant \ell$, and an n-tuple $\boldsymbol{b} = b_1 b_2 \ldots b_n \in \mathbb{N}^n$, a *$\boldsymbol{b}$-bounded n-composition of the interval* $[k, \ell]$ is an n-tuple $\boldsymbol{c} = c_1 c_2 \ldots c_n \in \mathbb{N}^n$ such that

- $k \leqslant \sum_{i=1}^{n} c_i \leqslant \ell$, and
- $0 \leqslant c_i \leqslant b_i$, for $1 \leqslant i \leqslant n$.

We denote by $W^{\boldsymbol{b}}_{k,\ell}$ the set of $\boldsymbol{b}$-bounded n-compositions of $[k, \ell]$, and obviously $W^{\boldsymbol{b}}_{k,\ell} = \bigcup_{i=k}^{\ell} W^{\boldsymbol{b}}_{i,i}$. For $\boldsymbol{c} \in W^{\boldsymbol{b}}_{k,\ell}$, succ$^{\boldsymbol{b}}_{k,\ell}(\boldsymbol{c})$ denotes the successor of $\boldsymbol{c}$ (in $\prec$ order) in the set $W^{\boldsymbol{b}}_{k,\ell}$ and in the following we will omit the upper index $\boldsymbol{b}$ when it does not create ambiguity.

Remark that in the previous definition k can be a negative number; in this case $W_{k,\ell} = W_{0,\ell}$. Similarly, $W_{k,\ell} = W_{k,\|\boldsymbol{b}\|}$ if $\ell \geqslant \|\boldsymbol{b}\|$.

Remark 1. As particular cases we obtain:

- $W_{0,\|\boldsymbol{b}\|}$ is the product set $[\boldsymbol{b}]$ generated looplessly in [16, p. 112].
- $W_{k,k}$ is the set of $\boldsymbol{b}$-bounded n-compositions of k generated looplessly in [13].
- If $\boldsymbol{b} = [k]^n$, then $W_{k,k}$ is the set of unrestricted n-compositions of k.

It can happen that first$(W_{k,\ell}) =$ first$(W_{k+1,\ell+1})$. This case occurs only if $00\ldots0$ belongs both to $W_{k,\ell}$ and $W_{k+1,\ell+1}$, and so when $k + 1 \leqslant 0$. Similarly, it can happen that last$(W_{k,\ell}) =$ last$(W_{k+1,\ell+1})$, and this case occurs

only if last($[\boldsymbol{b}]$) belongs both to $W_{k,\ell}$ and $W_{k+1,\ell+1}$. Formally, we have:

Lemma 1.

(i) first($W_{k+1,\ell+1}$) *and* first($W_{k,\ell}$) *are either both equal to* $00\ldots0$, *or they differ in precisely one position and with difference* 1 *in this position.*
(ii) last($W_{k+1,\ell+1}$) *and* last($W_{k,\ell}$) *are either both equal to* last($[\boldsymbol{b}]$), *or they differ in precisely one position and with difference* 1 *in this position.*
(iii) *If* $\boldsymbol{s} = \text{last}(W_{k+1,\ell+1})$ *differs from* $\boldsymbol{t} = \text{last}(W_{k,\ell})$ *in position* i, *then* $s_j = t_j \in \{0, b_j\}$ *for all* $j \neq i$.
(iv) $\text{last}(W_{k,\ell}) \in \{\text{last}(W_{k,k}), \text{last}(W_{\ell,\ell}), \text{last}([\boldsymbol{b}])\}$.

Proof. (i) first($W_{k,\ell}$) is the lexicographically least tuple $\boldsymbol{c} \in [\boldsymbol{b}]$ with $\|\boldsymbol{c}\| = \max\{0, k\}$, and first($W_{k+1,\ell+1}$) the lexicographically least tuple $\boldsymbol{d} \in [\boldsymbol{b}]$ with $\|\boldsymbol{d}\| = \max\{0, k+1\}$, and the result follows.

(ii) Suppose that $\text{last}(W_{k,\ell}) \neq \text{last}([\boldsymbol{b}])$ (that is, $\text{last}([\boldsymbol{b}]) \notin W_{k,\ell}$). The case $\text{last}(W_{k+1,\ell+1}) \neq \text{last}([\boldsymbol{b}])$ (that is, $\text{last}([\boldsymbol{b}]) \notin W_{k+1,\ell+1}$) is similar.

- If $0 \leqslant \ell < b_1$, then

  $$\text{last}(W_{k,\ell}) = \ell 0\ldots0$$

  and

  $$\text{last}(W_{k+1,\ell+1}) = (\ell+1)0\ldots0$$

  and the statement follows.
- If $\ell \geqslant b_1$ and b_1 is odd, then

  $$\text{last}(W_{k,\ell}) = b_1 \cdot \text{first}(W_{k-b_1,\ell-b_1})$$

  and

  $$\text{last}(W_{k+1,\ell+1}) = b_1 \cdot \text{first}\big(W^{\boldsymbol{b}'}_{k+1-b_1,\ell+1-b_1}\big)$$

  with $\boldsymbol{b}' = b_2 b_3 \ldots b_n$. But $\text{first}(W^{\boldsymbol{b}'}_{k-b_1,\ell-b_1}) \neq 00\ldots0$, otherwise $\text{last}(W_{k,\ell}) = \text{last}([\boldsymbol{b}])$, and by (i) of the present lemma the statement follows.
- If $k \geqslant b_1$ and b_1 is even, then

  $$\text{last}(W_{k,\ell}) = b_1 \cdot \text{last}\big(W^{\boldsymbol{b}'}_{k-b_1,\ell-b_1}\big)$$

  and

  $$\text{last}(W_{k+1,\ell+1}) = b_1 \cdot \text{last}\big(W^{\boldsymbol{b}'}_{k+1-b_1,\ell+1-b_1}\big).$$

  But $\text{last}(W^{\boldsymbol{b}'}_{k-b_1,\ell-b_1}) \neq \text{last}([\boldsymbol{b}'])$, otherwise $\text{last}(W_{k,\ell}) = \text{last}([\boldsymbol{b}])$, and induction on n completes the proof.

(iii) and (iv) are consequences of the proof of (ii). □

If $\boldsymbol{c} \in W_{k,\ell}$ and $\text{succ}(\boldsymbol{c}) \in W_{k,\ell}$, then $\text{succ}_{k,\ell}(\boldsymbol{c}) = \text{succ}(\boldsymbol{c})$; otherwise, Proposition 1 below states that, $\text{succ}_{k,\ell}(\boldsymbol{c}) = \text{succ}_{p,p}(\boldsymbol{c})$ with $p = \|\boldsymbol{c}\|$. Before proving this proposition we need a technical lemma.

Lemma 2. *Let* $\boldsymbol{c}, \boldsymbol{c}' \in W_{k,\ell}$ *with* $\boldsymbol{c}' = \text{succ}_{k,\ell}(\boldsymbol{c})$ *and let* u *be the leftmost position where* $c'_i \neq c_i$. *Then we have either* $c'_u = c_u + 1$ *or* $c'_u = c_u - 1$.

Proof. Let $I \subset \{1, 2, \ldots, n\}$ be the set of indices i with $c'_i \neq c_i$. So, u is the minimal element of I and suppose that $c'_u = c_u + p$ for some $p > 1$; the case where $c'_u = c_u - p$ is similar.

- If $c'_i > c_i$ for all $i \in I$, then the tuple $\boldsymbol{c}''$ defined by

  $$c''_i = \begin{cases} c_i & \text{if } i \neq u, \\ c_i + 1 & \text{if } i = u, \end{cases}$$

  belongs to $W_{k,\ell}$ and is such that $\boldsymbol{c} \prec \boldsymbol{c}'' \prec \boldsymbol{c}'$ (actually, in this case $\boldsymbol{c}'' = \text{succ}(\boldsymbol{c})$). This is in contradiction with $\boldsymbol{c}' = \text{succ}_{k,\ell}(\boldsymbol{c})$.
- If there exists $v \in I$, $v > u$, such that $c'_v = c_v - r$ for some $r \geqslant 1$, then the tuple $\boldsymbol{c}''$ defined by

  $$c''_i = \begin{cases} c_i & \text{if } i \neq u, v, \\ c_i + (p-1) & \text{if } i = u, \\ c_i - (r-1) & \text{if } i = v, \end{cases}$$

  belongs to $W_{k,\ell}$ and is such that $\boldsymbol{c} \prec \boldsymbol{c}'' \prec \boldsymbol{c}'$, which yields again a contradiction. □

Proposition 1. *Let* $\boldsymbol{c}, \boldsymbol{c}' \in W_{k,\ell}$ *with* $\boldsymbol{c}' = \text{succ}_{k,\ell}(c)$. *Then one of the two statements below holds.*

(i) $\boldsymbol{c}' = \text{succ}(\boldsymbol{c})$, *and so* $\boldsymbol{c}$ *and* $\boldsymbol{c}'$ *differ in precisely one position and with difference* 1 *in this position,*
(ii) $\|\boldsymbol{c}\| = \|\boldsymbol{c}'\|$, *and* $\boldsymbol{c}$ *and* $\boldsymbol{c}'$ *differ in two positions and by* $+1$ *and* -1 *in these positions.*

Proof. Let u be the leftmost position where $\boldsymbol{c}$ differs from $\boldsymbol{c}'$. By Lemma 2, $c'_u = c_u + \alpha$, with $\alpha \in \{-1, 1\}$.

If $\beta = \sum_{i=1}^{u} c_i$ is odd, then

$$\boldsymbol{c} = c_1 c_2 \ldots c_u \,\text{first}\big(W^{\boldsymbol{b}'}_{k-\beta,\ell-\beta}\big)$$

and

$$\boldsymbol{c}' = c_1 c_2 \ldots (c_u + \alpha)\,\text{first}\big(W^{\boldsymbol{b}'}_{k+\alpha-\beta,\ell+\alpha-\beta}\big)$$

with $\boldsymbol{b}' = b_{u+1} b_{u+2} \ldots b_n$. In this case, by Lemma 1(i) we have either

- $\text{first}(W^{\boldsymbol{b}'}_{k-\beta,\ell-\beta}) = \text{first}(W^{\boldsymbol{b}'}_{k+\alpha-\beta,\ell+\alpha-\beta}) = 00\ldots0$, and so $\boldsymbol{c}' = \text{succ}(\boldsymbol{c})$ and (i) follows, or
- $\text{first}(W^{\boldsymbol{b}'}_{k-\beta,\ell-\beta})$ differs from $\text{first}(W^{\boldsymbol{b}'}_{k+\alpha-\beta,\ell+\alpha-\beta})$ in a single position and by $-\alpha$ in this position and (ii) follows.

If $\beta = \sum_{i=1}^{u} c_i$ is even, then

$$\boldsymbol{c} = c_1 c_2 \ldots c_u \,\text{last}\big(W^{\boldsymbol{b}'}_{k-\beta,\ell-\beta}\big)$$

and

$$\boldsymbol{c}' = c_1 c_2 \ldots (c_u + \alpha)\,\text{last}\big(W^{\boldsymbol{b}'}_{k+\alpha-\beta,\ell+\alpha-\beta}\big).$$

Now by applying Lemma 1(ii) the result holds. □

A consequence of the previous proposition and of its proof is the next corollary.

Corollary 1. *Let $\boldsymbol{c} \in W_{k,\ell}$ and $\boldsymbol{c}' = \mathrm{succ}_{k,\ell}(\boldsymbol{c})$.*

(i) *If $\mathrm{succ}(\boldsymbol{c}) \in W_{k,\ell}$, then $\boldsymbol{c}' = \mathrm{succ}(\boldsymbol{c})$.*
(ii) *If $\mathrm{succ}(\boldsymbol{c}) \notin W_{k,\ell}$, then $\boldsymbol{c}' = \mathrm{succ}_{p,p}(\boldsymbol{c})$ with $p = \|\boldsymbol{c}\|$. In this case $p = k$ or $p = \ell$, and if u and v, $u \prec v$, are the two positions where $\boldsymbol{c}$ and $\boldsymbol{c}'$ differ, then $c_i = c'_i \in \{0, b_i\}$ for all $i > u$, $i \neq v$.*

Combining Proposition 1 and Corollary 1, we have:

Theorem 1. *The list $\mathcal{W}_{k,\ell}$ of tuples in the set $W_{k,\ell}$ listed in $\prec$ order is a Gray code where two consecutive tuples differ in at most two positions and by 1 in these positions. In particular, $\mathcal{W}_{k,k}$ is a Gray code for the $\boldsymbol{b}$-bounded compositions of the integer k.*

An alternative Gray code for $W_{k,\ell}$ is given by the next corollary.

Corollary 2. *For $k \leqslant \ell \leqslant \|\boldsymbol{b}\|$, the list*

$$\mathcal{W}_{k,k}, \overline{\mathcal{W}_{k+1,k+1}}, \mathcal{W}_{k+2,k+2}, \ldots, \mathcal{W}'_{\ell,\ell}$$

is a Gray code for the set $W_{k,\ell}$, where $\mathcal{W}'_{\ell,\ell}$ is $\mathcal{W}_{\ell,\ell}$ or $\overline{\mathcal{W}_{\ell,\ell}}$ according as $\ell - k + 1$ is odd or even.

Proof. By Theorem 1, for each i, $k \leqslant i \leqslant \ell$, $\mathcal{W}_{i,i}$ is a Gray code for $W_{i,i}$. The last tuple of $\mathcal{W}_{i,i}$ is $\mathrm{last}(W_{i,i})$ and the first tuple of $\overline{\mathcal{W}_{i+1,i+1}}$ is $\mathrm{last}(W_{i+1,i+1})$, and by Lemma 1(ii) they differ in precisely one position and with difference 1 in this position. Similarly, by Lemma 1(i) the last tuple of $\overline{\mathcal{W}_{i,i}}$ and the first tuple of $\mathcal{W}_{i+1,i+1}$ differ in the same way. □

For example, for $\boldsymbol{b} = 44 \in \mathbb{N}^2$, we have the following Gray code lists for the set $W_{3,4}$:

- $\mathcal{W}_{3,4} = 03, 04, 13, 12, 21, 22, 31, 30, 40$, and
- $\mathcal{W}_{3,3}, \overline{\mathcal{W}_{4,4}} = 03, 12, 21, 30, 40, 31, 22, 13, 04$.

In the first list there are 3 transitions of size 2; in the second one there are 7 transitions of size 2. However, the list $04, 03, 13, 12, 22, 21, 31, 30, 40$ for the same set is more restrictive since there is no transition of size 2, and it can be considered 'more optimal'. The existence of 'minimal-change lists' in the general case remains an open problem, see the acknowledgment at the end of this paper.

As mentioned earlier, the set $W_{k,\ell}$ generalizes the notions of product set, unrestricted and bounded compositions of an integer. The next remark says that this remains true in the ordered case.

Remark 2. As particular cases we have:

- $\mathcal{W}_{0,\|\boldsymbol{b}\|}$ is the Reflected Gray Code, $\mathcal{G}_n(\boldsymbol{b})$, for the product set $[\boldsymbol{b}]$ defined by (1) (cf. Er in [1]), and which is generated looplessly by Williamson in [16, p. 112],
- if $\boldsymbol{b} = [k]^n$, then $\mathcal{W}_{k,k}$ is Knuth's [16,7] Gray code for unrestricted n-compositions of k,
- $\mathcal{W}_{k,k}$ becomes Walsh's Gray code for $\boldsymbol{b}$-bounded n-compositions of k, defined and generated looplessly in [13].

For two lists $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} \subset \mathcal{B}$ means that $\mathcal{A}$ is a (possibly scattered) sublist of $\mathcal{B}$; in this case the corresponding subsets satisfy $A \subset B$. With this notation we have

Remark 3. $W^{\boldsymbol{b}}_{u,v} \subset W^{\boldsymbol{c}}_{k,\ell}$ if $[u, v]$ is a sub-interval of $[k, \ell]$ and $\boldsymbol{b}$ is componentwise smaller than or equal to $\boldsymbol{c}$. In particular, $\mathcal{W}^{\boldsymbol{b}}_{k,k} \subset \mathcal{W}^{\boldsymbol{b}}_{k,\ell} \subset \mathcal{W}^{\boldsymbol{b}}_{0,\|\boldsymbol{b}\|} = \mathcal{G}_n(\boldsymbol{b})$.

4. Restricted permutations

There is a natural correspondence between the product set $[0] \times [1] \times \cdots \times [n-1]$ and the set S_n of length-n permutations. Let $\mathcal{G}_n([0] \times [1] \times \cdots \times [n-1])$ be the previous defined Gray code for the product set $[0] \times [1] \times \cdots \times [n-1]$ and $\mathcal{S}_n$ its *list-image* through this correspondence. Lemma 4 says that $\mathcal{S}_n$ is a Gray code for S_n, which is actually the well-known Johnson–Trotter–Steinhaus Gray code for permutations [6,11,10]. Moreover, Propositions 2 and 3 say that the restriction of $\mathcal{S}_n$ to some particular classes of permutations yields still a Gray code.

In a permutation $\tau \in S_n$ a couple (i, j) is an inversion if $i < j$ but $\tau(i) > \tau(j)$. The array $\boldsymbol{t} = t_1 t_2 \ldots t_n \in [0] \times [1] \times \cdots \times [n-1]$ is the *inversion table* (see [9, p. 20]) of a permutation $\tau \in S_n$ if, for any i, $1 \leqslant i \leqslant n$,

$t_i =$ the number of elements in τ smaller than i and at its right.

Clearly, $\|\boldsymbol{t}\|$ is the number of inversions in the permutation τ, denoted by $\mathrm{inv}\,\tau$; it is also the number of adjacent transpositions needed to sort the permutation τ.

For every $n \in \mathbb{N}$, the function

$$\phi : [0] \times [1] \times \cdots \times [n-1] \to S_n$$

defined by $\tau = \phi(\boldsymbol{t})$ where $\boldsymbol{t}$ is the inversion table of τ, is a bijection from $[0] \times [1] \times \cdots \times [n-1]$ into S_n.

If two permutations differ by an adjacent transposition, then their inversion tables differ in precisely one position and with difference 1 in this position. Conversely, it is easy to see that if two inversion tables differ in precisely one position and with difference 1 in this position, then their corresponding permutations differ by the transposition of two elements, which are not necessarily adjacent. For example, $\boldsymbol{s} = 011032$ differs from $\boldsymbol{t} = 001032$ in a single position and $253614 = \phi(\boldsymbol{s})$ differs from $153624 = \phi(\boldsymbol{t})$ by a non-adjacent transposition. However, under additional constraints, the adjacency property is preserved.

Lemma 3. *Let $\boldsymbol{s}, \boldsymbol{t} \in [0] \times [1] \times [2] \times [n-1]$. If there is an $i \in \{1, 2, \ldots, n\}$ with:*

- *$t_i = s_i + 1$ and $s_j = t_j$ for all $j \neq i$,*
- *$s_j = t_j \in \{0, j-1\}$ for all $j > i$,*

then $\sigma = \phi(\boldsymbol{s})$ and $\tau = \phi(\boldsymbol{t})$ differ by an adjacent transposition.

Proof. If $i = n$, then n is not on the leftmost position in σ, and τ is obtained from σ by transposing n and the element at its left.

If $i \neq n$, then n is the leftmost or rightmost element of σ according as s_n is $n-1$ or 0, and generally, any j, $j > i$, is the leftmost or rightmost element of the permutation obtained from σ by deleting all elements larger than j. So, σ has the form

$$\sigma = \underbrace{\sigma_1\sigma_2\ldots\sigma_u}_{>i}\,\underbrace{\sigma_{u+1}\sigma_{u+2}\ldots\sigma_{u+i}}_{\leqslant i}\,\underbrace{\sigma_{u+i+1}\sigma_{u+i+2}\ldots\sigma_n}_{>i}$$

with $\sigma_{u+1}\sigma_{u+2}\ldots\sigma_{u+i}$ a permutation in S_i with the inversion table $s_1s_2\ldots s_i$. Since $s_i \neq i-1$, i is not the leftmost element of $\sigma_{u+1}\sigma_{u+2}\ldots\sigma_{u+i}$, and (as in the case $i = n$) τ is obtained from σ by transposing i and the element at its left. □

We say that two permutations differ by an *adjacent transposition* if one can be obtained from the other by transposing two adjacent elements.

By the previous lemma and Lemma 1(iii) we have:

Lemma 4. *If $\boldsymbol{s}$ and $\boldsymbol{t}$ are two successive tuples in $\mathcal{G}_n([0]\times[1]\times\cdots\times[n-1])$ then the permutations $\phi(\boldsymbol{s})$ and $\phi(\boldsymbol{t})$ in S_n differ by an adjacent transposition, and so $\mathcal{S}_n$ is a Gray code for S_n.*

Actually $\mathcal{S}_n$ is classical Johnson–Trotter–Steinhaus Gray code for the set of length n permutations [6,11,10].

Proposition 2. *The restriction of $\mathcal{S}_n$ to the set of permutations with a number of inversions lying between two integers is a Gray code where two consecutive permutations differ by one or two adjacent transpositions.*

Proof. Let σ and τ be two consecutive permutations in the restriction of $\mathcal{S}_n$ to the set of permutations with a number of inversions between k and ℓ. Let $\boldsymbol{s}$ and $\boldsymbol{t}$ be the corresponding tuples in $\boldsymbol{b} = [0]\times[1]\times\cdots\times[n-1]$ with $\phi(\boldsymbol{s}) = \sigma$ and $\phi(\boldsymbol{t}) = \tau$. By the definition of $\mathcal{S}_n$, $\boldsymbol{s}$ and $\boldsymbol{t}$ are consecutive in $\mathcal{W}^{\boldsymbol{b}}_{k,\ell}$. So, by Proposition 1 either: $\boldsymbol{t} = \mathrm{succ}^{\boldsymbol{b}}(\boldsymbol{s})$ and in this case, by Lemma 4, σ differs from τ by an adjacent transposition; or $\boldsymbol{s}$ and $\boldsymbol{t}$ differ in two positions, say u and v, $u < v$, and by $+1$ and -1 in these positions. In this last case according to Corollary 1(ii) $s_i = t_i \in \{0, i-1\}$ for all $i > u$ and $i \neq v$. We will show that σ and τ differ by two adjacent transpositions.

Let σ' and π' be the permutations in S_{v-1} with the transposition table $s_1s_2\ldots s_{v-1}$ and $t_1t_2\ldots t_{v-1}$. By Lemma 3, σ' and π' differ by an adjacent transposition. Now, the permutations σ'' and π'' in S_v with the transposition table $s_1s_2\ldots s_{v-1}s_v$ and $t_1t_2\ldots t_{v-1}t_v$ differ by two adjacent transpositions. Indeed, σ'' is obtained from σ' by inserting v in the s_vth position from right to left, and τ'' is the permutation obtained from τ' by inserting v in the t_vth position from right to left and, the rightmost position being position zero. Now, since $s_i = t_i \in \{0, i-1\}$ for $i > v$, it results that σ and τ differ as σ'' and τ'', that is by two adjacent transpositions. □

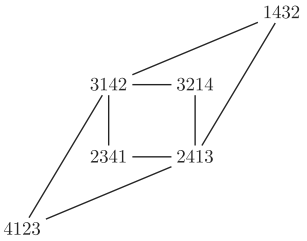


Fig. 1. The graph with vertex set the permutations in S_4 with 3 inversions and two permutations are connected if they differ in three positions.

The previous proposition says that two consecutive permutations in the restriction of $\mathcal{S}_n$ to the set of permutations with a number of inversions lying between two integers differ in at most four positions. Remark that in general there is no more restrictive Gray code for this set. Indeed, an example is given by the graph in Fig. 1 which is not Hamiltonian (recall that a Gray code corresponds to a Hamiltonian path in the induced graph). See also [3] for a (non-Gray code) generating algorithm for the permutations with a fixed number of inversions.

A *cycle* in a permutation $\pi \in S_n$ is a sequence $(a_0a_1\ldots a_{j-1})$ such that $\pi(a_i) = a_{(i+1)\bmod j}$ for all i, $0 \leqslant i \leqslant j-1$. Any permutation is the union of disjoint cycles. For example, the permutation $\pi = 4251763 \in S_7$ is the union of four cycles, namely (41), (2), (573) and (6).

A permutation is called *even* (resp. *odd*) if it has an even (resp. odd) number of inversions. The set of even permutations forms a subgroup of S_n denoted by A_n and it is called the alternating group. Its cardinality is $\frac{n!}{2}$.

Proposition 3. *The restriction of $\mathcal{S}_n$ to the following sets yields Gray codes where two consecutive permutations differ by two adjacent transpositions*:

1. *The set of even permutations*;
2. *The set of odd permutations*;
3. *The set of permutations with an even number of cycles*;
4. *The set of permutations with an odd number of cycles.*

Proof. For the points 1 and 2 the proof is based on the following remark: if $\boldsymbol{s}$ is the successor of $\boldsymbol{r}$, and $\boldsymbol{t}$ that of $\boldsymbol{s}$, in the list $\mathcal{G}_n([0]\times[1]\times\cdots\times[n-1])$, then $\|\boldsymbol{r}\|$ and $\|\boldsymbol{t}\|$ have the same parity. The permutations $\phi(\boldsymbol{r})$ and $\phi(\boldsymbol{t})$ differ by two adjacent transpositions, and have the same parity.

For the points 3 and 4 the proof is similar to that of the points 1 and 2 and is based on the following remark: a transposition (not necessarily adjacent) in a permutation glues two cycles in a single one, or splits one cycle in two ones. □

Acknowledgements

The authors would like to thank one of the anonymous referees for helpful suggestions and for providing the open problem following Corollary 2.

References

[1] M.C. Er, On generating the N-ary reflected Gray code, IEEE Trans. Comput. 33 (8) (1987) 739–741.
[2] G. Blin, M. Crochemore, S. Hamel, S. Vialette, Finding the median of three permutations under the Kendall-τ distance, in: Permutation Patterns 2009 Firenze, 13–17 July 2009, pp. 31–36.
[3] S. Effler, F. Ruskey, A CAT algorithm for generating permutations with a fixed number of inversions, Inform. Process. Lett. 86 (2003) 107–112.
[4] F. Gray, Pulse code communication, U.S. Patent 2632058, 1953.
[5] J. Joichi, D.E. White, S.G. Williamson, Combinatorial Gray codes, SIAM J. Comput. 9 (1980) 130–141.
[6] S.M. Johnson, Generation of permutations by adjacent transpositions, Math. Comp. 17 (1963) 282–285.
[7] P. Klingsberg, A Gray Code for compositions, J. Algorithms 3 (1) (1981) 41–44.
[8] D.E. Knuth, The Art of Computer Programming, vol. 4, Addison–Wesley, 2005.
[9] R. Stanley, Enumerative Combinatorics, vol. 1, Cambridge University Press, Cambridge, England, 1997.
[10] H. Steinhaus, One Hundred Problems in Elementary Mathematics, Dover Publications, 1979 (in Polish 1958).
[11] H.F. Trotter, Perm (Algorithm 115), Comm. ACM 5 (8) (1962) 434–435.
[12] V. Vajnovszki, T. Walsh, A loop-free two-close Gray code algorithm for listing k-ary Dyck words, J. Discrete Algorithms 4 (4) (2006) 633–648.
[13] T. Walsh, Loop-free sequencing of bounded integer compositions, J. Combin. Math. Combin. Comput. 33 (2000) 323–345.
[14] T. Walsh, Generating Gray codes in $O(1)$ worst-case time per word, in: 4th Discrete Mathematics and Theoretical Computer Science Conference, Dijon, France, 7–12 July 2003, in: LNCS, vol. 2731, 2003, pp. 71–88.
[15] H.S. Wilf, Combinatorial Algorithms: An Update, SIAM, Philadelphia, 1989.
[16] S.G. Williamson, Combinatorics for Computer Science, Computer Science Press, Rockville, MD, 1985.

Correspondance des références de cet article avec la bibliographie générale :

[1] → [41]
[2] → [19]
[3] → [40]
[4] → [52]
[5] → [58]
[6] → [57]
[7] → [61]
[8] → [63]
[9] → [85]
[10] → [87]
[11] → [91]
[12] → [98]
[13] → [100]
[14] → [101]
[15] → [75]
[16] → [104]

Liste des tables

2.1 Code de Gray binaire réfléchi de longueur 3 : $\mathscr{G}_3$. 33
2.2 Un exemple de code de Gray plus restrictif proposé par Knuth. 34
2.3 Exemple de code de Gray pour $\mathscr{L}(3,4)$. 38

4.1 Exemple de $\mathcal{C}(p) = \mathcal{S}(B)$ pour $p = 1$ et $p = 2$. 60
4.2 Premières cardinalités de la classe $\mathcal{C}''(p)$. 72

5.1 Liste $\mathscr{G}_3(2\,1\,3)$. 79
5.2 Exemples de 3-compositions en fonction des contraintes fixées. 82
5.3 Un extrait de l'ensemble ordonné $W_{3,9}$ avec $\boldsymbol{b} = 3124532$. 85
5.4 Un extrait de l'ensemble ordonné $W_{3,9}$ avec $\boldsymbol{b} = 3124532$. 87
5.5 Les ensembles $[\boldsymbol{b}]$, $W_{2,2}$, $W_{3,3}$, $W_{2,3}$, $W_{4,4}$, $W_{2,4}$ listés dans l'ordre $\prec$ pour $n = 3$ et $\boldsymbol{b} = 2\,1\,3 \in \mathbb{N}^3$. 90
5.6 Correspondance entre les 5-compositions $\boldsymbol{b}$-bornées par $\boldsymbol{b} = 01234$ de l'entier 5 et les permutations de longueur 5 avec 5 inversions. 92

Liste des figures

2.1 Représentations schématiques de roues codeuses : code lexicographique et code de Gray de taille 4. 30
2.2 Chronogramme : code de Gray de taille 4. 30
2.3 Hypercube Q_4 : code de Gray de taille 4. 31
2.4 Génération des permutations de taille 4 grâce à l'algorithme de Johson-Trotter. 36

3.1 Représentation graphique de la permutation $\sigma = 84625713$. 40
3.2 Les montées dans la permutation $\sigma = 76345821$: (a) montée en 3, (b) montée en 4 et (c) montée en 5. 42
3.3 Différents *run-up* dans la permutation $\sigma = 76345821$: (a) 345 est un *run-up*, (b) 3458 et (c) 7 sont des *run-up* maximaux. 42
3.4 Mise en valeur du motif $\pi = 231$ dans la permutation $\sigma = 84625713$: en (a) la permutation de départ, en (b) la suppression des éléments représentés en blancs et en (c) la normalisation de la séquence obtenue. 45
3.5 Représentation de la stabilité de la relation d'inclusion $\prec$. 46

4.1 De la cellule aux acides de base. 51
4.2 Duplication par segment de longueur 5 avec perte aléatoire. 52
4.3 Duplication miroir avec perte aléatoire de longueur 5. 56
4.4 Duplication miroir complète de la permutation 5421673. Les points entourés sont supprimés lors de la procédure de perte aléatoire. . . . 56
4.5 Les 3 cas non-isomorphes de la démonstration dulemme 4.1 page 56. 57
4.6 Décomposition $\sigma = \tau\tau'$ et la permutation S obtenue après le processus décrit dans lethéorème 4.7 page 57. Nous avons τ = 11 6 5 10 3, τ' = 9 8 1 7 4 2, u_1 est vide, d_1 = 11 6 5, u_2 est vide, d_2 = 10 3, u_3 est vide, d_3 = 9 8 1, u_4 est vide, d_4 = 7 4 2 et D_2 = 10 3, U_2 = 1 8 9 ; D_1 = 11 6 5 ; U_1 = 2 4 7. D'où $S = U_1D_1U_2D_2$ = 2 4 7 11 6 5 1 8 9 10 3. 59
4.7 Algorithme 1 : étape 0. 63

4.8 Algorithme 1 : étape 1. 64
4.9 Algorithme 1 : étape 2. 65
4.10 Algorithme 1 : étape 3. 65
4.11 Algorithme 1 : chemin final. 66
4.12 Étiquetage de $\sigma = 231684597$ avec le code de Gray binaire réfléchi en fonction de ses *run-up* et *run-down*. Exemple d'un chemin de WM-duplications de l'identité vers σ. 68
4.13 Algorithme 2 : étape 0. 68
4.14 Algorithme 2 : étape 1. 68
4.15 Algorithme 2 : étape 2. 69
4.16 Algorithme 2 : étape 3. 69
4.17 Algorithme 2 : chemin final. 69
4.18 Chemins de duplications. 73
4.19 Permutation $\sigma = 3\,14\,2\,10\,4\,15\,5\,17\,11\,9\,6\,8\,13\,12$. 74

5.1 Le graphe avec S_4 comme ensemble des sommets. 95
5.2 Principe de l'algorithme de rejet. 96

Toutes les figures de cette thèse, à l'exception de la figure 4.1 (page 51), sont la propriété exclusive de l'auteur et ne peuvent être reproduites sans son accord formel.

Bibliographie

[1] D. André : « Sur les permutations alternées ». *Journal de Mathématiques Pures et Appliquées*, 7:167–184, 1881.

[2] O. Attie, R. Friedberg et S. Yancopoulos : « Efficient sorting of genomic permutations by translocation, inversion and block interchange ». *Bioinformatics*, 21(16):3340, 2005.

[3] E. Babson et E. Steingrimsson : « Generalized permutation patterns and a classification of the Mahonian statistics ». *Séminaire Lotharingien de Combinatoire*, 44:18, 2000.

[4] S. Bacchelli, E. Barcucci, E. Grazzini et E. Pergola : « Exhaustive generation of combinatorial objects by ECO ». *Acta Informatica*, 40(8):585–602, juil. 2004.

[5] E. Barcucci, A. Bernini et M. Poneti : « From Fibonacci to Catalan permutations ». *arXiv*, math/0612277v1, déc. 2006.

[6] E. Barcucci, A. Del Lungo, E. Pergola et R. Pinzani : « ECO: a methodology for the enumeration of combinatorial objects ». *Journal of Difference Equations and Applications*, 5(4):435–490, 1999.

[7] E. Barcucci, A. Del Lungo et R. Pinzani : « Deco polyominoes, permutations and random generation ». *Theoretical Computer Science*, 159(1):29–42, 1996.

[8] J.-L. Baril : « Gray Code for Cayley permutations ». *Computer Science Journal of Moldova*, 11(2):124–136, 2003.

[9] J.-L. Baril : « Gray code for permutations with a fixed number of cycles ». *Discrete Mathematics*, 307(13):1559–1571, juin 2007.

[10] J.-L. Baril : « More restrictive Gray codes for some classes of pattern avoiding permutations ». *Information Processing Letters*, 109(14):799–804, juin 2009.

[11] J.-L. Baril : « Gray code for permutations with a fixed number of left-to-right minima ». *Ars Combinatoria*, (à paraître), 2011.

[12] J.-L. Baril et V. Vajnovszki : « Gray code for derangements ». *Discrete Applied Mathematics*, 140(1-3):207–221, 2004.

[13] J.-L. Baril et R. Vernay : « Whole mirror duplication-random loss model and pattern avoiding permutations ». *In Pattern Permutations 2009*, Firenze, Italy, juil. 2009. Dipartimento Sistemi e Informatica.

[14] J.-L. Baril et R. Vernay : « Whole mirror duplication-random loss model and pattern avoiding permutations ». *Information Processing Letters*, 110(11):474–480, mai 2010.

[15] B. Bauslaugh et F. Ruskey : « Generating alternating permutations lexicographically ». *BIT Numerical Mathematics*, 30(1):17–26, jan. 1990.

[16] A. Bergeron, J. Mixtacki et J. Stoye : « A unifying view of genome rearrangements ». *Algorithms in Bioinformatics*, 4175:163–173, 2006.

[17] A. Bernini, E. Grazzini, E. Pergola et R. Pinzani : « A general exhaustive generation algorithm for Gray structures ». *Acta Informatica*, 44(5):361–376, 2007.

[18] J. R. Bitner, G. Ehrlich et E. M. Reingold : « Efficient generation of the binary reflected Gray code and its applications ». *Communications of the ACM*, 19(9): 517–521, sep. 1976.

[19] G. Blin, M. Crochemore, S. Hamel et S. Vialette : « Finding the median of three permutations under de Kendall-τ distance ». *Pattern Permutations 2009*, Firenze:31–36, juil. 2009.

[20] G. Bourque et P. Pevzner : « Genome-scale evolution: reconstructing gene orders in the ancestral species ». *Genome Research*, 12(1):26, 2002.

[21] M. Bouvel : « Quelques problèmes combinatoires et algorithmiques sur les classes de permutations ». Thèse de doctorat, Université Paris Diderot - Paris VII, Paris, déc. 2009.

[22] M. Bouvel et E. Pergola : « Posets and Permutations in the duplication-loss model ». *Pure Math. Appl.*, 19(2-3):71–80, sep. 2008.

[23] M. Bouvel et E. Pergola : « Posets and permutations in the duplication-loss model: minimal permutations with d descents ». *Theoretical Computer Science*, 411(26-28):2487–2501, juin 2010.

[24] M. Bouvel et D. Rossin : « A variant of the tandem duplication-random loss model of genome rearrangement ». *Theoretical Computer Science*, 410(8-10): 847–858, mar. 2009.

[25] M. D. V. Braga : « On Sorting Genomes with DCJ and Indels ». *Comparative Genomics*, p. 62–73, 2011.

[26] S. Bérard, A. Bergeron, C. Chauve et C. Paul : « Perfect sorting by reversals is not always difficult ». *IEEE ACM Transactions on Computational Biology and Bioinformatics*, 4(1):4–16, 2007.

[27] S. Bérard, C. Chauve et C. Paul : « A more efficient algorithm for perfect sorting by reversals ». *Information processing letters*, 106(3):90–95, 2008.

[28] E. R. Canfield et H. Wilf : « Counting permutations by their alternating runs ». *Journal of Combinatorial Theory*, (115):213–225, 2008.

[29] C. C. Chang, C. Y. Chen et H. Y. Chen : « Symbolic Gray code as a data allocation scheme for two-disc systems ». *The Computer Journal*, 35(3):299, 1992.

[30] C. C. Chang, M. W. Du et R. C. T. Lee : « Symbolic Gray code as a perfect multiattribute hashing scheme for partial match queries ». *Software Engineering, IEEE Transactions on*, 8(3):235–249, 1982.

[31] P. J. Chase : « Combination generation and graylex ordering ». *Congressus Numerantium*, 69(19):215–242, 1989.

[32] K. Chaudhuri, K. Chen, R. Mihaescu et S. Rao : « On the tandem duplication-random loss model of genome rearrangement ». *In Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, p. 564–570. ACM New York, NY, USA, 2006.

[33] H. D. Chen, W. L. Fan, S. G. Kong, H. Lee, B. Zheng et N. Zhou : « Inverse symmetry in genomes and whole-genome inverse duplication ». *In International Bioinformatics Workshop (IBW2008)*. Yunnan University, Kunming, Yunnan, China, juin 2008.

[34] M. C. Chen et R. C. T. Lee : « Sorting by transpositions based on the first increasing substring concept ». *In BIBE*, p. 553, Washington, DC, USA, 2004. IEEE Computer Society.

[35] M. Cohn et J. P. Robinson : « Counting sequences ». *IEEE Transactions on Computers*, 30(1):17–23, 1981.

[36] G. R. Crabtree, A. J. Fornace, J. A. Kant, O. W. McBride, D. Saxe et M. I. Simon : « Evolution and organization of the fibrinogen locus on chromosome 4: gene duplication accompanied by transposition and inversion ». vol. 82, p. 2344. National Academy of Sciences of the United States of America, 1985.

[37] Y. Diekmann, M.-F. Sagot et E. Tannier : « Evolution under Reversals: Parsimony and Conservation of Common Intervals ». 4(2):301–309, 2007.

[38] P.-T. Do : « Arbres de génération et génération exhaustive ». Thèse de doctorat, Université de Bourgogne, juil. 2008.

[39] M. Dukes, M. Flanagan, T. Mansour et V. Vajnovszki : « Combinatorial Gray codes for classes of pattern avoiding permutations ». *Theoretical Computer Science*, (396):35–49, mai 2008.

[40] S. Effler et F. Ruskey : « A CAT algorithm for generating permutations with a fixed number of inversions ». *Information Processing Letters*, (86):107–112, mar. 2003.

[41] M. C. Er : « On generating the n-ary reflected Gray codes ». *IEEE Transactions on computers*, 33(8):739–741, 1984.

[42] M. Figeac et J.-S. Varre : « Sorting by Reversals with Common Intervals ». 3240: 26–37, 2004.

[43] P. Flajolet et L. Ramshaw : « A note on Gray code and odd-even merge ». *SIAM Journal on Computing*, 9:142, 1980.

[44] D. Foata : « On the Netto inversion number of a sequence ». *Proceedings of the American Mathematical Society*, 19:236–240, 1968.

[45] D. Foata : « Rearrangements of words », vol. 17, chap. 10. Encyclopedia of Mathematics and its Applications, 1983.

[46] D. Foata et M. P. Schutzenberger : « Major index and inversion number of permutations ». *Mathematische Nachrichten*, 83(1):143–159, 1978.

[47] M. Gardner : « The curious properties of the Gray code and how it can be used to solve puzzles ». *Scientific American*, 227:106–109, aoû. 1972.

[48] S. Getu, L. W. Shapiro et W. J. Woan : « Runs, slides and moments ». *SIAM Journal on Algebraic and Discrete Methods*, 4:459, 1983.

[49] E. Gilbert : « Gray codes and paths on the n-cube ». *Bell System Technical Journal*, 37:815–826, 1958.

[50] L. Goddyn, G. M. Lawrence et E. Nemeth : « Gray codes with optimized run lengths ». *Utilitas Mathematica*, 34:179–192, 1988.

[51] H. W. Gould : « The q-Stirling numbers of first and second kinds ». *Duke mathematical journal*, 28(2):281–289, 1961.

[52] F. Gray : « Pulse Code Communication ». *US Patent 2632058*, mar. 1953.

[53] L. A. Gros : « La théorie du baguenodier ». 1872.

[54] O. Guibert : « Combinatoire des permutations à motifs exclus en liaison avec mots, cartes planaires et tableaux de Young ». Thèse de doctorat, Université de Bordeaux I, Bordeaux, déc. 1995.

[55] M. T. Hallett et J. Lagergren : « New algorithms for the duplication-loss model ». *In Proceedings of the fourth annual international conference on Computational molecular biology*, p. 138–146. ACM New York, NY, USA, 2000.

[56] F. Heath : « Origins of the binary code ». *Scientific American*, 227(2):76–83, 1972.

[57] S. M. Johnson : « Generation of permutations by adjacent transposition ». *Mathematics of Computation*, (17):282–285, juil. 1962.

[58] J. T. Joichi, D. White et S. G. Williamson : « Combinatorial gray codes ». *SIAM Journal on Computing*, 9:130–141, 1980.

[59] S. Kitaev : « Introduction to partially ordered patterns ». *Discrete Applied Mathematics*, (155):929–944, 2007.

[60] S. Kitaev et A. Pyatkin : « On avoidance of V- and A-patterns in permutations ». oct. 2006.

[61] P. Klingsberg : « A Gray Code for compositions ». *Journal of Algorithms*, 3(1):41–44, mai 1981.

[62] D. Knuth : « Sorting and searching ». *The art of computer programming*, 3, 1982.

[63] D. Knuth : « Generating all combinations and partitions ». *The art of computer programming*, 4(3), juil. 2005.

[64] D. Knuth : « Generating all tuples and permutations ». *The art of computer programming*, 4(2), fév. 2005.

[65] A. Labarre : « A new tight upper bound on the transposition distance ». *In Algorithms in Bioinformatics*, p. 216–227, Mallorca, Spain, oct. 2005. 5th International Workshop WABI 2005.

[66] A. Labarre : « New Bounds and Tractable Instances for the Transposition Distance ». *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 3(4):380–394, 2006.

[67] R. M. Losee : « A Gray code based ordering for documents on shelves: Classification for browsing and retrieval ». *Journal of the American Society for Information Science*, 43(4):312–322, 1992.

[68] M. Lothaire : « Combinatorics on words ». Cambridge University Press, 1997.

[69] J. M. Lucas, F. Ruskey et D. R. van Baronaigien : « On rotations and the generation of binary trees ». *Journal of Algorithms*, 15(3):343–366, 1993.

[70] J. E. Ludman : « Gray code generation for MPSK signals ». *Communications, IEEE Transactions on*, 29(10):1519–1522, 2002.

[71] P. A. Mac Mahon : « The indices of permutations and the derivation therefrom of functions of a single variable associated with the permutations of any assemblage of objects ». *American Journal of Mathematics*, 35(3):281–322, 1913.

[72] P. A. Mac Mahon : « Combinatory Analysis », vol. 1. Cambridge University Press, 1915.

[73] P. A. Mac Mahon : « Combinatory Analysis », vol. 2. Cambridge University Press, 1915.

[74] T. Mansour et S. Yan : « Minimal permutations with d descents ». *European Journal of Combinatorics*, 31(5):1445–1460, fév. 2010.

[75] A. Nijenhuis et H. Wilf : « Combinatorial algorithms ». Academic Press, 1978.

[76] R. Nussinov : « Some indications for inverse DNA duplication ». *Journal of theoretical biology*, 95(4):783, 1982.

[77] D. Richards : « Data compression and Gray-code sorting ». *Information Processing Letters*, 22(4):201–205, 1986.

[78] R. G. Rieper et M. Zeleke : « Valleyless Sequences ». *arXiv*, math/0005180, mai 2000.

[79] F. Ruskey : « Generating t-ary trees lexicographically ». *SIAM Journal on Computing*, 7:424, 1978.

[80] F. Ruskey : « Adjacent interchange generation of combination ». *Journal of Algorithms*, 9(2):162–180, 1988.

[81] D. Sankoff : « Gene and genome duplication ». *Current Opinion in Genetics and Development*, 11:681–684, 2001.

[82] D. Sankoff : « Short inversions and conserved gene clusters ». p. 164–167. ACM symposium on Applied computing, 2002.

[83] C. Savage : « A survey of combinatorial Gray codes ». *SIAM review*, 39(4):605–629, 1997.

[84] N. J. A. Sloane : « The on-line encyclopedia of integer sequences ». *http://oeis.org*, 1996.

[85] R. Stanley : « Enumerative Combinatorics », vol. 1. Cambridge University Press, 1997.

[86] E. Steingrimsson : « Statistics on ordered partitions of sets ». *arXiv*, math/0605670, avr. 2007.

[87] H. Steinhaus : « One hundred problems in elementary mathematics ». Dover Pubns, 1979.

[88] G. Stibitz : « Binary counter ». *US Patent 2307868*, jan. 1943.

[89] J. Stirling : « The Differential Method ». Printed for E. Cave, 1749.

[90] N. M. Temme : « Asymptotic estimates of Stirling numbers ». *Stud. Appl. Math*, 89(3):233–243, 1993.

[91] H. F. Trotter : « Algorithm 115: Perm ». *Communications of the ACM*, 5(8):434–435, 1962.

[92] V. Vajnovszki : « Le codage des arbres binaires ». *Computer Science Journal of Moldova*, 3(2):194–209, 1995.

[93] V. Vajnovszki : « Generating a Gray code for p-sequences ». *Journal of Mathematical Modelling and Algorithms*, 1(1):31–41, mar. 2002.

[94] V. Vajnovszki : « Gray visiting motzkins ». *Acta Informatica*, 38(11):793–811, 2002.

[95] V. Vajnovszki : « Generating involutions, derangements, and relatives by ECO ». *Discrete Mathematics and Theoretical Computer Science*, 12(1):109, 2010.

[96] V. Vajnovszki et R. Vernay : « Restricted compositions and permutations: from old to new Gray codes ». *In 8th French Combinatorial Conference*, Orsay, Paris, juin 2010. Université Paris Sud XI.

[97] V. Vajnovszki et R. Vernay : « Restricted compositions and permutations: from old to new Gray codes ». *Information Processing Letters*, 111(13):650–655, juil. 2011.

[98] V. Vajnovszki et T. Walsh : « A loop-free two-close Gray-code algorithm for listing k-ary Dyck words ». *Journal of Discrete Algorithms*, 4(4):633–648, déc. 2006.

[99] M. Wachs et D. White : « p,q-Stirling numbers and set partition statistics ». *Journal of Combinatorial Theory*, 56(1):27–46, 1991.

[100] T. Walsh : « Loop-free sequencing of bounded integer compositions ». *Journal of Combinatorial Mathematics and Combinatorial Computing*, 33:323–345, 2000.

[101] T. Walsh : « Generating Gray codes in $O(1)$ worst-case time per word ». *In 4th international conference on Discrete mathematics and theoretical computer science*, p. 73–88, Dijon, France, juil. 2003. LNCS 2731.

[102] J. West : « Generating trees and the Catalan and Schröder numbers ». *Discrete Mathematics*, 146(1–3):246–262, 1995.

[103] H. Wilf : « Combinatorial algorithms: an update ». Philadelphia, 1989. SIAM.

[104] S. G. Williamson : « Combinatorics for computer science ». Dover Pubns, 1985.

Étude d'objets combinatoires
Applications à la bio-informatique

—

Thèse de doctorat

—

Rémi Vernay
remi.vernay@u-bourgogne.fr

—

Achevé d'imprimer le 20 juin 2011 à Dijon (Bourgogne)

—

Imprimé en France

Study of Combinatorial Objects Applications to Bioinformatics

PhD Thesis – Computer Science – 29 June 2011

This thesis considers classes of combinatorial objects that model data in bioinformatics. We have studied two methods of mutation of genes within the genome: duplication and inversion.

At first, we study the problem of the whole mirror duplication-random loss model in terms of pattern avoiding permutations. We prove that the class of permutations obtained with this method after p duplications from the identity is the class of permutations avoiding alternating permutations of length 2^p+1. We also enumerate the number of duplications that are necessary and sufficient to obtain any permutation of length n from the identity. We also suggest two efficient algorithms to reconstruct two different paths between the identity and a specified permutation. Finally, we give related results on other classes nearby.

The restriction of the order relation $\prec$ induced by the reflected Gray code for the sets of compositions and bounded compositions gives new Gray codes for these sets. The order relation $\prec$ restricted to the set of bounded compositions of an interval also yields a Gray code. The set of bounded n-compositions of an interval simultaneously generalizes product set and compositions of an integer, and so $\prec$ puts under a single roof all these Gray codes. We re-express Walsh's and Knuth's Gray codes for (bounded) compositions of an integer in terms of a unique order relation, and so Walsh's Gray code becomes a sublist of Knuth's code, which in turn is a sublist of the Reflected Gray Code.

Keywords: combinatorics, permutations, compositions of integers, Gray codes, duplications, inversions, bioinformatics.

Rémi Vernay
remi.vernay@u-bourgogne.fr

Étude d'objets combinatoires
Applications à la bio-informatique

Thèse de doctorat – Informatique – 29 juin 2011

Cette thèse porte sur des classes d'objets combinatoires, qui modélisent des données en bio-informatique. Nous étudions notamment deux méthodes de mutation des gènes à l'intérieur du génome : la duplication et l'inversion.

Nous étudions d'une part le problème de la duplication-miroir complète avec perte aléatoire en termes de permutations à motifs exclus. Nous démontrons que la classe de permutations obtenue avec cette méthode après p duplications à partir de l'identité est la classe de permutations qui évite les permutations alternées de longueur $2^p + 1$. Nous énumérons également le nombre de duplications nécessaires et suffisantes pour obtenir une permutation quelconque de longueur n à partir de l'identité. Nous proposons également deux algorithmes efficaces permettant de reconstituer deux chemins différents entre l'identité et une permutation déterminée. Nous donnons enfin des résultats connexes sur d'autres classes proches.

La restriction de la relation d'ordre $\prec$ induite par le code de Gray réfléchi à l'ensemble des compositions et des compositions bornées induit de nouveaux codes de Gray pour ces ensembles. La relation d'ordre $\prec$ restreinte à l'ensemble des compositions bornées d'un intervalle fournit encore un code de Gray. L'ensemble des n-compositions bornées d'un intervalle généralise simultanément l'ensemble produit et l'ensemble des compositions d'un entier et donc la relation $\prec$ définit de façon unifiée tous ces codes de Gray. Nous réexprimons les codes de Gray de Walsh et Knuth pour les compositions (bornées) d'un entier à l'aide d'une unique relation d'ordre. Alors, le code de Gray de Walsh pour des classes de compositions et de permutations devient une sous-liste de celui de Knuth, lequel est à son tour une sous-liste du code de Gray réfléchi.

Mot clés : combinatoire, permutations, compositions d'entiers, codes de Gray, duplication, inversion, bio-informatique.

Rémi Vernay
remi.vernay@u-bourgogne.fr