Aspects algorithmiques de la combinatoire

Robert Cori

Table des matières

Première partie

Les Mots

CHAPITRE 1

Combinatoire des mots et aspects algorithmiques

1. Notions élémentaires sur les mots

1.1. Définitions. Dans ce chapitre on considère un *alpahabet* A, c'est un ensemble fini dont les éléments sont appelés des *lettres*. Un mot f sur l'alphabet A est une suite finie de lettres notée $f_1 f_2 \dots f_n$, la *longueur* n du mot f est notée $|f|$. L'ensemble de tous les mots sur l'alphabet A est noté A^* La *concaténation* de deux mots $f = f_1 f_2 \dots f_n$ et $g = g_1 g_2 \dots g_m$ est le mot obtenu en écrivant g à la suite de f soit

$$fg = f_1 f_2 \dots f_n g_1 g_2 \dots g_m$$

Ce mot a pour longueur $n + m$.

Pour un mot f et entier p, f^p désigne la concaténation de p fois le mot f.

Il est facile de vérifier que la concaténation est une opération associative ayant pour élément neutre le mot vide de longueur 0 que l'on notera dans la suite ϵ. Ainsi A^* est un monoïde pour la concaténation.

On peut vérifier par contre que la concaténantion n'est pas commutative comme le montre l'exemple $f = a, g = b, \quad ab \neq ba$; on peut toutefois caractériser les cas où $fg = gf$ par la proposition suivante :

PROPOSITION 1. *Pour deux mots f et g, on a $fg = gf$ si et seulement s'ils sont puissance d'un même mot w. C'est à dire s'il existe un mot w et deux entiers p, q tels que $f = w^p, g = w^q$.*

Preuve: Cette preuve se fait par récurrence sur la longueur de fg.

On peut, sans perte de généralité supposer que $|f| < |g|$,

Lorsque la longueur de fg est 1, on a $f = \epsilon, g = a$ ceci donne alors $w = a, p = 0, q = 1$. Si la longueur de fg est supérieure à 1, on a alors $g = fg'$ ce qui donne :

$$ffg' = fg'f$$

soit en simplifiant :

$$fg' = g'f$$

alors par récurrence il existe un mot w et des entiers p, q' tels que $f = w^p, g' = w^{q'}$, ce qui donne le résultat en posant $q = p + q'$. □

La recherche de *facteurs* est un des problèmes algorithmiques les plus considérés sur les mots. En effet, tous les systèmes de traitement de texte offrent cette possibilité de recherche. Un mot g est *facteur* d'un mot f si f s´écrit $f = f'gf$". Ainsi *aba* est un facteur de *abbaabaab* par contre *bab* n'en est pas un.

1.2. Conjugaison. Une relation importante sur A^* est la *relation de conjugaison*. Deux mots f et g sont conjugués s'il existe u et v tels que

$$f = uv \quad g = vu$$

On vérifie facilement que la conjugaison est une relation d'équivalence. Un mot f qui n'est pas puissance d'un autre mot a exactement $|f|$ conjugués différents. Un tel mot est dit *primitif*. On peut noter que tout mot f est puissance d'un mot primitif ; dans ce cas si $f = g^p$, avec g primitif, alors le nombre de conjugués de f est $|g|$.

2. Enumération, génération aléatoire

Le nombre de mots de longueur n sur un alphabet à k lettres est égal à

$$k^N$$

en effet, pour chaque f_i on doit choisir une lettre parmi les k lettres de A.

PROPOSITION 2. *Le nombre p_n de mots primitifs de longueur n vérifie la relation de récurrence*

$$k^n = \sum_{d|n} p_d$$

Preuve: Ceci résulte immédiatement de la remarque sur le fait qu'un mot non primitif de la longueur est puissance d'un mot primitif dont la longueur divise n.

□

Lorsqu'on se limite à l'alphabet $\{a, b\}$, le nombre de mots ayant p lettres a et q lettres b est donné par le coefficient binomial :

$$\binom{p+q}{p} = \frac{(p+q)!}{p!q!}$$

On remarque que le nombre de tels mots qui commencent par a est

$$\frac{(p+q-1)!}{(p-1)!q!} \quad = \quad \frac{p}{p+q}\binom{p+q}{p}$$

Dit d'une autre façon, la probabilité pour qu'un tel mot commence par a est égale à

$$\frac{p}{p+q}$$

Ceci implique que pour tirer au hasard un mot qui contient p lettres a et q lettres b, on peut déterminer la premiere lettre en effectuant un tirage aléatoire en donnant comme probabilité $\frac{p}{p+q}$ pour la lettre a et $\frac{q}{p+q}$ pour la lettre b. On poursuit alors récursivement en tirant au hasard un mot de longueur $p+q-1$ contenant p ou $p-1$ lettres a suivant le résltat du tirage effectué. Ceci donne par exemple :

```
char[] genAlea (int p, int q, char[] x){
     if (p + q  == 0) return x;
     else
          if (alea(p+q) < p){
              x[p+q-1] = 'a';
```

```
            return genAlea(p-1, q, x);
        }
        else {
            x[p+q-1] = 'b';
            return genAlea(p, q-1, x);
        }
}
```

3. Mots sans carré

3.1. exposé du problème. Un mot f non vide est un *carré* s'il s'écrit gg. Un mot contient un carré si un de ses facteurs est un carré, s'il ne contient pas de carré le mot est dit *sans carré*. La recherche de mots sans carré est un problème ludique considéré par de nombreux auteurs. Il est clair que sur un alphabet à deux lettres les seuls mots sans carré sont $\epsilon, a, b, ab, ba, aba, bab$, se pose donc naturellement la question de savoir s'il existe des mots sans carré de longueur arbitraire sur un alphabet à trois lettres. Voici un moyen d'en construire :

3.2. suite de Thue Morse. On commence par définir cette suite formée de mots sur un alphabet à deux lettres $\{, 0, 1\}$, cette suite possède de nombreuses propriétés et a été largement exploitée en combinatoire et en arithmétique.

Elle peut être définie de la manière suivante $t_1 = 0$ et $t_{i+1} = \mu(t_i)$ où μ est le morphisme qui remplace 0 par 01 et 1 par 10. Ainsi les premiers mots de cette suite sont :

$$t_1 = 0, t_2 = 01, t_3 = 0110, t_4 = 01101001, t_5 = 0110100110010110$$

Il est assez facile de voir que chaque t_i ne contient pas les facteurs $111, 000$

On montre aussi que le mot $u - i$ tel que $\phi(u_i) = t_i$, où ϕ est le morphisme donné par :

$$\phi(a) = 011, \phi(b) = 01, \phi(c) = 0$$

est pour tout i un mot sans carré. On obtient par exemple

$$u_1 = c, u_2 = b, u_3 = ac, u_4 = abc, u_5 = abcacbac$$

4. Code de Gray

On appelle, abusivement, code de Gray une suite de mots formée de tous les mots de longeur n de façon telle que deux mots consécutifs de cette suite diffèrent en une seule position. Un code de Gray des mots de longeur 2 sur un alphabet à 3 lettres est par exemple

$$aa, ab, ac, bc, bb, ba, ca, cb, cc$$

Pour les mots de longeur 3 sur un alphabet à deux lettres on peut proposer :

$$000, 001, 011, 010, 110, 111, 101, 100$$

5. Algorithmes de recherche de facteur

5.1. Analyse de l'algorithme naïf. Il est assez étonnant de constater que l'algorithme naïf de recherche de facteur donné ci-dessous est de fait très efficace en moyenne :

On se donne un mot f de longueur n dont les lettres sont placées dans un tableau noté `f` et un mot g de longueur $m < n$ on souhaite afficher toutes les positions i où débute dans f le facteur g ; c'est à dire telle que $f_i f_{i+1} \dots f_{i+m-1} = g_0 g_1 \dots g_{m-1}$.

```
for (int i = 0; i <= n -m; ++i){
     j = 0;
     while (j < m && g[j] == f[i+j]) j++;
     if (j == m) affiche(i);
     }
```

Cet algorithme consiste à se placer en toutes les positions possibles pour i et à tester l'égalité ci-dessus en s'arrêtant dès que l'on constate la présence de lettres différentes dans f et g. On note $C(f, g)$ le nombre de comparaisons de lettres effectuées par cet algorithme. Par exemple pour $f = 01010100011$ et $g = 011$ le nombre de comparaisons de lettres est $3+1+3+1+3+1+2+2+3 = 19$ On remarque que ce nombre est égal à $n - m + 1$ augmenté du nombre d'occurences de g_o dans $f_0 f_1 \dots f_{n-m}$, du nombre d'occurences du facteur $g_0 g_1$ dans $f_0 f_1 \dots f_{n-m+1}$ et aisni de suite jusqu'au nombre d'occurences de $g_0 g_1 \dots g_{m-2}$ dans $f_0 f_1 \dots f_{n-2}$. dans notre exemple $n = 11, m = 3$ le nombre d'occurences de g_0 est 6 et celui de 01 est 4.

PROPOSITION 3. *La somme des $C(f, g)$ pour tous les mots f de longeur n est indépendante de g et vaut :*

$$\sum_{|f|=n} C(f, g) = (n - m + 1)(k^n + k^{n-1} + \dots + k^{n-m})$$

5.2. Algorithme de Knuth Morris et Pratt.

6. Mots de Lyndon

6.1. Défintions. On rappelle la définition formelle de l'ordre lexicographique, ordre correspondant au rangement des mots dans un dictionnaire.

DÉFINITION 1. *Pour deux mots f et g on dit que f est antérieur à g pour l'ordre lexicographique et on note $f \preceq g$ si f est facteur gauche de g ou si les premières lettres de f et g qui diffèrent sont telles que f_i précède g_i. Soit :*

$$g = fg'$$

ou

$$f = hf_i f' \quad g = hg_i g', \quad f_i \preceq g_i$$

Ainsi *bal* $\preceq$ *ballon* $\preceq$ *baluchon*

Un mot de Lyndon est un mot qui est plus petit, pour l'ordre lexicographique, que tous ses facteurs droits. Les mots de Lyndon de longueur inférieure ou égale à 4 sur un alaphabet à deux lettres sont :

$$a, b, ab, aab, abb, aaab, abbb, aabb$$

6.2. Enumération. Le résultat suivant permet d'énumérer les mots de Lyndon :

PROPOSITION 4. *Tout mot primitif admet un conjugué et un seul qui est un mot de Lyndon.*

Preuve: Soit f un mot primitif et soit g le plus petit de ses conjugués, ainsi $f = uv$ et $g = vu$. On montre que g est un mot de Lyndon ; en effet un facteur droit g" de g est ou bien facteur droit de u ou bien de la forme $v'u$ où v' est facteur droit de v. Dans le premier cas $u = u'g$" et $g \prec g$"vu' implique $g \prec g$", dans le second cas $g \prec$ AFAIRE □

On déduit alors :

COROLLAIRE 1. *Le nombre $L_{k,n}$ de mots de Lyndon sur un alphabet à k lettres satisfait la relation :*

$$k^n = \sum_{d|n} dL_{d,k}$$

Preuve: En effet le membre gauche énumère tous les mots de longueur n, d'après la proposition précédente $nL_{n,k}$ est le nombre de mots primitifs de longueur n. Il nous reste à montrer que pour tout diviseur d de n tel que $d < n$, le nombre de mots de longueur n puissance d'un mot primitif de longueur d est $dL_{d,k}$. Or ceci résulte de l'observation suivante : si $f = u^p$ où u est primitif de longueur d, alors f admet d conjugués dont un seul est puissance d'un mot de Lyndon de longueur d.

□

6.3. Propriétés remarquables. L'algorithme suivant donne tous les mots de Lyndon de longueur plus petite que n et ceci dans l'ordre lexicographique :

Soit u un mot de Lyndon de longueur $m \leq n$, soit w le mot de longuer n facteur gauche de $u^{\frac{n}{m}}$, on écrit $w = w'xz^k$, où z est la plus grande lettre de A, $x \neq z$ une lettre de A et $k \geq 0$ un entier. On pose alors

$$Suiv(u) \quad = \quad w'y$$

où y est la lettre qui suit x dans l'alphabet A.

Par exemple soit $u = aabac$, $A = \{a, b, c\}$ et $n = 7$; alors $w = aabacaa$ et $Suiv(u) = aabacab$, de la même façon :

$$Suiv(aabacab) = aabacac, \quad Suiv(aabacac) = aabacb)$$

PROPOSITION 5. *Soit u un mot de Lyndon de longueur inférieure ou égale à n, soit $v = Suiv(u)$ le mot obtenu par l'algorithme décrit plus haut. Alors v est le plus petit mot de Lyndon, plus grand que u et de longueur inférieure ou égaleà n.*

6.4. Mots universels. Un mot f est *n-universel* si tout mot de longeur n est un facteur de f, ainsi *abbaa* est 2-universel pour l'alphabet $\{a, b\}$. Il est assez facile de voir qu'un mot n universel pour un alphabet à k lettres est de longueur au moins $k^n + n - 1$ car un mot de longueur m a $m - n + 1$ facteurs de longueur n.

La procédure suivante permet d'obtenir un mot universel de longueur exactement $k^n + n - 1$:

Soit $\lambda_1, \lambda_2, \ldots, \lambda_N$ la suite des mots de Lyndon de longueur divisible par n que l'on suppose écrite dans l'ordre lexicographique on a alors

THEORÈME 1. *Le mot*

$$\lambda_1, \lambda_2, \ldots \lambda_N a^{n-1}$$

où a est la plus petite lettre de l'alpahabet, est un mot n-universel.

7. Mots de parenthèses

7.1. Défintions.

7.2. Enumération.

7.3. Génération aléatoire.

8. Compléments

CHAPITRE 2

Deuxième partie

Permutations

CHAPITRE 3

Inversions et analyse des algorithmes de tri

1. Notions élémentaires sur les permutations

1.1. Définitions. Une *permutation* est une suite $a_1, a_2, \ldots, a_n$ d'entiers tous distincts et tous compris entre 1 et n. Ainsi pour tout $1 \leq i \leq n$, il existe un j et un seul tel que $a_j = i$, ceci montre qu'une permutation peut être considérée comme une bijection de $\{1, 2, \ldots, n\}$ dans lui même.

De cette façon, pour une permutation $\alpha = a_1, a_2, \ldots, a_n$ on définit une permutation inverse $\alpha^{-1} = b_1, b_2, \ldots, b_n$ telle que $b_i = j$ si et seulement si $a_j = i$.

Dans la suite une permutation α sera représentée par un tableau d'entiers `a` tel que `a[i]` est égal à a_i, noter que dans cette représentation pour des langages qui commencent les tableaux à 0, cela fait un espace de perdu, mais correspond mieux aux conventions habituelles.

Le nombre de permutations sur n est $n!$, un moyen d'engendrer une permutation au hasard consiste à appliquer la procédure suivante :

```
for (int i = 1; i <= n; ++i) a[i] = i;
for (int i = n ; i > 1; i--){
       int k = 1+ random(i);
       //donne un nombre entier compris entre 1 et i
       t= a[k]; a[k] = a[i]; a[i] = t; //echange de a[k] et a[i]
     }
```

PROPOSITION 6. *L'algorithme ci dessus engendre uniformément au hasard toutes les permutations de* $\mathbb{S}_n$

Preuve: Cet algorithme engendre une suite de nombres $k_n, k_{n-1}, \ldots, k_3, k_2$ telle que $1 \leq k_i \leq i$ et ceci uniformément au hasard. Il est facile de vérifier que si deux suites $k_n, k_{n-1}, \ldots, k_2, k_2$ et $k'_n, k'_{n-1}, \ldots, k'_3, k'_2$ sont différentes alors les deux permutations α et α' qu'elles permettent de construire seront aussi différentes : si j et la première valeur où ces deux suites différent les valeurs de `a[j]` et `a'[j]` seront différentes. La conclusion résulte que le nombres de suites construites est égal à $n!$, ainsi toute permutation peut être ainsi obtenue. □

Une *inversion* dans une permutation est un couple `(p,q)` tel que si $p = a_i,\ q = a_j$ on a :

$$i < j, \quad \text{et} \quad p > q$$

PROPOSITION 7. *Le nombre d'inversions de* α *est le même que celui de* α^{-1}

Preuve: Si (p, q) est une inversion de α, soit i et j tels que $p = a_i, q = a_j$, alors (j, i) est une inversion de α^{-1}. □

Par exemple pour $\alpha = 3, 2, 5, 7, 1, 9, 6, 8, 4$ les inversions sont :

$$(3,2),(3,1),(2,1),(5,1),(5,4),(7,1),(7,6)(7,4),(9,6),(9,8),(9,4),(6,4),(8,4)$$

soit un total de 13 inversions, l'inverse de α est $5, 2, 1, 9, 3, 7, 4, 8, 6$ qui possede $4+1+0+5+0+2+0+1+0 = 13$ inversions

2. Tableaux des inversions

Pour une permutation α on note $t_\alpha[p]$ le nombre d'entiers plus petits que p situés après p dans α, c'est le nombre d'inversions de α de la forme p, q. Pour l'exemple $\alpha = 3, 2, 5, 7, 1, 9, 6, 8, 4$ on obtient

$$t_\alpha = [0, 1, 2, 0, 2, 1, 3, 1, 3]$$

il est assez facile de vérifier que

$$0 \leq t_\alpha[i] \leq i - 1$$

A partir de tout tableau t satisfaisant cette condition on construit une permutation β tel que $t_\beta = t$ par l'algoritme suivant :

On construit itérativement la permutation en insérant à chaque étape l'entier i dans une permutation contenant $1, 2, \ldots i-1$; ceci doit être fait en laissant à droite de i, $t_\beta[i]$ éléments, ceux-ci devront donc être décalés d'une place vers la droite. Ce qui s'écrit de façon plus précise par :

```
beta[1] = 1;
for (int i = 2; i <= n; ++i) {
      for (int j = 1; j <= t[i] ;++j)
           beta[i+1-j] = beta[i-j];
      beta[i-t[i]] = i;
}
```

Comme le nombre de tableaux satisfaisant les conditions données est $n!$ on voit bien que l'on a défini une bijection entre les permutations et leurs tableaux d'inversions.

3. Enumération

3.1. Formule de récurrence. On souhaite compter le nombre de permutations contenant k inversions, on note ce nombre par $I_{n,k}$, la plus grande valeur de k pour un n donné est :

$$N = \sum_{i=0}^{n-1} i \quad = \quad \frac{n(n-1)}{2}$$

On vérifie assez facilement les valeurs suivantes de $I_{n,k}$

- $I_{n,0} = 1$
- $I_{n,N} = 1$
- $I_{n,k} = I_{n,N-k}$

PROPOSITION 8. *Les $I_{n,k}$ satisfont la relation de récurrence suivante, dans cette formule $I_{n,p}$ est supposé égal à 0 si $p < 0$:*

$$I_{n,k} = \sum_{j=0}^{n-1} I_{n-1,k-j}$$

Preuve: Il s'agit de compter le nombre de tableaux de taille n satisfaisant la condition

$$0 \leq t[i] \leq i-1$$

et dont la somme des $t[i]$ est k. Or chacun de ces tableaux est formé d'un entier $j = t[n]$ compris entre 0 et $n-1$ et d'un tableau de taille $n-1$ dont la somme des valeurs est $k-j$; ce qui donne le résultat. □

3.2. Polynômes énumérateurs. Pour faciliter la détermination des nombres $I_{n,k}$ on introduit le polynôme :

$$I(x) = \sum_{k=0}^{N} I_{n,k} x^k$$

On traduit la proposition précédente sur les polynômes par :

COROLLAIRE 2. *Les polynômes $I_n(x)$ satisfont la relation de récurrence :*

$$I_n(x) \quad = \quad (1 + x + x^2 + \cdots + x^{n-1}) I_{n-1}(x)$$

Preuve: La formule de récurrence indique que le coefficient de x^k dans le polynôme I_n s'obtent en ajoutant les coefficients de $x^k, x^{k-1}, \ldots, x^{k-n+1}$ de I_{n-1} ; mais ceci revient à multiplier ce polynôme par $1 + x + x^2 + \ldots + x^{n-1}$.

□

Ceci nous donne les valeurs suivantes pour les petites valeurs de n.

$$I_1(x) = 1, I_2(x) = 1 + x, I_3(x) = (1+x)(1+x+x^2) = 1 + 2x + 2x^2 + x^3$$

plus généralement on peut écrire :

$$I_n(x) = \pi_{i=1}^{n} P_i(x), \qquad P_i(x) = 1 + x + x^2 + \ldots + x^{i-1}$$

On vérifie sur cette formule que l'on a bien $I_n(1) = n!$.

3.3. Valeur moyenne. Pour calculer la valeur moyenne du nombre d'inversions dans une permutation on fait la somme sur toutes les permutations α du nombre d'inversions de α et on divise par le nombre total de permutations, c'est à dire $n!$. Ainsi cette valeur M_n pour toutes les permutations sur n éléments est donnée par la formule suivante, où $nbInv(\alpha)$ dénote le nombre d'inversions de la permutation α :

$$M_n = \frac{\sum_{\alpha \in \mathbb{S}_n} nbInv(\alpha)}{n!}$$

soit en regroupant toutes les permutations présentant k inversions

$$M_n = \frac{\sum_{k=0}^{N} k I_{n,k}}{n!}$$

Ainsi

On remarque que le numérateur est la valeur au point 1 de la dérivée de $I_n(x)$ soit

$$M_n = \frac{I'_n(1)}{n!}$$

Remarque Le calcul effectué ici peut être généralisé à toute famille d'objets comptés par deux paramètre n et k. Si n est la taille et k une caractéristique pour les objets ; on note $A_{n,k}$ le nombre d'objets de talle n et de caractéristique k. Alors le polynôme $A_n(x) = \sum_{k=o}^{N} A_{n,k}x^k$, où N est la valeur maximale prise par la caractéristique parmi tous les objets de taille n, permet de déterminer la valeur moyenne M_n de la caractéristique pour les objets de taille n par la formule :

$$M_n = \frac{I'_n(1)}{I_n(1)}$$

Dans le cas des inversions d'une permutation on calcule la dérivée du produit des polynômes

$$\pi_{i=1}^{n} P_i(x), \qquad P_i(x) = 1 + x + x^2 + \ldots + x^{i-1}$$

par :

$$I'_n(x) \quad = \quad \sum_{i=1}^{n} I_n \frac{P'_i(x)}{P_i(x)}$$

Or

$$P'_i(1) = 1 + 2 + \ldots + i = \frac{i(i-1)}{2}$$

et $P_i(1) = i$ ce qui donne

$$M_n \; = \; \sum_{i=1}^{n} \frac{(i-1)}{2} \quad = \quad \frac{n(n-1)}{4}$$

Remarque : Dans le cas particulier des inversions de permutations, on pouvait obtenir ce résultat plus rapidement en remarquant que la la distribution des valeurs des $I_{n,k}$ est symétrique. En effet, on note que $I_{n,k} = I_{n,N-k}$ car l'opération qui associe à la permutation $a_1, a_2, \ldots, a_n$ la permutation $a_n, a_{n-1}, \ldots, a_1$ est une bijection entre les permutations qui présentent k inverssions et celles qui en présentent $N - k$. Ainsi la valeur moyenne des $I_{n,k}$ est la valeur médiane :

$$\frac{N}{2} \quad = \quad \frac{n(n-1)}{4}$$

4. Analyse du tri par sélection et du tri par insertion

4.1. Lien entre permutation et tris. On considère ici des algorithmes de tri qui consistent à remettre dans l'ordre croissant les éléments d'un tableau en effectuant des comparaisons entre les éléments deux à deux et en échangeant les valeurs de deux variables.

Dans ce cadre analyser un algorithme consiste à déterminer le nombre moyen de comparaisons effectuées et le nombre moyen d'échanges, ceci pour toutes les suites possibles d'objets à trier. On se contente de ne considérer ici que les tableaux correspondant à des permutations. Bien entendu, trier un tableau de taille n dont on sait à priori qu'il contient

tous les nombres entre 1 et n est une opération qui peut paraître absurde puisque le résultat sera $1, 2, \ldots, n$. Toutefois l'algorithme se comporte dans ce cas comme il se comporterait pour une suite de n objets différents rangés dans l'ordre donné par la permutation.

Note méthode d'analyse, consitant à se reduire aux permutations est donc valable si l'ordre des éléments des tableaux à trier se reparti uniformément parmi toutes les permutations de $\mathbb{S}_n$. C'est notre hypothèse ici, notre analyse n'est donc plus valide si les algorithmes s'appliquent à des tableaux qui sont presque triés où pour lesquels l'ordre des élements varie peu.

4.2. Tri par sélection. Le tri par selection du tableau `x[1], x[2], ... x[n]` consiste à chercher le plus grand élément, le placer en position `x[n]` puis recommencer récursivement l'opération sur le tableau `x[1], x[2], ... x[n-1]`.

Ceci s'arrête lorsque tous les éléments sont à leur place, soit lorsque `n` vaut 1.

ceci s'écrit :

```
triSelection(int[] x, int i){
     if (i == 1)  return;
     else {
          for (j = 0; j < i; ++j)
               if (x[j] > x[i]) {
                   // echange de x[j] et x[i]
                   t = x[j]; x[j] = x[i]; x[i] = t;
                }
          triSelection(x, i-1);
          }
 }
```

On constate que l'on effectue systématiquement $n-1$ comparaisons avant l'appel récursif ainsi le nombre de comparaisons cs_n pour trier n éléments par l'algorithme du tri par sélection vérifie la relation de récurrence :

$$cs_n = n - 1 + cs_{n-1}$$

ce qui donne

$$cs_n = \frac{n(n-1)}{2}$$

Pour le nombre d'échanges on ne peut avoir de résultat précis simplement. On peut tout juste constater que cahque échange diminue le nombre d'inversions de la permutation triée donc le nombre moyen est majoré par

$$\frac{n(n-1)}{4}$$

4.3. Tri par insertion. Le tri par insertion consiste à insérer successivement les éléments dans un tableau que l'on suppose trié. Ainsi on commence par insérer `x[2]` dans le tableau réduit à l´élément `x[1]`, puis on insère `x[3]` dans un tableau contenant `x[1]` et `x[2]` et ainsi de suite. L'algorithme d'insertion de `x[i]` dans `x[1], x[2], ..., x[i-1]` consiste à décaler les éléments plus grands que `x[i]` de cet ensemble jusqu'à trouver la place où il faut insérer `x[i]`, soit plus précisément :

```
triInsertion (int[] x, int i, int n){
     //n est la taille du tableau
     if (i > n) return ;
     else {
         j = i;
         t = x[i];
         while (j > 0 && x[j-1] > t){
              x[j] = x[j-1];  j--;
         }
         x[j] = t;
         triInsertion(x,i+1,n);
    }
}
```

Pour l'analyse de cet algorithme, on remarque que chaque comparaison entre deux éléments du tableau est suivie d'un décalage qui diminue d'une unité le nombre d'inversions (sauf pour la dernière comparaison de la boucle `while` qui dans le cas où `j> 0` n'est pas suivie d'un tel décalage). On peut donc majorer le nombre de comparaisons par le nombre d'inversions de la permutation associée au tableau à trier augmenté de n. Le nombre moyen de comparaisons est alors de :

$$\frac{n(n-1)}{4} \quad + \quad n \quad = \frac{n^2}{4} + \frac{3n}{4}$$

Ce qui, pour n assez grand, est deux fois inférieur au nombre de comparaisons du tri par sélection.

CHAPITRE 4

Les cycles d'une permutation

Dans la suite il est commode de considérer une permutation $\alpha = a_1, a_2, \ldots, a_n$ comme une application bijective de $\{1, 2, \ldots, n\}$ dans lui même. Ainsi on peut écrire : $a_i = \alpha(i)$.

1. Préliminaires

1.1. Définitions. Les cycles d'une permutation se lisent simplement sur le graphe orienté dont les sommets sont $\{1, 2, \ldots, n\}$ et qu, pour chaque $1 \leq i \leq n$, a un arc d'origine i et d'extrémité a_i. Ce graphe peut avoir des boucles dans le cas où il existe i tel que $i = a_i$, i. e. des points fixes pour α.

Un cycle de la permutation est constitué d'un circuit de ce graphe. Plus formellement on a :

DÉFINITION 2. *Un cycle d'une permutation α est une suite d'éléments $b_1, b_2, \ldots, b_k$ tous distincts telle que $b_{i+1} = \alpha(b_i)$ pour $i < k$ et $b_1 = \alpha(b_k)$.*

Par exemple, la permutation $\beta = 3, 5, 1, 2, 8, 6, 9, 4, 7$ possède 4 cycles qui sont respectivement $1, 3$ puis $2, 5, 8, 4$ puis 6 et enfin $7, 9$.

On représente souvent une permutation comme un produit de cycles en écrivant les cycles les uns à la suite et en mettant chaque cycle entre parenthèses ; par exemple la permutation β s'écrit :

$$\beta \quad = \quad (1, 3)(2, 5, 8, 4)(6)(7, 9)$$

Il faut noter que cette écriture n'est pas unique, l'ordre des cycles n'étant pas précisé, de même que le premier élément de chaque cycle.

On aussi par exemple :

$$\beta = (4, 2, 5, 8)(7, 9)(6)(3, 1)$$

1.2. Algorithmes. Le calcul du cycle contenant un élement `i` se fait simplement par l'algorithme suivant :

```
cycle(int i, int[] alpha) {
   b = i; afficher(b);
   while (alpha[b] != i) {
       b = tab[b];
       afficher(b);
       }
}
```

Ainsi la détermination du nombre de cycles d'une permutation consiste à considerer chaque élement et parcourir son cycle, cela nécessite un tableau de booléens indiquant si on a déjà calculé le cycle d'un élément afin de ne pas le recalculer, ceci donne :

```
int nbCycles(int[] alpha, int n) {
     for (int i = 1; i <= n ; ++i) trouve[i] = false;
     for (int i = 1; i < n ; ++i)
          if (!trouve[i]) {
               k++;  trouve[i] = true;
               for (int j = i; alpha[j] != i; j = alpha[j])
                    trouve[alpha[j]] = true;}
     return k;
    }
```

2. Structure cyclique

La structure cyclique d'une permutation α est la suite de nombres $c_1, c_2, \ldots, c_n$ où c_i est le nombre de cycles de longueur i on a ainsi :

$$\sum_{i=1}^{n} ic_i \quad = \quad n$$

Une involution est une permutation dont les cycles sont de longueur 1 ou 2, on a donc $c_i = 0$ pour $i > 2$. Une permutation circulaire n'a qu'un cycle de longueur 1, sa structure cyclique est donc $0, 0, \ldots, 0, 1$; enfin l'identité a pour structure cyclique $n, 0, 0 \ldots, 0$ et celle d'une transposition est $n-2, 1, 0, 0, \ldots 0$.

PROPOSITION 9. *Le nombre de permutations de structure cyclique $c_1, c_2, \ldots, c_n$ est donné par*

$$\frac{n!}{\Pi_{i=1}^{n}(c_i!)i^{c_i}}$$

Preuve: Pour construire une permutation de structure cyclique $c_1, c_2, \ldots, c_n$ on part d'une permutation quelconque $a_1, a_2, \ldots, a_n$ et on met des parenthèses autour des éléments de façon à avoir c_1 cycles de longueur 1, puis c_2 cycles de longueur 2 et ainsi de suite. Il y a $n!$ permutations, chacune donnant une permutation de structure cyclique $c_1, c_2, \ldots, c_n$, or chacune de ces permutations est obtenue $\Pi_{i=1}^{n}(c_i!)i^{c_i}$ fois ; en effet on peut choisir le premier élément de chaque cycle de $\Pi_{i=1}^{n} i^{c_i}$ façons et permuter entre eux les c_i cycles de longueur i de $\Pi_{i=1}^{n}(c_i!)$ façons. □

COROLLAIRE 3. – *Le nombre de permutations circulaires est* $(n-1)!$
– *Le nombre de transpositions est* $\frac{n(n-1)}{2}$
– *Le nombre d'involutions sans point fixe sur* $\{1, 2, \ldots, 2m\}$ *est*

$$\frac{(2m)!}{m!2^m} = (2m)!!$$

où $(2m)!!$ *est la factorielle impaire, c'est à dire le produit de tous les nombres impairs compris entre 1 et* $2m$.

Preuve:
– Pour une permutation circulaire on a $c_n = 1$, ainsi la propostion donne le nombre $\frac{n!}{1!n^1}$
– Pour le nombre de transpositions la formule de la proposition donne $\frac{n!}{(n-2)!2}$

– Une involution sans point fixe sur $\{1, 2, \ldots, 2m\}$ a m cycles de longueur 2, la formule indique que leur nombre est $\frac{(2m)!}{m!2^m}$ dans cette formule le dénominateur est le produit de tous les nombres pairs compris entre 1 et $2m$, ainsi il reste après simplification du numérateur $(2m!)$ tous les nombres impairs.

□

Un algorithme de tirage aléatoire uniforme parmi toutes les permutations de structure cyclique donnée $c_1, c_2, \ldots, c_n$ consiste à tirer une permutation au hasard et à regrouper les c_1 premiers éléments en c_1 cycles de longueur 1, puis les $2c_2$ suivant en c_2 cycles de longueur 2 et ainsi de suite.

3. Enumération

On se propose d'appliquer ici la technique développée au chapitre précédent afin de déterminer le nombre moyen de cycles dans une permutation sur n éléments. Pour cela on note $C_{n,k}$ le nombre de permutations sur $\{1, 2, \ldots, n\}$ ayant k cycles et on commence par obtenir une formule de récurrence sur ces nombres.

3.1. Formule de récurrence.

PROPOSITION 10. *Les nombres $C_{n,k}$ staisfont la relation :*

$$C_{n,k} \quad = \quad C_{n-1,k-1} \; + \; (n-1)C_{n-1,k}$$

Preuve: Cette formule résulte de la construction suivante pour la construction d'une permutation sur $\{1, 2, \ldots, n\}$ ayant k cycles, à partir d'une permutation sur $\{1, 2, \ldots, n-1\}$.

– On peut chosir une permutations ayant $k-1$ cycles et ajouter n pour former un nouveau cycle ; ceci peut se faire de $C_{n-1,k-1}$ façons différentes.
– On peut aussi choisir une permutation ayant k cycles et insérer n dans l'un des cycles. Cette insertion se fait dans l'une des permutations de $n-1$ façons différentes ; le fait qu'il y ait $C_{n-1,k}$ permutations où l'on peut insérer donne le résultat.

□

Ceci nous permet de donner un tableau des premières valeurs de $C_{n,k}$:

k	1	2	3	4	5	6	7
n = 2	1	1					
n= 3	2	3	1				
n = 4	6	11	6	1			
n = 5	24	50	35	10	1		
n = 6	120	274	225	85	15	1	
n = 7	720	1764	1624	735	172	21	1

3.2. Valeur moyenne. L'étape suivante conssiste à introduire les polynômes :

$$C_n(x) = \sum_{k=0}^{n} C_{n,k} x^k$$

La propostion se traduit pour les polynômes par

COROLLAIRE 4. *Les polynômes $C_n(x)$ satisfont la relation de récurrence :*

$$C_n(x) \quad = \quad (x \ + \ n-1)C_{n-1}(x)$$

Preuve: Il suffit de constater que le coefficent de x^n dans le produit du polynôme C_{n-1} par $(x+n-1)$ est égal à la somme du coefficient de x^{n-1} et de $(n-1)$ fois le coefficient de x^n. □

Ceci permet alors de donner la formule

$$C_n(x) \quad = \quad \Pi_{k=o}^{n-1}(x+k)$$

On applique la formule donnant la moyenne du nombre de la caratéristique dans les permutations de taille n par

$$M_n \quad = \quad \frac{C_n'(1)}{C_n(1)}$$

THEORÈME 2. *Le nombre moyen de cycles dans une permutation de $\mathbb{S}_n$ est le nombre harmonique :*

$$H_n \quad = \quad 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

Preuve: Il suffit de calculer la dérivée de $C_n(x)$. On note $\pi_k(x)$ le polynôme $x+k$, ceci donne la formule suivante pour C_n' :

$$C_n'(x) \quad = \quad \Pi_{k=0}^{n-1}\frac{C_n(x)}{\pi_k(x)}$$

Or $C_n(1) = N!$ et $\pi_k(1) = k+1$
Ainsi :

$$C_n'(1) \quad = \quad \Pi_{k=0}^{n-1}\frac{n!}{k+1} \quad = n!H_n$$

Ce qui nous donne le résultat en divisant par $C_n(1) = n!$.

□

4. Cycles et minima partiels

On appelle *minimum partiel* d'une permutation $a_1, a_2, \ldots, a_n$ un a_i tel que $a_i > a_j$ pour tout $i < j$. Ainsi a_1 est toujours un minimum partiel, ainsi que 1. Une permutation telle que $a_1 = 1$ n'a qu'un seul minimum partiel.

On remarque que les minima partiels sont les valeurs que prend la variable `min` dans le calcul du plus petit élément d'un tableau par l'algorithme suivant :

```
int min = x[1];
for (int i = 2; i < n; ++i)
     if (x[i] < min)
          min = x[i];
```

4.1. Valeur moyenne. Pour calculer le nombre moyen de valeurs que prend la variable `min` dans la recherche du plus petit élément d'un tableau on peut se ramener au cas où le tableau est une permutation tirée au hasard et rechercher le nombre moyen de minima partiels d'une permutation. Or ceci

PROPOSITION 11. *Le nombre moyen de minima partiels dans une permutation de* $\mathbb{S}_n$ *est le nombre harmonique*

$$H_n \quad = \quad 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

Preuve:

On peut remarquer que les nombres $M_{n,k}$ de permutations ayant k minima partiels satisfont la même récurrence que les nombres $C_{n,k}$, en effet pour construire une permutation sur $\{1, 2, , \ldots, n\}$ qui présente k minima partiels, on peut soit partir d'une permutation sur $\{1, 2, , \ldots, n-1\}$ présentant $k-1$ minima partiels et insérer n en première position, soit en prendre une sur le même ensemble qui présente k minima partiels et insérer n en une autre position que la première. Dans le premier cas on obtient une seule permutation, dans le second il y a $n-1$ façons d'insérer ce qui donne la formule :

$$M_n, k \quad = \quad M_{n-1,k-1} + (n-1)M_{n-1,k}$$

Les deux formules de récurrences étant les mêmes et, puisque l'on a aussi $M_{1,1} = C_{1,1} = 1$, on obtient $M_{n,k} = C_{n,k}$ et la veleur moyenne est la même.

□

COROLLAIRE 5. *Le nombre moyen d'échanges dans l'algorithme du tri par sélection est la somme des nombres harmoniques :*

$$H_1 + H_2 + \ldots + H_{n-1}$$

4.2. Transformation de Foata. En fait il existe une bijection entre l'ensemble des permutations ayant k cycles et celles qui ont k minima partiels. Cette bijection se trouve être utile pour plusieurs résultats sur la combinatoire des permutations et a pris le nom du premier à l'avoir considérée : D. Foata.

Cette transformation s'exprime de la manière suivante :

La donnée est une permutation α qui a k cycles, on écrit alors chaque cycle de façon à ce que son plus petit élément soit en tête, ensuite on écrit les cycles dans l'ordre des plus petits éléments décroissants, enfin on supprime les parenthèses pour obtenir β. Il est alors facile de voir que les plus petits éléments de chaque cycle de α sont devenus les minima partiels de β.

Par exemple pour $\alpha = 1, 9, 6, 3, 5, 4, 8, 7, 2$ les cycles sont

$$(9, 2) \quad (4, 3, 6) \quad (1) \quad (5) \quad (7, 8)$$

Ceux-ci écrits avec leur lus petits élements en tête et dans l'ordre décroissants :

$$(7, 8)(5)(3, 6, 4)(2, 9)(1)$$

Enfin la suppression des parenthèses donne :

$$\beta = 7, 8, 5, 3, 6, 4, 2, 9, 1$$

Réciproquement pour une permutation β qui possède k minima partiels on insère une parenthèse ouvrante au début et une parenthèse fermante suivie d'une ouvrante avant chaque

minimum partiel $a_i, i > 1$ enfin une parenthèse fermante à la fin. Ceci donne la permutation α sous forme d'un produit de cycles. Par exemple pour

$$\beta = 7, 8, 5, 3, 6, 4, 2, 9, 1$$

les minima partiels sont $7, 5, 3, 2, 1$ l'insertion de parenthèses donnent

$$(7, 8)(5)(3, 6, 4)(2, 9)(1)$$

qui est la représentation de α sous forme de produits de cycles.

5. Descentes et excédents

5.1. Définitions.

– Une *descente* dans une permutation $\alpha = a_1, a_2, \ldots a_n$ est un a_i tel que $i < n$ et $a_i > a_{i+1}$.

– *Un excédent* dans une permutation $\alpha = a_1, a_2, \ldots a_n$ est un a_i tel que $a_i \geq i$.

– *Un excédent fort* dans une permutation $\alpha = a_1, a_2, \ldots a_n$ est un a_i tel que $a_i > i$.

On remarque que la permutation identité a 0 descente n, 0 excédent forts et n excédents. 'par contre la permutation $n, n-1, \ldots, 1$ a $n-1$ descentes et lLa permutation $2, 3, 4, \ldots, n, 1$ a $n-1$ excédents qui sont tous forts.

5.2. Résultat principal. On note respectivement $D_{n,k}, E_{n,k}, F_{n,k}$, les nombres des permutations de $\mathbb{S}_n$ qui ont k descentes, k excédents, k excédents forts.

THEORÈME 3. *Les nombres $D_{n,k}, E_{n,k+1}$ et $F_{n,k}$ sont égaux.*

Pour démontrer ce théorème on passe par trois remarques :

Remarque 1. $D_{n,k} = D_{n,n-1-k}$

Preuve: Pour vérifier cette formule il suffit de noter que si la permutation $\alpha = a_1, a_2, \ldots a_n$ a k descentes alors la permutation $\tilde{\alpha} = a_n, a_{n-1}, \ldots a_2, a_1$ en a $n-1-k$. En effet tout a_i, a_{i+1} qui correspond à une descente de α ne correspond plus à une de plus à une descente de $\tilde{\alpha}$ et réciproquement. □

Remarque 2. $E_{n,k} = F_{n,n-k}$

Preuve: Il faut ici considérer la permutation inverse $\alpha^{-1} = b_1, b_2, \ldots b_n$, elle est telle que $a_i = j, b_j = i$ pour tout i. Ainsi si $a_i \geq i$ est un excédent de α alors $b_j = i \leq a_i = j$ et n'est pas un excédent fort de α^{-1}. □

Remarque 3. $D_{n,k} = F_{n,n-1-k}$

Preuve: On considère la transformation de Foata qui a une permutation α qui a k descentes fait correspondre une permutation β. Si a_i, a_{i+1} correpond à une montée de α (i. e. $a_{i+1} > a_i$) alors a_{i+1} va suivre a_i dans la représentation en produits de cycles de β et □

Preuve: *(du théorème)*
Par la Remarque 1, on a

$$D_{n,k} = D_{n,n-1-k}$$

en appliquant la Remarque 3, à $D_{n,k'}$ avec $k' = n-1-k$ on obtient

$$D_{n,k} = D_{n,k'} = F_{n,n-1-k'} = F_{n,k}$$

De même en appliquant la Remarque 2 à $k' = k+1$ on obtient :

$$E_{n,k+1} = F_{n,n-k-1} = D_{n,n-k-1} = D_{n,k}$$

□

5.3. Formule de récurrence.

PROPOSITION 12. *Les nombres $D_{n,k}$ satisfont la relation de récurrence*

$$D_{n,k} \quad = (k+1)D_{n-1,k} + (n-k)D_{n-1,k-1}$$

Preuve: Si on supprime n de l'écriture usuelle d'une permutation α on obtient une permutation α' qui a le même nombre de descentes que α ou une descente en moins. Réciproquement pour obtenir une permutation α de $\mathbb{S}_n$ ayant k descentes on peut

- soit insérer n à la fin ou au milieu d'une descente d'une permutation α' de $\mathbb{S}_{n-1}$ ayant k descentes. Ceci peut se faire de $k+1$ façons.
- soit insérer n au début ou au milieu d'une montée d'une permutation α' de $\mathbb{S}_{n-1}$ ayant $k-1$ descentes. Ceci peut se faire de $n-k$ façons.

□

CHAPITRE 5

Tableaux de Young

1. Sous-suite croissante maximale

Un bel algorithme de la combinatoire est celui de la recherche d'une sous suite maximale d'une suite d'entiers.

Le problème est le suivant : on se donne une suite d'entiers $x_0, x_1, \ldots x_{n-1}$ et on cherche une des plus longue sous-suite $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ telle que

$$i_1 < i_2 < \ldots < i_k \quad \text{et} \quad x_{i_1} < x_{i_2} < \ldots < x_{i_k}$$

On constate sur l'exemple suivant que le problème n'est pas simple :

$$6, 13, 4, 8, 5, 14, 7, 10, 23, 55, 76, 81, 2, 3, 1, 9, 11, 12, 15$$

en effet, si on examine les cinq premiers termes on obteint plusieurs sous-suites croissantes maximales l'une est $6, 13$ toutefois il vaut mieux retenir la sous-suite $4, 5$ qui est plus facilement extensible, en effet on peut lui ajouter par la suite $7, 10$. Il faut donc lorsqu'on lit 4, le retenir comme premier élément possible. Plus généralement on détermine dans un tableau `p` et en `p[k]` le plus petit dernier élément possible pour une sous-suite croissante de longueur `k`. Pour chaque `x[i]` on cherche alors dans `p` le plus grand élément `p[k]` plus petit que lui, `x[i]` doit alors être rangé dans `p[k+1]` ceci donne l'algorithme suivant :

```
traiter (int[] x, int n){
   //n est la longueur de x
        p = new int[n];
        for (int i = 0; i < x.length; ++i) {
            k = maxp;
            while (k > 0 && p[k]  > x[i]) k--;
            p[k+1]= x[i];
            if (k == maxp) maxp++;
}
}
```

Cet algorithme permet alors de déterminer `maxp` la taille maximale d'une sous-suite croissante formée d'éléments de `x`, ainsi que le dernier élément de cette suite (`p[maxp]` toutefois on ne connait pas la suite elle même ; pour l'obtenir il faudrait que chaque élément connaisse le nombre qui se trouve juste avant lui dans la suite croissante la plus longue dont il est le dernier élément. Ceci se fait plus facilement si dans le tableau `t` on range non pas `x[i]` mais `i`, un tableau `pred` sera alors tel que `pred[i]` est l'indice de l'élément qui précède `x[i]` dans la plus longue sous suite croissante qui se termine par `x[i]`. Ceci donne :

```
  static void traiter (int[] x){
        int n = x.length;
        p = new int[n+1];
```

```
        p[0] = -1;
        pred = new int[n];
        for (int i = 0; i < x.length; ++i) {
            int k = maxp;
            while (k > 0 && x[p[k]] > x[i]) k--;
             p[k+1]= i;
            if (k == maxp) maxp++;
             pred[i] = p[k];
        }
    }
```

et pour l'affichage de la suite maximale :

```
int i = p[maxp];
        while (i != -1) {
           afficher(x[i]);
           i = pred[i];
           }
```

2. Diagramme de Ferrers et tableaux

.

Un tableau de Young est déterminé par un diagramme de Ferrers et par des entiers que l'on inscrit dans ce diagramme. Donnons d'abord la définition d´un diagramme de Ferrers :

DÉFINITION 3. *Un diagramme de Ferrers F est un ensemble de couples d'entiers strictement positifs tels que $(i,j) \in F$, $0 < i' \leq i$, $0 < j' \leq j$ implique $(i',j') \in F$, la taille du diagramme est égale au nombre d'éléments de F notée $|F|$. Les éléments de F sont les* cases *du diagramme.*

Un tel diagramme se dessine de la façon suivante :

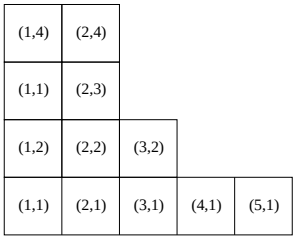

FIG. 1. Un diagramme de Ferrers de taille 12

Un diagramme est déterminé par les longueurs $\ell_1, \ell_2, \ldots, \ell_k$ de ses lignes. On a ainsi $\ell_1 < \ell_2 < \ldots < \ell_k$ Dans la suite on utilisera le symbole $\preceq$ pour représenter la relation d'ordre sur $\mathcal{N} \times \mathcal{N}$ donnée par

$$(a,b) \preceq (c,d) \Leftrightarrow a \leq c, \quad b \leq d$$

On notera $(a,b) \prec (c,d)$ si $(a,b) \preceq (c,d)$ et $(a,b) \neq (c,d)$.

Soit F un diagramme de Ferrers et $(a,b) \in F$ *l'équerre* $Eq_F(a,b)$ de (a,b) dans F est l'ensemble des éléments (c,d) de F tels que $a = c$ et $b \leq d$ ou $a \leq c$ et $b = d$. L'équerre de $(2,2)$ dans le digramme de la Figure 1 est représentée par la Figure 2 ci-dessous.

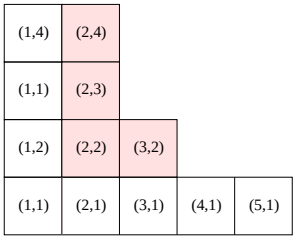

FIG. 2. L'equerre de (2,2)

DÉFINITION 4. *Soit F un diagramme de Ferrers de taille n, un tableau de Young T de forme F est une application ϕ de F dans $\{1,2,\ldots,n\}$ telle que :*

$$(i,j) \in F, \quad (i',j') \prec (i,j) \quad \Rightarrow \quad \phi(i',j') < \phi(i,j)$$

Un exemple de tableau de Young de forme F est donné par la figure suivante :

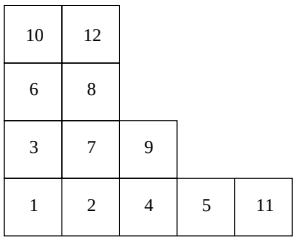

FIG. 3. Un tableau de Young de forme F

3. Algorithme de Robinson Schenstedt

Cet algorithme consiste à associer deux tableaux à ume permutation d'une manière qui s'inspire de l'algorithme donné plus haut pour la détermination de la plus longue sous suite croissante.

Une description suit

```
void ajoutSimple(int a, int b, int x) {
if (a +1 > nblignes) nblignes = a+1;
        if( b + 1 > taille[a]) taille[a]= b+1;
        tab[a][b] = x;
    }

    /* Algorithme d ajout dans une ligne
       donne comme resultat le nombre expulse
       si pas d expulse donne comme resultat -1 */
```

```
    static int ajoutLigne(YTableau p, YTableau q,  int a, int x, int m){
/* a numero ligne, x nombre a ajouter, m numero de l insertion */
        int res = -1;
          if (a == p.nblignes ) {
    p.ajoutSimple(a, 0, x);
    q.ajoutSimple(a, 0, m);
        }
        else {
    int j = p.taille[a] -1;
      if (x > p.tab[a][j]){
p.ajoutSimple(a,j+1,x);
                q.ajoutSimple(a,j+1,m);
    }
            else {
                while (j >= 0 && p.tab[a][j] > x) j--;
                res = p.tab[a][j+1];
                p.ajoutSimple(a,j+1,x);
         }
    }
return res;
    }

    /* Ajout du mieme  element x dans p q */
    static void ajout(YTableau p, YTableau q, int x, int m){

int y = x; int a = 0;
while (y != -1){
    y = ajoutLigne(p, q, a, y, m);
            a++;
}
    }

    static void creer (int[] alpha){
/* Creation de deux tableaux et affichage */
YTableau p = new YTableau(alpha.length);
YTableau q = new YTableau(alpha.length);
        for (int m = 1; m <= alpha.length; ++m){
    ajout(p,q,alpha[m-1],m);
    p.affiche(); System.out.println();
            q.affiche();System.out.println();
        }
            p.n = alpha.length; q.n = alpha.length;
    }
```

4. Formule des équerres

La taille de l'équerre de (a, b) c'est à dire le nombre des éléments de $Eq_F(a, b)$ est notée $e_F(a, b)$.

Les tailles des équerres de tous les éléments du diagramme de Ferrers ci-dessus sont données sur la figure ci-dessous :

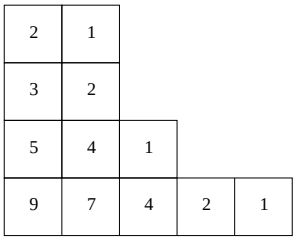

FIG. 4. Les tailles des équerres des cases du diagramme F

4.1. Enoncé de la formule. On donne d'abord une définition formelle des tableaux de Young qui aide dans la notation.

DÉFINITION 5. *Un diagramme de Ferrers F est un ensemble de couples d'entiers strictement positifs tels que $(i,j) \in F$, $0 < i' \leq i$, $0 < j' \leq j$ implique $(i',j') \in F$, la taille du diagramme est égale au nombre d'éléments de F notée $|F|$. Les éléments de F sont les* cases *du diagramme.*

Un tel diagramme se dessine de la façon suivante :

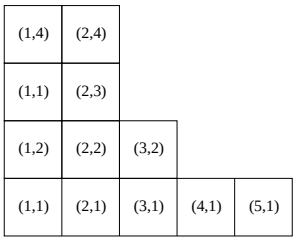

FIG. 5. Un diagramme de Ferrers de taille 12

Dans la suite on utilisera le symbole $\preceq$ pour représenter la relation d'ordre sur $\mathcal{N} \times \mathcal{N}$ donnée par

$$(a,b) \preceq (c,d) \Leftrightarrow a \leq c, \quad b \leq d$$

On notera $(a,b) \prec (c,d)$ si $(a,b) \preceq (c,d)$ et $(a,b) \neq (c,d)$.

Soit F un diagramme de Ferrers et $(a,b) \in F$ *l'équerre* $Eq_F(a,b)$ de (a,b) dans F est l'ensemble des éléments (c,d) de F tels que $a = c$ et $b \leq d$ ou $a \leq c$ et $b = d$. L'équerre de $(2,2)$ dans le digramme de la Figure 1 est représentée par la Figure 2 ci-dessous.

La taille de l'équerre de (a,b) c'est à dire le nombre des éléments de $Eq_F(a,b)$ est notée $e_F(a,b)$.

Les tailles des équerres de tous les éléments du diagramme de Ferrers ci-dessus sont données sur la figure ci-dessous :

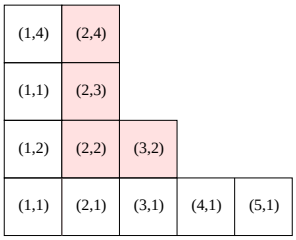

FIG. 6. L'equerre de (2,2)

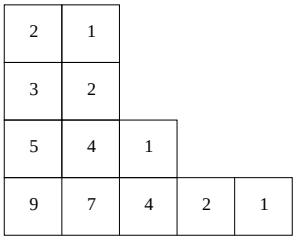

FIG. 7. Les tailles des équerres des cases du diagramme F

4.2. Chemins dans un diagramme de Ferrers. Un élément $u = (a, b)$ du diagramme de Ferrers F est dit *maximal* si $v \in F$, $u \preceq v$ implique $u = v$, ou de manière équivalente $Eq(u) = \{u\}$. Dans l'exemple ci-dessus les éléments maximaux sont $(5, 1), (3, 2)$ et $(2, 4)$.

DÉFINITION 6. *Un chemin dans un diagramme de Ferrers F est une suite $u_1, u_2, \ldots, u_m$ de cases telle que u_1 est quelconque, u_m est maximal et pour tout $1 \leq i < m$ on a :*

$$u_{i+1} \in Eq(u_i) - \{u_i\}$$

Le poids $w(f)$ d'un chemin $f = u_1, u_2, \ldots, u_p$ est défini par la formule

$$w(f) = \prod_{i=1}^{m-1} \frac{1}{e(u_i) - 1}$$

par convention le poids d'un chemin réduit à une seule case (maximale) est égal à 1 (c'est toujours le cas pour un produit vide).

Ainsi sur notre exemple le chemin $(2, 1), (2, 3), (2, 4)$ a pour poids

$$\frac{1}{6}\frac{1}{2} = \frac{1}{12}$$

L'origine d'un chemin $f = u_1, u_2, \ldots u_m$ (notée $or(f)$) est u_1 et son extremité (que l'on notera aussi $ext(f)$) est u_m.

Noter que le poids d'un chemin est en quelque sorte la probabilité de l'emprunter en partant d'une case donnée u_1 et en choisissant à chaque étape au hasard uniformément un élément de $Eq(u_i) - \{u_i\}$ pour terminer sur une case maximale.

Le but de cette première partie est de démontrer le résultat (difficile) suivant :

PROPOSITION 13. *Soit F un diagramme de Ferrers et soit u une case maximale de F la somme des poids des chemins d'extremité u est donnée par :*

$$\sum_{ext(f)=u} w(f) \quad = \quad \prod_{v\in F, u\in Eq(v)} (1+\frac{1}{e(v)-1})$$

Afin d'illustrer par un exemple ce résultat on considèrera le diagramme ci-dessus avec les noms suivants pour les cases :

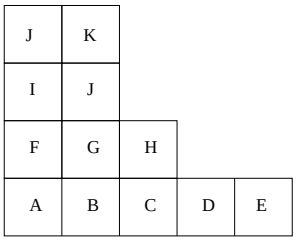

FIG. 8. Les noms des cases du diagramme F

Ainsi les chemins ayant pour extrémité H sont $H, CH, FH, GH, ACH, BCH, FGH, BGH, AFH, ABCH$ $AFGH, ABGH$ la somme de leurs poids vaut :

$$1+\frac{1}{3}+\frac{1}{4}+\frac{1}{3}+\frac{1}{7}\frac{1}{3}+\frac{1}{6}\frac{1}{3}+\frac{1}{4}\frac{1}{3}+\frac{1}{6}\frac{1}{3}+\frac{1}{7}\frac{1}{4}+\frac{1}{7}\frac{1}{6}\frac{1}{3}+\frac{1}{7}\frac{1}{4}\frac{1}{3}+\frac{1}{7}\frac{1}{6}\frac{1}{3}=\frac{20}{9}$$

On remarque que c'est aussi la valeur de

$$(1+\frac{1}{e(C)-1})(1+\frac{1}{e(F)-1})(1+\frac{1}{e(G)-1}) \quad = \quad (1+\frac{1}{3})(1+\frac{1}{4})(1+\frac{1}{3})$$

Pour démontrer ce résultat on considère les projections d'un chemin $f=(a_1,b_1),(a_2,b_2),\ldots,(a_m,b_m)$. La première projection, de f notée $I(f)$, est constituée de l'ensemble formé des a_i différents de a_m ; la deuxième notée $J(f)$ est l'ensemble des b_j différents de b_m. Ainsi pour le chemin $f=(2,2),(2,4),(3,4),(3,5)$, on a $I(f)=\{2\}, J(f)=\{2,4\}$.

Dans la suite, pour une forme F, un élément maximal $u=(a,b)$ de F et deux ensembles d'entiers $I\subset\{1,2,\ldots a\}$ et $J\subset\{1,2,\ldots b\}$, on note $\gamma_F(I,J,u)$ l'ensemble des chemins f dans F tels que $I(F)=I, J(F)=J$ et qui ont pour extrémité u. Le lemme suivant est une étape utile pour la preuve de la Proposition 1.

LEMME 1. *La somme des poids des chemins de $\gamma_F(I,J,u)$ pour $u=(a,b)$ est donnée par :*

$$\prod_{a_i\in I}(1+\frac{1}{e(a_i,b)-1})\prod_{b_j\in J}(1+\frac{1}{e(a,b_j)-1})$$

La preuve du Lemme s'effectue par récurrence sur $|I|+|J|$.

Dans le cas où J est vide il n'y a qu'un seul chemin dans $\gamma_F(I,J,u)$ qui est $(a_1,b),(a_2,b)\ldots(a_p,b),(a,b)$ (où $I=\{a_1,a_2,\ldots a_p\}$) son poids est bien

$$\prod_{a_i\in I}(1+\frac{1}{e(a_i,b)-1})$$

Le cas où $I = \emptyset$ se traite de manière similaire.

Si ni $I = \{a_1, a_2, \ldots a_p\}$ ni $J\{b_1, b_2, \ldots b_q\}$ ne sont vides tout chemin débute par (a_1, b_1) et se poursuit par un chemin g tel que $I(g) = I, J(g) = J - \{b_1\}$ ou tel que $J(g) = J, I(g) = I - \{a_1\}$. Notons $\overline{I} = I - \{a\}, \overline{J} = J - \{b\}$ on a alors :

$$\sum_{f \in \gamma_F(I,J,(a,b))} w(f) \quad = \quad \frac{1}{e(a_1,b_1)-1}[\sum_{g \in \gamma_F(\overline{I},J,(a,b))} w(g) + \sum_{g \in \gamma_F(I,\overline{J},(a,b))} w(g)]$$

Puisque $|\overline{I}| < |I|$ on a :

$$\sum_{g \in \gamma_F(\overline{I},J,(a,b))} w(g) \quad = \quad \prod_{a_i \in \overline{I}} (1 + \frac{1}{e(a_i,b)-1}) \prod_{b_j \in J} (1 + \frac{1}{e(a,b_j)-1})$$

Ainsi en utilisant une formule analogue pour $\gamma_F(I, \overline{J}, (a,b))$ on obtient :

$$\sum_{f \in \gamma_F(I,J,(a,b))} w(f) \quad = \quad \frac{1}{e(a_1,b_1)-1} \prod_{a_i \in \overline{I}} (1+\frac{1}{e(a_i,b)-1}) \prod_{b_j \in \overline{J}} (1+\frac{1}{e(a,b_j)-1})[\frac{1}{e(a_1,b)-1}+\frac{1}{e(a,b_1)-1}]$$

$$= \frac{e(a_1,b)-1+e(a,b_1)-1}{e(a_1,b_1)-1} \prod_{a_i \in I} (1 + \frac{1}{e(a_i,b)-1}) \prod_{b_j \in J} (1 + \frac{1}{e(a,b_j)-1})$$

La figure ci dessous montre que $e(a_1,b)-1+e(a,b_1)-1 = e(a_1,b_1)-1$ ce qui donne le résultat :

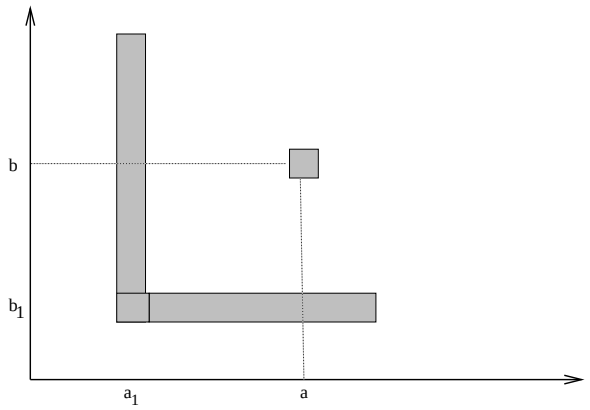

FIG. 9. L'équerre de (a_1, b_1)

Preuve de la Proposition 1 :

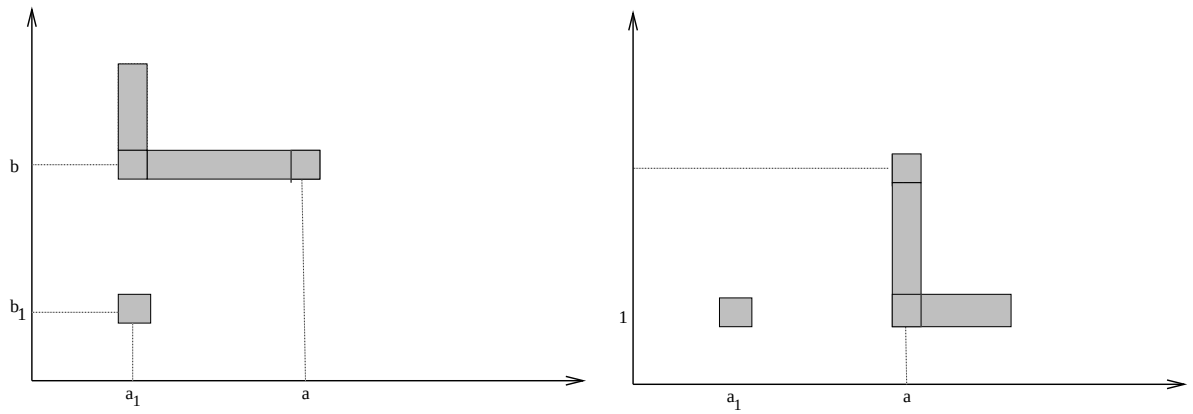

FIG. 10. L'équerre de (a_1, b)
et celle de (a, b_1)

En utilisant le Lemme on obtient que la somme des poids des chemins d'extremité (a, b) est donnée par :

$$\sum_{ext(f)=(a,b)} w(f) \quad = \quad \sum_{\substack{I \subset \{1,\ldots,a-1\} \\ J \subset \{1,\ldots,b-1\}}} \prod_{a_i \in I} \frac{1}{e(a_i,b)-1} \prod_{b_j \in J} \frac{1}{e(a,b_j)-1}$$

Or la formule à démontrer dans la Proposition fait intervenir :

$$\prod_{v \in F, u \in Eq(v)} (1 + \frac{1}{e(v)-1})$$

pour $u = (a, b)$; on a ainsi tous les v de la forme (a, b') avec $b' < b$ ou (a', b) avec $a' < a$. D'autre part, dans le produit

$$\prod_{a'=1}^{a-1} (1 + \frac{1}{e(a',b)-1})$$

on peut pour chaque terme choisir soit 1 soit $\frac{1}{e(a',b)-1}$ ce qui donne la somme obtenue plus haut et termine la preuve de la Proposition.

4.3. Poids d'un tableau de Young. On considère un tableau de Young (F, ϕ) tel que $|F| = n$ et soit u l'élément de F tel que $\phi(u) = n$; cet élément est maximal dans F. On définit alors le poids $W(F, \phi)$ du tableau de la manière récursive suivante :

DÉFINITION 7. *Le poids du tableau constitué d'un seul élément est 1, le poids de* (F, ϕ) *est égal à*

$$W(F, \phi) = P_F(u) W(F - \{u\}, \phi')$$

où $(F - \{u\}, \phi')$ *est le tableau obtenu à partir de* F, ϕ *en supprimant la case qui contient* n *et où* $P_F(u)$ *est la somme des poids des chemins d'extrémité* u.

On alors le résultat

THEORÈME 4. *Le poids d'un tableau de Young (F,ϕ) ne dépend que de sa forme F et est égal au produit des équerres de toutes les cases de F*

$$W(F,\phi) \quad = \quad \prod_{(a,b)\in F} e(a,b)$$

Preuve: On procède par récurrence sur le cardinal n de F. On a ainsi, puisque $|F'| < |F|$ (en notant $u = (a,b)$) :

$$W(F,\phi) = P_F(u)W(F-\{u\},\phi') = P_F((a,b)) \prod_{(p,q)\in F'} e'(p,q)$$

Il nous reste à calculer les valeurs des equerres $e'(p,q)$ dans $F' = F - \{(a,b)\}$ en fonction de celles dans F, on obtient :

$$e'(p,q) = e(p,q) \quad \text{si} \quad p \neq a, \quad q \neq b$$

et

$$e'(a,q) \quad = \quad e(a,q)-1 \quad e'(p,b) \quad = \quad e(p,b)-1$$

Ceci donne :

$$W(F,\phi) = P_F((a,b)) \prod_{(p,q)\in F} e(p,q) \prod_{p=1}^{a-1} \frac{e(p,b)-1}{e(p,b)} \prod_{q=1}^{b-1} \frac{e(a,q)-1}{e(a,q)}$$

Or d'après la Proposition 1 :

$$P_f((a,b)) \quad = \quad \prod_{p=1}^{a-1}(1+\frac{1}{e(p,b)-1}) \prod_{q=1}^{b-1}(1+\frac{1}{e(a,q)-1})$$

Ce qui s'écrit

$$P_f((a,b)) \quad = \quad \prod_{p=1}^{a-1} \frac{e(p,b)}{e(p,b)-1} \prod_{q=1}^{b-1} \frac{e(a,q)}{e(a,q)-1}$$

et en reportant dans la formule plus haut on obtient le résultat cherché. □

Le lemme suivant permet de calculer la somme des poids de tous les tableaux de forme F donnée :

LEMME 2. *Soit F un diagramme de Ferrers et soit $u \in F$, la somme des poids de tous les chemins de F d'origine u est égale à 1.*

Preuve: Ceci résulte de ce que le poids d'un chemin est la probabilité de l'emprunter lorsque l'on part d'une case quelconque en allant au hasard d'une case à une autre appartenant à son équerre. Plus formellement, on peut procéder par récurrence sur la distance de u à un élément maximal. Si u est lui même maximal il n'y a qu'un chemin réduit à u, ce chemin est de poids 1. Sinon

$$\sum_{or(f)=(i,j)} w(f) \quad = \quad \frac{1}{e(i,j)-1} \sum_{(p,q)\in eq(i,j)} \sum_{or(g)=(p,q)} w(g)$$

En appliquant l'hypothèse de récurrence à (p,q) qui est plus proche d'une case maximale, on obtient

$$\sum_{or(f)=(i,j)} w(f) \quad = \quad \frac{1}{e(i,j)-1} \sum_{(p,q)\in Eq_F(i,j)} 1 \quad = \quad 1$$

□

THEORÈME 5. *La somme des poids de tous les tableaux de Young de forme F est indépendante de F et vaut $n!$, où n est le nombre d'élémenst de F.*

Preuve: On remarque d'abord d'après le lemme précédent que la somme des poids de tous les chemins d'une forme F est égale au nombre n d'éléments de F. On procède encore par récurrence sur n.

On a alors par définition du poids d'un tableau

$$\sum_{\phi} w(F,\phi) \quad = \quad \sum_{u\in max(F)} \sum_{\phi} w(F',\phi) \quad P_F(u) = \quad \sum_{u\in max(F)} (n-1)!$$

où $max(F)$ désigne l'ensemble des éléments maximaux de F. Le lemme précédent donne alors le résultat. □

4.4. Fin de la preuve et remarques. Pour terminer la preuve de la formule des équerres on note que la somme de tous les poids des tableaux de forme F est $n!$ et que le poids de chacun d'entre eux est le produit des équerres des cases de F, ainsi le nombre de tableaux est le quotient de $n!$ par le produit des équerres.

On peut noter que cette preuve fournit un algorithme de tirage aléatoire d'un tableau de forme F, il procède en itérant la procèdure suivante de placement de n dans le diagramme :

Choisir une case (i,j) de F puis, cheminer au hasard d'une case à une autre appartenant à l´équerre de la précédente, jusqu'à arriver à une case maximale u, poser $\phi(u) = n$. If $\mathcal{R} = \cup_{n\geq 0}\mathcal{R}_{n+1}$ and $\mathcal{N} = \cup_{n\geq 0}\mathcal{N}_{n+1}$, then :

$$\sum_{r\in\mathcal{R}} \sum_{i} z^{|r|} u^{add_K(r,i)} = \sum_{r\in\mathcal{N}} \sum_{i} z^{|r|} u^{add_K(r,i)} |r|$$