Deux démonstrations de la formule des équerres

Exposé de maîtrise
Ecole Normale Supérieure
Philippe Marchal

Paul Gassiat
<paul.gassiat@ens.fr>

Maik Schwarz
<maik.schwarz@ens.fr>

Paris, le 23 juin 2005

1 Introduction

Soit $n \in \mathbb{N}$ et $\lambda = \{\lambda_1 \geq \cdots \geq \lambda_m\}$ une partition de n, i.e. une famille $\lambda_i \in \mathbb{N}$ tels que $n = \sum \lambda_i$. On appelle *diagramme de Ferrers de forme* λ un tableau de m lignes ayant λ_i cases dans la i-ième ligne. Par exemple, le diagramme de Ferrers de la forme $\lambda = \{6 \geq 4 \geq 4 \geq 1\}$ est

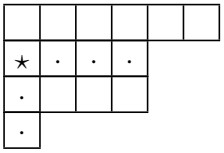


Fig. 1 – Un diagramme de Ferrers

On designe par (i, j) la case se trouvant dans la i-ième ligne et dans la j-ième colonne, par exemple l'étoile dans fig. 1 se trouve à la position $(2, 1)$. Un *tableau de Young de forme* λ est le diagramme de Ferrers de forme λ rempli de n entiers disctincts (normalement $1, \ldots, n$) de manière à ce que dans toutes les lignes et toutes les colonnes, on ait des suites croissantes, par exemple

1	3	7
2	4	
5	6	

Fig. 2 – Un tableau de Young

On appellera dans la suite un diagramme de Ferrers rempli de n entiers distincts qui n'est pas nécessairement un tableau de Young un *tableau*. Le nombre de tableaux de Young de forme λ est noté f_λ. *L'équerre* associée à une case (i, j) du diagramme est l'ensemble $H_{i,j}$ qui contient (i, j) ainsi que les cases du diagramme se trouvant en-dessous ou à droite de (i, j). On désigne par $h_{i,j}$ (h comme «hook length») la longueur de l'équerre partant de la case (i, j), c'est-à-dire $h_{i,j} := |H_{i,j}|$. Par exemple, dans fig. 1, l'équerre partant de l'étoile (marquée par des points) est de longueur 6, on a donc $h_{2,1} = 6$. Notons λ' la partition transposée de λ, i.e. $\lambda'_i := \max\{k \mid \lambda_k \geq i\}$. Le but de ce texte est de démontrer, de deux façons différentes, le théorème suivant.

Théorème 1 (Formule des équerres) *Soient $n \in \mathbb{N}$ et λ une partition de n. Alors on a la formule suivante pour le nombre de tableaux de Young de forme λ.*

$$f_\lambda = \frac{n!}{\prod h_{i,j}}$$

où le produit porte sur toutes les cases du diagramme de Ferrers de forme λ.

2 Les tableaux de Young et $\mathfrak{S}_n$

Afin de donner une idée de ce à quoi peuvent servir les tableaux de Young, on présente dans ce paragraphe quelques résultats faisant un lien entre les tableaux et les groupes symétriques. Par exemple, des nombreux résultats tournent autour des tableaux de Young dans la théorie des représentations. Mentionnons ici juste que les diagrammes de Ferrers de grandeur n sont en bijection avec les représentations irreductibles du groupe symétrique $\mathfrak{S}_n$ et que la dimension d'une représentation est égale à f_λ, où λ est la partition du diagramme de Ferrers qui lui correspond.
Continuons par deux algorithmes établissant une bijection entre $\mathfrak{S}_n$ et l'ensemble des couples (P, Q) de tableaux de Young de grandeur n, où P et Q sont de même forme (cf. [Knu98]). L'algorithme suivant insère un nouvel élément dans un tableau de Young P (qui n'est pas nécessairement composé des entiers $1, \ldots, n$).

Algorithme I

Entrée :	Un tableau de Young $P = (P_{ij})$ et un entier x ne se trouvant pas dans P.
Sortie :	Un tableau de Young contenant les éléments de P et x ; la coordonnée de la ligne dans laquelle se trouve x.
Etape 0	Soit $k := 1$.
Etape 1	Si x est plus grand que tous les éléments dans la k-ième ligne dans P (y compris le cas où cette ligne est vide), alors x est placé à la fin de cette ligne et l'algorithme s'arrête en sortant le nouveau diagramme ainsi que k. Sinon on choisit $j > 0$ tel que $y := P_{kj}$ soit l'élément le plus à gauche étant plus grand que x.
Etape 2	On place x dans la case (k, j). On retourne à l'étape 1 avec $x := y$ et en incrémentant k par un.

◇

Il est facile à voir que P reste un tableau de Young pendant tout l'algorithme. L'algorithme suivant élimine un élement d'un tableau.

Algorithme E

Entrée :	Un tableau de Young $P = (P_{ij})$ et la coordonnée s d'une ligne du diagramme dont on peut enlever la dernière case en obtenant un nouveau diagramme de Ferrers.
Sortie :	Un tableau de Young de même forme que P sauf que la dernière case de la ligne s en est enlevée.
Etape 0	On pose $x := \infty$ et $k := s$.
Etape 1	On choisit $j > 0$ tel que $y := P_{kj}$ soit l'élément le plus à droite étant plus petit que x.
Etape 2	Si $x = \infty$ on ôte la case (k, j) du diagramme, sinon on y place x.
Etape 3	Si $k > 1$ on retourne à l'étape 1 avec $x := y$ et en décrementant k par un. Si $k = 1$, l'algorithme s'arrête.

$\diamond$

En admettant que ces deux algorithmes sont inverses l'un à l'autre on prouve le théorème suivant.

Théorème 2 (L'algorithme RSK) *Il y a une bijection entre $\mathfrak{S}_n$ et l'ensemble des couples (P, Q) de tableaux de Young composés des entiers $1, \dots, n$ où P et Q sont de la même forme.*

L'algorithme qui sert à démontrer cet énoncé a été inventé par Robinson, Schensted et Knuth (d'où son nom).

Preuve : On se donne une permutation $\sigma \in \mathfrak{S}_n$. Soient d'abord P, Q deux tableaux vides. Alors, pour $i = 1, \dots, n$ (dans cet ordre), on itère l'opération suivante : on insère $\sigma(i)$ dans P en appliquant l'algorithme I qui sort la coordonnée k de la ligne où une nouvelle case a été ajoutée au diagramme ; puis on ajoute i à la fin de la k-ième ligne dans Q.

Il est alors évident que P et Q sont toujours de la même forme. De plus P et Q sont des tableaux de Young : P l'est grâce aux propriétés de l'algorithme I, et Q puisque on a ajouté les nouveaux éléments, en ordre croissant, au bord du diagramme.

Inversement, soient P et Q deux tableaux de Young. On en construit une permutation $\sigma \in \mathfrak{S}_n$ comme suit. On itère, pour $i = n, \dots, 2, 1$ (dans cet ordre !), l'opération suivante : soit k la ligne dans laquelle se trouve i dans Q ; alors on applique l'algorithme E au tableau P (resp. à ce qui en reste), et on note $\sigma(i)$ l'élément éliminé.

Comme les algorithmes I et E sont inverses l'un à l'autre et comme les deux constructions qu'on vient de décrire le sont aussi, on a bien établi bijection.

$\square$

A titre d'exemple, considérons la permutation $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 5 & 3 & 2 \end{pmatrix}$ en précisant les couples (P, Q) apparaissant au cours du calcul. Les numéros

au-dessus des flèches indiquent l'élément inséré dans P.

$$(\varnothing,\varnothing) \xrightarrow{4} \left(\begin{array}{|c|}\hline 4\\\hline\end{array}\,,\,\begin{array}{|c|}\hline 1\\\hline\end{array}\right) \xrightarrow{1} \left(\begin{array}{|c|}\hline 1\\\hline 4\\\hline\end{array}\,,\,\begin{array}{|c|}\hline 1\\\hline 2\\\hline\end{array}\right) \xrightarrow{5} \left(\begin{array}{|c|c|}\hline 1&5\\\hline 4\\\cline{1-1}\end{array}\,,\,\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\right)$$
$$\xrightarrow{3} \left(\begin{array}{|c|c|}\hline 1&3\\\hline 4&5\\\hline\end{array}\,,\,\begin{array}{|c|c|}\hline 1&3\\\hline 2&4\\\hline\end{array}\right) \xrightarrow{2} \left(\begin{array}{|c|c|}\hline 1&2\\\hline 3&5\\\hline 4\\\cline{1-1}\end{array}\,,\,\begin{array}{|c|c|}\hline 1&3\\\hline 2&4\\\hline 5\\\cline{1-1}\end{array}\right)$$

Fig. 3 – Exemple d'une application de l'algorithme RSK

Corollaire 3 *On a*

$$n! = \sum_{\lambda \vdash n} (f_\lambda)^2$$

Où la somme porte sur toutes les partitions de n. □

Citons encore un théorème simple mais remarquable.

Théorème 4 *Soit $\sigma \in \mathfrak{S}_n$ une permutation et (P,Q) son image sous l'algorithme RSK (cf. théorème 2). Alors l'image de σ^{-1} est (Q,P).*

On en déduit immédiatement le corollaire suivant.

Corollaire 5 *Le nombre de tableaux de Young contenant exactement les entiers $1,\dots,n$ est égale au nombre d'involutions dans $\mathfrak{S}_n$.* □

La forme de tableau de Young associée à une permutation est liée à certaines caractéristiques de la permutation (cf. [AD99]). On a par exemple le résultat suivant :

Proposition 6 *Soit σ une permutation, et $l(\sigma)$ la longueur de sa plus longue sous-suite croissante. Alors $l(\sigma)$ est la longueur de la première ligne des tableaux de Young correspondant à σ par l'algorithme RSK.*

Preuve : Soit P correspondant à σ par RSK, et λ_1 la longueur de sa première ligne. On a d'abord $\lambda_1 \geq l(\sigma)$. En effet, soit $a_1,\dots,a_k$ une sous-suite croissante de σ. On montre alors par récurrence sur i que lorsque a_i est introduit dans P, la première ligne de P contient aux moins $i-1$ éléments, et que si b_j est le j-ième, $b_j \leq a_j$. En effet, sous cette hypothèse, comme $a_{i+1} > a_i$, a_{i+1} est introduit au moins en (i+1)-ième position, et comme le contenu d'une case ne peut que diminuer au cours de la progression de l'algorithme, on conclut. Finalement, à la fin de l'algorithme, la première ligne de P a au mons k éléments. On a donc bien $\lambda_1 \geq l(\sigma)$.

Montrons maintenant $\lambda_1 \leq l(\sigma)$. Lorsque c est introduit dans P dans une colonne autre que la première, notons $\phi(c)$ l'élément à sa gauche sur sa ligne. On a alors $\phi(c) < c$ qui est une sous-suite croissante de σ. Soit a_l l'élément le plus à droite de la première ligne à la fin de l'algorithme. Alors

$$a_1 \overset{\phi}{\leftarrow} \ldots \overset{\phi}{\leftarrow} a_l$$

est une sous-suite croissante de σ de longueur λ_1. □

Ce résultat, combiné à la formule des équerres, permet d'obtenir le comportement asymptotique de $l(\sigma)$ pour les grandes valeurs de n :

Proposition 7 *Soit L_n la variable qui à un élément de $\mathfrak{S}_n$ associe la longueur de sa plus longue sous-suite croissante, $E[L_n]$ son espérance et $\sigma[L_n]$ son écart-type (pour la loi uniforme).*
Alors, quand $n \to \infty$, $E[L_n] \sim 2n^{1/2}$ et $\sigma[L_n] = o(n^{1/2})$.

3 Une preuve probabiliste

Cette preuve du théorème 1 (cf. [GNW79]) s'appuie sur un processus aléatoire qui consiste à remplir pas à pas un diagramme de Ferrers de façon à obtenir un tableau de Young à la fin. Dans la suite, tout tableau de Young contiendra les entiers $1, \dots, n$.
Définissons d'abord

$$F(\lambda_1, \dots, \lambda_m) = \begin{cases} \frac{n!}{\prod h_{i,j}} & \text{si } \lambda_1 \geq \cdots \geq \lambda_m \\ 0 & \text{sinon} \end{cases}$$

Soit λ une partition de n. Dans aucun tableau de Young de forme λ, il ne peut y avoir des cases ni en dessous ni à droite de la case contenant le nombre n. Ce dernier est par conséquent situé dans un «coin» du diagramme que l'on peut en ôter en obtenant un nouveau tableau de Young de grandeur $n-1$. Inversement, un tableau de Young de grandeur $n-1$ donné, on préserve les propriétés d'un tableau de Young en lui ajoutant un nouveau coin contenant n. On pourra donc conclure par induction si on arrive à montrer que

$$F(\lambda_1, \dots, \lambda_m) = \sum_\alpha F(\lambda_1, \dots, \lambda_{\alpha-1}, \lambda_\alpha - 1, \lambda_{\alpha+1}, \dots, \lambda_m)$$

ce que l'on notera en abrégé

$$F = \sum_\alpha F_\alpha \tag{1}$$

Remarquons que $F_\alpha = 0$ si (α, λ_α) n'est pas un coin dans le tableau en question, auquel cas on a $\lambda_\alpha - 1 > \lambda_{\alpha+1}$. Les termes non nuls de la somme correspondent ainsi aux coins du diagramme. Plutôt que (1) on démontrera

$$\sum_\alpha \frac{F_\alpha}{F} = 1 \tag{2}$$

en interprétant les F_α/F comme des probabilités qui apparaissent dans l'expérience suivante.
Dans le diagramme de Ferrers de forme λ, on choisit une case (i,j) avec probabilité uniforme $1/n$. Parmi les cases dans l'équerre $H_{i,j} \setminus (i,j)$ on choisit à nouveau une case, avec probabilité $1/(h_{i,j}-1)$ pour chaque case. Ainsi on continue jusqu'à ce qu'on atteigne un coin (α, β) (dit «terminus») où l'on arrête. L'ensemble des terminus possibles coïncide évidemment avec l'ensemble des coins du diagramme. Pour un coin quelconque (α, β) du diagramme, notons $p(\alpha, \beta)$ la probabilité que (α, β) soit le terminus d'un chemin aléatoire choisi selon l'algorithme ci-dessus.
La proposition suivante, dont la formule des équerres découle comme corollaire, nous fournit l'interprétation probabiliste des termes de la somme dans la formule (2).

Proposition 8 *Soit (α, β) un coin d'un diagramme de Ferrers. Alors*

$$p(\alpha, \beta) = \frac{F_\alpha}{F}$$

Corollaire 9 $\sum_\alpha \frac{F_\alpha}{F} = 1$

Peuve du corollaire : Tout chemin aléatoire se termine dans un coin. Or la somme porte sur tous les coins. □

Comme on vient de constater plus haut, ce corollaire donne une preuve du théorème 1.

Preuve de la proposition 8 : Comme (α, β) est un coin, F_α ne vaut pas 0. Par définition des fonctions F et F_α, on a

$$F = \frac{n!}{\prod h_{i,j}} \qquad F_\alpha = \frac{(n-1)!}{\prod h'_{i,j}}$$

où $h'_{i,j}$ désigne la longueur de l'équerre éventuellement raccourcie par rapport au diagramme original, plus précisément :

$$h'_{i,j} = \begin{cases} h_{i,j} & \text{si } i \neq \alpha \text{ et } j \neq \beta \\ h_{i,j} - 1 & \text{sinon} \end{cases}$$

Par exemple, dans la figure 4, les équerres correspondant aux cases marqués d'un point sont raccourcies par un si on enlève le coin noir.

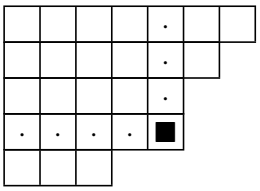


Fig. 4 – Equerres raccourcies

Le quotient vaut donc

$$\frac{F_\alpha}{F} = \frac{1}{n} \frac{\prod h_{i,j}}{\prod h'_{i,j}} = \frac{1}{n} \prod_{1 \leq j < \beta} \frac{h_{\alpha,j}}{h_{\alpha,j} - 1} \prod_{1 \leq i < \alpha} \frac{h_{i,\beta}}{h_{i,\beta} - 1}$$

ce qui est égal à

$$\frac{1}{n} \prod_{1 \leq j < \beta} \left(1 + \frac{1}{h_{\alpha,j} - 1}\right) \prod_{1 \leq i < \alpha} \left(1 + \frac{1}{h_{i,\beta} - 1}\right) \tag{3}$$

Pour un chemin aléatoire $(a, b) = (a_1, b_1) \to \cdots \to (a_\nu, b_\nu) = (\alpha, \beta)$ dans un diagramme de Ferrers, définissons ses projections $A := \{a_1, \ldots, a_\nu\}$ et $B := \{b_1, \ldots, b_\nu\}$. Appelons $p(A, B|a, b)$ la probabilité qu'un chemin ait les projections A et B sachant qu'il commence à (a, b).

Lemme 10 *Soient A et B comme ci-dessus. Alors*

$$p(A,B|a,b) = \prod_{\substack{i\in A\\ i\neq\alpha}} \frac{1}{h_{i,\beta}-1} \prod_{\substack{j\in B\\ j\neq\beta}} \frac{1}{h_{\alpha,j}-1}$$

Calculons $p(\alpha,\beta)$ en admettant ce résultat. On a certainement

$$p(\alpha,\beta) = \sum_{\substack{A,B\\ \max A=\alpha\\ \max B=\beta}} p(A,B) \tag{4}$$

où la somme porte sur toutes les projections possibles se terminant en (α,β). Trivialement, on a

$$p(A,B) = \sum_{(a,b)} p(A,B|a,b) \underbrace{p(a,b)}_{=\frac{1}{n}}$$

où (a,b) parcourt tout le diagramme de Ferrers. On utilise cette égalité dans (4) et on obtient

$$p(\alpha,\beta) = \frac{1}{n} \sum_{\substack{A,B\\ \max A=\alpha\\ \max B=\beta}} \sum_{(a,b)} p(A,B|a,b) = \frac{1}{n} \sum_{\substack{(a,b),A,B\\ \max A=\alpha\\ \max B=\beta}} p(A,B|a,b) \tag{5}$$

Considérons l'expression (3).

$$\begin{aligned}
&\frac{1}{n} \prod_{1\leq j<\beta} \left(1+\frac{1}{h_{\alpha,j}-1}\right) \prod_{1\leq i<\alpha} \left(1+\frac{1}{h_{i,\beta}-1}\right)\\
&= \frac{1}{n} \left(\sum_{\substack{B\subseteq\{1,\dots,\beta\}\\ \max B=\beta}} \prod_{\substack{k\in B\\ k\neq\beta}} \frac{1}{h_{\alpha,j}-1} \right) \left(\sum_{\substack{A\subseteq\{1,\dots,\alpha\}\\ \max A=\alpha}} \prod_{\substack{k\in A\\ k\neq\alpha}} \frac{1}{h_{i,\beta}-1} \right)\\
&= \frac{1}{n} \sum_{\substack{A,B\\ \max A=\alpha\\ \max B=\beta}} \left(\prod_{\substack{i\in A\\ i\neq\alpha}} \frac{1}{h_{i,\beta}-1} \prod_{\substack{j\in B\\ j\neq\beta}} \frac{1}{h_{\alpha,j}-1} \right)
\end{aligned}$$

D'après le lemme 10, c'est exactement le membre de droite de (5) ce qui montre la proposition 8. □

Preuve du lemme 10 : Notons $\mathcal{P}$ le membre de droite de l'énoncé.
Soient $A := \{a = a_1, \dots, a_\mu = \alpha\}$ et $B := \{b = b_1, \dots, b_\nu = \beta\}$ deux projections possibles. Vu le fonctionnement de l'algorithme choisissant le

chemin, le deuxième arrêt d'un chemin qui a pour projections A et B est soit (a, b_2), soit (a_2, b). On peut donc écrire

$$\begin{aligned} p(A,B|a,b) &= p\big((a_2,b) = 2^e\text{arrêt}\big)\; p(A\setminus\{a\}, B|a_2,b) \\ &\quad + p\big((a,b_2) = 2^e\text{arrêt}\big)\; p(A, B\setminus\{b\}|a,b_2) \\ &= \underbrace{\frac{1}{h_{a,b}-1}}_{\substack{\text{le } 2^e \text{ arrêt est choisi} \\ \text{à prob. uniforme} \\ \text{dans } H_{a,b}\setminus\{(a,b)\}}} \big(p(A\setminus\{a\}, B|a_2,b) + p(A, B\setminus\{b\}|a,b_2)\big) \end{aligned}$$

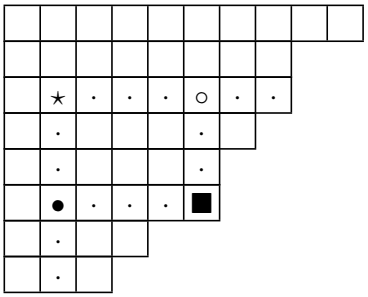
Par récurrence sur $\mu+\nu$, on suppose que

$$p(A\setminus\{a\}, B|a_2,b) = (h_{a,\beta}-1)\cdot\mathcal{P} \quad ; \quad p(A, B\setminus\{b\}|a,b_2) = (h_{\alpha,b}-1)\cdot\mathcal{P},$$

d'où

$$p(A,B|a,b) = \frac{1}{h_{a,b}-1}\big(\underbrace{(h_{a,\beta}-1)+(h_{\alpha,b}-1)}_{=:\eta}\big)\cdot\mathcal{P}$$

On constate que $\eta = h_{a,b}-1$. En effet, s'il y a r cases en dessous de (a,b), il y en a $\alpha-a$ en dessous de (a,β) et $r-(\alpha-a)$ en dessous de (α,b) ; au total on a donc r cases en dessous de (a,β) et (α,b). L'argument pour les parties horizontales des équerres est similaire. (Cf. aussi fig. 5)



$\star = (a,b)$, $\bullet = (\alpha,b)$, $\circ = (a,\beta)$, $\blacksquare = (\alpha,\beta)$

Fig. 5 – $(h_{a,b}-1) = \eta$

Ceci complète la démonstration du lemme. □

4 Une preuve algorithmique

Dans ce paragraphe, on redémontrera le théorème 1 en utilisant deux algorithmes qui transforment des tableaux de Young selon une règle appelée «jeu de taquin» (cf. [NPS97]). Commençons par définir un objet auxiliaire dont on aura besoin dans la suite.

Définition 11 (Fonction équerre) *Soit λ une partition de l'entier n. Une fonction équerre de forme λ est une application de l'ensemble des cases du diagramme de forme λ dans les entiers qui vérifie*

$$\forall\, (i,j) \in \lambda \ : -(\lambda'_j - i) \leq f(i,j) \leq (\lambda_i - j).$$

En d'autres mots, une fonction équerre associe à chaque case α un entier entre moins le nombre de cases strictement au-dessous de α et plus le nombre de cases strictement à droite de α.
On fixe une partition λ de n. Notre preuve consistera en construire une bijection entre les deux ensembles suivants :

$$\mathcal{Y} := \{(Y,f) \mid Y \text{ un tableau de Young, } f \text{ une fonction équerre}\}$$
$$\mathcal{T} := \{T \mid T \text{ un tableau quelconque}\}$$

Une fois que cette bijection sera établie, on pourra conclure puisque

$$|\mathcal{Y}| = f_\lambda \times \prod_{(i,j)\in\lambda} (\lambda_i + \lambda'_j - i - j + 1) = f_\lambda \times \prod_{(i,j)\in\lambda} h_{i,j}$$
$$|\mathcal{T}| = n!$$

4.1 Une esquisse de la preuve

Définissons d'abord un ordre sur les cases d'un tableau.

Définition 12 (Successeur) *Notons s l'application qui à une case d'un diagramme associe son successeur selon l'ordre antilexicographique sur les coordonnées. Le précédent sera noté s^{-1}.*

9	5	2
8	4	1
7	3	
6		

Fig. 6 – L'ordre des cases dans un diagramme

Dans la suite on constuira les applications entre $\mathcal{T}$ et $\mathcal{Y}$. Donnons déjà une idée générale du fonctionnement de la démonstration.

Pour transformer un tableau quelconque en un tableau de Young, on l'ordonne pas à pas en commençant par la case la plus petite dans l'ordre des cases : on considère son successeur et, si nécessaire, on échange les deux. Ayant un tableau qui est déjà ordonné jusqu'à une certaine position (i, j), on fait glisser (par l'algorithme P) l'élément de la case $s(i, j)$ dans la partie du diagramme déjà ordonnée de manière à obtenir un tableau qui est ordonné jusqu'à $s(i, j)$. En itérant cette opération, on tombe sur un tableau de Young à la fin.
La transformation qu'on vient d'esquisser n'est pas réversible de façon unique, aussi est-il nécessaire de tracer (par l'algorithme 1) les changements qu'on fait. Pour cela on utilisera les fonctions équerre définies plus haut. L'algorithme 2 qu'on va définir plus bas prendra donc un tableau quelconque comme argument, et il sortira un tableau de Young ainsi qu'une fonction équerre qui servira à retrouver le tableau original.
L'algorithme inverse consiste, cette fois en commençant par la case $(1, 1)$, en identifier l'un après l'autre les éléments qui ont été déplacés et en les remettre en place (par l'algorithme P', dit «jeu de taquin»). La fonction équerre permet d'identifier des «candidats» parmi lesquels se trouve l'élément cherché. En comparant les «chemins» des candidats dans le tableau on retrouve finalement l'élément qu'il faut. Dans un certain sens qui est encore à préciser il s'agit du candidat «maximal».

4.2 Algorithme $\mathcal{T} \to \mathcal{Y}$

On défine deux algorithmes simples qui seront appliqués plusieurs fois pour obtenir une application de $\mathcal{T}$ dans $\mathcal{Y}$. **Algorithme P**

Entrée : Un couple (A, a) d'un tableau A et d'un entier $a \leq n$
Sortie : Un tableau

Etape 0 Noter (i_0, j_0) les coordonnées de la case contenant a.
Etape 1 On définit

$$b = A(i_0, j_0 + 1) \text{ si } j_0 < \lambda_{i_0} \text{ et } b = n + 1 \text{ sinon,}$$
$$c = A(i_0 + 1, j_0) \text{ si } i_0 < \lambda'_{j_0} \text{ et } c = n + 1 \text{ sinon.}$$

a	b
c	

Etape 2 Si a est plus grand que b ou c, alors on définit un nouveau tableau B en échangeant a avec le plus petit des deux. On continue à l'étape 0 avec $A = B$. Si a est plus petit que b et c l'algorithme termine, la sortie étant A.

$\diamond$

L'exemple donné dans figure 7 montre le fonctionnement de l'algorithme P : le «5» est décalé vers la droite et puis vers le bas.

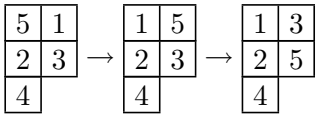


Fig. 7 – Exemple d'une application de l'algorithme P

Algorithme 1

Entrée :	Un triplet $(A, f, (i_0, j_0))$, où A est un tableau, f une fonction équerre, et $(i_0, j_0) \neq (1, 1)$ une case jusqu'à laquelle le tableau est ordonné
Sortie :	Un triplet $(B, g, s(i_0, j_0))$, où B est un tableau ordonné jusqu'à la position $s(i_0, j_0)$, et g une fonction équerre.
Etape 0	On pose $(i_1, j_1) := s(i_0, j_0)$ et $a = A(i_1, j_1)$.
Etape 1	Calculer B en appliquant l'algorithme P à (A, a).
Etape 2	Soit (i_2, j_2) la position de a dans B. – Pour tout $i_1 \leq i < i_2$ on pose $g(i, j_1) = f(i + 1, j_1) - 1$, – $g(i_2, j_1) = j_2 - j_1$, – $g(i, j) = f(i, j)$ sinon. L'algorithme sort $(B, g, (i_1, j_2))$.

◇

Il est évident que le tableau B sorti par l'algorithme 1 est ordonné jusqu'à la position (i_1, j_1), de plus il découle de la définition de la fonction équerre que g en est une. On voit ainsi que l'algorithme 1 et bien défini.
Définissons maintenant l'application de $\mathcal{T}$ dans $\mathcal{Y}$.

Algorithme 2

Entrée :	$A \in \mathcal{T}$
Sortie :	$(A, f) \in \mathcal{Y}$
Etape 0	On pose $f := 0$ et on désigne par (i_0, j_0) les coordonnées de la plus petite case de A selon l'ordre défini dans la figure 6.
Etape 1	On applique l'algorithme 1 au triplet $(A, f, (i_0, j_0))$ jusqu'à ce qu'on ait $(i_0, j_0) = (1, 1)$ auquel cas l'algorithme 2 se termine en sortant (A, f).

◇

La remarque que l'on vient de faire sur l'algorithme 1 assure que le tableau A ainsi obtenu est un tableau de Young.

4.3 Algorithme $\mathcal{Y} \to \mathcal{T}$

Comme dans le cas inverse, nous allons introduire deux algorithmes auxiliaires d'abord.

Algorithme P' (Jeu de taquin)

Entrée : Un triplet $(T, a', (i'_0, j'_0))$ où T est un tableau et $a' \leq n$ un entier.

Sortie : Un tableau

Etape 0 Soit (i'_1, j'_1) la position de a' dans T. Si $(i'_1, j'_1) \geq (i'_0, j'_0)$ l'algorithme termine en sortant T.

Etape 1 On pose

$$b' = T(i'_1, j'_1 - 1) \text{ si } j'_1 > j'_0 \text{ et } b' = 0 \text{ sinon,}$$
$$c' = T(i'_1 - 1, j'_1) \text{ si } i'_1 > 0 \text{ et } c' = 0 \text{ sinon.}$$

$*$	c'
b'	a'

Etape 2 On échange a' avec le sup de b' et c'. Cette opération fournit un nouveau tableau U. On continue à l'étape 0 avec $T = U$.

◇

Notons qu'à chaque fois que l'étape 2 est parcourue, l'entier a' est déplacé soit vers la gauche soit vers le haut, donc la position à laquelle il se trouve devient strictement plus petite par rapport à l'ordre selon définition 12. Vu les conditions de l'étape 0, on sait donc que l'algorithme P' se termine en temps fini indépendemment de l'entrée. Remarquons également que la case à laquelle s'arrête a' ne dépend que de sa position initiale dans le tableau et non pas de sa valeur.

Définition 13 (Code du chemin) *On considère le chemin d'un entier a' dans un tableau en lui appliquant l'algorithme P'. On code par N chaque déplacement de a' vers le haut (vers le «nord», i.e. tout échange de a' avec c') et par O chaque mouvement vers le gauche (vers le «ouest», i.e. tout échange avec b'). On appelera* code du chemin *de a' (par rapport à $(T, (i'_1, j'_1))$) la concaténation de ces codes, lue de droite à gauche. On met un ordre total sur l'alphabet $\{N, \varnothing, O\}$ en posant*

$$N < \varnothing < O$$

L'ensemble des codes est alors ordonné lexicographiquement.

Afin d'illustrer cette définition, regardons un exemple simple. Soit $a' = *$ un entier quelconque, $(i'_0, j'_0) = (1, 2)$ et T le tableau de gauche dans la figure 8. Appliquons alors l'algorithme 1.

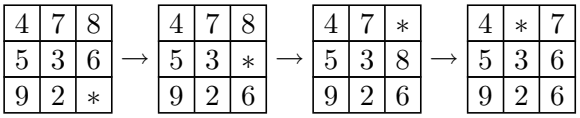


Fig. 8 – Le code du chemin est ONN

Définition 14 (Candidat) *Soit U un tableau, g' une fonction équerre, et (i'_1, j'_1) les coordonnées d'une case dans U. Une* case candidate *associée à $(U, g', (i'_1, j'_1))$ est une case (i, j) telle que $i \geq i'_1$, $g'(i, j'_1) \geq 0$, et $j = j'_1 + g'(i, j'_1)$. Un* élément candidat*, ou simplement un* candidat*, est un entier se trouvant dans une case candidate. Le* candidat maximal *est celui dont le code du chemin en appliquant l'algorithme P' est maximal pour l'ordre lexicographique.*

Il est clair que dans chaque ligne du tableau il ne peut y avoir qu'un seul candidat. On remarque qu'il existe toujours au moins un candidat puisque g' ne prend que de valeurs positives sur la dernière ligne du diagramme. L'exemple suivant illustre la définition des cases candidates. On marque la case (i'_1, j'_1) par une étoile et les cases candidates par des points. Le diagramme de droite représente la fonction équerre g'.

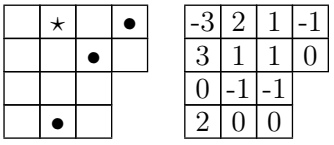


Fig. 9 – Cases candidates

Algorithme 1'

Entrée : Un triplet $(T, g', (i'_0, j'_0))$ où T est un tableau, g' une fonction équerre, et (i'_0, j'_0) une case (différente de la plus petite) jusqu'à laquelle T est ordonné.

Sortie : Un triplet $(U, f', s^{-1}(i'_0, j'_0))$ où U est un tableau et f' une fonction équerre.

Etape 0 Soient p le candidat maximal dans T et (i'_1, j'_1) ses coordonnées.

Etape 1 On applique l'algorithme P' à $(T, p, (i'_0, j'_0))$ et on note U le résultat.

Etape 2 On pose

$$f'(i,j) = \begin{cases} g'(i-1, j'_0) + 1 & \forall\, i'_0 < i \leq i'_1, j = j'_0 \\ 0 & \text{pour } (i,j) = (i'_0, j'_0) \\ g'(i,j) & \text{sinon} \end{cases}$$

L'algorithme termine et sort $(U, f', s^{-1}(i'_0, j'_0))$.

◇

On constate que U n'est pas forcément ordonné jusqu'à la position $s^{-1}(i'_0, j'_0)$. Ceci étant évidemment le cas si le candidat maximal s'arrête, à l'étape 1, à la position (i'_0, j'_0), ça peut bien ne pas être les cas dès qu'il s'arrête ailleurs. On prouvera, par la suite, qu'en effet le candidat maximal est toujours déplacé à la case (i'_0, j'_0). De plus, on verra que la fonction f' sortie par l'algorithme est effectivement une fonction équerre, c'est-à-dire que l'algorithme ci-dessus est bien défini.
Formulons maintenant l'algorithme associant à chaque élément de $\mathcal{Y}$ un élément de $\mathcal{T}$.

Algorithme 2'

Entrée : $(T, g') \in \mathcal{Y}$

Sortie : $T \in \mathcal{T}$

Etape 0 On pose $(i'_0, j'_0) = (1, 1)$.

Etape 1 On applique l'algorithme 1' autant de fois à $(T, g', (i'_0, j'_0))$ jusqu'à ce que (i'_0, j'_0) soit la plus petite case selon l'ordre de la définition 12. L'algorithme termine alors en sortant T.

◇

Pour prouver qu'on a bien une bijection, il faut montrer que les algorithmes 2 et 2' sont inverses l'un de l'autre.Comme ils consistent en des applications successives des algorithmes 1 et 1', il suffit de montrer que ces derniers sont inverses. Malheureusement, ce n'est pas vrai en général : étant donné un triplet qui satisfait les conditions de l'algorithme 1, si on lui applique successivement les algorithmes 1 et 1', on ne retombe pas nécessairement sur le même triplet. Mais on peut trouver un sous-ensemble de triplets pour

lesquels c'est le cas, et comme ce sous-ensemble contient tous les triplets intermédiaires obtenus lorsque l'on applique les algorithmes 2 et 2', on a bien le résultat voulu.

Définition 15 *Soit $\mathcal{C}$ l'ensemble des triplets $(A, f, (i_0, j_0))$ tels que :*

(i) (i_0, j_0) *sont les coordonnées d'une case de* A.

(ii) A *est ordonné jusqu'à* (i_0, j_0).

(iii) f *est une fonction équerre telle que* $f(i, j) = 0$ *pour* $(i, j) > (i_0, j_0)$.

(iv) Tous les éléments candidats p' *associés au triplet* $(A, f, (i_0, j_0))$ *vont en* (i_0, j_0) *lorsqu'on applique l'algorithme P' à* $(A, p', (i_0, j_0))$.

Pour prouver que $\mathcal{C}$ contient tous les triplets que l'on obtient en appliquant plusieurs fois l'algorithme 1 à un triplet consistant en un tableau, la fonction équerre nulle et la plus petite case du tableau, il faut établir deux choses : qu'un tel triplet appartient à $\mathcal{C}$, et que $\mathcal{C}$ est stable par l'algorithme 1. La première assertion est évidente, et pour la deuxième on voit déjà qu'en appliquant l'algorithme 1 à un élément de $\mathcal{C}$, on obtient un triplet vérifiant les 3 premières conditions de la définition. On verra plus tard qu'il vérifie également la quatrième.
De même, on veut prouver que $\mathcal{C}$ contient tous les triplets obtenus en appliquant l'algorithme 1' à un triplet constitué d'un tableau de Young, d'une fonction équerre et $(1, 1)$. Il est évident qu'un tel triplet appartient à $\mathcal{C}$, et il reste donc à montrer que $\mathcal{C}$ est stable par l'algorithme 1'. On voit déjà que le triplet obtenu en appliquant 1' à un élément de $\mathcal{C}$ vérifie les 2 premières conditions de la définition (la deuxième étant vraie grâce à la quatrième condition).
Nous allons maintenant définir le *chemin* et le *chemin-inverse* d'un élément, et établir une relation entre eux. Cette relation sera le principal résultat qui nous permettra de finir la preuve.

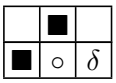
Définition 16 *Le* chemin *d'un élément a dans un tableau T est l'ensemble des cases par lesquelles a passe lorsqu'on applique l'algorithme P' à (T, a). Le* chemin-inverse *d'un élément a' par rapport à une case (i_0', j_0') dans un tableau T est l'ensemble des cases par lesquelles passe a' lorsqu'on applique l'algorithme P' à $(T', a', (i_0', j_0'))$.*

Définition 17 *Soit π le chemin d'un élément a dans la case (i_0, j_0). Soit a' un élément du tableau et π' son chemin-inverse par rapport à $s^{-1}(i_0, j_0)$. On dit que a' est* à gauche *de π si au moins une ligne du tableau contient à la fois des éléments de π et de π', et que sur la plus basse de ces lignes toutes les cases de π' sont (non strictement) à gauche de toutes celles de π. Dans le cas contraire, on dit que a' est à droite de π.*

On arrive maintenant au principal résultat sur les chemins et chemins-inverses.

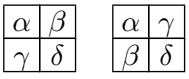
Théorème 18 *Soit A un tableau classé jusqu'en (i_0, j_0) et π le chemin de $A(s(i_0, j_0))$ dans A. Alors si $a' \leq n$ et π' est le chemin-inverse de $(a', (i_0, j_0))$ dans A, tous les éléments de π' , à part peut-être $s(i_0, j_0)$, sont du même côté de π.*

Preuve : D'abord, on remarque que si $i_0 = 1$, le théorème est évident puisque tous les éléments sont à droite de π, à part ceux se trouvant dans une case à gauche de $s(i_0, j_0)$. On suppose donc $i_0 \neq 1$.
On considère maintenant une case (i, j) contenant l'élément δ. Si c'est le seul élément de son chemin-inverse π', le théorème est évident. Dans le cas contraire, $(i-1, j)$ ou $(i, j-1)$ appartient à π'. Si on prouve que celle de ces deux cases qui contient le plus grand élément est du même côté de π que δ, alors le théorème est démontré par induction sur la longueur de π'.
Supposons d'abord $(i-1, j-1) \notin \pi$.
Si $(i, j-1) \in \pi'$: supposons qu'elle ne soit pas du même côté de π que (i, j). Alors nécessairement $(i, j-1)$ est à gauche de π et (i, j) est à droite de π, et donc $(i, j-1) \in \pi$ (voir la figure ci-dessous, où (i, j) est la case en bas à droite). Or $(i, j-2)$ ne peut pas appartenir à π car $(i, j-1)$ est à gauche de π. Comme $(i-1, j-1) \notin \pi$, $(i-1, j)$ est la première case de π et est $s(i_0, j_0)$, comme prévu dans l'énoncé du théorème.



$\circ \in \pi$, $\blacksquare \notin \pi$

Si $(i-1, j) \in \pi'$, les deux cases sont du même côté de π car comme $(i-1, j-1) \notin \pi$, l'existence d'une case de π strictement à gauche de l'une de ces deux cases implique la même chose pour l'autre case.
On suppose donc que $(i-1, j-1) \in \pi$. Si c'est la dernière case de π, le résultat est également vrai par définition. On suppose donc que $(i-1, j)$ ou $(i, j-1)$ appartient à π. On est alors dans un des deux cas suivants (où la case (i, j) est celle en bas à droite et $\alpha < \beta < \gamma < \delta$) :



En considérant les règles d'échange des algorithmes P et P', on voit que dans le premier cas, δ et γ sont à gauche de π. Dans le deuxième cas, δ et γ ont à droite de π. On a donc prouvé que dans tous les cas, δ et son plus grand voisin sont du même côté de π. □

Corollaire 19 *Avec les notations du théorème 18, on suppose que a' est dans une case différente de $s(i_0, j_0)$ et que π n'est pas inclus dans la ligne de $s(i_0, j_0)$ (en particulier $i_0 \neq 1$) . Alors a' est à gauche de π si et seulement si $(i_0, j_0) \in \pi'$.*

Preuve : Comme π contient des cases sur la ligne i_0, qui sont nécessairement à droite de (i_0, j_0), (i_0, j_0) est à gauche de π pour $i_0 \neq 1$, et il est clair que si $(i_0, j_0) \in \pi'$, a' est à gauche de π. Réciproquement, on suppose que a' est à gauche de π. On remarque que $s(i_0, j_0) = (i_0 - 1, j_0)$, appartient à π, et donc $(i_0 - 1, j_1)$ est à droite de π pour $j_1 > j_0$. Comme a' est dans une case différente de $(i_0 - 1, j_0)$ et à gauche de π, (i_0, j_0) ou $(i_0 - 1, j_1)$ pour un $j_1 > j_0$ appartient à π'. Le théorème précédent nous permet donc de conclure que $(i_0, j_0) \in \pi'$. □

Corollaire 20 *Soit $(A, f, (i_0, j_0))$ un élément de $\mathcal{C}$ et π le chemin de l'élément $A(s(i_0, j_0))$ dans A. Avec l'hypothèse que π n'est pas inclus dans la ligne de $s(i_0, j_0)$, tous les candidats sont à gauche de π.*

Preuve : C'est une conséquence immédiate du corollaire 19 et du point (iv) dans la définition de $\mathcal{C}$. □

Nous pouvons maintenant montrer que $\mathcal{C}$ est stable par les deux algorithmes.

Théorème 21 *L'ensemble $\mathcal{C}$ est stable par l'algorithme 1.*

Preuve : Soit $(A, f, (i_0, j_0))$ un élément de $\mathcal{C}$ et $(B, g, s(i_0, j_0))$ son image par l'algorithme 1. On a vu que $(B, g, s(i_0, j_0))$ vérifie les 3 premières conditions de la définition de $\mathcal{C}$, il reste à prouver qu'il vérifie la quatrième.
Par l'étape 2 de l'algorithme 1, on sait que $a = A(s(i_0, j_0))$ est un candidat, et a revient en $s(i_0, j_0)$ lorsqu'on lui applique l'agorithme P'. Soit π le chemin de a dans A, qui est aussi son chemin-inverse dans B. Si le chemin-inverse d'un autre candidat croise π, alors nécessairement les deux chemins-inverses coïncident ensuite jusqu'en $s(i_0, j_0)$, le chemin-inverse ne dépendant pas du contenu de la case.
On remarque d'abord que si $i_0 = 1$, le théorème est évident puisqu'il y a un seul candidat : $A(s(i_0, j_0))$. On suppose donc $i_0 \neq 1$, et donc $s(i_0, j_0) = (i_0 - 1, j_0)$.
Supposons d'abord que π est contenu dans la ligne $i_0 - 1$. Alors les autres candidats sont inchangés, et comme leur chemin-inverse dans A se terminait en (i_0, j_0) dans A, et que les cases de la forme $(i_0 - 1, j_1)$, $j_1 > j_0$ ont leur contenu inchangé ou plus petit dans B que dans A, leur chemin-inverse dans B passe par (i_0, j_0), et termine donc en $(i_0 - 1, j_0)$.
Supposons maintenant que π ne soit pas inclus dans la ligne $i_0 - 1$. Les candidats différents de a sont dans des lignes au-dessus ou en-dessous de celle de a dans B. Puisque la fonction équerre n'a pas changé dans les lignes en-dessous de a, ces candidats sont les mêmes dans B que dans A. Jusqu'à ce qu'ils atteignent la ligne en-dessous de a, leur chemin-inverse reste inchangé, et sur cette ligne il est inchangé ou plus à gauche que dans A. Puisqu'ils étaient à gauche du chemin de a dans A par le corollaire 19, ils atteignent la

ligne de a à gauche de la position de a dans B. Comme le chemin de a part de $(i_0 - 1, j_0)$, il est alors clair que le chemin-inverse de ces éléments croise π, et arrive donc en $(i_0 - 1, j_0)$.
On considère maintenant un candidat de B qui est dans une ligne au-dessus de celle de a. Notons (i, j) sa case. Par l'étape 2 de l'algorithme 1, on sait qu'il y a un candidat dans la case $(i + 1, j + 1)$ de A. Puisque ce candidat est à gauche du chemin de a dans A, on en déduit que (i, j) est strictement à gauche de l'élément le plus à droite de π dans la ligne i. Comme précédemment, on en conclut que le chemin-inverse de (i, j) croise π, et s'arrête en $(i_0 - 1, j_0)$. Le théorème est donc démontré. □

Théorème 22 *$\mathcal{C}$ est stable par l'algorithme 1'.*

Preuve : Soit $(A, f, (i_0, j_0))$ un élément de $\mathcal{C}$, et $(A', f', s^{-1}(i_0, j_0))$ son image par l'algorithme 1'. On a vu que $(A', f', s^{-1}(i_0, j_0))$ vérifie les deux premières conditions de la définition de $\mathcal{C}$.
Regardons la troisième. Soit a le candidat maximal de A. Si b est un candidat strictement au-dessus de a dans A, disons dans la case (i, j), il est aussi strictement à gauche de a : en effet, considérons la case la plus en bas à droite appartenant aux deux chemins inverses. A partir de cette case, les deux chemins doivent coïncider et rejoindre (i_0, j_0). Puisque le code du chemin de a est plus grand que celui de b, et que cette case ne peut pas être la première case du chemin-inverse de a, le mouvement précédent de a est nécessairement W. Puisque la ligne de a est plus basse que celle de b, et que leurs chemins-inverses ne se croisent pas avant, (i, j) est nécessairement strictement à gauche de la case de a dans A. La case $(i+1, j+1)$ appartient donc au diagramme, et la nouvelle valeur de la fonction équerre ne dépasse pas le nombre de cases à droite d'une case donnée.
Voyons maintenant la quatrième condition. La preuve est très proche de celle du théorème précédent. D'abord, si (i_0, j_0) est en bas de sa colonne, la condition est clairement satisfaite puisque le nouveau candidat va nécessairement en $s^{-1}(i_0, j_0)$. On suppose donc que ce n'est pas le cas.
Notons π le chemin de $A'(i_0, j_0)$ dans A', on remarque que que c'est le même ensemble de cases que le chemin-inverse de $A'(i_0, j_0)$ dans A. Supposons d'abord que π est contenu dans la ligne i_0. On a la situation suivante :

a_0	$\dots$	a_k	a
b_0	$\dots$	b_k	b_{k+1}

$\rightarrow$

a	$\dots$	a_{k-1}	a_k
b_0	$\dots$	b_k	b_{k+1}

Comme la fonction équerre ne change pas en-dessous de a, les autres candidats sont inchangés. Comme leur code de chemin est plus petit que celui de a dans A, leur chemin-inverse dans A atteint la ligne de a dans une case à

sa gauche. Donc dans A', leur chemin-inverse contient une case en-dessous d'une case de π. En remarquant que $a_i < b_i$, leur chemin-inverse dans A' va donc en $s^{-1}(i_0, j_0)$.
Supposons maintenant que π n'est pas contenu dans la ligne i_0. Les valeurs de la fonction équerre ne changent pas pour les candidats en-dessous de la dernière ligne de π, et comme leur code de chemin est plus petit que celui de a, par le même argument que dans le cas précédent, leur chemin-inverse atteint la ligne la plus basse de π à gauche de la case la plus à gauche de π sur cette ligne. Ils sont à gauche de π dans A'. Pour ceux qui sont au-dessus, l'idée est la même que précédemment : puisque $A'(i_0, j_0)$ a le plus grand code du chemin, les candidats de A qui sont sur π sont suivis (dans π) par un mouvement vers la droite (ce qui correspond à W) et donc les candidats correspondants dans A' sont à gauche de π. L'idée est la même pour les éléments qui sont strictement à gauche du chemin-inverse de $A'(i_0, j_0)$ dans A. Le corollaire 19 nous permet de conclure. □

Nous pouvons maintenant prouver que les algorithmes sont inverses l'un de l'autre.

Théorème 23 *Soit $(A, f, (i_0, j_0))$ un élément de $\mathcal{C}$, et $(A', f', s^{-1}(i_0, j_0))$ son image par l'algorithme 1'. En appliquant l'algorithme 1 à ce triplet,on obtient $(B', g', (i_0, j_0))$. Alors $A = B'$ et $f = g'$.*

Preuve : D'abord, il est clair que $A = B'$: on applique l'algorithme 1 à l'entier qui était le candidat maximal dans A, et son chemin dans A' passe par les mêmes cases que son chemin-inverse dans A. Comme $A = B'$, on a $f = g'$ car les changements respectifs des fonctions d'équerre sont inverses l'un de l'autre et dépendent uniquement des cases initiales et finales du chemin suivi. □

Théorème 24 *Soit $(A, f, (i_0, j_0))$ un élément de $\mathcal{C}$, et $(B, g, s(i_0, j_0))$ son image par l'algorithme 1. En appliquant l'algorithme 1' à ce triplet,on obtient $(B', g', (i_0, j_0))$. Alors $A = B'$ et $f = g'$.*

Preuve : Montrons que $A = B'$. On a déjà prouvé dans la preuve du théorème 18 que les codes des autres candidats sont plus petits que celui de $A(s(i_0, j_0))$. En effet, soit ils se trouvent sur π et alors le mouvement suivant dans π est vers la droite (W), soit ils rejoignent π par un mouvement vers le haut (N). Le candidat maximal est donc $A(s(i_0, j_0))$. Comme les cases de son chemin-inverse dans B sont les mêmes que celles de son chemin dans A, on a $A = B'$. Le même argument que précédemment nous permet de conclure que $f = g'$. □

Références

[AD99] Aldous and Diaconis. Longest increasing subsequences : From patience sorting to the Baik-Deift-Johansson theorem. *Bulletin of the AMS*, 36(4) :413–432, 1999.

[GNW79] Greene, Nijenhuis, and Wilf. A probabilistic proof of a formula for the number of Young Tableaux of a given shape. *Advances in Mathematics*, (31) :104–109, 1979.

[Knu98] Knuth. *The art of computer programming*, volume 3. Addison-Wesley, 1998.

[NPS97] Novelli, Pak, and Stoyanovskii. A direct bijective proof of the hook-length formula. *Discrete Mathematics and Theoretical Computer Science*, (1) :53–66, 1997.