Probabilités, Statistiques, Combinatoire

Philippe Duchon

Université de Bordeaux – Licence Informatique

<http://www.labri.fr/perso/duchon/Enseignements/Probas/>

2019-2020

Codage des arbres binaires complets

Vu en TD : on peut coder les arbres binaires complets de taille n , par des mots de longueur $2n$ sur un alphabet à 2 lettres. Le codage proposé est défini récursivement (pour toutes les tailles, donc) :

- ▶ l'arbre réduit à une feuille ($n = 0$) est codé par le mot vide ε
- ▶ un arbre non réduit à une feuille, $t = (\bullet, t_g, t_d)$ est codé par le mot

$$w = a \cdot w_1 \cdot b \cdot w_2,$$

où w_1 est le codage de t_g et w_2 est le codage de t_d .

Codage des arbres binaires complets

Vu en TD : on peut coder les arbres binaire complets de taille n , par des mots de longueur $2n$ sur un alphabet à 2 lettres. Le codage proposé est défini récursivement (pour toutes les tailles, donc) :

- ▶ l'arbre réduit à une feuille ($n = 0$) est codé par le mot vide ε
- ▶ un arbre non réduit à une feuille, $t = (\bullet, t_g, t_d)$ est codé par le mot

$$w = a \cdot w_1 \cdot b \cdot w_2,$$

où w_1 est le codage de t_g et w_2 est le codage de t_d .

- ▶ Ce codage est **injectif** (admis) : deux arbres distincts ont des mots de codage différents.

Codage des arbres binaires complets

Vu en TD : on peut coder les arbres binaire complets de taille n , par des mots de longueur $2n$ sur un alphabet à 2 lettres. Le codage proposé est défini récursivement (pour toutes les tailles, donc) :

- ▶ l'arbre réduit à une feuille ($n = 0$) est codé par le mot vide ε
- ▶ un arbre non réduit à une feuille, $t = (\bullet, t_g, t_d)$ est codé par le mot

$$w = a \cdot w_1 \cdot b \cdot w_2,$$

où w_1 est le codage de t_g et w_2 est le codage de t_d .

- ▶ Ce codage est **injectif** (admis) : deux arbres distincts ont des mots de codage différents.
- ▶ **Peut-on décrire l'ensemble des mots qui codent des arbres ?**

Codage des arbres binaires complets

Vu en TD : on peut coder les arbres binaire complets de taille n , par des mots de longueur $2n$ sur un alphabet à 2 lettres. Le codage proposé est défini récursivement (pour toutes les tailles, donc) :

- ▶ l'arbre réduit à une feuille ($n = 0$) est codé par le mot vide ε
- ▶ un arbre non réduit à une feuille, $t = (\bullet, t_g, t_d)$ est codé par le mot

$$w = a \cdot w_1 \cdot b \cdot w_2,$$

où w_1 est le codage de t_g et w_2 est le codage de t_d .

- ▶ Ce codage est **injectif** (admis) : deux arbres distincts ont des mots de codage différents.
- ▶ **Peut-on décrire l'ensemble des mots qui codent des arbres ?**
- ▶ (Si on peut identifier quels sont les mots qui codent les arbres d'une taille donnée, on peut ensuite compter les arbres de cette taille en comptant à la place les mots qui les codent)

Codage des arbres (2)

- ▶ **Mots équilibrés** : tout mot de codage a autant d'occurrences de a que de b : $|w|_a = |w|_b$ (par récurrence sur le nombre de nœuds ; si w_1 et w_2 ont chacun autant d'occurrences de a que de b , c'est vrai aussi pour $w = a \cdot w_1 \cdot b \cdot w_2$)

Codage des arbres (2)

- ▶ **Mots équilibrés** : tout mot de codage a autant d'occurrences de a que de b : $|w|_a = |w|_b$ (par récurrence sur le nombre de nœuds ; si w_1 et w_2 ont chacun autant d'occurrences de a que de b , c'est vrai aussi pour $w = a \cdot w_1 \cdot b \cdot w_2$)
- ▶ **Mots positifs** : dans tout mot w de codage, si on “coupe” w en deux mots u et v (*i.e.* $w = u.v$), **dans le mot u , il y a au moins autant de a que de b** : $|u|_a \geq |u|_b$. (Preuve aussi par récurrence sur n ; attention il faut distinguer plusieurs cas selon où on coupe par rapport à w_1 et w_2)

Positivité des mots de codage

(Propriété P_n de récurrence à l'ordre n : Pour tout arbre binaire t de taille $k \leq n$, son mot de codage w est tel que, si u et v sont deux mots tels que $w = u \cdot v$, alors $|u|_a \geq |u|_b$)

- ▶ La propriété P_0 ne parle que de l'arbre de taille 0 : son mot de codage est ε , et si $u \cdot v = \varepsilon$, on a forcément $u = \varepsilon$; et on a bien $|\varepsilon|_a = 0 \leq 0 = |\varepsilon|_b$.

Positivité des mots de codage

(Propriété P_n de récurrence à l'ordre n : Pour tout arbre binaire t de taille $k \leq n$, son mot de codage w est tel que, si u et v sont deux mots tels que $w = u \cdot v$, alors $|u|_a \geq |u|_b$)

- ▶ La propriété P_0 ne parle que de l'arbre de taille 0 : son mot de codage est ε , et si $u \cdot v = \varepsilon$, on a forcément $u = \varepsilon$; et on a bien $|\varepsilon|_a = 0 \leq 0 = |\varepsilon|_b$.
- ▶ Supposons que pour un entier n , P_n soit vraie; on doit en déduire P_{n+1} .

Positivité des mots de codage

(Propriété P_n de récurrence à l'ordre n : Pour tout arbre binaire t de taille $k \leq n$, son mot de codage w est tel que, si u et v sont deux mots tels que $w = u \cdot v$, alors $|u|_a \geq |u|_b$)

- ▶ La propriété P_0 ne parle que de l'arbre de taille 0 : son mot de codage est ε , et si $u \cdot v = \varepsilon$, on a forcément $u = \varepsilon$; et on a bien $|\varepsilon|_a = 0 \leq 0 = |\varepsilon|_b$.
- ▶ Supposons que pour un entier n , P_n soit vraie ; on doit en déduire P_{n+1} .
 - ▶ Soit donc t un arbre de taille au plus $n + 1$, on doit montrer que son mot de codage w est positif.

Positivité des mots de codage

(Propriété P_n de récurrence à l'ordre n : Pour tout arbre binaire t de taille $k \leq n$, son mot de codage w est tel que, si u et v sont deux mots tels que $w = u \cdot v$, alors $|u|_a \geq |u|_b$)

- ▶ La propriété P_0 ne parle que de l'arbre de taille 0 : son mot de codage est ε , et si $u \cdot v = \varepsilon$, on a forcément $u = \varepsilon$; et on a bien $|\varepsilon|_a = 0 \leq 0 = |\varepsilon|_b$.
- ▶ Supposons que pour un entier n , P_n soit vraie; on doit en déduire P_{n+1} .
 - ▶ Soit donc t un arbre de taille au plus $n + 1$, on doit montrer que son mot de codage w est positif.
 - ▶ **Premier cas** : si t est de taille $\leq n$, on sait que w est un mot positif d'après P_n ;

Positivité des mots de codage

(Propriété P_n de récurrence à l'ordre n : Pour tout arbre binaire t de taille $k \leq n$, son mot de codage w est tel que, si u et v sont deux mots tels que $w = u \cdot v$, alors $|u|_a \geq |u|_b$)

- ▶ La propriété P_0 ne parle que de l'arbre de taille 0 : son mot de codage est ε , et si $u \cdot v = \varepsilon$, on a forcément $u = \varepsilon$; et on a bien $|\varepsilon|_a = 0 \leq 0 = |\varepsilon|_b$.
- ▶ Supposons que pour un entier n , P_n soit vraie ; on doit en déduire P_{n+1} .
 - ▶ Soit donc t un arbre de taille au plus $n + 1$, on doit montrer que son mot de codage w est positif.
 - ▶ **Premier cas** : si t est de taille $\leq n$, on sait que w est un mot positif d'après P_n ;
 - ▶ **Second cas** : si t est de taille $n + 1$, alors t a un sous-arbre gauche t_1 et un sous-arbre droit t_2 , qui sont tous les deux de taille au plus n , donc (d'après P_n) leurs mots de codage w_1 et w_2 sont positifs.

Positivité des mots de codage

(Propriété P_n de récurrence à l'ordre n : Pour tout arbre binaire t de taille $k \leq n$, son mot de codage w est tel que, si u et v sont deux mots tels que $w = u \cdot v$, alors $|u|_a \geq |u|_b$)

- ▶ La propriété P_0 ne parle que de l'arbre de taille 0 : son mot de codage est ε , et si $u \cdot v = \varepsilon$, on a forcément $u = \varepsilon$; et on a bien $|\varepsilon|_a = 0 \leq 0 = |\varepsilon|_b$.
- ▶ Supposons que pour un entier n , P_n soit vraie ; on doit en déduire P_{n+1} .
 - ▶ Soit donc t un arbre de taille au plus $n + 1$, on doit montrer que son mot de codage w est positif.
 - ▶ **Premier cas** : si t est de taille $\leq n$, on sait que w est un mot positif d'après P_n ;
 - ▶ **Second cas** : si t est de taille $n + 1$, alors t a un sous-arbre gauche t_1 et un sous-arbre droit t_2 , qui sont tous les deux de taille au plus n , donc (d'après P_n) leurs mots de codage w_1 et w_2 sont positifs.
 - ▶ Il reste à montrer que si w_1 et w_2 sont des mots positifs, alors le mot $w = a \cdot w_1 \cdot b \cdot w_2$ est un mot positif.

Positivité (2)

$w = aw_1bw_2$; soient u et v deux mots tels que $w = u \cdot v$. On distingue plusieurs cas, selon "où se termine u ".

- ▶ **premier cas** : $u = \varepsilon$, $v = aw_1bw_2$: ε est bien un mot positif ;

Positivité (2)

$w = aw_1bw_2$; soient u et v deux mots tels que $w = u \cdot v$. On distingue plusieurs cas, selon "où se termine u ".

- ▶ **premier cas** : $u = \varepsilon$, $v = aw_1bw_2$: ε est bien un mot positif ;
- ▶ **deuxième cas** : $u = a \cdot u'$ et $v = v'.b.w_2$, avec $u' \cdot v' = w_1$: comme w_1 est positif, on a $|u'|_a \geq |u'|_b$ et donc $|u|_a > |u|_b$;

Positivité (2)

$w = aw_1bw_2$; soient u et v deux mots tels que $w = u \cdot v$. On distingue plusieurs cas, selon "où se termine u ".

- ▶ **premier cas** : $u = \varepsilon$, $v = aw_1bw_2$: ε est bien un mot positif ;
- ▶ **deuxième cas** : $u = a \cdot u'$ et $v = v'.b.w_2$, avec $u' \cdot v' = w_1$:
comme w_1 est positif, on a $|u'|_a \geq |u'|_b$ et donc $|u|_a > |u|_b$;
- ▶ **troisième cas** : $u = aw_1bu'$ et $v = v'$, avec $u' \cdot v' = w_2$;
comme w_2 est positif, on a $|u'|_a \geq |u'|_b$, et comme par ailleurs $|w_1|_a \geq |w_1|_b$, on a bien $|u|_a \geq |u|_b$.

Positivité (2)

$w = aw_1bw_2$; soient u et v deux mots tels que $w = u \cdot v$. On distingue plusieurs cas, selon "où se termine u ".

- ▶ **premier cas** : $u = \varepsilon$, $v = aw_1bw_2$: ε est bien un mot positif ;
- ▶ **deuxième cas** : $u = a \cdot u'$ et $v = v'.b.w_2$, avec $u' \cdot v' = w_1$: comme w_1 est positif, on a $|u'|_a \geq |u'|_b$ et donc $|u|_a > |u|_b$;
- ▶ **troisième cas** : $u = aw_1bu'$ et $v = v'$, avec $u' \cdot v' = w_2$; comme w_2 est positif, on a $|u'|_a \geq |u'|_b$, et comme par ailleurs $|w_1|_a \geq |w_1|_b$, on a bien $|u|_a \geq |u|_b$.
- ▶ Dans tous les cas on a bien $|u|_a \geq |u|_b$, ce qui termine la preuve de P_{n+1} .

Mots de Dyck

Définition - Théorème

Un mot w sur l'alphabet $A = \{a, b\}$ est un mot de Dyck s'il satisfait aux deux conditions suivantes :

- ▶ $|w|_a = |w|_b$
- ▶ Pour tous mots u et v tels que $w = u.v$, on a $|u|_a \geq |u|_b$.

Le codage précédent établit une **bijection** entre les arbres binaires (complets) à n nœuds internes, et les mots de Dyck de longueur $2n$.

Mots de Dyck

Définition - Théorème

Un mot w sur l'alphabet $A = \{a, b\}$ est un mot de Dyck s'il satisfait aux deux conditions suivantes :

- ▶ $|w|_a = |w|_b$
- ▶ Pour tous mots u et v tels que $w = u.v$, on a $|u|_a \geq |u|_b$.

Le codage précédent établit une **bijection** entre les arbres binaires (complets) à n nœuds internes, et les mots de Dyck de longueur $2n$.

Un mot de Dyck, c'est la même chose qu'un mot de parenthèses bien formé : il doit y avoir autant de parenthèses fermantes que d'ouvrantes, et il faut ouvrir une parenthèse avant de la fermer.

Mots de Dyck

Définition - Théorème

Un mot w sur l'alphabet $A = \{a, b\}$ est un mot de Dyck s'il satisfait aux deux conditions suivantes :

- ▶ $|w|_a = |w|_b$
- ▶ Pour tous mots u et v tels que $w = u.v$, on a $|u|_a \geq |u|_b$.

Le codage précédent établit une **bijection** entre les arbres binaires (complets) à n nœuds internes, et les mots de Dyck de longueur $2n$.

Un mot de Dyck, c'est la même chose qu'un mot de parenthèses bien formé : il doit y avoir autant de parenthèses fermantes que d'ouvrantes, et il faut ouvrir une parenthèse avant de la fermer. Pour raisonner sur les mots de Dyck, on peut travailler sur une représentation des mots comme des *chemins*, dans lesquels la lettre a code un pas Nord-Est $(1, 1)$, et la lettre b code un pas Sud-Est ; les chemins commencent en $(0, 0)$. ("Chemins de Dyck")

Chemins de Dyck

On dessine un mot sur l'alphabet $\{a, b\}$ comme décrivant un "chemin" partant de l'origine $(0, 0)$:

- ▶ pour chaque lettre a , on fait un pas vers le Nord-Est (de (x, y) à $(x + 1, y + 1)$)
- ▶ pour chaque lettre b , on fait un pas vers le Sud-Est (de (x, y) à $(x + 1, y - 1)$)

Dire que le mot w est un mot de Dyck, c'est équivalent à deux conditions sur le chemin :

- ▶ il fait autant de pas Nord-Est que Sud-Est : il se termine donc sur l'axe horizontal, en un point $(2n, 0)$;
- ▶ n'importe quel point du chemin (correspondant à ne lire que le début du mot) peut se trouver au-dessus de l'axe horizontal, ou sur l'axe, mais pas dessous (car parmi les k premiers pas, il y a au moins autant de Nord-Est que de Sud-Est)

Comptage des mots de Dyck et arbres binaires complets

- ▶ On a vu que les arbres binaires complets non vides (avec comme taille, le nombre de nœuds internes) ont pour suite de comptage les **nombre de Catalan** ;

Comptage des mots de Dyck et arbres binaires complets

- ▶ On a vu que les arbres binaires complets non vides (avec comme taille, le nombre de nœuds internes) ont pour suite de comptage les **nombre de Catalan** ;
- ▶ On connaît une formule de récurrence satisfaite par les nombres de Catalan : $c_0 = 1$, et pour $n \geq 1$,

$$c_n = \sum_{k=0}^{n-1} c_k c_{n-1-k}$$

Comptage des mots de Dyck et arbres binaires complets

- ▶ On a vu que les arbres binaires complets non vides (avec comme taille, le nombre de nœuds internes) ont pour suite de comptage les **nombre de Catalan** ;
- ▶ On connaît une formule de récurrence satisfaite par les nombres de Catalan : $c_0 = 1$, et pour $n \geq 1$,

$$c_n = \sum_{k=0}^{n-1} c_k c_{n-1-k}$$

- ▶ Cela permet de calculer rapidement les premiers termes de la suite : 1, 1, 2, 5, 14, 42, 129 . . . , mais cela ne donne pas une formule directe pour c_n en fonction de n .

Comptage des mots de Dyck et arbres binaires complets

- ▶ On a vu que les arbres binaires complets non vides (avec comme taille, le nombre de nœuds internes) ont pour suite de comptage les **nombre de Catalan** ;
- ▶ On connaît une formule de récurrence satisfaite par les nombres de Catalan : $c_0 = 1$, et pour $n \geq 1$,

$$c_n = \sum_{k=0}^{n-1} c_k c_{n-1-k}$$

- ▶ Cela permet de calculer rapidement les premiers termes de la suite : 1, 1, 2, 5, 14, 42, 129 . . . , mais cela ne donne pas une formule directe pour c_n en fonction de n .
- ▶ Il se trouve qu'on connaît une formule assez simple pour c_n :

$$c_n = \frac{(2n)!}{n!(n+1)!} = \frac{1}{n+1} \binom{2n}{n}$$

Comptage des mots de Dyck et arbres binaires complets

- ▶ On a vu que les arbres binaires complets non vides (avec comme taille, le nombre de nœuds internes) ont pour suite de comptage les **nombre de Catalan** ;
- ▶ On connaît une formule de récurrence satisfaite par les nombres de Catalan : $c_0 = 0$, et pour $n \geq 1$,

$$c_n = \sum_{k=0}^{n-1} c_k c_{n-1-k}$$

- ▶ Cela permet de calculer rapidement les premiers termes de la suite : 1, 1, 2, 5, 14, 42, 129 . . . , mais cela ne donne pas une formule directe pour c_n en fonction de n .
- ▶ Il se trouve qu'on connaît une formule assez simple pour c_n :

$$c_n = \frac{(2n)!}{n!(n+1)!} = \frac{1}{n+1} \binom{2n}{n}$$

- ▶ On va donner deux preuves distinctes :

Comptage des mots de Dyck et arbres binaires complets

- ▶ On a vu que les arbres binaires complets non vides (avec comme taille, le nombre de nœuds internes) ont pour suite de comptage les **nombre de Catalan** ;
- ▶ On connaît une formule de récurrence satisfaite par les nombres de Catalan : $c_0 = 0$, et pour $n \geq 1$,

$$c_n = \sum_{k=0}^{n-1} c_k c_{n-1-k}$$

- ▶ Cela permet de calculer rapidement les premiers termes de la suite : 1, 1, 2, 5, 14, 42, 129 . . . , mais cela ne donne pas une formule directe pour c_n en fonction de n .
- ▶ Il se trouve qu'on connaît une formule assez simple pour c_n :

$$c_n = \frac{(2n)!}{n!(n+1)!} = \frac{1}{n+1} \binom{2n}{n}$$

- ▶ On va donner deux preuves distinctes :
 - ▶ Une preuve sur les mots de Dyck ;

Comptage des mots de Dyck et arbres binaires complets

- ▶ On a vu que les arbres binaires complets non vides (avec comme taille, le nombre de nœuds internes) ont pour suite de comptage les **nombre de Catalan** ;
- ▶ On connaît une formule de récurrence satisfaite par les nombres de Catalan : $c_0 = 1$, et pour $n \geq 1$,

$$c_n = \sum_{k=0}^{n-1} c_k c_{n-1-k}$$

- ▶ Cela permet de calculer rapidement les premiers termes de la suite : 1, 1, 2, 5, 14, 42, 129 . . . , mais cela ne donne pas une formule directe pour c_n en fonction de n .
- ▶ Il se trouve qu'on connaît une formule assez simple pour c_n :

$$c_n = \frac{(2n)!}{n!(n+1)!} = \frac{1}{n+1} \binom{2n}{n}$$

- ▶ On va donner deux preuves distinctes :
 - ▶ Une preuve sur les mots de Dyck ;
 - ▶ Une preuve sur les arbres, en prouvant une récurrence

Chemins de Dyck : le “lemme cyclique”

On doit prouver : $\binom{2n}{n} = (n+1)c_n$

- ▶ à gauche, on reconnaît le nombre de **mots équilibrés** de longueur $2n$, donc de chemins (à pas Nord-Est et Sud-Est) allant de $(0,0)$ à $(2n,0)$;

Chemins de Dyck : le “lemme cyclique”

On doit prouver : $\binom{2n}{n} = (n+1)c_n$

- ▶ à gauche, on reconnaît le nombre de **mots équilibrés** de longueur $2n$, donc de chemins (à pas Nord-Est et Sud-Est) allant de $(0,0)$ à $(2n,0)$;
- ▶ à droite, c_n est le nombre de chemins (mots) de Dyck (équilibrés et positifs) de longueur $2n$; $n+1$, c'est le nombre d'entiers de 1 à $n+1$ (bof).

Chemins de Dyck : le “lemme cyclique”

On doit prouver : $\binom{2n}{n} = (n+1)c_n$

- ▶ à gauche, on reconnaît le nombre de **mots équilibrés** de longueur $2n$, donc de chemins (à pas Nord-Est et Sud-Est) allant de $(0,0)$ à $(2n,0)$;
- ▶ à droite, c_n est le nombre de chemins (mots) de Dyck (équilibrés et positifs) de longueur $2n$; $n+1$, c'est le nombre d'entiers de 1 à $n+1$ (bof).
- ▶ Pour “interpréter” plus facilement $n+1$, on ajoute un pas Sud-Est à la fin du chemin (ou une lettre b à la fin du mot : $w' = w.b$)

Chemins de Dyck : le “lemme cyclique”

On doit prouver : $\binom{2n}{n} = (n+1)c_n$

- ▶ à gauche, on reconnaît le nombre de **mots équilibrés** de longueur $2n$, donc de chemins (à pas Nord-Est et Sud-Est) allant de $(0,0)$ à $(2n,0)$;
- ▶ à droite, c_n est le nombre de chemins (mots) de Dyck (équilibrés et positifs) de longueur $2n$; $n+1$, c'est le nombre d'entiers de 1 à $n+1$ (bof).
- ▶ Pour “interpréter” plus facilement $n+1$, on ajoute un pas Sud-Est à la fin du chemin (ou une lettre b à la fin du mot : $w' = w.b$)
- ▶ $(n+1)$, c'est le nombre de pas Sud-Est dans le chemin prolongé (lettres b dans w')

Chemins de Dyck : le “lemme cyclique”

On doit prouver : $\binom{2n}{n} = (n+1)c_n$

- ▶ à gauche, on reconnaît le nombre de **mots équilibrés** de longueur $2n$, donc de chemins (à pas Nord-Est et Sud-Est) allant de $(0,0)$ à $(2n,0)$;
- ▶ à droite, c_n est le nombre de chemins (mots) de Dyck (équilibrés et positifs) de longueur $2n$; $n+1$, c'est le nombre d'entiers de 1 à $n+1$ (bof).
- ▶ Pour “interpréter” plus facilement $n+1$, on ajoute un pas Sud-Est à la fin du chemin (ou une lettre b à la fin du mot : $w' = w.b$)
- ▶ $(n+1)$, c'est le nombre de pas Sud-Est dans le chemin prolongé (lettres b dans w')
- ▶ $(n+1)c_n$, c'est le nombre de chemins de Dyck prolongés, avec un des pas Sud-Est “marqué” (ou : le nombre de mots de Dyck prolongés, avec une lettre b marquée)

Chemins de Dyck : le “lemme cyclique”

On doit prouver : $\binom{2n}{n} = (n+1)c_n$

- ▶ à gauche, on reconnaît le nombre de **mots équilibrés** de longueur $2n$, donc de chemins (à pas Nord-Est et Sud-Est) allant de $(0,0)$ à $(2n,0)$;
- ▶ à droite, c_n est le nombre de chemins (mots) de Dyck (équilibrés et positifs) de longueur $2n$; $n+1$, c'est le nombre d'entiers de 1 à $n+1$ (bof).
- ▶ Pour “interpréter” plus facilement $n+1$, on ajoute un pas Sud-Est à la fin du chemin (ou une lettre b à la fin du mot : $w' = w.b$)
- ▶ $(n+1)$, c'est le nombre de pas Sud-Est dans le chemin prolongé (lettres b dans w')
- ▶ $(n+1)c_n$, c'est le nombre de chemins de Dyck prolongés, avec un des pas Sud-Est “marqué” (ou : le nombre de mots de Dyck prolongés, avec une lettre b marquée)
- ▶ On va montrer une bijection : comment passer du chemin C à un chemin de Dyck prolongé D , avec un pas Sud-Est marqué ; et comment revenir en arrière.

Bijection : sens direct

- ▶ On commence par “marquer” le dernier pas du chemin C .

Bijection : sens direct

- ▶ On commence par “marquer” le dernier pas du chemin C .
- ▶ On identifie, sur le chemin marqué, le point le plus bas ; en cas d'égalité, le plus à gauche parmi les points les plus bas. Soit $M = (x, y)$ ce point.

Bijection : sens direct

- ▶ On commence par “marquer” le dernier pas du chemin C .
- ▶ On identifie, sur le chemin marqué, le point le plus bas ; en cas d'égalité, le plus à gauche parmi les points les plus bas. Soit $M = (x, y)$ ce point.
- ▶ Forcément, $y < 0$, donc M n'est pas le point initial $(0, 0)$; et le pas qui précède M est un pas Sud-Est.

Bijection : sens direct

- ▶ On commence par “marquer” le dernier pas du chemin C .
- ▶ On identifie, sur le chemin marqué, le point le plus bas ; en cas d'égalité, le plus à gauche parmi les points les plus bas. Soit $M = (x, y)$ ce point.
- ▶ Forcément, $y < 0$, donc M n'est pas le point initial $(0, 0)$; et le pas qui précède M est un pas Sud-Est.
- ▶ On obtient le mot de Dyck prolongé, en suivant les pas du chemin à partir du point M jusqu'à la fin, puis depuis le début jusqu'au point M ; le pas marqué est celui qui était à la fin avant l'opération.

Bijection : sens direct

- ▶ On commence par “marquer” le dernier pas du chemin C .
- ▶ On identifie, sur le chemin marqué, le point le plus bas ; en cas d'égalité, le plus à gauche parmi les points les plus bas. Soit $M = (x, y)$ ce point.
- ▶ Forcément, $y < 0$, donc M n'est pas le point initial $(0, 0)$; et le pas qui précède M est un pas Sud-Est.
- ▶ On obtient le mot de Dyck prolongé, en suivant les pas du chemin à partir du point M jusqu'à la fin, puis depuis le début jusqu'au point M ; le pas marqué est celui qui était à la fin avant l'opération.
- ▶ En termes de mots : on avait un mot $w.b$; pour “marquer” le dernier b on le remplace par un b' ; on “coupe” $w.b'$ en w_1 et w_2b' , et on échange les deux parties : on obtient w_2bw_1 .
Note : le mot w_1 se termine par un b (le dernier pas avant M était un Sud-Est).

Bijection : sens direct

- ▶ On commence par “marquer” le dernier pas du chemin C .
- ▶ On identifie, sur le chemin marqué, le point le plus bas ; en cas d'égalité, le plus à gauche parmi les points les plus bas. Soit $M = (x, y)$ ce point.
- ▶ Forcément, $y < 0$, donc M n'est pas le point initial $(0, 0)$; et le pas qui précède M est un pas Sud-Est.
- ▶ On obtient le mot de Dyck prolongé, en suivant les pas du chemin à partir du point M jusqu'à la fin, puis depuis le début jusqu'au point M ; le pas marqué est celui qui était à la fin avant l'opération.
- ▶ En termes de mots : on avait un mot $w.b$; pour “marquer” le dernier b on le remplace par un b' ; on “coupe” $w.b'$ en w_1 et w_2b' , et on échange les deux parties : on obtient w_2bw_1 .
Note : le mot w_1 se termine par un b (le dernier pas avant M était un Sud-Est).
- ▶ Parce qu'on a choisi pour M le point le plus à gauche parmi les plus bas, le nouveau chemin D reste au-dessus de l'axe horizontal, sauf lors de son dernier pas.

Bijection : sens indirect

- ▶ On part d'un chemin de Dyck prolongé d'un pas Sud-Est, avec l'un de ses pas Sud-Est marqués.

Bijection : sens indirect

- ▶ On part d'un chemin de Dyck prolongé d'un pas Sud-Est, avec l'un de ses pas Sud-Est marqués.
- ▶ Le découpage-échange doit amener le pas marqué à la fin : on n'a pas le choix, il faut couper juste après le pas marqué et échanger les parties gauche et droite.

Bijection : sens indirect

- ▶ On part d'un chemin de Dyck prolongé d'un pas Sud-Est, avec l'un de ses pas Sud-Est marqués.
- ▶ Le découpage-échange doit amener le pas marqué à la fin : on n'a pas le choix, il faut couper juste après le pas marqué et échanger les parties gauche et droite.
- ▶ L'échange amène la fin du chemin en position du point le plus à gauche parmi les plus bas, et on obtient bien un chemin équilibré prolongé d'un pas Sud-Est.

Bijection sur les arbres : algorithme de Remy

- ▶ On pose $c'_n = \frac{(2n)!}{n!(n+1)!}$; il faut prouver $c'_n = c_n$.

Bijection sur les arbres : algorithme de Remy

- ▶ On pose $c'_n = \frac{(2n)!}{n!(n+1)!}$; il faut prouver $c'_n = c_n$.
- ▶ En calculant c'_{n+1}/c'_n , on obtient la relation :

$$c'_{n+1} = \frac{2(2n+1)}{n+2} c'_n$$

Bijection sur les arbres : algorithme de Remy

- ▶ On pose $c'_n = \frac{(2n)!}{n!(n+1)!}$; il faut prouver $c'_n = c_n$.
- ▶ En calculant c'_{n+1}/c'_n , on obtient la relation :

$$c'_{n+1} = \frac{2(2n+1)}{n+2} c'_n$$

- ▶ Comme on a $c_0 = c'_0$, si on peut montrer que la suite de Catalan vérifie la même relation de récurrence, on peut en déduire que $c_n = c'_n$ pour tout n .

Bijection sur les arbres : algorithme de Remy

- ▶ On pose $c'_n = \frac{(2n)!}{n!(n+1)!}$; il faut prouver $c'_n = c_n$.
- ▶ En calculant c'_{n+1}/c'_n , on obtient la relation :

$$c'_{n+1} = \frac{2(2n+1)}{n+2} c'_n$$

- ▶ Comme on a $c_0 = c'_0$, si on peut montrer que la suite de Catalan vérifie la même relation de récurrence, on peut en déduire que $c_n = c'_n$ pour tout n .
- ▶ On réécrit la relation de récurrence pour faire apparaître des produits :

$$2(2n+1)c_n = (n+2)c_{n+1}$$

Bijection sur les arbres : algorithme de Remy

- ▶ On pose $c'_n = \frac{(2n)!}{n!(n+1)!}$; il faut prouver $c'_n = c_n$.
- ▶ En calculant c'_{n+1}/c'_n , on obtient la relation :

$$c'_{n+1} = \frac{2(2n+1)}{n+2} c'_n$$

- ▶ Comme on a $c_0 = c'_0$, si on peut montrer que la suite de Catalan vérifie la même relation de récurrence, on peut en déduire que $c_n = c'_n$ pour tout n .
- ▶ On réécrit la relation de récurrence pour faire apparaître des produits :

$$2(2n+1)c_n = (n+2)c_{n+1}$$

- ▶ On interprète le produit de gauche et le produit de droite, et on prouve une bijection !