Εφαρμοσμένη Συνδυαστική

Κώστας Μανές

Τμήμα Πληροφορικής, Πανεπιστήμιο Πειραιώς

2021-2022

Βιβλιογραφία

Σύγγραμμα μαθήματος:

Ε. Φούντας, Εφαρμογές Συνδυαστικής Ανάλυσης 2, Εκδόσεις Βαρβαρήγου (2013).

Συμπληρωματική βιβλιογραφία:

1. D.L. Kreher, D.R. Stinson, Combinatorial Algorithms: Generation, Enumeration and Search, CRC press LTC, Florida (1998)
2. D. E. Knuth, The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Addison-Wesley (2011)
3. F. Ruskey, Combinatorial generation (2003), available online at https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.93.5967
4. T. Keller, T. Trotter, Applied Combinatorics, CreateSpace Independent Publishing Platform (2017)

Βιβλιογραφία

Συμπληρωματική βιβλιογραφία (2):

1. A. Tucker, Applied Combinatorics, Wiley, 6th edition (2012)
2. E. A. Bender, S. G. Williamson, Foundations of Combinatorics with Applications, Dover Publications (2013)
3. D. B. West, Combinatorial Mathematics, Cambridge University Press (2021)
4. R. P. Stanley, Algebraic Combinatorics: Walks, Trees, Tableaux and more, Springer (2018)

Συνδυαστικά αντικείμενα

Μαθηματικά αντικείμενα που έχουν συνδυαστική δομή, δηλαδή κατασκευάζονται συνδυάζοντας αντικείμενα ‘‘μικρότερου’’ μεγέθους της ίδιας ή άλλων κατηγοριών.

- Βασικά αντικείμενα: Λέξεις, μεταθέσεις, συνδυασμοί, σύνολα, κτλ.
- Πιο σύνθετα αντικείμενα: Γραφήματα, δένδρα, σχέσεις, διαμερίσεις, πίνακες, tableaux, κτλ.

Συνδυαστικοί Αλγόριθμοι

Αλγόριθμοι που αφορούν συνδυαστικά αντικείμενα.

Βασικές κατηγορίες προβλημάτων που επιλύουν οι συνδυαστικοί αλγόριθμοι:

- **Generation**: Κατασκευή όλων των αντικειμένων ενός συγκεκριμένου τύπου
- **Enumeration**: Απαρίθμηση όλων των αντικειμένων ενός συγκεκριμένου τύπου (χωρίς να γίνεται απαραίτητα η κατασκευή τους).
- **Search**: Αναζήτηση (κατασκευή) ενός αντικειμένου με συγκεκριμένες ιδιότητες (αν υπάρχει).
- **Optimization** (Βελτιστοποίηση): Εύρεση του αντικειμένου εκείνου που είναι το καλύτερο από όλα κατά κάποια έννοια.

Τα προβλήματα αυτά συνήθως είναι ‘‘δύσκολα’’, δηλαδή δεν υπάρχουν γνωστοί γρήγοροι αλγόριθμοι γι΄ αυτά.

Παραδείγματα

- Εύρεση της μέγιστης κλίκας σε ένα γράφημα.
- Εύρεση κύκλου Hamilton σε ένα γράφημα.

Μέθοδοι προσέγγισης δύσκολων προβλημάτων αναζήτησης-βελτιστοποίησης

- **Exhaustive search** (backtracking, branch and bound)
- **Dynamic programming**
- **Heuristic search**: Εύρεση μιας σχεδόν βέλτιστης λύσης με χρονικό περιορισμό, χωρίς εξάντληση όλων των δυνατών επιλογών.
- **Approximation Algorithms**

Άλλα προβλήματα κατασκευής

Έστω $\mathcal{A}_n$ το σύνολο των δυαδικών λέξεων μήκους n χωρίς διαδοχικά 1.

- **Λεξικογραφική κατασκευή**
  Κατασκευή της λίστας των λέξεων του $\mathcal{A}_n$ με τέτοια σειρά ώστε κάθε λέξη να είναι μικρότερη λεξικογραφικά από την επόμενή της στην λίστα.
- **Ranking - Unranking**
  Rank: Πόσες προηγούνται λεξικογραφικά της $101001 \in A_6$;
  Unrank: Ποια είναι η 5η λέξη στη λεξικογραφική λίστα του A_6;
- **Random generation**
  Κατασκευή μιας τυχαίας λέξης του $\mathcal{A}_n$, όταν όλες είναι ισοπίθανες.
- **Gray codes**
  Κατασκευή της λίστας των λέξεων του $\mathcal{A}_n$ με τέτοια σειρά ώστε κάθε λέξη να διαφέρει από την επόμενή της στην λίστα σε ακριβώς ένα γράμμα.
- **Κατασκευή και απαρίθμηση με συμμετρίες**
  Κατασκευή των λέξεων του $\mathcal{A}_n$ όταν π.χ. κάθε λέξη $w_1 w_2 \cdots w_n$ θεωρείται ίδια με την $w_n w_{n-1} \cdots w_1$ και κατασκευάζεται μόνο η μία εκ των δύο.

n-queens puzzle: Να τοποθετηθούν n βασίλισσες σε μια $n \times n$ σκακιέρα ώστε να μην υπάρχουν 2 στην ίδια γραμμή, στήλη ή διαγώνιο.

Προφανώς, σε κάθε λύση, κάθε στήλη περιέχει ακριβώς μια βασίλισσα.
Έστω x_k η γραμμή στην οποία βρίσκεται η βασίλισσα της k στήλης.
Μια λύση $\mathbf{x} = (x_1, \ldots, x_n)$ προφανώς είναι μια μετάθεση του $[n]$, αφού $1 \leq i < j \leq n \Rightarrow x_i \neq x_j$.
Επιπλέον, η συνθήκη

$$1 \leq i < j \leq n \Rightarrow |x_i - x_j| \neq j - i$$

εξασφαλίζει ότι δεν υπάρχουν δύο βασίλισσες στην ίδια διαγώνιο. Για την εύρεση όλων των λύσεων, αρκεί λοιπόν να κατασκευαστούν όλες οι μεταθέσεις του $[n]$ που ικανοποιούν την παραπάνω συνθήκη.

Backtracking - n-queens

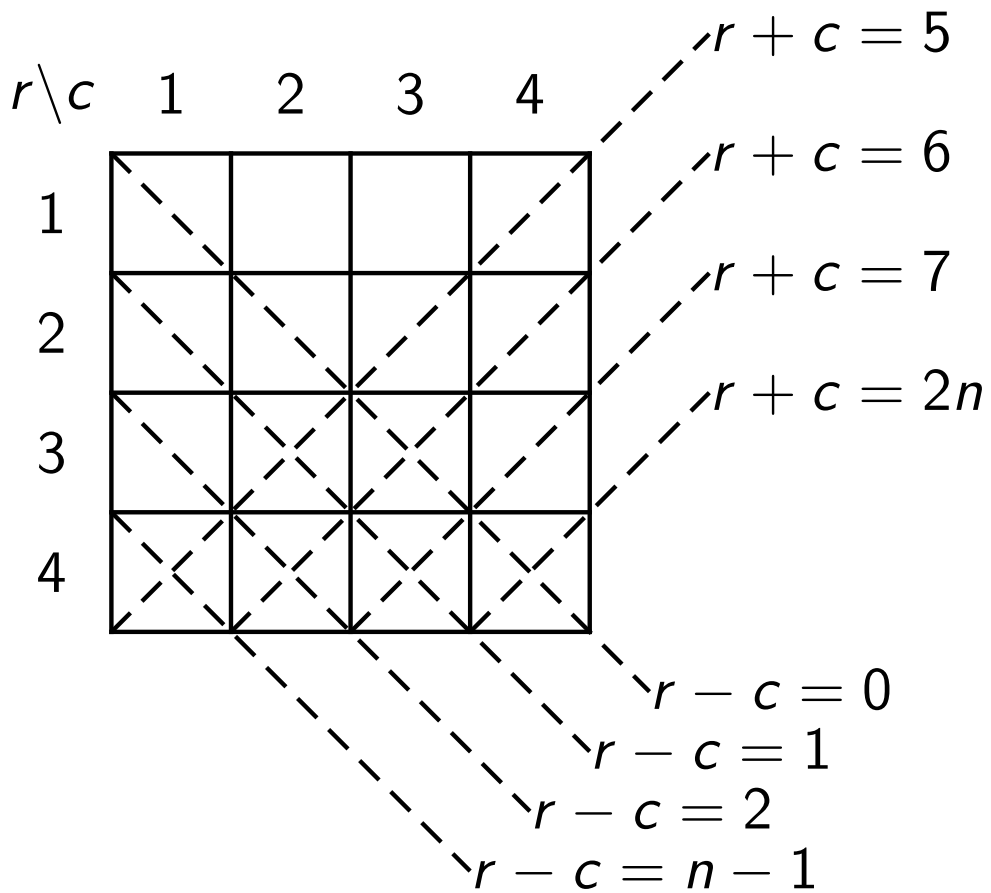
Για τον αποδοτικό έλεγχο της συνθήκης, ορίζουμε τις μεταβλητές (arrays) A, B, C, με

$$A[i] = \text{false} \Leftrightarrow \text{ η γραμμή } i \text{ περιέχει βασίλισσα}, 1 \leq i \leq n$$
$$B[i] = \text{false} \Leftrightarrow \text{ η διαγώνιος } r + c = i \text{ περιέχει βασίλισσα}, 2 \leq i \leq 2n$$
$$C[i] = \text{false} \Leftrightarrow \text{ η διαγώνιος } r - c = i \text{ περιέχει βασίλισσα}, |i| \leq n - 1$$

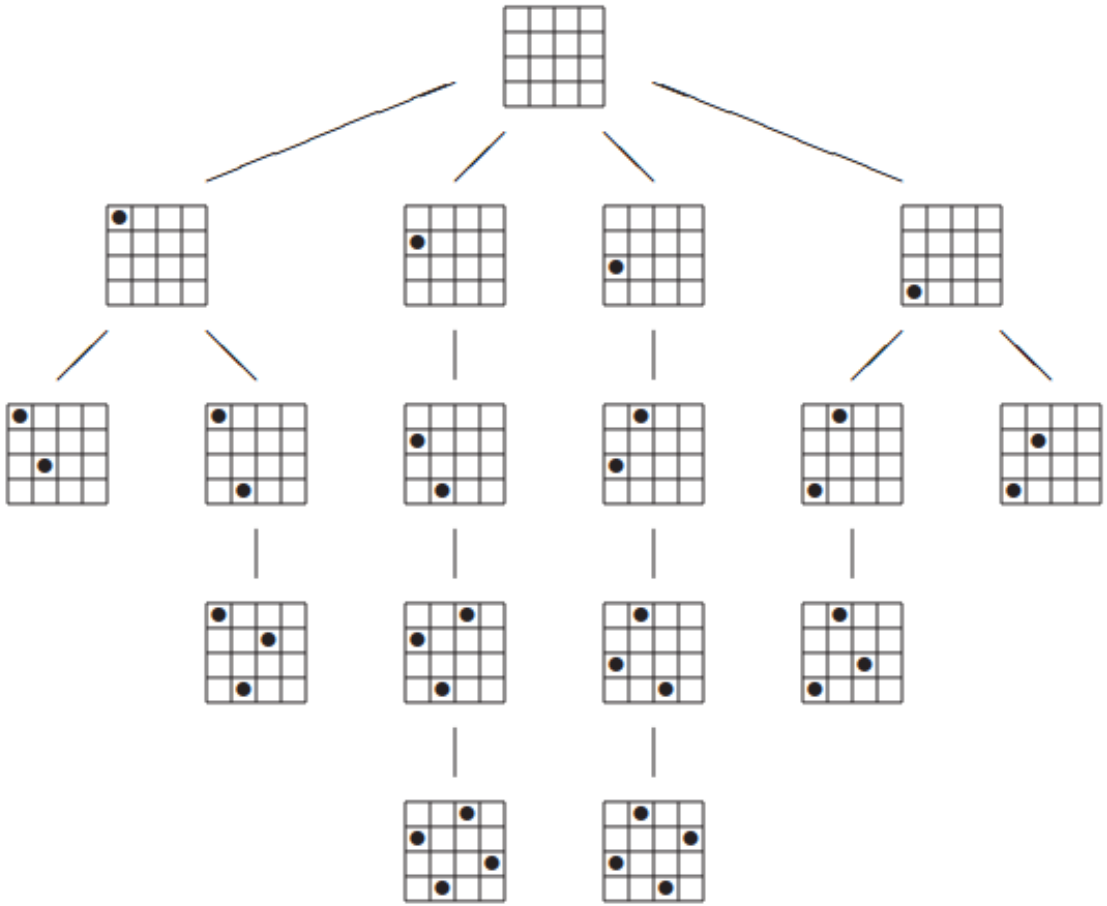
```
1 Queens(c) begin
2     for r := 1 to n do
3         if A[r] and B[r + c] and C[r − c] then
4             x_c := r;
5             A[r] := B[r + c] := C[r − c] := false;
6             if c < n then Queens(c + 1) else output(x);
7             A[r] := B[r + c] := C[r − c] := true;
```

Αλγόριθμος 1: Αλγόριθμος backtracking για το n-queens puzzle.

Backtracking n-queens



Backtracking n-queens



Σχήμα: Το δένδρο της αναδρομής για το 4-queens puzzle.

Η μέθοδος backtracking εξετάζει εξαντλητικά τον χώρο (search space) όλων των πιθανών λύσεων. Συνήθως κάθε τέτοια δυνατή λύση εκφράζεται ως μια λέξη $\mathbf{x} = x_1 \cdots x_n$ σε κάποιο αλφάβητο $A_1 \times \cdots \times A_n$. Η $\mathbf{x}$ είναι αποδεκτή λύση όταν ικανοποιεί κάποιο κατηγόρημα P. Επεκτείνοντας το P σε ένα κατηγόρημα P_k για τα προθέματα $x_1 \cdots x_k$ μήκους k, τέτοιο ώστε $(\exists y \in A_k, P_k(x_1 \cdots x_{k-1} y)) \Rightarrow P_{k-1}(x_1 \cdots x_{k-1})$ και ορίζοντας $C_k(\mathbf{x}) = \{y \in A_k : P_k(\mathbf{x}y)\}$, ο αλγόριθμος περιγράφεται γενικά όπως παρακάτω:

```
1 Back(k, x) begin
2     if P(x) then output(x);
3     compute(C_k(x));
4     foreach y ∈ C_k(x) do Back(k + 1, xy);
```

Αλγόριθμος 2: Γενικός αλγόριθμος backtracking.

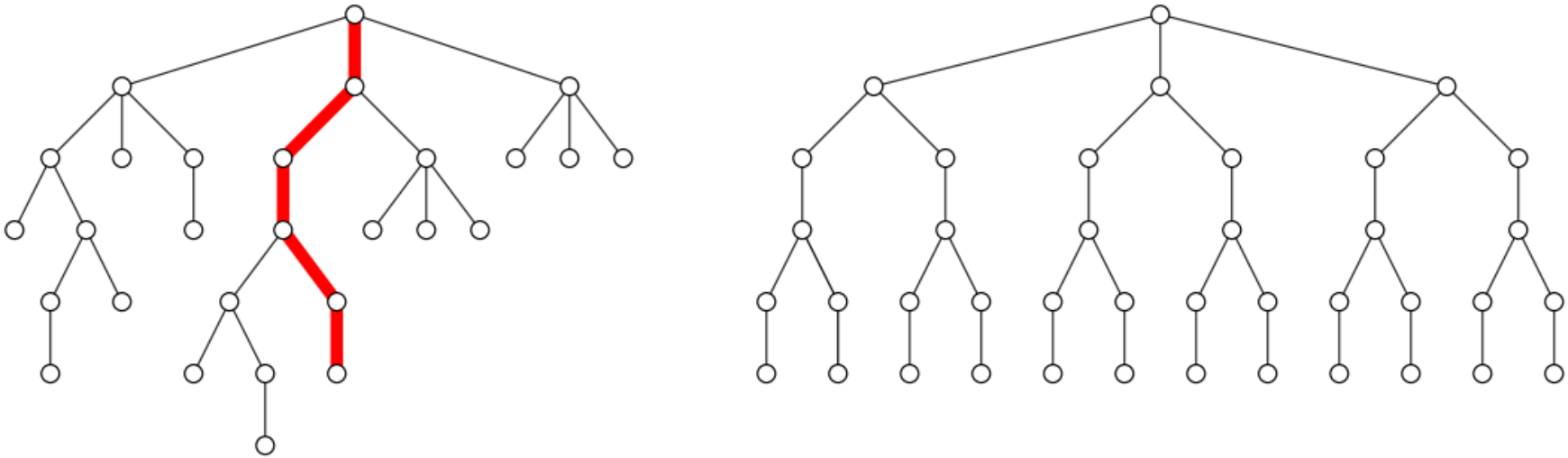
Backtracking - Εκτίμηση χώρου αναζήτησης

Για να εκτιμήσουμε το μέγεθος του χώρου αναζήτησης, δηλαδή το μέγεθος $|T|$ του δένδρου αναδρομής T, προτού εκτελέσουμε έναν αλγόριθμο backtracking, επιλέγουμε τυχαία ένα μονοπάτι $P = (x_1 = r, x_2, \ldots, x_t)$ από τη ρίζα r μέχρι κάποιο φύλλο x_t, όπου ο κόμβος x_i έχει $\mathrm{outdeg}(x_i) = n_i$ παιδιά, και θεωρούμε ότι κάθε μονοπάτι στο δένδρο έχει την ίδια ακολουθία βαθμών. Το εκτιμώμενο πλήθος κόμβων στο επίπεδο i είναι μια τυχαία μεταβλητή $X_i = n_1 n_2 \cdots n_{i-1} = X_{i-1} n_{i-1}$, $X_1 = 1$, και θέτουμε $X = \sum_{i=1}^{t} X_i$. Επιπλέον ορίζουμε για κάθε $v \in T$ την τ.μ. $I(v) = [v \in P]$, και τη συνάρτηση $\tau(v) = \mathrm{outdeg}(\bar{v})\tau(\bar{v})$, όπου $\bar{v}$ ο γονιός του v και $\tau(r) = 1$, οπότε $\tau(v) = \prod_{u \text{ πρόγονος του } v} \mathrm{outdeg}(u)$ και $X = \sum_{v \in T} \tau(v) I(v)$. Κατόπιν τούτων, η μέση τιμή της τ.μ. X ισούται με

$$E(X) = \sum_{v \in T} \tau(v) E(I(v)) = \sum_{v \in T} \tau(v) \frac{1}{\tau(v)} = |T|$$

δηλαδή η X είναι μια αμερόληπτη εκτιμήτρια του $|T|$.

Backtracking - Εκτίμηση χώρου αναζήτησης



Σχήμα: Ένα τυχαία επιλεγμένο μονοπάτι (κόκκινο) και το εκτιμώμενο δένδρο της αναδρομής (δεξιά) βάσει αυτού του μονοπατιού.

Backtracking - Εκτίμηση χώρου αναζήτησης

Επιλέγοντας αρκετά τέτοια τυχαία μονοπάτια, έχουμε τελικά μια πολύ καλή εκτίμηση για το $|T|$.

```
1  sum := 0;
2  for i := 1 to N do
3      X := product := k := 1;
4      compute(C_1);
5      while C_k ≠ ∅ do
6          n_k := |C_k|;
7          product := n_k · product;
8          X := X + product;
9          x_k := an element of C_k, chosen at random;
10         k := k + 1;
11         compute(C_k);
12     sum := sum + X;
13 output(sum/N);
```

Αλγόριθμος 3: Αλγόριθμος εκτίμησης του $|T|$.

Backtracking - Περισσότερα παραδείγματα

- $(0, 1)$-knapsack problem
- Εύρεση κλικών ενός γραφήματος
- Εύρεση μέγιστης κλίκας γραφήματος

Backtracking - Ασκήσεις

- Knight's tour: Να υλοποποιηθεί αλγόριθμος backtracking για την εύρεση ενός μονοπατιού ενός ίππου σε μια 6×6 σκακιέρα που ξεκινά από την πάνω αριστερή γωνία και επισκέπτεται κάθε τετράγωνο ακριβώς μία φορά.
- Να υλοποιηθεί αλγόριθμος backtracking για την απαρίθμηση των διαφορετικών χρωματισμών των κορυφών ενός γραφήματος με k χρώματα (δύο γειτονικές κορυφές δεν επιτρέπεται να έχουν το ίδιο χρώμα).

Κατασκευή συνδυαστικών αντικειμένων

Δίνεται ένα σύνολο $\mathcal{A}$ συνδυαστικών αντικειμένων και ζητείται η κατασκευή όλων των στοιχείων του $\mathcal{A}$ μεγέθους n. Το σύνολο αυτών των στοιχείων συμβολίζεται με $\mathcal{A}_n$, ενώ η έννοια του μεγέθους έχει προκαθορισθεί κατάλληλα ώστε το $\mathcal{A}_n$ να είναι πάντα πεπερασμένο. Στις εφαρμογές, συνήθως ο πληθάριθμος $|\mathcal{A}_n|$ απειρίζεται με γρήγορο ρυθμό, ώστε κάθε αλγόριθμος κατασκευής να είναι αναγκαστικά αργός, αφού ακόμα και αν κάθε στοιχείο κατασκευάζεται σε σταθερό χρόνο, η πολυπλοκότητα χρόνου θα είναι $\Omega(|\mathcal{A}_n|)$. Για το λόγο αυτό η ανάλυση αυτών των αλγορίθμων ασχολείται με τον χρόνο κατασκευής κάθε αντικειμένου, παρά με τον συνολικό χρόνο εκτέλεσης.

Constant Amortized Time (CAT)

Ένας αλγόριθμος κατασκευής είναι CAT (constant amortized time) αν κάθε αντικείμενο κατασκευάζεται κατά μέσο όρο σε σταθερό χρόνο, δηλαδή

$$\frac{\text{συνολικός χρόνος για όλα τα αντικείμενα}}{\text{πλήθος αντικειμένων}} \leq \text{σταθερά}.$$

Ένας τέτοιος αλγόριθμος κατασκευής έχει τη θεωρητικά βέλτιστη πολυπλοκότητα, δηλαδή $\Theta(|\mathcal{A}_n|)$.

Constant Amortized Time (CAT)

Στην περίπτωση πολλών αναδρομικών αλγορίθμων, για να αποδείξουμε ότι έχουμε CAT αλγόριθμο, εξετάζουμε το δένδρο της αναδρομής του αλγορίθμου. Πρόκειται για ένα δένδρο με ρίζα, N κόμβους και N_0 φύλλα, όπου κάθε κόμβος-παιδί αντιστοιχεί σε μια αναδρομική κλήση, ενώ τα αντικείμενα που κατασκευάστηκαν αντιστοιχούν σε φύλλα.

Γενικά ισχύει ότι $|\mathcal{A}_n| \leq N_0$, δηλαδή κάποια φύλλα μπορεί να μην αντιστοιχούν σε κατασκευή αντικειμένου, αλλά σε αποτυχία.

Αν $N_0 = O(|\mathcal{A}_n|)$ (δηλαδή οι αποτυχίες δεν είναι πάρα πολλές, αλλά το πολύ όσες και οι επιτυχίες επί μια σταθερά) και κάθε αναδρομική κλήση εκτελεί εντολές σταθερού χρόνου (μια υπόθεση που συνήθως ισχύει) και επιπλέον δείξουμε ότι το πλήθος N των κλήσεων αυτών είναι $O(N_0)$, έχουμε δείξει την ιδιότητα CAT.

Constant Amortized Time (CAT)

Έστω δένδρο με ρίζα, στο οποίο υπάρχουν N_i κόμβοι με i παιδιά, όπου $0 \leq i \leq t$. Τότε, προφανώς ισχύουν οι σχέσεις

$$N = N_0 + N_1 + N_2 + \cdots + N_t$$

και, αφού υπάρχουν $N - 1$ ακμές,

$$N - 1 = 0N_0 + 1N_1 + 2N_2 + \cdots + tN_t.$$

Από τις δύο παραπάνω σχέσεις, προκύπτει ότι

$$N_0 - 1 = N_2 + 2N_3 + \cdots + (t-1)N_t \geq N_2 + N_3 + \cdots + N_t = N - N_0 - N_1,$$

άρα

$$N \leq 2N_0 + N_1 - 1. \tag{1}$$

Κατόπιν τούτων, αν $N_1 = O(N_0)$, τότε ο αλγόριθμος είναι CAT. Στην κατεύθυνση αυτή, είναι χρήσιμο το επόμενο λήμμα:

Λήμμα

Αν το πλήθος κόμβων σε κάθε μονοπάτι με κόμβους βαθμού 1 είναι φραγμένο από μια σταθερά $c > 0$, τότε $N_1 = O(N_0)$.

Απόδειξη.

Αφού από κάθε κόμβο βαθμού $k > 1$ ξεκινούν το πολύ k τέτοια μεγιστικά μονοπάτια, έπεται ότι

$$N_1 \leq c(2N_2 + 3N_3 + \cdots + tN_t) = c(N - 1 - N_1) \overset{(1)}{\leq} c(2N_0 - 2)$$

άρα $N_1 = O(N_0)$. □

Για την κατασκευή ενός συνόλου αντικειμένων, είναι πολλές φορές βολικό να δημιουργούμε τα αντικείμενα αυτά με μια συγκεκριμένη σειρά, σύμφωνα με μια ολική διάταξη που έχει προηγουμένως ορισθεί στο σύνολο. Αν τα αντικείμενα κωδικοποιούνται από λέξεις σε κάποιο αλφάβητο, τότε συνήθως χρησιμοποιείται η λεξικογραφική διάταξη $\leq_L$ ή απλά $\leq$:

Ορισμός (Λεξικογραφική διάταξη)

Για δύο λέξεις $a = a_1 a_2 \cdots a_n$ και $b = b_1 b_2 \cdots b_m$ ισχύει $a < b$ ανν

- για κάποιο $k \leq \min\{n, m\}$ είναι $a_k < b_k$ και $a_i = b_i$ για κάθε $i < k$ ή
- $n < m$ και $a_i = b_i$, για κάθε $i \leq n$.

Δηλαδή, πρόκειται για μια ολική διάταξη που επεκτείνει στο σύνολο των λέξεων μια ήδη ορισμένη ολική διάταξη στο αλφάβητο.

Άλλες ολικές διατάξεις που μπορεί να οδηγήσουν σε πιο απλούς ή και πιο αποδοτικούς αλγορίθμους είναι οι ακόλουθες:

Ορισμός (Αντίστροφη λεξικογραφική διάταξη $\leq_R$ - relex)

Για δύο λέξεις a, b είναι

$$a <_R b \Leftrightarrow b < a.$$

Ορισμός (Συμμετρική λεξικογραφική διάταξη $\leq_C$ - colex)

Για δύο λέξεις a, b είναι

$$a <_C b \Leftrightarrow \mathrm{rev}(a) < \mathrm{rev}(b),$$

όπου $\mathrm{rev}(a) = a_n \cdots a_2 a_1$ είναι η συμμετρική λέξη της $a = a_1 a_2 \cdots a_n$.

Έστω $\mathcal{B}_n$ το σύνολο των υποσυνόλων ενός συνόλου S, με $|S| = n$, όπου $n \in \mathbb{N}^*$. Χωρίς βλάβη της γενικότητας, μπορούμε να θεωρήσουμε ότι $S = [n] = \{1, 2, \ldots, n\}$.
Κάθε σύνολο $B \in \mathcal{B}_n$, μπορεί να αναπαρασταθεί ως μια δυαδική λέξη $b_1 b_2 \cdots b_n$, όπου για κάθε $i \in [n]$, είναι $b_i = [i \in B]$.

lex	**relex**	**colex**
000	111	000
001	110	100
010	101	010
011	100	110
100	011	001
101	010	101
110	001	011
111	000	111

Υποσύνολα - Επαναληπτικός αλγόριθμος κατασκευής

- Κάθε δυαδική λέξη $B \in \mathcal{B}_n$ (εκτός από την $11\cdots1$) έχει τη μορφή $b_1 b_2 \cdots b_{k-1}01\cdots1$, για κάποιο $k \in [n]$. Η επόμενή της λεξικογραφικά είναι η $b_1 b_2 \cdots b_{k-1}10\cdots0$.
- Βάσει αυτής της παρατήρησης, μπορούμε να ορίσουμε μια ρουτίνα $\mathrm{next}()$ που θα υπολογίζει την επόμενη της B, εντοπίζοντας την δεξιότερη θέση k με $b_k = 0$, θέτοντας $b_k = 1$ και $b_i = 0$, για κάθε $i > k$.
- Για ευκολία, θεωρούμε ένα επιπλέον ψηφίο $b_0 = 0$ για κάθε λέξη, οπότε κάθε λέξη ξεκινά με 0 και η επόμενη της τελευταίας λέξης $011\cdots1$ θα είναι η $100\cdots0$.
- Το σύνολο $\mathcal{B}_n$ κατασκευάζεται ξεκινώντας από την πρώτη λέξη $00\cdots0$ και καλώντας επαναληπτικά την ρουτίνα $\mathrm{next}()$ μέχρις ότου $b_0 = 1$.

```
1 subsetNext() begin                                  /* global variables:
   n, B = b_0 b_1 b_2 ··· b_n ,  b_i = 0 */
2    k := n;
3    while b_k = 1 do                  /* search for the right-most 0 */
4       b_k := 0;                  /* zero all trailing ones to minimize
         suffix */
5       k := k − 1;
6    b_k := 1;
7    return (k ≠ 0);                /* return false when last object is
      generated */
```

Αλγόριθμος 4: Λεξικογραφική κατασκευή υποσυνόλων B του $[n]$ σε δυαδική αναπαράσταση.

Για την ανάλυση του αλγορίθμου, αρκεί να υπολογίσουμε το πλήθος των ελέγχων ($b_k = 1$) που εκτελούνται στη γραμμή 3 (while). Ο έλεγχος προφανώς αποτυγχάνει μια φορά για κάθε B, δηλαδή συνολικά 2^n φορές. Από την άλλη, είναι επιτυχής στην θέση i κάθε φορά που το B είναι της μορφής $b_1 b_2 \cdots b_{i-1} 11 \cdots 1$. Επειδή υπάρχουν 2^{i-1} λέξεις αυτής της μορφής, έπεται ότι συνολικά (σε όλες τις θέσεις) επιτυγχάνει $\sum_{i=1}^{n} 2^{i-1} = 2^n - 1$ φορές. Επομένως, ο συνολικός αριθμός εκτελέσεων του ελέγχου είναι $2^{n+1} - 1$ και διαιρώντας με το $|\mathcal{B}_n| = 2^n$, βρίσκουμε ότι ο μέσος αριθμός εκτελέσεων είναι $2 - 2^{-n} = O(1)$. Επομένως, ο αλγόριθμος είναι CAT.

Υποσύνολα - Αναδρομικός αλγόριθμος κατασκευής

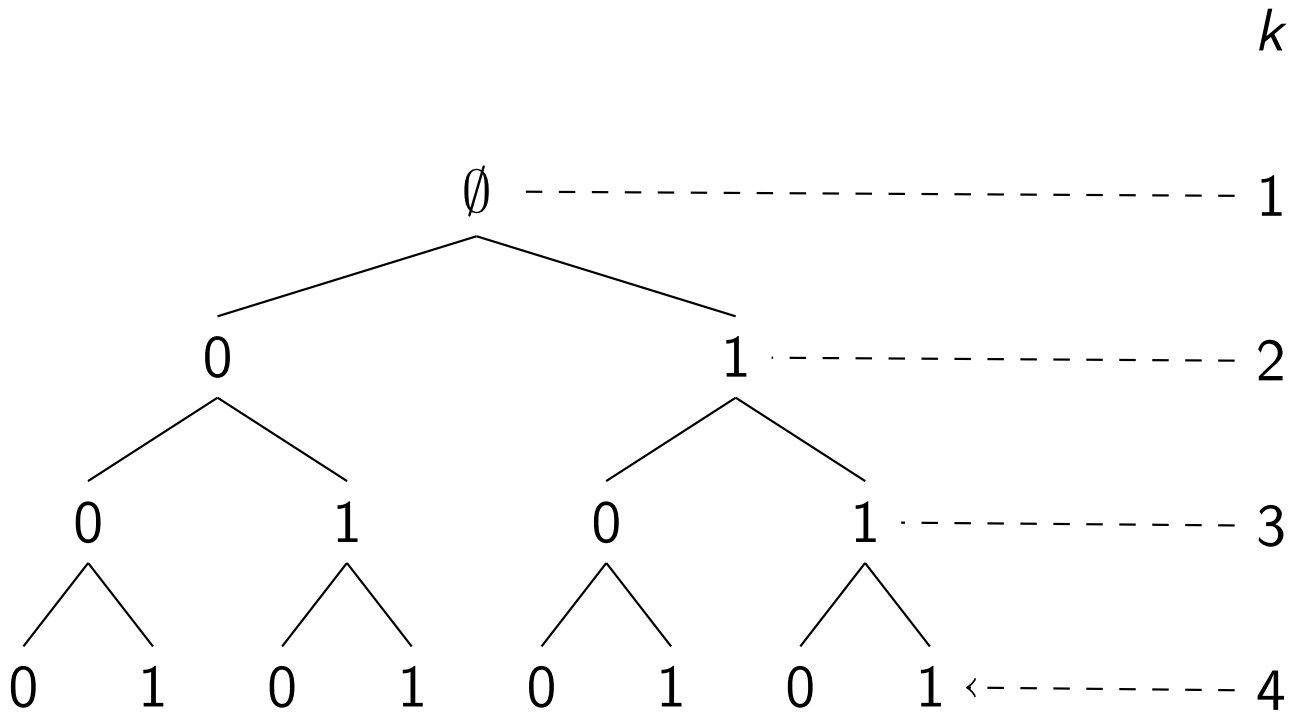
Για την αναδρομική κατασκευή των στοιχείων του $\mathcal{B}_n$ σε λεξικογραφική διάταξη, κατασκευάζουμε πρώτα τις λέξεις $b_1 b_2 \cdots b_n$ με $b_1 = 0$ και έπειτα αυτές με $b_1 = 1$. Σε κάθε περίπτωση, οι υπολέξεις $b_2 \cdots b_n \in \mathcal{B}_{n-1}$ κατασκευάζονται αναδρομικά.

```
1 subsetRec(k) begin
2     if k > n then output (B);
3     else
4         b_k := 0;
5         subsetRec(k + 1);
6         b_k := 1;
7         subsetRec(k + 1);
```

Αλγόριθμος 5: Αναδρομική λεξικογραφική κατασκευή όλων των υποσυνόλων B του $[n]$ σε δυαδική αναπαράσταση.

Υποσύνολα - Αναδρομικός αλγόριθμος κατασκευής

Παρακάτω φαίνεται το δένδρο της αναδρομής T με αρχική κλήση subsetRec(1), η οποία αντιστοιχεί στη ρίζα του δένδρου, και για $n = 3$. Κάθε υποσύνολο αντιστοιχεί σε ένα μονοπάτι από τη ρίζα μέχρι κάποιο φύλλο και η σειρά κατασκευής τους ακολουθεί την προδιάταξη του δένδρου.



Για τον χρόνο εκτέλεσης, αρκεί να μετρήσουμε το πλήθος των αναδρομικών κλήσεων, αφού κάθε μία εκτελεί εντολές σταθερού χρόνου (γραμμές 2, 4, 6).
Το δένδρο της αναδρομής είναι προφανώς ένα πλήρες δυαδικό δένδρο με $n+1$ επίπεδα, $N = \sum_{k=1}^{n+1} 2^{k-1} = 2^{n+1} - 1$ κόμβους, που αντιστοιχούν στις αναδρομικές κλήσεις, και $N_0 = 2^n$ φύλλα, που αντιστοιχούν στα αντικείμενα που παράχθηκαν. Επομένως, είναι
$N/N_0 = 2 - 2^{-n} = O(1)$, δηλαδή ο αλγόριθμος είναι CAT.

Παρατηρήστε ότι τα στοιχεία προκύπτουν σε διάταξη relex, αν διασχίσουμε σε προδιάταξη το συμμετρικό δένδρο του T (όπου κάθε αριστερό παιδί γίνεται δεξί και αντίστροφα).
Επιπλέον, τα στοιχεία προκύπτουν σε διάταξη colex, αν κάθε στοιχείο αντιστοιχιστεί σε ένα μονοπάτι από κάποιο φύλλο προς τη ρίζα.

Άσκηση

Να τροποποιηθεί η συνάρτηση subsetRec, ώστε να κατασκευάζει τα στοιχεία του $\mathcal{B}_n$ σύμφωνα με τη διάταξη

- relex
- colex

Γενικά, η συνάρτηση rank είναι μια αμφιμονοσήμαντη (ένα προς ένα και επί) συνάρτηση $\mathrm{rank}_{\tau,n} : \mathcal{A}_n \to \{0, 1, \ldots, |\mathcal{A}_n| - 1\}$, η οποία αντιστοιχίζει τα στοιχεία του $\mathcal{A}_n$ σε φυσικούς αριθμούς σύμφωνα με μια προκαθορισμένη ολική διάταξη $\leq_\tau$ στο $\mathcal{A}_n$, έτσι ώστε

$$\mathrm{rank}_{\tau,n}(a) = |\{b \in \mathcal{A}_n : b <_\tau a\}|, \qquad \forall a \in \mathcal{A}_n,$$

δηλαδή το rank του αντικειμένου $a \in \mathcal{A}_n$ ισούται με το πλήθος των αντικειμένων στο σύνολο $\mathcal{A}_n$ που είναι μικρότερα (προηγούνται) του a. Οι δείκτες τ, n παραλείπονται όταν δεν υπάρχει κίνδυνος σύγχυσης. Η συνάρτηση unrank είναι η αντίστροφη της $\mathrm{rank}_{\tau,n}$. Δέχεται ως είσοδο έναν φυσικό αριθμό $0 \leq r \leq |\mathcal{A}_n| - 1$ και επιστρέφει το αντικείμενο a με $\mathrm{rank}_{\tau,n}(a) = r$.

Για το τυχαίο στοιχείο $B = b_1 b_2 \cdots b_n \in \mathcal{B}_n$, είναι

$$\mathrm{rank}(B) = \sum_{i=1}^{n} 2^{n-i}[b_i = 1],$$

οπότε το $\mathrm{rank}(B)$ μπορεί να υπολογισθεί με τον ακόλουθο αλγόριθμο:

```
1 subsetRank(n, B) begin
2     r := 0;
3     for i := 1 to n do
4         r := 2r;
5         if b_i = 1 then r := r + 1;
6     return r;
```

Αλγόριθμος 6: Αλγόριθμος ranking για το $B = b_1 \cdots b_n \in \mathcal{B}_n$.

Η γραμμή 5 προσθέτει μια μονάδα στο αποτέλεσμα, η οποία στη συνέχεια της εκτέλεσης πολλαπλασιάζεται συνολικά $n - i$ φορές με το 2, δίνοντας έτσι τον όρο $2^{n-i}[b_i = 1]$ του αθροίσματος.

Ο αλγόριθμος unranking είναι ο ακόλουθος:

```
1 subsetUnrank(n, r) begin
2     B := ∅;
3     for i := n down to 1 do
4         if 1 ≡ r (mod 2) then b_i := 1;
5         r := ⌊r/2⌋;
6     return B;
```

Αλγόριθμος 7: Αλγόριθμος unranking για το $\mathcal{B}_n$.

Ο αλγόριθμος αυτός ουσιαστικά μετατρέπει τον φυσικό r στη δυαδική του αναπαράσταση, η οποία ταυτίζεται με το υποσύνολο B.

- Να κατασκευαστεί το σύνολο των διατεταγμένων t-άδων του συνόλου $A_1 \times A_2 \times \cdots \times A_t$, όπου $A_i = \{0, 1, \ldots, n_i\}$, $i \in [t]$, για δοσμένους ακεραίους n_i.
- Να δειχθεί ότι το σύνολο των ακολουθιών $a_1 a_2 \cdots a_n$, με

$$a_i \geq -m + 1, \qquad \sum_{i=1}^{n} a_i = n, \qquad a_k > 0 \Rightarrow \sum_{i=1}^{k} a_i = k$$

είναι ισοδύναμο με το σύνολο των $(m+1)$-αδικών λέξεων μήκους $n-1$ και στη συνέχεια να κατασκευαστούν τα στοιχεία του.

k-υποσύνολα

Ένα υποσύνολο B του $S = [n]$, τέτοιο ώστε $|B| = k$, όπου $0 \leq k \leq n$, ονομάζεται k-υποσύνολο του S ή απλά k-συνδυασμός. Το σύνολο αυτών θα συμβολίζεται με $\mathcal{B}_{n,k}$. Ως γνωστό, είναι $|\mathcal{B}_{n,k}| = \binom{n}{k}$. Όπως και πριν, το B μπορεί να κωδικοποιηθεί από μια δυαδική λέξη με ακριβώς k μονάδες.
Εναλλακτικά, αν a_j είναι το j-οστό στοιχείο του B σε αύξουσα σειρά, τότε το B κωδικοποιείται από μια γνησίως αύξουσα ακολουθία θετικών ακεραίων

$$a = (a_1, \ldots, a_k), \quad \text{με } 1 \leq a_1 < a_2 < \cdots < a_k \leq n.$$

Το σύνολο αυτών θα το συμβολίζουμε με $A_{n,k}$.

Στον παρακάτω πίνακα φαίνονται οι διατάξεις lex και colex των στοιχείων του $A_{5,3}$ και τα αντίστοιχα στοιχεία του $\mathcal{B}_{5,3}$. Παρατηρήστε ότι οι διατάξεις colex ταυτίζονται στα δύο σύνολα.

lex	**relex**	**colex**	**colex**
123	11100	123	11100
124	11010	124	11010
125	11001	134	10110
134	10110	234	01110
135	10101	125	11001
145	10011	135	10101
234	01110	235	01101
235	01101	145	10011
245	01011	245	01011
345	00111	345	00111

Η βασική παρατήρηση για την κατασκευή του $A_{n,k}$ είναι η εξής: Αν το j-οστό στοιχείο έχει την τιμή a_j, τότε θα πρέπει το σύνολο $\{a_j + 1, \ldots, n\}$ να περιέχει τουλάχιστον $k - j$ στοιχεία ώστε να πάρουν τιμή τα επόμενα στοιχεία του j-οστού. Με άλλα λόγια, θα πρέπει για κάθε $j \in [k]$ να ισχύει $a_j \leq n - k + j$.

```
1 ksubsetNext() begin        /* assumes 0 < k ≤ n, a_0 < 0.   */
2     j := k;
3     while a_j = n − k + j do j := j − 1;
4     a_j := a_j + 1;
5     for i := j + 1 to k do a_i := a_{i−1} + 1;
6     return (j ≠ 0);        /* return false when last object is
       generated */
```

Αλγόριθμος 8: Λεξικογραφική κατασκευή όλων των ακολουθιών του $A_{n,k}$.

Για το χρόνο εκτέλεσης, αρκεί να μετρήσουμε το πλήθος ελέγχων $a_j = n - k + j$ της γραμμής 3 (οι έλεγχοι της γραμμής 5 έχουν ακριβώς ίδιο πλήθος). Ο έλεγχος αποτυγχάνει μια φορά για κάθε ακολουθία και πετυχαίνει στη θέση $j + 1$, όπου $0 \leq j \leq k - 1$, για $\binom{n-k+j}{j}$ ακολουθίες με $a_{j+1} = n - k + j + 1$. Επομένως, ο συνολικός αριθμός ελέγχων είναι

$$\binom{n}{k} + \sum_{j=0}^{k-1} \binom{n-k+j}{j} = \sum_{j=0}^{k} \binom{n-k+j}{j} = \binom{n+1}{k}$$

Διαιρώντας με το πλήθος $\binom{n}{k}$, προκύπτει ότι το μέσο πλήθος συγκρίσεων ανά αντικείμενο είναι

$$\frac{n+1}{n+1-k} \leq c \Leftrightarrow k \leq \frac{c-1}{c}(n+1),$$

άρα ο αλγόριθμος είναι CAT, όταν $k \leq n/2$.
Αν $k > n/2$, είναι προτιμότερο να κατασκευάσουμε τους συμπληρωματικούς $(n - k)$-συνδυασμούς.

Ο επόμενος αλγόριθμος βελτιώνει τον 8 κατά μια σταθερά, καταργώντας το βρόχο while βάσει της παρατήρησης ότι ο μέγιστος δείκτης j για τον οποίο το a_j μπορεί να αυξηθεί είναι k όταν $a_k < n$, αλλιώς είναι κατά 1 μικρότερος από τον δείκτη της τελευταίας αλλαγής.

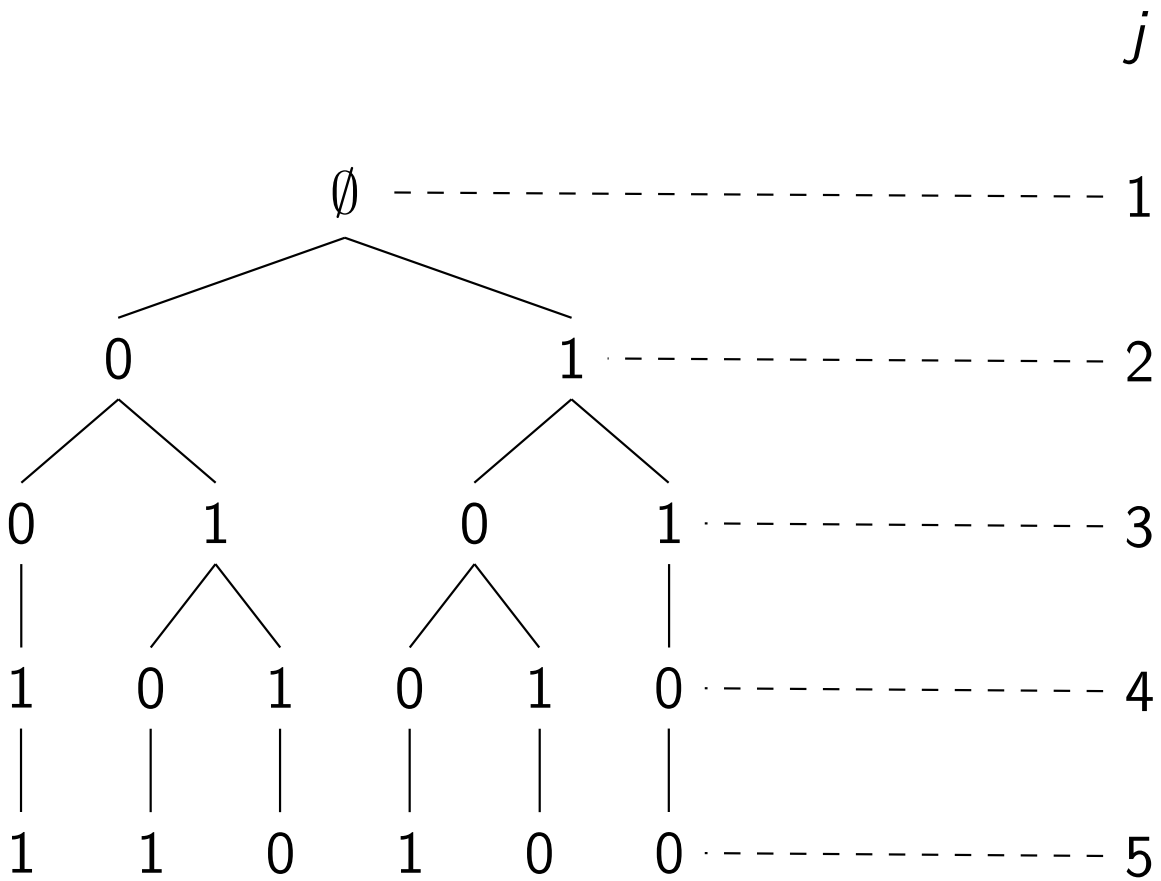
```
1 ksubsetNextB() begin /* assumes 0 < k ≤ n, a_0 < 0, j = k is
   global. */
2     a_j := a_j + 1;
3     for i := j + 1 to k do a_i := a_{i−1} + 1;
4     if j = 0 then return false;
5     if a_k < n then j := k else j := j − 1;
6     return true;
```

Αλγόριθμος 9: Βελτιωμένος αλγόριθμος λεξικογραφικής κατασκευής όλων των ακολουθιών του $A_{n,k}$.

```
1  ksubsetRec(j, m) begin   /* initial call:  ksubsetRec(1, 0)
   */
2      if j > n then output(B);
3      else /* m is the number of ones that have been placed
        so far */
4          if k − m < n − j + 1 then
5              b_j := 0;
6              ksubsetRec(j + 1, m)
7          if m < k then
8              b_j := 1;
9              ksubsetRec(j + 1, m + 1)
```

Αλγόριθμος 10: Αναδρομικός αλγόριθμος λεξικογραφικής κατασκευής όλων των στοιχείων του $\mathcal{B}_{n,k}$ σε δυαδική αναπαράσταση.

Παρακάτω φαίνεται το δένδρο της αναδρομής T με αρχική κλήση ksubsetRec(1,0) και για $(n, k) = (4, 2)$.



Ο επόμενος αλγόριθμος (colex) δεν χρησιμοποιεί τη βοηθητική μεταβλητή m, έχει το ίδιο δένδρο αναδρομής με τον προηγούμενο, αλλά τώρα τα στοιχεία αντιστοιχούν σε μονοπάτια από τα φύλλα προς τη ρίζα.

```
1 ksubsetColexRec(n, k) begin
2     if n = 0 then output (B);
3     else
4         if k < n then
5             b_n := 0;
6             ksubsetColexRec(n − 1, k)
7         if k > 0 then
8             b_n := 1;
9             ksubsetColexRec(n − 1, k − 1)
```

Αλγόριθμος 11: Αναδρομικός (colex) αλγόριθμος κατασκευής όλων των στοιχείων του $\mathcal{B}_{n,k}$ σε δυαδική αναπαράσταση.

Ο αλγόριθμος 11 παράγει τους k-συνδυασμούς σε διάταξη colex, αλλά είναι πιο φυσικός από τον προηγούμενο. Το πλήθος $t(n,k)$ των αναδρομικών κλήσεων ικανοποιεί την αναδρομική σχέση

$$t(n,k) = \begin{cases} 1, & n = 0 \\ 1 + t(n-1,0), & k = 0 \\ 1 + t(n-1,n-1), & k = n \\ 1 + t(n-1,k) + t(n-1,k-1), & 0 < k < n \end{cases}$$

η οποία με επαγωγή δίνει $t(n,k) = \binom{n+2}{k+1} - 1$. Επομένως

$$(t(n,k)+1)/\binom{n}{k} = \frac{(n+2)(n+1)}{(k+1)(n-k+1)}$$

δηλαδή ο αλγόριθμος δεν είναι CAT όταν το k είναι κοντά στο 0 ή στο n.

Αυτό συμβαίνει διότι στο δένδρο της αναδρομής υπάρχουν πολλοί κόμβοι βαθμού 1. Όμως, οι απόγονοι αυτών έχουν πάντα βαθμό 1 και την ίδια ετικέτα μεταξύ τους (γιατί;). Το πλήθος των κόμβων βαθμού 1 με $i \geq 1$ απογόνους με ετικέτα 0 είναι

$$|\{B \in \mathcal{B}_{n,k} : B = 0 \cdots 0 b_{i+1} \cdots b_n\}| = \binom{n-i}{k}$$

και

$$\sum_{i \geq 1} \binom{n-i}{k} = \binom{n}{k+1},$$

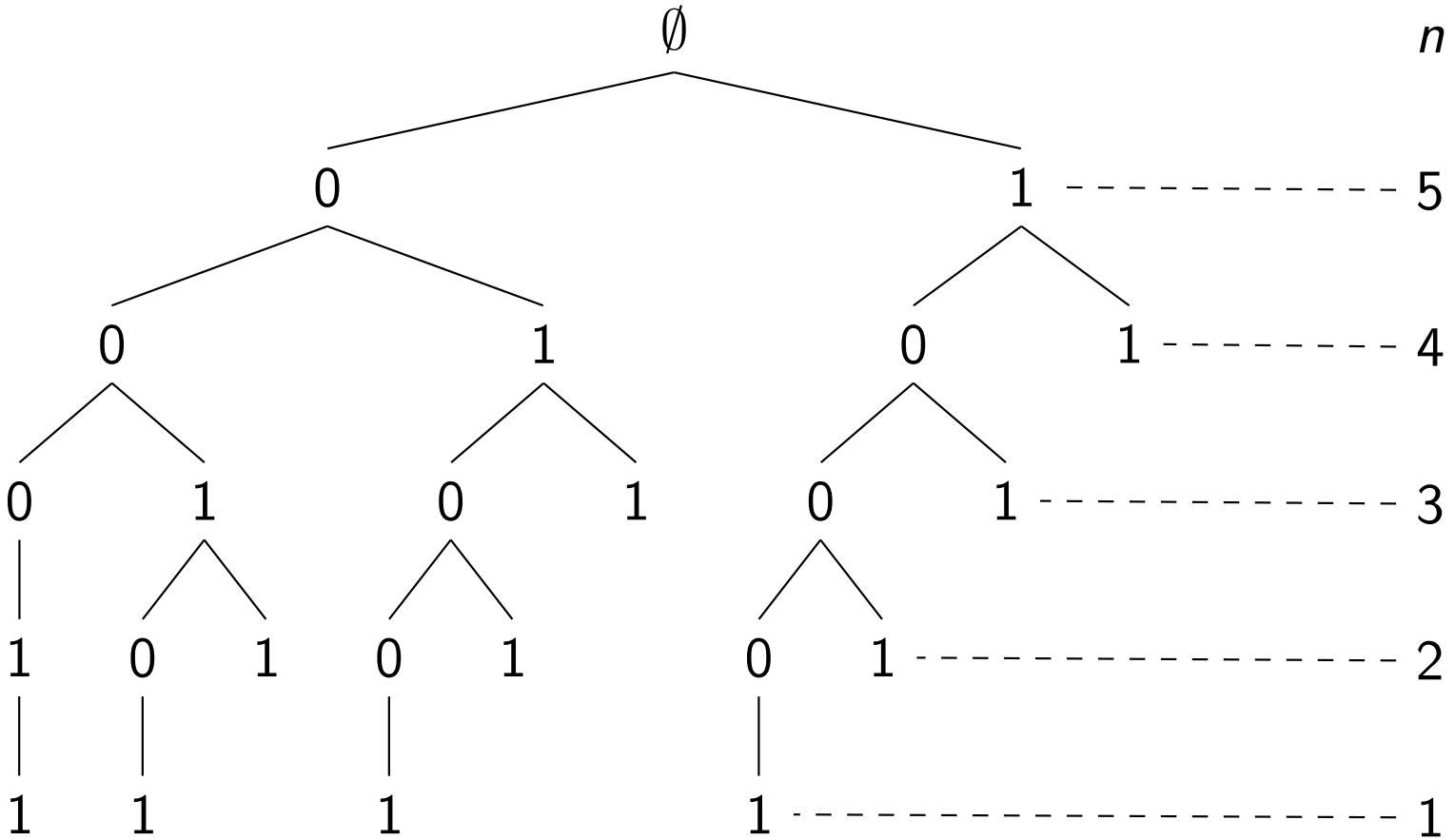
επομένως αν τους αφαιρέσουμε, το πλήθος των αναδρομικών κλήσεων θα γίνει $\binom{n+2}{k+1} - 1 - \binom{n}{k+1} = \binom{n+1}{k} + \binom{n}{k} - 1$ και ο αλγόριθμος θα είναι CAT για $k \leq n/2$, όπως και ο 8. Αυτό μπορούμε να το πετύχουμε σταματώντας την αναδρομή όταν $k = 0$.

```
1  ksubsetColexRecB(n, k) begin
2      if k = 0 then output (B);
3      else
4          if k < n then
5              b_n := 0;
6              ksubsetColexRecB(n − 1, k)
7          if k > 0 then
8              b_n := 1;
9              ksubsetColexRecB(n − 1, k − 1);
10             b_n := 0
```

Αλγόριθμος 12: Αναδρομικός CAT αλγόριθμος κατασκευής του $\mathcal{B}_{n,k}$ για $k \leq n/2$.

- (Άσκηση.) Αν διακόψουμε την αναδρομή όταν $k = n$ (οπότε τα υπόλοιπα γράμματα της λέξης θα πρέπει να συμπληρωθούν με 1), τότε θα εξαλείψουμε τους κόμβους με απογόνους με ετικέτα 1 και ο αλγόριθμος θα είναι CAT όταν $k > n/2$.
- Η εντολή στη γραμμή 10 είναι απαραίτητη, διότι όταν σταματάει η αναδρομή σε κάποια θέση i επειδή $k = 0$, οι θέσεις $b_1, \ldots, b_{i-1}$ της B που θα τυπωθεί θα πρέπει να περιέχουν 0. Όμως, ενδέχεται η θέσεις αυτές να συμπληρώθηκαν με 1 από προηγούμενες κλήσεις-λέξεις. Για το λόγο αυτό, οι τελευταίες θα πρέπει να αποκαταστήσουν την τιμή 0, αφού ολοκληρώσουν την εκτέλεσή τους.
- Η εντολή στη γραμμή 5 δεν είναι απαραίτητη. Όμως, αν αφαιρεθεί, τότε απαιτείται οπωσδήποτε αρχικοποίηση της B.

Παρακάτω φαίνεται το δένδρο της αναδρομής του αλγορίθμου 12 με αρχική κλήση ksubsetColexRecB(5,2), για $(n, k) = (5, 2)$.



```
1 ksubsetColexRecC(n, k) begin
2     if k = n then output(A);
3     else
4         ksubsetColexRecC(n − 1, k);
5         if k > 0 then
6             a_k := n;
7             ksubsetColexRecC(n − 1, k − 1);
8             a_k := k
```

Αλγόριθμος 13: Αναδρομικός CAT αλγόριθμος κατασκευής του $A_{n,k}$ για $k \geq n/2$.

- Η ιδέα της εξάλειψης μονοπατιών με κόμβους βαθμού 1 (path elimination) μπορεί να εφαρμοσθεί σε οποιονδήποτε αναδρομικό αλγόριθμο, ώστε να φραχθεί ο λόγος N/N_0 των κόμβων προς τα φύλλα και ο αλγόριθμος να γίνει CAT. Αυτό επιτυγχάνεται σταματώντας την αναδρομή όταν οι παράμετροι (εδώ τα n, k) φτάσουν σε κάποια οριακή τιμή.
- Η διακοπή της αναδρομής μπορεί να γίνει ακόμα νωρίτερα και τα αντικείμενα που αντιστοιχούν σε αυτό το υποδένδρο να παραχθούν επαναληπτικά. Για παράδειγμα, στον αλγόριθμο 12, μπορούμε να διακόψουμε την αναδρομή όταν $k = 1$ ή $k = n - 1$, και να συμπληρώσουμε επαναληπτικά τα αντίστοιχα εναπομείναντα προθέματα

$$\{0^{j-1}10^{n-j} : j \in [n]\} \quad \text{και} \quad \{1^{n-j}01^{j-1} : j \in [n]\}.$$

Η επόμενη άσκηση ζητά την υλοποίηση αυτής της μετατροπής και την απόδειξη ότι ο αλγόριθμος είναι CAT.

Άσκηση

- Να τροποποιήσετε τον αλγόριθμο 12 σύμφωνα με την προηγούμενη παρατήρηση και στη συνέχεια να συγκρίνετε τους χρόνους εκτέλεσης των 2 αλγορίθμων.
- Να δείξετε ότι κάθε δυαδικό δένδρο με N_0 φύλλα, στο οποίο κάθε εσωτερικός κόμβος (όχι φύλλο) έχει 2 παιδιά, έχει συνολικά $N = 2N_0 - 1$ κόμβους (εσωτερικούς και φύλλα).
  Με τη βοήθεια αυτού του αποτελέσματος, δείξτε ότι ο τροποποιημένος αλγόριθμος είναι CAT.

k-υποσύνολα - Τυχαία κατασκευή

Η τυχαία επιλογή ενός αντικειμένου ισοδυναμεί με την τυχαία επιλογή ενός μονοπατιού από τη ρίζα προς κάποιο φύλλο στο δένδρο της αναδρομικής κατασκευής. Λαμβάνοντας υπόψη π.χ. το δένδρο του αλγορίθμου 12, ο τυχαίος κόμβος v αντιστοιχεί σε μια κλήση με ορίσματα n, k, κατά την οποία απομένουν n στοιχεία από τα οποία πρέπει να επιλεχθούν τα k. Τα φύλλα του υποδένδρου με ρίζα v είναι σε πλήθος $\binom{n}{k}$, τα φύλλα του αριστερού παιδιού του v είναι $\binom{n-1}{k}$, ενώ του δεξιού είναι $\binom{n-1}{k-1}$. Επομένως, το τυχαίο μονοπάτι διακλαδίζεται δεξιά με πιθανότητα $\binom{n-1}{k-1}/\binom{n}{k} = k/n$ ή αριστερά με πιθανότητα $\binom{n-1}{k}/\binom{n}{k} = 1 - k/n$. Η δεξιά διακλάδωση ισοδυναμεί με την προσθήκη του στοιχείου i στο τυχαίο k-υποσύνολο, όπου i το επίπεδο του v (η ρίζα είναι στο επίπεδο 1), ή ισοδύναμα, με την ανάθεση $b_i = 1$ στη δυαδική λέξη $B = b_1 \cdots b_n$ που κωδικοποιεί το τυχαίο k-υποσύνολο.

k-υποσύνολα - Τυχαία κατασκευή

Βάσει αυτών των παρατηρήσεων, προκύπτει ο ακόλουθος αλγόριθμος κατασκευής τυχαίας ακολουθίας $a \in A_{n,k}$ σε χρόνο $O(n)$:

```
1  ksubsetRand(n, k) begin                          /* assumes 0 ≤ k ≤ n */
2      m := 1;
3      i := 1;
4      while k > 0 do
5          U = rand(0, 1);                  /* choose random U ∈ [0, 1) */
6          if nU < k then                                /* include m */
7              a_i := m;
8              k := k − 1;
9              i := i + 1;
10         m := m + 1;
11         n := n − 1;
12     return a;
```

Αλγόριθμος 14: Κατασκευή τυχαίας ακολουθίας του $A_{n,k}$.

Εφαρμογή

Να διατυπωθεί αλγόριθμος κατασκευής ενός τυχαίου μη κατευθυνόμενου γραφήματος $G = (V, E)$, με $V = \{1, 2, \ldots, n\}$ και $|E| = k$.
Υπόδειξη: Ως γνωστό, είναι $0 \leq k \leq \binom{n}{2}$.

k-υποσύνολα - Τυχαία κατασκευή

Ακόμα και αν το μέγεθος n του συνόλου δεν είναι εκ των προτέρων γνωστό (π.χ. κατά την ανάγνωση των εγγραφών ενός αρχείου), είναι δυνατό να κατασκευάσουμε ένα τυχαίο δείγμα μεγέθους k με μία μόνο σάρωση των στοιχείων. Αρκεί να εξασφαλίσουμε ότι όταν θα έχουμε σαρώσει τα πρώτα $t \geq k$ στοιχεία, θα έχουμε ένα τυχαίο δείγμα μεγέθους k από αυτά.
Για να το πετύχουμε αυτό, κρατάμε μια λίστα I με k στοιχεία, η οποία αρχικά αποτελείται από (δείκτες προς) τα k πρώτα στοιχεία του συνόλου. Σαρώνουμε τα υπόλοιπα στοιχεία συνόλου και όταν διαβάζουμε το t-οστό στοιχείο, τότε το προσθέτουμε στη λίστα I με πιθανότητα k/t, αντικαθιστώντας ένα τυχαίο στοιχείο της λίστας. Έτσι, η νέα λίστα αποτελεί ένα τυχαίο δείγμα μεγέθους k από τα πρώτα t στοιχεία. Η διαδικασία αυτή επαναλαμβάνεται για κάθε t με $k < t \leq n$, δηλαδή με μία μόνο σάρωση όλων των δεδομένων.

k-υποσύνολα - Τυχαία κατασκευή

```
1 ReservoirSampling(k, data) begin    /* assumes data records
   are numbered 1, 2, 3, ... */
2   for t := 1 to k do l_t := t;    /* t is the number of current
     record in data  */
3   while not end of file data do
4     choose random integer U in [t];
5     if U ≤ k then                                  /* include t */
6       l_U := t;
7       t := t + 1;
8   return l;
```

Αλγόριθμος 15: Κατασκευή τυχαίου δείγματος μεγέθους k από αρχείο αγνώστου πλήθους εγγραφών.

k-υποσύνολα - Τυχαία κατασκευή

Η μέθοδος αυτή είναι γνωστή ως reservoir sampling και μπορεί να χρησιμοποιηθεί ακόμα και για την κατασκευή τυχαίου δείγματος λαμβάνοντας υπόψη πολλά ανεξάρτητα χαρακτηριστικά. Για παράδειγμα, αν έχουμε ένα μεγάλο αρχείο με κατοίκους 10 πόλεων, μπορούμε με μία μόνο σάρωση να κατασκευάσουμε ένα τυχαίο δείγμα που θα περιέχει ακριβώς 100 κατοίκους από κάθε πόλη.
Όταν τα δεδομένα αποτελούν ένα μεγάλο αρχείο, συνήθως χρησιμοποιείται ένα βοηθητικό αρχείο R (reservoir), στο οποίο εισέρχονται όσες εγγραφές επιλέγονται με πιθανότητα k/t (καθώς και οι k πρώτες). Οι δείκτες της λίστας I δείχνουν στις εγγραφές του R. Εύκολα αποδεικνύεται (άσκηση) ότι το μέσο μέγεθος του R μετά το τέλος του αλγορίθμου ισούται με

$$k(1 + H_n - H_k), \qquad \text{όπου } H_n := \sum_{i=1}^{n} \frac{1}{i},$$

οπότε η ανάγνωση των εγγραφών με δείκτες στο I στο τέλος του αλγορίθμου γίνεται πολύ πιο γρήγορα από ότι αν οι δείκτες έδειχναν στο αρχικό αρχείο.

Προφανώς, $\mathrm{rank}_{L,n}(a_1) = a_1 - 1$ και θέτοντας $a_0 = 0$, έχουμε ότι

$$\begin{aligned}
&\mathrm{rank}_{L,n}(a_1 a_2 \cdots a_k) \\
=&|\{a_1 b \in A_{n,k} : b < a_2 \cdots a_k\}| + |\{ib : i < a_1, b \in i + A_{n-i,k-1}\}| \\
=&\,\mathrm{rank}_{L,n-a_1}((a_2 - a_1)(a_3 - a_1)\cdots(a_k - a_1)) + \sum_{i=1}^{a_1-1} \binom{n-i}{k-1} = \cdots \\
=&\sum_{j=0}^{k-1} \sum_{i=1}^{a_{j+1}-a_j-1} \binom{n-a_j-i}{k-j-1} = \sum_{j=0}^{k-1} \left(\binom{n-a_j}{k-j} - \binom{n-a_{j+1}+1}{k-j} \right) \\
=&\binom{n}{k} - 1 - \sum_{j=1}^{k} \binom{n-a_j}{k-j+1},
\end{aligned}$$

όπου χρησιμοποιήθηκε η ταυτότητα $\displaystyle\sum_{j=0}^{n} \binom{j}{m} = \binom{n+1}{m+1}$.

Επομένως, $\displaystyle\mathrm{rank}_{R,n}(a_1 a_2 \cdots a_k) = \sum_{j=1}^{k} \binom{n-a_j}{k-j+1}$.

Σε διάταξη colex, ο υπολογισμός είναι πολύ πιο εύκολος:

$$\mathrm{rank}_{C,n}(a_1 a_2 \cdots a_k) = |\{x = x_1 \cdots x_k \in A_{n,k} : x <_C a\}|$$

$$= |\biguplus_{j=1}^{k} \{x_1 \cdots x_{j-1} x_j a_{j+1} \cdots a_k \in A_{n,k} : x_j < a_j\}| = \sum_{j=1}^{k} \binom{a_j - 1}{j}.$$

Κατόπιν τούτων, το $\mathrm{rank}_L(a)$ μπορεί να υπολογισθεί βάσει του επόμενου τύπου:

Πρόταση

Αν $a = a_1 \cdots a_k \in A_{n,k}$ *και* $a' = a'_1 \cdots a'_k$, *όπου* $a'_i = n + 1 - a_i$, *για κάθε* $i \in [k]$, *τότε* $\mathrm{rev}(a') \in A_{n,k}$ *και*

$$\mathrm{rank}_{L,n}(a) + \mathrm{rank}_{C,n}(\mathrm{rev}(a')) = \binom{n}{k} - 1.$$

Απόδειξη.

Άσκηση. □

```
1 ksubsetColexUnrank(r) begin
2     for j = k down to 1 do
3         p := j;
4         while (p choose j) ≤ r do p := p + 1;
5         r := r − (p−1 choose j);
6         a_j := p;
7     return a;
```

Αλγόριθμος 16: Αλγόριθμος unrank για το $A_{n,k}$ σε διάταξη colex.

Οι προηγούμενοι αλγόριθμοι ranking-unranking χρησιμοποιούν τις τιμές των διωνυμικών συντελεστών, οι οποίες μπορούν να υπολογισθούν εξ αρχής και να αποθηκευθούν σε πίνακα, οπότε οι χρόνοι εκτέλεσης θα είναι $O(n)$.

Επαναληπτικοί συνδυασμοί

Οι προηγούμενοι αλγόριθμοι μπορούν να χρησιμοποιηθούν και για την κατασκευή άλλων αντικειμένων που είναι συναφή με τους συνδυασμούς:

- Ένας επαναληπτικός συνδυασμός n στοιχείων ανά k είναι μια αύξουσα ακολουθία $a^* = (a_1^*, a_2^*, \ldots, a_k^*)$ k ακεραίων στο $[n]$, δηλαδή $1 \leq a_1^* \leq a_2^* \leq \cdots \leq a_k^* \leq n$.
  Το σύνολο αυτών θα το συμβολίζουμε με $A_{n,k}^*$.
  Η σχέση τους με τους συνδυασμούς είναι άμεση: Θέτοντας, για κάθε $j \in [k]$, $a_j = a_j^* + j - 1$, προκύπτει η ακολουθία $a = (a_1, a_2, \ldots, a_k) \in A_{n+k-1,k}$. Αντίστροφα, αν δίνεται μια $a \in A_{n+k-1,k}$, μέσω της σχέσης $a_j^* = a_j - (j-1)$, προκύπτει η $a^* \in A_{n,k}^*$, επομένως

$$|A_{n,k}^*| = \binom{n+k-1}{k}$$

Επαναληπτικοί συνδυασμοί

- Αν συμβολίσουμε με x_i το πλήθος των i στην ακολουθία $a^* \in A^*_{n,k}$, τότε

$$x_1 + x_2 + \cdots + x_n = k, \qquad x_i \in \mathbb{N} \tag{*}$$

δηλαδή το σύνολο των λύσεων της (*) απεικονίζεται αμφιμονοσήμαντα στο $A^*_{n,k}$, οπότε το πλήθος αυτών ισούται με $\binom{n+k-1}{k}$.
Επομένως, η a^* μπορεί να κωδικοποιηθεί από τη δυαδική λέξη

$$1^{x_1}01^{x_2}0\cdots 1^{x_{n-1}}01^{x_n}.$$

Οι διαδοχικές μονάδες αντιστοιχούν στον αριθμό επαναλήψεων κάθε στοιχείου i και τα $n-1$ μηδενικά παρεμβάλλονται για να προσδιορίζουν σε ποιο i αντιστοιχεί κάθε μπλοκ από μονάδες. Το πλήθος αυτών των δυαδικών λέξεων είναι προφανώς ίσο με $\binom{n+k-1}{k}$.

Διασπάσεις ακεραίου

- Οι λύσεις της

$$x_1 + x_2 + \cdots + x_n = k, \qquad x_i \in \mathbb{N}^* \qquad (**)$$

ονομάζονται n-διασπάσεις (compositions) του k.
Θέτοντας $y_i = x_i - 1$, παίρνουμε την ισοδύναμη εξίσωση

$$y_1 + y_2 + \cdots + y_n = k - n, \qquad y_i \in \mathbb{N}$$

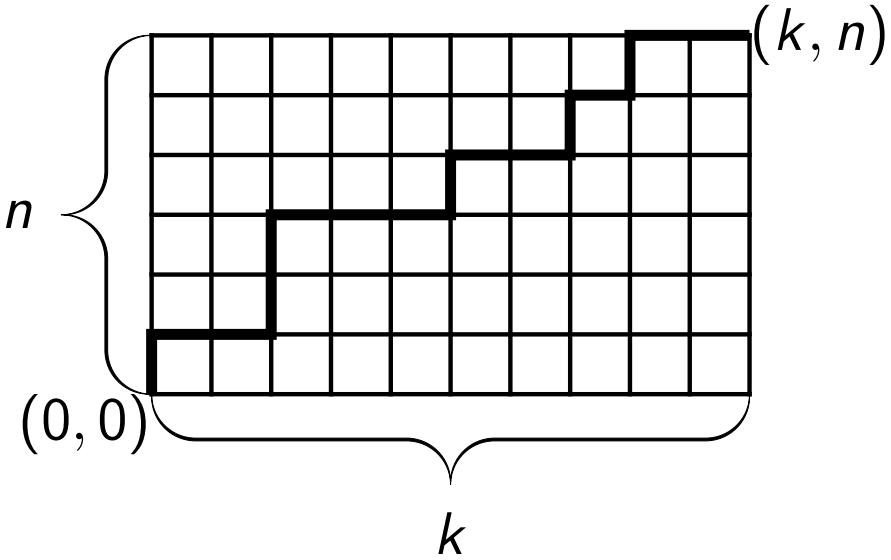
δηλαδή το πλήθος των λύσεων της (**) ισούται με $\binom{n+k-n-1}{k-n} = \binom{k-1}{n-1}$.

Εφαρμογή

Να βρεθεί το πλήθος των δυαδικών λέξεων που αποτελούνται από k μονάδες και n μηδενικά, οι οποίες περιέχουν ακριβώς m μεγιστικές υπολέξεις (runs) από διαδοχικές μονάδες (π.χ. η λέξη 10110011100 περιέχει 3 τέτοιες υπολέξεις).

Συνδυασμοί και μονοπάτια

- Το πλήθος των μονοπατιών από το σημείο $(0,0)$ στο (k,n) που αποτελούνται από βήματα $N=(0,1)$ και $E=(1,0)$ είναι προφανώς ίσο με $\binom{n+k}{k}$.



Σχήμα: Το μονοπάτι *NEENNEEENEENENEE*

Αν θέσουμε x_i το πλήθος των βημάτων πάνω στην ευθεία $y=i$, $0 \leq i \leq n$, τότε κάθε μία από τις $\binom{n+k}{k}$ λύσεις της

$$x_0 + x_1 + \cdots + x_n = k, \qquad x_i \in \mathbb{N}$$

κωδικοποιείται από ένα τέτοιο μονοπάτι.

Εφαρμογή

Να αποδειχθούν οι ταυτότητες

$$\sum_{i=0}^{n} \binom{i+k}{i} = \binom{n+k+1}{n}, \qquad \sum_{i=0}^{n} \binom{i}{k} = \binom{n+1}{k+1}.$$

Υπόδειξη: Για την πρώτη ταυτότητα, θεωρήστε το σύνολο των μονοπατιών που καταλήγουν στο $(k+1, n)$ και διαμερίστε το με βάση το σημείο $(k+1, i)$ στο οποίο τέμνουν για πρώτη φορά την ευθεία $x = k+1$. Η δεύτερη λύνεται ανάλογα, ή βάσει της πρώτης, κατόπιν αλλαγής μεταβλητής.

k-υποσύνολα - Ασκήσεις

- Να διατυπωθεί αλγόριθμος κατασκευής του συνόλου $\mathcal{F}_{n,k} \subseteq \mathcal{B}_{n,k}$ των δυαδικών λέξεων μήκους n με k μονάδες και χωρίς διαδοχικές μονάδες
- Να διατυπωθεί αναδρομικός αλγόριθμος κατασκευής των k-υποσυνόλων του $[n]$ που το άθροισμα των στοιχείων τους ισούται με s.
- Να διατυπωθεί αλγόριθμος κατασκευής των στοιχείων του $\{b \in \mathcal{B}_n : b \leq_L \mathrm{rev}(b)\}$.

Μεταθέσεις

- Μια μετάθεση σ είναι μια αμφιμονοσήμαντη απεικόνιση $\sigma : S \to S$, όπου S μη κενό, πεπερασμένο σύνολο. Όπως και στα προηγούμενα, θεωρούμε χωρίς βλάβη της γενικότητας ότι $S = [n]$, οπότε το σύνολο των μεταθέσεων συμβολίζεται με $\mathbb{S}_n$.
- Μια μετάθεση $\sigma \in \mathbb{S}_n$ συμβολίζεται ως μια λέξη $\sigma = \sigma_1\sigma_2\cdots\sigma_n$, όπου $\sigma_i = \sigma(i)$.
- Μια αντιστροφή (inversion) της $\sigma \in \mathbb{S}_n$ είναι ένα ζεύγος δεικτών (i, j) με $i < j$ και $\sigma_i > \sigma_j$. Το πλήθος των αντιστροφών της σ συμβολίζεται με $\mathrm{inv}(\sigma)$.
- Το πρόσημο της σ ορίζεται ως $\mathrm{sign}(\sigma) = (-1)^{\mathrm{inv}(\sigma)}$ και αν είναι θετικό ονομάζεται άρτια, αλλιώς περιττή.

Μεταθέσεις - ακολουθία αντιστροφών

- Η ακολουθία αντιστροφών της σ είναι η λέξη $I(\sigma) = I_1 I_2 \cdots I_n$, όπου $I_i = |\{j > i : \sigma_j < \sigma_i\}|$.
- Προφανώς, $0 \leq I_i \leq n - i$ και $\mathrm{inv}(\sigma) = \sum_{i=1}^{n} I_i$.
- Η απεικόνιση $I : \mathbb{S}_n \to R_n := \{0, 1, \ldots, n-1\} \times \cdots \times \{0, 1\} \times \{0\}$ είναι αμφιμονοσήμαντη και επιπλέον διατηρεί τη λεξικογραφική διάταξη (άσκηση).
- Επομένως, προκύπτει ότι

$$\sum_{\sigma \in \mathbb{S}_n} q^{\mathrm{inv}(\sigma)} = \sum_{I_1 \cdots I_n \in R_n} q^{\sum_{i=1}^{n} I_i} = \sum_{I_1=0}^{n-1} \sum_{I_2=0}^{n-2} \cdots \sum_{I_n=0}^{0} q^{I_1} \cdots q^{I_n} = \prod_{i=1}^{n} \sum_{I_i=0}^{n-i} q^{I_i}$$
$$= (1+q)(1+q+q^2) \cdots (1+q+\cdots+q^{n-1})$$

- Η μετατροπή της σ στην $I(\sigma)$ γίνεται με $O(n^2)$ πράξεις, αλλά μπορεί να επιτευχθεί και με $O(n \log n)$ πράξεις, τροποποιώντας τον γνωστό αλγόριθμο ταξινόμησης mergesort (άσκηση).

Μεταθέσεις - ακολουθία αντιστροφών

Αν γνωρίζουμε την ακολουθία αντιστροφών $r = r_1 \cdots r_n \in R_n$, τότε μπορούμε να ανακτήσουμε τη μετάθεση $\sigma = I^{-1}(r)$ ως εξής:

```
1 InvToPerm(r) begin                                  /* S = [1, 2, ..., n] */
2     for i = 1 to n do
3         x := S[1 + r_i];
4         σ_i := x;
5         S.remove(x);
6     return σ;
```

Αλγόριθμος 17: Υπολογισμός της μετάθεσης $\sigma = I^{-1}(r)$, $r \in R_n$.

Για παράδειγμα, η ακολουθία

$$r = 252100010 \in R_9$$

αντιστοιχεί στη μετάθεση

$$\sigma = 374215698 \in \mathbb{S}_9.$$

Συμβολισμοί: Αν S πεπερασμένο σύνολο, τότε

- $2^S = \{A : A \subseteq S\}$
- $\binom{S}{k} = \{A \subseteq S : |A| = k\}$, $k \in \mathbb{N}$.

Άσκηση

Ονομάζουμε σύνολο τόξων χωρίς αριστερές κορυφές κάθε σύνολο $A \subset \binom{[n]}{2}$ τέτοιο ώστε $\min B \neq \min C$, για κάθε $B, C \in A$. Συμβολίζουμε με A_n το σύνολο αυτών.
Να δειχθεί ότι $|A_n| = |R_n|$ και να δοθεί μια αμφιμονοσήμαντη απεικόνιση $f : A_n \to R_n$.
Στη συνέχεια, να δοθεί μια απλή περιγραφή της απεικόνισης $I^{-1} \circ f : A_n \to \mathbb{S}_n$.

Ο υπολογισμός της επόμενης λεξικογραφικά μετάθεσης, π.χ. της $892157643 \in \mathbb{S}_9$, μπορεί να γίνει ως εξής:

- Εντοπίζουμε το δεξιότερο στοιχείο που μπορεί να αυξηθεί χωρίς να αλλάξουν τα στοιχεία αριστερά από αυτό.
  Αυτό αντιστοιχεί σε έναν δείκτη k με $\sigma_k < \sigma_{k+1} > \cdots > \sigma_n$. Στο συγκεκριμένο παράδειγμα είναι το 5.
- Το στοιχείο αυτό θα εναλλαχθεί με το $\sigma_j = \min\{\sigma_i > \sigma_k : i > k\}$.
  Στο συγκεκριμένο παράδειγμα είναι το 6.
- Τέλος, τα στοιχεία δεξιά του θα διαταχθούν σε αύξουσα σειρά, ώστε να δημιουργήσουν το ελάχιστο δυνατό επίθεμα (suffix).
  Επειδή τα στοιχεία αυτά είναι ήδη σε φθίνουσα σειρά, αρκεί να αντιστραφεί η σειρά τους.

$$892157643 \to 892167543 \to 892163457$$

Μεταθέσεις - Επαναληπτικός αλγόριθμος κατασκευής

```
1  permLex() begin                                   /* assumes σ_0 = 0 */
2      k := n − 1;
3      while σ_k > σ_{k+1} do k := k − 1;
4      if k = 0 then return false;
5      j := n;
6      while σ_k > σ_j do j := j − 1;
7      σ_k :=: σ_j;                                          /* swap */
8      r := n;
9      s := k + 1;
10     while s < r do                                /* reverse suffix */
11         σ_s :=: σ_r;                                      /* swap */
12         r := r − 1;
13         s := s + 1
14     return true;
```

Αλγόριθμος 18: Επαναληπτικός CAT αλγόριθμος λεξικογραφικής κατασκευής του $\mathbb{S}_n$.

Μεταθέσεις - Επαναληπτικός αλγόριθμος κατασκευής

Για την ανάλυση του αλγορίθμου, αρκεί να μετρήσουμε το πλήθος εναλλαγών (swaps), το οποίο ας συμβολίσουμε με t_n. Για τις μεταθέσεις που ξεκινούν με i γίνονται t_{n-1} εναλλαγές. Επιπλέον, για $i \leq n-1$, η επόμενη της $in(n-1)\cdots(i+1)(i-1)\cdots 1$ είναι η $(i+1)12\cdots i(i+2)\cdots n$, η οποία προκύπτει με $\lfloor (n+1)/2 \rfloor$ εναλλαγές, άρα

$$t_n = nt_{n-1} + (n-1)\lfloor (n+1)/2 \rfloor, \qquad t_1 = 0.$$

Θέτοντας $s_n = t_n + \lfloor (n+1)/2 \rfloor$ προκύπτει η σχέση

$$s_n = n(s_{n-1} + [n \text{ περιττός}]), \qquad s_1 = 1,$$

η οποία δίνει

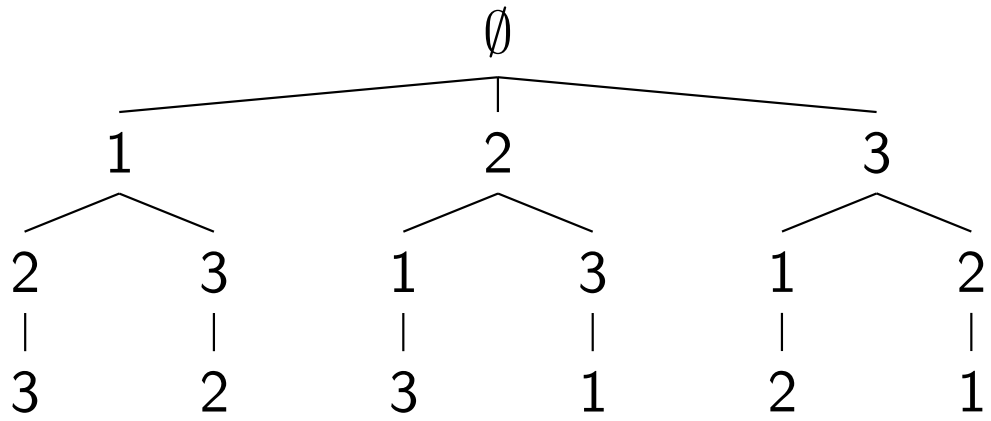
$$s_n = n! \left(1 + \frac{1}{2!} + \frac{1}{4!} + \cdots + \frac{1}{2\lfloor (n-1)/2 \rfloor}\right)$$

Άρα $s_n/n! \sim \cosh 1 \simeq 1.53$, δηλαδή $t_n/n! = O(1)$.

```
1 permRec(k) begin                    /* used[] initialized to false */
2     for i = 1 to n do
3         if used[i] = false then
4             P[k] := i;
5             if k = n then output(P);
6             else
7                 used[i] := true;
8                 permRec(k + 1);
9                 used[i] := false;
```

Αλγόριθμος 19: Αναδρομικός αλγόριθμος λεξικογραφικής κατασκευής του $\mathbb{S}_n$. Η κατασκευή του $\mathbb{S}_n$ πραγματοποιείται με την κλήση permRec(1).

Παρακάτω φαίνεται το δένδρο της αναδρομής για $n = 3$.



Το πλήθος $N(n)$ των αναδρομικών κλήσεων ισούται με το πλήθος όλων των κόμβων πλην των φύλλων στο δένδρο της αναδρομής. Η ρίζα (επίπεδο 1) έχει n παιδιά, οι κόμβοι στο επίπεδο 2 έχουν από $n-1$ παιδιά κοκ. Επομένως,

$$N(n) = 1+n+n(n-1)+n(n-1)(n-2)+\cdots+n! = 1+nN(n-1), \quad N(1) = 1,$$

οπότε

$$\frac{N(n)}{n!} = \frac{1}{n!} + \frac{N(n-1)}{(n-1)!} = \cdots = \sum_{k=1}^{n} \frac{1}{k!} \leq e.$$

Όμως, σε κάθε κόμβο εκτελούνται $\Theta(n)$ σε πλήθος εντολές, αφού το for της γραμμής 2 εκτελεί πάντα n επαναλήψεις. Επομένως, ο αλγόριθμος δεν είναι CAT.
Αν όμως τροποποιήσουμε τον αλγόριθμο ώστε σε κάθε κόμβο στο επίπεδο $k \in [n]$ να δοκιμάζουμε για το $P[k]$ μόνο τις $n-k+1$ τιμές που υπολείπονται, χωρίς την ανάγκη για τον έλεγχο της γραμμής 3, τότε ο συνολικός χρόνος $T(n)$ των αναδρομικών κλήσεων θα ισούται με

$$T(n) = n+n(n-1)+n(n-1)(n-2)+\cdots+n! = n+nT(n-1), \quad T(1)=1$$

και ο νέος αλγόριθμος θα είναι CAT.
Αυτό επιτυγχάνεται αποθηκεύοντας σε μια δομή συνόλου τα στοιχεία του $[n]$ που δεν έχουν ακόμα χρησιμοποιηθεί.

```
1 permRecB(k) begin      /* unused initialized to {1,...,n} */
2     for i ∈ unused do
3         P[k] := i;
4         if k = n then output(P);
5         else
6             unused = unused \ {i};
7             permRecB(k + 1);
8             unused = unused ∪ {i};
```

Αλγόριθμος 20: Αναδρομικός CAT αλγόριθμος κατασκευής του $\mathbb{S}_n$.

Ο αλγόριθμος 20 είναι CAT όταν οι πράξεις προσθήκης και διαγραφής ενός στοιχείου από το σύνολο μεγέθους k απαιτούν χρόνο $O(1)$ κατά μέσο όρο. Η κατασκευή είναι λεξικογραφική, εφόσον τα στοιχεία του συνόλου επιλέγονται πάντα σε αύξουσα σειρά.

```
1 permRecC(i) begin                    /* P initialized to [1,2,...,n] */
2     if i = n then output(P);
3     else
4         for j = i to n do
5             P[i] :=: P[j];                                  /* swap */
6             permRecC(i + 1);
7             P[i] :=: P[j];                              /* swap back */
```

Αλγόριθμος 21: Αναδρομικός CAT αλγόριθμος κατασκευής του $\mathbb{S}_n$.

Ο αλγόριθμος 21 είναι CAT αλλά η κατασκευή δεν είναι λεξικογραφική. Η ορθότητά του αποδεικνύεται επαγωγικά και στηρίζεται στο γεγονός ότι κατά την κλήση permRecC(i) το πρόθεμα $\sigma_1 \cdots \sigma_{i-1}$ έχει οριστικοποιηθεί και ο βρόχος for θα ελέξγει όλες τις δυνατές επιλογές για την τιμή σ_i.

Η τυχαία επιλογή ενός αντικειμένου ισοδυναμεί με την τυχαία επιλογή ενός μονοπατιού από τη ρίζα προς κάποιο φύλλο στο δένδρο της αναδρομικής κατασκευής. Έτσι, βάσει του προηγούμενου αλγορίθμου, προκύπτει ο εξής απλός αλγόριθμος κατασκευής τυχαίας μετάθεσης:

```
1 permRand(n) begin          /* P initialized to [1, 2, ..., n] */
2     for i = 1 to n do
3         j = rand(i, n);    /* choose random j ∈ {i, i + 1, ..., n} */
4         P[i] :=: P[j];                                  /* swap */
```

Αλγόριθμος 22: Κατασκευή τυχαίας μετάθεσης του $\mathbb{S}_n$.

Έστω $\sigma = \sigma_1\sigma_2\cdots\sigma_n \in \mathbb{S}_n$ και έστω $I(\sigma) = I_1 I_2 \cdots I_n$.
Το rank της σ σε λεξικογραφική διάταξη προφανώς ικανοποιεί τη σχέση

$$\mathrm{rank}_L(\sigma) = (\sigma_1 - 1)(n-1)! + \mathrm{rank}_L(\sigma'_2\sigma'_3\cdots\sigma'_n),$$

όπου $\sigma'_i = \sigma_i - [\sigma_i > \sigma_1]$, για $2 \le i \le n$. Δηλαδή αφαιρούμε μια μονάδα από κάθε στοιχείο σ_i μεγαλύτερο του σ_1, ώστε $\sigma' = \sigma'_2\cdots\sigma'_n \in \mathbb{S}_{n-1}$.
Η υλοποίηση της παραπάνω αναδρομικής σχέσης δίνει αλγόριθμο ranking πολυπλοκότητας $O(n^2)$.
Εναλλακτικά, η σχέση

$$\mathrm{rank}_L(\sigma) = \mathrm{rank}_L(I(\sigma)) = (n-1)!I_1 + \mathrm{rank}_L(I_2\cdots I_n) = \cdots = \sum_{i=1}^{n}(n-i)!I_i$$

μπορεί να δώσει αλγόριθμο ranking πολυπλοκότητας $O(n \log n)$, αν η απεικόνιση I υλοποιηθεί κατάλληλα.

```
1  permRankRevColex(σ) begin
2      r := 0;
3      f := 1;
4      for i := 2 to n do
5          c := 0;
6          for j := 1 to i − 1 do
7              if σ_j > σ_i then c := c + 1;
8          r := r + c ∗ f;
9          f := f ∗ i;
10     return r;
```

Αλγόριθμος 23: Αλγόριθμος ranking του $\mathbb{S}_n$ σε αντίστροφη colex διάταξη.

Μεταθέσεις - Ranking

S_4 in lex	S_4 in colex	$I(S_4)$ in lex	$I(S_4)$ in colex
123	321	000	000
132	231	010	100
213	312	100	200
231	132	110	010
312	213	200	110
321	123	210	210

Πολυμεταθέσεις

Μια πολυμετάθεση είναι μια αναδιάταξη των στοιχείων μιας συλλογής C (αντί συνόλου) με στοιχεία στο $\{1, \ldots, t\}$, όπου η C περιέχει n_i εμφανίσεις του $i \in [t]$. Το διάνυσμα $(n_1, n_2 \ldots, n_t)$ ονομάζεται τύπος της συλλογής και θέτουμε $n = |C| = n_1 + n_2 + \cdots + n_t$.
Ως γνωστό, το πλήθος των πολυμεταθέσεων αυτών ισούται με

$$\binom{n}{n_1, n_2, \ldots, n_t} = \frac{n!}{n_1! n_2! \cdots n_t!}$$

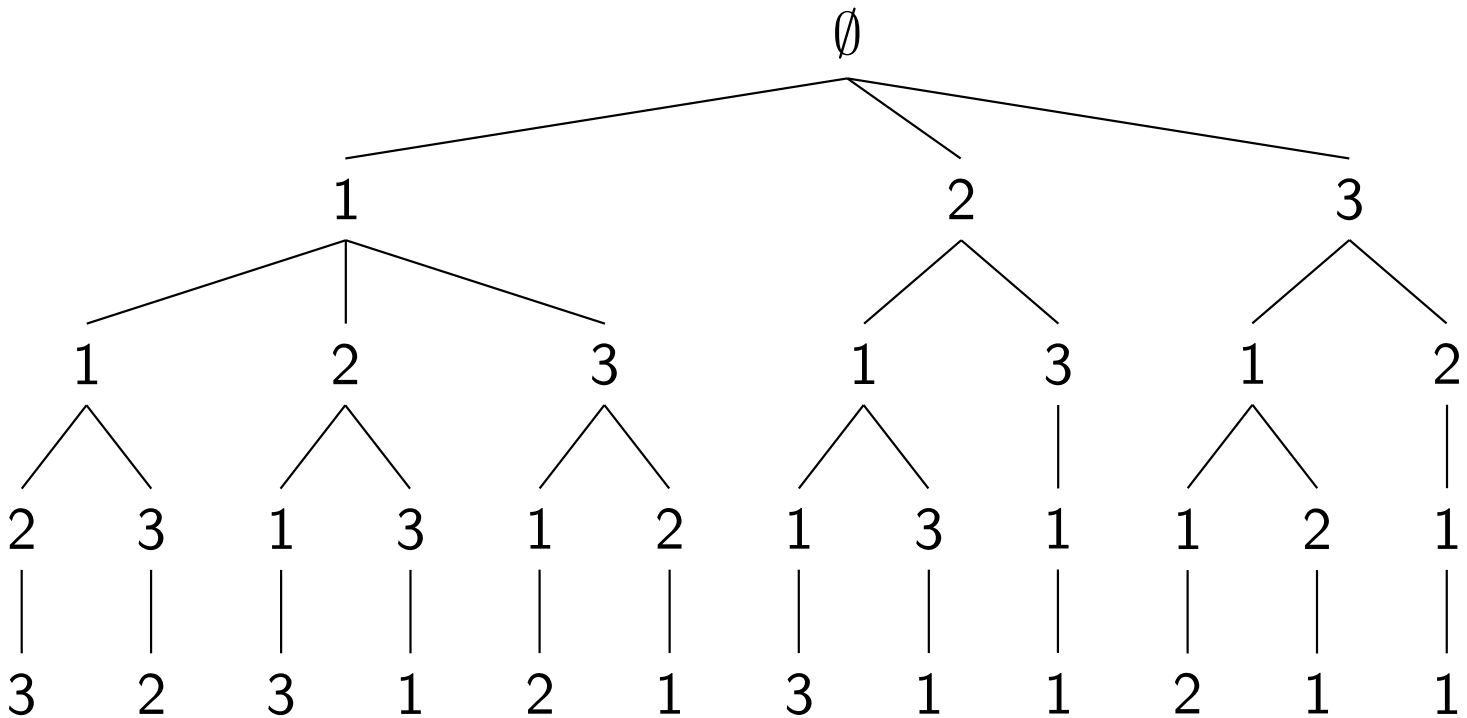
Ειδικές περιπτώσεις

- Αν $t = 2$, προκύπτουν απλοί συνδυασμοί των n ανά n_1.
- Αν $n_1 = n_2 = \cdots = n_t = 1$, προκύπτουν απλές μεταθέσεις του $[t]$.

```
1 multiPermRecColex(k) begin
2     if k = n_1 then output(M);
3     else
4         for j ∈ {p ∈ [t] : n_p > 0} do
5             M[k] := j;
6             n_j := n_j − 1;
7             multiPermRecColex (k − 1);
8             n_j := n_j + 1;
9             M[k] := 1;
```

Αλγόριθμος 24: Αναδρομικός αλγόριθμος κατασκευής πολυμεταθέσεων τύπου $(n_1, \ldots, n_t)$ σε colex διάταξη.

Παρακάτω φαίνεται το δένδρο της αναδρομής όταν $t = 3$ και $(n_1, n_2, n_3) = (2, 1, 1)$.



Επαναληπτικοί συνδυασμοί

Όπως έχουμε δει, οι επαναληπτικοί συνδυασμοί μπορούν να θεωρηθούν ως αύξουσες ακολουθίες. Όπως και στις πολυμεταθέσεις, τα στοιχεία επιλέγονται από μια συλλογή, η οποία μπορεί να περιέχει επαναλήψεις. Αν η συλλογή είναι συγκεκριμένου τύπου $(n_1, n_2, \ldots, n_t)$, ένα ερώτημα είναι πόσοι επαναληπτικοί k-συνδυασμοί υπάρχουν. Σύμφωνα με τα προηγούμενα, θα ισοδυναμούν με τις ακέραιες λύσεις της εξίσωσης

$$x_1 + x_2 + \cdots + x_t = k, \qquad 0 \leq x_i \leq n_i$$

Ειδικά για $i = t$, θα πρέπει $\max\{0, k - n + n_t\} \leq x_t \leq \min\{n_t, k\}$. Συμβολίζουμε ως $C(k; n_1, \ldots, n_t)$ το πλήθος των λύσεων αυτών. Επομένως,

$$C(k; n_1, \ldots, n_t) = \sum_{i=\max\{0,k-n+n_t\}}^{\min\{n_t,k\}} C(k - i; n_1, \ldots, n_{t-1})$$

όπου $C(0; n_1, \ldots, n_t) = C(n; n_1, \ldots, n_t) = 1$.

Επαναληπτικοί συνδυασμοί - Αναδρομικοί αλγόριθμοι κατασκευής

Το πλήθος αυτό μπορεί να υπολογισθεί ως ο συντελεστής του x^k ενός πολυωνύμου του x:

$$C(k; n_1, \ldots, n_t) = [x^k] \prod_{i=1}^{t} (1 + x + \cdots + x^{n_i}) = [x^k] \prod_{i=1}^{t} \frac{1 - x^{1+n_i}}{1 - x}$$

```
1 multiCombRecColex(k, t, n) begin
2     if k = 0 then output(x);
3     else
4         for i := max{0, k − n − n_t} to min{n_t, k} do
5             x[t] := i;
6             multiCombRecColex (k − i, t − 1, n − n_t);
7             x[t] := 0
```

Αλγόριθμος 25: Αναδρομικός αλγόριθμος κατασκευής του $C(k; n_0, n_1, \ldots, n_k)$ σε colex διάταξη, CAT για $k \leq n/2$.

```
1 multiCombRecColexB(k, t, n) begin
2     if k = n then output(x);
3     else
4         for i := max{0, k − n − n_t} to min{n_t, k} do
5             x[t] := i;
6             multiCombRecColexB (k − i, t − 1, n − n_t);
7             x[t] := n_t
```

Αλγόριθμος 26: Αναδρομικός αλγόριθμος κατασκευής του $C(k; n_0, n_1, \ldots, n_k)$ σε colex διάταξη, CAT για $k > n/2$.

- Να διατυπωθεί αλγόριθμος κατασκευής των k-διατάξεων του $[n]$.
- Να διατυπωθεί αλγόριθμος κατασκευής των up-down μεταθέσεων του $[n]$, δηλαδή μεταθέσεων $\sigma \in S_n$, με $\sigma_1 < \sigma_2 > \sigma_3 < \sigma_4 \cdots$. Να δειχθεί ότι το πλήθος $E(n,k)$ αυτών με $\sigma_1 = k$, ικανοποιεί τη σχέση

$$E(n,k) = E(n,k+1) + E(n-1,n-k)$$

- Το στοιχείο $i \in [n]$ ονομάζεται σταθερό σημείο της μετάθεσης $\sigma \in S_n$ ανν $\sigma(i) = i$. Οι μεταθέσεις χωρίς σταθερά σημεία ονομάζονται derangements.
  Να διατυπωθεί αλγόριθμος κατασκευής των derangements του $[n]$ και να δειχθεί ότι το πλήθος τους d_n ικανοποιεί τη σχέση

$$d_n = (n-1)(d_{n-1} + d_{n-2})$$

Δένδρα

Γενικά, ένα δένδρο T είναι ένα άκυκλο συνεκτικό γράφημα, στο οποίο μπορεί να έχουμε ορίσει:

- Μια ρίζα ως έναν συγκεκριμένο κόμβο (rooted (planted) trees)
- Μια ολική διάταξη στους γείτονες κάθε κόμβου (ordered (plane) trees)
- Μια ετικέτα σε κάθε κόμβο (labeled trees)

Συμβολίζουμε με

- $|T|$ το πλήθος των κόμβων του T.
- $\deg v$ το βαθμό του κόμβου v, δηλαδή το πλήθος γειτόνων του.

Κόμβοι βαθμού ≤ 1 ονομάζονται εξωτερικοί κόμβοι ή φύλλα. Οι υπόλοιποι ονομάζονται εσωτερικοί.

Πρόταση

Έμεσα προκύπτει ότι κάθε δένδρο με n κόμβους έχει $n-1$ ακμές και, για $n \geq 2$, τουλάχιστον δύο φύλλα.

Απόδειξη.

Με επαγωγή στο n. □

Πρόταση

Αν $d_1, d_2, \ldots, d_n$ θετικοί ακέραιοι, $n \geq 2$, με άθροισμα $2n-2$, τότε υπάρχει δένδρο με n κόμβους με αυτούς τους βαθμούς.

Απόδειξη.

Διαγραφή ενός φύλου και επαγωγή στο n. □

Δένδρα με ετικέτες και χωρίς ρίζα (free labeled trees)

Συμβολίζουμε με L_n το πλήθος των (μη διατεταγμένων) δένδρων χωρίς ρίζα με n κόμβους και ετικέτες $1, 2, \ldots, n$.

Πρόταση

Για $n \geq 2$ ισχύει $|L_n| = n^{n-2}$.
Επιπλέον, αν d_i, $i \in [n]$, είναι ο βαθμός του κόμβου με ετικέτα i, τότε υπάρχουν $\dfrac{(n-2)!}{(d_1-1)! \cdots (d_n-1)!}$ δένδρα του L_n με αυτούς τους βαθμούς.

Το πρώτο σκέλος της παραπάνω πρότασης ουσιαστικά λέει ότι τα δένδρα αυτά είναι ισοπληθικά με τις απεικονίσεις $p : [n-2] \to [n]$, δηλαδή ακολουθίες $(p_1, \ldots, p_{n-2})$ με όρους στο $[n]$. Επομένως κωδικοποιώντας τα δένδρα αυτά από τέτοιες ακολουθίες, μπορούμε να τα κατασκευάσουμε αποδοτικά.
Η κωδικοποίηση αυτή ονομάζεται κώδικας Prufer.

Κώδικας Prufer

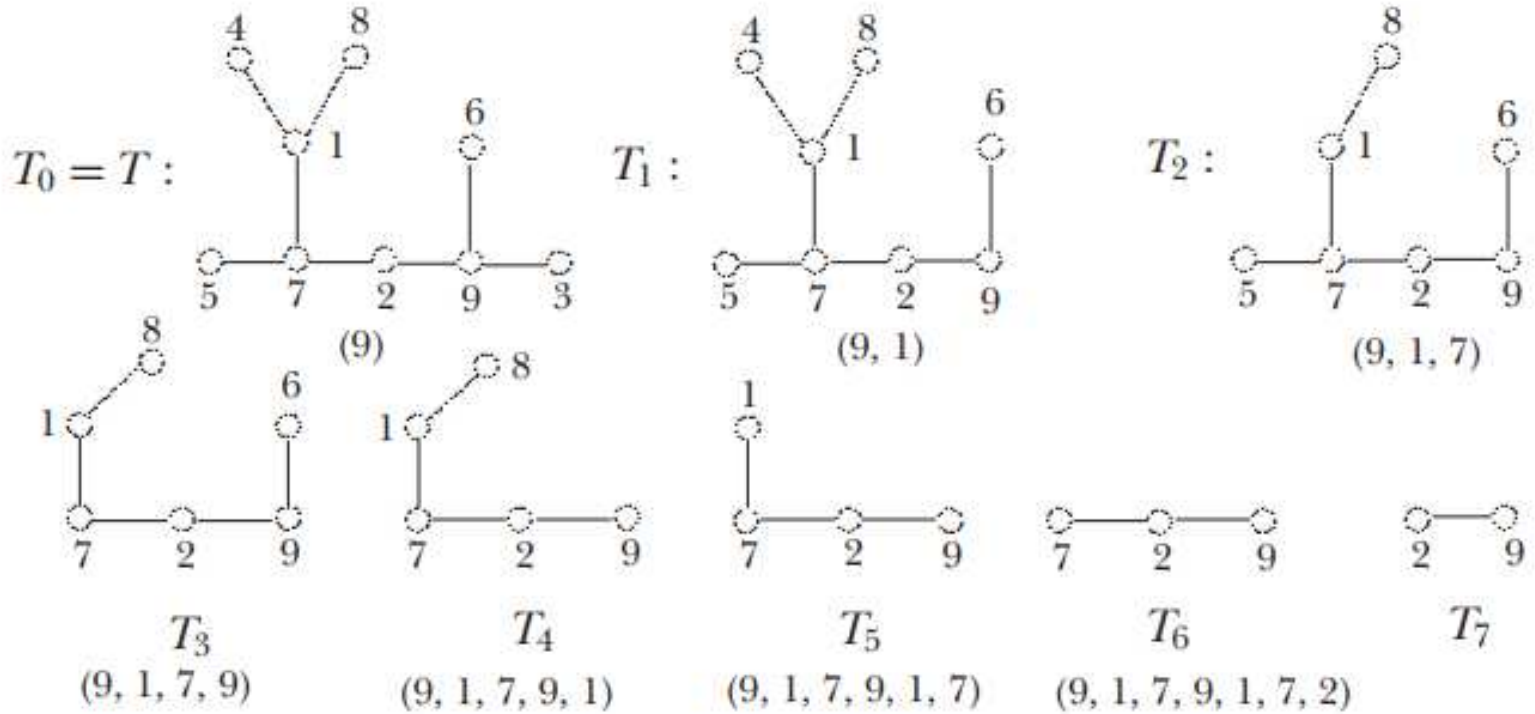
Δοθέντος ενός δένδρου $T \in L_n$, ορίζουμε την ακολουθία δένδρων $T = T_1, T_2, \ldots, T_{n-1} = K_2$ όπου το T_{i+1} προκύπτει από το T_i αφαιρώντας το φύλλο με την μικρότερη ετικέτα και K_2 είναι το δένδρο με 2 κόμβους και μία ακμή.
Επιπλέον, ορίζουμε την ακολουθία $(p_i)_{1 \leq i \leq n-2}$ ως εξής:
p_i είναι η ετικέτα του γείτονα του φύλλου που αφαιρέθηκε από το T_i για να προκύψει το T_{i+1}.
Πρακτικά, η p κατασκευάζεται επαναληπτικά εισάγοντας κάθε φορά ως τελευταίο όρο της ακολουθίας την ετικέτα του γείτονα του μικρότερου φύλλου και κατόπιν σβήνοντας το φύλλο αυτό.
Προφανώς, $p \in [n]^{[n-2]}$.

Κώδικας Prufer

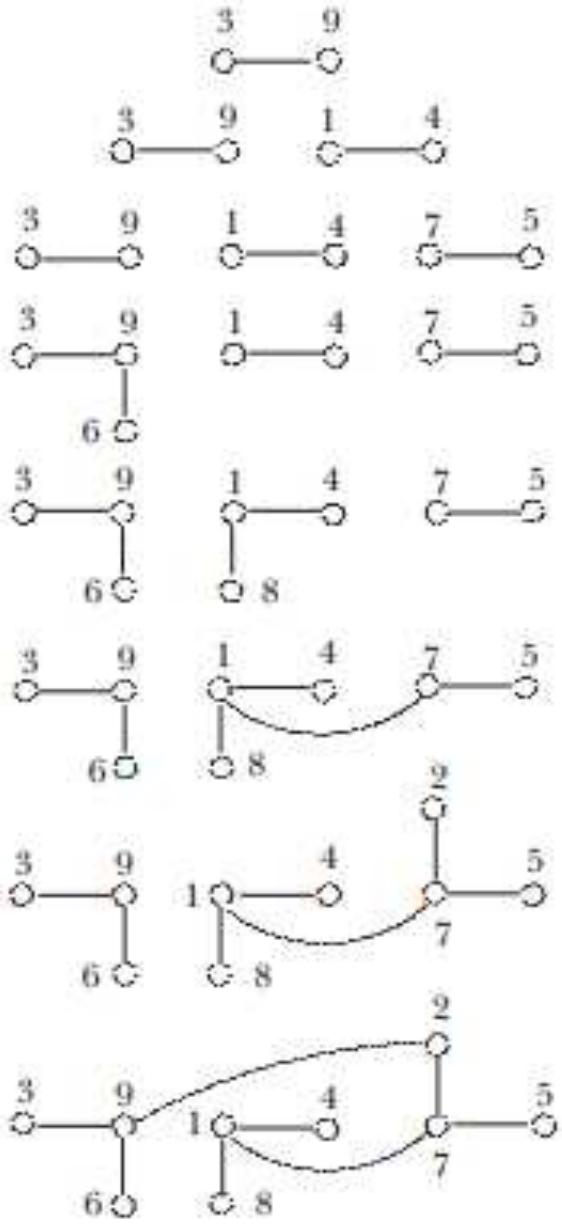
Για την αντίστροφη διαδικασία της ανάκτησης του δένδρου από την ακολουθία p εργαζόμαστε ως εξής:

- Ξεκινάμε με το σύνολο $S = \{1, 2, \ldots, n\}$ και την ακολουθία p.
- Επιλέγουμε το μικρότερο στοιχείο του S που δεν εμφανίζεται στην p και το ενώνουμε με το πρώτο στοιχείο της p.
- Αφαιρούμε τα δύο αυτά στοιχεία από το S και από την p και επαναλαμβάνουμε, έως ότου η p γίνει κενή.
- Τότε θα έχουν απομείνει 2 στοιχεία στο S, τα οποία ενώνουμε.

Κώδικας Prufer - Παράδειγμα



Κώδικας Prufer - Παράδειγμα

Sequence	Set	Tree
(9, 1, 7, 9, 1, 7, 2)	{1, 2, 3, 4, 5, 6, 7, 8, 9}	
(1, 7, 9, 1, 7, 2)	{1, 2, 4, 5, 6, 7, 8, 9}	
(7, 9, 1, 7, 2)	{1, 2, 5, 6, 7, 8, 9}	
(9, 1, 7, 2)	{1, 2, 6, 7, 8, 9}	
(1, 7, 2)	{1, 2, 7, 8, 9}	
(7, 2)	{1, 2, 7, 9}	
(2)	{2, 7, 9}	
	{2, 9}	

Διατεταγμένα δένδρα με ρίζα

Τα διατεταγμένα δένδρα με ρίζα με $n+1$ κόμβους (όπως και τα δυαδικά δένδρα με n κόμβους) είναι σε πλήθος C_n, όπου

$$C_n = \frac{1}{n+1}\binom{2n}{n}, \qquad n \geq 0$$

είναι ο n-οστός αριθμός Catalan.
Η κατασκευή τους απλοποιείται αν κωδικοποιηθούν με κατάλληλες δυαδικές λέξεις, οι οποίες ονομάζονται λέξεις Dyck.
Για την ανάλυση των αλγορίθμων κατασκευής, θα χρησιμοποιηθούν έννοιες που σχετίζονται με τις γεννήτριες συναρτήσεις και παρουσιάζονται παρακάτω.
Επιπλέον θα παρουσιαστούν διάφορα αντικείμενα που επίσης απαριθμούνται από τους αριθμούς Catalan, οπότε μπορούν να κατασκευαστούν με τους ίδιους αλγορίθμους.

Ορισμός

Η (συνήθης) γεννήτρια συνάρτηση (ΓΣ) της ακολουθίας $(a_n)_{n\in\mathbb{N}}$ πραγματικών αριθμών ορίζεται ως η τυπική δυναμοσειρά

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots = \sum_{n\geq 0} a_n x^n$$

Ανάλογα ορίζονται και οι γεννήτριες συναρτήσεις πολλών μεταβλητών για ακολουθίες πολλών μεταβλητών.

Συμβολισμοί-ορισμοί

- $[x^n]A(x) = a_n$
- $A(0) = [x^0]A(x) = a_0$
- (Τυπική παράγωγος) $\frac{d}{dx}A(x) = A'(x) = \sum_{n\geq 1} n a_n x^{n-1}$
- (Σύνθεση) $A(B(x)) = \sum_{n\geq 0}(B(x))^n$ ορίζεται ανν $B(0) = 0$
- (Συνέλιξη) $A(x)B(x) = \sum_{n\geq 0}\sum_{k=0}^{n} a_k b_{n-k} x^n$
  Ειδική περίπτωση: Για $B(x) = 1/(1-x)$, προκύπτει
  $[x^n]\frac{1}{1-x}A(x) = \sum_{k=0}^{n} a_k$

Εφαρμογή: Γεννήτρια συνάρτηση των αρμονικών αριθμών

$$H_n := \sum_{k=1}^{n} \frac{1}{k} = [x^n]\frac{1}{1-x}\ln\frac{1}{1-x}$$

Θεώρημα αντιστροφής Lagrange (LIF)

Αν $F(x), H(x)$ γεννήτριες συναρτήσεις, με $F(x) = 1 + xH(F(x))$, τότε

$$[x^n]F^s(x) = \frac{s}{n}[\lambda^{n-1}](1+\lambda)^{s-1}H^n(1+\lambda), \qquad n > 0, s \in \mathbb{R}$$

Εφαρμογή

Η ΓΣ $F(x)$ που ικανοποιεί τη σχέση $F(x) = 1 + xF^k(x)$ έχει συντελεστές τους αριθμούς k-Catalan $C_n^{(k)} = \frac{1}{kn+1}\binom{kn+1}{n}$.
Πράγματι, για $H(\lambda) = \lambda^k$, έχουμε $F(x) = 1 + xH(F(x))$, οπότε

$$[x^n]F^s(x) = \tfrac{s}{n}[\lambda^{n-1}](1+\lambda)^{kn+s-1} = \tfrac{s}{n}\binom{kn+s-1}{n-1} = \tfrac{s}{kn+s}\binom{kn+s}{n}$$

Οι συντελεστές

$$C_{n,s}^{(k)} = \frac{s}{kn+s}\binom{kn+s}{n}, \qquad n \geq 0, s \in \mathbb{N}^*, \tag{2}$$

της προηγούμενης εφαρμογής ονομάζονται γενικευμένοι αριθμοί k-Catalan (ή γενικευμένοι αριθμοί Fuss-Catalan) και απαριθμούν τα διατεταγμένα δάση από s k-αδικά δένδρα με συνολικά n εσωτερικούς κόμβους. Επιπλέον,

- Για $k = 2$, $s = 1$ προκύπτουν οι αριθμοί Catalan C_n.
- Για $k = 2$ προκύπτουν οι γενικευμένοι αριθμοί Catalan, $C_{n,s} = \frac{s}{2n+s}\binom{2n+s}{n}$, οι οποίοι απαιτούνται για τους αλγορίθμους ranking-unranking των αντικειμένων της οικογένειας Catalan.

Γεννήτριες συναρτήσεις

Βασικές γεννήτριες συναρτήσεις:

- $(1+x)^r = \sum_{n\geq 0} \binom{r}{n} x^n$, $r \in \mathbb{R}$ (διωνυμική σειρά)
- $\frac{1}{(1-x)^{r+1}} = \sum_{n\geq 0} \binom{r+n}{n} x^n$, $r \in \mathbb{R}$
- $\frac{x^m}{(1-x)^{m+1}} = \sum_{n\geq 0} \binom{n}{m} x^n$, $m \in \mathbb{N}$

Εφαρμογή

Να αποδειχθούν με τη βοήθεια των παραπάνω γεννητριών συναρτήσεων οι ταυτότητες

$$\sum_{i=0}^{n} \binom{i+k}{i} = \binom{n+k+1}{n}, \qquad \sum_{i=0}^{n} \binom{i}{k} = \binom{n+1}{k+1}.$$

Για την απόδειξη της $\sum_{i=0}^{n} \binom{i}{k} = \binom{n+1}{k+1}$, θα χρησιμοποιηθεί η βασική ΓΣ

$$\frac{x^m}{(1-x)^{m+1}} = \sum_{n \geq 0} \binom{n}{m} x^n \qquad (*)$$

καθώς και οι ιδιότητες

$$a_n = [x^n]A(x) \Rightarrow \sum_{i=0}^{n} a_i = [x^n]\frac{A(x)}{1-x} \qquad (**)$$

και

$$[x^n]x^k A(x) = [x^{n-k}]A(x), \quad 0 \leq k \leq n \qquad (***)$$

Βάσει της (*), έχουμε ότι

$$\sum_{i=0}^{n} \binom{i}{k} \overset{(*)}{=} \sum_{i=0}^{n} [x^i]\frac{x^k}{(1-x)^{k+1}} \overset{(**)}{=} [x^n]\frac{x^k}{(1-x)^{k+2}}$$

$$\overset{(***)}{=} [x^{n+1}]\frac{x^{k+1}}{(1-x)^{k+2}} \overset{(*)}{=} \binom{n+1}{k+1}$$

Γεννήτριες συναρτήσεις

Άλλες βασικές γεννήτριες συναρτήσεις είναι οι

$$C(x) = \frac{1-\sqrt{1-4x}}{2x} = \sum_{n\geq 0} C_n x^n, \qquad G(x) = \frac{1}{\sqrt{1-4x}} = \sum_{n\geq 0} \binom{2n}{n} x^n,$$

οι οποίες ορίζονται ισοδύναμα από τις σχέσεις

$$C(x) = 1 + xC^2(x), \qquad G(x) = 1 + 2xC(x)G(x).$$

Όπως είδαμε (LIF), ισχύει

$$[x^n]C^s(x) = \frac{s}{2n+s}\binom{2n+s}{n}, \qquad s \neq 0. \tag{3}$$

Επιπλέον, αποδεικνύεται ότι

$$[x^n]C^a(x)G(x) = \binom{2n+a}{n}, \qquad a \in \mathbb{R}. \tag{4}$$

Γεννήτρια συνάρτηση συνόλου

Έστω $\mathcal{A}$ ένα σύνολο συνδυαστικών αντικειμένων

Ορισμός

Μια (συνδυαστική) παράμετρος (ή στατιστική) είναι μια απεικόνιση $p : \mathcal{A} \to \mathbb{N}$.

Αν

$$\mathcal{A}_n = \{a \in \mathcal{A} : p(a) = n\}, \qquad a_n = |\mathcal{A}_n|.$$

και τα σύνολα $\mathcal{A}_n$ είναι πεπερασμένα για κάθε n, τότε η p καλείται κύρια παράμετρος στο $\mathcal{A}$.
Εισάγοντας μια δεύτερη παράμετρο $q : \mathcal{A} \to \mathbb{N}$, διαμερίζουμε περαιτέρω το $\mathcal{A}$ στις κλάσεις

$$\mathcal{A}_{n,k} = \{a \in \mathcal{A} : p(a) = n, q(a) = k\}, \qquad a_{n,k} = |\mathcal{A}_{n,k}|$$

κοκ.

Γεννήτρια συνάρτηση συνόλου

Ορισμός

Η γεννήτρια συνάρτηση του συνόλου $\mathcal{A}$ ως προς τη βασική παράμετρο p ορίζεται ως το άθροισμα

$$A(x) = \sum_{a \in \mathcal{A}} x^{p(a)}$$

Θέτοντας $|\mathcal{A}_n| = a_n$, οι δύο παραπάνω ορισμοί ταυτίζονται, αφού

$$\sum_{a \in \mathcal{A}} x^{p(a)} = \sum_{n=0}^{\infty} \sum_{a \in \mathcal{A}_n} x^n = \sum_{n=0}^{\infty} |\mathcal{A}_n| x^n$$

Γεννήτρια συνάρτηση συνόλου

Σε πολλές περιπτώσεις, χρησιμοποιούμε πολυμεταβλητές ΓΣ, ώστε να μελετήσουμε πολλές παραμέτρους ταυτόχρονα.
Με κατάλληλες αλλαγές μεταβλητής εξάγουμε επιπλέον πληροφορίες από την ίδια ΓΣ.
Προσδιορίζουμε στατιστικά μεγέθη μέσω των μερικών παραγώγων.
Για παράδειγμα, αν $A(x,y) = \sum_{a\in\mathcal{A}} x^{p(a)} y^{q(a)}$, τότε

$$A_y(x,y) := \frac{\partial}{\partial y} A(x,y) = \sum_{a\in\mathcal{A}} q(a) x^{p(a)} y^{q(a)-1} = \sum_{n\geq 0} \sum_{a\in\mathcal{A}_n} q(a) y^{q(a)-1} x^n$$

οπότε η μέση τιμή της q στο $\mathcal{A}_n$ ισούται με

$$E_n(q) := \frac{1}{|\mathcal{A}_n|} \sum_{a\in\mathcal{A}_n} q(a) = \frac{1}{|\mathcal{A}_n|} [x^n] A_y(x,1)$$

Ομοίως υπολογίζουμε ροπές ανώτερης τάξης.

ΓΣ συνόλου - Παράδειγμα

Για παράδειγμα, έστω $q(T)$ το βάθος του αριστερότερου φύλλου ενός δυαδικού δένδρου. Προφανώς, είναι $q(T) = 0$, όταν το T είναι κενό, αλλιώς είναι $q(T) = 1 + q(T_1)$, όπου T_1, T_2 είναι αντίστοιχα το αριστερό και το δεξί παιδί της ρίζας του T. Επομένως, ορίζοντας την αντίστοιχη διμεταβλητή ΓΣ $F = F(x, y)$ (με $F(x, 1) = C(x)$), είναι

$$F = \sum_T x^{|T|} y^{q(T)} = 1 + \sum_{T_1, T_2} x^{|T_1|+|T_2|+1} y^{1+q(T_1)}$$
$$= 1 + xy \sum_{T_1} x^{|T_1|} y^{q(T_1)} \sum_{T_2} x^{|T_2|} = 1 + xyFC$$

οπότε $F_y = xFC + xyCF_y$ και $F_y(x, 1) = xC^2 + xCF_y(x, 1)$.

Απο την τελευταία παίρνουμε ότι $F_y(x,1) = \dfrac{xC^2}{1-xC} = xC^3$, οπότε

$$[x^n]F_y(x,1) = [x^{n-1}]C^3(x) = \frac{3}{2n+1}\binom{2n+1}{n-1}$$

και διαιρώντας με το συνολικό πλήθος $C_n = \binom{2n}{n}/(n+1)$ των δυαδικών δένδρων, βρίσκουμε τελικά ότι

$$E_n(q) = \frac{3}{2n+1}\frac{(2n+1)!}{(n+2)!(n-1)!}\frac{(n+1)n!n!}{(2n)!} = \frac{3n}{n+2}$$

ΓΣ συνόλου - Παράδειγμα

Έστω $\mathcal{B}$ το σύνολο των δυαδικών λέξεων και έστω $|a|$ το μήκος της $a \in \mathcal{B}$ και $q(a)$ το πλήθος των 10 στην a. Κάθε $a \in \mathcal{B}$ γράφεται ως

$$a = \begin{cases} 0b, & b \in \mathcal{B}, \text{ ή} \\ 1^k, & k \geq 0, \text{ ή} \\ 1^k 0b, & b \in \mathcal{B}, k \geq 1 \end{cases}$$

Είναι $q(1^k) = 0, \quad q(1^k 0b) = 1 + q(b), \quad q(0b) = q(b)$, οπότε

$$\begin{aligned} F := F(x,y) &:= \sum_{a \in \mathcal{B}} x^{|a|} y^{q(a)} \\ &= \sum_{a=1^k, k \geq 0} x^k y^0 + \sum_{a=1^k 0b, b \in \mathcal{B}} x^{1+k+|b|} y^{1+q(b)} + \sum_{a=0b, b \in \mathcal{B}} x^{1+|b|} y^{q(b)} \\ &= \sum_{k \geq 0} x^k + \sum_{k \geq 1} x^{k+1} y \sum_{b \in \mathcal{B}} x^{|b|} y^{q(b)} + x \sum_{b \in \mathcal{B}} x^{|b|} y^{q(b)} \\ &= \frac{1}{1-x} + xyF \sum_{k \geq 1} x^k + xF = \frac{1 + x^2 yF}{1-x} + xF \end{aligned}$$

ΓΣ συνόλου - Παράδειγμα (συνέχεια)

Κατόπιν τούτων,

$$(1-x)^2F = 1 + x^2yF, \qquad F = 1/((1-x)^2 - x^2y) \qquad (*)$$

Παραγωγίζοντας ως προς y και ύστερα θέτοντας $y = 1$, έχουμε

$$(1-x)^2F_y = x^2F + x^2yF_y \Rightarrow (1-2x+x^2-x^2y)F_y = x^2F$$

$$\Rightarrow (1-2x)F_y(x,1) = \frac{x^2}{1-2x}$$

$$\Rightarrow F_y(x,1) = \frac{x^2}{(1-2x)^2} = \frac{x^2}{2}\frac{d}{dx}\sum_{n\geq 0} 2^n x^n$$

$$= \frac{x^2}{2}\sum_{n\geq 1} n2^n x^{n-1} = \sum_{n\geq 1} n2^{n-1}x^{n+1} = \sum_{n\geq 2}(n-1)2^{n-2}x^n$$

και τελικά,

$$E_n[q] = \frac{1}{2^n}[x^n]F_y(x,1) = \frac{(n-1)2^{n-2}}{2^n} = \frac{n-1}{4}$$

ΓΣ συνόλου - Παράδειγμα (συνέχεια)

Στο ίδιο συμπέρασμα καταλήγουμε πιο απλά, ορίζοντας την τυχαία μεταβλητή

$$X_i = \begin{cases} 1, & b_i b_{i+1} = 10 \\ 0, & \text{αλλιώς} \end{cases} \quad \text{όπου } i \in [n-1], b = b_1 b_2 \cdots b_n \in \mathcal{B}$$

Προφανώς είναι $E[X_i] = 1/4$ και

$$E_n[q] = E[X_1 + \cdots + X_{n-1}] = E[X_1] + \cdots + E[X_{n-1}] = (n-1)/4$$

ΓΣ συνόλου - Παράδειγμα (συνέχεια)

Για τον υπολογισμό του συντελεστή $[x^n y^k]F$ στην ΓΣ $F = F(x,y) = \dfrac{1}{(1-x)^2 - x^2 y}$, έχουμε ότι

$$[x^n y^k]F = [x^n]\frac{1}{(1-x)^2}[y^k]\frac{1}{1-\frac{x^2}{(1-x)^2}y} = [x^n]\frac{1}{(1-x)^2}\left(\frac{x^2}{(1-x)^2}\right)^k$$
$$= [x^n]\frac{x^{2k}}{(1-x)^{2k+2}} = [x^{n+1}]\frac{x^{2k+1}}{(1-x)^{2k+2}} = \binom{n+1}{2k+1},$$

βάσει των γνωστών ΓΣ

$$\frac{1}{1-ay} = \sum_{k\geq 0} a^k y^k, \qquad \frac{x^m}{(1-x)^{m+1}} = \sum_{n\geq 0}\binom{n}{m}x^n.$$

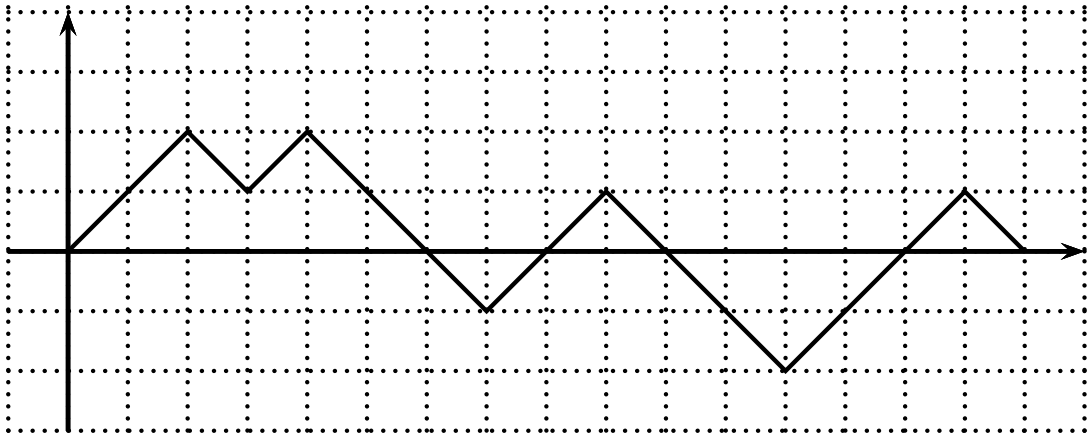
Μονοπάτια Grand-Dyck και Dyck

Ένα μονοπάτι Grand-Dyck α μήκους $2n$ είναι ένα μονοπάτι στο δικτυωτό $\mathbb{Z}^2$ από το σημείο $(0,0)$ στο $(2n,0)$, με βήματα $\mathrm{u} = (1,1)$ (άνοδοι) και $\mathrm{d} = (1,-1)$ (κάθοδοι).
Προφανώς, υπάρχουν $\binom{2n}{n}$ τέτοια μονοπάτια.
Μια άνοδος του α κάτω από τον άξονα x ονομάζεται βυθισμένη άνοδος. Τα μονοπάτια Grand-Dyck χωρίς βυθισμένες ανόδους ονομάζονται μονοπάτια Dyck.

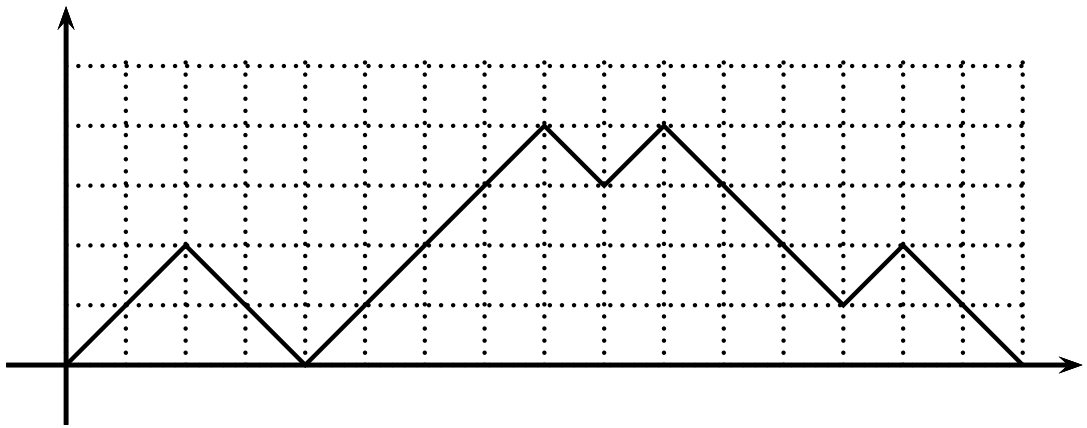
Συμβολισμοί

- ε το κενό μονοπάτι
- $\mathcal{G}_n$ το σύνολο των μονοπατιών Grand-Dyck μήκους $2n$ και $\mathcal{G} = \bigcup_{n\geq 0} \mathcal{G}_n$.
- $\mathcal{D}_n$ το σύνολο των μονοπατιών Dyck μήκους $2n$ και $\mathcal{D} = \bigcup_{n\geq 0} \mathcal{D}_n$.
- $|w|$ το μήκος του μονοπατιού w.
- $|w|_\tau$ το πλήθος εμφανίσεων της λέξης $\tau \in \{\mathrm{u},\mathrm{d}\}^*$ στο w.

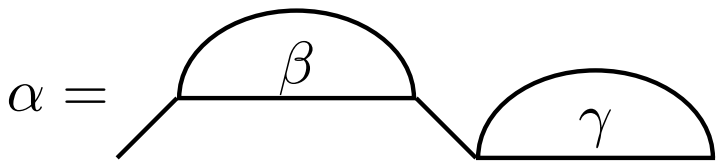
Μονοπάτια Grand-Dyck και Dyck



Σχήμα: Το μονοπάτι Grand-Dyck uududddduuddduuud.



Σχήμα: Το μονοπάτι Dyck uudduuuududddudd.



Σχήμα: Η διάσπαση πρώτης επιστροφής ενός μη κενού μονοπατιού Dyck α.

Από τη διάσπαση έχουμε μια συναρτησιακή εξίσωση της ΓΣ:

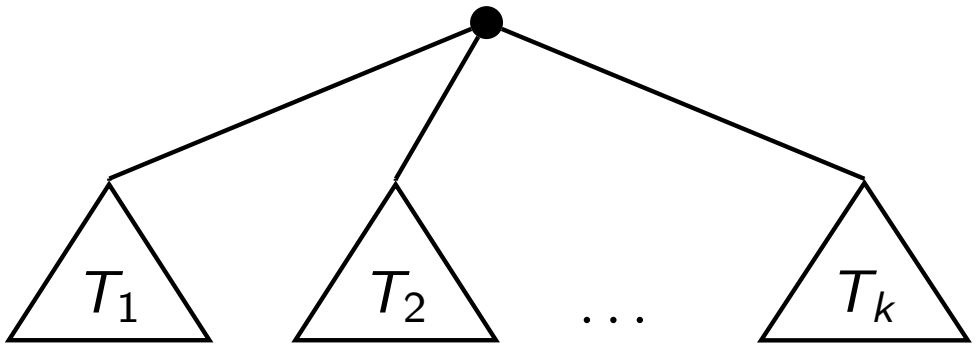
$$D(x) = \sum_{\alpha \in \mathcal{D}} x^{|\alpha|_{\mathrm{u}}} = 1 + \sum_{\beta,\gamma \in \mathcal{D}} x^{|\mathrm{u}\beta\mathrm{d}\gamma|_{\mathrm{u}}} = 1 + x \sum_{\beta \in \mathcal{D}} x^{|\beta|_{\mathrm{u}}} \sum_{\gamma \in \mathcal{D}} x^{|\gamma|_{\mathrm{u}}}$$
$$= 1 + xD^2(x)$$

Επομένως, η $D(x)$ ισούται με τη ΓΣ των αριθμών Catalan:

$$C(x) = \frac{1 - \sqrt{1-4x}}{2x} = \sum_{n \geq 0} C_n x^n, \qquad \text{όπου } C_n = \frac{1}{n+1}\binom{2n}{n}$$

Διασπάσεις - k-αδικά δένδρα

Ένα k-αδικό δένδρο T, $k \geq 2$, είναι είτε κενό και συμβολίζεται με $\square$, είτε ένας (εσωτερικός) κόμβος που καλείται ρίζα του δένδρου και έχει k διατεταγμένα παιδιά $T_1, T_2, \ldots, T_k$, που είναι k-αδικά δένδρα.



Σχήμα: Η διάσπαση ενός k-αδικού δένδρου $T = (T_1, T_2, \ldots, T_k)$.

Από την παραπάνω διάσπαση προκύπτει ότι η ΓΣ $T(x)$ ως προς το πλήθος των εσωτερικών κόμβων ικανοποιεί τη σχέση $T(x) = 1 + xT^k(x)$, άρα

$$[x^n]\,T(x) = C_n^{(k)} = \frac{1}{kn+1}\binom{kn+1}{n} = \frac{1}{(k-1)n+1}\binom{kn}{n}.$$

Θεώρημα (Chung-Feller)

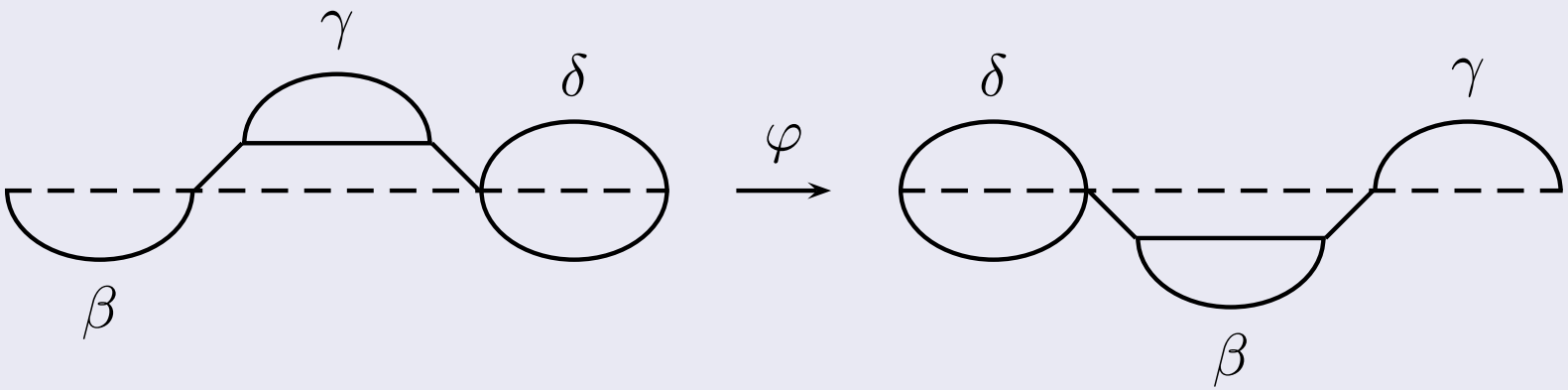
Ο πληθάριθμος του συνόλου $\mathcal{G}_{n,m}$ των μονοπατιών Grand-Dyck από το $(0,0)$ στο $(2n,0)$ με m βυθισμένες ανόδους, είναι ανεξάρτητο του m και ίσο με

$$|\mathcal{G}_{n,m}| = C_n = \frac{1}{n+1}\binom{2n}{n},$$

για κάθε $0 \leq m \leq n$.

Απόδειξη.

Ορίζουμε την απεικόνιση $\varphi : \mathcal{G} \setminus \overline{\mathcal{D}} \to \mathcal{G} \setminus \mathcal{D}$, όπου $\overline{\mathcal{D}}$ το σύνολο των αντεστραμμένων μονοπατιών Dyck ως προς τον άξονα x, όπως παρακάτω:



᾽μεσα προκύπτει ότι η φ είναι αμφιμονοσήμαντη, διατηρεί το μήκος και αυξάνει τις βυθισμένες ανόδους κατά 1. Επομένως, ο περιορισμός της στο $\mathcal{G}_{n,m}$, με $0 \leq m < n$, είναι αμφιμονοσήμαντη απεικόνιση από το $\mathcal{G}_{n,m}$ στο $\mathcal{G}_{n,m+1}$. Επομένως, $|\mathcal{G}_{n,m}| = |\mathcal{G}_{n,m+1}|$, για κάθε $0 \leq m < n$, και αφού $\sum_{m=0}^{n} |\mathcal{G}_{n,m}| = \binom{2n}{n}$, έπεται ότι $|\mathcal{G}_{n,m}| = \frac{1}{n+1}\binom{2n}{n}$, για κάθε $0 \leq m \leq n$. □

Η οικογένεια Catalan

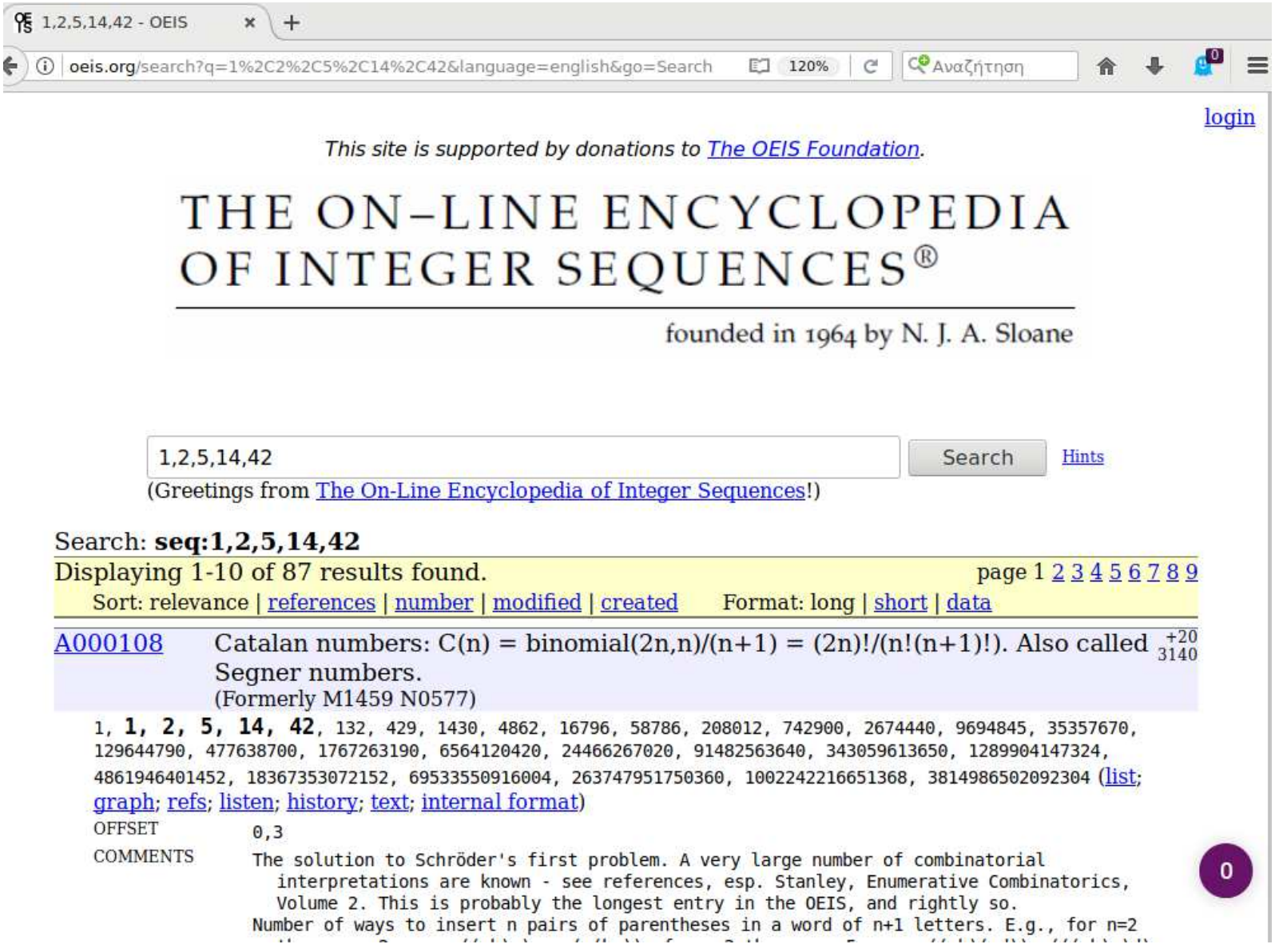
Η ακολουθία των αριθμών Catalan

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

είναι ίσως η πιο συχνά εμφανιζόμενη στη Συνδυαστική. Υπάρχουν πάνω από 200 διαφορετικά αντικείμενα που απαριθμούνται από αυτούς:

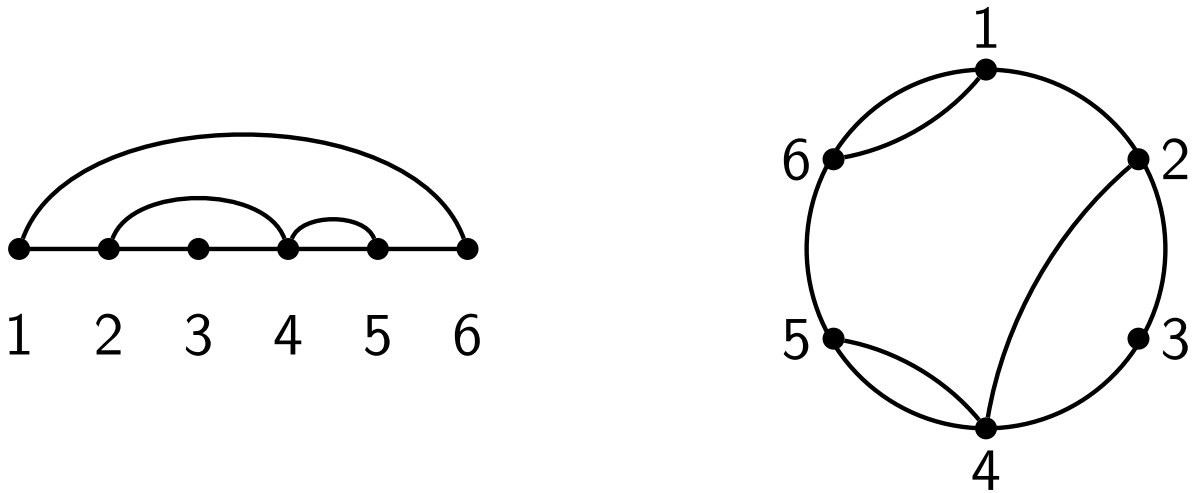
- The On-Line Encyclopedia of Integer Sequences, `http://oeis.org`
- R. P. Stanley, *Catalan Addendum*, `http://www-math.mit.edu/~rstan/ec/catadd.pdf`

OEIS


1,2,5,14,42 - OEIS
oeis.org/search?q=1%2C2%2C5%2C14%2C42&language=english&go=Search
120%
Αναζήτηση
login
This site is supported by donations to The OEIS Foundation.
THE ON–LINE ENCYCLOPEDIA OF INTEGER SEQUENCES®
founded in 1964 by N. J. A. Sloane
1,2,5,14,42
Search
Hints
(Greetings from The On-Line Encyclopedia of Integer Sequences!)
Search: seq:1,2,5,14,42
Displaying 1-10 of 87 results found.
page 1 2 3 4 5 6 7 8 9
Sort: relevance | references | number | modified | created
Format: long | short | data
A000108
Catalan numbers: C(n) = binomial(2n,n)/(n+1) = (2n)!/(n!(n+1)!). Also called Segner numbers.
(Formerly M1459 N0577)
+20 3140
1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, 18367353072152, 69533550916004, 263747951750360, 1002242216651368, 3814986502092304 (list; graph; refs; listen; history; text; internal format)
OFFSET
0,3
COMMENTS
The solution to Schröder's first problem. A very large number of combinatorial interpretations are known - see references, esp. Stanley, Enumerative Combinatorics, Volume 2. This is probably the longest entry in the OEIS, and rightly so.
Number of ways to insert n pairs of parentheses in a word of n+1 letters. E.g., for n=2
0

Μη τεμνόμενες διαμερίσεις (Noncrossing partitions)

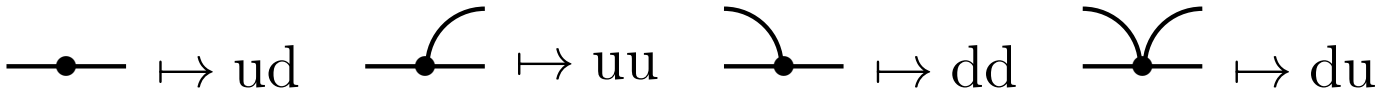
Μια **διαμέριση** $\pi = B_1/B_2/\cdots/B_k$ του συνόλου $[n] = \{1, 2, \ldots, n\}$ είναι μια συλλογή μη κενών, ανά δύο ξένων υποσυνόλων B_i του $[n]$, που καλούνται blocks (μέρη) της π, τέτοια ώστε $\bigcup_i \mathcal{B}_i = [n]$. Μια διαμέριση είναι **μη τεμνόμενη** (ΜΤΔ) ανν για οποιαδήποτε $a, b \in B_i$, $c, d \in B_j$, $i \neq j$, δεν ισχύει $a < c < b < d$.



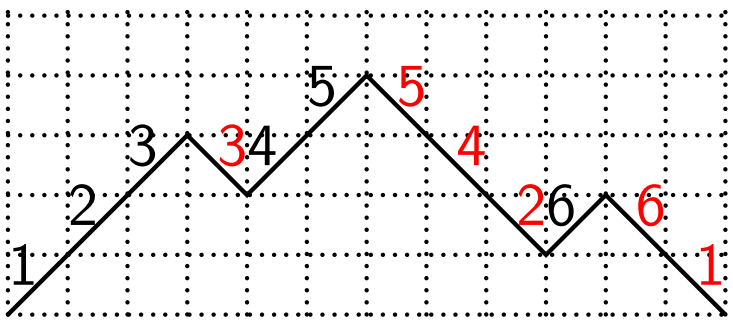
Σχήμα: Η ΜΤΔ $\pi = 1, 6/2, 4, 5/3$ (γραμμική και κυκλική αναπαράσταση).

Μη τεμνόμενες διαμερίσεις

Από ΜΤΔ σε μονοπάτια Dyck: Κάθε $i \in [n]$ αντιστοιχίζεται σε δύο γράμματα σύμφωνα με τους κανόνες:

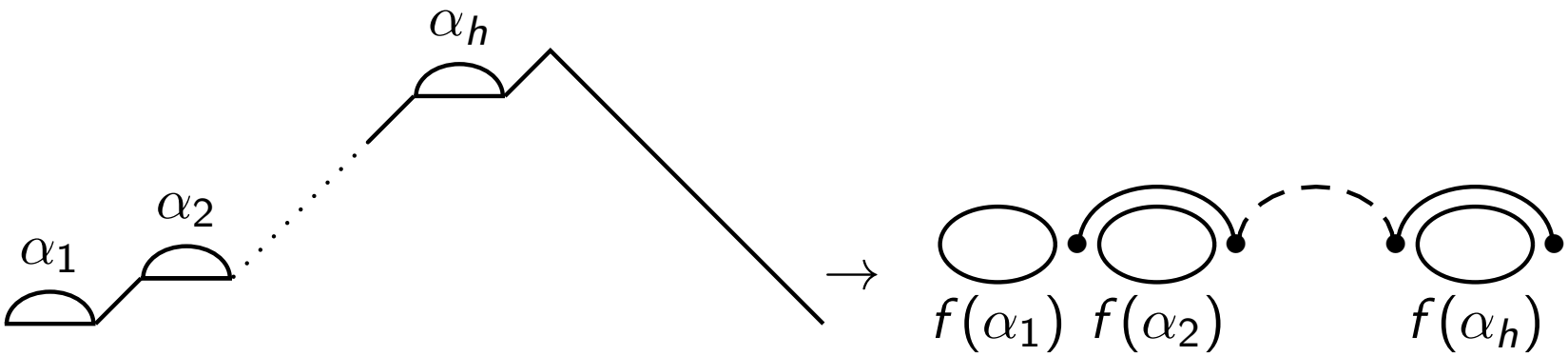


Από μονοπάτια Dyck σε ΜΤΔ: Αρίθμησε τα u του μονοπατιού με τους αριθμούς $1, 2, \ldots, n$ από αριστερά προς τα δεξιά και έπειτα αρίθμησε κάθε d όπως το συζυγές του u. Τελικά, κάθε μπλοκ αποτελείται από τους αριθμούς στην ίδια κατάβαση.



$\mapsto 1, 6/2, 4, 5/3$

Μη τεμνόμενες διαμερίσεις



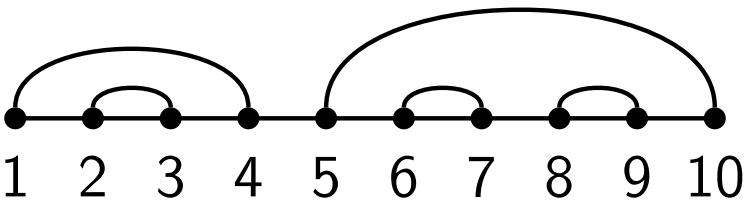
Σχήμα: Αναδρομικός ορισμός της προηγούμενης απεικόνισης.

Παρατήρηση: Η f απεικονίζει τα ud (κορυφές-peaks) σε μπλοκ. Ως γνωστό, το πλήθος των ΜΤΔ του $[n]$ με k μπλοκ ισούται με το πλήθος των μονοπατιών Dyck με $2n$ βήματα και k κορυφές και απαριθμείται από τους αριθμούς Narayana (A001263, OEIS):

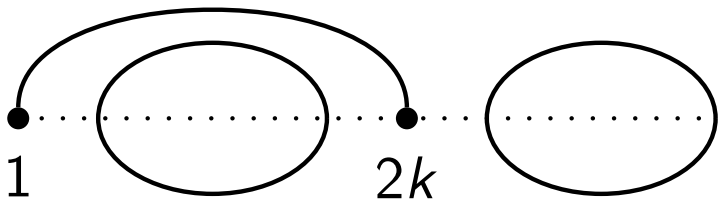
$$N_{n,k} = \frac{1}{n}\binom{n}{k}\binom{n}{k-1}, \qquad 0 < k \leq n.$$

Noncrossing matchings

Μια ΜΤΔ του $[2n]$ της οποίας κάθε μπλοκ έχει δύο στοιχεία ονομάζεται (complete) noncrossing matching.



$\pi = 1, 4/2, 3/5, 10/6, 7/8, 9$

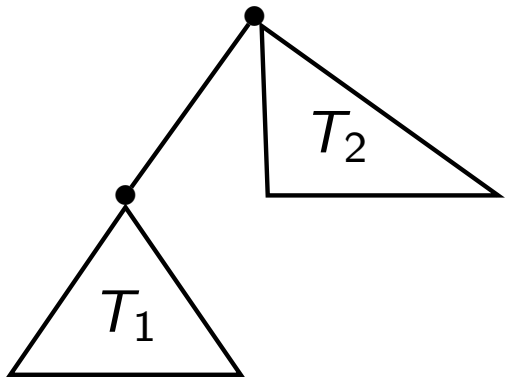


Διάσπαση

Η μετατροπή σε μονοπάτι Dyck είναι άμεση.

Διατεταγμένα δένδρα

Ένα διατεταγμένο δένδρο (ΔΔ) T αποτελείται από μια κορυφή, τη ρίζα r, η οποία έχει 0 ή περισσότερα παιδιά, διατεταγμένα από αριστερά προς τα δεξιά, τα οποία είναι επίσης διατεταγμένα δένδρα.

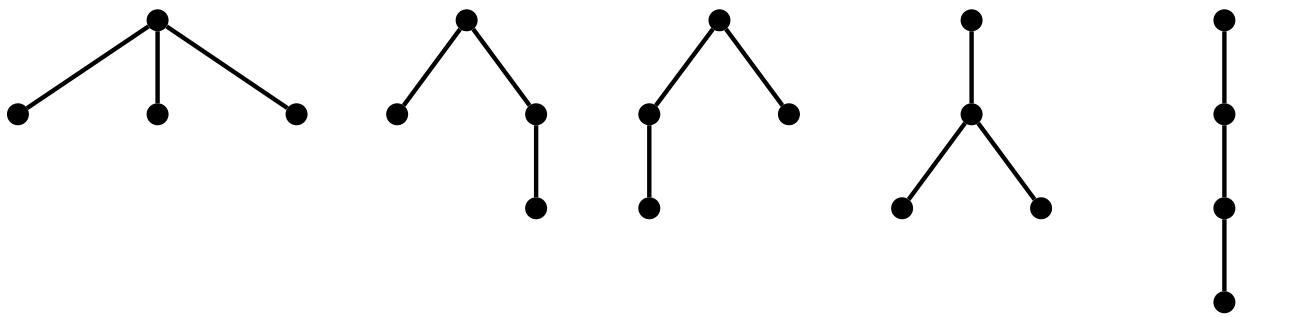


Σχήμα: Διάσπαση ενός ΔΔ με τουλάχιστον 2 κορυφές $T = \widehat{T_1\, T_2}$.

Μια γνωστή αμφιμονοσήμαντη απεικόνιση f μεταξύ μονοπατιών Dyck και ΔΔ ορίζεται αναδρομικά ως εξής:

$$f(r) = \varepsilon \qquad \text{και} \qquad f(\widehat{T_1\, T_2}) = \mathrm{u} f(T_1) \mathrm{d} f(T_2).$$

Η ίδια απεικόνιση περιγράφεται χωρίς αναδρομή ως εξής: Ξεκινώντας με ένα κενό μονοπάτι α, διασχίζουμε το ΔΔ σε προδιάταξη και εισάγουμε στο τέλος του α ένα u (αντ. d) όταν διασχίζουμε μια ακμή προς τα φύλλα (αντ. προς τη ρίζα).
Η απεικόνιση αυτή μετατρέπει τα φύλλα σε εμφανίσεις του ud, ώστε το πλήθος των ΔΔ με n ακμές ($n+1$ κορυφές) και k φύλλα ισούται με τον αριθμό Narayana $N_{n,k}$.

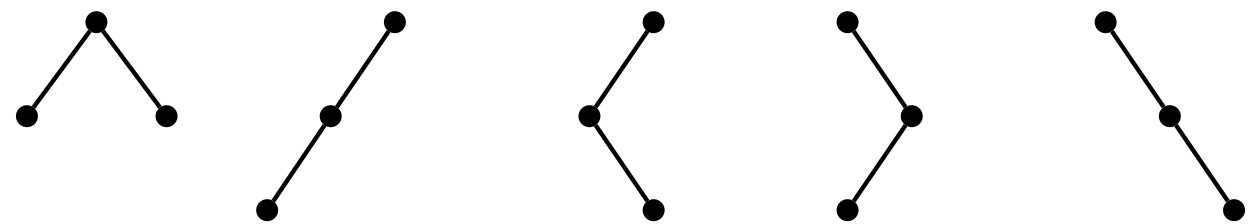


Δυαδικά δένδρα

Ειδική περίπτωση των k-αδικών δένδρων για $k = 2$. Μια γνωστή αμφιμονοσήμαντη απεικόνιση f από δυαδικά δένδρα σε μονοπάτια Dyck ορίζεται αναδρομικά ως εξής:

$$f(\square) = \varepsilon \qquad \text{και} \qquad f(T_1, T_2) = \mathrm{u} f(T_1) \mathrm{d} f(T_2).$$

Η μη αναδρομική περιγραφή έχει ως εξής: Ξεκινάμε με ένα κενό μονοπάτι α, διασχίζουμε το δένδρο σε προδιάταξη και βάζουμε στο τέλος του α ένα u (αντ. d) όταν συναντάμε αριστερό (αντ. δεξί) παιδί, λαμβάνοντας υπόψη και τα κενά παιδιά.



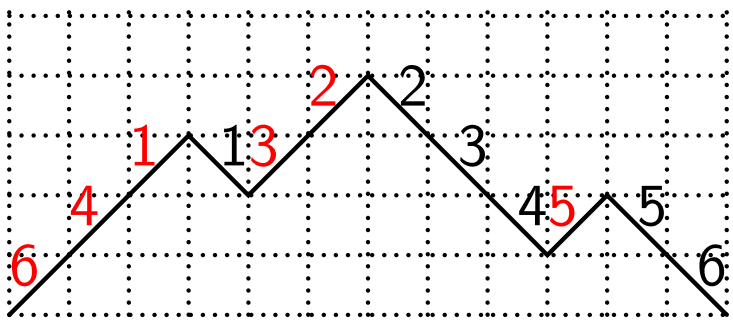
Μεταθέσεις που αποφεύγουν το 231

Μια μετάθεση σ του $[n]$ αποφεύγει το πρότυπο 231 ανν δεν υπάρχουν δείκτες $i < j < k$ τέτοιοι ώστε η λέξη $\sigma_i\sigma_j\sigma_k$ να έχει ίδια διάταξη με την 231. Γενικά, με $\mathbb{S}_n(\tau)$ συμβολίζουμε το σύνολο των μεταθέσεων του $[n]$ που αποφεύγουν το πρότυπο $\tau \in [n]^*$, δηλαδή δεν περιέχουν υπακολουθία ισόμορφη με την τ ως προς τη διάταξη των φυσικών. Είναι γνωστό ότι $|\mathbb{S}_n(\tau)| = C_n$, για κάθε πρότυπο τ μήκους 3. Ειδικά, οι μεταθέσεις που αποφεύγουν το 231 ονομάζονται και stack sortable permutations, γιατί μπορούν να ταξινομηθούν (σε χρόνο $O(n)$) με μια στοίβα ως εξής:

Η στοίβα είναι αρχικά κενή. Διάβασε τα $\sigma(1), \sigma(2), \ldots$ και για κάθε στοιχείο x που διαβάστηκε, αφαίρεσε (pop) την κορυφή της στοίβας όσο είναι μικρότερη του x, και έπειτα πρόσθεσε (push) το x. Τέλος, αφαίρεσε όσα στοιχεία απέμειναν στη στοίβα.

Μεταθέσεις που αποφεύγουν το 231

Για τη μετατροπή ενός μονοπατιού Dyck σε μετάθεση του $\mathbb{S}(231)$, αρίθμησε τα d από αριστερά προς τα δεξιά και έπειτα αρίθμησε τα συζυγή u με το ίδιο γράμμα και διάβασέ τα από αριστερά προς τα δεξιά.



$\mapsto 6\ 4\ 1\ 3\ 2\ 5$

Μεταθέσεις που αποφεύγουν το 231

Για τον αναδρομικό ορισμό της τελευταίας απεικόνισης, αρχικά παρατηρούμε ότι κάθε $\sigma \in S_n(231)$ διασπάται ως

$$\sigma = \tau n \rho.$$

Δεδομένου ότι τα στοιχεία του τ πρέπει να είναι μικρότερα από αυτά του ρ, συμπεραίνουμε ότι η τ είναι μετάθεση του $[k]$, για κάποιο $k \geq 0$, ενώ η ρ είναι μετάθεση του $k + [n-k-1] := \{k+1, \ldots, n-1\}$, οι οποίες επίσης αποφεύγουν το 231.
Επομένως η f που ορίζεται ως

$$f(\varepsilon) = \varepsilon \qquad \text{και} \qquad f(\tau n \rho) = f(\tau)\mathrm{u} f(\rho)\mathrm{d}$$

είναι αμφιμονοσήμαντη και στέλνει το n στο τελευταίο d και το συζυγές του u, άρα έχει το ίδιο αποτέλεσμα με την προηγούμενη διαδικασία της αρίθμησης.

Κάθε λέξη $\alpha = a_1 a_2 \cdots a_{2n} \in \mathcal{D}_n \setminus \{\mathrm{u}^n \mathrm{d}^n\}$ γράφεται ως

$$\alpha = a_1 \cdots a_k \mathrm{d} \underbrace{\mathrm{u} \cdots \mathrm{u}}_{p} \underbrace{\mathrm{d} \cdots \mathrm{d}}_{q}$$

για κάποια $p, q > 0$. Η λεξικογραφικά επόμενη της α, είναι η

$$a_1 \cdots a_k \mathrm{u} \underbrace{\mathrm{d} \cdots \mathrm{d}}_{q-p+2} \underbrace{\mathrm{ud} \cdots \mathrm{ud}}_{p-1}$$

Προφανώς, είναι $p \leq q$, επομένως η θέση $k+1$ εντοπίζεται σαρώνοντας την α από το τέλος προς την αρχή μετά από $p + q + 1 = O(q)$ ελέγχους. Στη συνέχεια, η επόμενη λέξη κατασκευάζεται μετά από $O(q)$ αντικαταστάσεις.
Επομένως, ο αριθμός $q = q(\alpha)$, ο οποίος αντιστοιχεί στο μήκος της τελευταίας κατάβασης της α, καθορίζει τον χρόνο που απαιτείται για την κατασκευή της επόμενης λέξης.

Αλγόριθμοι κατασκευής λέξεων Dyck

```
1  DyckLex() begin                                   /* a_0 = d */
2      k := 2n;
3      while a_k = d do k := k − 1;
4      q := 2n − k;
5      while a_k = u do k := k − 1;
6      if k = 0 then return false;
7      p := 2n − k − q;
8      a_k := u;
9      for j := k + 1 to k + q − p + 2 do a_j := d;
10     j := 2n − 2p + 3;
11     while j ≤ 2n − 1 do
12         (a_j, a_{j+1}) := (u, d);
13         j := j + 2;
14     return true;
```

Αλγόριθμος 27: Αλγόριθμος κατασκευής του $\mathcal{D}_n$ σε λεξικογραφική διάταξη.

Για την ανάλυση του αλγορίθμου, θεωρούμε την παράμετρο $q(\alpha) =$ μήκος της τελευταίας κατάβασης του $\alpha \in \mathcal{D}$ και τη γεννήτρια συνάρτηση $F = F(x, y) = \sum_{\alpha \in \mathcal{D}} x^{|\alpha|_{\mathrm{u}}} y^{q(\alpha)}$. Από τη διάσπαση της τελευταίας επιστροφής

$$\alpha \in \mathcal{D} \setminus \{\varepsilon\} \Rightarrow \alpha = \beta \mathrm{u} \gamma \mathrm{d}, \qquad \beta, \gamma \in \mathcal{D}$$

προκύπτει ότι $|\alpha|_{\mathrm{u}} = |\beta|_{\mathrm{u}} + |\gamma|_{\mathrm{u}} + 1$ και $q(\alpha) = q(\gamma) + 1$, επομένως

$$F = 1 + \sum_{\beta, \gamma \in \mathcal{D}} x^{|\beta|_{\mathrm{u}} + |\gamma|_{\mathrm{u}} + 1} y^{q(\gamma)+1} = 1 + xyC(x)F.$$

Παραγωγίζοντας ως προς y, έχουμε $F_y = xC(x)F + xyC(x)F_y$, επομένως $F_y(x, 1) = \frac{xC^2(x)}{1 - xC(x)} = xC^3(x)$. 'ρα,

$$E_n(q) = \frac{1}{C_n}[x^{n-1}]C^3 = \frac{1}{C_n} \frac{3}{2n+1} \binom{2n+1}{n-1} = \frac{3n}{n+2} < 3,$$

δηλαδή ο αλγόριθμος είναι CAT.

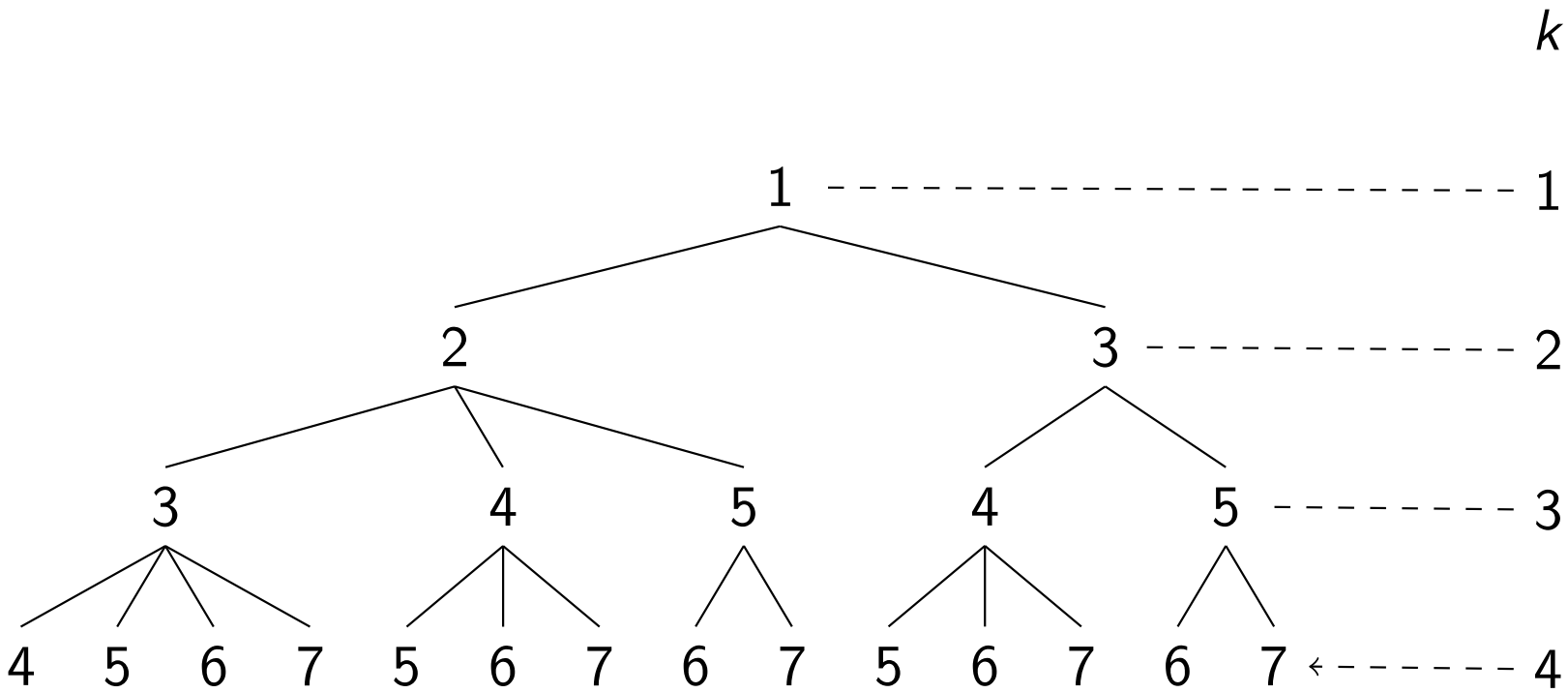
Αλγόριθμοι κατασκευής λέξεων Dyck

Κωδικοποιούμε κάθε λέξη Dyck $\alpha \in \mathcal{D}_n$ από μια ακολουθία $(e_1, e_2, \ldots, e_n)$, όπου e_i η θέση του i-οστού u στην α. Προφανώς η λεξικογραφική διάταξη αυτών των ακολουθιών αντιστοιχεί σε αντίστροφη λεξικογραφική των λέξεων Dyck, όταν $\mathrm{d} < \mathrm{u}$.
Επιπλέον, η ακολουθία είναι γνησίως αύξουσα και ικανοποιεί τη σχέση $i \leq e_i \leq 2i - 1$, για κάθε $i \in [n]$. Συμβολίζουμε με E_n το σύνολο αυτών των ακολουθιών μήκους n. Ο επόμενος αναδρομικός αλγόριθμος τις κατασκευάζει λεξικογραφικά.

```
1 DyckRec(k) begin                                   /* e_0 = 0 */
2     if k = n + 1 then output(e);
3     else
4         for j := e_{k-1} + 1 to 2k - 1 do
5             e_k := j;
6             DyckRec (k + 1);
```

Αλγόριθμος 28: Αναδρομικός αλγόριθμος κατασκευής του E_n σε λεξικογραφική διάταξη. (Αρχική κλήση: DyckRec(1))

Παρακάτω φαίνεται το δένδρο αναδρομής του αλγορίθμου 28, για $n = 4$. Το πλήθος κόμβων στο επίπεδο k είναι C_k, οπότε το συνολικό πλήθος κόμβων είναι $T(n) = \sum_{k=1}^{n} C_k$



Θα δείξουμε ότι $\lim_{n\to\infty} \frac{T(n)}{C_n} = 4/3$, οπότε ο αλγόριθμος είναι CAT.
Με τη βοήθεια της σχέσης $C_{n+1} = \frac{4n+2}{n+2} C_n$ (σημειώνεται ότι μέσω της σχέσης αυτής υπολογίζονται επαναληπτικά οι αριθμοί Catalan με τον πιο αποδοτικό τρόπο) βρίσκουμε ότι

$$\frac{T(n+1) - T(n)}{C_{n+1} - C_n} = \frac{C_{n+1}}{C_{n+1} - C_n} = \frac{(4n+2)/(n+2)}{(4n+2)/(n+2) - 1}$$
$$= \frac{(4n+2)}{3n} \to 4/3$$

Επομένως, από την Πρόταση Stolz έχουμε το ζητούμενο.

```
1  DyckRelex() begin                                   /* e_{n+1} = 2n + 1 */
2      k := n;
3      while k ≥ 1 do
4          if e_k < 2k − 1 and e_k < e_{k+1} − 1 then break;
5          k := k − 1;
6      if k = 0 then return false;
7      e_k := e_k + 1;
8      while k < n do
9          e_{k+1} := e_k + 1;
10         k := k + 1;
11     return true;
```

Αλγόριθμος 29: Αλγόριθμος κατασκευής του E_n σε λεξικογραφική διάταξη.

Αλγόριθμοι κατασκευής λέξεων Dyck

Για την ανάλυση του τελευταίου αλγορίθμου αρκεί να παρατηρήσουμε ότι κάθε $\alpha \in \mathcal{D}_n \setminus \{(\mathrm{ud})^n\}$ γράφεται ως $\beta\mathrm{dd}(\mathrm{ud})^{n-k}$, για κάποια λέξη β και $2 \le k \le n$, και ότι το while στη γραμμή 3 (και αυτό στη γραμμή 8) θα ελέγξει $n-k+1$ θέσεις, μέχρι να εντοπίσει το στοιχείο e_k που θα αλλάξει. Επομένως, ορίζουμε την παράμετρο $q(a) =$ ημιμήκος επιθέματος μετά το τελευταίο dd και τη ΓΣ $F = F(x,y) = \sum_{\alpha\in\mathcal{D}} x^{|\alpha|_\mathrm{u}} y^{q(a)}$. Βάσει της διάσπασης

$$\alpha \in \mathcal{D}^* \Rightarrow \alpha = \beta\mathrm{ud} \text{ ή } \alpha = \beta\mathrm{u}\gamma\mathrm{d}, \qquad \beta \in \mathcal{D}, \gamma \in \mathcal{D}^*,$$

βρίσκουμε ότι $F = 1 + xyF + xC(x)(C(x)-1) = \frac{1-x}{1-xy}C(x)$, οπότε $F_y = xF + xyF_y = xF/(1-xy)$. Επομένως,

$$E_n(q) = \frac{1}{C_n}[x^n]F_y(x,1) = \frac{1}{C_n}[x^n]\frac{x}{1-x}C(x) = \frac{1}{C_n}\sum_{k=0}^{n-1} C_k \to 1/3$$

Το τελευταίο όριο προκύπτει βάσει του $\lim \frac{1}{C_n}\sum_{k=1}^{n} C_k = 4/3$, που υπολογίστηκε νωρίτερα. Επομένως ο αλγόριθμος είναι CAT.

Αλγόριθμος ranking λέξεων Dyck

Έστω $\alpha \in \mathcal{D}_n$, $n \in \mathbb{N}^*$. Για να υπολογίσουμε το $\mathrm{rank}(\alpha)$, σκεφτόμαστε ως εξής: Αν $\alpha = p\mathrm{u}q$, για κάποιες λέξεις $p, q \in \{\mathrm{u}, \mathrm{d}\}^*$, τότε η α είναι μεγαλύτερη (λεξικογραφικά) από κάθε $\beta \in \mathcal{D}_n$, με $\beta = p\mathrm{d}s$, $s \in \{\mathrm{u}, \mathrm{d}\}^*$. Δεδομένου ότι το rank της α είναι ίσο με το πλήθος των λέξεων που είναι μικρότερες της α, θα ισχύει

$$\mathrm{rank}(\alpha) = \sum_{\substack{p \in \{\mathrm{u},\mathrm{d}\}^* \\ \alpha = p\mathrm{u}q}} |\{\beta \in \mathcal{D}_n : \beta \text{ έχει πρόθεμα } p\mathrm{d}\}|.$$

Πράγματι, το σύνολο των λέξεων που είναι μικρότερες της α διαμερίζεται με βάση το μέγιστο κοινό πρόθεμά τους p με την α. Έτσι, κάθε τέτοια λέξη υπολογίζεται ακριβώς μία φορά στο παραπάνω άθροισμα. Επομένως, το πρόβλημα υπολογισμού του $\mathrm{rank}(\alpha)$ ανάγεται στην απαρίθμηση των δυνατών επιθεμάτων s, με $p\mathrm{d}s \in \mathcal{D}_n$, για κάθε πρόθεμα $p\mathrm{u}$ του α.

Αλγόριθμος ranking λέξεων Dyck

Συμβολίζουμε με $s(i,j)$ το πλήθος των δυνατών επιθεμάτων Dyck s με $|s|_{\mathrm{u}} = i$ και $|s|_{\mathrm{d}} = j$. Κάθε τέτοιο επίθεμα ξεκινά με d ή με u, και ακολουθείται από ένα μικρότερο σε μήκος επίθεμα Dyck, οπότε προκύπτει η αναδρομική σχέση

$$s(i,j) = \begin{cases} 0, & j < i \text{ ή } j \leq 0 \text{ ή } i < 0 \\ 1, & i = 0 \\ s(i-1,j) + s(i,j-1), & 0 < i \leq j \end{cases}$$

Οι τιμές $s(i,j)$, $0 \leq i \leq j \leq n$, μπορούν να υπολογιστούν μια φορά πριν από την εκτέλεση του αλγορίθμου ranking. Ο υπολογισμός έχει πολυπλοκότητα χρόνου και χώρου $O(n^2)$.
Σημειώνεται ότι, εφαρμόζοντας κατάλληλη διάσπαση στα επιθέματα Dyck, προκύπτει ότι

$$s(i,j) = [x^i]C^{j-i+1}(x) = \frac{j-i+1}{j+i+1}\binom{j+i+1}{i} = \frac{j-i+1}{j+1}\binom{i+j}{j}.$$

Αλγόριθμος ranking λέξεων Dyck

Οι τιμές $s(i,j)$, όπως προκύπτουν από την εφαρμογή της παραπάνω αναδρομικής σχέσης δίνονται στον επόμενο πίνακα.

$j \setminus i$	0	1	2	3	4	5	6	$\cdots$
0	1							
1	1	1						
2	1	2	2					
3	1	3	5	5				
4	1	4	9	14	14			
5	1	5	14	28	42	42		
6	1	6	20	48	90	132	132	
$\vdots$								$\ddots$

Ο προηγούμενος πίνακας συναντάται στη βιβλιογραφία με την ονομασία *τρίγωνο Catalan*. Οι τιμές της διαγωνίου είναι οι όροι της ακολουθίας Catalan, δηλαδή $s(i,i) = C_i$.

Αλγόριθμος ranking λέξεων Dyck

Κατόπιν τούτων, για $\alpha \in \mathcal{D}_n$, είναι

$$\operatorname{rank}(\alpha) = \sum_{\substack{p \in \{\mathrm{u},\mathrm{d}\}^* \\ \alpha = p\mathrm{u}q}} s(n - |p|_{\mathrm{u}}, n - |p|_{\mathrm{d}} - 1). \tag{5}$$

```
1  rank(α = α_1 α_2 ··· α_2n) begin
2      r := 0;
3      ups := 0;
4      for i := 1 to 2n − 1 do
5          if α_i = u then
6              r := r + s(n − ups, n + ups − i);
7              ups := ups + 1;
8      return r;
```

Αλγόριθμος 30: Αλγόριθμος ranking για λέξεις Dyck.

Αλγόριθμος unranking λέξεων Dyck

```
1  unrank(r, n) begin
2      ups := 0;
3      for i := 1 to 2n − 1 do
4          if r ≥ s(n − ups, n + ups − i) then
5              α_i := u;
6              r := r − s(n − ups, n + ups − i);
7              ups := ups + 1;
8          else
9              α_i := d;
10     α_2n := d;
```

Αλγόριθμος 31: Αλγόριθμος unranking για λέξεις Dyck.

Η οικογένεια Motzkin

Ο n-οστός αριθμός Motzkin M_n ισούται με

$$M_n = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} C_k$$

Η ΓΣ $M(x) = \sum_{n \geq 0} M_n x^n$ των αριθμών Motzkin ικανοποιεί τη σχέση

$$M(x) = 1 + xM(x) + x^2 M^2(x)$$

και ισούται με $M(x) = \dfrac{1 - x - \sqrt{(1-x)^2 - 4x^2}}{2x^2}$.

Σχετίζεται με τη ΓΣ των αριθμών Catalan μέσω των σχέσεων

$$M(x) = \frac{1}{1-x} C\left(\frac{x^2}{(1-x)^2}\right), \qquad C(x) = 1 + \frac{x}{1-x} M\left(\frac{x}{1-x}\right).$$

Πράγματι, βάσει των σχέσεων

$$C(x) = 1 + xC^2(x) \qquad (*)$$

και

$$M(x) = 1 + xM(x) + x^2M^2(x) \qquad (**)$$

έχουμε ότι

$$A(x) := (1-x)M(x) \overset{(**)}{=} 1 + x^2M^2(x) = 1 + \frac{x^2}{(x-1)^2}A^2(x)$$

Επομένως $A(x) \overset{(*)}{=} C\left(\frac{x^2}{(1-x)^2}\right)$, δηλαδή

$$M(x) = \frac{1}{1-x}C\left(\frac{x^2}{(1-x)^2}\right).$$

Ομοίως προκύπτει και η σχέση

$$C(x) = 1 + \frac{x}{1-x}M\left(\frac{x}{1-x}\right).$$

Η οικογένεια Motzkin

Ο συντελεστής $M_n = [x^n]M(x)$ μπορεί να υπολογισθεί από τον τύπο αντιστροφής του Lagrange. Εναλλακτικά, βάσει της σχέσης

$$M(x) = \frac{1}{1-x} C\left(\frac{x^2}{(1-x)^2}\right),$$

έχουμε ότι

$$M(x) = \sum_{k\geq 0} C_k \frac{x^{2k}}{(1-x)^{2k+1}} = \sum_{k\geq 0} C_k \sum_{n\geq 0} \binom{n}{2k} x^n = \sum_{n\geq 0} \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} C_k x^n$$

και άρα

$$M_n = [x^n]M(x) = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} C_k.$$

Ωστόσο, ο υπολογισμός του M_n μπορεί να γίνει πιο αποδοτικά χρησιμοποιώντας την επόμενη αναγωγική σχέση.

Αναγωγική σχέση των αριθμών Motzkin

Έστω $F(x) = \frac{1-x}{2x^2} - M(x) = \frac{\sqrt{1-2x-3x^2}}{2x^2}$. Παρατηρούμε ότι για κάθε $n \geq 0$ ισχύει ότι $[x^n]F(x) = -M_n$. Επίσης,

$$\ln F(x) = \ln \sqrt{1-2x-3x^2} - \ln 2x^2 = \frac{1}{2}\ln(1-2x-3x^2) - \ln 2 - 2\ln x.$$

Παραγωγίζοντας ως προς x έχουμε ότι

$$\frac{F'(x)}{F(x)} = \frac{1}{2}\frac{(1-2x-3x^2)'}{(1-2x-3x^2)} - \frac{2}{x} = \frac{3x^2+3x-2}{x-2x^2-3x^3}.$$

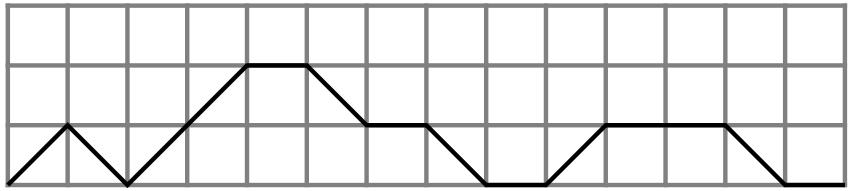
Επομένως, $(x-2x^2-3x^3)F'(x) = (3x^2+3x-2)F(x)$. Βάσει των σχέσεων $[x^n]F'(x) = (n+1)[x^{n+1}]F(x) = -(n+1)M_{n+1}$ και $[x^n]x^kF(x) = [x^{n-k}]F(x) = -M_{n-k}$, έχουμε τελικά ότι

$$nM_n - 2(n-1)M_{n-1} - 3(n-2)M_{n-2} = 3M_{n-2} + 3M_{n-1} - 2M_n$$

ή, ισοδύναμα, ότι

$$(n+2)M_n = (2n+1)M_{n-1} + (3n-3)M_{n-2}, \qquad n \geq 2$$

Μονοπάτια Motzkin



Ένα μονοπάτι Motzkin περιέχει βήματα $\mathrm{u} = (1, 1)$, $\mathrm{d} = (1, -1)$ καθώς και οριζόντια βήματα $\mathrm{h} = (1, 0)$, η αφαίρεση των οποίων δίνει ένα μονοπάτι Dyck.
Συμβολίζουμε με $\mathcal{M}$ το σύνολο των μονοπατιών Motzkin. Ένα μονοπάτι $\alpha \in \mathcal{M}$ διασπάται ως εξής:

$$\alpha = \varepsilon, \quad \text{ή} \quad \alpha = \mathrm{h}\beta, \quad \text{ή} \quad \alpha = \mathrm{u}\beta\mathrm{d}\gamma, \qquad \beta, \gamma \in \mathcal{M}.$$

Από τη διάσπαση αυτή προκύπτει άμεσα ότι η ΓΣ του $\mathcal{M}$ ως προς το μήκος ικανοποιεί τη σχέση

$$M(x) = 1 + xM(x) + x^2M^2(x)$$

άρα ταυτίζεται με τη ΓΣ των αριθμών Motzkin, δηλαδή $|\mathcal{M}_n| = M_n$.

Μπορούμε να κατασκευάσουμε το σύνολο $\mathcal{M}$ των μονοπατιών Motzkin από τα στοιχεία του $\mathcal{D}$ ως εξής:

Έστω $\alpha \in \mathcal{D}$. Εισάγοντας μια (ενδεχομένως κενή) ακολουθία από διαδοχικά h αμέσως μετά από κάθε βήμα του α καθώς και πριν το πρώτο βήμα του α, παίρνουμε ένα μονοπάτι Motzkin β.

Αφού έχουμε άπειρες επιλογές για αυτές τις ακολουθίες από διαδοχικά h (οι οποίες καλούνται plateaux), κάθε $\alpha \in \mathcal{D}$ παράγει ένα άπειρο σύνολο $f(\alpha)$ μονοπατιών Motzkin.
Για την ακρίβεια, αυτή η κατασκευή είναι μια απεικόνιση $f : \mathcal{D} \to 2^{\mathcal{M}}$, τέτοια ώστε η οικογένεια $(f(\alpha))_{\alpha \in \mathcal{D}}$ είναι διαμέριση του $\mathcal{M}$.

Τέτοιες απεικονίσεις μεταφράζονται σε συνθέσεις των αντίστοιχων ΓΣ. Στη συγκεκριμένη περίπτωση, αυτή η κατασκευή δίνει τη σχέση

$$M(x) = \frac{1}{1-x} C\left(\frac{x^2}{(1-x)^2}\right)$$

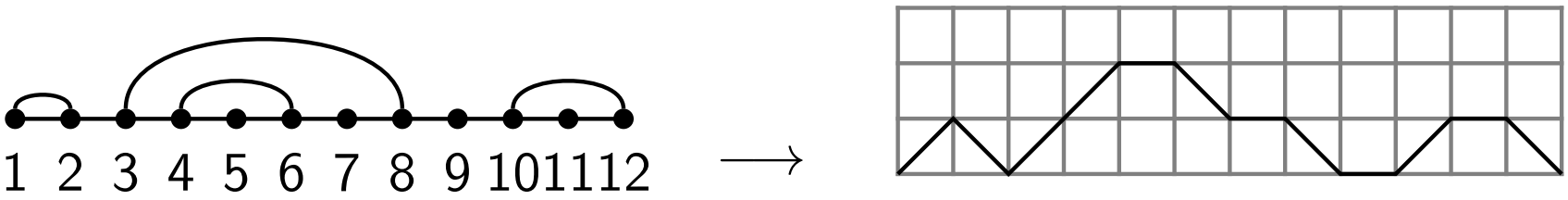
Πράγματι, η ΓΣ των μονοπατιών Dyck ως προς το μήκος τους είναι η $C(x^2)$. Αφού η ΓΣ για το σύνολο

$$\{h^n : n \in \mathbb{N}\} = \{\varepsilon, h, h^2, h^3, \ldots\}$$

των plateux είναι η $\frac{1}{1-x}$, η αντικατάσταση $x \mapsto \frac{x}{1-x}$ στην $C(x^2)$ αντιστοιχεί στην εισαγωγή ενός plateau μετά από κάθε βήμα, ενώ ο παράγοντας $\frac{1}{1-x}$ αντιστοιχεί στην εισαγωγή ενός plateau στην αρχή του μονοπατιού.

$\{1,2\}$-μη τεμνόμενες διαμερίσεις

Οι ΜΤΔ του $[n]$ που αποτελούνται μόνο από μπλοκ μεγέθους 1 ή 2 απαριθμούνται από τους αριθμούς Motzkin.



Τα αντικείμενα αυτά κωδικοποιούν τη δευτεροταγή δομή του RNA: D. Doslic, D. Svrtan, D. Veljan, Enumerative aspects of secondary structures, *Discrete Math.* **285** (2004), 67–82.
Οι κόμβοι στον οριζόντιο άξονα αντιστοιχούν στην ακολουθία των βάσεων A, C, G, U οι οποίες ενώνονται με p-δεσμούς και τα τόξα αντιστοιχούν στους h-δεσμούς. Οι h-δεσμοί δεν μπορούν να ενώσουν δύο βάσεις i, j που είναι πολύ κοντινές, οπότε απαιτούμε να ισχύει $|i-j|>\ell$, για κάποια τιμή ℓ που καλείται τάξη της δομής. Επιπλέον, οι h-δεσμοί μπορεί να αντιστοιχούν σε τεμνόμενα τόξα όταν σχεδιαστούν στο επίπεδο, αλλά τα τόξα αυτά θεωρείται ότι ανήκουν στην τριτοταγή δομή.

Μονοπάτια 2-Motzkin

Ένα μονοπάτι 2-Motzkin είναι ένα μονοπάτι Motzkin του οποίου κάθε οριζόντιο βήμα έχει χρώμα από ένα σύνολο 2 δυνατών χρωμάτων.
Το πλήθος των μονοπατιών 2-Motzkin με n βήματα ισούται με C_{n+1}.
Το σύνολο αυτών συμβολίζεται με $\mathcal{M}^{(2)}$
Μια απλή απεικόνιση που μετατρέπει ένα τέτοιο μονοπάτι σε μονοπάτι Dyck είναι η ακόλουθη:

Αντικατάστησε κάθε u με uu, κάθε d με dd, κάθε h_1 με ud και κάθε h_2 με du. Τέλος, πρόσθεσε ένα u στην αρχή και ένα d στο τέλος του μονοπατιού.

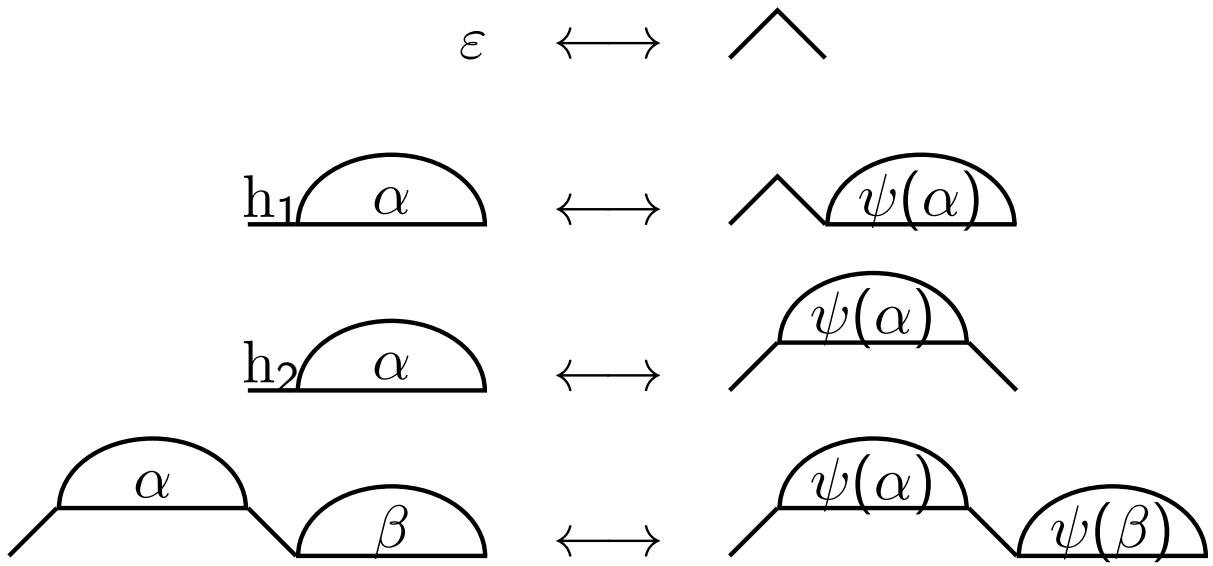
Κατόπιν τούτων, η ΓΣ του $\mathcal{M}^{(2)}$ ως προς το μήκος είναι η

$$C^2(x) = \frac{C(x)-1}{x} = \frac{1}{x}\sum_{n\geq 1} C_n x^n = \frac{1}{x}\sum_{n\geq 0} C_{n+1} x^{n+1} = \sum_{n\geq 0} C_{n+1} x^n.$$

Μονοπάτια 2-Motzkin

Μια διαφορετική αμφιμονοσήμαντη απεικόνιση, η οποία ορίζεται αναδρομικά, είναι η ακόλουθη:

$$\mu \in \mathcal{M}^{(2)} \quad \longleftrightarrow \quad \psi(\mu) \in \mathcal{D}^*$$



Σχήμα: Η απεικόνιση $\psi : \mathcal{M}^{(2)} \to \mathcal{D}^*$.

Η απεικόνιση ψ έχει διάφορες ενδιαφέρουσες ιδιότητες. Μια από αυτές είναι ότι αντιστοιχεί τα h_1 σε udu, επομένως τα μονοπάτια Dyck ημιμήκους $n+1$ χωρίς udu είναι σε πλήθος M_n.