ΙΩΑΝΝΑ Α. ΑΡΣΕΝΟΠΟΥΛΟΥ

ΑΠΑΡΙΘΜΗΣΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΣΥΝΟΛΩΝ ΔΥΑΔΙΚΩΝ ΛΕΞΕΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

ΠΜΣ ''ΠΡΟΗΓΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΠΛΗΡΟΦΟΡΙΚΗΣ''
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΕΠΤΕΜΒΡΙΟΣ 2010

ΙΩΑΝΝΑ Α. ΑΡΣΕΝΟΠΟΥΛΟΥ

ΑΠΑΡΙΘΜΗΣΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΣΥΝΟΛΩΝ ΔΥΑΔΙΚΩΝ ΛΕΞΕΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

Επιβλέπων: Π. Τσικούρας
Καθηγητής Πανεπιστημίου Πειραιώς

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ

Πρόλογος

Η παρούσα μεταπτυχιακή διατριβή εκπονήθηκε στα πλαίσια του Μεταπτυχιακού Προγράμματος Σπουδών του Τμήματος Πληροφορικής ''Προηγμένα Συστήματα Πληροφορικής'', του Πανεπιστημίου Πειραιώς. Στόχος αυτής της διπλωματικής είναι η μελέτη απαρίθμησης συνόλων δυαδικών λέξεων με ορισμένους περιορισμούς ή ιδιότητες, που εδώ και αρκετά χρόνια έχει κερδίσει το ενδιαφέρον πολλών ερευνητών. Τη μελέτη αυτή έρχεται να συμπληρώσει μία εφαρμογή σε ηλεκτρονικό υπολογιστή, η οποία υλοποιεί τους αλγορίθμους που περιγράφονται στην εργασία.

Σε αυτό το σημείο, επιθυμώ να ευχαριστήσω θερμά τον επιβλέποντα της εργασίας μου, Καθηγητή Παναγιώτη Τσικούρα, που μου έδωσε την ευκαιρία να εκπονήσω την παρούσα διατριβή, καθώς και για την καθοδήγηση και την βοήθεια που μου πρόσφερε καθ' όλη τη διάρκεια της προσπάθειας αυτής. Επίσης, ευχαριστώ τα μέλη της τριμελούς επιτροπής Καθηγητές Αριστείδη Σαπουνάκη και Ευάγγελο Φούντα. Τέλος, θέλω να ευχαριστήσω τον διδάκτορα Γιάννη Τασούλα και τον υποψήφιο διδάκτορα Κώστα Μανέ για όλα όσα με διδάξανε, για το επιστημονικό υλικό που μου προσφέρανε, τη συμπαράστασή τους και τις ώρες που μου αφιερώσανε.

Τέλος, θέλω να ευχαριστήσω την οικογένειά μου για την οικονομική και όχι μόνο στήριξη όλα αυτά τα χρόνια που ζω και σπουδάζω στην Αθήνα.

Περίληψη

Η διπλωματική αυτή ασχολείται με την απαρίθμηση και κατασκευή συνόλων δυαδικών λέξεων με ορισμένους περιορισμούς ή ιδιότητες.

Στο πρώτο κεφάλαιο δίδονται βασικές έννοιες, οι οποίες χρησιμοποιούνται στα υπόλοιπα κεφάλαια.

Στο δεύτερο κεφάλαιο μελετάται η απαρίθμηση συνόλων δυαδικών λέξεων με ορισμένους περιορισμούς, με τη μέθοδο των γεννητριών συναρτήσεων αλλά και με συνδυαστικές απεικονίσεις. Πιο συγκεκριμένα, γίνεται μελέτη για λέξεις Fibonacci, λέξεις χωρίς zig-zag, λέξεις Dyck και γενικά για λέξεις που αποφεύγουν συγκεκριμένα πρότυπα.

Στο τρίτο κεφάλαιο παρουσιάζεται η κατασκευή συνόλων δυαδικών λέξεων με ορισμένους περιορισμούς, για λέξεις Fibonacci, λέξεις χωρίς zig-zag και λέξεις Dyck και δίνονται επαναληπτικοί και αναδρομικοί αλγόριθμοι επαναληπτικής κατασκευής, ranking-unranking και κατασκευής σε κώδικα Gray για κάθε ένα από τα παραπάνω σύνολα.

Τέλος, τη μεταπτυχιακή αυτή διατριβή συμπληρώνει λογισμικό, το οποίο υλοποιεί τους αλγόριθμους που προκύπτουν στο τρίτο κεφάλαιο.

Περιεχόμενα

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Κεφάλαιο 1

Βασικές έννοιες

1.1 Λέξεις και διατάξεις

Αλφάβητο ονομάζεται κάθε μη κενό πεπερασμένο σύνολο $\mathcal{V}$. Τα στοιχεία του $\mathcal{V}$ ονομάζονται **γράμματα ή σύμβολα**. Κάθε πεπερασμένη ακολουθία από γράμματα του $\mathcal{V}$ ονομάζεται **λέξη** στο αλφάβητο $\mathcal{V}$. Ο αριθμός των γραμμάτων μιας λέξης α λέγεται **μήκος** της λέξης α και συμβολίζεται με $|\alpha|$. Η λέξη με μήκος 0 ονομάζεται **κενή λέξη** και συμβολίζεται με ε.

Το σύνολο όλων των λέξεων με γράμματα από το $\mathcal{V}$, συμπεριλαμβανομένης και της κενής λέξης, συμβολίζεται με $\mathcal{V}^*$ (**άστρο του Kleene**).

Αν $\alpha, \beta \in \mathcal{V}^*$ με $\alpha = \alpha_1\alpha_2\cdots\alpha_l$ και $\beta = \beta_1\beta_2\cdots\beta_m$ τότε η λέξη

$$\alpha\beta = \alpha_1\alpha_2\cdots\alpha_l\beta_1\beta_2\cdots\beta_m$$

ονομάζεται **σύζευξη ή γινόμενο** των α και β. Επιπλέον $\alpha\varepsilon = \varepsilon\alpha = \alpha$ για κάθε $\alpha \in \mathcal{V}^*$, οπότε το σύνολο $\mathcal{V}^*$, εφοδιασμένο με την πράξη της σύζευξης, αποτελεί μονοειδές.

Με $[k, n]$, όπου $k \leq n$, συμβολίζεται το σύνολο των ακεραίων $\{k, k+1, \ldots, n\}$. Ειδικά το $[1, n]$ συμβολίζεται με $[n]$.

Η λέξη $\alpha = \alpha_1\alpha_2\cdots\alpha_n$ έχει ως **υπολέξη** την $\beta = \beta_1\beta_2\cdots\beta_k$, όπου $k \leq n$ αν και μόνο αν υπάρχει γνησίως αύξουσα απεικόνιση $f : [k] \to [n]$ με $\alpha_i = \beta_{f(i)}$ $\forall i \in [k]$.

Για κάθε $\alpha, \beta \in \mathcal{V}^*$, η β είναι **πρόθεμα** (αντίστοιχα **επίθεμα**) της α, αν και μόνο αν υπάρχει $\gamma \in \mathcal{V}^*$ τέτοια ώστε $\alpha = \beta\gamma$ (αντίστοιχα $\alpha = \gamma\beta$).

Οι δυνάμεις μιας λέξης $\alpha \in \mathcal{V}^*$ ορίζονται επαγωγικά ως εξής:

$$\alpha^0 = \varepsilon \text{ και } \alpha^n = \alpha\alpha^{n-1}\ , n \in \mathbb{N}^*.$$

Αν $\alpha = \beta\tau\gamma$, με $\alpha, \beta, \gamma, \tau \in \mathcal{V}^*$ τότε η λέξη τ ονομάζεται **τμήμα** της α.

Δύο τμήματα τ_1, τ_2 της α ονομάζονται **διαδοχικά** όταν υπάρχουν $\beta, \gamma \in \mathcal{V}^*$, τέτοια ώστε $\alpha = \beta\tau_1\tau_2\gamma$.

Αν $\alpha, \tau \in \{0,1\}^*$, τότε για κάθε διακεκριμένο ζεύγος λέξεων β, γ με $\alpha = \beta\tau\gamma$, λέμε ότι η α περιέχει μία **εμφάνιση** της λέξης τ. Αν δεν υπάρχει τέτοιο ζεύγος β, γ, τότε λέμε ότι η α **αποφεύγει** την τ. Το πλήθος των εμφανίσεων της τ στην α συμβολίζεται με $|\alpha|_\tau$.

Έστω $\mathcal{V}$ ένα αλφάβητο και α μια λέξη στο $\mathcal{V}^*$. Συμβολίζουμε με $\mathcal{V}^*_\alpha$ το σύνολο όλων των λέξεων του $\mathcal{V}^*$ με πρόθεμα τη λέξη α, δηλαδή που αρχίζουν με α.

1.1.1 Λεξικογραφική διάταξη

Με τη βοήθεια της σύζευξης, μπορεί να οριστεί μια ολική διάταξη στο $\mathcal{V}^*$. Για να γίνει αυτό, απαιτείται μια ολική διάταξη "$\preceq$" στο αλφάβητο $\mathcal{V}$, όπου $x \prec y$ όταν το γράμμα x προηγείται του γράμματος y στο $\mathcal{V}$.

Μια τέτοια διάταξη ονομάζεται **αλφαβητική διάταξη** του $\mathcal{V}$ και επεκτείνεται στο σύνολο $\mathcal{V}^*$, ορίζοντας τη **λεξικογραφική διάταξη** ($\preceq$) ως εξής:

Για δύο λέξεις α, β του $\mathcal{V}^*$ ορίζεται

$$\alpha \preceq \beta \Leftrightarrow \begin{cases} \beta = \alpha\gamma, \text{ ή} \\ \alpha = \gamma x \delta \text{ και } \beta = \gamma y \delta' \text{ και } x \prec y, \text{ με } x \neq y \in \mathcal{V} \text{ και } \gamma, \delta, \delta' \in \mathcal{V}^*. \end{cases}$$

Αν $\alpha \preceq \beta$ θα λέμε ότι η λέξη α (αντ. β) **προηγείται** ή είναι **μικρότερη** (αντ. **έπεται** ή είναι **μεγαλύτερη**) της λέξης β (αντ. α). Αν $\alpha \preceq \beta$, αλλά $\alpha \neq \beta$ τότε γράφουμε $\alpha \prec \beta$.

Παραδείγματα.

1. Αν $\alpha = 0010101$ και $\beta = 0010110$, τότε $\alpha \preceq \beta$, διότι

$$\alpha = \gamma 0 \delta \text{ και } \beta = \gamma 1 \delta',$$

όπου $\gamma = 00101$, $\delta = 01$, $\delta' = 10$ και $0 \preceq 1$.

2. Αν $\alpha = 011$ και $\beta = 01101$, τότε $\alpha \preceq \beta$, διότι

$$\beta = \alpha\gamma,$$

όπου $\gamma = 01$.

Εύκολα προκύπτει ότι η σχέση $\preceq$ είναι σχέση ολικής διάταξης.

1.1.2 Ranking και unranking

Έχοντας ορίσει τη λεξικογραφική διάταξη στο $\mathcal{V}^*$, άμεσα προκύπτει η αμφιμονοσήμαντη απεικόνιση
$r : \mathcal{V}^* \to \mathbb{N}^*$ με

$$r(\varepsilon) = 1 \quad \text{και} \quad r(\alpha) \leq r(\beta) \Leftrightarrow \alpha \preceq \beta, \qquad \alpha, \beta \in \mathcal{V}^*.$$

Η απεικόνιση αυτή καλείται απεικόνιση **ranking**. Η τιμή $r(\alpha)$ ονομάζεται **βαθμός** (rank) της λέξης α και αντιστοιχεί στο πλήθος των λέξεων στο $\mathcal{V}^*$ που είναι μικρότερες (προηγούνται) ή ίσες της α. Έτσι, η κενή λέξη θεωρείται η πρώτη ή ελάχιστη λέξη του συνόλου, αφού $r(\varepsilon) = 1$ και η β είναι η επόμενη (αντίστοιχα προηγούμενη) της α αν και μόνο αν $r(\beta) = r(\alpha) + 1$ (αντίστοιχα $r(\beta) = r(\alpha) - 1$). Ειδικά, θα συμβολίζουμε την επόμενη λέξη της α με $\text{next}(\alpha)$.

Η έννοια του βαθμού μιας λέξης είναι πολύ σημαντική για τις εφαρμογές που απαιτούν την κατασκευή τυχαίων λέξεων. Ο λόγος είναι ότι ενώ υπάρχουν αποτελεσματικοί αλγόριθμοι για την κατασκευή τυχαίων αριθμών, δεν υπάρχουν γενικές μέθοδοι για την κατασκευή τυχαίων λέξεων από ένα σύνολο με συγκεκριμένες ιδιότητες ή περιορισμούς. Όμως, χρησιμοποιώντας την έννοια του βαθμού μιας λέξης το πρόβλημα της κατασκευής μιας τυχαίας λέξης ανάγεται στο πρόβλημα της επιλογής ενός τυχαίου αριθμού από το διάστημα $[n]$ όπου n ο πληθάριθμος του συνόλου των λέξεων που μας ενδιαφέρουν.

Για να είναι αποδοτική αυτή η μέθοδος, απαιτείται ένας αλγόριθμος (unranking) ο οποίος υπολογίζει ποια λέξη έχει ένα συγκεκριμένο βαθμό χωρίς να απαιτείται η κατασκευή άλλων λέξεων.

Στενά συνδεδεμένο είναι και το πρόβλημα του υπολογισμού του βαθμού μιας λέξης (ranking) χωρίς να απαιτείται η κατασκευή όλων των λέξεων που είναι λεξικογραφικά μικρότερες από αυτή.

1.1.3 Απόσταση Hamming

Απόσταση Hamming ή **απόσταση** δύο λέξεων $\alpha = \alpha_1\alpha_2\cdots\alpha_n$ και $\beta = \beta_1\beta_2\cdots\beta_n$ ονομάζεται ο αριθμός των θέσεων i στις οποίες $\alpha_i \neq \beta_i$. Θα συμβολίζουμε την απόσταση των λέξεων α, β με $d(\alpha, \beta)$.

Για παράδειγμα, οι δυαδικές λέξεις 010100 και 011000 έχουν απόσταση 2.

Εύκολα προκύπτει ότι $d(\alpha, \beta) \leq d(\alpha, \gamma) + d(\gamma, \beta)$, για οποιεσδήποτε λέξεις α, β, γ, οπότε η απόσταση Hamming είναι μια μετρική και ο μετρικός χώρος που ορίζει ονομάζεται κύβος Hamming.

1.1.4 Κώδικες Gray

Μια ακολουθία που περιέχει κάθε λέξη ενός συνόλου ακριβώς μία φορά, έτσι ώστε δύο διαδοχικές λέξεις να έχουν απόσταση Hamming μικρότερη ή ίση με k, όπου $k \in \mathbb{N}^*$, ονομάζεται k-**κώδικας Gray** του συνόλου αυτού. Ειδικά για $k = 1$, ο κώδικας ονομάζεται απλά κώδικας Gray.

Παραδείγματα.

1. Η επόμενη ακολουθία είναι κώδικας Gray για τις δυαδικές λέξεις μήκους 4:
   0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000.

2. Η επόμενη ακολουθία είναι 2-κώδικας Gray για τις μεταθέσεις μήκους 4:
   1234, 1243, 1423, 4123, 4132, 1432, 1342, 1324, 3124, 3142, 3412, 4312, 4321, 3421, 3241, 3214, 2314, 2341, 2431, 4231, 4213, 2413, 2143, 2134.

Οι κώδικες των προηγούμενων παραδειγμάτων έχουν την επιπλέον ιδιότητα ότι το πρώτο στοιχείο της ακολουθίας με το τελευταίο έχουν απόσταση 1 και 2 αντίστοιχα. Στην περίπτωση αυτή ο κώδικας Gray ονομάζεται **κυκλικός**.

Οι κώδικες Gray είναι πολύ σημαντικοί για τις εφαρμογές όπου απαιτείται κατασκευή όλων των στοιχείων ενός συνόλου, διότι κατασκευάζουν αντικείμενο προς αντικείμενο όλα τα στοιχεία του συνόλου με όσο το δυνατόν λιγότερους μετασχηματισμούς.

Επιπλέον, επιτρέπουν την γρήγορη εύρεση και ομαδοποίηση στοιχείων που διαφέρουν κατά συγκεκριμένη απόσταση.

1.2 Γεννήτριες συναρτήσεις

Οι γεννήτριες συναρτήσεις αποτελούν ένα από τα σημαντικότερα εργαλεία για την επίλυση προβλημάτων απαρίθμησης. Στην παράγραφο αυτή δίδονται ορισμένα βασικά στοιχεία για τις συνήθεις γεννήτριες συναρτήσεις ή απλά γεννήτριες συναρτήσεις.

Υπάρχουν δύο ισοδύναμοι τρόποι ορισμού των γεννητριών συναρτήσεων: στην πρώτη προσέγγιση ξεκινάμε με βάση μια ακολουθία (a_n), ενώ στη δεύτερη δίδεται ένα σύνολο και μια παράμετρός του.

Έστω S ένα σύνολο συνδυαστικών αντικειμένων. Κάθε απεικόνιση $p : S \to \mathbb{N}$ ονομάζεται **παράμετρος**.
(1η προσέγγιση) **Γεννήτρια συνάρτηση της ακολουθίας** (a_n) ονομάζεται το άθροισμα

$$F(x) = \sum_{n=0}^{\infty} a_n x^n.$$

(2η προσέγγιση) **Γεννήτρια συνάρτηση του συνόλου** S ως προς την παράμετρο p ονομάζεται το άθροισμα

$$F(x) = \sum_{\alpha \in S} x^{p(\alpha)}.$$

Παρατήρηση. Η παράμετρος p διαμερίζει το σύνολο S σε κλάσεις ισοδυναμίας S_n, $n \in \mathbb{N}$ όπου

$$S_n = \{\alpha \in S : p(\alpha) = n\},$$

οπότε

$$\sum_{\alpha \in S} x^{p(\alpha)} = \sum_{n \in \mathbb{N}} \sum_{\alpha \in S_n} x^{p(a)} = \sum_{n \in \mathbb{N}} \sum_{\alpha \in S_n} x^n = \sum_{n \in \mathbb{N}} x^n \sum_{\alpha \in S_n} 1 = \sum_{n \in \mathbb{N}} |S_n| x^n.$$

Οι παραπάνω δύο προσεγγίσεις είναι ισοδύναμες, αρκεί να θεωρήσουμε $a_n = |S_n|$, $n \in \mathbb{N}$.

Με άλλα λόγια, μια συνήθης γεννήτρια συνάρτηση είναι μια δυναμοσειρά του x, στην οποία ο συντελεστής του x^n ισούται με το πλήθος των στοιχείων του S που έχουν τιμή της παραμέτρου p ίση με n.

Στη βιβλιογραφία χρησιμοποιείται ο συμβολισμός $[x^n]F$ για τον συντελεστή του x^n στην $F(x)$.

Γεννήτρια συνάρτηση δύο μεταβλητών της ακολουθίας $(a_{n,k})$ ονομάζεται το διπλό άθροισμα

$$F(x, y) = \sum_{n=0}^{\infty} \sum_{k=0}^{\infty} a_{n,k} x^n y^k.$$

Γεννήτρια συνάρτηση δύο μεταβλητών του συνόλου S **ως προς τις παραμέτρους** $p, q : S \to \mathbb{N}$, όπου τα x, y μετρούν τα p, q αντίστοιχα, ονομάζεται το άθροισμα

$$F(x, y) = \sum_{\alpha \in S} x^{p(\alpha)} y^{q(\alpha)}.$$

Ο συντελεστής του $x^n y^k$ στην $F(x, y)$ συμβολίζεται με $[x^n y^k]F$.

Ανάλογα ορίζονται γεννήτριες συναρτήσεις τριών ή περισσότερων μεταβλητών.

Σε πολλά συνδυαστικά αντικείμενα συχνά ορίζεται μια παράμετρος αναφοράς που ονομάζεται συνήθως ''μέγεθος'' ή ''μήκος''. Η παράμετρος αυτή είναι πολύ σημαντική, αφού δίνει μια φυσική διαμέριση του αντίστοιχου συνόλου σε πεπερασμένα υποσύνολα και συνδέεται άμεσα με την απαρίθμηση και κατασκευή των αντίστοιχων αντικείμενων.

Παραδείγματα τέτοιων παραμέτρων είναι για τις δυαδικές λέξεις το ''μήκος της λέξης'', για τα δένδρα το ''πλήθος δεσμών ενός δένδρου'', για τις μεταθέσεις ''το μήκος της μετάθεσης''.

Μια άλλη πολύ σημαντική έννοια που σχετίζεται με τις παραμέτρους είναι η έννοια της στατιστικής. Αν θεωρήσουμε δύο παραμέτρους όπου η πρώτη είναι παράμετρος αναφοράς για το συνδυαστικό αντικείμενο που μας ενδιαφέρει, τότε **στατιστική** της δεύτερης παραμέτρου

(ως προς την πρώτη παράμετρο) ονομάζεται η κατανομή των αντικειμένων με συγκεκριμένο μέγεθος σε μικρότερες κλάσεις με βάση τις τιμές της δεύτερης παραμέτρου.

Για παράδειγμα, στις δυαδικές λέξεις για τις παραμέτρους ''μήκος της λέξης'' και ''αριθμός μονάδων της λέξης'', η στατιστική ''αριθμός μονάδων της λέξης'' ισούται με

$$\begin{array}{cccc} 1 & & & \\ 1 & 1 & & \\ 1 & 2 & 1 & \\ 1 & 3 & 3 & 1 \end{array}$$
κ.ο.κ.

ή σε συμβολισμό μιας γραμμής με $1; 1, 1; 1, 2, 1; 1, 3, 3, 1, \ldots$, διότι η κενή λέξη δεν έχει καμία μονάδα, από τις 2 λέξεις μήκους 1, η μία δεν έχει καμία μονάδα και η άλλη έχει μία, από τις 4 λέξεις μήκους 2, 1 δεν έχει καμία μονάδα, 2 έχουν μία μονάδα και 1 έχει δύο μονάδες, από τις 8 λέξεις μήκους 3, 1 δεν έχει καμία μονάδα, 3 έχουν μία μονάδα, 3 έχουν δύο μονάδες και 1 έχει τρεις μονάδες.

Δύο παράμετροι q_1, q_2 ονομάζονται **ισοκατανεμημένες** (ως προς τις παραμέτρους αναφοράς p_1, p_2) όταν

$$|\{\alpha \in S_n : q_1(\alpha) = k\}| = \left|\left\{\alpha \in S_n^{'} : q_2(\alpha) = k\right\}\right|$$

όπου $S_n = \{a \in S : p_1(\alpha) = n\}$ και $S_n^{'} = \{a \in S^{'} : p_2(\alpha) = n\}$, για κάθε $k, n \in \mathbb{N}$.

Ισοδύναμα, οι q_1, q_2 είναι ισοκατανεμημένες όταν οι γεννήτριες συναρτήσεις που ορίζονται από αυτές είναι ίσες, δηλαδή

$$\sum_{\alpha \in S} x^{p_1(\alpha)} y^{q_1(\alpha)} = \sum_{\alpha \in S^{'}} x^{p_2(\alpha)} y^{q_2(\alpha)}.$$

Στην περίπτωση αυτή, οι αντίστοιχες στατιστικές ονομάζονται **ισοκατανεμημένες**.

Εύκολα αποδεικνύεται ότι αν υπάρχει μια αμφιμονοσήμαντη απεικόνιση $\phi : S \to S^{'}$ με

$$p_1(\alpha) = \phi(p_2(\alpha)) \text{ και } q_1(\alpha) = \phi(q_2(\alpha))$$

τότε οι παράμετροι q_1, q_2 είναι ισοκατανεμημένες ως προς τις παραμέτρους αναφοράς p_1, p_2.

1.2.1 Το γενικευμένο διωνυμικό θεώρημα

Οι (γενικευμένοι) διωνυμικοί συντελεστές $\binom{x}{k}$ δίδονται από τον τύπο

$$\binom{x}{k} = \frac{x(x-1)\cdots(x-k+1)}{k!}, x \in \mathbb{R} \text{ και } k \in \mathbb{N}.$$

Ειδικά όταν $x = n$, όπου n φυσικός αριθμός, μπορεί να οριστεί μία επέκτασή τους $\binom{x}{k}$, όπου το x είναι πραγματικός αριθμός και το k φυσικός αριθμός. Τότε ο προηγούμενος τύπος μπορεί να γραφτεί στη μορφή:

$$\binom{n}{k} = \begin{cases} \dfrac{n!}{k!(n-k)!}, & \text{αν } n \geq k, \\ 0, & \text{αν } n < k. \end{cases}$$

Παραδείγματα. Ισχύει ότι:

1. $\binom{-n}{k} = (-1)^k \binom{n+k-1}{k}$, για κάθε φυσικό αριθμό n.
2. $\binom{-\frac{1}{2}}{k} = \dfrac{(-1)^k}{4^k}\binom{2k}{k}$ και $\binom{\frac{1}{2}}{k} = \dfrac{(-1)^{k-1}}{2^{2k-1}k}\binom{2k-2}{k-1}$.
3. $\binom{-\frac{3}{2}}{k} = \dfrac{(2k+2)(-1)^k}{2^{2k+1}}\binom{2k+1}{k+1}$ και $\binom{\frac{3}{2}}{k} = \dfrac{3(-1)^k}{2^{2k-2}(k-2)(k-3)}\binom{2k-4}{k-4}$.

Χρησιμοποιώντας τη γενίκευση των διωνυμικών συντελεστών, το κλασικό διωνυμικό θεώρημα του Newton $(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$, όπου $n \in \mathbb{N}^*$, επεκτείνεται για κάθε ρητό αριθμό n, και ισχύει ότι

$$(a+b)^n = \sum_{k=0}^{\infty} \binom{n}{k} a^k b^{n-k}, \tag{1.1}$$

όπου $n \in \mathbb{Q}$, $b \neq 0$ και $\frac{a}{b} \in (-1, 1)$.

1.2.2 Μετασχηματισμοί με συνδυαστικές ιδιότητες

Έστω $F(x, y)$ η γεννήτρια συνάρτηση ενός συνόλου ως προς δύο παραμέτρους p και q, όπου το x μετράει την p και το y την q.

1. Αν τεθεί $y = 1$, τότε η $F(x, 1) = F(x)$ μετράει το σύνολο μόνο ως προς την παράμετρο p.
2. Αν τεθεί $y = 0$, τότε η $F(x, 0)$ μετράει το σύνολο των στοιχείων με τιμή παραμέτρου q ίση με 0 ως προς την παράμετρο p.
3. Αν τεθεί $y = 1$ στη μερική παράγωγο της $F(x, y)$ ως προς y, τότε η $\dfrac{\partial F(x,y)}{\partial y}|_{y=1}$ μετράει το σύνολο ως προς την παράμετρο p και την παράμετρο συνολικό άθροισμα των τιμών της q.

4. Η γεννήτρια συνάρτηση $\frac{1}{2}(F(x,y)+F(-x,y))$ μετράει τα στοιχεία με τιμή της παραμέτρου p άρτιο αριθμό ως προς την παράμετρο q, ενώ η γεννήτρια συνάρτηση $\frac{1}{2}(F(x,y)-F(-x,y))$ μετράει τα στοιχεία με τιμή της παραμέτρου p περιττό αριθμό ως προς την παράμετρο q.

Παράδειγμα. Έστω $F(x,y)$ η γεννήτρια συνάρτηση του συνόλου των δυαδικών λέξεων $\{0,1\}^*$ ως προς τις παραμέτρους μήκος $|\ |$ και αριθμός εμφανίσεων του 1 $|\ |_1$, όπου το x μετράει το μήκος και το y τις εμφανίσεις του 1.

Κάθε μη κενή δυαδική λέξη α διασπάται κατά μοναδικό τρόπο υπό τη μορφή

$$\alpha = 0\beta,\ \text{ή}\ \alpha = 1\beta,$$

όπου β είναι μια δυαδική λέξη.

Ισχύει ότι

$$\begin{aligned} F(x,y) &= \sum_{\alpha\in\{0,1\}^*} x^{|\alpha|}y^{|\alpha|_1} \\ &= 1 + \sum_{\substack{\alpha=0\beta \\ \beta\in\{0,1\}^*}} x^{|0\beta|}y^{|0\beta|_1} + \sum_{\substack{\alpha=1\beta \\ \beta\in\{0,1\}^*}} x^{|1\beta|}y^{|1\beta|_1} \\ &= 1 + x\sum_{\beta\in\{0,1\}^*} x^{|\beta|}y^{|\beta|_1} + xy\sum_{\beta\in\{0,1\}^*} x^{|\beta|}y^{|\beta|_1} \\ &= 1 + x(1+y)F(x,y). \end{aligned}$$

Επομένως,

$$F(x,y) = \frac{1}{1-x(1+y)}.$$

1. Αν τεθεί $y=1$, τότε
$$F(x,1) = \frac{1}{1-2x} = \sum_{n=0}^{\infty} 2^n x^n.$$
Η $F(x,1)$ μετράει τις δυαδικές λέξεις ως προς το μήκος τους και επομένως, το πλήθος των δυαδικών λέξεων μήκους n ισούται με 2^n.

2. Αν τεθεί $y=0$, τότε
$$F(x,0) = \frac{1}{1-x} = \sum_{n=0}^{\infty} x^n.$$
Η $F(x,0)$ μετράει τις δυαδικές λέξεις που δεν περιέχουν 1 ως προς το μήκος τους, και επομένως, για κάθε μήκος υπάρχει ακριβώς μια τέτοια λέξη, (η 0^n).

3. Η μερική παράγωγος της $F(x,y)$ ως προς y ισούται με $\dfrac{x}{(1-x-xy)^2}$. Αν τεθεί $y=1$, τότε προκύπτει η γεννήτρια

$$\frac{x}{(1-2x)^2} = \sum_{n=0}^{\infty}\sum_{k=0}^{n} 2^k 2^{n-k} x^{n+1} = \sum_{n=0}^{\infty}(n+1)2^n x^{n+1} = \sum_{n=1}^{\infty} n 2^{n-1} x^n,$$

η οποία μετράει τις δυαδικές λέξεις ως προς το μήκος τους και τον συνολικό αριθμό των μονάδων για κάθε μήκος. Επομένως, ο αριθμός των εμφανίσεων του 1 σε όλες τις δυαδικές λέξεις μήκους n ισούται με $n \cdot 2^{n-1}$. Συνεπώς, ο μέσος όρος των εμφανίσεων του 1 στις δυαδικές λέξεις μήκους n ισούται με

$$\frac{n2^{n-1}}{2^n} = \frac{n}{2},$$

δηλαδή σε μια δυαδική λέξη μήκους n, κατά μέσο όρο τα μισά γράμματά της είναι 1.

1.2.3 Το θεώρημα αντιστροφής του Lagrange

Ένα σημαντικό εργαλείο για την επίλυση συναρτησιακών εξισώσεων που αφορούν γεννήτριες συναρτήσεις είναι το θεώρημα αντιστροφής του Lagrange. Παρακάτω δίδεται μια ειδική μορφή του, όπως παρουσιάζεται στην εργασία του Deutsch [4].

Θεώρημα 1.2.1 (Θεώρημα αντιστροφής του Lagrange). *Αν η γεννήτρια συνάρτηση $F(x)$ ικανοποιεί την εξίσωση*

$$F(x) = 1 + xH\left(F(x)\right),$$

όπου $H(\lambda)$ είναι πολυώνυμο του λ, τότε για κάθε πολυώνυμο $G(\lambda)$ του λ ισχύει ότι

$$[x^n]G(F(x)) = \frac{1}{n}[\lambda^{n-1}]G'(1+\lambda)\left(H(1+\lambda)\right)^n.$$

Ειδικά για $G(x) = x$ ισχύει ότι

$$[x^n]F(x) = \frac{1}{n}[\lambda^{n-1}]\left(H(1+\lambda)\right)^n. \tag{1.2}$$

1.2.4 Ασυμπτωτικές προσεγγίσεις

Σε πολλά προβλήματα απαρίθμησης, είτε είναι αδύνατη η εύρεση ενός τύπου για τους συντελεστές μιας γεννήτριας συνάρτησης, είτε οι τύποι που προκύπτουν δεν επιτρέπουν τη σύγκριση των συντελεστών που αντιστοιχούν σε διαφορετικές απαριθμήσεις. Επιπλέον, σε πολλές περιπτώσεις, αυτό που μας ενδιαφέρει δεν είναι η ακριβής τιμή κάθε συντελεστή αλλά ο ρυθμός αύξησης των συντελεστών. Σε αυτές τις περιπτώσεις, προτιμάται η όσο

το δυνατό καλύτερη εκτίμηση της τάξης[1] των συντελεστών αυτών. Στην ενότητα αυτή θα δοθούν ορισμένες προτάσεις οι οποίες αναφέρονται σε ασυμπτωτικές προσεγγίσεις των συντελεστών μιας γεννήτριας.

Ενώ μέχρι τώρα αντιμετωπίζαμε τις γεννήτριες συναρτήσεις από αλγεβρική σκοπιά, στο επόμενο αποτέλεσμα χρησιμοποιούνται ορισμένες αναλυτικές ιδιότητες των γεννητριών συναρτήσεων.

Μια (μιγαδική) σειρά $F(x)$ έχει **ιδιομορφία (singularity) στο σημείο** $x = r$ αν δεν ορίζεται η παράγωγος της F στο σημείο αυτό. Στις εφαρμογές που θα συναντήσουμε, τα πιο συνηθισμένα σημεία ιδιομορφίας είναι τα σημεία μηδενισμού υπόρριζων εκφράσεων.

Πρόταση 1.2.2. *Έστω ότι η σειρά* $F(x) = \sum_{n=0}^{\infty} a_n x^n$ *συγκλίνει για κάποιο* $x > 0$, *όπου* a_n *είναι μια ακολουθία θετικών ακεραίων. Αν*

$$F(x) = f(x)g(x) + h(x),$$

όπου

i) $f(x) = (-\ln(1 - x/r))^b(1 - x/r)^c$, *όπου* c *δεν είναι θετικός ακέραιος και δεν ισχύει* $b = c = 0$,

ii) *Η* $F(x)$ *δεν έχει ιδιομορφία για* $-r \leq x < r$,

iii) *Το όριο* $L = \lim_{x \to r} g(x)$ *υπάρχει και είναι μη μηδενικό,*

iv) *Η* $h(x)$ *δεν έχει ιδιομορφία στο* $x = r$,

τότε [2]

$$a_n \sim \begin{cases} \dfrac{L(\ln n)^b (1/r)^n}{n^{c+1}\Gamma(-c)}, & \text{αν } c \neq 0, \\ \dfrac{bL(\ln n)^{b-1}(1/r)^n}{n}, & \text{αν } c = 0, \end{cases}$$

όπου Γ *είναι η* **συνάρτηση Γάμμα**.

[1] Λέμε ότι η $f(n)$ έχει τάξη $g(n)$ όταν $f(n) = \Theta(g(n))$, δηλαδή υπάρχουν σταθερές $c_1, c_2 > 0$ και $n_0 \in \mathbb{N}$, ώστε $n \geq n_0 \Rightarrow c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|$. Ως $g(n)$ επιλέγεται συνήθως μια απλή συνάρτηση (πολυωνυμική, εκθετική, λογαριθμική ή συνδυασμός των προηγούμενων).

[2] Υπενθυμίζουμε ότι για δύο ακολουθίες f, g, είναι $f(n) \sim g(n) \Leftrightarrow \lim_{n\to\infty} \dfrac{f(n)}{g(n)} = 1$.

1.2.5 The Online Encyclopedia of Integer Sequences

Στα προβλήματα απαρίθμησης που αφορούν ετερόκλητα αντικείμενα, εμφανίζονται συχνά οι ίδιες ακολουθίες αριθμών. Το γεγονός αυτό δεν είναι καθόλου συμπτωματικό, διότι όταν δύο συνδυαστικά αντικείμενα απαριθμούνται από την ίδια ακολουθία βαθμών συνήθως υπάρχει μια ''δομική'' συγγένεια μεταξύ τους. Παραδείγματος χάριν, τα αντικείμενα στα οποία εμφανίζονται οι αριθμοί Catalan μπορούν να ορισθούν με παρόμοιο τρόπο αναδρομικά από ''μικρότερα'' αντικείμενα του ίδιου τύπου με αυτά. Τα τελευταία χρόνια, πολλές εργασίες ασχολούνται με την κατασκευή αμφιμονοσήμαντων απεικονίσεων ανάμεσα σε διάφορα αντικείμενα, για να ερμηνεύσουν αυτές τις κοινές απαριθμήσεις.

Από το γεγονός αυτό προέκυψε η ανάγκη συστηματικής μελέτης και καταγραφής ορισμένων ακολουθιών αριθμών. Το 1973, ο Sloane εξέδωσε το βιβλίο A Handbook of Integer Sequences, το οποίο περιλάμβανε στοιχεία για περίπου 2300 ακολουθίες. Το 1995, σε συνεργασία με τον Plouffe, ακολούθησε νέα έκδοση υπό τον τίτλο The Encyclopedia of Integer Sequences, η οποία περιλάμβανε περίπου 5000 ακολουθίες. Η χρησιμότητα αυτών των βιβλίων έγινε φανερή από την αρχή (βλ. για παράδειγμα τις βιβλιοκριτικές των Borwein και Corless [3], ή του Guy [8]).

Από το 1996, τα βιβλία αυτά πέρασαν σε ηλεκτρονική μορφή διαθέσιμη στο διαδίκτυο, γνωστή ως The Online Encyclopedia of Integer Sequences [13], η οποία σήμερα περιλαμβάνει πάνω από 175.000 ακολουθίες ακεραίων αριθμών, και ανανεώνεται καθημερινά από ερευνητές με νέες ακολουθίες και με νέα στοιχεία για τις υπάρχουσες ακολουθίες.

Για παράδειγμα, η καταχώριση της ακολουθίας των αριθμών Fibonacci στην Online Encyclopedia of Integer Sequences μεταξύ άλλων συνίσταται από τα εξής στοιχεία:

Στοιχείο	Παράδειγμα	Επεξήγηση
ID Number	A000045	Μοναδικός κωδικός για κάθε ακολουθία
Name	Fibonacci numbers	Σύντομη περιγραφή της ακολουθίας
Sequence	0, 1, 1, 2, 3, 5, 8, 13, 21, ...	Αρχικοί όροι της ακολουθίας
Comments	Also called Lamé's sequence, number of subsets of [n] that contain no consecutive integer	Επιπλέον στοιχεία για την ακολουθία
References	Mohammad K. Azarian, The Generating Function for the Fibonacci Sequence, Missouri Journal of Mathematical Sciences, Vol. 2, No. 2, Spring 1990, pp. 78-79	Εργασίες σχετικές με την ακολουθία
Links	R. Dickau, Fibonacci numbers	Σύνδεσμοι με επιπλέον στοιχεία
Formula	G.f.: $x/(1-x-x^2)$. F(n) = round(phi^n/sqrt(5))	Τύποι και σχέσεις για την ακολουθία
Maple	···	Κώδικας υπολογισμού για Maple
Mathematica	···	Κώδικας υπολογισμού για Mathematica
Crossrefs	A039834	Σχετικές ακολουθίες
Author	N. J. A. Sloane	Δημιουργός της καταχώρισης
Extensions	Edited by C R Greathouse IV	Αναθεωρητής της καταχώρισης

Κεφάλαιο 2

Απαρίθμηση δυαδικών λέξεων με περιορισμούς

2.1 Λέξεις Fibonacci

Μια δυαδική λέξη $\alpha \in \{0,1\}^*$ ονομάζεται **λέξη Fibonacci**, αν δεν περιέχει δύο διαδοχικά 0.

Παράδειγμα. Οι λέξεις Fibonacci μήκους το πολύ 4 είναι οι εξής: ε, 0, 1, 01, 10, 11, 010, 011, 101, 110, 111, 0101, 0110, 0111, 1010, 1011, 1101, 1110, 1111.

2.1.1 Διάσπαση των λέξεων Fibonacci

Πρόταση 2.1.1. *Κάθε μη κενή λέξη Fibonacci α διασπάται κατά μοναδικό τρόπο υπό τη μορφή*

$$\alpha = 0,\ \text{ή}\ \alpha = 01\beta,\ \text{ή}\ \alpha = 1\beta,$$

όπου β είναι λέξη Fibonacci.

Απόδειξη. Κάθε μη κενή λέξη Fibonacci α είτε αρχίζει με 0 είτε αρχίζει με 1.

Στην περίπτωση όπου $\alpha = 0\gamma$, η γ δεν περιέχει δύο διαδοχικά 0 και επιπλέον δεν αρχίζει με 0. Οπότε, $\gamma = \varepsilon$, ή $\gamma = 1\beta$, όπου β δεν περιέχει δύο διαδοχικά 0, δηλαδή είναι λέξη Fibonacci.

Στην περίπτωση όπου $\alpha = 1\beta$, η β είναι λέξη Fibonacci.

Άρα, τελικά είναι $\alpha = 0,\ \text{ή}\ \alpha = 01\beta,\ \text{ή}\ \alpha = 1\beta$, όπου β είναι λέξη Fibonacci. □

2.1.2 Απαρίθμηση ως προς το μήκος και τον αριθμό των μονάδων

Πρόταση 2.1.2.

i) Ο αριθμός των λέξεων Fibonacci μήκους n ισούται με τον $(n+1)$-οστό αριθμό Fibonacci.

ii) Ο αριθμός των λέξεων Fibonacci μήκους n με k μονάδες ισούται με $\binom{k+1}{n-k}$.

Απόδειξη. Έστω $F(x,y)$ η γεννήτρια συνάρτηση του συνόλου Φ των λέξεων Fibonacci ως προς τις παραμέτρους μήκος της λέξης $|\ |$ και αριθμός εμφανίσεων του 1 $|\ |_1$, δηλαδή

$$F(x,y) = \sum_{\alpha\in\Phi} x^{|\alpha|}y^{|\alpha|_1}.$$

Από την προηγούμενη πρόταση, κάθε μη κενή λέξη Fibonacci διασπάται κατά μοναδικό τρόπο υπό την μορφή

$$\alpha = 0 \text{ ή } \alpha = 01\beta \text{ ή } \alpha = 1\beta.$$

Για τις παράμετρους $|\ |, |\ |_1 : \Phi \to \mathbb{N}$ ισχύει ότι

$$|\varepsilon| = 0, \quad |01\beta| = 2 + |\beta|, \quad |1\beta| = 1 + |\beta|$$

και

$$|\varepsilon|_1 = 0, \quad |01\beta|_1 = 1 + |\beta|_1, \quad |1\beta|_1 = 1 + |\beta|_1.$$

Επομένως,

$$\begin{aligned}
F(x,y) &= \sum_{\alpha\in\Phi} x^{|\alpha|}y^{|\alpha|_1} \\
&= \sum_{\alpha=\varepsilon} x^{|\varepsilon|}y^{|\varepsilon|_1} + \sum_{\alpha=0} x^{|0|}y^{|0|_1} + \sum_{\substack{\alpha=01\beta \\ \beta\in\Phi}} x^{|01\beta|}y^{|01\beta|_1} + \sum_{\substack{\alpha=1\beta \\ \beta\in\Phi}} x^{|1\beta|}y^{|1\beta|_1} \\
&= 1 + x + \sum_{\beta\in\Phi} x^{|\beta|+2}y^{|\beta|_1+1} + \sum_{\beta\in\Phi} x^{|\beta|+1}y^{|\beta|_1+1} \\
&= 1 + x + x^2y\sum_{\beta\in\Phi} x^{|\beta|}y^{|\beta|_1} + xy\sum_{\beta\in\Phi} x^{|\beta|}y^{|\beta|_1} \\
&= 1 + x + x^2yF(x,y) + xyF(x,y).
\end{aligned}$$

Άρα,

$$F(x,y) = 1 + x + xy(x+1)F(x,y). \tag{2.1}$$

i) Αν στην (2.1) τεθεί $y = 1$, προκύπτει ότι

$$F(x,1) = 1 + x + x(x+1)F(x,1) = \frac{1+x}{1-x(x+1)}. \tag{2.2}$$

Επομένως, από την (2.2) ισχύει ότι

$$F(x,1) = \sum_{m=0}^{\infty} x^m (x+1)^{m+1} = \sum_{m=0}^{\infty}\sum_{k=0}^{m+1}\binom{m+1}{k}x^{m+k} = \sum_{n=0}^{\infty}\sum_{k=0}^{[\frac{n+1}{2}]}\binom{n+1-k}{k}x^n.$$

Άρα, ο αριθμός $a_n = [x^n]F(x,1)$ των λέξεων Fibonacci μήκους n δίδεται από τον τύπο

$$a_n = \sum_{k=0}^{[\frac{n+1}{2}]}\binom{n+1-k}{k}. \tag{2.3}$$

Θα αποδειχθεί ότι $a_n = F_{n+1}$, όπου F_n είναι ο n-οστός αριθμός Fibonacci. (Υπενθυμίζουμε ότι οι αριθμοί Fibonacci ορίζονται από την αναδρομική σχέση $F_{n+2} = F_{n+1}+F_n$, για $n \geq 0$, όπου $F_0 = F_1 = 1$.)

Για $n = 0$, είναι $a_0 = \binom{1}{0} = 1 = F_1$. Για $n = 1$, είναι $a_1 = \binom{2}{0} + \binom{1}{1} = 2 = F_2$. Επίσης,

$$\begin{aligned}
a_n + a_{n+1} &= \sum_{k=0}^{[\frac{n+1}{2}]}\binom{n+1-k}{k} + \sum_{k=0}^{[\frac{n+2}{2}]}\binom{n+2-k}{k} \\
&= \sum_{k=0}^{[\frac{n+1}{2}]}\binom{n+1-k}{k} + \binom{n+2}{0} + \sum_{k=1}^{[\frac{n+2}{2}]}\binom{n+2-k}{k} \\
&= \sum_{k=0}^{[\frac{n+1}{2}]}\binom{n+1-k}{k} + \sum_{k=0}^{[\frac{n}{2}]}\binom{n+1-k}{k+1} + 1 \\
&= \sum_{k=0}^{[\frac{n}{2}]}\left(\binom{n+1-k}{k} + \binom{n+1-k}{k+1}\right) + \left(\left[\frac{n+1}{2}\right] - \left[\frac{n}{2}\right]\right)\binom{n+1-\left[\frac{n+1}{2}\right]}{\left[\frac{n+1}{2}\right]} + 1 \\
&= \sum_{k=0}^{[\frac{n}{2}]}\binom{n+2-k}{k+1} + \left(\left[\frac{n+1}{2}\right] - \left[\frac{n}{2}\right]\right) + 1 \\
&= \sum_{k=1}^{[\frac{n+2}{2}]}\binom{n+3-k}{k} + \left(\left[\frac{n+3}{2}\right] - \left[\frac{n+2}{2}\right]\right)\binom{n+3-\left[\frac{n+3}{2}\right]}{\left[\frac{n+3}{2}\right]} + \binom{n+3}{0} \\
&= \sum_{k=0}^{[\frac{n+3}{2}]}\binom{n+3-k}{k} = a_{n+2},
\end{aligned}$$

οπότε $a_n = F_{n+1}$.

ii) Αν τεθεί $H(\lambda) = 1 + (x+1)y\lambda$, η σχέση (2.1) γράφεται $F(x,y) = 1 + xH(F(x,y))$, οπότε από το θεώρημα αντιστροφής του Lagrange προκύπτει ότι

$$\begin{aligned}
[x^n]F(x,y) &= \frac{1}{n}[\lambda^{n-1}](H(\lambda+1))^n = \frac{1}{n}[\lambda^{n-1}](1+y(x+1)(\lambda+1))^n \\
&= \frac{1}{n}[\lambda^{n-1}]\sum_{j=0}^{n}\binom{n}{j}(x+1)^j y^j(\lambda+1)^j \\
&= \frac{1}{n}[\lambda^{n-1}]\sum_{j=0}^{n}\sum_{i=0}^{j}\binom{n}{j}\binom{j}{i}(x+1)^j y^j\lambda^i \\
&= \frac{1}{n}[\lambda^{n-1}]\sum_{i=0}^{n}\sum_{j=i}^{n}\binom{n}{j}\binom{j}{i}(x+1)^j y^j\lambda^i \\
&= \frac{1}{n}\sum_{j=n-1}^{n}\binom{n}{j}\binom{j}{n-1}(x+1)^j y^j \\
&= \frac{1}{n}\left(\binom{n}{n-1}\binom{n-1}{n-1}(x+1)^{n-1}y^{n-1} + \binom{n}{n}\binom{n}{n-1}(x+1)^n y^n\right) \\
&= (x+1)^{n-1}y^{n-1} + (x+1)^n y^n.
\end{aligned}$$

Επομένως,

$$\begin{aligned}
F(x,y) &= 1 + \sum_{n=1}^{\infty}((x+1)^{n-1}y^{n-1} + (x+1)^n y^n)x^n \\
&= 1 + \sum_{n=1}^{\infty}(x+1)^{n-1}y^{n-1}x^n + \sum_{n=1}^{\infty}(x+1)^n y^n x^n \\
&= \sum_{n=0}^{\infty}(x+1)^n y^n x^{n+1} + \sum_{n=0}^{\infty}(x+1)^n y^n x^n \\
&= \sum_{n=0}^{\infty}(x+1)^{n+1}y^n x^n \\
&= \sum_{n=0}^{\infty}\sum_{j=0}^{n+1}\binom{n+1}{j}y^n x^{n+j} \\
&= \sum_{n=0}^{\infty}\sum_{j=0}^{[\frac{n+1}{2}]}\binom{n-j+1}{j}y^{n-j}x^n \\
&= \sum_{n=0}^{\infty}\sum_{k=[\frac{n-1}{2}]}^{n}\binom{k+1}{n-k}y^k x^n.
\end{aligned}$$

Άρα, ο αριθμός των λέξεων Fibonacci μήκους n με k μονάδες ισούται με

$$\binom{k+1}{n-k}.$$

□

Οι πρώτοι όροι της ακολουθίας $[x^n y^k]F(x,y)$ δίδονται στον επόμενο πίνακα.

$n \setminus k$	0	1	2	3	4	5	6	7	8	9	10	
0	1											1
1		1	1									2
2		2	1									3
3		1	3	1								5
4			3	4	1							8
5			1	6	5	1						13
6				4	10	6	1					21
7				1	10	15	7	1				34
8					5	20	21	8	1			55
9					1	15	35	28	9	1		89
10						6	35	56	36	10	1	144

Παράδειγμα. Το πλήθος των λέξεων Fibonacci μήκους 7 με 4 μονάδες ισούται με $\binom{4+1}{7-4} = \binom{5}{3} = 10$. Πράγματι, οι λέξεις αυτές είναι οι ακόλουθες:

$$\begin{array}{ccccc} 0101011 & 0101101 & 0101110 & 0110101 & 0110110 \\ 0111010 & 1010101 & 1010110 & 1011010 & 1101010 \end{array}$$

Η έκφραση των αριθμών a_n όπως δίνεται από την σχέση (2.3) δεν δίνει άμεσα μια εκτίμηση της τάξης αυτών των αριθμών. Για το σκοπό αυτό, θα δοθεί και μία δεύτερη έκφραση του αριθμού a_n.

Από την (2.2) ισχύει ότι

$$\begin{aligned} F(x,1) &= \frac{1+x}{1-x(x+1)} \\ &= \frac{1+x}{(1-p_1 x)(1-p_2 x)}, \end{aligned}$$

όπου $p_1 = \frac{1+\sqrt{5}}{2}$ και $p_2 = \frac{1-\sqrt{5}}{2}$. Επίσης, ισχύει ότι $p_1 p_2 = -1$ και $p_1 + p_2 = 1$.

Με ανάλυση σε απλά κλάσματα, προκύπτει ότι

$$\begin{aligned} \frac{1+x}{(1-p_1 x)(1-p_2 x)} &= \frac{A}{(1-p_1 x)} + \frac{B}{(1-p_2 x)} \\ 1+x &= A(1-p_2 x) + B(1-p_1 x). \end{aligned}$$

Θέτοντας $x = \frac{1}{p_1}$ και $x = \frac{1}{p_2}$ αντίστοιχα, προκύπτει ότι $A = \frac{p_1+1}{p_1-p_2}$ και $B = \frac{p_2+1}{p_2-p_1}$. Επομένως,

$$\begin{aligned} F(x,1) &= \frac{p_1+1}{p_1-p_2}\sum_{n=0}^{\infty} p_1^n x^n + \frac{p_2+1}{p_2-p_1}\sum_{n=0}^{\infty} p_2^n x^n \\ &= \sum_{n=0}^{\infty}\left(\frac{p_1+1}{p_1-p_2}p_1^n + \frac{p_2+1}{p_2-p_1}p_2^n\right)x^n. \end{aligned}$$

Άρα, ο αριθμός $a_n = [x^n]F(x,1)$ των λέξεων Fibonacci μήκους n, δίδεται από τον τύπο

$$a_n = \frac{p_1+1}{p_1-p_2}p_1^n + \frac{p_2+1}{p_2-p_1}p_2^n, \tag{2.4}$$

όπου $p_1 = \frac{1+\sqrt{5}}{2}$ και $p_2 = \frac{1-\sqrt{5}}{2}$.

Πόρισμα 2.1.3.

i) Ο αριθμός των λέξεων Fibonacci μήκους n είναι ασυμπτωτικά ισοδύναμος με

$$\left(\frac{3+\sqrt{5}}{2\sqrt{5}}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n \simeq 1,17\cdot(1,61)^n.$$

ii) Ο αριθμός των λέξεων Fibonacci μήκους n με k μηδενικά ισούται με

$$\binom{n-k+1}{k}.$$

iii) Ο μέσος όρος του αριθμού των μονάδων στις λέξεις Fibonacci μήκους n είναι ασυμπτωτικά ισοδύναμος με

$$\frac{1+\sqrt{5}}{2\sqrt{5}}n - \frac{3-\sqrt{5}}{5}.$$

Απόδειξη.

i) Από την σχέση (2.4) ισχύει ότι

$$a_n = \frac{p_1+1}{p_1-p_2}p_1^n + \frac{p_2+1}{p_2-p_1}p_2^n,$$

όπου $p_1 = \frac{1+\sqrt{5}}{2}$ και $p_2 = \frac{1-\sqrt{5}}{2}$.

Επειδή $|p_2| < 1$, έπεται ότι

$$a_n \sim \frac{p_1+1}{p_1-p_2}p_1^n = \left(\frac{3+\sqrt{5}}{2\sqrt{5}}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n \simeq 1,17\cdot(1,61)^n. \tag{2.5}$$

Στον επόμενο πίνακα δίδονται για σύγκριση οι τιμές του αριθμού των λέξεων Fibonacci, οι εκτιμήσεις αυτών των αριθμών και το αντίστοιχο σφάλμα.

μήκος	αριθμός λέξεων	εκτίμηση	σφάλμα
1	2	1.89443	-0.105573
2	3	3.06525	0.0652476
4	8	8.02492	0.0249224
8	55	55.0036	0.00363612
16	2584	2584.000077	$< 10^{-4}$
32	5702887	5702887	$< 10^{-7}$
64	27777890035288	27777890035288	$< 10^{-14}$
128	659034621587630041982498215	659034621587630041982498215	$< 10^{-27}$

ii) Επειδή μια λέξη Fibonacci μήκους n με $n-k$ μονάδες έχει k μηδενικά, προκύπτει ότι ο αριθμός των λέξεων Fibonacci μήκους n με k μηδενικά ισούται με

$$\binom{n-k+1}{k}.$$

iii) Από τη σχέση (2.1), παραγωγίζοντας ως προς y, προκύπτει ότι

$$\frac{\partial F(x,y)}{\partial y} = \frac{x(1+x)^2}{(1-x(1+x)y)^2}$$

και, θέτοντας $y = 1$, προκύπτει η γεννήτρια

$$\frac{x(1+x)^2}{(1-x(1+x))^2} = x(F(x,1))^2,$$

για την οποία ισχύει ότι

$$
\begin{aligned}
x(F(x,1))^2 &= x\left(\frac{p_1+1}{p_1-p_2}\frac{1}{1-p_1x}+\frac{p_2+1}{p_2-p_1}\frac{1}{1-p_2x}\right)^2\\
&= x\left(\frac{(p_1+1)^2}{(p_1-p_2)^2(1-p_1x)^2}-2\frac{(p_1+1)(p_2+1)}{(1-p_1x)(p_2-p_1)^2(1-p_2x)}+\frac{(p_2+1)^2}{(p_2-p_1)^2(1-p_2x)^2}\right)\\
&= x\left(\frac{7+3\sqrt{5}}{10(1-p_1x)^2}-\frac{2}{5\sqrt{5}}\left(\frac{p_1}{1-p_1x}-\frac{p_2}{1-p_2x}\right)+\frac{7-3\sqrt{5}}{10(1-p_2x)^2}\right)\\
&= x\left(\frac{7+3\sqrt{5}}{10}\sum_{n=0}^{\infty}\binom{-2}{n}(-p_1)^nx^n-\frac{2}{5\sqrt{5}}\sum_{n=0}^{\infty}(p_1^{n+1}-p_2^{n+1})x^n+\frac{7-3\sqrt{5}}{10}\sum_{n=0}^{\infty}\binom{-2}{n}(-p_2)^nx^n\right)\\
&= \sum_{n=0}^{\infty}\left(\frac{7+3\sqrt{5}}{10}(n+1)p_1^n-\frac{2}{5\sqrt{5}}(p_1^{n+1}-p_2^{n+1})+\frac{7-3\sqrt{5}}{10}(n+1)p_2^n\right)x^{n+1}\\
&= \sum_{n=0}^{\infty}\left(\frac{7+3\sqrt{5}}{10}np_1^{n-1}-\frac{2p_1^n}{5\sqrt{5}}+\frac{2p_2^n}{5\sqrt{5}}+\frac{7-3\sqrt{5}}{10}np_2^{n-1}\right)x^n.
\end{aligned}
$$

Επομένως,

$$
[x^n]x(F(x,1))^2 = p_1^{n-1}\left(\frac{7+3\sqrt{5}}{10}n-\frac{1+\sqrt{5}}{5\sqrt{5}}\right)+p_2^{n-1}\left(\frac{7-3\sqrt{5}}{10}n+\frac{1-\sqrt{5}}{5\sqrt{5}}\right).
$$

Επειδή $|p_2|<1$, έπεται ότι

$$
[x^n]x(F(x,1))^2 \sim p_1^{n-1}\left(\frac{7+3\sqrt{5}}{10}n-\frac{1+\sqrt{5}}{5\sqrt{5}}\right).
$$

Συνεπώς, από την σχέση (2.5) προκύπτει ότι ο μέσος όρος του αριθμού των μονάδων στις λέξεις Fibonacci μήκους n ισούται περίπου με

$$
\frac{\frac{7+3\sqrt{5}}{10}n-\frac{1+\sqrt{5}}{5\sqrt{5}}}{\left(\frac{3+\sqrt{5}}{2\sqrt{5}}\right)\left(\frac{1+\sqrt{5}}{2}\right)} = \frac{7\sqrt{5}+15}{20+10\sqrt{5}}n-\frac{1+\sqrt{5}}{10+5\sqrt{5}} = \frac{1+\sqrt{5}}{2\sqrt{5}}n-\frac{3-\sqrt{5}}{5}.
$$

Επομένως, ο μέσος όρος είναι

$$
\frac{p_1}{\sqrt{5}}n-\frac{3-\sqrt{5}}{5} \simeq 0,723607n-0,152786.
$$

Στον επόμενο πίνακα δίδονται για σύγκριση οι τιμές του πραγματικού μέσου όρου, η εκτίμηση του μέσου όρου και το αντίστοιχο σφάλμα.

μήκος	μέσος όρος	εκτίμηση	σφάλμα
1	0,5	0,5708204	-0,07082039
2	1,333333	1.2944272	0,03890614
4	2,75	2.7416408	0.008359214
8	5.636364	5.6360680	0.00029566
16	11.4249226	11.4249223	$< 10^{-6}$
32	23.00263112	23.00263112	$< 10^{-13}$
64	46.15804865	46.15804865	$< 10^{-26}$
128	92.46888371	92.46888371	$< 10^{-52}$

□

Για περισσότερα στοιχεία σχετικά με τους αριθμούς Fibonacci και με τους συντελεστές της γεννήτριας $F(x, y)$ βλέπε τις ακολουθίες A000045, A030528, A098925 και A102426 στην Online Encyclopedia of Integer Sequences [13].

2.2 Λέξεις χωρίς zig-zag

Μια δυαδική λέξη $\alpha \in \{0, 1\}^*$ δεν περιέχει zig-zag αν δεν περιέχει τα τμήματα 010 και 101.

Παραδείγματα. Οι δυαδικές λέξεις χωρίς zig-zag μήκους το πολύ 4 είναι οι εξής: ε, 0, 1, 00, 01, 10, 11, 000, 001, 011, 100, 110, 111, 0000, 0001, 0011, 0110, 0111, 1000, 1001, 1100, 1110, 1111.

2.2.1 Διάσπαση των λέξεων χωρίς zig-zag

Πρόταση 2.2.1. *Κάθε μη κενή δυαδική λέξη α χωρίς zig-zag διασπάται κατά μοναδικό τρόπο υπό τη μορφή*

$$\alpha = 0,\ \acute{\eta}\ \alpha = 1,\ \acute{\eta}\ \alpha = 01,\ \acute{\eta}\ \alpha = 10,\ \acute{\eta}\ \alpha = 0\beta,\ \acute{\eta}\ \alpha = 01\gamma,\ \acute{\eta}\ \alpha = 10\beta,\ \acute{\eta}\ \alpha = 1\gamma,$$

όπου β, γ είναι δυαδικές λέξεις χωρίς zig-zag οι οποίες αρχίζουν με 0 και 1 αντίστοιχα.

Απόδειξη. Κάθε μη κενή δυαδική λέξη α χωρίς zig-zag είτε αρχίζει με 0 είτε αρχίζει με 1.

Στην περίπτωση όπου $\alpha = 0\delta$, πρέπει η δ να μην αρχίζει από 10 και να μην περιέχει zig-zag, δηλαδή $\delta = \varepsilon$, ή δ αρχίζει από 0, ή $\delta = 1$, ή δ αρχίζει από 11, άρα

$$\alpha = 0,\ \acute{\eta}\ \alpha = 0\beta,\ \acute{\eta}\ \alpha = 01,\ \acute{\eta}\ \alpha = 01\gamma.$$

όπου β, γ είναι δυαδικές λέξεις χωρίς zig-zag οι οποίες αρχίζουν από 0 και 1 αντίστοιχα.

Στην περίπτωση όπου $\alpha = 1\delta$, πρέπει η δ να μην αρχίζει από 01 και να μην περιέχει zig-zag, δηλαδή $\delta = \varepsilon$, ή δ αρχίζει από 1, ή $\delta = 0$, ή δ αρχίζει από 00, άρα

$$\alpha = 1, \text{ ή } \alpha = 1\gamma, \text{ ή } \alpha = 10, \text{ ή } \alpha = 10\beta.$$

όπου β, γ είναι δυαδικές λέξεις χωρίς zig-zag οι οποίες αρχίζουν από 0 και 1 αντίστοιχα. □

2.2.2 Απαρίθμηση ως προς το μήκος και τον αριθμό των μονάδων

Πρόταση 2.2.2.

i) Ο αριθμός των δυαδικών λέξεων μήκους n χωρίς zig-zag ισούται με $2F_n$, όπου F_n είναι ο n-οστός αριθμός Fibonacci.

ii) Ο αριθμός $a_{n,k}$ των δυαδικών λέξεων μήκους n χωρίς zig-zag και με k μονάδες ικανοποιεί την αναγωγική σχέση

$$\alpha_{n,k} - \alpha_{n-1,k} - \alpha_{n-1,k-1} + \alpha_{n-2,k-1} - \alpha_{n-4,k-2} = 0, \quad n \geq 4 \text{ και } k \leq 2,$$

όπου $\alpha_{n,k} = 0$ για κάθε $n < k < 0$ και $\alpha_{n,0} = 1$, $\alpha_{n,1} = 2 - \delta_{n,1}$ για κάθε $n \geq 1$ και

$$\delta_{i,j} = \begin{cases} 1, & \text{αν } i = j, \\ 0, & \text{αν } i = j, \end{cases}$$

η γεννήτρια συνάρτηση του Kronecker.

Απόδειξη. Έστω $F(x, y)$ η γεννήτρια συνάρτηση του συνόλου $\mathcal{Z}$ των λέξεων χωρίς zig-zag ως προς τις παραμέτρους μήκος της λέξης και αριθμός εμφανίσεων του 1, οπότε

$$F(x, y) = \sum_{\alpha \in \mathcal{Z}} x^{|\alpha|} y^{|\alpha|_1}.$$

Έστω Z_0 (αντίστοιχα Z_1) το υποσύνολο του $\mathcal{Z}$ τα στοιχεία του οποίου αρχίζουν με 0 (αντίστοιχα 1) και $F_0(x, y)$ (αντίστοιχα $F_1(x, y)$) η γεννήτρια συνάρτησή του. Προφανώς, $\mathcal{Z} = Z_0 \cup Z_1 \cup \varepsilon$ και $Z_0 \cap Z_1 = \varepsilon$.

Από την προηγούμενη πρόταση, κάθε μη κενή λέξη χωρίς zig-zag διασπάται κατά μοναδικό τρόπο υπό την μορφή

$$\alpha = 0, \text{ ή } \alpha = 1, \text{ ή } \alpha = 01, \text{ ή } \alpha = 10, \text{ ή } \alpha = 0\beta, \text{ ή } \alpha = 01\gamma, \text{ ή } \alpha = 10\beta, \text{ ή } \alpha = 1\gamma,$$

όπου β, γ είναι δυαδικές λέξεις χωρίς zig-zag οι οποίες αρχίζουν με 0 και 1 αντίστοιχα.

Για τις παράμετρους $|\ |, |\ |_1 : \mathcal{Z} \to \mathbb{N}$ ισχύει ότι

$$|\varepsilon| = 0,\ |1| = 1,\ |01| = |10| = 2,\ |0\beta| = 1+|\beta|,\ |01\gamma| = 2+|\gamma|,\ |10\beta| = 2+|\beta|,\ |1\gamma| = 1+|\gamma|$$

και

$$|\varepsilon|_1 = 0,\ |1|_1 = |01|_1 = |10|_1 = 1,\ |0\beta|_1 = |\beta|_1,\ |01\gamma|_1 = 1+|\gamma|_1,\ |10\beta|_1 = 1+|\beta|_1,\ |1\gamma|_1 = 1+|\gamma|_1.$$

Ισχύει ότι

$$\begin{aligned}
F(x,y) &= \sum_{\alpha\in Z} x^{|\alpha|}y^{|\alpha|_1} \\
&= \sum_{\alpha=\varepsilon} x^{|\varepsilon|}y^{|\varepsilon|_1} + \sum_{\alpha=0} x^{|0|}y^{|0|_1} + \sum_{\alpha=1} x^{|1|}y^{|1|_1} + \sum_{\alpha=01} x^{|01|}y^{|01|_1} + \sum_{\alpha=10} x^{|10|}y^{|10|_1} + \sum_{\substack{\alpha=0\beta\\ \beta\in Z_0}} x^{|0\beta|}y^{|0\beta|_1} \\
&\quad + \sum_{\substack{\alpha=01\gamma\\ \gamma\in Z_1}} x^{|01\gamma|}y^{|01\gamma|_1} + \sum_{\substack{\alpha=10\beta\\ \beta\in Z_0}} x^{|10\beta|}y^{|10\beta|_1} + \sum_{\substack{\alpha=1\gamma\\ \gamma\in Z_1}} x^{|1\gamma|}y^{|1\gamma|_1} \\
&= 1 + x + xy + 2x^2y + x\sum_{\beta\in Z_0} x^{|\beta|}y^{|\beta|_1} + x^2y\sum_{\gamma\in Z_1} x^{|\gamma|}y^{|\gamma|_1} + x^2y\sum_{\beta\in Z_0} x^{|\beta|}y^{|\beta|_1} + xy\sum_{\gamma\in Z_1} x^{|\gamma|}y^{|\gamma|_1} \\
&= 1 + x + xy + 2x^2y + (x+x^2y)F_0(x,y) + (x^2y+xy)F_1(x,y).
\end{aligned}$$

Άρα

$$F(x,y) = 1 + x + xy + 2x^2y + (x+x^2y)F_0(x,y) + (x^2y+xy)F_1(x,y). \tag{2.6}$$

Επιπλέον, κάθε λέξη που ανήκει στο σύνολο Z_0 γράφεται υπό τη μορφή

$$\alpha = 0,\ \text{ή}\ \alpha = 0\beta,\ \text{ή}\ \alpha = 01\gamma,$$

οπότε

$$\begin{aligned}
F_0(x,y) &= \sum_{\alpha\in Z_0} x^{|\alpha|}y^{|\alpha|_1} \\
&= \sum_{\alpha=0} x^{|0|}y^{|0|_1} + \sum_{\substack{\alpha=0\beta\\ \beta\in Z_0}} x^{|0\beta|}y^{|0\beta|_1} + \sum_{\substack{\alpha=01\gamma\\ \gamma\in Z_1}} x^{|01\gamma|}y^{|01\gamma|_1} \\
&= x + x\sum_{\beta\in Z_0} x^{|\beta|}y^{|\beta|_1} + x^2y\sum_{\gamma\in Z_1} x^{|\gamma|}y^{|\gamma|_1} \\
&= x + xF_0(x,y) + x^2yF_1(x,y).
\end{aligned}$$

Άρα

$$F_0(x,y) = x + xF_0(x,y) + x^2yF_1(x,y). \tag{2.7}$$

Επιπρόσθετα, ισχύει ότι

$$F(x,y) = 1 + F_0(x,y) + F_1(x,y). \tag{2.8}$$

Συνεπώς, από τις σχέσεις (2.6), (2.7) και (2.8), προκύπτει μετά από πράξεις ότι

$$F(x,y) = \frac{1 + x^2y + x^4y^2}{1 - x - xy + x^2y - x^4y^2}. \tag{2.9}$$

i) Αν στην (2.9) τεθεί $y = 1$, προκύπτει ότι

$$\begin{aligned} F(x,1) &= \frac{1 + x^2 + x^4}{1 - 2x + x^2 - x^4} \\ &= \frac{2(1 - x + x^2)}{(1 - x + x^2)(1 - x - x^2)} - 1 \\ &= \frac{2}{1 - x - x^2} - 1. \end{aligned}$$

Επομένως,

$$\begin{aligned} F(x,1) &= \frac{2}{1 - x - x^2} - \frac{1 - x - x^2}{1 - x - x^2} \\ &= \frac{2x(1 + x) + 1 - x - x^2}{1 - x - x^2} \\ &= 2xG(x,1) + 1 \end{aligned}$$

όπου $G(x,1)$ είναι η γεννήτρια συνάρτηση των λέξεων Fibonacci ως προς την παράμετρο μήκος (βλ. σχέση (2.2)).

Άρα

$$\begin{aligned} [x^n]F(x,1) &= [x^n](2xG(x,1) + 1) \\ &= [x^n]\left(\sum_{n=0}^{\infty} 2F_{n+1}x^{n+1} + 1\right) \\ &= [x^n]\left(\sum_{n=1}^{\infty} 2F_n x^n + 1\right) \\ &= \begin{cases} 2F_n, & \text{αν } n \geq 1\ , \\ 1, & \text{αν } n = 0\ . \end{cases} \end{aligned}$$

Συνεπώς, ο αριθμός των δυαδικών λέξεων μήκους n χωρίς zig-zag ισούται με τον $2F_n$.

ii) Από τη σχέση (2.9) προκύπτει ότι

$$(1 - x - xy + x^2y - x^4y^2)F(x,y) = 1 + x^2y + x^4y^2$$

οπότε, αν $F(x,y) = \sum_{n=0}^{\infty}\sum_{k=0}^{n} a_{n,k}x^n y^k$, τότε

$$\sum_{n=0}^{\infty}\sum_{k=0}^{n} a_{n,k}x^n y^k - \sum_{n=0}^{\infty}\sum_{k=0}^{n} a_{n,k}x^{n+1} y^k - \sum_{n=0}^{\infty}\sum_{k=0}^{n} a_{n,k}x^{n+1} y^{k+1} + \sum_{n=0}^{\infty}\sum_{k=0}^{n} a_{n,k}x^{n+2} y^{k+1} - \sum_{n=0}^{\infty}\sum_{k=0}^{n} a_{n,k}x^{n+4} y^{k+2} = 1 + x^2y + x^4y^2$$

ή, ισοδύναμα,

$$\sum_{n=0}^{\infty}\sum_{k=0}^{n} a_{n,k}x^n y^k - \sum_{n=1}^{\infty}\sum_{k=0}^{n-1} a_{n-1,k}x^n y^k - \sum_{n=1}^{\infty}\sum_{k=1}^{n} a_{n-1,k-1}x^n y^k + \sum_{n=2}^{\infty}\sum_{k=1}^{n-1} a_{n-2,k-1}x^n y^k - \sum_{n=4}^{\infty}\sum_{k=2}^{n-2} a_{n-4,k-2}x^n y^k = 1 + x^2y + x^4y^2.$$

Μετά από πράξεις, προκύπτει ότι

$$\sum_{n=4}^{\infty}\sum_{k=0}^{n} a_{n,k}x^n y^k - \sum_{n=4}^{\infty}\sum_{k=0}^{n-1} a_{n-1,k}x^n y^k - \sum_{n=4}^{\infty}\sum_{k=1}^{n} a_{n-1,k-1}x^n y^k + \sum_{n=4}^{\infty}\sum_{k=1}^{n-1} a_{n-2,k-1}x^n y^k - \sum_{n=4}^{\infty}\sum_{k=2}^{n-2} a_{n-4,k-2}x^n y^k = x^4y^2$$

ή, ισοδύναμα,

$$\sum_{n=4}^{\infty}\sum_{k=2}^{n-2} a_{n,k}x^n y^k - \sum_{n=4}^{\infty}\sum_{k=2}^{n-2} a_{n-1,k}x^n y^k - \sum_{n=4}^{\infty}\sum_{k=2}^{n-2} a_{n-1,k-1}x^n y^k + \sum_{n=4}^{\infty}\sum_{k=2}^{n-2} a_{n-2,k-1}x^n y^k - \sum_{n=4}^{\infty}\sum_{k=2}^{n-2} a_{n-4,k-2}x^n y^k = x^4y^2.$$

Επομένως,

$$\sum_{n=4}^{\infty}\sum_{k=2}^{n-2} \left(a_{n,k} - a_{n-1,k} - a_{n-1,k-1} + a_{n-2,k-1} - a_{n-4,k-2}\right) x^n y^k = x^4y^2$$

Συνεπώς,

$$a_{n,k} - a_{n-1,k} - a_{n-1,k-1} + a_{n-2,k-1} - a_{n-4,k-2} = 0,$$

για κάθε $n \geq 4$ και $k \geq 2$.

□

Οι πρώτοι όροι της ακολουθίας $a_{n,k}$ δίδονται στον επόμενο πίνακα.

$n \setminus k$	0	1	2	3	4	5	6	7	8	9	10	
0	1											1
1	1	1										2
2	1	2	1									4
3	1	2	2	1								6
4	1	2	4	2	1							10
5	1	2	5	5	2	1						16
6	1	2	6	8	6	2	1					26
7	1	2	7	11	11	7	2	1				42
8	1	2	8	14	18	14	8	2	1			68
9	1	2	9	17	26	26	17	9	2	1		110
10	1	2	10	20	35	42	35	20	10	2	1	178

Παράδειγμα. Το πλήθος των δυαδικών λέξεων μήκος 7 χωρίς zig-zag, με 4 μονάδες ισούται με 11. Πράγματι, οι λέξεις αυτές είναι οι ακόλουθες:

0001111 0011110 0011011 0110011 0111001 0111100
1000111 1100011 1100110 1110001 1111000

Από την προηγούμενη πρόταση παρατηρούμε ότι ο αριθμός των δυαδικών λέξεων μήκους $n+1$ χωρίς zig-zag ισούται με το διπλάσιο του αριθμού των λέξεων Fibonacci μήκους n.

Παρακάτω θα δοθεί μία συνδυαστική ερμηνεία αυτού του αποτελέσματος χρησιμοποιώντας δύο αμφιμονοσήμαντες απεικονίσεις.

Ορίζουμε τις απεικονίσεις $\phi_0, \phi_1 : \Phi \to \mathcal{Z}$ αναδρομικά ως εξής

$$\phi_0(\varepsilon) = 0 \text{ και } \phi_1(\varepsilon) = 1$$
$$\phi_0(0) = 01 \text{ και } \phi_1(0) = 10$$
$$\phi_0(1\beta) = 0\phi_0(\beta) \text{ και } \phi_1(1\beta) = 1\phi_1(\beta)$$
$$\phi_0(01\beta) = 01\phi_1(\beta) \text{ και } \phi_1(01\beta) = 10\phi_0(\beta)$$

Παράδειγμα. Για τις λέξεις Fibonacci μήκους 4:

$$0101, \quad 0110, \quad 0111, \quad 1010, \quad 1011, \quad 1101, \quad 1110, \quad 1111,$$

ισχύει ότι

$$\begin{aligned}
&\phi_0(0101) = 01\phi_1(01) = 0110\phi_0(\varepsilon) = 01100,\\
&\phi_0(0110) = 01\phi_1(10) = 011\phi_1(0) = 01110,\\
&\phi_0(0111) = 01\phi_1(11) = 011\phi_1(1) = 0111\phi_1(\varepsilon) = 01111,\\
&\phi_0(1010) = 0\phi_0(010) = 001\phi_1(0) = 00110,\\
&\phi_0(1011) = 0\phi_0(011) = 001\phi_1(1) = 0011\phi_1(\varepsilon) = 00111,\\
&\phi_0(1101) = 0\phi_0(101) = 00\phi_0(01) = 0001\phi_1(\varepsilon) = 00011,\\
&\phi_0(1110) = 0\phi_0(110) = 00\phi_0(10) = 000\phi_0(0) = 00001,\\
&\phi_0(1111) = 0\phi_0(111) = 00\phi_0(11) = 000\phi_0(1) = 0000\phi_0(\varepsilon) = 00000,
\end{aligned}$$

και

$$\begin{aligned}
&\phi_1(0101) = 10\phi_0(01) = 1001\phi_1(\varepsilon) = 10011,\\
&\phi_1(0110) = 10\phi_0(10) = 100\phi_0(0) = 10001,\\
&\phi_1(0111) = 10\phi_0(11) = 100\phi_0(1) = 1000\phi_0(\varepsilon) = 10000,\\
&\phi_1(1010) = 1\phi_1(010) = 110\phi_0(0) = 11001,\\
&\phi_1(1011) = 1\phi_1(011) = 110\phi_0(1) = 1100\phi_0(\varepsilon) = 11000,\\
&\phi_1(1101) = 1\phi_1(101) = 11\phi_1(01) = 1110\phi_0(\varepsilon) = 11100,\\
&\phi_1(1110) = 1\phi_1(110) = 11\phi_1(10) = 111\phi_1(0) = 11110,\\
&\phi_1(1111) = 1\phi_1(111) = 11\phi_1(11) = 111\phi_1(1) = 1111\phi_1(\varepsilon) = 11111,
\end{aligned}$$

δηλαδή οι εικόνες των λέξεων Fibonacci μήκους 4 είναι όλες οι δυαδικές λέξεις μήκους 5 χωρίς zig-zag.

Επίσης, για κάθε $\alpha \in \mathcal{Z}$, συμβολίζουμε με $r(\alpha)$ τον αριθμό των **εναλλαγών** στην α, δηλαδή τον αριθμό των εμφανίσεων των 01 και 10 στην α. Για παράδειγμα, για την λέξη 111001010, είναι $r(\alpha) = 5$.

Πρόταση 2.2.3.

i) Η απεικόνιση ϕ_0 (αντίστοιχα ϕ_1) στέλνει τις λέξεις Fibonacci μήκους n σε λέξεις μήκους $n+1$ χωρίς zig-zag, οι οποίες αρχίζουν από 0 (αντίστοιχα από 1).

ii) Οι ϕ_0, ϕ_1 είναι αμφιμονοσήμαντες απεικονίσεις.

iii) $r(\phi_0(\alpha)) = r(\phi_1(\alpha)) = |\alpha|_0$ για κάθε λέξη Fibonacci α.

Απόδειξη.

i) Θα χρησιμοποιηθεί επαγωγή ως προς το μήκος n της λέξης.

Για $n = 0$ είναι $\phi_0(\varepsilon) = 0$ και $\phi_1(\varepsilon) = 1$ και για $n = 1$ είναι $\phi_0(0) = 01$ και $\phi_1(0) = 10$, άρα ισχύει.

Έστω ότι ισχύει για κάθε $k \leq n$, θα αποδειχθεί ότι ισχύει και για $n + 1$.

Έστω α μια λέξη Fibonacci μήκους $n + 1$. Διακρίνουμε δύο περιπτώσεις:

1. Αν $\alpha = 1\beta$, όπου β μία λέξη Fibonacci μήκους n, τότε

$$\phi_0(1\beta) = 0\phi_0(\beta) \text{ και } \phi_1(1\beta) = 1\phi_1(\beta).$$

Από την υπόθεση της επαγωγής, η λέξη $\phi_0(\beta)$ (αντίστοιχα $\phi_1(\beta)$) είναι μια λέξη μήκους $n + 1$ χωρίς zig-zag η οποία αρχίζει από 0 (αντίστοιχα από 1).

Επομένως, η λέξη $0\phi_0(\beta)$ (αντίστοιχα $1\phi_1(\beta)$) έχει μήκος $n + 2$, αρχίζει επίσης από 0 (αντίστοιχα από 1) και άρα δεν περιέχει zig-zag, οπότε η πρόταση ισχύει.

2. Αν $\alpha = 01\beta$, όπου β μία λέξη Fibonacci μήκους $n - 1$, τότε

$$\phi_0(01\beta) = 01\phi_1(\beta) \text{ και } \phi_1(01\beta) = 10\phi_0(\beta).$$

Από την υπόθεση της επαγωγής, η λέξη $\phi_0(\beta)$ (αντίστοιχα $\phi_1(\beta)$) είναι μια λέξη μήκους n χωρίς zig-zag η οποία αρχίζει από 0 (αντίστοιχα από 1).

Επομένως, η λέξη $01\phi_1(\beta)$ (αντίστοιχα $10\phi_0(\beta)$) έχει μήκος $n+2$, αρχίζει επίσης από 0 (αντίστοιχα από 1) και άρα δεν περιέχει zig-zag, αφού η $\phi_1(\beta)$ (αντίστοιχα $\phi_0(\beta)$) αρχίζει από 1 (αντίστοιχα από 0), οπότε η πρόταση ισχύει.

ii) Πρέπει να δειχθεί ότι κάθε $\alpha \in \mathcal{Z}$ είναι εικόνα μίας μοναδικής λέξης μέσω της ϕ_0, και μίας μοναδικής λέξης μέσω της ϕ_1. Θα χρησιμοποιηθεί και πάλι επαγωγή ως προς το μήκος n της λέξης α.

Για $n = 1$, υπάρχει μία λέξη μήκους 1 που αρχίζει από 0 (αντίστοιχα από 1) χωρίς zig-zag, η 0 (αντίστοιχα η 1), για την οποία είναι $0 = \phi_0(\varepsilon)$ (αντίστοιχα $1 = \phi_1(\varepsilon)$), άρα ισχύει.

Για $n = 2$, υπάρχει μία λέξη μήκους 2 που αρχίζει από 0 (αντίστοιχα από 1) χωρίς zig-zag, η 01 (αντίστοιχα η 10), για την οποία είναι $01 = \phi_0(0)$ (αντίστοιχα $10 = \phi_1(0)$), άρα ισχύει.

Έστω ότι ισχύει για κάθε $k \leq n$, θα αποδειχθεί ότι ισχύει και για $n + 1$.

Έστω α (αντίστοιχα α') μία λέξη μήκους $n+1$ που αρχίζει από 0 (αντίστοιχα από 1) χωρίς zig-zag. Διακρίνουμε δύο περιπτώσεις:

1. Αν $\alpha = 0\beta$ (αντίστοιχα $\alpha^{'} = 0\beta^{'}$) όπου β μία λέξη μήκους n η οποία αρχίζει από 0 (αντίστοιχα από 1), τότε, από την υπόθεση της επαγωγής, υπάρχει μοναδική λέξη Fibonacci γ (αντίστοιχα $\gamma^{'}$) μήκους $n-1$ με $\phi_0(\gamma) = \beta$ (αντίστοιχα $\phi_1(\gamma) = \beta^{'}$). Άρα,

$$\alpha = 0\phi_0(\gamma) = \phi_0(1\gamma) \text{ (αντίστοιχα } \alpha^{'} = 1\phi_1(\gamma^{'}) = \phi_1(1\gamma^{'})).$$

Έστω ότι υπάρχει λέξη Fibonacci δ (αντίστοιχα $\delta^{'}$) μήκους n, τέτοια ώστε $\phi_0(\delta) = \alpha$ (αντίστοιχα $\phi_1(\delta^{'}) = \alpha^{'}$), τότε $\phi_0(\delta) = 0\phi_0(\gamma)$ (αντίστοιχα $\phi_1(\delta^{'}) = 1\phi_1(\gamma)$). Αφού $0\phi_0(\gamma)$ (αντίστοιχα $1\phi_1(\gamma)$) αρχίζει από 00 (αντίστοιχα 11), πρέπει δ (αντίστοιχα $\delta^{'}$) να αρχίζει από 1, οπότε $\delta = 1\zeta$ (αντίστοιχα $\delta^{'} = 1\zeta^{'}$).

Επομένως,

$$\begin{aligned}
&\phi_0(\delta) = \phi_0(1\zeta) = 0\phi_0(\zeta) = 0\phi_0(\gamma) \Rightarrow \\
&\phi_0(\zeta) = \phi_0(\gamma) \Leftrightarrow \\
&\zeta = \gamma \Leftrightarrow \\
&\delta = 1\gamma
\end{aligned}$$

(αντίστοιχα

$$\begin{aligned}
&\phi_1(\delta') = \phi_1(1\zeta') = 1\phi_1(\zeta') = 1\phi_1(\gamma') \Rightarrow \\
&\phi_1(\zeta') = \phi_1(\gamma') \Leftrightarrow \\
&\zeta' = \gamma' \Leftrightarrow \\
&\delta^{'} = 1\gamma^{'})
\end{aligned}$$

και άρα, πράγματι, η λέξη της οποίας η α είναι εικόνα μέσω της ϕ_0 (αντίστοιχα της ϕ_1) είναι μοναδική.

2. Αν $\alpha = 01\beta$ (αντίστοιχα $\alpha^{'} = 10\beta^{'}$) όπου β μία λέξη μήκους $n-1$ η οποία αρχίζει από 1 (αντίστοιχα από 0), τότε, από την υπόθεση της επαγωγής, υπάρχει μοναδική λέξη Fibonacci γ (αντίστοιχα $\gamma^{'}$) μήκους $n-2$ με $\phi_1(\gamma) = \beta$ (αντίστοιχα $\phi_0(\gamma^{'}) = \beta^{'}$).

Άρα,

$$\alpha = 01\phi_1(\gamma) = \phi_0(01\gamma) \text{ (αντίστοιχα } \alpha^{'} = 10\phi_0(\gamma^{'}) = \phi_1(01\gamma^{'})).$$

Έστω ότι υπάρχει λέξη Fibonacci δ (αντίστοιχα $\delta^{'}$) μήκους n, τέτοια ώστε $\phi_0(\delta) = \alpha$ (αντίστοιχα $\phi_1(\delta^{'}) = \alpha^{'}$), τότε $\phi_0(\delta) = 01\phi_1(\gamma)$ (αντίστοιχα $\phi_1(\delta^{'}) = 10\phi_0(\gamma^{'})$). Αφού $01\phi_1(\gamma)$ (αντίστοιχα $10\phi_0(\gamma^{'})$) αρχίζει από 001 (αντίστοιχα 100), πρέπει δ (αντίστοιχα $\delta^{'}$) να αρχίζει από 01, οπότε $\delta = 01\zeta$ (αντίστοιχα $\delta^{'} = 01\zeta^{'}$).

Επομένως,

$$\begin{aligned}
&\phi_0(\delta) = \phi_0(01\zeta) = 01\phi_1(\zeta) = 01\phi_1(\gamma) \Rightarrow \\
&\phi_1(\zeta) = \phi_1(\gamma) \Leftrightarrow \\
&\zeta = \gamma \Leftrightarrow \\
&\delta = 01\gamma
\end{aligned}$$

(αντίστοιχα

$$\begin{aligned}
&\phi_1(\delta') = \phi_1(01\zeta') = 10\phi_0(\zeta') = 10\phi_0(\gamma') \Rightarrow \\
&\phi_1(\zeta') = \phi_1(\gamma') \Leftrightarrow \\
&\zeta' = \gamma' \Leftrightarrow \\
&\delta^{'} = 01\gamma^{'})
\end{aligned}$$

και άρα, πράγματι, και στην περίπτωση αυτή, η λέξη της οποίας η α είναι εικόνα μέσω της ϕ_0 (αντίστοιχα της ϕ_1) είναι μοναδική.

Άρα η ϕ_0 (αντίστοιχα της ϕ_1) είναι 1-1 και επί.

iii) Θα χρησιμοποιηθεί επίσης επαγωγή ως προς το μήκος n της λέξης.

Για $n = 0$, είναι

$$r(\phi_0(\varepsilon)) = r(0) = 0 = |\varepsilon|_0 \text{ και } r(\phi_1(\varepsilon)) = r(1) = 0 = |\varepsilon|_0.$$

Για $n = 1$, είναι

$$r(\phi_0(0)) = r(01) = 1 = |0|_0 \text{ και } r(\phi_1(0)) = r(10) = 1 = |0|_0.$$

Έστω ότι ισχύει για κάθε $k \leq n$, θα αποδειχθεί ότι ισχύει και για $n+1$. Έστω α μια λέξη Fibonacci μήκους $n+1$. Διακρίνουμε δύο περιπτώσεις:

1. Αν $\alpha = 1\beta$, όπου β μια λέξη Fibonacci μήκους n, τότε

$$r(\phi_0(\alpha)) = r(\phi_0(1\beta)) = r(0\phi_0(\beta)) = r(\phi_0(\beta)) = |\beta|_0 = |1\beta|_0 = |\alpha|_0 \text{ και}$$
$$r(\phi_1(\alpha)) = r(\phi_1(1\beta)) = r(1\phi_1(\beta)) = r(\phi_1(\beta)) = |\beta|_0 = |1\beta|_0 = |\alpha|_0,$$

οπότε ισχύει.

2. Αν $\alpha = 01\beta$, όπου β μια λέξη Fibonacci μήκους $n-1$, τότε

$$r(\phi_0(\alpha)) = r(\phi_0(01\beta)) = r(01\phi_1(\beta)) = 1 + r(\phi_1(\beta)) = 1 + |\beta|_0 = |01\beta|_0 = |\alpha|_0 \text{ και}$$
$$r(\phi_1(\alpha)) = r(\phi_1(01\beta)) = r(10\phi_0(\beta)) = 1 + r(\phi_0(\beta)) = 1 + |\beta|_0 = |01\beta|_0 = |\alpha|_0,$$

οπότε ισχύει.

□

Πόρισμα 2.2.4. *Ο αριθμός των δυαδικών λέξεων μήκους n χωρίς zig-zag με k εναλλαγές ισούται με*

$$2\binom{n-k}{k}.$$

Απόδειξη. Έστω $a_{n,k}$ ο αριθμός των δυαδικών λέξεων μήκους n χωρίς zig-zag, με k εναλλαγές. Από την προηγούμενη πρόταση, ισχύει ότι η ϕ_0 (αντ. ϕ_1) στέλνει τις λέξεις Fibonacci μήκους n στις δυαδικές λέξεις μήκους $n+1$ χωρίς zig-zag που αρχίζουν με 0 (αντ. 1) και για κάθε λέξη Fibonacci α, ο αριθμός των εναλλαγών της λέξης $\phi_0(\alpha)$ (αντ. $\phi_1(\alpha)$) ισούται με τον αριθμό των μηδενικών της λέξης α, δηλαδή

$$r(\phi_0(\alpha)) = r(\phi_1(\alpha)) = |a|_0.$$

Επιπλέον, από το Πόρισμα 2.1.3, ισχύει ότι ο αριθμός των λέξεων Fibonacci μήκους n με k μηδενικά ισούται με

$$\binom{n-k+1}{k}.$$

Επομένως,

$$\begin{aligned}
a_{n,k} &= |\{\beta \in Z : |\beta| = n \text{ και } r(\beta) = k\}| \\
&= |\{\alpha \in \Phi : |\phi_0(\alpha)| = n, r(\phi_0(\alpha)) = k\}| + |\{\alpha \in \Phi : |\phi_1(\alpha)| = n, r(\phi_1(\alpha)) = k\}| \\
&= |\{\alpha \in \Phi : |\alpha| = n-1, |a|_0 = k\}| + |\{\alpha \in \Phi : |\alpha| = n-1, |a|_0 = k\} \\
&= \binom{n-1-k+1}{k} + \binom{n-1-k+1}{k} \\
&= 2\binom{n-k}{k}.
\end{aligned}$$

□

Παρατηρήσεις.

i) Από την προηγούμενη πρόταση προκύπτει ότι για κάθε εμφάνιση μηδενικού σε μια λέξη Fibonacci α, οι απεικονίσεις ϕ_0, ϕ_1 εναλλάσσουν τα αντίστοιχα ψηφία των λέξεων $\phi_0(\alpha)$, $\phi_1(\alpha)$. Με βάση αυτή την παρατήρηση οι ϕ_0, ϕ_1 μπορούν να ορισθούν επαναληπτικά ως εξής: Έστω $\alpha = \alpha_1\alpha_2\cdots\alpha_n$ μια λέξη Fibonacci μήκους n. Τότε, $\phi_0(\alpha) = 0\beta_1\beta_2\cdots\beta_n$ και $\phi_1(\alpha) = 1\gamma_1\gamma_2\cdots\gamma_n$, όπου

$$\beta_1 = \begin{cases} 1, & \text{αν } \alpha_1 = 0, \\ 0, & \text{αν } \alpha_1 = 1, \end{cases} \qquad \gamma_1 = \begin{cases} 0, & \text{αν } \alpha_1 = 0, \\ 1, & \text{αν } \alpha_1 = 1, \end{cases}$$

και

$$\beta_i = \begin{cases} 1-\beta_{i-1}, & \text{αν } \alpha_i = 0, \\ \beta_{i-1}, & \text{αν } \alpha_i = 1, \end{cases} \qquad \gamma_i = \begin{cases} 1-\gamma_{i-1}, & \text{αν } \alpha_i = 0, \\ \gamma_{i-1}, & \text{αν } \alpha_i = 1, \end{cases}$$

για κάθε $i \in [2, n]$.

ii) Η ϕ_0 (αντίστοιχα ϕ_1) στέλνει τις λέξεις Fibonacci οι οποίες αρχίζουν από 1 ή από 01 στις λέξεις χωρίς zig-zag οι οποίες αρχίζουν από 00 ή από 011 (αντίστοιχα από 11 ή από 100) αντίστοιχα.

iii) Η λέξη $\phi_1(\alpha)$ είναι το συμπλήρωμα της λέξης $\phi_0(\alpha)$.

Για κάθε λέξη Fibonacci α, συμβολίζουμε με $\Upsilon^{\phi_0}(\alpha)$ (αντίστοιχα $\Upsilon^{\phi_1}(\alpha)$) τον αριθμό των βημάτων για τον αναδρομικό υπολογισμό της α μέσω της ϕ_0 (αντιστοιχα ϕ_1). Τότε, ισχύει η επόμενη πρόταση.

Πρόταση 2.2.5.

$$\Upsilon^{\phi_0}(\alpha) = \Upsilon^{\phi_1}(\alpha) = |\alpha|_1 + 1.$$

Απόδειξη. Θα χρησιμοποιηθεί επαγωγή ως προς το μήκος n της λέξης.

Για $n = 0$, είναι

$$\Upsilon^{\phi_0}(\varepsilon) = 1 = |\varepsilon|_1 + 1 \text{ και } \Upsilon^{\phi_1}(\varepsilon) = 1 = |\varepsilon|_1 + 1.$$

Για $n = 1$, είναι

$$\Upsilon^{\phi_0}(0) = 1 = |0|_1 + 1 \text{ και } \Upsilon^{\phi_1}(0) = 1 = |0|_1 + 1.$$

Έστω ότι ισχύει για κάθε $k \leq n$. Θα αποδειχθεί ότι ισχύει και για $n + 1$. Έστω α μια λέξη Fibonacci μήκους $n + 1$. Διακρίνουμε δύο περιπτώσεις:

1. Αν $\alpha = 1\beta$ όπου β είναι μια λέξη Fibonacci μήκους n, τότε

   $$\Upsilon^{\phi_0}(\alpha) = \Upsilon^{\phi_0}(1\beta) = \Upsilon^{\phi_0}(\beta) + 1 = |\beta|_1 + 1 + 1 = |1\beta|_1 + 1 = |\alpha|_1 + 1$$

   και

   $$\Upsilon^{\phi_1}(\alpha) = \Upsilon^{\phi_1}(1\beta) = \Upsilon^{\phi_1}(\beta) + 1 = |\beta|_1 + 1 + 1 = |1\beta|_1 + 1 = |\alpha|_1 + 1,$$

   οπότε ισχύει.

2. Αν $\alpha = 01\beta$ όπου β είναι μια λέξη Fibonacci μήκους $n - 1$, τότε

   $$\Upsilon^{\phi_0}(\alpha) = \Upsilon^{\phi_0}(01\beta) = \Upsilon^{\phi_1}(\beta) + 1 = |\beta|_1 + 1 + 1 = |01\beta|_1 + 1 = |\alpha|_1 + 1$$

   και

   $$\Upsilon^{\phi_1}(\alpha) = \Upsilon^{\phi_1}(01\beta) = \Upsilon^{\phi_1}(\beta) + 1 = |\beta|_1 + 1 + 1 = |10\beta|_1 + 1 = |\alpha|_1 + 1,$$

   οπότε ισχύει.

□

Για περισσότερα στοιχεία σχετικά με τις δυαδικές λέξεις χωρίς zig-zag, βλέπε την εργασία των Munarini και Salvi [11] καθώς επίσης την ακολουθία A078678 στην Online Encyclopedia of Integer Sequences [13].

2.3 Λέξεις Dyck

Μια δυαδική λέξη $\alpha \in \{0,1\}^*$ ονομάζεται λέξη Dyck αν ικανοποιεί τις εξής δύο ιδιότητες:

i) $|\alpha|_1 = |\alpha|_0$,

ii) Για κάθε πρόθεμα β της α ισχύει ότι $|\beta|_1 \geq |\beta|_0$.

Παράδειγμα. Οι λέξεις Dyck μήκους το πολύ 8 είναι οι εξής: ε, 10, 1010, 1100, 101010, 101100, 110010, 110100, 111000, 10101010, 10101100, 10110010, 10110100, 10111000, 11001010, 11001100, 11010010, 11010100, 11011000, 11100010, 11100100, 11100100, 11101000, 11110000.

2.3.1 Διάσπαση των λέξεων Dyck

Πρόταση 2.3.1. *Κάθε μη κενή λέξη Dyck α διασπάται κατά μοναδικό τρόπο υπό τη μορφή*

$$\alpha = 1\beta 0\gamma$$

όπου β, γ λέξεις Dyck.

Απόδειξη. Έστω α μια μη κενή λέξη Dyck μήκους k. Προφανώς η α αρχίζει με 1. Επειδή ο αριθμός των μονάδων στην α ισούται με τον αριθμό των μηδενικών έπεται ότι η α έχει άρτιο μήκος, επομένως $k = 2n$ για κάποιο μη μηδενικό αριθμό n, οπότε

$$\alpha = \alpha_1\alpha_2\cdots\alpha_{2n}, \text{ όπου } \alpha_1 = 1.$$

Έστω j ο ελάχιστος μη μηδενικός φυσικός αριθμός στο $[n]$ τέτοιος ώστε

$$|\alpha_1\alpha_2\cdots\alpha_{2j}|_1 = |\alpha_1\alpha_2\cdots\alpha_{2j}|_0.$$

(Για $j = n$ ισχύει ότι $|\alpha|_0 = |\alpha|_1$, άρα πάντα ορίζεται κάποιο j.)

Τότε ισχύει ότι $\alpha_{2j} = 0$. Πράγματι, αν υποτεθεί ότι $\alpha_{2j} = 1$, τότε είναι

$$\begin{aligned} |\alpha_1\alpha_2\cdots\alpha_{2j-1}|_1 &= |\alpha_1\alpha_2\cdots\alpha_{2j-1}\alpha_{2j}|_1 - 1 \\ &= |\alpha_1\alpha_2\cdots\alpha_{2j-1}\alpha_{2j}|_0 - 1 \\ &< |\alpha_1\alpha_2\cdots\alpha_{2j-1}|_0, \end{aligned}$$

το οποίο είναι άτοπο διότι η λέξη $\alpha_1\alpha_2\cdots\alpha_{2j}$ είναι πρόθεμα της λέξης α.

Θα αποδειχθεί ότι οι λέξεις

$$\alpha_2\alpha_3\cdots\alpha_{2j-1} \text{ και } \alpha_{2j+1}\alpha_{2j+2}\cdots\alpha_{2n}$$

είναι λέξεις Dyck.

Πράγματι,

$$\begin{aligned}|\alpha_2\alpha_3\cdots\alpha_{2j-1}|_1 &= |\alpha_1\alpha_2\cdots\alpha_{2j-1}\alpha_{2j}|_1 - 1\\ &= |\alpha_1\alpha_2\cdots\alpha_{2j-1}\alpha_{2j}|_0 - 1\\ &= |\alpha_2\alpha_2\cdots\alpha_{2j-1}|_0.\end{aligned}$$

Επίσης, για κάθε $k \leq 2j-1$ ισχύει ότι

$$|\alpha_2\alpha_3\cdots\alpha_k|_1 \geq |\alpha_2\alpha_3\cdots\alpha_k|_0$$

Πράγματι, αν υποτεθεί ότι υπάρχουν $k \leq 2j-1$ με

$$|\alpha_2\alpha_3\cdots\alpha_k|_1 < |\alpha_2\alpha_3\cdots\alpha_k|_0,$$

έστω k_0 το ελάχιστο από αυτά. Τότε, πρέπει

$$|\alpha_2\alpha_3\cdots\alpha_{k_0}|_1 = |\alpha_2\alpha_3\cdots\alpha_{k_0}|_0 - 1.$$

Σ᾽ αυτή την περίπτωση όμως ισχύει ότι

$$\begin{aligned}|\alpha_1\alpha_2\cdots\alpha_{k_0}|_1 &= |\alpha_2\alpha_3\cdots\alpha_{k_0}|_1 + 1\\ &= |\alpha_2\alpha_3\cdots\alpha_{k_0}|_0\\ &= |\alpha_1\alpha_2\cdots\alpha_{k_0}|_0,\end{aligned}$$

το οποίο είναι άτοπο, διότι το j είναι το ελάχιστο στοιχείο του $[n]$ με αυτή την ιδιότητα.

Επομένως, η λέξη $\alpha_2\alpha_3\cdots\alpha_{2j-1}$ είναι λέξη Dyck και κατά συνέπεια και η λέξη $\alpha_1\alpha_2\cdots\alpha_{2j}$ είναι επίσης λέξη Dyck.

Επιπρόσθετα, ισχύει ότι

$$\begin{aligned}|\alpha_{2j+1}\alpha_{2j+2}\cdots a_{2n}|_1 &= |\alpha_1\alpha_2\cdots\alpha_{2n}|_1 - |\alpha_1\alpha_2\cdots\alpha_{2j}|_1\\ &= |\alpha_1\alpha_2\cdots\alpha_{2n}|_0 - |\alpha_1\alpha_2\cdots\alpha_{2j}|_0\\ &= |\alpha_{2j+1}\alpha_{2j+2}\cdots a_{2n}|_0.\end{aligned}$$

Επίσης, για κάθε $k \in [2j+1, 2n]$ ισχύει ότι

$$|\alpha_{2j+1}\alpha_{2j+2}\cdots\alpha_k|_1 \geq |\alpha_{2j+1}\alpha_{2j+2}\cdots\alpha_k|_0.$$

Πράγματι,

$$\begin{aligned}|\alpha_{2j+1}\alpha_{2j+2}\cdots a_k|_1 &= |\alpha_1\alpha_2\cdots\alpha_k|_1 - |\alpha_1\alpha_2\cdots\alpha_{2j}|_1\\ &= |\alpha_1\alpha_2\cdots\alpha_k|_1 - |\alpha_1\alpha_2\cdots\alpha_{2j}|_0\\ &\geq |\alpha_1\alpha_2\cdots\alpha_k|_0 - |\alpha_1\alpha_2\cdots\alpha_{2j}|_0\\ &= |\alpha_{2j+1}\alpha_{2j+2}\cdots a_k|_0.\end{aligned}$$

Τέλος, από τον ορισμό του j, οι λέξεις $\beta = \alpha_2\alpha_3\cdots\alpha_{2j-1}$ και $\gamma = \alpha_{2j+1}\alpha_{2j+2}\cdots\alpha_{2n}$ είναι μοναδικές. □

2.3.2 Απαρίθμηση ως προς το μήκος και τον αριθμό των εμφανίσεων του 10

Πρόταση 2.3.2.

i) Ο αριθμός των λέξεων Dyck μήκους $2n$ ισούται με τον n-οστό αριθμό Catalan.

ii) Ο αριθμός των λέξεων Dyck μήκους $2n$ με ακριβώς k εμφανίσεις του 10 ισούται με $\frac{1}{n}\binom{n}{k}\binom{n-1}{k}$.

Απόδειξη. Επειδή το μήκος σε κάθε λέξη Dyck ισούται με το διπλάσιο του αριθμού εμφανίσεων των μονάδων, θεωρούμε τη γεννήτρια συνάρτηση $F(x, y)$ των λέξεων Dyck ως προς τον αριθμό των εμφανίσεων του 1 και ως προς τον αριθμό εμφανίσεων του 10.

Από την προηγούμενη πρόταση, κάθε μη κενή λέξη Dyck διασπάται κατά μοναδικό τρόπο υπό τη μορφή

$$\alpha = 1\beta 0\gamma,$$

όπου β, γ λέξεις Dyck.

Για τις παραμέτρους $|\ |_1, |\ |_{10} : \mathcal{D} \to \mathbb{N}$ ισχύει ότι

$$|\varepsilon|_1 = 0, \qquad |1\beta 0\gamma|_1 = 1 + |\beta|_1 + |\gamma|_1$$

και

$$|\varepsilon|_{10} = 0, \qquad |1\beta 0\gamma|_{10} = |\beta|_{10} + |\gamma|_{10} + [\beta = \varepsilon].^1$$

[1] Συμβολισμός Iverson: $[P] = 1$, αν P αληθής, αλλιώς $[P] = 0$.

Ισχύει ότι

$$\begin{aligned}
F(x,y) &= \sum_{\alpha\in\mathcal{D}} x^{|\alpha|_1} y^{|\alpha|_{10}} \\
&= \sum_{\alpha=\varepsilon} x^{|\varepsilon|_1} y^{|\alpha|_{10}} + \sum_{\substack{\alpha=1\beta0\gamma \\ \beta,\gamma\in\mathcal{D}}} x^{|1\beta0\gamma|_1} y^{|1\beta0\gamma|_{10}} \\
&= 1 + \sum_{\beta,\gamma\in\mathcal{D}} x^{1+|\beta|_1+|\gamma|_1} y^{|\beta|_{10}+|\gamma|_{10}+[\beta=\varepsilon]} \\
&= 1 + x\sum_{\beta\in\mathcal{D}} x^{|\beta|_1} y^{|\beta|_{10}+[\beta=\varepsilon]} \sum_{\gamma\in\mathcal{D}} x^{|\gamma|_1} y^{|\gamma|_{10}} \\
&= 1 + x\Big(\sum_{\beta=\varepsilon} x^{|\varepsilon|_1} y^{|\varepsilon|_{10}+[\beta=\varepsilon]} + \sum_{\substack{\beta\in\mathcal{D} \\ \beta\neq\varepsilon}} x^{|\beta|_1} y^{|\beta|_{10}}\Big) F(x,y) \\
&= 1 + x\left(y + F(x,y) - 1\right) F(x,y).
\end{aligned}$$

Συνεπώς,

$$F(x,y) = 1 + x(y-1)F(x,y) + xF^2(x,y). \tag{2.10}$$

i) Αν στην (2.10) τεθεί $y = 1$, προκύπτει ότι

$$F(x,1) = 1 + xF^2(x,1).$$

Επιπλέον, αν τεθεί $H(\lambda) = \lambda^2$ τότε $F(x,1) = 1 + xH(F(x,1))$, οπότε, από το θεώρημα αντιστροφής του Lagrange, προκύπτει ότι

$$\begin{aligned}
[x^n]F(x,1) &= \frac{1}{n}[\lambda^{n-1}](H(\lambda+1))^n \\
&= \frac{1}{n}[\lambda^{n-1}](\lambda+1)^{2n} \\
&= \frac{1}{n}[\lambda^{n-1}]\sum_{j=0}^{2n}\binom{2n}{j}\lambda^j \\
&= \frac{1}{n}\binom{2n}{n-1}.
\end{aligned}$$

Επομένως,

$$\begin{aligned} F(x,1) &= 1+\sum_{n=1}^{\infty}\frac{1}{n}\binom{2n}{n-1}x^n \\ &= 1+\sum_{n=1}^{\infty}\frac{1}{n+1}\binom{2n}{n}x^n \\ &= \sum_{n=0}^{\infty}\frac{1}{n+1}\binom{2n}{n}x^n. \end{aligned}$$

Άρα, ο αριθμός των λέξεων Dyck με n μονάδες, δηλαδή μήκους $2n$, ισούται με

$$\frac{1}{n+1}\binom{2n}{n}.$$

Οι αριθμοί αυτοί ονομάζονται αριθμοί Catalan και συνήθως συμβολίζονται με c_n.

ii) Αν τεθεί $H(\lambda)=(y-1+\lambda)\lambda$ τότε η σχέση (2.10) γράφεται $F(x,y)=1+xH(F(x,y))$, οπότε, από το θεώρημα αντιστροφής του Lagrange, προκύπτει ότι

$$\begin{aligned} [x^n]F(x,y) &= \frac{1}{n}[\lambda^{n-1}](H(\lambda+1))^n \\ &= \frac{1}{n}[\lambda^{n-1}](y+\lambda)^n(\lambda+1)^n \\ &= \frac{1}{n}[\lambda^{n-1}]\sum_{j=0}^{n}\binom{n}{j}(y+\lambda)^n\lambda^j \\ &= \frac{1}{n}[\lambda^{n-1}]\sum_{j=0}^{n}\sum_{k=0}^{n}\binom{n}{j}\binom{n}{k}y^k\lambda^{j+n-k} \\ &= \frac{1}{n}[\lambda^{n-1}]\sum_{i=0}^{2n}\sum_{k=0}^{\left[\frac{n}{2}\right]}\binom{n}{i+k-n}\binom{n}{k}y^kx^i \\ &= \frac{1}{n}\binom{n}{k-1}\binom{n}{k}y^k. \end{aligned}$$

Επομένως,

$$F(x,y)=1+\sum_{n=1}^{\infty}\frac{1}{n}\binom{n}{k-1}\binom{n}{k}y^kx^n.$$

Άρα, για $n\geq 1$, ο αριθμός των λέξεων Dyck μήκους $2n$ και με ακριβώς k εμφανίσεις

του 10 ισούται με

$$\frac{1}{n}\binom{n}{k-1}\binom{n}{k}.$$

Οι αριθμοί αυτοί ονομάζονται αριθμοί Narayana.

□

Οι πρώτοι όροι της ακολουθίας $[x^n y^k]F(x,y)$ δίδονται στον επόμενο πίνακα.

$n \setminus k$	0	1	2	3	4	5	6	7	8	9	10	
0	1											0
1		1										1
2		1	1									2
3		1	3	1								5
4		1	6	6	1							14
5		1	10	20	10	1						42
6		1	15	50	50	15	1					132
7		1	21	105	175	105	21	1				429
8		1	28	196	490	490	196	28	1			1430
9		1	36	336	1176	1764	1176	336	36	1		4862
10		1	45	540	2520	5992	5292	2520	540	45	1	16796

Παράδειγμα. Το πλήθος των λέξεων Dyck μήκους 10 με ακριβώς 3 εμφανίσεις του 10 ισούται με 20. Πράγματι, οι λέξεις αυτές είναι οι ακόλουθες:

1010111000 1011011000 1011001100 1011100100 1011101000
1011100010 1100101100 1100110100 1100110010 1101001100
1101011000 1101100010 1101100100 1101101000 1110001010
1110010010 1110010100 1110100010 1110100100 1110101000

Πόρισμα 2.3.3.

i) Ο αριθμός των λέξεων Dyck μήκους $2n$ είναι περίπου ίσος με

$$\frac{4^n}{n^{3/2}\sqrt{\pi}}.$$

ii) Ο μέσος όρος του αριθμού των εμφανίσεων του 10 στις λέξεις Dyck μήκους $2n$ ισούται με

$$\frac{n+1}{2}.$$

Απόδειξη.

i) Η $C(x)$ έχει ένα σημείο ιδιομορφίας για $x = \frac{1}{4}$. Αν τεθεί

$$f(x) = (1-4x)^{1/2},\ g(x) = -\frac{1}{2x} \text{ και } h(x) = \frac{1}{2x},$$

τότε ισχύει ότι

i) $C(x) = f(x)g(x) + h(x)$.

ii) Η $C(x)$ δεν έχει άλλο σημείο ιδιομορφίας στο διάστημα $[-1/4, 1/4)$.[2]

iii) $L = \lim_{x\to\frac{1}{4}} g(x) = -2 \neq 0$.

iv) Η $h(x)$ δεν έχει ιδιομορφία στο $x = \frac{1}{4}$.

Άρα, από την Πρόταση 1.2.2 για $c = \frac{1}{2}$, $b = 0$ και $L = -2$ προκύπτει ότι

$$c_n \sim \frac{(-2)\cdot 4^n}{n^{\frac{1}{2}+1}\Gamma(-\frac{1}{2})} = \frac{(-2)\cdot 4^n}{n^{3/2}(-2)\sqrt{\pi}},$$

και επομένως,

$$c_n \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}.$$

Στον επόμενο πίνακα δίδονται για σύγκριση οι τιμές του αριθμού των λέξεων Dyck, οι εκτιμήσεις αυτών των αριθμών και ο αντίστοιχος λόγος του σφάλματος.

ημιμήκος	αριθμός λέξεων	εκτίμηση	λόγος σφάλματος
2	2	3.191538243	1.595769122
4	14	18.05406667	1.289576191
6	132	157.2382262	1.191198684
8	1430	1634.067581	1.142704602
10	16796	18707.89729	1.113830513
12	208012	227705.2745	1.094673742
14	2674440	2.89116507×10^6	1.081035683
16	35357670	3.78621220×10^7	1.070831931

ii) Από τη σχέση (2.10), παραγωγίζοντας ως προς y, παίρνουμε

$$\frac{\partial F(x,y)}{\partial y} = \frac{1}{2}\left(\frac{1+x(1-y)}{\sqrt{(1-x(1-y))^2-4x}} - 1\right)$$

[2] Το $x = 0$ δεν είναι σημείο ιδιομορφίας.

και θέτοντας $y = 1$ προκύπτει η γεννήτρια

$$\frac{1}{2}\left(\frac{1}{\sqrt{1-4x}} - 1\right).$$

Από την επέκταση του διωνυμικού θεωρήματος (1.1) ισχύει ότι

$$\begin{aligned}\frac{1}{2}\left(\frac{1}{\sqrt{1-4x}} - 1\right) &= \frac{1}{2}\left(\sum_{n=0}^{\infty}\binom{-\frac{1}{2}}{n}(-4x)^n - 1\right)\\ &= \frac{1}{2}\left(\sum_{n=0}^{\infty}\frac{(-1)^n}{4^n}\binom{2n}{n}(-4)^n x^n - 1\right)\\ &= \frac{1}{2}\left(\sum_{n=0}^{\infty}\binom{2n}{n}x^n - 1\right),\end{aligned}$$

οπότε για $n \geq 1$ ο συντελεστής του x^n στη γεννήτρια αυτή ισούται με $\frac{1}{2}\binom{2n}{n}$.

και επομένως, ο μέσος όρος του αριθμού των εμφανίσεων του 10 στις λέξεις Dyck μήκους $2n$ ισούται με

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

δηλαδή είναι περίπου ίσος με το ένα τέταρτο του μήκους τους.

□

Για περισσότερα στοιχεία σχετικά με τους αριθμούς Catalan βλέπε στο λήμμα A000108 στο [13] και στο βιβλίο του Stanley [14] όπου υπάρχουν 66 συνδυαστικές ερμηνείες για τους αριθμούς Catalan και πολλά άλλα στοιχεία, τα οποία ανανεώνονται τακτικά με την ειδική έκδοση Catalan Addendum [15] όπου σήμερα περιέχονται πάνω από 180 συνδυαστικές ερμηνείες. Επίσης, για περισσότερα στοιχεία σχετικά με τις λέξεις Dyck, βλέπε τη διδακτορική διατριβή του Τασούλα [16].

2.4 Λέξεις και πρότυπα

Ένα σημαντικό θέμα με το οποίο έχουν ασχοληθεί πολλοί ερευνητές τα τελευταία 30 χρόνια είναι η απαρίθμηση των λέξεων σύμφωνα με το μήκος τους και τον αριθμό των εμφανίσεων μιας δοσμένης λέξης, η οποία καλείται **πρότυπο**, μέσα σ' αυτά.

Έτσι, για παράδειγμα, αν θεωρήσουμε την δυαδική λέξη $\tau = 011001$ διατυπώνεται το ερώτημα: Πόσες δυαδικές λέξεις μήκους 100 περιέχουν την λέξη τ ακριβώς 18 φορές;

2.4.1 Απαρίθμηση λέξεων που αποφεύγουν συγκεκριμένο πρότυπο

Σε προηγούμενες παραγράφους, μελετήθηκε το πρόβλημα της απαρίθμησης των λέξεων που αποφεύγουν το πρότυπο w, όταν $w = 00$ και όταν $w = 010$ ή $w = 101$.

Στην παράγραφο αυτή θα μελετηθεί το πρόβλημα αυτό για οποιοδήποτε πρότυπο w.

Ένα τμήμα σ μιας λέξης w ονομάζεται **σύνορο** της w, αν είναι ταυτόχρονα πρόθεμα και επίθεμα της λέξης w. Για κάθε λέξη w, συμβολίζουμε με $\mathcal{B}_w$ το σύνολο όλων των συνόρων της. Για κάθε σύνορο σ μιας λέξης w, συμβολίζουμε με $\widetilde{\sigma}$ τη λέξη για την οποία $w = \sigma\widetilde{\sigma}$. Η $\widetilde{\sigma}$ ονομάζεται **συμπλήρωμα** του σ ως προς το w.

Μερικές χρήσιμες ιδιότητες σχετικά με τα σύνορα μιας λέξης δίδονται στο επόμενο λήμμα.

Λήμμα 2.4.1.

i) Αν σ_1, σ_2 είναι σύνορα μιας λέξης w με $|\sigma_1| > |\sigma_2|$ τότε το σ_2 είναι σύνορο του σ_1.

ii) Αν σ είναι σύνορο μια της λέξης w με $2|\sigma| > |w|$, τότε υπάρχει σ' με $2|\sigma'| < |w|$ και σ' σύνορο της w.

Απόδειξη.

i) Επειδή σ_1, σ_2 είναι σύνορα της w, ισχύει ότι

$$w = \sigma_1\beta = \beta'\sigma_1 \text{ και } w = \sigma_2\beta = \gamma'\sigma_2.$$

Επιπλέον, επειδή $|\sigma_1| > |\sigma_2|$, προκύπτει ότι

$$\sigma_1 = \sigma_2\delta \text{ και } \sigma_1 = \delta'\sigma_2.$$

Άρα, σ_2 είναι σύνορο του σ_1.

ii) Έστω $|w| = n$ και $|\sigma| = k$ με $2k > n$, τότε $\sigma = w_1w_2\cdots w_k = w_{n-k}w_{n-k+1}\cdots w_n$. Επειδή $n - k < k$ η λέξη $w_kw_{k+1}\cdots w_{n-k}$ είναι μη κενή και είναι πρόθεμα και επίθεμα του σ, άρα είναι σύνορο του σ και, κατά συνέπεια, σύνορο του w.

□

Πρόταση 2.4.2. *Η γεννήτρια συνάρτηση $A(x, y)$ των δυαδικών λέξεων που αποφεύγουν το πρότυπο w, όπου το x μετράει το μήκος και το y μετράει τον αριθμό των μονάδων, δίδεται από τον τύπο*

$$A(x, y) = \frac{1 + B(x, y)}{x^{|w|}y^{|w|_1} + (1 - (1 + y)x)(1 + B(x, y))} \tag{2.11}$$

όπου $B(x, y) = \sum_{\sigma\in\mathcal{B}_w} x^{|w|-|\sigma|}y^{|w|_1-|\sigma|_1}$.

Απόδειξη. Έστω $\mathcal{A}$ το σύνολο όλων των δυαδικών λέξεων που αποφεύγουν την λέξη w. Τότε

$$A(x,y) = \sum_{\alpha \in \mathcal{A}} x^{|\alpha|} y^{|\alpha|_1}.$$

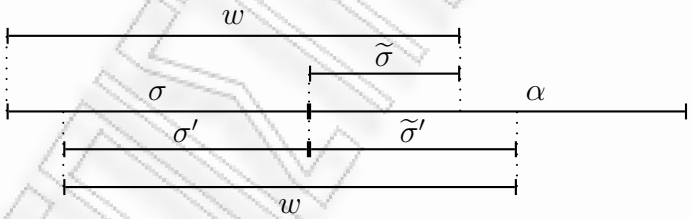
Κάθε λέξη $\alpha \in \mathcal{A}$ είτε αρχίζει με κάποιο συμπλήρωμα ενός συνόρου $\sigma \in \mathcal{B}_w$, είτε όχι.

Οι λέξεις που αρχίζουν με κάποιο συμπλήρωμα ενός συνόρου $\sigma \in \mathcal{B}_w$, διαμερίζονται ως προς το μεγαλύτερο σε μήκος από αυτά, και έστω $\mathcal{A}_{\widetilde{\sigma}}$ η αντίστοιχη κλάση της διαμέρισης. Έστω $\mathcal{A}'$ το σύνολο των λέξεων που δεν αρχίζουν με κάποιο συμπλήρωμα ενός συνόρου $\sigma \in \mathcal{B}_w$.

Έστω $\mathcal{W}$ το σύνολο των λέξεων που αρχίζουν με w και δεν περιέχουν άλλη εμφάνιση του προτύπου w, και $W(x,y)$ η γεννήτρια συνάρτηση του συνόλου $\mathcal{W}$ ως προς τις παραμέτρους μήκος και αριθμός μονάδων.

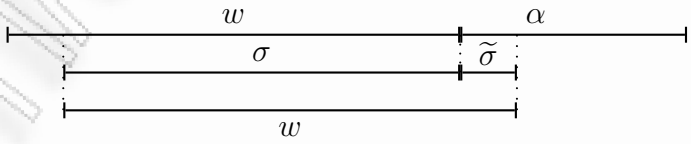
Για κάθε λέξη $\alpha \in \mathcal{A}_{\widetilde{\sigma}}$ ισχύει ότι η λέξη $\sigma\alpha \in \mathcal{W}$. Πράγματι, επειδή η λέξη α αρχίζει με $\tilde{\sigma}$, η λέξη $\sigma\alpha$ αρχίζει με w. Έστω ότι η $\sigma\alpha$ περιέχει και άλλη εμφάνιση του w. Η εμφάνιση αυτή τελειώνει μετά το $\widetilde{\sigma}$, διότι για την υπολέξη σp, όπου p γνήσιο πρόθεμα του $\tilde{\sigma}$, ισχύει ότι $|\sigma p| < |w|$.

Αν σ' είναι το κοινό τμήμα της δεύτερης εμφάνισης του w και του σ, τότε το σ' είναι σύνορο του w, αφού είναι επίθεμα του σ (που είναι επίθεμα του w) και πρόθεμα του w.



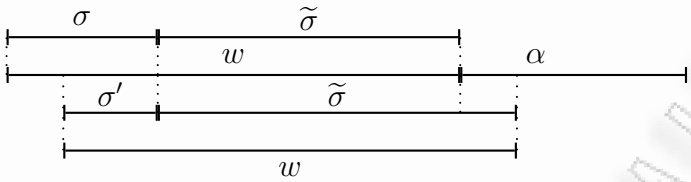
Επομένως $|\widetilde{\sigma}'| > |\tilde{\sigma}|$ και το α ξεκινάει με $\widetilde{\sigma}'$ το οποίο είναι άτοπο.

Επιπρόσθετα για κάθε $\alpha \in \mathcal{A}'$ η λέξη $w\alpha \in \mathcal{W}$. Πράγματι, η $w\alpha$ αρχίζει με w. Έστω ότι περιέχει και άλλη εμφάνιση του w, τότε δεδομένου ότι η δεύτερη εμφάνιση δεν ανήκει εξ ολοκλήρου στο α, θα υπάρχει ένα κοινό μη κενό τμήμα σ των δύο εμφανίσεων το οποίο είναι σύνορο του w και επομένως το α ξεκινάει με $\widetilde{\sigma}$ το οποίο είναι άτοπο.



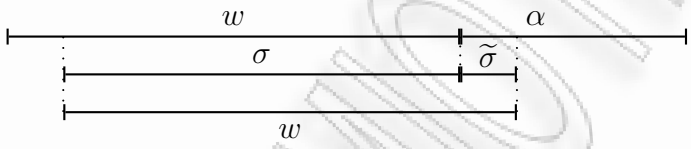
Αντίστροφα, έστω $w\alpha \in \mathcal{W}$. Για κάθε σύνορο $\sigma \in \mathcal{B}_w$ ισχύει ότι $w\alpha = \sigma\widetilde{\sigma}\alpha$ και η λέξη $\widetilde{\sigma}\alpha$ ανήκει στο $\mathcal{A}_{\widetilde{\sigma}}$.

Πράγματι, προφανώς η $\widetilde{\sigma}\alpha \in \mathcal{A}$ διότι αποφεύγει το w. Επίσης, έστω ότι υπάρχει $\sigma' \in \mathcal{B}_w$ τέτοιο ώστε η $\widetilde{\sigma}\alpha$ αρχίζει με $\widetilde{\sigma}'$, όπου $|\widetilde{\sigma}| < |\widetilde{\sigma}'|$.



Επειδή $|\widetilde{\sigma}| < |\widetilde{\sigma}'|$, έπεται ότι $|\sigma'| < |\sigma|$, οπότε από το προηγούμενο λήμμα ισχύει ότι σ' είναι σύνορο του σ, άρα ακριβώς πριν την αρχή του $\widetilde{\sigma}'$ (και στο τέλος του σ) υπάρχει το σύνορο σ', οπότε η λέξη $w\alpha$ περιέχει και δεύτερη εμφάνιση του w, το οποίο είναι άτοπο. Άρα, $\widetilde{\sigma}\alpha \in \mathcal{A}_\sigma$.

Επιπλέον, αν $w\alpha \in \mathcal{W}$, τότε $\alpha \in \mathcal{A}'$. Πράγματι, προφανώς $\alpha \in \mathcal{A}$, διότι αποφεύγει το w. Επίσης, αν υπάρχει $\sigma \in \mathcal{B}_w$ τέτοιο ώστε η λέξη α αρχίζει με $\widetilde{\sigma}$, τότε η $w\alpha$ περιέχει και δεύτερη εμφάνιση του w, το οποίο είναι άτοπο.



Κατόπιν τούτων, προκύπτει ότι

$$\mathcal{W} = \{w\alpha : \alpha \in \mathcal{A}'\} = \{\sigma\alpha : \alpha \in \mathcal{A}_{\widetilde{\sigma}}\}, \text{ για κάθε } \sigma \in \mathcal{B}_w.$$

Επομένως,

$$\begin{aligned}
A(x,y) &= \sum_{\alpha\in\mathcal{A}} x^{|\alpha|}y^{|\alpha|_1} \\
&= \sum_{\alpha\in\mathcal{A}'} x^{|\alpha|}y^{|\alpha|_1} + \sum_{\sigma\in\mathcal{B}_w}\sum_{\alpha\in\mathcal{A}_{\widetilde{\sigma}}} x^{|\alpha|}y^{|\alpha|_1} \\
&= x^{-|w|}y^{-|w|_1}\sum_{\alpha\in\mathcal{A}'} x^{|w\alpha|}y^{|w\alpha|_1} + \sum_{\sigma\in\mathcal{B}_w} x^{-|\sigma|}y^{-|\sigma|_1}\sum_{\alpha\in\mathcal{A}_{\widetilde{\sigma}}} x^{|\sigma\alpha|}y^{|\sigma\alpha|_1} \\
&= x^{-|w|}y^{-|w|_1}\left(\sum_{\substack{w\alpha\in\mathcal{W}\\ \alpha\in\mathcal{A}'}} x^{|w\alpha|}y^{|w\alpha|_1} + \sum_{\sigma\in\mathcal{B}_w} x^{|w|-|\sigma|}y^{|w|_1-|\sigma|_1}\sum_{\substack{\sigma\alpha\in\mathcal{W}\\ \alpha\in\mathcal{A}_{\widetilde{\sigma}}}} x^{|\sigma\alpha|}y^{|\sigma\alpha|_1}\right) \\
&= x^{-|w|}y^{-|w|_1}\left(\sum_{\beta\in\mathcal{W}} x^{|\beta|}y^{|\beta|_1} + \sum_{\sigma\in\mathcal{B}_w} x^{|w|-|\sigma|}y^{|w|_1-|\sigma|_1}\sum_{\beta\in\mathcal{W}} x^{|\beta|}y^{|\beta|_1}\right).
\end{aligned}$$

Άρα,

$$A(x,y) = x^{-|w|}y^{-|w|_1}\left(1 + \sum_{\sigma\in\mathcal{B}_w} x^{|w|-|\sigma|}y^{|w|_1-|\sigma|_1}\right)W(x,y). \tag{2.12}$$

Έστω $\alpha \in \mathcal{A} \cup \mathcal{W}$. Κάθε μη κενή λέξη $\alpha \in \mathcal{A} \cup \mathcal{W}$ γράφεται κατά μοναδικό τρόπο υπό τη μορφή

$$\alpha = 0\beta \text{ ή } \alpha = 1\beta$$

όπου $\beta \in \mathcal{A}$.

Πράγματι, αν $\alpha \in \mathcal{A}$, τότε διακρίνουμε δύο περιπτώσεις: Αν $\alpha = 0\beta$, τότε πρέπει και η β να αποφεύγει το w, οπότε $\beta \in \mathcal{A}$. Αν $\alpha = 1\beta$, τότε πρέπει και η β να αποφεύγει το w, οπότε $\beta \in \mathcal{A}$. Αν $\alpha \in \mathcal{W}$, τότε είτε $\alpha = 0\beta$ είτε $\alpha = 1\beta$. Και στις δύο περιπτώσεις η β αποφεύγει την λέξη w, γιατί αλλιώς η λέξη α έχει τουλάχιστον δύο εμφανίσεις, το οποίο είναι άτοπο, άρα $\beta \in \mathcal{A}$.

Αντίστροφα, έστω $\beta \in \mathcal{A}$. Ισχύει ότι 0β, $1\beta \in \mathcal{A} \cup \mathcal{W}$, διότι η προσθήκη ενός 0 ή 1 στην αρχή της λέξης β δημιουργεί το πολύ μια νέα εμφάνιση του w.

Επομένως,

$$\sum_{\alpha \in \mathcal{A} \cup \mathcal{B}} x^{|\alpha|} y^{|\alpha|_1} = \sum_{a=\varepsilon} x^{|\varepsilon|} y^{|\varepsilon|_1} + \sum_{\substack{\alpha=0\beta \\ \beta \in \mathcal{A}}} x^{|0\beta|} y^{|0\beta|_1} + \sum_{\substack{\alpha=1\beta \\ \beta \in \mathcal{A}}} x^{|1\beta|} y^{|1\beta|_1}$$

ή ισοδύναμα,

$$\sum_{\alpha \in \mathcal{A}} x^{|\alpha|} y^{|\alpha|_1} + \sum_{\alpha \in \mathcal{W}} x^{|w\beta|} y^{|w\beta|_1} = 1 + x \sum_{\beta \in \mathcal{A}} x^{|\beta|} y^{|\beta|_1} + xy \sum_{\beta \in \mathcal{A}} x^{|\beta|} y^{|\beta|_1}.$$

Άρα,

$$A(x,y) + W(x,y) = 1 + x(1+y)A(x,y). \tag{2.13}$$

Από τις σχέσεις (2.12) και (2.13) προκύπτει η αποδεικτέα σχέση. □

Πόρισμα 2.4.3. *Η γεννήτρια $W(x,y)$ συνάρτηση των δυαδικών λέξεων που αρχίζουν με w και δεν περιέχουν άλλη εμφάνιση του w, όπου το x μετράει το μήκος και το y τον αριθμό των μονάδων, δίδεται από τον τύπο*

$$W(x,y) = \frac{x^{|w|} y^{|w|_1}}{x^{|w|} y^{|w|_1} + (1 - (1+y)x)(1 + B(x,y))}. \tag{2.14}$$

Εφαρμογές

1. Αν $w = 0^k$, όπου $k \geq 1$, τότε $\mathcal{B}_w = \{0^i, 1 \leq i \leq k-1\}$, οπότε

$$B(x,y) = \sum_{i=1}^{k-1} x^{k-i} y^0 = \frac{x(1 - x^{k-1})}{1-x}.$$

Άρα,

$$\begin{aligned} A(x,y) &= \frac{1+\frac{x(1-x^{k-1})}{1-x}}{x^k y^0 + (1-(1+y)x)\left(1+\frac{x(1-x^{k-1})}{1-x}\right)} \\ &= \frac{1-x^k}{x^k(1-x)+(1-(1+y)x)(1-x^k)} \\ &= \frac{1-x^k}{1-(1+y)x+yx^{k+1}}. \end{aligned} \tag{2.15}$$

Αν $w = 00$, τότε από την (2.15) προκύπτει ότι

$$A(x,y) = \frac{1-x^2}{1-(1+y)x+yx^3} = \frac{(1+x)(1-x)}{(1-x)(1-yx(1+x))} = \frac{1+x}{1+yx+yx^2},$$

η οποία ταυτίζεται με την γεννήτρια συνάρτηση των αριθμών Fibonacci ως προς το μήκος και τον αριθμό των μονάδων (βλ. σχέση (2.1)).

2. Η γεννήτρια συνάρτηση $L(x,z)$ των δυαδικών λέξεων, όπου το x μετράει το μήκος και το z μετράει την θέση της πρώτης εμφάνισης του w, δίδεται από τον τύπο

$$L(x,z) = \frac{z^{-|w|+1}}{1-2x} W(xz,1) + A(x,1). \tag{2.16}$$

Πράγματι, κάθε δυαδική λέξη α είτε περιέχει εμφάνιση του w, είτε όχι.

Έστω ότι η α περιέχει μια τουλάχιστον εμφάνιση του w, τότε η α γράφεται κατά μοναδικό τρόπο υπό τη μορφή

$$\alpha = \beta\gamma,$$

όπου β τελειώνει με w και δεν περιέχει άλλη εμφάνιση του w και γ είναι μια δυαδική λέξη.

Αν η λέξη α δεν περιέχει εμφάνιση του w, τότε $\alpha \in \mathcal{A}$.

Έστω $\mathcal{W}_r$ το σύνολο των δυαδικών λέξεων που τελειώνουν με w και δεν περιέχουν άλλη εμφάνιση του w. Προφανώς, κάθε λέξη $\alpha \in \mathcal{W}_r$ αντιστοιχεί μονοσήμαντα σε μια λέξη $\alpha' \in W$, όπου α' είναι η κατοπτρική λέξη της α.

Άρα,

$$\begin{aligned}
L(x,z) &= \sum_{\alpha\in\{0,1\}^*} x^{|a|} z^{\text{θέση πρώτης εμφάνισης της } w} \\
&= \sum_{\substack{\alpha=\beta\gamma \\ \beta\in\mathcal{W}_r \\ \gamma\in\{0,1\}^*}} x^{|\beta|+|\gamma|} z^{|\beta|-|w|+1} + \sum_{\alpha\in\mathcal{A}} x^{|\alpha|} z^0 \\
&= z^{-|w|+1} \sum_{\beta\in\mathcal{W}_r} x^{|\beta|} z^{|\beta|} \sum_{\gamma\in\{0,1\}^*} x^{|\gamma|} + A(x,1) \\
&= \frac{z^{-|w|+1}}{1-2x} \sum_{\beta\in\mathcal{W}} (xz)^{|\beta|} + A(x,1) \\
&= \frac{z^{-|w|+1}}{1-2x} W(xz,1) + A(x,1).
\end{aligned}$$

3. Ο αριθμός των μη αρνητικών ακέραιων λύσεων της εξίσωσης

$$x_1 + x_2 + \cdots + x_k = n$$

για τις οποίες δεν υπάρχουν $x_i, x_j \neq 0$ με $|i-j| = 1$, δηλαδή δεν υπάρχουν δύο διαδοχικοί μη μηδενικοί όροι, ισούται με

$$\sum_{j=0}^{n} (-1)^j \binom{n-1}{j} \binom{n+k-1-2j}{k-1-j}.$$

Πράγματι, κάθε λύση της εξίσωσης αυτής κωδικοποιείται από την δυαδική λέξη

$$0^{x_1} 1 0^{x_2} 1 \cdots 1 0^{x_k},$$

η οποία έχει μήκος $n+k-1$, περιέχει $k-1$ μονάδες και αποφεύγει το πρότυπο 010.

Αν $w = 010$, τότε $\mathcal{B}_w = \{0\}$, οπότε $B(x,y) = x^{3-1}y^{1-0} = x^2 y$. Άρα,

$$A(x,y) = \frac{1+x^2y}{x^3y^1 + (1-(1+y)x)(1+x^2y)} = \frac{1+x^2y}{1-x-xy+x^2y-x^3y^2},$$

οπότε

$$A(x,y) = 1 + x^2y + (x + xy - x^2y + x^3y^2)A(x,y).$$

Αν τεθεί $H(\lambda) = xy + (1 + y - xy + x^2y^2)\lambda$, τότε η παραπάνω σχέση γράφεται ως $A(x,y) = 1 + xH(A(x,y))$, οπότε από το θεώρημα αντιστροφής του Lagrange

προκύπτει ότι

$$
\begin{aligned}
[x^n]A(x,y) &= \frac{1}{n}[\lambda^{n-1}]((xy+(1+y-xy+x^2y^2)(1+\lambda))^n \\
&= \frac{1}{n}[\lambda^{n-1}]\sum_{j=0}^{n}\binom{n}{j}(xy)^{n-j}(1+y-xy+x^2y^2)^j(1+\lambda)^j \\
&= \frac{1}{n}[\lambda^{n-1}]\sum_{j=0}^{n}\sum_{i=0}^{j}\binom{n}{j}\binom{j}{i}(xy)^{n-j}(1+y-xy+x^2y^2)^j\lambda^i \\
&= \frac{1}{n}[\lambda^{n-1}]\sum_{i=0}^{n}\sum_{j=i}^{n}\binom{n}{j}\binom{j}{i}(xy)^{n-j}(1+y-xy+x^2y^2)^j\lambda^i \\
&= \frac{1}{n}\left(\binom{n}{n-1}\binom{n-1}{n-1}xy(1+y-xy+x^2y^2)^{n-1}+\binom{n}{n}\binom{n}{n-1}(1+y-xy+x^2y^2)^n\right) \\
&= (1+y-xy+x^2y^2)^{n-1}(1+y+x^2y^2).
\end{aligned}
$$

Επομένως,

$$
\begin{aligned}
A(x,y) &= 1+\sum_{n=1}^{\infty}(1+y-xy+x^2y^2)^{n-1}(1+y+x^2y^2)x^n \\
&= 1+\sum_{n=1}^{\infty}\sum_{j=0}^{n-1}\binom{n-1}{j}(-xy)^j(1+y+x^2y^2)^{n-1-j+1}x^n \\
&= 1+\sum_{n=1}^{\infty}\sum_{j=0}^{n-1}\sum_{i=0}^{n-j}\binom{n-1}{j}\binom{n-j}{i}(-xy)^jy^i(1+x^2y)^ix^n \\
&= 1+\sum_{n=1}^{\infty}\sum_{j=0}^{n-1}\sum_{i=0}^{n-j}\sum_{k=0}^{i}\binom{n-1}{j}\binom{n-j}{i}\binom{i}{k}(-1)^j(xy)^jy^ix^{2k}y^kx^n \\
&= 1+\sum_{n=1}^{\infty}\sum_{j=0}^{n-1}\sum_{i=0}^{n-j}\sum_{k=0}^{i}\binom{n-1}{j}\binom{n-j}{i}\binom{i}{k}(-1)^jx^{n+2k+j}y^{k+j+i} \\
&= 1+\sum_{\alpha=1}^{\infty}\sum_{\beta=0}^{\alpha}\sum_{i=2\beta+1-\alpha}^{\beta}\sum_{k=0}^{i}\binom{\alpha-\beta-k+i-1}{\beta-k-i}\binom{\alpha-2\beta+2i}{i}\binom{i}{k}(-1)^{\beta-k-i}x^\alpha y^\beta \\
&= 1+\sum_{\alpha=1}^{\infty}\sum_{\beta=0}^{\alpha}\sum_{i=2\beta+1-\alpha}^{\beta}\binom{\alpha-2\beta+2i}{i}(-1)^{\beta-i}\sum_{k=0}^{i}(-1)^kx^\alpha y^\beta\binom{\alpha-\beta+i-1-k}{\alpha-2\beta+2i-1}\binom{i}{k}.
\end{aligned}
$$

Χρησιμοποιώντας την επόμενη ταυτότητα (βλ. σχέση (3.49) στο [5])

$$\sum_{k=0}^{n}(-1)^k\binom{n}{k}\binom{x-k}{r}=\binom{x-n}{r-n}$$

προκύπτει ότι

$$\begin{aligned}A(x,y)&=1+\sum_{\alpha=1}^{\infty}\sum_{\beta=0}^{\alpha}\sum_{i=2\beta+1-\alpha}^{\beta}(-1)^{\beta-i}\binom{\alpha-2\beta+2i}{i}\binom{\alpha-\beta-1}{\alpha-2\beta+i-1}x^\alpha y^\beta\\&=1+\sum_{\alpha=1}^{\infty}\sum_{\beta=0}^{\alpha}\sum_{i=2\beta+1-\alpha}^{\beta}(-1)^{\beta-i}\binom{\alpha-\beta-1}{\beta-i}\binom{\alpha-2(\beta-i)}{i}x^\alpha y^\beta\\&=1+\sum_{\alpha=1}^{\infty}\sum_{\beta=0}^{\alpha}\sum_{\xi=0}^{\alpha-\beta}(-1)^{\xi}\binom{\alpha-\beta-1}{\xi}\binom{\alpha-2\xi}{\beta-\xi}x^\alpha y^\beta.\end{aligned}$$

Άρα, ο ζητούμενος αριθμός ισούται με

$$\sum_{j=0}^{n}(-1)^j\binom{n-1}{j}\binom{n+k-1-2j}{k-1-j}.$$

Για περισσότερες πληροφορίες σχετικά με τις λέξεις που αποφεύγουν το πρότυπο 010, βλ. λήμμα A180562 στη [13].

Παρατηρήσεις.

1. Από την Πρόταση 2.4.2 προκύπτει ότι ο αριθμός των δυαδικών λέξεων που αποφεύγουν ένα πρότυπο w δεν εξαρτάται από την ίδια την λέξη w αλλά από τα μήκη των συνόρων της, επομένως για δύο διαφορετικά πρότυπα w_1, w_2 τα οποία έχουν ίδια μήκη συνόρων οι αριθμοί των δυαδικών λέξεων που αποφεύγουν το w_1 και το w_2 αντίστοιχα είναι ίσοι.

2. Η γεννήτρια συνάρτηση $A(x,1)$ έχει μελετηθεί για πρώτη φορά από τους L. Guibas και A. Odlyzko κατά τη μελέτη των μη μεταβατικών παιγνίων [6], [7].

2.4.2 Απεριοδικές λέξεις

Μια ειδική κατηγορία προτύπων είναι τα πρότυπα που δεν έχουν σύνορα, τα οποία ονομάζονται **απεριοδικά**.

Ο αριθμός των δυαδικών λέξεων που αποφεύγουν ένα απεριοδικό πρότυπο μήκους r δίδεται από τον τύπο

$$\sum_{j=0}^{\left[\frac{n}{r}\right]} \binom{n-(r-1)j}{j} (-1)^j 2^{n-rj}.$$

Πράγματι, έστω ότι w δεν έχει σύνορα με $|w| = r$. Τότε ισχύει ότι $\mathcal{B}_w = \varnothing$, οπότε

$$B(x,1) = 0.$$

Άρα, από την Πρόταση 2.4.2

$$\begin{aligned} F(x,1) &= \frac{1}{1-2x+x^r} \\ &= \sum_{n=0}^{\infty} x^n (2-x^{r-1})^n \\ &= \sum_{n=0}^{\infty} \sum_{j=0}^{n} \binom{n}{j} (-1)^j 2^{n-j} x^{n+(r-1)j} \\ &= \sum_{n=0}^{\infty} \sum_{j=0}^{\left[\frac{n}{r}\right]} \binom{n-(r-1)j}{j} (-1)^j 2^{n-rj} x^n. \end{aligned}$$

Ένα ενδιαφέρον ερώτημα είναι πόσα απεριοδικά πρότυπα υπάρχουν με δεδομένο μήκος.

Παράδειγμα. Οι απεριοδικές δυαδικές λέξεις μήκους το πολύ 4 είναι οι εξής: ε, 0, 1, 01, 10, 001, 011, 100, 110, 0001, 0011, 0111, 1000, 1100, 1110.

Έστω $\mathcal{U}$ (αντ. $\mathcal{U}_n$) το σύνολο των απεριοδικών λέξεων (αντ. μήκους n).

Πρόταση 2.4.4. *Ο αριθμός u_n των απεριοδικών δυαδικών λέξεων μήκους n ικανοποιεί τις σχέσεις*

$$u_{2n+1} = 2u_{2n}, n \geq 1,$$

$$u_{2n} = 2u_{2n-1} - u_n, n \geq 2$$

όπου $u_0 = 1$ και $u_1 = 2$.

Απόδειξη. Έστω $\mathcal{U}_{2n+1}^{(0)}, \mathcal{U}_{2n+1}^{(1)}$ τα σύνολα των απεριοδικών δυαδικών λέξεων μήκους $2n+1$, $\alpha = \alpha_1 \cdots \alpha_n \alpha_{n+1} \alpha_{n+2} \cdots \alpha_{2n+1}$ για τις οποίες $\alpha_{n+1} = 0$ και $\alpha_{n+1} = 1$ αντίστοιχα.

Κάθε λέξη $\alpha \in \mathcal{U}_{2n+1}^{(0)}$ απεικονίζεται μονοσήμαντα σε μια λέξη $\alpha' \in \mathcal{U}_{2n+1}^{(1)}$ εναλλάσσοντας το ψηφίο α_{n+1}.

Αρκεί να αποδειχθεί ότι η εναλλαγή του ψηφίου α_{n+1} δεν δημιουργεί σύνορα. Πράγματι, έστω ότι η λέξη $\alpha_1\alpha_2\cdots\alpha_n\alpha'_{n+1}\alpha_{n+2}\cdots\alpha_{2n+1}$ περιέχει σύνορο, όπου α'_{n+1} είναι το συμπλήρωμα του α_{n+1}. Το σύνορο αυτό δεν είναι μήκους $k \leq n$, γιατί και η λέξη $\alpha_1\alpha_2\cdots\alpha_n\alpha_{n+1}\alpha_{n+2}\cdots\alpha_{2n+1}$ είναι απεριοδική. Αν $k \geq n+1$, τότε από το Λήμμα 2.4.1 προκύπτει ότι υπάρχει και ένα σύνορο με μήκος μικρότερο του n, το οποίο είναι άτοπο.

Επομένως,

$$|\mathcal{U}_{2n+1}^{(0)}| = |\mathcal{U}_{2n+1}^{(1)}| = \frac{|\mathcal{U}_{2n+1}|}{2}.$$

Επιπλέον, αν $\alpha = \alpha_1\cdots\alpha_n\alpha_{n+1}\cdots\alpha_{2n} \in \mathcal{U}_{2n}$ ισχύει ότι $\alpha_1\cdots\alpha_n 0\alpha_{n+1}\cdots\alpha_{2n} \in \mathcal{U}_{2n+1}^{(0)}$ και αντιστρόφως.

Αρκεί να αποδειχθεί ότι η λέξη $\alpha' = \alpha_1\cdots\alpha_n 0\alpha_{n+1}\cdots\alpha_{2n}$ δεν περιέχει σύνορα. Πράγματι, έστω ότι η α' περιέχει κάποιο σύνορο μήκους k, τότε $k \geq n+1$ (αφού η λέξη $\alpha_1\cdots\alpha_n 0\alpha_{n+1}\cdots\alpha_{2n}$ δεν περιέχει σύνορα) και επομένως, από το Λήμμα 2.4.1, προκύπτει ότι υπάρχει και ένα σύνορο της α' με μήκος μικρότερο του n, το οποίο είναι άτοπο. Αντίστροφα, αν $\alpha_1\cdots\alpha_n 0\alpha_{n+1}\cdots\alpha_{2n}$ δεν περιέχει σύνορα, τότε και $\alpha_1\cdots\alpha_n\alpha_{n+1}\cdots\alpha_{2n}$ δεν περιέχει σύνορα. Πράγματι, με το ίδιο επιχείρημα, αν η λέξη $\alpha_1\cdots\alpha_n\alpha_{n+1}\cdots\alpha_{2n}$ περιέχει σύνορα, θα περιέχει και ένα σύνορο μήκους $k \leq n$ το οποίο είναι σύνορο της $\alpha_1\cdots\alpha_n 0\alpha_{n+1}\cdots\alpha_{2n}$, άτοπο.

Επομένως,

$$|\mathcal{U}_{2n}| = |\mathcal{U}_{2n+1}^{(0)}|.$$

Άρα, $|\mathcal{U}_{2n}| = \frac{|\mathcal{U}_{2n+1}|}{2}$, δηλαδή

$$u_{2n+1} = 2u_{2n}.$$

Έστω $\mathcal{P}_{2n}$ το σύνολο των λέξεων $\alpha = \beta\beta$ όπου $\beta \in \mathcal{U}_n$. Προφανώς, κάθε λέξη $\alpha \in \mathcal{U}_n$ περιέχει ακριβώς ένα σύνορο μήκους n (την λέξη β) και αντιστοιχεί μονοσήμαντα στη λέξη $\beta \in \mathcal{U}_n$. Επομένως,

$$|\mathcal{P}_{2n}| = |\mathcal{U}_n|.$$

Έστω $\mathcal{P}_{2n}^{(0)}$, $\mathcal{P}_{2n}^{(1)}$ (αντ. $\mathcal{U}_{2n}^{(0)}, \mathcal{U}_{2n}^{(1)}$) τα σύνολα των λέξεων του $\mathcal{P}_{2n}$ (αντ. των απεριοδικών λέξεων) $\alpha = \alpha_1\alpha_2\cdots\alpha_n\alpha_{n+1}\cdots\alpha_{2n}$ για τις οποίες ισχύει ότι $\alpha_{n+1} = 0$ και $\alpha_{n+1} = 1$ αντίστοιχα.

Κάθε λέξη $\alpha \in \mathcal{U}_{2n}^{(0)} \cup \mathcal{P}_{2n}^{(0)}$ απεικονίζεται μονοσήμαντα σε μια λέξη $\alpha' \in \mathcal{U}_{2n}^{(1)} \cup \mathcal{P}_{2n}^{(1)}$ εναλλάσσοντας όλα τα ψηφία της.

Επομένως,

$$|\mathcal{U}_{2n}^{(0)} \cup \mathcal{P}_{2n}^{(0)}| = |\mathcal{U}_{2n}^{(1)} \cup \mathcal{P}_{2n}^{(1)}| = \frac{|\mathcal{U}_{2n} \cup \mathcal{P}_{2n}|}{2} = \frac{|\mathcal{U}_{2n}| + |\mathcal{P}_{2n}|}{2}.$$

Επίσης, αν $\alpha = \alpha_1 \cdots \alpha_n \alpha_{n+1} \cdots \alpha_{2n-1} \in \mathcal{U}_{2n-1}$, ισχύει ότι $\alpha' = \alpha_1 \cdots \alpha_n 0 \alpha_{n+1} \cdots \alpha_{2n-1} \in \mathcal{U}_{2n}^{(0)} \cup \mathcal{P}_n$ και αντιστρόφως.

Αρκεί να αποδειχθεί ότι η λέξη α' είτε δεν έχει σύνορα είτε έχει ακριβώς ένα σύνορο, τη λέξη $\alpha_1 \alpha_2 \cdots \alpha_n$ μήκους n.

Πράγματι, έστω ότι η α' περιέχει κάποιο σύνορο μήκους k. Όπως και προηγουμένως, αν $k < n$ ή $k > n$, τότε προκύπτει ότι η λέξη α περιέχει και ένα σύνορο με μήκος μικρότερο του n, το οποίο είναι άτοπο. Αν $k = n$, τότε η α' γράφεται στη μορφή $\alpha = \beta\beta$, όπου β δεν περιέχει σύνορα. Τελικά, είτε $\alpha' \mathcal{U}_{2n}^{(0)}$, είτε $\alpha' \in \mathcal{P}_{2n}$.

Αντίστροφα, αν $\alpha' \in \mathcal{U}_{2n}^{(0)} \cup \mathcal{P}_{2n}^{(0)}$, τότε η $\alpha = \alpha_1 \cdots a_n \alpha_{n+1} \cdots \alpha_{2n-1}$ δεν περιέχει σύνορα.

Πράγματι, έστω ότι η α περιέχει σύνορα. Όπως και προηγουμένως, θα περιέχει και ένα σύνορο μήκους $k \leq n-1$ το οποίο είναι σύνορο της α', το οποίο είναι άτοπο, αφού η α' ή δεν έχει σύνορα, ή έχει ακριβώς ένα σύνορο μήκους n.

Επομένως,

$$|\mathcal{U}_{2n-1}| = |\mathcal{U}_{2n}^{(0)} \cup \mathcal{P}_{2n}^{(0)}|.$$

Άρα, $|\mathcal{U}_{2n-1}| = \frac{|\mathcal{U}|_{2n} + |\mathcal{P}|_{2n}}{2} = \frac{|\mathcal{U}|_{2n} + |\mathcal{U}|_n}{2}$, δηλαδή

$$u_{2n} = 2u_{2n-1} - u_n.$$

□

Μερικές από τις αρχικές τιμές της ακολουθίας των αριθμών u_n δίδονται στον παρακάτω πίνακα:

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
u_n	1	2	2	4	6	12	20	40	74	148	284	568	1116	2232	4424	8848

Για περισσότερες πληροφορίες σχετικά με την ακολουθία u_n και τις απεριοδικές λέξεις, βλέπε την εργασία των Harju και Nowotka [9], καθώς και το λήμμα A003000 στη [12].

Στην επόμενη πρόταση, υπολογίζεται ο αριθμός των δυαδικών λέξεων που περιέχουν συγκεκριμένες εμφανίσεις ενός απεριοδικού προτύπου.

Πρόταση 2.4.5. *Ο αριθμός των δυαδικών λέξεων μήκους n με k εμφανίσεις ενός απεριοδικού προτύπου w ισούται με*

$$\sum_{j=k}^{[n/r]} \binom{l-(r-1)j}{j} \binom{j}{k} 2^{n-rj} (-1)^{j-k},$$

όπου $r = |w|$.

Απόδειξη. Έστω $w = bw'$, $b \in \{0,1\}$ ένα απεριοδικό πρότυπο μήκους r και $F(x,y)$ η γεννήτρια συνάρτηση του συνόλου των δυαδικών λέξεων, όπου το x μετράει το μήκος και το y τον αριθμό των εμφανίσεων του w, δηλαδή

$$F(x,y) = \sum_{\alpha \in \{0,1\}^*} x^{|\alpha|} y^{|\alpha|_w},$$

όπου $|\alpha|$ το μήκος της λέξης α και $|\alpha|_w$ ο αριθμός των εμφανίσεων του w στο α.

Έστω $\mathcal{S}$ το σύνολο των λέξεων που αρχίζουν με w' και $S(x,y)$ η γεννήτρια συνάρτηση του συνόλου αυτού, όπου το x μετράει το μήκος και το y τον αριθμό των εμφανίσεων του w.

Για τις παραμέτρους $|\ |$ και $|\ |_w$ ισχύει ότι

$$|\varepsilon| = 0, \text{ και } |0\beta| = |1\beta| = 1 + |\beta|$$

και

$$|\varepsilon|_w = 0,\ |0\beta|_w = |\beta|_w + [b=0][\beta \in \mathcal{S}] \text{ και } |1\beta|_w = |\beta|_w + [b=1][\beta \in \mathcal{S}].$$

Επομένως,

$$\begin{aligned}
F(x,y) &= \sum_{\alpha \in \{0,1\}^*} x^{|\alpha|} y^{|\alpha|_w} \\
&= \sum_{\alpha=\varepsilon} x^{|\alpha|} y^{|\alpha|_w} + \sum_{\alpha=0\beta} x^{|0\beta|} y^{|0\beta|_w} + \sum_{\alpha=1\beta} x^{|1\beta|} y^{|1\beta|_w} \\
&= 1 + \sum_{\beta \in \{0,1\}^*} x^{|\beta|+1} y^{|\beta|_w + [b=0][\beta \in \mathcal{S}]} + \sum_{\beta \in \{0,1\}^*} x^{|\beta|+1} y^{|\beta|_w + [b=1][\beta \in \mathcal{S}]} \\
&= 1 + x \sum_{\beta \in \{0,1\}^*} x^{|\beta|} y^{|\beta|_w} \left(y^{[b=0][\beta \in \mathcal{S}]} + y^{[b=1][\beta \in \mathcal{S}]} \right) \\
&= 1 + x \sum_{\beta \in \{0,1\}^*} x^{|\beta|} y^{|\beta|_w} \left(1 + y^{[\beta \in \mathcal{S}]} \right) \\
&= 1 + 2x \sum_{\beta \in \{0,1\}^* \setminus \mathcal{S}} x^{|\beta|} y^{|\beta|_w} + x \sum_{\beta \in \mathcal{S}} x^{|\beta|} y^{|\beta|_w} (1+y) \\
&= 1 + 2x(F(x,y) - S(x,y)) + x(1+y)S(x,y).
\end{aligned}$$

Συνεπώς,

$$F(x,y) = 1 + 2xF(x,y) + x(y-1)S(x,y). \tag{2.17}$$

Κάθε λέξη $\alpha \in \mathcal{S}$ γράφεται υπό την μορφή

$$\alpha = w'\beta \text{ όπου } \beta \in \{0,1\}^*.$$

Προφανώς,

$$|w'\beta| = |w'| + |\beta| = |w| + |\beta| - 1$$

και

$$|w'\beta|_w = |\beta|_w,$$

διότι αν υπάρχει εμφάνιση του w εκτός της β τότε το κοινό τμήμα αυτής της εμφάνισης και της λέξης w' είναι σύνορο του w, το οποίο είναι άτοπο, αφού το w είναι απεριοδικό.

Επομένως,

$$\begin{aligned} S(x,y) &= \sum_{\alpha\in\mathcal{S}} x^{|\alpha|}y^{|\alpha|_w} \\ &= \sum_{\substack{\alpha=w'\beta \\ \beta\in\{0,1\}^*}} x^{|w'\beta|}y^{|w'\beta|_w} \\ &= \sum_{\beta\in\{0,1\}^*} x^{|w|+|\beta|-|1|}y^{|\beta|_w} \\ &= x^{|w|-1}\sum_{\beta\in\{0,1\}^*} x^{|\beta|}y^{|\beta|_w}. \end{aligned}$$

Συνεπώς,

$$S(x,y) = x^{|w|-1}F(x,y). \tag{2.18}$$

Από τις σχέσεις (2.17) και (2.18) προκύπτει ότι

$$\begin{aligned} F(x,y) &= 1 + 2xF(x,y) + (y-1)x^{|w|}F(x,y) \\ &= \frac{1}{1-2x-x^{|w|}(y-1)}. \end{aligned} \tag{2.19}$$

Αν τεθεί $H(\lambda) = (2 + x^{|w|-1}(y-1))\lambda$, τότε $F(x,y) = 1 + xH(F(x,y))$, οπότε, από το θεώρημα αντιστροφής του Lagrange, προκύπτει ότι

$$\begin{aligned} [x^n]F(x,y) &= \frac{1}{n}[\lambda^{n-1}]\left(H(\lambda+1)\right)^n \\ &= \frac{1}{n}[\lambda^{n-1}](2+(y-1)x^{|w|-1})^n(\lambda+1)^n \\ &= \frac{1}{n}(2+(y-1)x^{|w|-1})^n[\lambda^{n-1}]\sum_{j=0}^{n}\binom{n}{j}\lambda^j \\ &= \frac{1}{n}(2+(y-1)x^{|w|-1})^n\binom{n}{n-1} \\ &= (2+(y-1)x^{|w|-1})^n, \end{aligned}$$

οπότε, αν τεθεί $|w| = r$, προκύπτει ότι

$$\begin{aligned}
F(x,y) &= 1 + \sum_{n=1}^{\infty} (2 + (y-1)x^{r-1})^n x^n \\
&= \sum_{n=0}^{\infty} \sum_{j=0}^{n} \binom{n}{j} 2^{n-j} (y-1)^j x^{n+(r-1)j} \\
&= \sum_{l=0}^{\infty} \sum_{j=0}^{[l/r]} \binom{l-(r-1)j}{j} 2^{l-rj} (y-1)^j x^l \\
&= \sum_{l=0}^{\infty} \sum_{j=0}^{[l/r]} \sum_{k=0}^{j} \binom{l-(r-1)j}{j} \binom{j}{k} 2^{l-rj} (-1)^{j-k} y^k x^l \\
&= \sum_{l=0}^{\infty} \sum_{k=0}^{[l/r]} \sum_{j=k}^{[l/r]} \binom{l-(r-1)j}{j} \binom{j}{k} 2^{l-rj} (-1)^{j-k} y^k x^l.
\end{aligned}$$

Συνεπώς, ο αριθμός των δυαδικών λέξεων με μήκος n, και k εμφανίσεις ενός προτύπου w χωρίς σύνορα ισούται με

$$\sum_{j=k}^{[n/r]} \binom{l-(r-1)j}{j} \binom{j}{k} 2^{n-rj} (-1)^{j-k},$$

όπου $r = |w|$. □

Παρατήρηση. Το πρόβλημα της απαρίθμησης των λέξεων με συγκεκριμένες εμφανίσεις ενός ή περισσοτέρων προτύπων έχει μελετηθεί για πρώτη φορά από τους Goulden και Jackson [10]. Για μια πιο πρόσφατη προσέγγιση, βλέπε την εργασία των Noonan και Zeilberger [12], ή την εργασία των Bassino, Clément, Fayolle και Nicodème [1].

Κεφάλαιο 3

Κατασκευή δυαδικών λέξεων με περιορισμούς

Στο κεφάλαιο αυτό δίδονται ορισμένοι αλγόριθμοι κατασκευής αντικειμένων που συνδέονται με τις έννοιες των προηγούμενων κεφαλαίων. Συγκεκριμένα, δίνονται αλγόριθμοι επαναληπτικής και αναδρομικής κατασκευής σε λεξικογραφική διάταξη, καθώς και οι αλγόριθμοι ranking - unranking για τις λέξεις Fibonacci, τις λέξεις χωρίς zig-zag και τις λέξεις Dyck. Επίσης, σε ορισμένες περιπτώσεις, δίνονται και αλγόριθμοι κατασκευής σε διάταξη κώδικα Gray.

Θα συμβολίζουμε με $\bar{\alpha}$, τη **συμπληρωματική** λέξη της $\alpha \in \{0,1\}^*$, δηλαδή τη λέξη που προκύπτει αν μετατρέψουμε τα 0 της α σε 1 και τα 1 σε 0. Για παράδειγμα, $\overline{11001} = 00110$.

Επίσης, ορίζουμε για κάθε μη κενή λέξη $\alpha = \alpha_1\alpha_2\cdots\alpha_n$, τη δύναμη $\alpha^{\frac{m}{n}}$, $m \in \mathbb{N}$, ως

$$\alpha^{\frac{m}{n}} = \alpha^q\alpha_1\alpha_2\cdots\alpha_r,$$

όπου q και $r > 0$ είναι αντίστοιχα το πηλίκο και το υπόλοιπο της διαίρεσης m δια n.

3.1 Κατασκευή των λέξεων Fibonacci

Αν α είναι μια λέξη Fibonacci, τότε η $\bar{\alpha}$ προφανώς αποφεύγει το 11. Στο εξής θα συμβολίζουμε με Φ_n το σύνολο των δυαδικών λέξεων μήκους n που αποφεύγουν το 11, και θα ονομάζουμε τις λέξεις αυτές επίσης λέξεις Fibonacci (προφανώς το πλήθος τους είναι F_{n+1}). Σημειώνεται ότι πολλές φορές στη βιβλιογραφία οι λέξεις Fibonacci ορίζονται σε αυτή τη μορφή.

Οι αλγόριθμοι που ακολουθούν αναφέρονται στην κατασκευή των λέξεων που αποφεύγουν το 11. Ο λόγος αυτής της επιλογής είναι ότι έτσι γίνεται πιο εύκολη η διατύπωση και παρουσίαση των αλγορίθμων αυτών.

3.1.1 Επαναληπτική κατασκευή

Προκειμένου να κατασκευάσουμε το σύνολο των λέξεων Fibonacci επαναληπτικά και σε λεξικογραφική διάταξη, θα πρέπει να μπορούμε να υπολογίσουμε την επόμενη λέξη $\mathrm{next}(\alpha)$, για κάθε $\alpha \in \Phi_n$ (πλην της τελευταίας). Προφανώς, η μικρότερη λέξη και η μεγαλύτερη λέξη του Φ_n θα είναι οι 0^n και $(10)^{\frac{n}{2}}$ αντίστοιχα. Η λέξη $\mathrm{next}(\alpha)$ θα είναι της μορφής $\mathrm{next}(\alpha) = p1q'$, με $\alpha = p0q$, όπου p είναι το μεγιστικό κοινό πρόθεμα των $\mathrm{next}(\alpha)$ και α. Επιπλέον, θα είναι $q' = 00\cdots0$, αφού για οποιοδήποτε άλλο επίθεμα q' θα ήταν $\alpha \prec p100\cdots0 \prec p1q'$. Έτσι, ο αλγόριθμος υπολογισμού της $\mathrm{next}(\alpha)$ μπορεί να συνοψιστεί στα ακόλουθα βήματα:

- Βρες τη δεξιότερη εμφάνιση του 00 στην α.
- Άλλαξέ την σε 01 και θέσε όλα τα γράμματα δεξιά της ίσα με 0.
- Αν δεν υπάρχει τέτοια εμφάνιση, τότε
  - αν το αριστερότερο γράμμα είναι 0, τότε άλλαξέ το σε 1 και θέσε όλα τα υπόλοιπα ίσα με 0,
  - αλλιώς, δεν υπάρχει επόμενη.

Το πρόθεμα p, αν δεν είναι κενό, θα πρέπει να τελειώνει με 0, αλλιώς η $\mathrm{next}(\alpha)$ θα περιείχε εμφάνιση του 11, οπότε η αναζήτηση της δεξιότερης εμφάνισης του 00 αντιστοιχεί στην εύρεση του μεγιστικού κατάλληλου προθέματος p. Αν δεν υπάρχει εμφάνιση του 00 στην α, τότε το p είναι κενό και η α είναι της μορφής $(01)^{\frac{n}{2}}$ (οπότε η επόμενή της είναι η 10^{n-1}) ή $(10)^{\frac{n}{2}}$ (οπότε είναι η τελευταία).

Είσοδος: Το πλήθος $n \in \mathbb{N}^*$ των γραμμάτων της λέξης $\alpha = \alpha_1\alpha_2\cdots\alpha_n$.

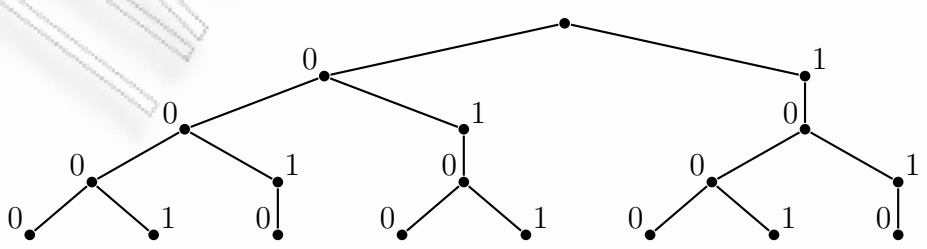
for $i \leftarrow 1$ **to** n **do** $\alpha_i \leftarrow 0$;
output (α);
while $1 = 1$ **do**
  $i \leftarrow n-1$;
  while $i > 0$ **and** $(\alpha_i = 1$ **or** $\alpha_{i+1} = 1)$ **do** $i \leftarrow i-1$;
  if $i > 0$ **then**
    $\alpha_{i+1} \leftarrow 1$;
    if $i < n-1$ **then for** $j \leftarrow i+2$ **to** n **do** $\alpha_j \leftarrow 0$;
  else if $\alpha_1 = 1$ **then exit**;
  else
    $\alpha_1 \leftarrow 1$;
    for $j \leftarrow 2$ **to** n **do** $\alpha_j \leftarrow 0$;
  endif
  output (α);
endw

Αλγόριθμος 3.1.1: Επαναληπτική κατασκευή του Φ_n σε λεξικογραφική σειρά.

3.1.2 Αναδρομική κατασκευή

Το σύνολο Φ_n μπορεί να αναπαρασταθεί ως ένα δυαδικό δένδρο με τις εξής ιδιότητες:

1. Κάθε κόμβος πλην της ρίζας είναι αριθμημένος με 0 ή 1.
2. Η ρίζα έχει δύο παιδιά, το αριστερό αριθμείται με 0 και το δεξί με 1.
3. Κάθε κόμβος που είναι αριθμημένος με 0 έχει δύο παιδιά, το αριστερό αριθμείται με 0 και το δεξί με 1.
4. Κάθε κόμβος που είναι αριθμημένος με 1 έχει ένα παιδί, που αριθμείται με 0.



Σχήμα 3.1: Το σύνολο Φ_4.

Κάθε λέξη του Φ_n κατασκευάζεται μονοσήμαντα, επιλέγοντας ένα μονοπάτι από τη ρίζα προς κάποιο φύλλο του δένδρου αυτού και διαβάζοντας τις επιγραφές των κόμβων με φορά προς το φύλλο.

Ο αναδρομικός αλγόριθμος που θα δοθεί στη συνέχεια, προσομοιώνει τη διάτρεξη σε προδιάταξη του δένδρου αυτού. Πιο συγκεκριμένα, δέχεται ως είσοδο ένα πρόθεμα p λέξης Fibonacci (το οποίο αντιστοιχεί στο μονοπάτι από τη ρίζα μέχρι τον τελευταίο κόμβο που έχει επισκεφθεί ο αλγόριθμος) καθώς και το πλήθος $k = n - |p|$ των γραμμάτων που απομένουν για να συμπληρωθεί μια λέξη του Φ_n, και κατασκευάζει αναδρομικά όλες τις λέξεις του Φ_n με πρόθεμα p. Σημειώνεται ότι κατά τη διάτρεξη σε προδιάταξη του συγκεκριμένου δένδρου, οι λέξεις θα προκύψουν σε λεξικογραφική διάταξη.

Είσοδος: ένα πρόθεμα p και το πλήθος k των γραμμάτων που απομένουν.
FR(p, k) **begin**
  if $k = 1$ **then**
    output ($p0$);
    output ($p1$);
  else if $k = 2$ **then**
    output ($p00$);
    output ($p01$);
    output ($p10$);
  else if $k > 2$ **then**
    FR ($p0$, $k-1$);
    FR ($p10$, $k-2$);
  endif
end

Αλγόριθμος 3.1.2: Η αναδρομική κατασκευή του Φ_n επιτυγχάνεται με την κλήση $\mathrm{FR}(\varepsilon, n)$.

Η ορθότητα του αλγορίθμου αποδεικνύεται επαγωγικά. Συγκεκριμένα, αρκεί να δειχθεί ότι η κλήση της αναδρομικής συνάρτησης $\mathrm{FR}(p, n)$ κατασκευάζει λεξικογραφικά το σύνολο των λέξεων Fibonacci με πρόθεμα p (το οποίο δεν τελειώνει με 1) και μήκος $p+|n|$, δηλαδή το σύνολο $\{p\gamma : \gamma \in \Phi_n\}$, για κάθε $n \in \mathbb{N}^*$. Για $n = 1$ και $n = 2$, ο ισχυρισμός προφανώς ισχύει, αφού οι ακολουθίες των λέξεων που κατασκευάζονται είναι αντίστοιχα οι $(p0, p1)$ και $(p00, p01, p10)$. Έστω ότι ο ισχυρισμός ισχύει για κάθε $k \leq n$, με $n \geq 2$. Τότε, για την κλήση της $\mathrm{FR}(p, n+1)$ θα είναι $n+1 > 2$, οπότε θα πραγματοποιηθούν διαδοχικά οι κλήσεις $\mathrm{FR}(p0, n)$ και $\mathrm{FR}(p10, n-1)$, οι οποίες, από την υπόθεση της επαγωγής, θα κατασκευάσουν αντίστοιχα σε λεξικογραφική σειρά τα σύνολα $\{p0\alpha : \alpha \in \Phi_n\}$ και $\{p10\beta : \beta \in \Phi_{n-1}\}$. Η ένωση των δύο συνόλων είναι το σύνολο των λέξεων Fibonacci μήκους $|p|+n+1$ με πρόθεμα p, δηλαδή το $\{p\gamma : \gamma \in \Phi_{n+1}\}$, και η λεξικογραφική σειρά διατηρείται, αφού τα στοιχεία του πρώτου συνόλου είναι μικρότερα από τα στοιχεία του δευτέρου, συνεπώς ο ισχυρισμός ισχύει

για $n+1$. Έτσι, η κλήση $\mathrm{FR}(\varepsilon, n)$ θα κατασκευάσει λεξικογραφικά όλο το Φ_n.

3.1.3 Αλγόριθμοι ranking - unranking

Ορίζουμε την απεικόνιση $r_n : \Phi_n \mapsto \mathbb{N}^*$, με $r_n(\varepsilon) = 1$ και $r_n(\alpha) = 1 + \sum_{i=1}^{n} \alpha_i F_i$, όπου $\alpha = \alpha_n \alpha_{n-1} \cdots \alpha_2 \alpha_1 \in \Phi_n$, $n \in \mathbb{N}^*$. Θα δειχθεί, με επαγωγή ως προς το μήκος n της λέξης, ότι η r_n είναι απεικόνιση ranking με βάση τη λεξικογραφική διάταξη.

Για $n = 1$, έχουμε ότι $r_1(0) = 1$ και $r_1(1) = 1 + F_1 = 2 = F_2$. Για $n = 2$, έχουμε ότι $r_2(00) = 1$, και $r_2(01) = 1 + F_1 = 2$ και $r_2(10) = 1 + F_2 = 3 = F_3$. Έστω ότι ο ισχυρισμός ισχύει για κάθε $k \in [n]$, όπου $n \geq 2$.

Αν $\alpha \in \Phi_{n+1}$, με $\alpha = \alpha_{n+1}\alpha_n \cdots \alpha_2 \alpha_1$, τότε θα είναι $\alpha = 0\beta$ ή $\alpha = 10\gamma$, με $\beta \in \Phi_n$, $\gamma \in \Phi_{n-1}$. Για την πρώτη περίπτωση ισχύει ότι

$$r_{n+1}(\alpha) = r_{n+1}(0\beta) = 0 + r_n(\beta),$$

οπότε, από την υπόθεση της επαγωγής, η α απεικονίζεται αμφιμονοσήμαντα σε έναν φυσικό αριθμό στο $[F_{n+1}]$. Για τη δεύτερη περίπτωση ισχύει ότι

$$r_{n+1}(\alpha) = r_{n+1}(10\gamma) = F_{n+1} + r_{n-1}(\gamma),$$

οπότε, από την υπόθεση της επαγωγής, η α απεικονίζεται αμφιμονοσήμαντα σε ένα φυσικό αριθμό στο διάστημα $[F_{n+1} + 1, F_{n+1} + F_n] = [F_{n+1} + 1, F_{n+2}]$. Επιπλέον, επειδή $0\beta \prec 10\gamma$ και $r_{n+1}(0\beta) < r_{n+1}(10\gamma)$, έπεται ότι για κάθε $\alpha, \alpha' \in \Phi_{n+1}$ θα είναι $r_{n+1}(\alpha) < r_{n+1}(\alpha') \Leftrightarrow \alpha \prec \alpha'$, ώστε η r_{n+1} είναι απεικόνιση ranking.

Έτσι, (αντιστρέφοντας τη σειρά των δεικτών των γραμμάτων της α) ορίζουμε το rank μιας λέξης Fibonacci ως

$$\mathrm{rank}(\alpha) = 1 + \sum_{i=1}^{n} \alpha_i F_{n-i+1}, \quad \alpha = \alpha_1 \alpha_2 \cdots \alpha_n \in \Phi_n, \quad n \in \mathbb{N}^*. \tag{3.1}$$

Από τη σχέση (3.1) είναι προφανές ότι κάθε λέξη $\alpha \in \Phi_n$ απεικονίζεται αμφιμονοσήμαντα στο φυσικό αριθμό $\mathrm{rank}(\alpha) - 1$. Είναι $\mathrm{rank}(\alpha) = 1$ αν και μόνο αν η α δεν περιέχει 1. Επιπλέον, αν $\beta \in \Phi_{n+k}$, είναι $\mathrm{rank}(\alpha) = \mathrm{rank}(\beta)$ αν και μόνο αν $\beta = 0^k\alpha$. Έτσι, δεδομένου ότι μια λέξη Fibonacci δεν περιέχει διαδοχικά 1, προκύπτει άμεσα το ακόλουθο Πόρισμα.

Πόρισμα 3.1.1. *Κάθε θετικός ακέραιος αριθμός γράφεται με μοναδικό τρόπο ως άθροισμα μη διαδοχικών όρων της ακολουθίας Fibonacci.*

Ο αλγόριθμος ranking προκύπτει άμεσα από τη σχέση (3.1).

Είσοδος: Η λέξη Fibonacci $\alpha = \alpha_1\alpha_2\cdots\alpha_n$.
Έξοδος: Ο φυσικός αριθμός $\text{rank}(\alpha)$.

$r \leftarrow 1$;
for $i \leftarrow 1$ **to** n **do** $r \leftarrow r + a_i F_{n-i+1}$;
return r;

Αλγόριθμος 3.1.3: Ο αλγόριθμος ranking για λέξεις Fibonacci.

Για την αντίστροφη διαδικασία, του unranking, θεωρούμε δεδομένα το $r = \text{rank}(\alpha)$ και το μήκος n της λέξης, και ζητάμε τη λέξη α. Προφανώς θα είναι $1 \leq \text{rank}(\alpha) \leq F_{n+1}$. Αν $\text{rank}(\alpha) > F_n$, τότε το αριστερότερο γράμμα είναι το 1 (όπως προκύπτει από τα προηγούμενα), δηλαδή $\alpha = 1\beta$. Για την $\beta \in \Phi_{n-1}$ προφανώς ισχύει ότι $\text{rank}(\beta) = \text{rank}(\alpha) - F_n$. Έτσι, αφαιρούμε το F_n από το r, ώστε να είναι $r = \text{rank}(\beta)$, και επαναλαμβάνουμε τη διαδικασία για τη λέξη β. Αν $\text{rank}(\alpha) \leq F_n$, τότε $\alpha = 0\beta$, και $\text{rank}(\beta) = \text{rank}(\alpha)$, οπότε συνεχίζουμε χωρίς να κάνουμε την αφαίρεση.

Είσοδος: Οι φυσικοί αριθμοί r και n.
Έξοδος: Η λέξη Fibonacci $\alpha = \alpha_1\alpha_2\cdots\alpha_n$ με $r = \text{rank}(\alpha)$.

$\alpha \leftarrow \varepsilon$;
for $i \leftarrow 1$ **to** n **do** **if** $r > F_{n-i+1}$ **then**
  $\alpha \leftarrow 1\alpha$;
  $r \leftarrow r - F_{n-i+1}$;
else
  $\alpha \leftarrow 0\alpha$;
endif
return α;

Αλγόριθμος 3.1.4: Ο αλγόριθμος unranking για λέξεις Fibonacci.

Παράδειγμα. για τη λέξη $\alpha = 0100101001$ μήκους 10 είναι

$$\text{rank}(\alpha) = 1 + \sum_{i=1}^{n} \alpha_i F_{n-i+1} = 1 + F_9 + F_6 + F_4 + F_1 = 1 + 55 + 13 + 5 + 1 = 75.$$

Η ανάκτηση της α από το 75 περιγράφεται στον επόμενο πίνακα.

	F_{10}	F_9	F_8	F_7	F_6	F_5	F_4	F_3	F_2	F_1
F_n	89	55	34	21	13	8	5	3	2	1
r	75	75	20	20	20	7	7	2	2	2
α	0	1	0	0	1	0	1	0	0	1

Αρχικά η δεύτερη και τρίτη γραμμή του πίνακα δεν περιέχουν στοιχεία. Γράφουμε στο πρώτο κελί της δεύτερης γραμμής το $r = 75$. Αν ο αριθμός Fibonacci στο αντίστοιχο κελί

της πρώτης γραμμής είναι μικρότερος, τότε τον αφαιρούμε από το r και συμπληρώνουμε το αντίστοιχο κελί της τρίτης γραμμής με 1. Αλλιώς, δεν κάνουμε την αφαίρεση και συμπληρώνουμε το αντίστοιχο κελί της τρίτης γραμμής με 0. Γράφουμε το νέο r στο επόμενο κελί της δεύτερης γραμμής και επαναλαμβάνουμε.

3.1.4 Κώδικας Gray

Ορίζουμε αναδρομικά την ακολουθία (από δυαδικές λέξεις) G_n ως εξής:

$$G_1 = (0, 1), \quad G_2 = (01, 00, 10), \quad \text{και} \quad G_n = 0\bar{G}_{n-1}10\bar{G}_{n-2}, n \geq 3,$$

δηλαδή η ακολουθία G_n προκύπτει από την παράθεση των ακολουθιών $0\bar{G}_{n-1}$ και $10\bar{G}_{n-2}$, όπου $\bar{G}_{n-1}$ είναι η ακολουθία G_{n-1} σε αντίστροφη διάταξη (αντίστοιχα προκύπτει η $\bar{G}_{n-2}$) και $0\bar{G}_{n-1}$ είναι η ακολουθία που προκύπτει αν σε κάθε λέξη της $\bar{G}_{n-1}$ προσθέσουμε το πρόθεμα 0 (αντίστοιχα προκύπτει η $10\bar{G}_{n-2}$).

Για παράδειγμα, είναι $\bar{G}_1 = (1, 0), \quad \bar{G}_2 = (10, 00, 01)$ και

$$G_3 = 0\bar{G}_2 10\bar{G}_1 = (010, 000, 001, 101, 100).$$

Οι G_1, G_2 και G_3 είναι κώδικες Gray των συνόλων Φ_1, Φ_2 και Φ_3.

Πρόταση 3.1.2. *Η ακολουθία G_n είναι κώδικας Gray του συνόλου Φ_n, για κάθε $n \in \mathbb{N}^*$. Επιπλέον, η πρώτη και η τελευταία λέξη του G_n είναι αντίστοιχα οι first$(G_n) = 0(100)^{\frac{n-1}{3}}$ και last$(G_n) = (100)^{\frac{n}{3}}$.*

Απόδειξη. Θα χρησιμοποιηθεί επαγωγή ως προς το n. Για $n = 1$ και $n = 2$, η πρόταση προφανώς ισχύει. Έστω ότι ισχύει για κάθε $k \in [n]$, με $n \geq 2$. Τότε, θα είναι

$$\text{first}(G_{n+1}) = 0\,\text{first}(\bar{G}_n) = 0\,\text{last}(G_n) = 0(100)^{\frac{n}{3}}$$

και

$$\text{last}(G_{n+1}) = 10\,\text{last}(\bar{G}_{n-1}) = 10\,\text{first}(G_{n-1}) = 100(100)^{\frac{n-2}{3}} = (100)^{1+\frac{n-2}{3}} = (100)^{\frac{n+1}{3}}.$$

Από την υπόθεση της επαγωγής προκύπτει άμεσα ότι οι ακολουθίες $0\bar{G}_n$ και $10\bar{G}_{n-1}$ είναι κώδικες Gray των συνόλων $\{0\alpha : \alpha \in \Phi_n\}$ και $\{10\alpha : \alpha \in \Phi_{n-1}\}$, τα οποία αποτελούν (λόγω της διάσπασης) διαμέριση του Φ_{n+1}. Επομένως, προκειμένου να δειχθεί ότι η G_n είναι κώδικας Gray του Φ_n, αρκεί να δειχθεί ότι οι $0\,\text{last}(\bar{G}_{n-1})$ και $10\,\text{first}(\bar{G}_{n-2})$ έχουν απόσταση Hamming ίση με 1. Αυτό αληθεύει, αφού είναι

$$0\,\text{last}(\bar{G}_{n-1}) = 0\,\text{first}(G_{n-1}) = 00(100)^{\frac{n-2}{3}}$$

και

$$10\,\mathrm{first}(\bar{G}_{n-2}) = 10\,\mathrm{last}(G_{n-2}) = 10(100)^{\frac{n-2}{3}}.$$

□

Από τον ορισμό της ακολουθίας G_n, προκύπτει άμεσα ότι

$$\bar{G}_1 = (1,0), \quad \bar{G}_2 = (10,00,01) \quad \text{και} \quad \bar{G}_n = 10G_{n-2}0G_{n-1}, n \geq 3.$$

Στη συνέχεια, δίνεται αναδρομικός αλγόριθμος κατασκευής του Φ_n σε κώδικα Gray, με βάση τους δύο αναδρομικούς τύπους των G_n και $\bar{G}_n$. Ο αλγόριθμος αποτελείται από δύο αναδρομικές συναρτήσεις GrayF(p,k) και rGrayF(p,k) οι οποίες κατασκευάζουν, κατ' αναλογία με τον αλγόριθμο 3.1.2, τις λέξεις Fibonacci με πρόθεμα p (που δεν τελειώνει με 1) και μήκος $|p|+k$, σε κώδικα Gray και αντίστροφο κώδικα Gray αντίστοιχα. Έτσι, το σύνολο Φ_n κατασκευάζεται με την εκτέλεση της GrayF(ε, n). Η απόδειξη της ορθότητας του αλγορίθμου είναι όμοια με εκείνη του αλγορίθμου 3.1.2.

Παράδειγμα. Οι κώδικες Gray των λέξεων Fibonacci μήκους $n \in [5]$.

G_1
0
1

G_2
01
00
10

G_3
010
000
001
101
100

G_4
0100
0101
0001
0000
0010
1010
1000
1001

G_5
01001
01000
01010
00010
00000
00001
00101
00100
10100
10101
10001
10000
10010

Είσοδος: ένα πρόθεμα p και το πλήθος k των γραμμάτων που απομένουν.

```
GrayF(p, k) begin
    if k = 1 then
        output (p0);
        output (p1);
    else if k = 2 then
        output (p01);
        output (p00);
        output (p10);
    else if k > 2 then
        rGrayF (p0, k − 1);
        rGrayF (p10, k − 2);
    endif
end
```

Αλγόριθμος 3.1.5: Η αναδρομική κατασκευή του pG_k.

Είσοδος: ένα πρόθεμα p και το πλήθος k των γραμμάτων που απομένουν.

```
rGrayF(p, k) begin
    if k = 1 then
        output (p1);
        output (p0);
    else if k = 2 then
        output (p10);
        output (p00);
        output (p01);
    else if k > 2 then
        GrayF (p10, k − 2);
        GrayF (p0, k − 1);
    endif
end
```

Αλγόριθμος 3.1.6: Η αναδρομική κατασκευή του $p\bar{G}_k$.

3.2 Κατασκευή των δυαδικών λέξεων χωρίς zig-zag

3.2.1 Επαναληπτική κατασκευή

Δεδομένης μιας λέξης $\alpha = \alpha_1\alpha_2\cdots\alpha_n \in \mathcal{Z}_n$, για την εύρεση της $\text{next}(\alpha)$ αρκεί να εντοπίσουμε το δεξιότερο 0 της α, του οποίου δεν προηγείται 10, να το θέσουμε ίσο με 1 και να ελαχιστοποιήσουμε το επίθεμα που ακολουθεί το γράμμα αυτό (λαμβάνοντας φυσικά υπόψη ότι η λέξη $\text{next}(\alpha)$ ανήκει στο $\mathcal{Z}_n$). Έτσι, αν υποθέσουμε ότι το γράμμα αυτό είναι το α_i, $i \in [n]$, τότε θα είναι $\alpha = p0q$, με $|p| = i-1$, και $\text{next}(\alpha) = p1q'$, με q' το ελάχιστο δυνατό επίθεμα. Διακρίνουμε περιπτώσεις:

1. Αν $i = 1$, τότε $\text{next}(\alpha) = 10^{n-1}$.
2. Αν $i = n$, τότε $\text{next}(\alpha) = p1$.
3. Αν $1 < i < n$ και το p δεν τελειώνει με 0, τότε $\text{next}(\alpha) = p10^{n-i}$.
4. Αν $1 < i < n$ και το p τελειώνει με 0, τότε $\text{next}(\alpha) = p110^{n-i-1}$.
5. Αν δεν υπάρχει τέτοιο i, τότε $\alpha = 1^n$ και δεν υπάρχει η $\text{next}(\alpha)$.

Από τα παραπάνω, προκύπτει ο ακόλουθος επαναληπτικός αλγόριθμος για τη λεξικογραφική κατασκευή του $\mathcal{Z}_n$.

Είσοδος: Το πλήθος $n \in \mathbb{N}^*$ των γραμμάτων της λέξης $\alpha = \alpha_1\alpha_2\cdots\alpha_n$.

for $i \leftarrow 1$ **to** n **do** $\alpha_i \leftarrow 0$;
while $i > 0$ **do**
  `output` (α);
  $i \leftarrow n$;
  while $i > 0$ **and** $(\alpha_i = 1$ **or** `isPrecededBy10`$(\alpha, i))$ **do** $i \rightarrow i-1$;
  if $i = 1$ **then** $\alpha \leftarrow 10^{n-1}$;
  if $i = n$ **then** $\alpha_n \leftarrow 1$;
  if $1 < i < n$ **then**
    $\alpha_i \leftarrow 1$;
    for $j \leftarrow i+1$ **to** n **do** $\alpha_j \leftarrow 0$;
    if $\alpha_{i-1} = 1$ **then** $\alpha_{i+1} \leftarrow 1$;
  endif
endw

Αλγόριθμος 3.2.1: Επαναληπτική κατασκευή του $\mathcal{Z}_n$ σε λεξικογραφική σειρά.

Η βοηθητική συνάρτηση isPrecededBy10 ορίζεται ως εξής:

$$\text{isPrecededBy10}(\alpha, i) = \begin{cases} \text{αληθής}, & \text{αν } 3 \leq i \leq n \text{ και } \alpha_{i-2} = 1, \alpha_{i-1} = 0, \\ \text{ψευδής}, & \text{αν αλλιώς}, \end{cases} \quad \text{για κάθε } \alpha \in \mathcal{Z}_n.$$

3.2.2 Αναδρομική κατασκευή

Για την εύρεση αναδρομικού αλγορίθμου για τη λεξικογραφική κατασκευή του $\mathcal{Z}_n$, θεωρούμε την ακόλουθη διαμέριση, για κάθε $n \geq 2$:

$$\mathcal{Z}_n = \mathcal{Z}_n^{00} \cup \mathcal{Z}_n^{01} \cup \mathcal{Z}_n^{10} \cup \mathcal{Z}_n^{11},$$

όπου $\mathcal{Z}_n^w$ είναι το σύνολο των λέξεων του $\mathcal{Z}_n$ που ξεκινούν με $w \in \{0,1\}^*$. Προφανώς, τα τέσσερα αυτά σύνολα είναι μη κενά και ανά δύο ξένα, για κάθε $n \geq 2$. Στο εξής, θα συμβολίζουμε επίσης με $\mathcal{Z}_n^w$ και την ακολουθία των στοιχείων του $\mathcal{Z}_n^w$ σε αύξουσα σειρά ως προς τη λεξικογραφική διάταξη. Έτσι, η παράθεση (των ακολουθιών) $\mathcal{Z}_n^{00}\mathcal{Z}_n^{01}\mathcal{Z}_n^{10}\mathcal{Z}_n^{11}$ θα ισούται με την ακολουθία $\mathcal{Z}_n$.

Οι ακολουθίες αυτές παράγονται αναδρομικά σύμφωνα με τους ακόλουθους κανόνες

1. $\mathcal{Z}_n^{00} = 0\mathcal{Z}_{n-1}^{00}0\mathcal{Z}_{n-1}^{01}$,
2. $\mathcal{Z}_n^{01} = 0\mathcal{Z}_{n-1}^{11}$,
3. $\mathcal{Z}_n^{10} = 1\mathcal{Z}_{n-1}^{00}$,
4. $\mathcal{Z}_n^{11} = 1\mathcal{Z}_{n-1}^{10}1\mathcal{Z}_{n-1}^{11}$,

για κάθε $n > 2$ και $\mathcal{Z}_2^{00} = (00)$, $\mathcal{Z}_2^{01} = (01)$, $\mathcal{Z}_2^{10} = (10)$, $\mathcal{Z}_2^{11} = (11)$. Με βάση τους παραπάνω κανόνες, προκύπτει ο αναδρομικός αλγόριθμος 3.2.2.

Η μεταβλητή f υποδεικνύει ποιος από τους τέσσερεις κανόνες θα εφαρμοστεί, παίρνοντας τιμές 00, 01, 10, 11. Αν έχει κάποια άλλη τιμή, τότε εφαρμόζονται όλοι οι κανόνες (η περίπτωση αυτή προκύπτει μόνο κατά την πρώτη κλήση $\text{ZR}(\varepsilon, \varepsilon, n)$). Η συνάρτηση $\text{ZR}(p, f, k)$, για $k > 2$ υπολογίζει αναδρομικά το σύνολο των λέξεων μήκους $|p| + k$ χωρίς zig-zag, με πρόθεμα $p0f$, (αντίστοιχα $p1f$), αν $0f \neq 010$, (αντίστοιχα $1f \neq 101$). Η σειρά με την οποία γίνονται οι αναδρομικές κλήσεις εξασφαλίζει τη λεξικογραφική κατασκευή των λέξεων.

Σημειώνεται ότι αν κάθε ένας από τους τέσσερεις κανόνες υλοποιηθεί ως ξεχωριστή αναδρομική συνάρτηση, τότε προκύπτει πιο αποδοτική υλοποίηση, αφού η ύπαρξη της μεταβλητής f θα είναι περιττή, οπότε θα αποφευχθούν όλες οι εντολές ελέγχου της τιμής της. Ο αλγόριθμος παρουσιάζεται εδώ με τη μορφή μίας ενιαίας αναδρομικής συνάρτησης, ώστε να είναι πιο εύκολα κατανοητός.

Είσοδος: ένα πρόθεμα p, μια λέξη $f \in \{0,1\}^*$ και το πλήθος k των γραμμάτων που απομένουν.

```
ZR(p, f, k) begin
  if k = 2 then
    if f = 00 then output (p00);
    else if f = 01 then output (p01);
    else if f = 10 then output (p10);
    else if f = 11 then output (p11);
    else
      output (p00);
      output (p01);
      output (p10);
      output (p11);
    endif
  else if k > 2 then
    if f = 00 then
      ZR (p0, 00, k − 1);
      ZR (p0, 01, k − 1);
    else if f = 01 then ZR (p0, 11, k − 1);
    else if f = 10 then ZR (p1, 10, k − 1);
    else if f = 11 then
      ZR (p1, 10, k − 1);
      ZR (p1, 11, k − 1);
    else
      ZR (p0, 00, k − 1);
      ZR (p0, 01, k − 1);
      ZR (p0, 11, k − 1);
      ZR (p1, 00, k − 1);
      ZR (p1, 10, k − 1);
      ZR (p1, 11, k − 1);
    endif
  endif
end
```

Αλγόριθμος 3.2.2: Αναδρομικός αλγόριθμος λεξικογραφικής κατασκευής του $\mathcal{Z}_k$, $k \geq 2$. Το $\mathcal{Z}_n$ κατασκευάζεται με την κλήση $\mathrm{ZR}(\varepsilon, \varepsilon, n)$.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

3.2.3 Αλγόριθμοι ranking - unranking

Έστω $\mathcal{A}_n$ το σύνολο των λέξεων μήκους n χωρίς zig-zag, οι οποίες δεν ξεκινούν με 10, $\mathcal{A} = \bigcup_{n\geq 0} \mathcal{A}_n$, και $\bar{\mathcal{A}}_n = \{\bar{\alpha} : \alpha \in \mathcal{A}_n\}$, όπου $\bar{\alpha}$ είναι η συμπληρωματική λέξη της α. Προφανώς, το $\bar{\mathcal{A}}_n$ είναι το σύνολο των λέξεων χωρίς zig-zag μήκους n που δεν ξεκινούν με 01, και $|\bar{\mathcal{A}}_n| = |\mathcal{A}_n|$, για κάθε $n \in \mathbb{N}$.

Κάθε λέξη $\alpha \in \mathcal{A}_n$ γράφεται με μοναδικό τρόπο στη μορφή

$$\alpha = 0\beta, \quad \text{ή} \quad \alpha = 11\gamma, \quad \beta \in \mathcal{A}_{n-1}, \gamma \in \bar{\mathcal{A}}_{n-2},$$

για κάθε $n \geq 2$. Έτσι, προκύπτει ο αναδρομικός τύπος $|\mathcal{A}_n| = |\mathcal{A}_{n-1}|+|\mathcal{A}_{n-2}|$ και δεδομένου ότι $|\mathcal{A}_0| = 1$ και $|\mathcal{A}_1| = 2$, έπεται ότι $|\mathcal{A}_n| = F_{n+1}$.

Επιπλέον, κάθε μη κενή λέξη $\alpha \in \mathcal{Z}_n$ γράφεται με μοναδικό τρόπο στη μορφή

$$\alpha = 0\beta, \quad \text{ή} \quad \alpha = 1\gamma, \quad \beta \in \mathcal{A}_{n-1}, \gamma \in \bar{\mathcal{A}}_{n-1},$$

οπότε το σύνολο $\mathcal{Z}_n^0$ των λέξεων του $\mathcal{Z}_n$ που ξεκινούν με 0 θα έχει πληθάριθμο $|\mathcal{Z}_n^0| = |\mathcal{A}_{n-1}| = F_n$.

Επομένως, αν $\alpha = \alpha_1\alpha_2\cdots\alpha_n \in \mathcal{Z}_n$, με $a_i = 1$ για κάποιο $i \in [n]$, οπότε $\alpha = p1q$, όπου $|p| = i-1$, τότε η α θα είναι μεγαλύτερη λεξικογραφικά από κάθε λέξη β του $\mathcal{Z}_n$ με πρόθεμα $p0$, δηλαδή $\beta \in \mathcal{Z}_n^{p0}$. Αν p τελειώνει με 01, τότε προφανώς $|\mathcal{Z}_n^{p0}| = 0$. Για την αντίθετη περίπτωση, διακρίνουμε τις ακόλουθες περιπτώσεις:

1. Αν $p = \varepsilon$, τότε $|\mathcal{Z}_n^{p0}| = |\mathcal{Z}_n^0| = F_n$.

2. Αν $p = p'1$, τότε $|\mathcal{Z}_n^{p0}| = |\mathcal{Z}_n^{p'10}| = |\mathcal{Z}_{n-i}^0| = F_{n-i}$, αφού για κάθε $\beta \in \mathcal{Z}_n^{p'10}$ είναι $\beta = p'10s$, με $s \in \mathcal{Z}_{n-i}^0$.

3. Αν $p = p'0$, τότε $|\mathcal{Z}_n^{p0}| = |\mathcal{Z}_n^{p'00}| = |\mathcal{A}_{n-i}| = F_{n-i+1}$, αφού για κάθε $\beta \in \mathcal{Z}_n^{p'00}$ είναι $\beta = p'00s$, με $s \in \mathcal{A}_{n-i}$.

Έτσι, προκειμένου να υπολογίσουμε το $r = \text{rank}(\alpha)$, θα θέσουμε αρχικά $r = 1$, και στη συνέχεια, διαβάζοντας ένα προς ένα τα γράμματα της α από αριστερά προς τα δεξιά, κάθε φορά που θα συναντάμε κάποιο 1 θα εξετάζουμε ποια από τις παραπάνω περιπτώσεις ισχύει, προσθέτοντας στο r τον αντίστοιχο αριθμό Fibonacci.

Είσοδος: Η λέξη $\alpha = \alpha_1\alpha_2\cdots\alpha_n \in \mathcal{Z}_n$.
Έξοδος: Ο φυσικός αριθμός $\mathrm{rank}(\alpha)$.

$r \leftarrow 1$;
if $\alpha_1 = 1$ **then** $r \leftarrow r + F_n$;
if $n = 1$ **then return** r ;
for $i \leftarrow 2$ **to** n **do**
  if $\alpha_i = 1$ **and not** `isPrecededBy01` (α, i) **then**
    if $\alpha_{i-1} = 1$ **then** $r \leftarrow r + F_{n-i}$;
    else $r \leftarrow r + F_{n-i+1}$;
  endif
endfor
return r;

Αλγόριθμος 3.2.3: Ο αλγόριθμος ranking για λέξεις χωρίς zig-zag.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ

Έξοδος: Η λέξη $\alpha = \alpha_1\alpha_2\cdots\alpha_n \in \mathcal{Z}_n$.
Είσοδος: Ο φυσικός αριθμός $r = \text{rank}(\alpha)$ και το μήκος n της λέξης α.

$\alpha \leftarrow \varepsilon$;
if $r > F_n$ **then**
  $\alpha \leftarrow \alpha 1$;
  $r \leftarrow r - F_n$;
else
  $\alpha \leftarrow \alpha 0$;
endif
for $i \leftarrow 1$ **to** n **do**
  if `isPrecededBy01` (α, i) **then** $\alpha \leftarrow \alpha 1$;
  else if `isPrecededBy10` (α, i) **then** $\alpha \leftarrow \alpha 0$;
  else
    if $\alpha_{i-1} = 1$ **then**
      if $r > F_{n-i}$ **then**
        $r \leftarrow r - F_{n-i}$;
        $\alpha \leftarrow \alpha 1$;
      else
        $\alpha \leftarrow \alpha 0$;
      endif
    else
      if $r > F_{n-i+1}$ **then**
        $r \leftarrow r - F_{n-i+1}$;
        $\alpha \leftarrow \alpha 1$;
      else
        $\alpha \leftarrow \alpha 0$;
      endif
    endif
  endif
endfor
return α;

Αλγόριθμος 3.2.4: Ο αλγόριθμος unranking για λέξεις χωρίς zig-zag.

Η βοηθητική συνάρτηση isPrecededBy01 ορίζεται ως εξής:

$$\text{isPrecededBy01}(\alpha, i) = \begin{cases} \text{αληθής}, & \text{αν } 3 \le i \le n+1 \text{ και } \alpha_{i-2} = 0, \alpha_{i-1} = 1, \\ \text{ψευδής}, & \text{αν αλλιώς}, \end{cases}$$

για κάθε $\alpha \in \mathcal{Z}_n$.

3.2.4 Κώδικας Gray

Πρόταση 3.2.1. *Για κάθε άρτιο $n > 2$ δεν υπάρχει κώδικας Gray για τις δυαδικές λέξεις μήκους n χωρίς zig-zag.*

Απόδειξη. Έστω ότι για κάποιο άρτιο $n > 2$ υπάρχει κώδικας Gray για το σύνολο των λέξεων μήκους n χωρίς zig-zag. Είναι προφανές ότι το πλήθος των 1 από τη μια λέξη στην επόμενη θα αλλάζει αρτιότητα. Αν d_n είναι η διαφορά του πλήθους των λέξεων με άρτιο πλήθος 1 πλην το πλήθος των λέξεων με περιττό πλήθος 1, θα αποδείξουμε ότι, $d_{2k} \geq 2$, για κάθε φυσικό $k > 1$, οπότε δεν θα υπάρχει κώδικας Gray.

Έστω $E(x, y)$ και $O(x, y)$ οι γεννήτριες συναρτήσεις των συνόλων των λέξεων με άρτιο και περιττό πλήθος 1 αντίστοιχα, χωρίς zig-zag (οι μεταβλητές x και y μετρούν το μήκος και το πλήθος των 1). Θα είναι

$$E(x,y) = \frac{F(x,y) + F(x,-y)}{2} \quad \text{και} \quad O(x,y) = \frac{F(x,y) - F(x,-y)}{2},$$

όπου $F(x,y) = \dfrac{1 + x^2 + x^4y^2}{1 - x - xy + x^2y - x^4y^2}$ είναι η γεννήτρια συνάρτηση του συνόλου $\mathcal{Z}$, όπως υπολογίστηκε στα προηγούμενα. Θέτοντας $D(x, y) = E(x, y) - O(x, y)$, έχουμε ότι

$$D(x,y) = F(x,-y) = \frac{1 - x^2 + x^4y^2}{1 - x + xy - x^2y - x^4y^2} \text{ και } D(x) = D(x,1) = \frac{1 - x^2 + x^4}{1 - x^2 - x^4}.$$

Ο συντελεστής $[x^n]D(x)$ θα ισούται με τη ζητούμενη διαφορά d_n. Συγκεκριμένα, είναι

$$\begin{aligned} D(x) &= \frac{1 - x^2 + x^4}{1 - x^2(1 + x^2)} = (1 - x^2 + x^4) \sum_{n \geq 0} x^{2n}(1 + x^2)^n \\ &= (1 - x^2 + x^4) \sum_{n \geq 0} \sum_{j=0}^{n} \binom{n}{j} x^{2n+2j} \overset{2n+2j=k}{=} (1 - x^2 + x^4) \sum_{k \geq 0} \sum_{j=0}^{[\frac{k}{4}]} \binom{\frac{k}{2} - j}{j} x^k, \end{aligned}$$

οπότε

$$\begin{aligned} d_{2k} = [x^{2k}]D(x) &= \sum_{j=0}^{[\frac{k}{2}]} \binom{k-j}{j} - \sum_{j=0}^{[\frac{k-1}{2}]} \binom{k-1-j}{j} + \sum_{j=0}^{[\frac{k-2}{2}]} \binom{k-2-j}{j} \\ &= \sum_{j=0}^{[\frac{k}{2}]} \binom{(k-1)-j+1}{j} - \sum_{j=0}^{[\frac{k-1}{2}]} \binom{(k-2)-j+1}{j} + \sum_{j=0}^{[\frac{k-2}{2}]} \binom{(k-3)-j+1}{j} \\ &= F_k - F_{k-1} + F_{k-2} = 2F_{k-2}, \end{aligned}$$

για κάθε $k > 1$. □

3.3 Κατασκευή των λέξεων Dyck

Κάθε λέξη Dyck $\alpha = \alpha_1\alpha_2\cdots\alpha_{2n}$, $n \in \mathbb{N}^*$, μπορεί να κωδικοποιηθεί από την ακολουθία $e(\alpha) = e_1e_2\cdots e_n$, όπου

$$e_i = k \quad \text{αν και μόνο αν} \quad |\alpha_1\alpha_2\cdots\alpha_{k-1}|_1 = i-1 \quad \text{και} \quad \alpha_k = 1, \quad i \in [n], \quad k \in [2n-1].$$

Πιο απλά, η τιμή του όρου e_i ισούται με τη θέση του i-οστού 1 στη λέξη α.

Προφανώς, η $e(\alpha)$ είναι γνησίως αύξουσα, με $e_1 = 1$ (αφού το πρώτο γράμμα μιας μη κενής λέξης Dyck είναι πάντα 1). Επιπλέον, ισχύει ότι

$$i \leq e_i \leq 2i-1, \quad \text{για κάθε } i \in [n].$$

Πράγματι, αν θεωρήσουμε το πρόθεμα $p1$ της α, με $|p| = k-1$, $k \in [2n-2]$, και $|p|_1 = i-1$, $i \in [n-1]$, τότε θα είναι $e_i = k$ (αφού το 1 που ακολουθεί το p είναι το i-οστό 1 της α και βρίσκεται στη θέση k). Επιπλέον, επειδή η α είναι λέξη Dyck, ισχύει ότι

$$|p|_1 \leq |p| \leq 2|p|_1 \Rightarrow i-1 \leq k-1 \leq 2i-2 \Rightarrow i \leq e_i \leq 2i-1.$$

Σημειώνεται ότι το e_i μπορεί να πάρει οποιαδήποτε τιμή από i έως και $2i-1$, αφού και το $|p|$ μπορεί να πάρει οποιαδήποτε τιμή από $|p|_1$ έως και $2|p|_1$.

Υπάρχει λοιπόν αμφιμονοσήμαντη απεικόνιση μεταξύ του $\mathcal{D}_n$, $n \in \mathbb{N}^*$, και του συνόλου $\mathcal{E}_n$ των ακολουθιών $e = e_1e_2\cdots e_n$, με $e_1 < e_2 < \cdots < e_n$ και $i \leq e_i \leq 2i-1$, $i \in [n]$. Η κωδικοποίηση αυτή των λέξεων Dyck θα χρησιμοποιηθεί στους αλγορίθμους που παρουσιάζονται στη συνέχεια, αφού οδηγεί πολλές φορές σε πιο αποδοτική υλοποίηση.

3.3.1 Επαναληπτική κατασκευή

Αν $\alpha, \beta \in \mathcal{D}_n$, $n \in \mathbb{N}^*$, τότε

$$\alpha \prec \beta \text{ αν και μόνο αν } e(\beta) \prec e(\alpha).$$

Πράγματι, αν υποτεθεί ότι $\alpha \prec \beta$, οπότε θα υπάρχει το μέγιστο κοινό πρόθεμα p των α, β, με $|p| = k$ και $|p|_1 = i-1$, τότε θα είναι $\alpha = p0\alpha'$ και $\beta = p1\beta'$. Οι ακολουθίες $e(\alpha) = e_1e_2\cdots e_n$ και $e(\beta) = e'_1e'_2\cdots e'_n$ θα συμπίπτουν στους $i-1$ πρώτους όρους, ενώ $e_i > e'_i$. Έτσι, αν οι $e(\alpha)$ και $e(\beta)$ θεωρηθούν ως λέξεις, θα ισχύει ότι $e(\beta) \prec e(\alpha)$. Ανάλογα προκύπτει και το αντίστροφο. Συνεπώς, προκειμένου να κατασκευάσουμε λεξικογραφικά το $\mathcal{D}_n$, αρκεί να κατασκευάσουμε το $\mathcal{E}_n$ σε αντίστροφη λεξικογραφική διάταξη.

Αν $e \in \mathcal{E}_n$, με $e = e_1e_2\cdots e_n$, η προηγούμενη λέξη της e προκύπτει αν μειωθεί κατά 1 το

δεξιότερο γράμμα της e, το οποίο είναι τουλάχιστον κατά 2 μεγαλύτερο από το προηγούμενό του, και τα υπόλοιπα δεξιά από αυτό πάρουν τη μέγιστη δυνατή τιμή. Δηλαδή, είναι

$$\text{previous}(e) = e'_1 e'_2 \cdots e'_{i-1} e'_i e'_{i+1} \cdots e'_n,$$

όπου $i = \max\{i \in [n] : e_i > e_{i-1} + 1\}$ και $e'_j = \begin{cases} e_j, & \text{αν } 1 \le j < i \\ e_i - 1, & \text{αν } j = i \\ 2j - 1, & \text{αν } i+1 \le j \le n. \end{cases}$

Ειδικά, η ακολουθία $123\cdots n$ δεν έχει προηγούμενη, αφού είναι η μικρότερη λεξικογραφικά ακολουθία του $\mathcal{E}_n$. Τα παραπάνω προκύπτουν άμεσα από τους ορισμούς του $\mathcal{E}_n$ και της λεξικογραφικής διάταξης.

Input: Το πλήθος $n \in \mathbb{N}^*$ των στοιχείων της ακολουθίας $e = e_1 e_2 \cdots e_n$.

for $i \leftarrow 1$ **to** n **do** $e_i \leftarrow 2i - 1$;
`output` (e);
while $1 = 1$ **do**
  $i \leftarrow n$;
  while $i > 1$ **and** $e_i < e_{i-1} + 2$ **do** $i \leftarrow i - 1$;
  if $i > 1$ **then**
    $e_i \leftarrow e_i - 1$;
    for $j \leftarrow i + 1$ **to** n **do** $e_j \leftarrow 2j - 1$;
    `output` (e);
  else
    exit;
  endif
endw

Αλγόριθμος 3.3.1: Επαναληπτική κατασκευή του $\mathcal{D}_n$ σε λεξικογραφική σειρά.

Η λεξικογραφική κατασκευή του $\mathcal{D}_n$ μπορεί να γίνει, χωρίς τη χρήση των στοιχείων του $\mathcal{E}_n$ ως εξής: Αν $\alpha \in \mathcal{D}_n$, με $\alpha = \alpha_1\alpha_2\cdots\alpha_{2n}$, τότε αρχικά εντοπίζουμε τη δεξιότερη μεγιστική ακολουθία από 1 στην α, της οποίας προηγείται 0 (αν δεν υπάρχει, τότε $\alpha = 1^n0^n$). Έστω ότι το αριστερότερο και το δεξιότερο 1 σε αυτή την ακολουθία είναι τα α_i και α_j αντίστοιχα. Θα είναι λοιπόν $\alpha = p01^{j-i+1}0^{2n-j}$, όπου $|p| = i - 2$ και $i > 2$. Η επόμενη της α θα έχει πρόθεμα $p1q$, όπου η q θα έχει μήκος $2n - i + 1$ και $j - i$ μονάδες στις δεξιότερες δυνατές θέσεις, δηλαδή θα τελειώνει με $(10)^{j-i}$. Συνεπώς, η q θα ξεκινά με $x = 2n - i + 1 - 2(j - i) = 2n - 2j + i + 1$ μηδενικά. Έτσι, θα είναι $\text{next}(\alpha) = p10^x(10)^{j-i}$. Τέλος, εφαρμόζουμε τα παραπάνω, ξεκινώντας με την ελάχιστη λέξη του $\mathcal{D}_n$, δηλαδή την $(10)^n$ και υπολογίζοντας επαναληπτικά την επόμενη κάθε φορά λέξη μέχρις ότου φτάσουμε στη μέγιστη λέξη 1^n0^n.

3.3.2 Αναδρομική κατασκευή

Για κάθε $\alpha \in \mathcal{D}_n$, $n \geq 2$, αν από την ακολουθία $e(\alpha) = e_1e_2\cdots e_n$ διαγράψουμε τον τελευταίο όρο, τότε προκύπτει η $e_1e_2\cdots e_{n-1}$ η οποία κωδικοποιεί μια λέξη του $\mathcal{D}_{n-1}$, αφού είναι γνησίως αύξουσα και ικανοποιεί την ιδιότητα $i \leq e_i \leq 2i-1$, για κάθε $i \in [n-1]$. Προφανώς, η διαγραφή αυτή αντιστοιχεί σε διαγραφή του δεξιότερου 10 στην α.

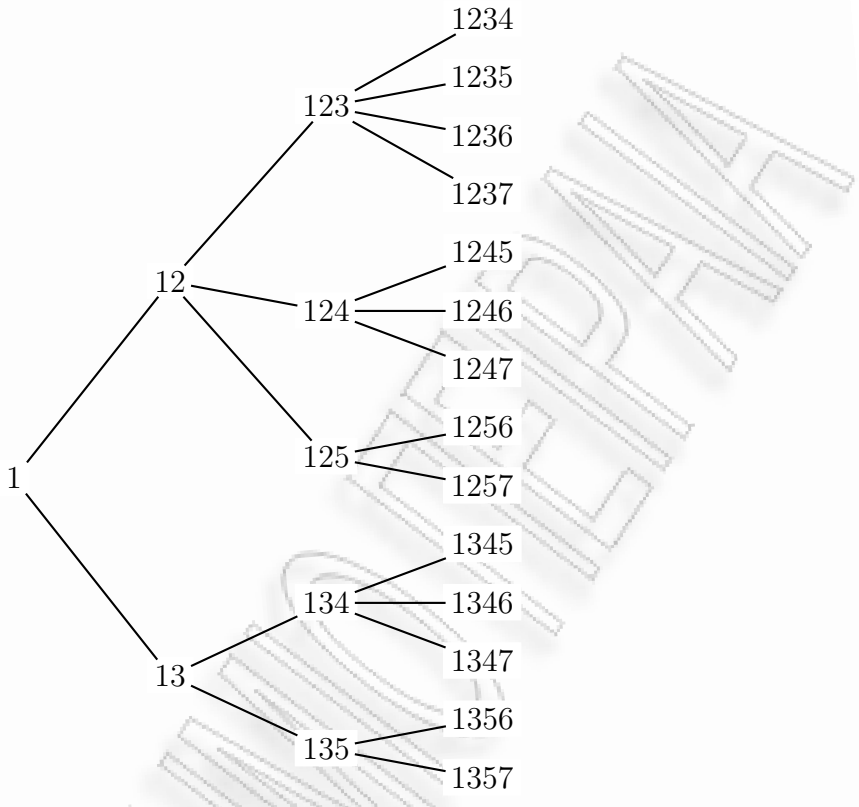
Αντίστροφα, αν προσθέσουμε στην $e(\alpha)$ έναν επιπλέον όρο e_{n+1} τέτοιον ώστε $e_n < e_{n+1} \leq 2(n+1)-1$, τότε προκύπτει μία ακολουθία που κωδικοποιεί μια λέξη του $\mathcal{D}_{n+1}$. Έτσι, από την α μπορούν να προκύψουν συνολικά $2n+1-e_n$ σε πλήθος λέξεις του $\mathcal{D}_{n+1}$. Το πλήθος αυτό εξαρτάται από την τιμή του e_n, δηλαδή από το μήκος της δεξιότερης ακολουθίας από 0 στην α. Με τον τρόπο αυτό, μπορούμε να κατασκευάσουμε όλα τα στοιχεία του $\mathcal{D}_{n+1}$, αρκεί να γνωρίζουμε όλα τα στοιχεία του $\mathcal{D}_n$. Επιπλέον, είναι προφανές ότι κάθε στοιχείο του $\mathcal{D}_{n+1}$ προκύπτει από μοναδικό στοιχείο του του $\mathcal{D}_n$.

Από τα παραπάνω, μπορεί να προκύψει ένας αλγόριθμος αναδρομικής κατασκευής του $\mathcal{D}_n$, ξεκινώντας από το $\mathcal{D}_1$, δηλαδή την ακολουθία $e(10) = 1$, και προσθέτοντας αναδρομικά κάθε φορά ένα επιπλέον στοιχείο στην εκάστοτε ακολουθία, μέχρις ότου το μήκος της να γίνει ίσο με n. Επιπλέον, αν προσθέτουμε τις δυνατές τιμές του νέου όρου σε φθίνουσα σειρά, τότε τα στοιχεία του $\mathcal{D}_n$ θα κατασκευαστούν σε λεξικογραφική σειρά.

```
DR(n, e = e_1 e_2 ··· e_k, k)  begin
    for i ← 2k + 1 downto e_k + 1 do
        if k < n − 1 then
            DR (n, e_1 e_2 ··· e_k i, k + 1);
        else
            output (e_1 e_2 ··· e_k i);
            exit;
        endif
    endfor
end
```

Αλγόριθμος 3.3.2: Η αναδρομική κατασκευή του $\mathcal{E}_n$ επιτυγχάνεται με την κλήση $\mathrm{DR}(n, 1, 1)$.

Παράδειγμα. Ο αλγόριθμος 3.3.2 κατασκευάζει τα στοιχεία των $\mathcal{E}_n$, για $n \in [4]$, με τη σειρά που τα συναντάμε διασχίζοντας το παρακάτω δένδρο σε προδιάταξη (όπου το πρώτο παιδί κάθε κόμβου τοποθετείται χαμηλότερα).



3.3.3 Αλγόριθμοι ranking - unranking

Αν $\alpha \in \mathcal{D}_n$, $n \in \mathbb{N}^*$, με $\alpha = p1q$, για κάποιες λέξεις $p, q \in \{0,1\}^*$, τότε η α είναι μεγαλύτερη (λεξικογραφικά) από κάθε $\beta \in \mathcal{D}_n$, με $\beta = p0s$, $s \in \{0,1\}^*$. Έτσι, το πλήθος των δυνατών επιθεμάτων s, με $p0s \in \mathcal{D}_n$ ισούται με το πλήθος των λέξεων του $\mathcal{D}_n$ που έχουν πρόθεμα p και είναι μικρότερες της α. Δεδομένου ότι το rank της α είναι ίσο με το πλήθος των λέξεων που είναι μικρότερες ή ίσες της α, θα ισχύει

$$\mathrm{rank}(\alpha) = 1 + \sum_{\substack{p \in \{0,1\}^* \\ \alpha = p1q}} |\{\beta \in \mathcal{D}_n : \beta \text{ έχει πρόθεμα } p0\}|.$$

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 $p0s \in \mathcal{D}_n$, για κάθε πρόθεμα $p1$ του α.

Ισχύει ότι $|s| = 2n - |p| - 1$ και $|p|_1 + |s|_1 - |p|_0 - |s|_0 = 1$, δηλαδή

$$|s|_1 = n - |p|_1, \quad |s|_0 = n - |p|_0 - 1.$$

Συμβολίζουμε με $s(x, y)$ το πλήθος των δυνατών επιθεμάτων s που μπορούν να εμφανιστούν σε οποιαδήποτε λέξη Dyck με $|s|_1 = x$ και $|s|_0 = y$. Κάθε τέτοιο επίθεμα ξεκινά με 0 ή με

1, και ακολουθείται από ένα μικρότερο σε μήκος επίθεμα που είναι επίσης επίθεμα λέξεων Dyck, οπότε προκύπτει η αναδρομική σχέση

$$s(x, y) = s(x - 1, y) + s(x, y - 1),$$

με $s(x, y) = 0$, όταν $y < x$ ή $y \leq 0$ ή $x < 0$, και $s(0, y) = 1$.

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

$y \setminus x$	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 $s(x, x) = C_x$ (αφού ένα επίθεμα λέξης Dyck με ίσο πλήθος 1 και 0 είναι λέξη Dyck).

Έτσι, αν $\alpha = p1q$, τότε το ζητούμενο πλήθος των λέξεων s, με $p0s \in \mathcal{D}_n$ είναι ίσο με $s(n - |p|_1, n - |p|_0 - 1)$, οπότε

$$\operatorname{rank}(\alpha) = 1 + \sum_{\substack{p \in \{0,1\}^* \\ \alpha = p1q \in \mathcal{D}_n}} s(n - |p|_1, n - |p|_0 - 1). \tag{3.2}$$

Από τα παραπάνω, προκύπτει ο επόμενος αλγόριθμος ranking.

Υποθέτουμε ότι οι τιμές $s(x, y)$ έχουν προϋπολογιστεί με τη βοήθεια της αναδρομικής σχέσης που ικανοποιούν, και έχουν αποθηκευτεί σε πίνακα διαστάσεων $n \times n$, ώστε να είναι προσβάσιμες από τον αλγόριθμο.

Έστω τώρα ότι έχουμε την τιμή $\operatorname{rank}(\alpha)$ και ζητάμε τη λέξη $\alpha \in \mathcal{D}_n$. Υποθέτουμε ότι ο n είναι γνωστός, αλλιώς μπορούμε να τον ορίσουμε ως τον ελάχιστο φυσικό για τον οποίο ισχύει $\operatorname{rank}(\alpha) \leq C_n$.

Υποθέτουμε ότι ο αλγόριθμος unranking, στο i-οστό βήμα, έχει εξετάσει τα πρώτα $i-1$ γράμματα της λέξης $\alpha \in \mathcal{D}_n$, $i \in [2n-1]$, και έχει υπολογίσει ότι σε αυτό το αρχικό τμήμα p_i υπάρχουν ακριβώς k μονάδες.

Είσοδος: Η λέξη Dyck $\alpha = \alpha_1\alpha_2\cdots\alpha_n$ και ο φυσικός n.
Έξοδος: Η τιμή $r = \text{rank}(\alpha)$.

$r \leftarrow 1$;
ones $\leftarrow 0$;
for $i \leftarrow 1$ **to** $2n-1$ **do**
  if $\alpha_i = 1$ **then**
    ones $\leftarrow$ ones $+ 1$;
    $r \leftarrow r + s(n - \text{ones}, n + \text{ones} - i + 1)$;
  endif
endfor
return r;

Αλγόριθμος 3.3.3: Αλγόριθμος ranking για λέξεις Dyck.

Έστω, $\alpha = p_i\alpha_i q$, όπου το α_i είναι το i-οστό γράμμα της α, και έστω ότι είναι γνωστό το πλήθος r_i των λέξεων του $\mathcal{D}_n$ που έχουν πρόθεμα p και είναι μικρότερες ή ίσες της α. Αν $\alpha_i = 0$, τότε για το επίθεμα q έχουμε ότι $|q| = 2n - i$, $|q|_1 = n - k$, οπότε και $|q|_0 = 2n - i - (n - k) = n + k - i$. Έτσι, το πλήθος των λέξεων του $\mathcal{D}_n$ που αρχίζουν με p_i0 είναι ίσο με $s(n-k, n+k-i)$. Οπότε $\alpha_i = 1$ αν και μόνο αν $r > s(n-k, n+k-i)$. Επιπλέον, στην περίπτωση αυτή, για την εξέταση του $(i+1)$-οστού γράμματος, το νέο πρόθεμα που θα έχει εξετασθεί θα είναι το $p_{i+1} = p_i1$ και το πλήθος r_{i+1} θα είναι ίσο με $r_i - s(n-k, n+k-i)$.

Επαναλαμβάνοντας την παραπάνω διαδικασία για κάθε $i \in [2n-1]$, και δεδομένου ότι το πλήθος r_1 του αρχικού βήματος είναι γνωστό και ίσο με $\text{rank}(\alpha)$ (αφού $p_0 = \varepsilon$), τελικά ο αλγόριθμος θα υπολογίσει σωστά τα πρώτα $2n-1$ γράμματα της α. Το τελευταίο γράμμα είναι πάντα 0.

Είσοδος: Η τιμή $r = \text{rank}(\alpha)$ και ο φυσικός n.
Έξοδος: Η λέξη Dyck $\alpha = \alpha_1\alpha_2\cdots\alpha_n$.

ones $\leftarrow 0$;
for $i \leftarrow 1$ **to** $2n-1$ **do**
  if $r > s(n - \text{ones}, n + \text{ones} - i)$ **then**
    $\alpha_i = 1$;
    $r \leftarrow r - s(n - \text{ones}, n + \text{ones} - i)$;
    ones $\leftarrow$ ones $+ 1$;
  else
    $\alpha_i = 0$;
  endif
endfor
$\alpha_{2n} = 0$;
return α;

Αλγόριθμος 3.3.4: Αλγόριθμος unranking για λέξεις Dyck.

3.3.4 Κώδικας Gray

Για τις λέξεις Dyck δεν υπάρχει κώδικας Gray, αφού το πλήθος των 1 σε μια λέξη Dyck είναι ίσο με το πλήθος των 0, οπότε αν υπήρχαν δύο λέξεις ίδιου μήκους με απόσταση Hamming ίση με 1, τότε δεν θα είχαν ίδιο πλήθος 0 και 1. Στη συνέχεια, θα διατυπωθεί ένας αναδρομικός αλγόριθμος κατασκευής του $\mathcal{D}_n$ σε 2-κώδικα Gray. Για το σκοπό αυτό, θα χρησιμοποιήσουμε και πάλι την κωδικοποίηση των λέξεων Dyck από την ακολουθία των θέσεων των μονάδων τους, δηλαδή το σύνολο $\mathcal{E}_n$. Συγκεκριμένα, θα ορίσουμε έναν κώδικα Gray G_n για τα στοιχεία του $\mathcal{E}_n$, ο οποίος αντιστοιχεί σε 2-κώδικα Gray για τα στοιχεία του $\mathcal{D}_n$.

Προφανώς, είναι $G_1 = (1)$. Έστω ότι υπάρχει ο G_{n-1}, για $n \geq 2$. Για να κατασκευάσουμε το G_n, ξεκινάμε με το πρώτο στοιχείο του G_{n-1}, έστω την ακολουθία $e = e_1 e_2 \cdots e_{n-1}$. Προσθέτοντας έναν επιπλέον όρο e_n, με $e_{n-1} < e_n < 2n$, προκύπτει μια ακολουθία του $\mathcal{E}_n$. Αν αυτή η πρόσθεση επαναληφθεί για κάθε δυνατή τιμή του e_n, έστω σε αύξουσα σειρά (δηλαδή για κάθε τιμή από $e_{n-1}+1$ έως και $2n-1$), τότε οι ακολουθίες που προκύπτουν είναι σε διάταξη κώδικα Gray. Η τελευταία ακολουθία που προέκυψε είναι η $e(2n-1)$. Επιλέγουμε το επόμενο στοιχείο e' του G_{n-1} και δημιουργούμε την $e'(2n-1)$, η οποία έχει απόσταση Hamming 1 από την $e(2n-1)$. Στη συνέχεια, επαναλαμβάνουμε τα ίδια για την e', καταλήγοντας στην $e'(2n-2)$ (επειδή η $e'(2n-1)$ έχει ήδη κατασκευαστεί). Αυτό είναι πάντα δυνατό, αφού ο τελευταίος όρος μιας ακολουθίας του $\mathcal{E}_{n-1}$ είναι πάντα μικρότερος ή ίσος του $2(n-1)-1 = 2n-3 < 2n-2$. Στη συνέχεια, επιλέγουμε το τρίτο στοιχείο e'' του G_{n-1}, κατασκευάζουμε την $e''(2n-2)$ και επαναλαμβάνουμε τα ίδια, παραλείποντας αυτή τη φορά την πρόσθεση του όρου $2n-2$. Η διαδικασία αυτή συνεχίζεται για όλα τα στοιχεία του G_{n-1}, οπότε τελικά προκύπτει ο G_n.

```
DGray(n) begin
    if n < 2 then return (1);
    G_{n-1} ← DGray(n − 1);
    skip ← n;
    foreach e ∈ G_{n-1} do
        output (eskip);
        for i ← e_{n-1} + 1 to skip − 1 do output (ei);
        for i ← skip + 1 to 2n − 1 do output (ei);
        if skip = 2n − 1 then skip ← skip − 1;
        else skip ← 2n − 1;
    endfch
end
```

Αλγόριθμος 3.3.5: Η αναδρομική κατασκευή του $\mathcal{E}_n$ σε κώδικα Gray επιτυγχάνεται με την κλήση DGray(n).

Αν $e = e_1 e_2 \cdots e_n$ και αλλάξει η τιμή του στοιχείου e_i, $2 \leq i \leq n-1$, η αλλαγή αυτή ισοδυναμεί σε μετατροπή της μονάδας αυτής σε μηδέν και κάποιου από τα μηδενικά σε 1, αφού μεταξύ της $(i-1)$-οστής και $(i+1)$-οστής μονάδας παρεμβάλλονται $e_{i+1} - e_{i-1} - 2$ μηδενικά και μία μονάδα. Έτσι, η απόσταση Hamming των αντίστοιχων λέξεων Dyck θα είναι ίση με 2.

Παράδειγμα. Στον επόμενο πίνακα δίνεται ο κώδικας Gray των λέξεων Dyck για $n = 5$.

$\mathcal{E}_n$	$\mathcal{D}_n$	rank	$\mathcal{E}_n$	$\mathcal{D}_n$	rank
12345	1111100000	42	12467	1101011000	24
12346	1111010000	41	12469	1101010010	22
12347	1111001000	40	12569	1100110010	17
12348	1111000100	39	12567	1100111000	19
12349	1111000010	38	12568	1100110100	18
12359	1110100010	34	12578	1100101100	16
12356	1110110000	37	12579	1100101010	15
12357	1110101000	36	13579	1010101010	1
12358	1110100100	35	13578	1010101100	2
12368	1110010100	32	13568	1010110100	4
12367	1110011000	33	13567	1010111000	5
12369	1110010010	31	13569	1010110010	3
12379	1110001010	29	13469	1011010010	8
12378	1110001100	30	13467	1011011000	10
12478	1101001100	21	13468	1011010100	9
12479	1101001010	20	13458	1011100100	12
12459	1101100010	25	13456	1011110000	14
12456	1101110000	28	13457	1011101000	13
12457	1101101000	27	13459	1011100010	11
12458	1101100100	26	13479	1011001010	6
12468	1101010100	23	13478	1011001100	7

Βιβλιογραφία

[1] F. Bassino, J. Clément, J. Fayolle and P. Nicodème, Counting occurrences for a finite set of words: an inclusion-exclusion approach, *DMTCS proc.* **AH** (2007), 29–44.

[2] E. A. Bender and S. Gill Williamson, *Foundations of Combinatorics with Applications*, (2006), Dover Publications.

[3] J. M. Borwein and R. M. Corless, An Encyclopedia of Integer Sequences, *SIAM Review* **38** (1996), 333–337.

[4] E. Deutsch, Dyck path enumeration, *Discrete Math.* **204** (1999), 167–202.

[5] H. Gould, *Combinatorial Identities*, Morgantown, W. Va. 1972.

[6] L. J. Guibas and A. M. Odlyzko, Periods in Strings, *J. Combin. Theory Ser. A* **30** (1981), 19–42.

[7] L. J. Guibas and A. M. Odlyzko, String Overlaps, Pattern Matching, and Nontransitive Games, *J. Combin. Theory Ser. A* **30** (1981), 183–208.

[8] R. Guy, The Encyclopedia of Integer Sequences *Amer. Math. Monthly* **104**(2), 180–184.

[9] T. Harju and D. Nowotka, Counting bordered and primitive words with a fixed weight, *Theoret. Comput. Sci.* **340** (2005), 273–279.

[10] D. M. Jackson and I. P. Goulden, An Inversion Theorem for Cluster Decompositions of Sequences with Distinguished Subsequences, *Stud. Appl. Math.* **61** (1979), 141–178.

[11] E. Munarini and N. Z. Salvi, Binary strings without zigzags, *Sém. Lothar. Combin.* **49** (2004), Article B49h.

[12] J. Noonan and D. Zeilberger, The Goulden-Jackson Cluster Method: Extensions, Applications and Implementations, *J. Differ. Equations Appl.* **5** (1999), 355–377.

[13] N. J. A. Sloane, *The On-Line Encyclopedia of Integer Sequences* (2010), published electronically at `http://www.research.att.com/~njas/sequences/`.

[14] R. P. Stanley, *Enumerative Combinatorics*, Vol. 2 (1999), Cambridge University Press.

[15] R. P. Stanley, *Catalan Addendum*, available at `http://www-math.mit.edu/~rstan/ec/catadd.pdf`.

[16] Ι. Τασούλας, *Πρότυπα σε μονοπάτια Dyck και διατεταγμένα δένδρα*, Διδακτορική διατριβή (2009), Πανεπιστήμιο Πειραιώς.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

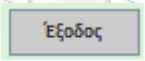
Παράρτημα - Λογισμικό της εργασίας

Το πρόγραμμα που υλοποιήθηκε έχει ως στόχο την κατασκευή συνόλων δυαδικών λέξεων, για λέξεις Fibonacci, λέξεις χωρίς zig-zag και λέξεις Dyck. Ο σχεδιασμός του έχει γίνει με τέτοιο τρόπο ώστε ακόμη και ο πλέον άπειρος χρήστης να μην αντιμετωπίσει προβλήματα με την λειτουργία και την χρήση του προγράμματος.

Για τη δημιουργία της εφαρμογής χρησιμοποιήθηκε το πρόγραμμα NetBeans IDE 6.9.1.

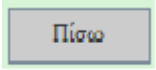
Πριν προχωρήσουμε στην παρουσίαση της εφαρμογής, σκόπιμο είναι να εξηγήσουμε την χρήση ορισμένων πλήκτρων που υπάρχουν σε όλες τις φόρμες της εφαρμογής.

Ξεκινάμε λοιπόν με το πλήκτρο τερματισμού της εφαρμογής που φαίνεται στο παρακάτω σχήμα.

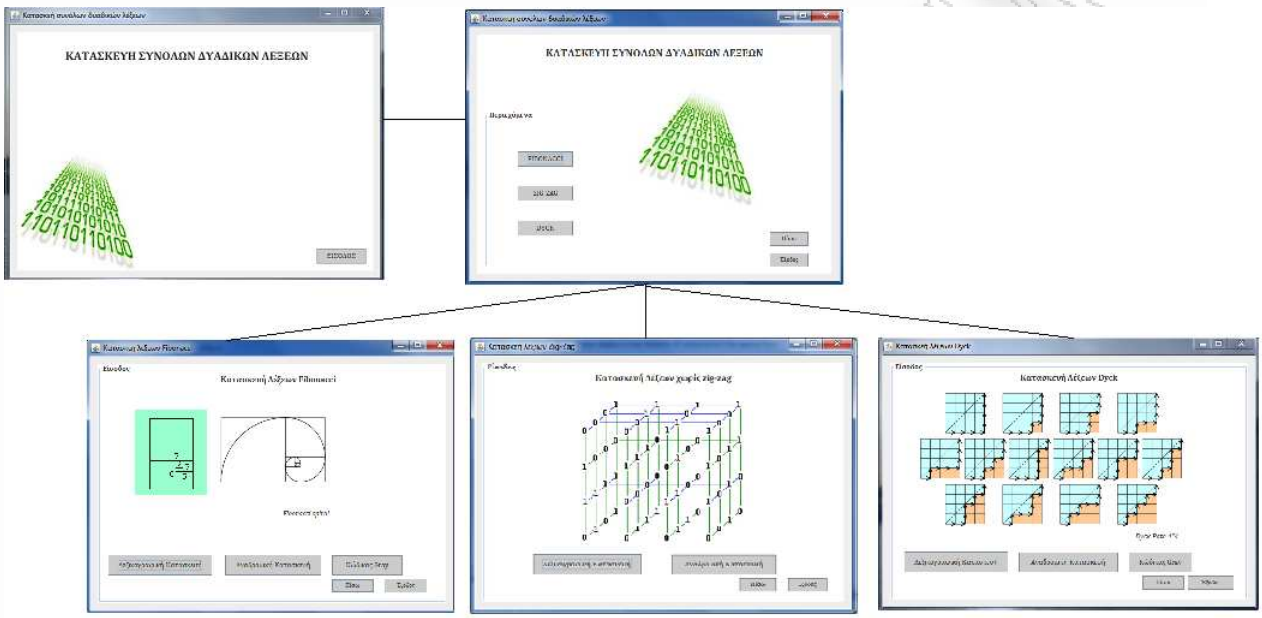


Σε οποιαδήποτε φόρμα και αν βρίσκεται ο χρήστης, μόλις κάνει κλικ σε αυτό το πλήκτρο, τερματίζεται η εφαρμογή.

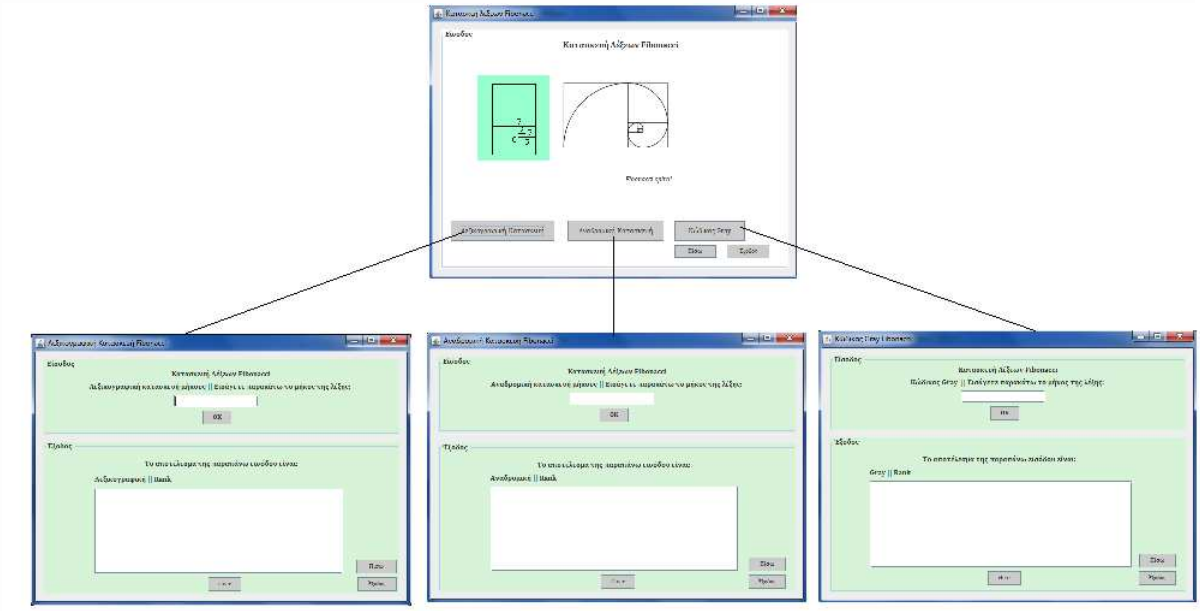
Σε κάθε φόρμα κάτω δεξιά υπάρχει το πλήκτρο «Πίσω» το οποίο δίνει την δυνατότητα στον χρήστη να επιστρέψει στην προηγούμενη φόρμα από αυτήν που βρίσκεται.



Παρουσίαση εφαρμογής

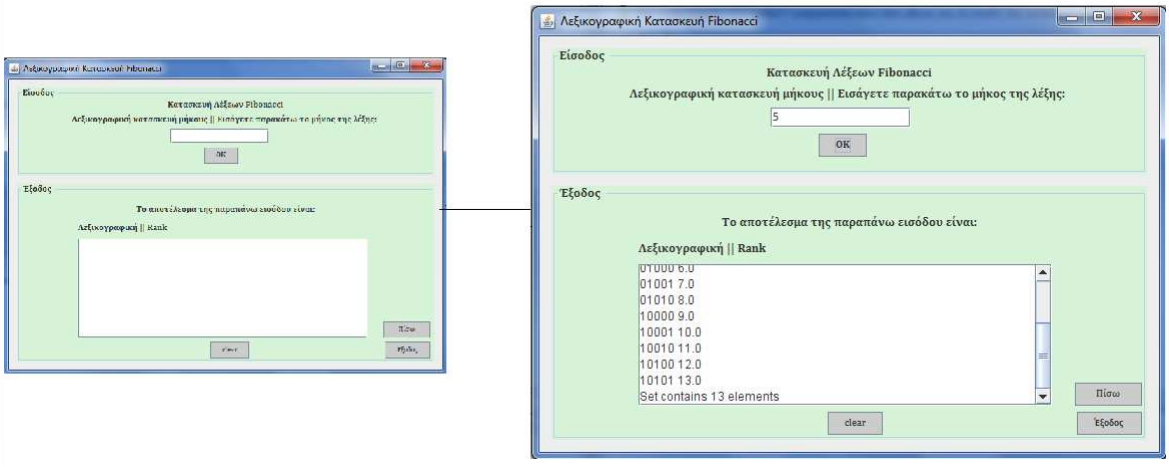


Ξεκινάμε λοιπόν με την αρχική φόρμα που εμφανίζεται στην οθόνη του υπολογιστή με την εκκίνηση της εφαρμογής. Σε αυτήν την φόρμα υπάρχει το πλήκτρο εισόδου. Εάν ο χρήστης κάνει κλικ στο πλήκτρο αυτό τότε εμφανίζεται η δεύτερη φόρμα, που περιέχει τα περιεχόμενα της εφαρμογής. Σε αυτήν την φόρμα εκτός από τα βασικά πλήκτρα που αναλύσαμε πριν, υπάρχουν όπως βλέπουμε τρία πλήκτρα. Κάνοντας κλικ σε οποιοδήποτε από αυτά, ο χρήστης οδηγείται στην φόρμα που αντιστοιχεί στο κάθε πλήκτρο, της οποίας το περιεχόμενο περιγράφεται από τον τίτλο του πλήκτρου αυτού. Πιο συγκεκριμένα, εάν ο χρήστης ενδιαφέρεται για τις λέξεις Fibonacci, κάνοντας κλικ στο κουμπί Fibonacci, θα οδηγηθεί στην κεντρική φόρμα «Κατασκευή των λέξεων Fibonacci» (το ίδιο ισχύει και για τις λέξεις χωρίς zig-zag και Dyck, όπως φαίνεται και στην παραπάνω εικόνα) όπως φαίνεται στην παρακάτω εικόνα.



Η κεντρική φόρμα των λέξεων Fibonacci περιέχει εκτός από τα βασικά πλήκτρα και τρία ακόμη τα οποία αναφέρονται στην λεξικογραφική και αναδρομική κατασκευή, καθώς επίσης και στον κώδικα Gray. Ανάλογα με το κουμπί που θα επιλέξει ο χρήστης, θα οδηγηθεί και στην αντίστοιχη φόρμα όπως φαίνεται και στην παραπάνω εικόνα. Σε κάθε μία από αυτές τις φόρμες, ο χρήστης μπορεί να εισάγει έναν αριθμό (δηλαδή το μήκος της δυαδικής λέξης που επιθυμεί) και κάνοντας κλίκ στο κουμπί «OK» θα του εμφανιστεί το συγκεκριμένο αποτέλεσμα.

Ένα παράδειγμα λεξικογραφικής κατασκευής των λέξεων Fibonacci με μήκος λέξης= 5 φαίνεται στην παρακάτω εικόνα.



Επειδή όλες οι φόρμες της εφαρμογής έχουν σχεδιαστεί με τον ίδιο τρόπο, έτσι ώστε να μην δίνεται στον χρήστη η αίσθηση της ασυνέχειας, και επιπλέον επειδή οι λειτουργίες

που παρέχονται από όλες τις φόρμες είναι οι ίδιες με τις λειτουργίες που περιγράψαμε για την περίπτωση της φόρμας των λέξεων Fibonacci, δεν θα επεκταθούμε στην επεξήγηση του τρόπου λειτουργίας των υπολοίπων φορμών της εφαρμογής.

Ακολουθεί κώδικας σε Java

Κατασκευή των λέξεων Fibonacci σε Java.

```
1  package binary_words;
2
3  import java.util.*;
4  import javax.swing.JTextArea;
5
6  // A Fibonacci word is a binary word avoiding the pattern "11"
7  public class FibonacciCode {
8
9      private ArrayList<String> wordsList ; //contains all Fibonacci
           wordsList of length n
10     private JTextArea Text;
11
12     public FibonacciCode(JTextArea Text){//constructor
13         wordsList = new  ArrayList<String>();
14         this.Text = Text;
15     }
16
17     public void populateLex(int n){//
18          wordsList.clear();
19          lexGen(n);
20     }
21
22     public void populateRec(int n){//
23          wordsList.clear();
24          recGen("",n);
25     }
26
27     public void populateGray(int n){//
28          wordsList.clear();
29          wordsList = gray(n);
30     }
31
32     public void populateGray2(int n){//
33          wordsList.clear();
34          gray2("",n);
35     }
36
37
38
```

```
39      //print contents of wordsList
40      public void print()
41      {
42          String s = "";
43          for(int i=0; i< wordsList.size();i++)
44          {
45              Text.append(wordsList.get(i));
46              Text.append("\n");
47          }
48          Text.append("Set contains "+wordsList.size()+" elements");
49      }
50
51
52
53      //print contents of wordsList
54      public void print2()
55      {
56          String s = "";
57          for(int i=0; i< wordsList.size();i++)
58          {
59              Text.append(wordsList.get(i));
60              Text.append(" "+ rank(wordsList.get(i)));
61              //Text.append(" " + unrank(rank(wordsList.get(i))));
62              Text.append("\n");
63          }
64          Text.append("Set contains "+wordsList.size()+" elements");
65      }
66
67
68
69      //Recursive generation of all Fibonacci wordsList of length n
70      //wordsList are generated lexicographically
71      private void recGen(String p, int n){//p: prefix
72          if (n==1){
73              wordsList.add(p+"0");
74              wordsList.add(p+"1");
75          }
76          else if (n==2){
77              wordsList.add(p+"00");
78              wordsList.add(p+"01");
79              wordsList.add(p+"10");
80          }
81          else if (n>2){
82              recGen(p+"0",n-1);
83              recGen(p+"10",n-2);
84          }
85      }
86
87
```

```
88
89      private String next(String s){//returns "" if s is maximum
90          int i;
91          String first = wordsList.get(0);//first=0^n, already in list
92                                          //before the first call of next()
93          //find the rightmost occurrence of 00
94          for(i = s.length()-2; i>=0; i--)
95              if(s.charAt(i)=='0' && s.charAt(i+1)=='0') break;
96          if(i<0){
97              if(s.charAt(0)=='1') return "";//not found, s has no
                    next
98              else return "1" + first.substring(1);
99          }
100         else
101             return s.substring(0, i+1) + "1" + first.substring(i+2);
102     }
103
104     public void lexGen(int n){
105         String s = new String();
106         for(int i = 0; i < n; i++) s +="0"; //first element is 0^n
107         for( ; !(s.isEmpty()); s= next(s)) wordsList.add(s);
108     }
109
110     //s must be a valid Fibonacci word
111     //returns the rank of s, i.e., its position in ascending
            lexicographic ordering
112     public double rank(String s){
113         double rank = 1;
114         int n = s.length();
115         double[] F = new double[n+1];
116         F[0]=1; F[1]=1;//calculate Fibonacci numbers from F_1 to F_n
117         for(int i=2;i<n+1;i++) F[i]=F[i-1]+F[i-2];
118
119         for(int i=n-1;i>=0;i--){
120             if(s.charAt(i)=='1') rank += F[n-i];
121         }
122         return rank;
123     }
124
125     //constructs the Fibonacci word s having rank r
126     //all leading zeros are deleted
127     public String unrank(double r){
128        String s="";
129        int n,i=1;
130        double f=1;
131        double ff=1;
132       if(r<=1)return "";
133       while(r>ff){//find the smallest Fibonacci number >= r
134           ff+=f;
```

```
135             f=ff-f;
136              i++;
137         }
138         n=i-1;
139         double[] F = new double[n+1];
140          F[0]=1; F[1]=1;//calculate Fibonacci numbers from F_1 to F_n
141          for( i=2;i<n+1;i++) F[i]=F[i-1]+F[i-2];
142
143          for(i=0;i<n;i++){//unrank
144              if(r>F[n-i]) {
145                   s=s+'1';
146                   r-=F[n-i];}
147                   else {
148                   s=s+'0';
149
150              }
151          }
152          return s;
153     }
154
155     public ArrayList<String> gray(int n){
156          ArrayList<String> G = new ArrayList<String>();
157          ArrayList<String> rG = new ArrayList<String>();
158          if(n==1) {G.add("0"); G.add("1"); return G;}
159          if(n==2) {G.add("01"); G.add("00"); G.add("10"); return G;}
160
161          rG = rgray(n-1);
162          for(int i=0; i<rG.size();i++){
163               G.add("0"+rG.get(i));
164          }
165          rG = rgray(n-2);
166          for(int i=0; i<rG.size();i++){G.add("10"+rG.get(i));}
167          return G;
168     }
169
170     public ArrayList<String> rgray(int n){
171          ArrayList<String> G = new ArrayList<String>();
172          ArrayList<String> G2 = new ArrayList<String>();
173          if(n==1) {G.add("1"); G.add("0"); return G;}
174          if(n==2) {G.add("10"); G.add("00"); G.add("01"); return G;}
175
176          G2 = gray(n-2);
177          for(int i=0; i<G2.size();i++){G.add("10"+G2.get(i));}
178          G2 = gray(n-1);
179          for(int i=0; i<G2.size();i++){G.add("0"+G2.get(i));}
180          return G;
181     }
182
183
```

```
184     //G_1 = (0,1) G_2 = (10, 00, 01)
185     private void gray2(String p, int n){
186          if (n==1){
187              wordsList.add(p+"0");
188              wordsList.add(p+"1");
189          }
190          else if (n==2){
191              wordsList.add(p+"01");
192              wordsList.add(p+"00");
193              wordsList.add(p+"10");
194          }
195          else if (n>2){
196              rgray2(p+"0",n-1);
197              rgray2(p+"10",n-2);
198          }
199     }
200
201
202
203     private void rgray2(String p, int n){
204          if (n==1){
205              wordsList.add(p+"1");
206              wordsList.add(p+"0");
207          }
208          else if (n==2){
209              wordsList.add(p+"10");
210              wordsList.add(p+"00");
211              wordsList.add(p+"01");
212          }
213          else if (n>2){
214              gray2(p+"10",n-2);
215              gray2(p+"0",n-1);
216          }
217     }
218 }
```

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ

Κατασκευή των λέξεων χωρίς zig-zag σε Java.

```
package binary_words;
import java.util.*;
import javax.swing.JTextArea;
//Zig-zag free words: binary words avoiding 101 and 010
public class ZigZagFree {
    private ArrayList<String> wordsList;
    private double[] F; //store Fibonacci numbers F_0 F_1 ... F_n
    private JTextArea Text;

    public ZigZagFree(JTextArea Text){
        wordsList = new  ArrayList<String>();
        this.Text = Text;
    }

    public ZigZagFree(int n){
        wordsList = new  ArrayList<String>();
        if(n>0){ //calculate Fibonacci numbers from F_1 to F_n
            F = new double[n+1];
            F[0]=1; F[1]=1;
            for(int i=2;i<n+1;i++) F[i]=F[i-1]+F[i-2];
        }
    }
    //flag encodes the prefix of the word: -1: anything, 0: "00", 1:
        "01", 2: "10", 3: "11"
    private void recGen(String p, int flag, int n){//p: prefix
        if(n==2){
            switch (flag){
              case 0:  wordsList.add(p+"00"); break;
              case 1:  wordsList.add(p+"01"); break;
              case 2:  wordsList.add(p+"10"); break;
              case 3:  wordsList.add(p+"11"); break;
              default: wordsList.add(p+"00"); wordsList.add(p+"01");
                       wordsList.add(p+"10"); wordsList.add(p+"11");
            }
        }
        else if (n>2){
          switch (flag){
            case 0:  recGen(p+"0",0,n-1); recGen(p+"0",1,n-1);break;
            case 1:  recGen(p+"0",3,n-1);break;
            case 2:  recGen(p+"1",0,n-1);break;
            case 3:  recGen(p+"1",2,n-1); recGen(p+"1",3,n-1);break;
            default: recGen(p+"0",0,n-1); recGen(p+"0",1,n-1);
                     recGen(p+"0",3,n-1); recGen(p+"1", 0,n-1);
                     recGen(p+"1",2,n-1); recGen(p+"1",3,n-1);
            }
        }
    }
```

```
47      public void populateRec(int n){//
48          wordsList.clear();
49          recGen("", -1, n);
50      }
51
52      public void populateLex(int n){//
53          wordsList.clear();
54          lexGen(n);
55      }
56
57      public void print(){//print contents of wordsList
58          for(int i=0; i< wordsList.size();i++){
59              Text.append(wordsList.get(i));
60              Text.append("\n");
61          }
62          Text.append("Set contains "+wordsList.size()+" elements");
63      }
64
65      public void print2(){//print contents of wordsList
66          for(int i=0; i< wordsList.size();i++){
67              Text.append(wordsList.get(i));
68              Text.append(" "+ rank(wordsList.get(i)));
69              Text.append("\n");
70          }
71          Text.append("Set contains "+wordsList.size()+" elements");
72      }
73      //s must be a valid ZigZagFree word
74      //returns the rank of s, i.e., its position in ascending
            lexicographic ordering
75      public static double rank(String s){
76          double rank = 1;
77          int n = s.length();
78          if(n<=0) return 1;
79          double[] F = new double[n+1];
80          F[0]=1; F[1]=1;//calculate Fibonacci numbers from F_1 to F_n
81          for(int i=2;i<n+1;i++) F[i]=F[i-1]+F[i-2];
82
83          if(s.charAt(0)=='1') rank += F[n];
84          if(n==1) return rank;
85          //else if n>1
86          for(int i=1;i<n-1;i++){
87              if(s.charAt(i)=='1' && !isPrecededBy01(s,i)) {
88                  if(s.charAt(i-1)=='1') rank += F[n-i-1];
89                  else rank += F[n-i];
90              }
91          }
92          if(s.charAt(n-1)=='1' && !isPrecededBy01(s,n-1)) rank += 1;
93          return rank;
94      }
```

```
95
96
97      //constructs the ZigZagFree word s having rank r and length n
98      public static String unrank(double r, int n){
99          String s="";
100         int i=1;
101
102         double[] F = new double[n+1];
103          F[0]=1; F[1]=1;//calculate Fibonacci numbers from F_1 to F_n
104          for( i=2;i<n+1;i++) F[i]=F[i-1]+F[i-2];
105
106          if(r> F[n]){s+="1"; r-=F[n];}
107          else{s+="0";}
108          for(i=1;i<n;i++){//unrank
109              if(isPrecededBy01(s,i)){s=s+"1";}
110              else if(isPrecededBy10(s,i)){s=s+"0";}
111              else{
112                  if(s.charAt(i-1)=='1'){
113                      if(r>F[n-i-1]){r -=F[n-i-1]; s=s+'1';}
114                      else{s=s+'0';}
115                  }
116                  else if(s.charAt(i-1)=='0'){
117                      if(r>F[n-i]){ r -=F[n-i]; s=s+"1";}
118                      else s = s+"0";
119                  }
120              }
121          }
122          return s;
123      }
124
125
126
127     private static boolean isPrecededBy01(String s, int i){
128         if(i<2 || i > s.length()) return false;
129         if(s.charAt(i-2)=='0' && s.charAt(i-1)=='1') return true;
130         return false;
131     }
132
133
134
135     private static boolean isPrecededBy10(String s, int i){
136         if(i<2 || i > s.length()) return false;
137         if(s.charAt(i-2)=='1' && s.charAt(i-1)=='0') return true;
138         return false;
139     }
140
141
142
143
```

```
144     private String next(String s){
145         String tail= new String();
146         int i = s.length()-1;
147         tail = wordsList.get(0);
148 
149         if (s.charAt(i)=='0' && !isPrecededBy10(s,i))
150             return s.substring(0,i)+"1";
151 
152         for(i = s.length()-2; i>0; i--){
153             if (s.charAt(i)=='0' && !isPrecededBy10(s,i)){
154                 if(s.charAt(i-1)=='1')
155                     return s.substring(0,i)+"1"+ tail.substring(i+1,
                            tail.length());
156                 else
157                     return s.substring(0,i)+"11"+ tail.substring(i
                            +2, tail.length());
158             }
159         }
160         if(s.charAt(0)=='0') return "1" + tail.substring(1, tail.
                length());
161         else return "";
162     }
163 
164     public void lexGen(int n){
165         String s = new String();
166 
167         for(int i = 0; i < n; i++) s +="0"; //first element is 0^n
168         for( ; !(s.isEmpty()); s= next(s)) wordsList.add(s);
169     }
170 
171 }
```

Κατασκευή των λέξεων Dyck σε Java.

```
1  package binary_words;
2  import java.util.*;
3  import javax.swing.JTextArea;
4
5  public class Dyck {
6      private ArrayList<String> wordsList ; //contains all Dyck words
           of length 2n
7      private JTextArea Text;
8
9       public Dyck(JTextArea Text){//constructor
10         wordsList = new  ArrayList<String>();
11         this.Text = Text;
12     }
13
14     public void populateLex(int n){//
15         wordsList.clear();
16         lexGen(n);
17     }
18
19     public void populateRec(int n){//
20         int[] e = {1};
21         wordsList.clear();
22         recGen(1, n, e);
23     }
24
25     public void populateGray(int n){//
26         ArrayList<int[]> G ;
27         String str_e="";
28         int[] e;
29         wordsList.clear();
30         G = Gray(n);
31         for(int i =0 ; i<G.size();i++){
32             str_e="";
33             e = G.get(i);
34             for(int j =0; j<e.length;j++)str_e = str_e + " " + e[j];
35             str_e = e2Dyck(e);
36             wordsList.add(str_e);
37         }
38     }
39
40     public void printArray(int[] a){
41         for(int i = 0; i<a.length;i++)
42            Text.append(a[i]+ " ");
43            Text.append("\n");
44         //System.out.println("");
45      }
46
```

```
47      public void lexGen(int n){
48          int[] e = new int[n];
49          String str_e="";
50          int i;
51
52          for(i =0; i<n;i++) {e[i]= 2*i+1; str_e = str_e + e[i]+" ";}
53           wordsList.add(str_e);
54
55          while(true){
56              i = n-1;
57              while(i>0 && e[i]<e[i-1]+2) i--;
58              if(i>0){
59                  e[i]--;
60                  for(int j=i+1; j<n;j++) e[j] = 2*j+1;
61
62                  str_e="";
63                  for(i=0;i<n;i++) str_e = str_e + e[i]+" ";
64                  wordsList.add(str_e);
65              }
66              else break;
67          }
68      }
69
70      public void recGen(int n, int N, int[] e){
71                  int[] e_new = new int[n+1];
72
73          for(int i=0; i< e.length; i++){
74              e_new[i] = e[i];
75          }
76              e_new[n] = 2*n+1;
77          while(e_new[n] > e_new[n-1]){
78              if(n < N-1){
79                  recGen(n+1,N,e_new);
80              }
81              else{
82                  //print
83                  for(int j=0; j<e_new.length; j++){
84                      Text.append(e_new[j]+ " ");
85                  }
86                  Text.append("");//count++;
87                  Text.append("\n");
88              }
89              e_new[n]--;
90          }
91      }
92
93
94
95
```

```
96      //generates Gray code for n
97      public ArrayList<int[]> Gray(int n){
98          ArrayList<int[]> G = new ArrayList<int[]>();
99          ArrayList<int[]> prevG;
100         int[] o = {1}; int[] e; int[] e_new = new int[n];
101         int skip = 2*n;
102         int new_skip=skip;
103         if(n<2) {G.add(o); return G;}
104
105         prevG = Gray(n-1);
106
107         e = prevG.get(0);
108         for(int j=0 ; j<n-1;j++){e_new[j]=e[j];}//copy array
109         for(int j=e[n-2]+1 ; j<2*n;j++){
110             e_new[n-1]= j;
111             int[] temp = new int[n];
112             for(int k=0;k<n;k++){temp[k]=e_new[k];}
113             G.add(temp);
114         }
115         skip = 2*n-1;
116
117         if(n==2) return G;
118
119         for (int i=1; i<prevG.size(); i++){
120             e = prevG.get(i);
121             for(int j=0 ; j<n-1;j++){e_new[j]=e[j];}//copy array
122             e_new[n-1]=skip;
123             int[] temp = new int[n];
124             for(int k=0;k<n;k++){temp[k]=e_new[k];}
125             G.add(temp);
126
127             for(int j=e[n-2]+1 ; j<skip;j++){//first interval
128                 e_new[n-1]=j;
129                 int[] temp2 = new int[n];
130                 for(int k=0;k<n;k++){temp2[k]=e_new[k];}
131                 G.add(temp2);
132                 new_skip = skip-1;
133             }
134             for(int j=skip+1 ; j<2*n;j++){//second interval
135                 e_new[n-1]=j;
136                 int[] temp2 = new int[n];
137                 for(int k=0;k<n;k++){temp2[k]=e_new[k];}
138                 G.add(temp2);
139                 new_skip = 2*n-1;
140             }
141             skip=new_skip;
142         }
143         return G;
144     }
```

```
145     public static double rank(String s){
146         double r =1;
147         int ones;
148         int x,y, n = s.length()/2;
149         double[][] S = new double[n+1][n+1];
150 
151         for(x=0; x<=n;x++){S[x][0]=0;}
152         for(y=0; y<=n;y++){S[0][y]=1;}
153 
154         for(y=1; y<=n;y++){
155             for(x=1; x<=y;x++){S[x][y] = S[x-1][y]+ S[x][y-1];}
156         }
157 
158         //p: prefix parsed, length of p = i
159         ones=1;
160         for(int i = 1; i<2*n-1;i++){
161             if(s.charAt(i)=='1'){
162                 r = r+ S[n-ones][n+ones-i-1];
163                 ones++;
164             }
165         }
166         return r;
167     }
168 
169     public static String unrank(double r, int n){
170         String s="";
171         int ones, x,y;
172 
173         if(n<1 || r<1) return "";
174         double[][] S = new double[n+1][n+1];
175         for(x=0; x<=n;x++){S[x][0]=0;}
176         for(y=0; y<=n;y++){S[0][y]=1;}
177         for(y=1; y<=n;y++){
178             for(x=1; x<=y;x++){S[x][y] = S[x-1][y]+ S[x][y-1];}
179         }
180         //p: prefix parsed, length of p = i
181         ones=1; s="1";
182         for(int i = 1; i<2*n-1;i++){
183             if(r>S[n-ones][n+ones-i-1]){
184                 s=s+"1";
185                 r=r-S[n-ones][n+ones-i-1];
186                 ones++;
187             }
188             else s=s+"0";
189         }
190         return s+"0";
191     }
192 
193 
```

```
194
195     public void print(){//print contents of wordsList
196
197         for(int i=0; i< wordsList.size();i++){
198             Text.append(wordsList.get(i));
199             Text.append("\n");
200         }
201         Text.append("Set contains "+wordsList.size()+" elements");
202     }
203     public void print2(){//print contents of wordsList
204
205         for(int i=0; i< wordsList.size();i++){
206             Text.append(wordsList.get(i));
207             Text.append(" "+rank(wordsList.get(i)));
208             Text.append("\n");
209         }
210         Text.append("Set contains "+wordsList.size()+" elements");
211     }
212
213      //convert the sequence e of the positions of 1's to a Dyck word
214     private String e2Dyck(int[] e){
215         String v, str_e ="";
216         int i;
217         if(e==null) return "";
218
219         str_e="1"; //Dyck word always starts with 1
220                 for(i=1;i<e.length;i++){ //convert to Dyck word
221                     for(int j=e[i-1]+1;j<e[i];j++)
222                         str_e = str_e + "0";//put zeros between ones
223                     str_e = str_e + "1";
224                 }
225         for(int j=e[i-1]+1;j<2*e.length+1;j++) str_e = str_e + "0";
               //append trailing 0's
226         return str_e;
227
228     }
229
230 }
```