Πανεπιστημίο Κρήτης
Τμήμα Μαθηματικών

Πιστοποίηση πρώτων αριθμών με μηχανές Turing

Μεταπτυχιακή εργασία

Δημητρίου Δ. Βάρσου

Ηράκλειο, Μάιος 2008

Πανεπιστημίο Κρήτης
Τμήμα Μαθηματικών

Πιστοποίηση πρώτων αριθμών με μηχανές Turing

Μεταπτυχιακή εργασία

Δημητρίου Δ. Βάρσου

Ηράκλειο, Μάιος 2008

Univercity of Crete
Department of Mathematics

Primality testing using Turing machines

Dimitrios D. Varsos

Master Thesis

Iraklio, May 2008

Η παρούσα εργασία έγινε στο Πανεπιστήμιο της Κρήτης την περίοδο 2007-2008, στα πλαίσια του Διατμηματικού Μεταπτυχιακού Προγράμματος "Τα Μαθηματικά και Εφαρμογές τους" στην κατεύθυνση "Θεωρητικά Μαθηματικά". Η τριμελής συμβουλευτική επιτροπή αποτελείται από τους:

Αντωνιάδη Ιωάννη

Σκανδάλη Κωνσταντίνο (επιβλέπων)

Φειδά Αθανάσιο.

This project has been made at the Univercity of Crete during the season 2007-2008 for the program "Pure Mathematics", part of the Postgraduate Program "Mathematics and its Applications". The evaluation committee is consisted of:

Antoniadis Ioannis

Pheidas Athanasios

Skandalis Konstantinos (supervisor).

Στον παππού μου, Δημήτρη.

Πρόλογος

Η εργασία αυτή σηματοδοτεί την λήξη ενός κύκλου σπουδών επτά ετών στο Πανεπιστήμιο Κρήτης, αρχικά ως προπτυχιακός και έπειτα ως μεταπτυχιακός φοιτητής, και δεν θα μπορούσε να ολοκληρωυεί χωρίς την καθοδήγηση από τον επιβλέποντα καθηγητή μου κ. Σκανδάλη, αλλά και τις χρήσιμες παρατηρήσεις από την συμβουλευτική επιτροπή, τον κ. Αντωνιάδη και τον κ. Φειδά.

Με την εργασία αυτή όμως κλείνει και μια περίοδος της ζωής μου διάρκειας τριών σχεδόν χρόνων, που σημαδεύτηκε από μια σειρά δυσκολίες που είχα να αντιμετωπίσω. Είμαι ευγνώμων που σε όλη αυτή τη προσπάθεια, σε κάθε μου βήμα, βρίσκονταν γύρω μου άνθρωποι που με στήριξαν. Ξεκινόντας από τους φίλους μου, τη Μαρία, τη Μαρία, τον Βασίλη και τη Μαργαρίτα που με ζήσαν από κοντά και η υπομονή τους στις παραξενιές μου με έχει συγκινήσει. Τον καθηγητή μου, τον κ. Σκανδάλη που -εκτός φυσικά από την επιστημονική βοήθεια- κατάφερε να με κατανοήσει και να με στηρίξει ηθικά. Την Αρετή και τον Βαγγέλη, που σε μια κρίσιμη στιγμή στάθηκαν δίπλα μου. Χωρις αυτούς ίσως αυτή η εργασία να μην είχε πραγματοποιηθεί. Τον Μάριο που κάθε στιγμή ήταν πρόθυμος να με ακούσει και να με βοηθήσει. Πάνω απ΄ όλα όμως γιατί με έμαθε τι σημαίνει επιστημονική σκέψη. Τον κ. Τζανάκη που μια στίγμη έντασης στάθηκε αφορμή να ξεδιπλώσει μια σπάνια ανθρωπιά. Δεν μπορώ να μην αναφερθώ στην οικογένειά μου που ήταν δίπλα μου σε κάθε μου βήμα. Χρωστάω σε όλους ένα μεγάλο ευχαριστώ!

Τέλος θελω να αναφερθώ συγκεκριμένα στον παππού μου, τον άνθρωπο που μου στάθηκε σα δεύτερος πατέρας. Την ηθική του, τον τρόπο ζωής του, αλλά και το γέλιο του δε θα ξεχάσω ποτέ. Τα μαθήματα που έχω πάρει από εκείνον με ακολουθούν σε κάθε μου βήμα. Δυστυχώς δεν πρόλαβε να δει αυτή την εργασία. Του την αφιερώνω.

Περίληψη

Οι μηχανές Turing αποτελούν το καλύτερο μαθηματικό μοντέλο για την μελέτη αλγορίθμων. Στην εργασία αυτή αποδεικνύουμε την ύπαρξη μηχανής Turing που λύνει το πρόβλημα πιστοποίησης πρώτων αριθμών σε πολυωνυμικό χρόνο ως προς το μέγεθος των δεδομένων του. Ο αλγοριθμος που χρησιμοποιούμε είναι μια παραλλαγή του αλγορίθμου των Agrawal, Kayal και Saxena (AKS) που παρουσιάστηκε το 2002. Χρησιμοποιούμε μηχανές Turing με πολλές ταινίες. Αρχικά κατασκευάζουμε μηχανές Turing που υλοποιούν βασικούς αριθμητικούς αλγορίθμους. Στην συνέχεια αποδεικνύουμε πώς μπορούμε να συνθέσουμε μηχανές Turing για υλοποίηση πιο περίπλοκων αλγορίθμων και υπό ποιες προϋποθέσεις αυτές τερματίζουν σε πολυωνυμικό χρόνο. Τέλος, συνθέτοντας κατάλληλα τις μηχανές που κατασκευάσαμε, αποδεικνύουμε την ύπαρξη μηχανής Turing που υλοποιεί τον αλγόριθμο πιστοποίησης πρώτων αριθμών σε πολυωνυμικό χρόνο.

Λέξεις κλειδιά: μηχανές Turing, πρώτοι αριθμοί, αλγόριθμος, πολυωνυμικός χρόνος, AKS.

Abstract

Turing Machines (TM) are the most suitable mathematical model for the study of algorithms. At this project we prove the existance of a TM that solves the PRIMES problem in polynomial time. We use a variation of the Agrawal-Kayal-Saxena (AKS) algorithm, published in 2002. For the cause we use multitaped TM. We start by constructing TM that implement simple arithmetic algorithms. Then we prove that we can combine TM to construct other TM that implement more complicated algorithms. We also demostrate under which circumstances these machines halt in polynomial time. Finaly, by combining the previously constructed machines, we prove the existence of a TM that implements AKS algorithm and halts in polynomial time.

Keywords: Turing machines, primes, algorithm, polynomial time, AKS.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Περιγραφή του προβλήματος

Το πρόβλημα πιστοποίησης πρώτων αριθμών (ΠΠΠ), δηλαδή το πρόβλημα απόφασης αν ένας φυσικός αριθμός n είναι πρώτος ή σύνθετος, ήταν γνωστό από την αρχαιότητα. Ο πρώτος αλγόριθμος επίλυσης του προβλήματος, το **κόσκινο του Ερατοσθένη**, ήταν ήδη γνωστός περίπου από το 240 π.Χ. Ο αλγόριθμος αυτός επιλύει όχι μόνον το ΠΠΠ, αλλά και το δυσκολότερο πρόβλημα της παραγοντοποίησης του αριθμού n. Το μειονέκτημα του αλγορίθμου του Ερατοσθένη είναι ότι χρειάζεται εκθετικό χρόνο ως προς το μέγεθος του αριθμού που εξετάζουμε. Οι σημερινοί ηλεκτρονικοί υπολογιστές, που εκτελούν εκατομμύρια πράξεις το δευτερόλεπτο, για αριθμούς με μερικές εκατοντάδες ψηφία θα χρειαστούν κάποιους αιώνες για να υλοποιήσουν τον αλγόριθμο του Ερατοσθένη. Για το ΠΠΠ, ένας αλγόριθμος θεωρούμε ότι είναι "ικανοποιητικός", αν επιτυγχάνει την επίλυση του προβλήματος σε χρόνο που φράσσεται από ένα **πολυώνυμο του μεγέθους των δεδομένων του**, δηλαδή φράσσεται από ένα **πολυώνυμο του λογαρίθμου των αριθμητικών δεδομένων**. Ένα πρόβλημα που επιλύεται από έναν τέτοιο αλγόριθμο λέμε ότι ανήκει στην κλάση **P** ([8] σελ. 275-278), που το όνομά της προέρχεται από το αρχικό γράμμα της φράσης Polynomial time .

Από την αρχαιότητα μέχρι και τα μέσα του 20ού αιώνα ελάχιστα σημαντικά καινούργια αποτελέσματα υπήρξαν στο συγκεκριμμένο πρόβλημα [9], [13]. Το αποφασιστικό κίνητρο εύρεσης νέων ταχύτερων αλγορίθμων ήταν η ανάπτυξη τότε μιας σειράς από κρυπτογραφικά συστήματα που χρησιμοποιούσαν μεγάλους πρώτους αριθμούς. Εν τω μεταξύ είχαν αναπτυχθεί πολύ, τόσο η Θεωρία Αριθμών όσο και η Θεωρία Αλγορίθμων.

Ένα πρόβλημα απόφασης λέμε ότι επιλύεται σε **μη-αιτιοκρατικό πολυωνυμικό χρόνο** αν υπάρχει ένας μη-αιτιοκρατικός αλγόριθμος που, εκμεταλλευόμενος μια τυχαία επιλογή, μπορεί σε πολυωνυμικό χρόνο να δώσει καταφατική απάντηση στο πρόβλημα ([8] σελ. 292,299). Η κλάση αυτών των προβλημάτων συμβολίζεται με **NP** (Nondeterministic Polynomial time). Α-

ντίστοιχα το $\mathbf{co-NP}$ είναι η κλάση των προβλημάτων που ένας αλγόριθμος απαντά καταφατικά στο συμπληρωματικό πρόβλημα.

Για έναν σύνθετο αριθμό μπορούμε να θεωρήσουμε ότι η "τυχαία επιλογή" είναι ένας γνήσιος διαιρέτης του. Προφανώς λοιπόν ΠΠΠ$\in \mathbf{co-NP}$, αφού το συμπληρωματικό του πρόβλημα, δηλαδή το πρόβλημα πιστοποίησης σύνθετου αριθμού, ανήκει στο $\mathbf{NP}$. Επιπλέον το 1974 ο Pratt [14] απέδειξε ότι το ΠΠΠ ανήκει στο $\mathbf{NP}$, άρα ΠΠΠ$\in (\mathbf{NP} \cap \mathbf{co-NP})$. Το γεγονός όμως ότι ΠΠΠ$\in (\mathbf{NP} \cap \mathbf{co-NP})$ **δεν** συνεπάγεται ότι ΠΠΠ$\in \mathbf{P}$. Τονίζουμε ότι το πρόβλημα αν $\mathbf{P} = (\mathbf{NP} \cap \mathbf{co-NP})$ παραμένει ένα **ανοικτό πρόβλημα** με επικρατούσα την εικασία ότι $\mathbf{P} = (\mathbf{NP} \cap \mathbf{co-NP})$. Παρ' όλα αυτά η επι μακρόν ενασχόληση μεγάλων μαθηματικών προς αυτήν την κατεύθυνση χωρίς να προκύψει κανένα ουσιαστικό αποτέλεσμα, αποτελεί σημαντικό επιχειρήμα των υποστηρικτών της εικασίας ότι η κλάση $\mathbf{P}$ είναι γνήσιο υποσύνολο της τομής των $\mathbf{NP}$ και $\mathbf{co-NP}$. Ο αλγόριθμος πάντως που θα παρουσιάσουμε εδώ έδωσε περισσότερες ελπίδες στους υποστηρικτές της ισότητας των δύο κλάσεων.

Ένα πρόβλημα απόφασης λέμε ότι είναι επιλύσιμο σε **πιθανοθεωριτικό πολυωνυμικό χρόνο** αν υπάρχει αλγόριθμος που να περιέχει την τυχαία επιλογή ενός αριθμού και η καταφατική απάντηση είναι πάντοτε σωστή, ενώ η αρνητική είναι σωστή με πιθανότητα μεγαλύτερη του 1/2. Η κλάση αυτών των προβλημάτων συμβολίζεται με $\mathbf{RP}$ (Randomized Polynomial time) και αντίστοιχα με $\mathbf{co-RP}$ για τα συμπληρωματικά τους προβλήματα [9]. Ένας αλγόριθμος που ανήκει και στις δύο κλάσεις λέμε ότι **επιλύεται με πιθανότητα λάθους μηδέν** και η κλάση αυτών των προβλημάτων συμβολίζεται με $\mathbf{ZPP}$ (Zero Posibility error Polynomial time). Με την παρουσίαση των αλγορίθμων των Solovay και Strassen [16] το 1977 και του Miller [9] το 1975 που βελτιώθηκε από τον Rabin [15] το 1980, αποδεικνύεται ότι το πρόβλημα πιστοποίησης πρώτων αριθμών ανήκει στο $\mathbf{co-RP}$. Λίγα χρόνια αργότερα, το 1986, παρουσιάζεται ο αλγόριθμος των Goldwasser και Kilian, που χρησιμοποιόντας τον με κατάλληλες μετατροπές οι Adleman και Huang απέδειξαν ότι το πρόβλημα πιστοποίησης πρώτων αριθμών ανήκει στο $\mathbf{RP}$ και συνεπώς στο $\mathbf{ZPP}$ [2].

Εν τω μεταξύ, το 1983, οι Adleman, Pomerance και Rumely βρήκαν έναν αιτιοκρατικό αλγόριθμο πιστοποίησης πρώτων αριθμών ο οποίος "τερματίζει" σε χρόνο $(\log n)^{O(\log\log\log n)}$ [1]. Το σημαντικό βήμα έγινε το 2002 από τους Agrawal, Kayal και Saxena, τρεις ινδούς μαθηματικούς, οι οποίοι απέδειξαν ότι η πιστοποίηση πρώτων αριθμών μπορεί να γίνει σε πολυωνυμικό χρόνο [3]. Ένα χαρακτηριστικό αυτού του αλγορίθμου είναι ότι χρησιμοποιεί σχεδόν στοιχειώδη θεωρήματα για την απόδειξή του, σε αντίθεση με πολλούς από τους προαναφερόμενους αλγορίθμους (όπως των Adleman, Pomerance, Rumely) [1] που χρησιμοποιούν πολύ ισχυρά αποτελέσματα κυρίως της Θεωρίας Αριθμών.

1.2 Περιγραφή της εργασίας

Στην παρούσα εργασία χρησιμοποιείται μια παραλλαγή του αλγορίθμου που δόθηκε από τους Agrawal, Kayal και Saxena το 2003 (AKS) και αποτελεί ουσιαστικά μικρή βελτίωση του αρχικού αποτελέσματος [4]. Ιδιαίτερη προσοχή δίδεται στην υλοποίηση του αλγορίθμου σε πολυωνυμικό χρόνο από μηχανές Turing το οποίο άλλωστε αποτελεί και το πρωτότυπο σκέλος της εργασίας αυτής. Αναλυτικά:

Στο κεφάλαιο 2 δίδονται οι κατάλληλοι ορισμοί για μηχανές Turing. Ανάμεσα στους διάφορους αποδεκτούς και ισοδύναμους ορισμούς επιλέχτηκαν αυτοί που δίνουν την δυνατότητα σε μη γνώστες της αντίστοιχης θεωρίας να παρακολουθήσουν ευκολότερα τα επόμενα αποτελέσματα.

Στο κεφάλαιο 3, που αποτελεί το κύριο μέρος της εργασίας, περιγράφεται ο τρόπος με τον οποίο μπορούν να υλοποιηθούν αλγόριθμοι, όπως πράξεις και έλεγχος αριθμητικών σχέσεων, από μηχανές Turing. Επίσης περιγράφονται αναλυτικά βασικοί αριθμητικοί αλγόριθμοι, όπως η υλοποίηση των τεσσάρων πράξεων της αριθμητικής και ο έλεγχος ανισότητας φυσικών αριθμών. Τέλος περιγράφονται και πιο περίπλοκοι και εξειδικευμένοι αλγόριθμοι, που αποδεικνύουμε ότι μπορούν να υλοποιηθούν από κατάλληλη σύνθεση προηγούμενων μηχανών Turing.

Τέλος, στο κεφάλαιο 4 παρουσιάζεται ο αλγόριθμος AKS και αποδεικνύεται ότι λύνει το ΠΠΠ σε πολυωνυμικό χρόνο.

Κεφάλαιο 2

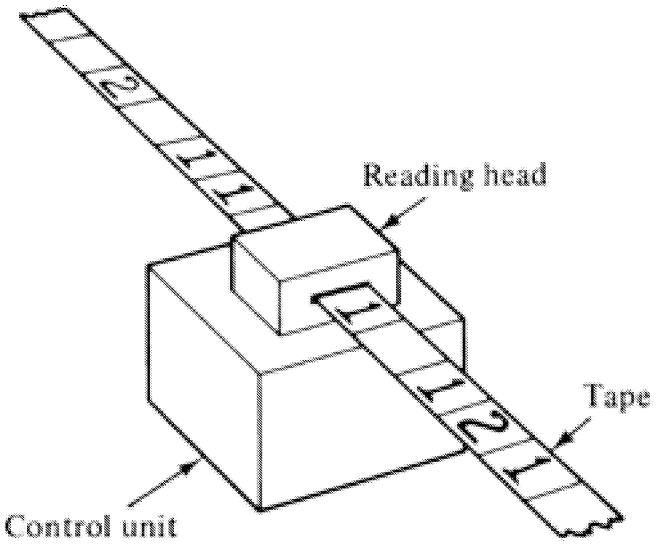
Γενικά

2.1 Περιγραφή μηχανών Turing

Ένας αιτιοκρατικός αλγόριθμος πρέπει να μπορεί να εκτελεστεί από μια αιτιοκρατική υπολογιστική μηχανή. Οι μηχανές Turing αποτελούν το καλύτερο ίσως μαθηματικό μοντέλο τέτοιων μηχανών. Αποτελούν δε μοντέλο διότι οι μηχανές Turing δεν είναι μηχανές με την στενή έννοια του όρου, καθότι δεν μπορούν να κατασκευαστούν μηχανικά. Μια **Βασική Μηχανή Turing** αποτελείται από μια άπειρα εκτεινόμενη ταινία προς τη μία κατεύθυνση, η οποία είναι χωρισμένη σε κελιά (όπως ένα φωτογραφικό film) και συνδέεται με έναν οδηγό (βλ'επε σχήμα στην επόμενη σελιδα [17]). Κάθε κελί της ταινίας περιέχει ένα και μόνο σύμβολο, που αποτελεί μία ατομική πληροφορία. Ο οδηγός μπορεί να διαβάζει κάθε φορά ένα και μόνο ένα κελί της μηχανής και συνδέεται με μια μονάδα ελέγχου που περιέχει κάποιο πρόγραμμα. Τέλος, το πρόγραμμα της μονάδας ελέγχου από κοινού με το σύμβολο που διαβάζει σε μια δεδομένη χρονική στιγμή ο οδηγός καθορίζουν μονοσήμαντα το πώς θα συμπεριφερθεί η μηχανή.

Οι διάφορες δράσεις που μπορεί να κάνει μια μηχανή Turing είναι σχετικά περιορισμένες. Πιο συγκεκριμένα η μηχανή είτε θα τερματίσει τη λειτουργία της, είτε θα κάνει ένα **βήμα**. Σε ένα βήμα η μηχανή μπορεί να γράψει στο κελί που βρίσκεται ο οδηγός ένα σύμβολο (πιθανόν το ίδιο με το προηγούμενο), να μετακινήσει τον οδηγό κατά μια θέση αριστερά ή δεξιά και να μπει η μονάδα ελέγχου σε μια νέα κατάσταση.

Η λειτουργία της μηχανής Turing χαρακτηρίζεται από την εξής αλληλουχία γεγονότων: Αρχικά η ταινία της μηχανής εφοδιάζεται με πεπερασμένα το πλήθος σύμβολα που σχηματίζουν μια λέξη, τη **λέξη εισόδου** ή **είσοδο**, ενώ όλη η υπόλοιπη ταινία είναι κενή. Ο οδηγός τοποθετείται στο πρώτο κελί της ταινίας και η μονάδα ελέγχου αρχίζει να εκτελεί το πρόγραμμα. Από το σημείο αυτό η μηχανή ξεκινά να λειτουργεί κάνοντας μια ακολουθία βημάτων. Ανάλογα με το πρόγραμμα της μηχανής και την είσοδο που δίνουμε, οι υπολογισμοί δύνανται να συνεχιστούν επ᾽ άπειρον ή να τερματιστούν μετά από ένα



Σχήμα 2.1: Μηχανή Turing

πεπερασμένο πλήθος βημάτων. Εάν αυτό συμβεί, τότε η ακολουθία συμβόλων που περιέχει η ταινία θα αποτελούν το αποτέλεσμα του υπολογισμού ή **έξοδο** της μηχανής.

2.2 Μηχανές Turing

Συμβολίζουμε με Σ το σύνολο των συμβόλων που μπορούν να γραφούν στα κελιά της ταινίας. Το Σ είναι ξένη ένωση δύο συνόλων $\Sigma_1 \cup \Sigma_2$ όπου το Σ_1 ονομάζεται **βασικό αλφάβητο** και είναι μη-κενό και το Σ_2 ονομάζεται **βοηθητικό αλφάβητο** και περιέχει τουλάχιστον τα σύμβολα b και $\triangleright$, με το πρώτο να συμβολίζει το κενό και το δεύτερο να χρησιμοποιείται βοηθητικά στο πρώτο κελί της ταινίας ώστε να είναι αναγνωρίσιμο από την μηχανή. Έπειτα από αυτή την εισαγωγή είμαστε έτοιμοι να δώσουμε ένα μαθηματικό ορισμό των μηχανών Turing.

Ορισμός 2.2.1. *Μια* **βασική μηχανή Turing** M *είναι μια πεντάδα* $< K, \Sigma, s, H, \delta >$, *όπου*

- K *είναι ένα πεπερασμένο σύνολο* **καταστάσεων**
- Σ *είναι ένα πεπερασμένο σύνολο συμβόλων (***αλφάβητο***),* $\mu\epsilon\ \Sigma = \Sigma_1 \cup \Sigma_2$ *όπως παραπάνω,*
- $s \in K$ *είναι η* **αρχική κατάσταση**
- $H \subseteq K$ *είναι ένα σύνολο* **τελικών καταστάσεων**

- $\delta : (K \backslash H) \times \Sigma \rightarrow K \times \Sigma \times \{$*αριστερά, δεξιά*$\}$ *είναι η* **συνάρτηση μετάβασης** *ή* **πρόγραμμα της M**.

Μπορεί να αποδειχθεί ότι η υπολογιστική ισχύς, δηλαδή η κλάση των αλγορίθμων (υπολογίσιμων συναρτήσεων) που μπορεί να υλοποιηθούν από μια μηχανή, Turing δεν εξαρτάται από το πλήθος των στοιχείων του αλφαβήτου [π.χ. Lewis-Papadimitriou, Hennie]. Επιπλέον μπορούμε στο σύνολο των μετακινήσεων του οδηγού να προσθέσουμε και μία τρίτη, αυτή της σταθερότητας, δηλαδή ο οδηγός να μην αλλάζει κελί. Και πάλι δεν αλλάζει η υπολογιστική ισχύς, όμως η λειτουργία των μηχανών γίνεται πολύ πιο κατανοητή για τον άνθρωπο-παρατηρητή. Τέλος, στις μηχανές που θα κασκευάσουμε στο πρώτο κελί της ταινίας θα τοποθετείται πάντοτε το σύμβολο $\triangleright$.

Στην συνέχεια θα δώσουμε μια σειρά από ορισμούς με στόχο να ορίσουμε τι σημαίνει "χρονική πολυπλοκότητα" μιας μηχανής Turing M.

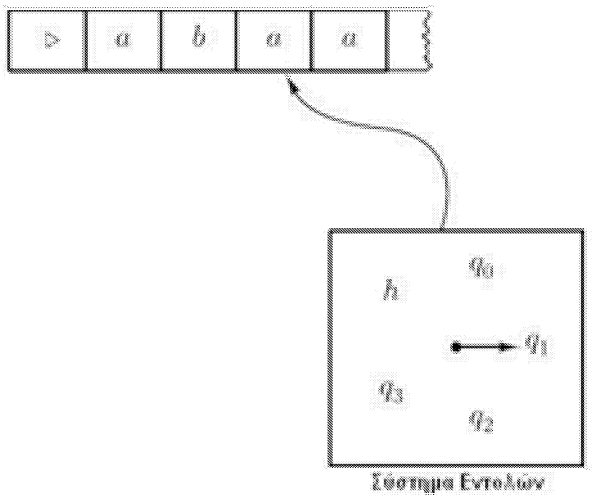
Ορισμός 2.2.2. *Έστω αλφάβητο* Σ. *Το* Σ^* *είναι το* **σύνολο των λέξεων** *από το αλφάβητο* Σ, *δηλαδή το σύνολο των πεπερασμένων ακολουθιών με τιμές στο* Σ. *Συμβολίζουμε με* e *την* **κενή λέξη**. *Επιπλέον αν* $w \in \Sigma^*$, *τότε το μήκος της λέξης* w *το συμβολίζουμε* $|w|$.

Ορισμός 2.2.3. Στιγμιότυπο *μιας μηχανής Turing* $M = <K, \Sigma, s, H, \delta>$ *είναι μία ακολουθία συμβόλων της μορφής* uqw, *όπου* $u \in \Sigma^*$, $q \in K$, $w \in \Sigma^*$, *και εάν* $w \neq e$, *τότε το τελευταίο σύμβολο της* w *είναι διάφορο του* b.

Σχόλιο. Κάθε στιγμιότυπο προσδιορίζει αμφιμονοσήμαντα μία φάση λειτουργίας της μηχανής, με την εξής ερμηνία: Η λέξη u, είναι το τμήμα της λέξης που περιέχει η ταινία μίας μηχανής μέχρι και το κελί που διαβάζει ο οδηγός και η λέξη w είναι η λέξη που βρίσκεται δεξιά του οδηγού, μέχρι και το τελευταίο μη-κενό κελί. Έτσι από κάθε στιγμιότυπο μπορούμε να προσδιορίσουμε το περιεχόμενο της ταινίας (από τις λέξεις u και w), την θέση του οδηγού (από το τελευταίο γράμμα της λέξης u) και την κατάσταση της μηχανής (από το q). Εδω σημειώνουμε το εξής: οι λέξεις u και w έχουν πεπερασμένο μήκος. Αρκούν όμως για να περιγράψουν το περιεχόμενο της ταινίας, επειδή σε κάθε φάση λειτουργίας της μηχανής το πλήθος των κελιών που είναι μη-κενά παραμένει πεπερασμένο. Στο παρακάτω σχήμα βλέπουμε μια μηχανή Turingμε τα χαρακτηριστικά που περιγριγράψαμε πριν.

Ορισμός 2.2.4. *Ορίζουμε* **αρχικό στιγμιότυπο** *με* **είσοδο** *την λέξη* w *το στιγμιότυπο* $\triangleright sw$ *και* **τελικό στιγμιότυπο** *κάθε στιγμιότυπο όπου εμφανίζεται τελική κατάσταση της μηχανής.*

Σε κάθε βήμα της λειτουργίας της μηχανής το τρέχον στιγμιότυπο και η συνάρτηση μετάβασης ορίζουν μονοσήμαντα το επόμενο βήμα της μηχανής. Για κάθε μη τελικό στιγμιότυπο ορίζεται το αμέσως επόμενο. Για τελικά στιγμιότυπα δεν ορίζεται αμέσως επόμενο στιγμιότυπο και η μηχανή τερματίζει τη λειτουργία της.



Σχήμα 2.2: Μηχανή Turing

Ορισμός 2.2.5. *Έστω μηχανή Turing* $M = < K, \Sigma, s, H, \delta >$. *Έστω επίσης* $w_1, w_2, u_1, u_2 \in \Sigma^*$, $a_1, a_2, a_3 \in \Sigma$, $q_1, q_2 \in K$ *με* $q_1 \notin H$. *Λέμε ότι το στιγμιότυπο* $w_1 q_1 a_1 u_1$ *είναι* **αμέσως επόμενο** *του* $w_2 q_2 a_2 u_2$ *αν:*

i) $\delta(q_1, a_1) = (q_2, a_3, \text{αριστερά})$ *και* $w_1 = w_2 a_2$, $u_2 = a_3 u_1$ *ή*

ii) $\delta(q_1, a_1) = (q_2, a_3, \text{δεξιά})$ *και* $w_2 = w_1 a_3$, $u_1 = a_2 u_2$,

και γράφουμε

$$w_1 q_1 a_1 u_1 \vdash_M w_2 q_2 a_2 u_2.$$

Συνεπώς κάθε μη-τελικό σιγμιότυπο έχει επόμενο και κανένα τελικό στιγμιότυπο δεν έχει επόμενο.

Ορισμός 2.2.6. *Μια ακολουθία στιγμιοτύπων* $\sigma_0, \sigma_1, ..., \sigma_n$ *της* M *λέγεται* **μερικός υπολογισμός** *αν για κάθε* $i = 0, ..., n-1$ *ισχύει* $\sigma_i \vdash \sigma_{i+1}$. *Ο αριθμός* n *ονομάζεται* **χρόνος του υπολογισμού**. *Αν επιπλέον το* σ_0 *είναι αρχικό στιγμιότυπο και το* σ_n *τελικό, τότε λέγεται* **ολοκληρωμένος υπολογισμός** *ή απλά* **υπολογισμός**.

Ένας υπολογισμός λοιπόν είναι μια ακολουθία διαδοχικών στιγμιοτύπων που περιγράφουν την λειτουργία της M. Αν στην μηχανή M εισάγουμε την λέξη w , τότε η μηχανή είτε θα τερματίσει μετά από πεπερασμένο πλήθος βημάτων είτε θα συνεχιζει την λειτουργία της επ᾽ άπειρον.

Ορισμός 2.2.7. *Έστω μηχανή Turing* M. *Ορίζουμε την* **συνάρτηση χρόνου υπολογισμού της** M, $\Upsilon_M : \Sigma^* \to \mathbb{N} \cup \{\infty\}$ *με* $\Upsilon_M(w)$ *να είναι το πλήθος βημάτων (ή* **χρόνος**) *υπολογισμού της* M *με είσοδο* w. *Αν ο υπολογισμός με είσοδο* w *τερματίζει θα έχουμε* $\Upsilon_M(w) \in \mathbb{N}$ *ενώ αν όχι* $\Upsilon_M(w) = \infty$.

Όλες οι μηχανές Turing, που θα μας απασχολήσουν στη συνέχεια, όποια και αν είναι η είσοδος που θα δώσουμε, θα τερματίζουν μετά από πεπερασμένα το πλήθος βήματα.

Ορισμός 2.2.8. *Έστω μηχανή Turing M που τερματίζει με κάθε είσοδο σε πεπερασμένα το πλήθος βήματα. Ορίζουμε ως* **συνάρτηση χρονικής πολυπλοκότητας** *ή απλά* **χρονική πολυπλοκότητα** *της M τη συνάρτηση $C_M : \mathbb{N} \to \mathbb{N}$ με τύπο*

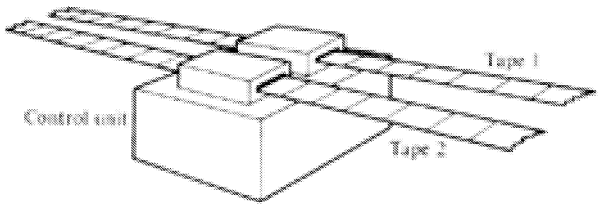
$$C_M(n) = \max\{\Upsilon_M(w) : |w| \geq n\}.$$

2.3 Μηχανές Turing με περισσότερες ταινίες

Μπορούμε να θεωρήσουμε μηχανές Turing με περισσότερες από μία ταινίες. Οι μηχανές αυτές εξακολουθούν να έχουν μια μονάδα ελέγχου (βλέπε σχήμα στην επόμενη σελίδα για μηχανή Turing με δύο ταινίες [7]), η οποία όμως συνδέεται με περισσότερους οδηγούς που ο κάθε ένας είναι προσαρμοσμένος σε μία ταινία και κινούνται ανεξάρτητα ο καθε ένας, ανάλογα με τη συνάρτηση μετάβασης. Η λειτουργία της μηχανής παραμένει αντίστοιχη με τη λειτουργία των βασικών μηχανών Turing, με τη διαφορά ότι η συνάρτηση μετάβασης μιας μηχανής με k ταινίες, είναι πλέον μια συνάρτηση της μορφής

$$\delta : K \times \Sigma^k \to K \times \Sigma^k \times \{\text{αριστερά, δεξιά}\}^k.$$

Κατά τα άλλα, όλες οι προηγούμενες έννοιες που έχουν οριστεί επεκτείνονται



Σχήμα 2.3: Μηχανή Turing

φυσιολογικά και για μηχανές με περισσότερες ταινίες: Η είσοδος θα είναι μια διατεταγμένη k-άδα λέξεων και τα στιγμιότυπα θα περιγράφουν την τρέχουσα κατάσταση, το τρέχον περιεχόμενο των ταινιών και τις θέσεις των οδηγών. Το **μήκος** μιας τέτοιας k-άδας $(w_1, ..., w_k)$ το ορίζουμε ως

$$\|w_1, ..., w_k\| = \max\{|w_1|, ..., |w_k|\}.$$

Θεώρημα 2.3.1. *Έστω μηχανή Turing M, που χρησιμοποιεί k ταινίες και εκτελεί έναν υπολογισμό σε χρόνο $O(t)$. Τότε μπορεί να κατασκευαστεί βασική μηχανή Turing M' που θα εκτελεί τον ίδιο υπολογισμό σε χρόνο $O(t \cdot (|x|+t))$, όπου x είναι λέξη εισόδου της M'.*

Για την απόδειξη του παραπάνω θεωρήματος βλέπε [8] σελ. 81-83 και [17].

Παρατήρηση: Ειδικά αν $C_M(|x|) \leq O(|x|^q)$, τότε $C_{M'}(|x|) \leq O(|x|^{2q})$. Άρα αν η M τερματίζει σε πολυωνυμικό χρόνο ως προς το μήκος εισόδου, τότε και η M' τερματίζει σε πολυωνυμικό χρόνο.

2.4 Υπολογίσιμες συναρτήσεις

Για κάθε μηχανή Turing μπορούμε να ορίσουμε μια συνάρτηση, η οποία για όρισμα μια είσοδο θα δίνει αποτέλεσμα την έξοδο της μηχανής. Λόγω του ότι η μηχανή μπορεί να μην τερματίζει, η συνάρτηση αυτή θα είναι στη γενική της μορφή μερική, άρα με όρισμα κάποιες εισόδους μπορεί να μη δίδει αποτέλεσμα. Σημειώνουμε πάντως ότι οι μηχανές που εξετάζονται στην παρούσα εργασία τερματίζουν πάντοτε και συνεπώς δεν μας απασχολούν τέτοιες περιπτώσεις.

Ορισμός 2.4.1. *Έστω μηχανή Turing M με k ταινίες. Η (μερική) συνάρτηση $\varphi_M : (\Sigma^*)^k \to (\Sigma^*)^k$ με $\varphi_M(u_1, ..., u_k) \simeq (w_1, ..., w_k)$ λέγεται* **συνάρτηση που υπολογίζει η μηχανή** *M αν ο υπολογισμός της M με είσοδο $(u_1, ..., u_k)$ δίνει έξοδο $(w_1, ..., w_k)$.*

Συχνά χρησιμοποιούμε μηχανές Turing με k ταινίες για έναν υπολογισμό $\varphi : (\Sigma^*)^q \to (\Sigma^*)^r$, με $p, q \leq k$. Τότε εφοδιάζουμε p από τις ταινίες με το όρισμα και αφήνουμε τις υπόλοιπες κενές. Στο τέλος θεωρούμε ως αποτέλεσμα τις λέξεις που περιέχονται σε q μόνον από τις k ταινίες.

Ορισμός 2.4.2. *Μια (μερική) συνάρτηση $\vartheta : (\Sigma^*)^q \to (\Sigma^*)^r$ λέγεται* **Turing υπολογίσιμη** *αν υπάρχει μηχανή Turing M τέτοια, ώστε:*

$$\forall \vec{\sigma} \in (\Sigma^*)^q \quad \forall \vec{\tau} \in (\Sigma^*)^r : \quad \vartheta(\vec{\sigma}) \simeq \vec{\tau} \quad \Longleftrightarrow \quad \varphi_M(\vec{\sigma}, \underbrace{e, ..., e}_{k-q}) \simeq (\vec{\tau}, \underbrace{e, ..., e}_{k-r})$$

Το γεγονός ότι θεωρούμε πως η "είσοδος" και η "έξοδος" βρίσκονται στις πρώτες q και r ταινίες αντίστοιχα προφανώς δεν μας περιορίζει, αφού μια εναλλαγή περιεχομένου δύο ταινιών γίνεται σε χρόνο όσο και η ανάγνωση των λέξεων αυτών. Έτσι μπορούμε αντ' αυτών να επιλέξουμε κάποιες άλλες ταινίες κατά βούληση, πρακτική που θα εφαρμοστεί στην παρούσα εργασία.

Κεφάλαιο 3

Αριθμητικοί αλγόριθμοι

3.1 Γενικά

Το σύνολο λέξεων $N = \{0\} \cup \{0,1\}^*1$ αποτελείται ακριβώς από τα ανεστραμμένα δυαδικά αναπτύγματα των φυσικών αριθμών. Ορίζουμε τη συνάρτηση $\mathbf{\Lambda} : \mathbb{N} \to N$, η οποία για όρισμα ένα φυσικό αριθμό n μας δίνει τη λέξη από το N που αντιστοιχεί στο ανεστραμμένο δυαδικό ανάπτυγμα του n και την αντίστροφή της $\mathbf{A} : N \to \mathbb{N}$, που μας δίνει την αριθμητική τιμή της ανεστραμμένης λέξης. Για συντομία, το $\mathbf{\Lambda}(n)$ το συμβολίζουμε και $\hat{n}$ ενώ το $\mathbf{A}(w)$ και $\check{w}$. Για κάθε πράξη $*$ μεταξύ φυσικών αριθμών, ορίζουμε την πράξη $*_\Lambda$ μεταξύ λέξεων από το N, με την ιδιότητα $u *_\Lambda v = w$, αν και μόνον αν $\check{u} * \check{v} = \check{w}$. Αντίστοιχα, για κάθε αριθμητική σχέση R ορίζουμε την σχέση R_Λ στο N, με την ιδιότητα $R_\Lambda(u_1, ...u_s)$ αν και μόνον αν $R(\check{u}_1, ..., \check{u}_s)$. Γενικότερα λοιπόν, για κάθε συνάρτηση $f : \mathbb{N}^q \to \mathbb{N}^r$, ορίζεται συνάρτηση $\hat{f} : N^q \to N^r$ όπου

$$\hat{f}(u_1, ..., u_q) = (w_1, ..., w_r) \quad \Longleftrightarrow \quad f(\check{u}_1, ..., \check{u}_q) = (\check{w}_1, ..., \check{w}_r).$$

Ο στόχος των παραπάνω είναι προφανής. Ορίζοντας κατάλληλα πράξεις μεταξύ λέξεων, μπορούμε να κατασκευάσουμε μηχανές Turing που να τις εξετάζουν. Έτσι μπορούμε να πούμε ότι έμμεσα οι μηχανές Turing εκτελούν πράξεις με φυσικούς αριθμούς.

Λέμε ότι μια μηχανή Turing εκτελεί έναν αριθμητικό υπολογισμό σε χρόνο $O(f(x))$, αν η μηχανή υλοποιεί τον αντίστοιχο αλγόριθμο στο N και η συνάρτηση χρονικής πολυπλοκότητας της μηχανής είναι της τάξης $O(f(x))$, όπου x θα είναι το **μήκος της λέξης εισόδου**. Παρατηρούμε ότι $|\mathbf{\Lambda}(n)| = O(\log n)$. Οπότε οι υπολογισμοί της μηχανής είναι της τάξης $O(f(\log n))$.

Με τον ίδιο τρόπο μπορούμε να ελέγξουμε την ισχύ ή όχι μιας αριθμητικής σχέσης. Ο ένας τρόπος να γίνει αυτό είναι να υπολογίσουμε την χαρακτηριστική συνάρτηση της σχέσης. Όμως στην συνέχεια της εργασίας επιλέγουμε μια διαφορετική προσσέγγιση. Η ιδέα που θα χρειαστεί είναι ορίσουμε περισσοτερες από μία τελικές καταστάσεις (π.χ. καταστάσεις $q_{\mathrm{NAI}}, q_{\mathrm{OXI}}$ με την προφανή ερμηνεία). Έτσι μια μηχανή Turing θα λέμε ότι επιβεβαιώνει μια

αριθμητική σχέση, με την έννοια ότι αν της δωθεί η λέξη που αντιστοιχεί στο δυαδικό ανάπτυγμα αυτού του αριθμού, εκείνη θα τερματίζει και ανάλογα με την λέξη θα προκύπτει σαν τελική κατάσταση εκείνη που προσδιορίζει την επιβεβαίωση ή όχι μιας σχέσης. Σημειώνουμε ότι το να ελέγχεται μια σχέση από μια μηχανή Turing είναι ισοδύναμο με το να υπολογίζεται η χαρακτηριστική της συνάρτηση. Η ιδέα με τις πολλές τελικές καταστάσεις, θα μας βοηθήσει αργότερα, όταν χρειαστεί να συνθέσουμε μηχανές.

Στην συνέχεια της εργασίας θα χρησιμοποιήσουμε μηχανές Turing με πολλές ταινίες και με αλφάβητο το $\Sigma = \{b, 0, 1, \triangleright, \#\}$. Οι είσοδοι που θα δέχεται μια μηχανή θα είναι λέξεις από το N, δηλαδή ανεστραμένα δυαδικά αναπτύγματα φυσικών αριθμών. Αν χρειαστεί να εισάγουμε περισσοτερα από ένα δυαδικά αναπτύγματα τότε αυτά θα εισάγονται είτε σε διαφορετικές ταινίες είτε στην ίδια ταινία στην σειρά από τα αριστερά προς τα δεξιά, παρεμβάλοντας ανάμεσά τους το σύμβολο $\#$. Επίσης είναι χρήσιμο να μπορεί να αναγνωρίζει η μηχανή το αρχικό κελί κάθε ταινίας. Γι᾽ αυτό και πάντοτε στο πρώτο κελί κάθε ταινίας θα εισάγεται το σύμβολο $\triangleright$. Τέλος οι κεφαλές τοποθετούνται έτσι ώστε να διαβάζουν σε κάθε ταινία το σύμβολο $\triangleright$. Σημειώνουμε επίσης ότι στην συνέχεια αντί να λέμε "εισάγουμε την λέξη $\triangleright w$", λέμε για συντομία "εισάγουμε την λέξη w".

3.2 Βασικές πράξεις σε μηχανές Turing

3.2.1 Σύγκριση φυσικών αριθμών

Πρόταση 3.2.1. *Υπάρχει μηχανή Turing M που, αν της δοθούν οι λέξεις $u, w \in N$, ελέγχει την σχέση $u <_\Lambda w$, σε χρόνο $O(\min\{|u|, |w|\})$.*

Παρατηρήσεις: Ο έλεγχος της σχέσης $m < n$ γίνεται από τη M μέσω των δυαδικών αναπτυγμάτων τους, εξετάζεται δηλαδή η σχέση $\hat{m} <_\Lambda \hat{n}$.

Η σχέση $<_\Lambda$ (αντίστοιχη της διάταξης $<$ στο $\mathbb{N}$) είναι η αντιλεξικογραφική διάταξη στο N, δηλαδή

$$u <_\Lambda w \quad \Leftrightarrow \quad |u| < |w| \vee (|u| = |w| \wedge \exists u_1 \exists u_2 \exists v (u = u_1 0 v \wedge w = w_1 1 v)).$$

Απόδειξη. Έστω μηχανή Turing M με δύο ταινίες. Εισάγουμε τα δυαδικά αναπτύγματα u, w στην πρώτη και δεύτερη ταινία αντίστοιχα. Η μηχανή λειτουργεί ως εξής: Αρχικά συγκρίνει τα μήκη των λέξεων u και w και αν είναι ίσα, τότε συγκρίνει τις λέξεις με βάση την αντιλεξικογραφική διάταξη. Τα βήματα που θα εκτελέσει η μηχανή όταν $|u| \neq |w|$ θα είναι $\min\{|u|, |w|\} + 1$ και όταν $|u| = |w|$ το πολύ $2 \cdot |u| + 2$. Άρα γενικά είναι $O(\min\{|u|, |w|\})$. $\square$

Με παρόμοιο τρόπο κατασκευάζεται αλγόριθμος που ελέγχει την σχέση $u = w$. Αξίζει βέβαια να σημειωθεί ότι στο N η σχέση "$=$" ταυτίζεται με την σχέση "$=_\Lambda$". Επίσης αντίστοιχα κατασκευάζουμε μηχανες Turing που να ελέγχουν τις σχέσεις $<_\Lambda$, $\leq_\Lambda$ και $\geq_\Lambda$.

3.2.2 Πρόσθεση φυσικών αριθμών

Πρόταση 3.2.2. *Υπάρχει μηχανή Turing, που αν της δοθούν οι λέξεις $u, w \in N$, δίνει σαν έξοδο την λέξη $u +_\Lambda w$ σε χρόνο $O(\max\{|u|, |w|\})$. Δηλαδή η αριθμητική τιμή της λέξης εξόδου είναι το άθροισμα των αριθμητικών τιμών των λέξεων εισόδου.*

Απόδειξη. Έστω μηχανή Turing M με τρεις ταινίες, και σύνολο καταστάσεων $K = \{s, q_0, q_1, h\}$. Στις δύο πρώτες ταινίες εισάγουμε τις λέξεις u και w αντίστοιχα ενώ η τρίτη θα μας δώσει την έξοδο. Οι καταστάσεις q_0, q_1 αντιστοιχούν στο "κρατούμενο μηδέν" και "κρατούμενο ένα". Η διαδικασία που ακολουθεί η μηχανή είναι ουσιαστικά η διαδικασία της πρόσθεσης δύο αριθμών όπως την κάνουμε με χαρτί και μολύβι στο δημοτικό σχολείο (αλλά σε δυαδικό σύστημα). Τα βήματα που θα κάνει η μηχανή μέχρι να τερματίσει θα είναι ακριβώς $\max\{|u|, |w|\} + 1$. □

Σχόλιο: Μπορούμε να φτιάξουμε μηχανή Turing με μία ταινία, που όταν εισάγουμε την λέξη u θα δίνει έξοδο την $u +_\Lambda 1$. Η χρονική πολυπλοκότητα της μηχανής είναι $O(|u|)$. Επίσης χρησιμοποιώντας την ιδέα της παραπάνω μηχανής εύκολα μπορεί να κατασκευαστεί μηχανή Turing με δύο ταινίες που στην πρώτη να δίνεται η λέξη u και στην δεύτερη να εξάγουμε την $\mathbf{\Lambda}(\lfloor \log \check{u} \rfloor)$. Η χρονική πολυπλοκότητα αυτής της μηχανής είναι μικρότερη από $O(|u|^{1+\varepsilon})$, για κάθε $\varepsilon > 0$. Τέλος για τον υπολογισμό της περιορισμένης αφαίρεσης φυσικών αριθμών μπορεί να κατασκευαστεί μηχανή Turing με χρονική πολυπλοκότητα $O(\max\{|u|, |w|\})$ που με είσοδο τα $u, w \in N$, να δίνει έξοδο $u -_\Lambda w$.

3.2.3 Πολλαπλασιασμός φυσικών αριθμών

Πρόταση 3.2.3. *Υπάρχει μηχανή Turing, που αν της δωθούν οι λέξεις $u, w \in N$, δίνει σαν έξοδο την λέξη $u \cdot_\Lambda w$ σε χρόνο $O(|u| \cdot |w|)$. Δηλαδή η αριθμητική τιμή της λέξης εξόδου είναι το γινόμενο των αριθμητικών τιμών των λέξεων εισόδου.*

Απόδειξη. Έστω μηχανή Turing M με τρεις ταινίες, και σύνολο καταστάσεων $K = \{s, q_0, q_1, q_\alpha, q_\delta, q_\triangleright, h\}$. Και εδώ η μηχανή θα μιμηθεί την μέθοδο με την οποία κάνουμε πολλαπλασιασμό με χαρτί και μολύβι, με δύο διαφορές: Πρώτον πρέπει να υπολογίζουμε απ᾽ ευθείας τα μερικά αθροίσματα και δεύτερον πρέπει να φροντίζουμε ώστε η μηχανή να υπολογίζει κάθε φορά πού ακριβώς θα γράφει το αποτέλεσμα της πράξης. Και πάλι εδώ οι καταστάσεις q_0, q_1 αντιπροσωπεύουν τις φάσεις "κρατούμενο μηδέν" και "κρατούμενο ένα". Αν στις τρείς ταινίες οι οδηγοί διαβάζουν a, b, c και η μηχανή βρίσκεται στην κατάσταση q_d, όπου $a, b, c, d \in \{0, 1\}$, τότε γράφει στο κελί της τρίτης ταινίας το ψηφίο μονάδων του αριθμού $a \cdot b + c + d$ και μπαίνει στην κατάσταση που υπαγορεύει το ψηφίο των δυάδων του αριθμού. Κάθε φορά που εκτελείται μια τέτοια πράξη οι οδηγοί στην πρώτη και την τρίτη ταινία μετακινούνται μια θέση δεξιά, ενώ ο οδηγός της δεύτερης ταινίας μένει στο ίδιο κελί. Αν μετά από

αυτήν τη μετακίνηση στην πρώτη ταινία ο οδηγός διαβάζει b, τότε η μηχανή μπαίνει στην κατάσταση q_δ. Στην q_δ η μηχανή μετακινεί δεξιά του οδηγούς της δεύτερης και τρίτης ταινίας, και η μηχανή μπαίνει στην κατάσταση q_α. Στην q_a αν στην δεύτερη ταινία διαβάζει b τότε η μηχανή μπαίνει στην κατάσταση h και τερματίζει, αλλιώς μετακινεί τους οδηγούς της πρώτης και της τρίτης ταινίας αριστερά μέχρι να εμφανιστεί στην πρώτη ταινία το σύμβολο $\triangleright$. Τότε μπαίνει στην κατάσταση $q_\triangleright$ και μετακινεί τους οδηγούς της πρώτης και της τρίτης ταινίας μια θέση δεξιά και μπαίνει στην κατάσταση q_0. Οι κινήσεις που θα κάνει η μηχανή είναι λιγότερες από $2 \cdot (|u| + 1) \cdot (|w| + 1)$, άρα η χρονική πολυπλοκότητα της μηχανής ειναι $O(|u| \cdot |w|)$. □

Σχόλιο: Ειδική περίπτωση είναι η "ύψωση στο τετράγωνο". Εννούμε φυσικά την εύρεση της λέξης που αντιστοιχεί στην αριθμητική τιμή του τετραγώνου της αριθμητικής τιμής του u. Τότε η χρονική πολυπλοκότητα είναι $O(|u|^2)$. Σημειώνουμε εδώ ότι το μήκος της λέξης εξόδου, θα είναι $|\mathbf{\Lambda}(n^2)| \leq 2 \cdot |\mathbf{\Lambda}(n)|$.

3.2.4 Διαίρεση φυσικών αριθμών

Πρόταση 3.2.4. *Υπάρχει μηχανή Turing M, που αν της δωθούν οι λέξεις $u, w \in N$, δίνει σαν έξοδο*[1] *τις λέξεις $u \div_\Lambda w$ και $u \bmod_\Lambda w$ σε χρόνο $O(|u| \cdot |w|)$.*

Σχόλιο: Η Turing-υπολογίσιμη συνάρτηση που εξετάζεται εδώ είναι η $f(x, y) = (x \div y, x \mod y)$. Αντί για την f όμως μπορούμε να πούμε ότι υπολογίζονται δύο συναρτήσεις την $f_1(x, y) = x \div y$ και την $f_2(x, y) = x \mod y$ ταυτόχρονα, το οποίο είναι τελικά προτιμότερο καθώς τονίζει ότι τα αποτελέσματα μπορούν να εξαχθούν ανεξάρτητα το ένα από το άλλο.

Απόδειξη. Έστω μηχανή Turing M με πέντε ταινίες. Για άλλη μία φορά η μηχανή θα μιμηθεί τη διαίρεση φυσικών αριθμών όπως την κάνουμε με χαρτί και μολύβι. Στις δύο πρώτες ταινίες εισάγονται οι λέξεις u, w. Στην τρίτη ταινία αντιγράφεται ένα γράμμα της λέξης u ξεκινώντας από το δεξιότερο και μάλιστα γράφεται σε σχετική θέση ως προς το $\triangleright$ αντίστοιχη με αυτή της πρώτης ταίνιας. Κάθε φορά που γίνεται αυτό η μηχανή εκτελεί τα ακόλουθα: Συγκρίνει, ως προς την σχέση $>_\Lambda$, τις λέξεις που βρίσκονται στη δεύτερη και τρίτη ταινία. Αν η λέξη στη δεύτερη ταινία είναι "μεγαλύτερη" τότε γράφει 0 στην πέμπτη ταινία, μετακινεί τον οδηγό της δεξιά και μπαίνει πάλι στην κατάσταση που αντιγράφει γράμματα από την πρώτη στην τρίτη ταινία. Αν πάλι η λέξη στην τρίτη ταινία είναι "μεγαλύτερη ή ίση" από τη λέξη στην δεύτερη, τότε γράφει 1 στην πέμπτη ταινία, μετακινεί τον οδηγό της δεξιά και: "αφαιρεί" την λέξη της δεύτερης από την λέξη της τρίτης ταινίας. Το αποτέλεσμα γράφεται στην τέταρτη ταινία. Στην συνέχεια αντιγράφεται η τέταρτη ταινία στην δεύτερη (στις αντίστοιχες πάλι θέσεις), η μηχανή αντιγράφει το επόμενο γράμμα της u αριστερά από την λέξη που υπάρχει στην τρίτη ταινία και επαναλαμβάνει την

[1] Για δύο φυσικούς αριθμούς m, n συμβολίζουμε με $m \div n$ το πηλίκον της ευκλείδειας διαίρεσης.

διαδικασία που περιγράψαμε. Όταν λήξει αυτή η διαδικασία αντιγράφεται ανεστραμμένη η λέξη της πέμπτης ταινίας στην τρίτη ταινία. Τότε το περιεχόμενο της τρίτης ταινίας είναι το $u \div_\Lambda w$ και της τέταρτης το $u \bmod_\Lambda w$. Παρατηρούμε ότι η διαδικασία αυτή που περιγράψαμε επαναλαμβάνεται $O(|u|)$ φορές. Επίσης στη χειρότερη περίπτωση κάθε φορά εκτελεί (**i**) έναν έλεγχο της σχέσης $>_\Lambda$ μεταξύ λέξεων μήκους $O(|w|)$, (**ii**) μία "αφαίρεση" λέξεων μήκους $O(|w|)$ και (**iii**) μία αντιγραφή λέξης μήκους πάλι $O(|w|)$. Άρα όλη η διαδικασία χρειάζεται $O(|w|)$ βήματα. Υπολογίζοντας τις επαναλήψεις προκύπτει ότι η χρονική πολυπλοκότητα της μηχανής είναι $O(|u| \cdot |w|)$. □

3.3 Σύνθεση μηχανών Turing

Πριν ξεκινήσουμε την περιγραφή σύνθετων μηχανών Turing, αναφέρουμε ότι σε κάποιες περιπτώσεις μάς είναι χρήσιμο μια λέξη που επεξεργάζεται η μηχανή να μπορεί να την κρατά κατά κάποιο τρόπο στην μνήμη της ώστε να την επεξεργαστεί και αργότερα. Εύκολα διαπιστόνουμε ότι στους προηγούμενους αλγορίθμους που δόθηκαν εφαρμώστηκε αυτή η τακτική. Για παράδειγμα η πρόσθεση γίνεται εξ ίσου εύκολα σε δύο ταινίες όπου το αποτέλεσμα θα γράφεται "πάνω" από τη δεύτερη λέξη που προσθέτουμε. Στις πιο περίπλοκες μηχανές αυτό πρέπει να γίνει με μεγαλύτερη προσοχή. Για τον παραπάνω λόγο προτιμούμε να κατασκευάζουμε μηχανές που πολλές φορές έχουν αρκετά περισσότερες ταινίες, ακριβώς για να μη δημιουργείται πρόβλημα με τη "μνήμη" της μηχανής.

3.3.1 Διακλάδωση

Πρόταση 3.3.1. *Έστω αλγόριθμος $Q(\vec{x})$ της μορφής*

$$\textbf{if} \quad \Phi \quad \textbf{then} \quad P_1 \quad \textbf{else} \quad P_2,$$

όπου το Φ είναι μια σχέση (συνθήκη), και τα P_1, P_2 δύο αλγόριθμοι. Έστω M_0 μηχανή Turing που ελέγχει τη Φ και M_1, M_2 μηχανές Turing που υλοποιούν τους αλγορίθμους P_1, P_2 αντίστοιχα. Τότε υπάρχει μηχανή Turing M ή οποία να υλοποιεί τον αλγόριθμο Q.

Απόδειξη. Έστω $M_0 = <K_0, \Sigma_0, s_0, H_0, \delta_0>$, $M_1 = <K_1, \Sigma_1, s_1, H_1, \delta_1>$ και $M_2 = < K_2, \Sigma_2, s_2, H_2, \delta_2 >$ με k_0, k_1, k_2 ταινίες αντίστοιχα. Χωρίς βλάβη της γενικότητας, μπορούμε να υποθέσουμε ότι το αλφάβητο Σ είναι κοινό και για τις τρεις μηχανές, ενώ τα σύνολα καταστάσεων K_0, K_1, K_2 είναι ξένα μεταξύ τους. Ορίζουμε μια μηχανή $M = <K, \Sigma, s, H, \delta>$, με $2 \cdot (k_0 + k_1 + k_2)$ ταινίες, όπου $s = s_0$, $H = H_1 \cup H_2$ και $K = (\cup_{i=1}^{3} K_i)$. Η ιδέα της κατασκευής είναι ότι μπορούμε να θεωρήσουμε ότι στις πρώτες k_0 ταινίες εκτελείται ο έλεγχος της Φ και οι καταστάσεις της M που αντιστοιχούν στις τελικές καταστάσεις αποδοχής και απόρριψης της M_0, οδηγούν την M στις

καταστάσεις που αντιστοιχούν στις αρχικές καταστάσεις s_1 και s_2 των M_1, M_2. Τέλος οι υπόλοιπες ταινίες υπάρχουν ώστε να μπορεί η μηχανή να κρατά στη μνήμη της τα αρχικά δεδομένα αν αυτά χρειαστεί να μεταβληθούν. □

Αν επιπλέον η M_0 τερματίζει σε χρόνο T_0, και οι M_1, M_2 σε χρόνο T_1, T_2 αντίστοιχα, με $T_0, T_1, T_2 \geq O(\|\vec{x}\|)$, τότε η M τερματίζει σε χρόνο $T_0 + \max\{T_1, T_2\}$. Πράγματι εκτελείται πάντοτε ο έλεγχος της Φ και έπειτα ένας από τους δύο αλγορίθμους. Η συνθήκη $T_0, T_1, T_2 \geq O(\|\vec{x}\|)$ χρειάζεται ώστε να εξασφαλίζεται ότι η αντιγραφή των δεδομένων διαρκεί χρόνο μικρότερης τάξης από αυτόν της εκτέλεσης των αλγορίθμων.

3.3.2 Ανακυκλώσεις και επαναλήψεις

Πρόταση 3.3.2. *Έστω αλγόριθμος $Q(\hat{r}, \vec{x})$ της μορφής*

$$\textbf{for} \quad i = 1 \quad \textbf{to} \quad r \quad \textbf{do} \quad P(\hat{i}, \hat{r}, \vec{x}),$$

όπου το $P(\hat{i}, \hat{r}, \vec{x})$ αντιπροσωπεύει έναν αλγόριθμο. Έστω επίσης μηχανή M που υπολογίζει τον αλγόριθμο $P(\hat{i}, \hat{r}, \vec{x})$. Τότε υπάρχει μηχανή Turing M' η οποία να υλοποιεί τον αλγόριθμο $Q(\hat{r}, \vec{x})$.

Απόδειξη. Έστω ότι η M έχει k ταινίες. Κατασκευάζουμε μηχανή M' η οποία θα έχει $2 \cdot k + 1$ ταινίες. Δουλεύουμε με αντίστοιχο τρόπο όπως αυτόν της προηγούμενης απόδειξης. Στην πρώτη ταινία εισάγουμε τη λέξη 1. Στις επόμενες k ταινίες αποθηκεύονται τα δεδομένα $\hat{r}, \vec{x}$ ενώ στις υπόλοιπες διεξάγεται ο υπολογισμός P και κάθε φορά που τελειώνει ο P "προσθέτουμε" μια μονάδα στη λέξη που βρίσκεται στην πρώτη ταινία και ελέγχουμε αν είναι "μεγαλύτερη" από την $\mathbf{\Lambda}(r)$. Αν αυτό συμβαίνει τότε η μηχανή μπαίνει σε τελική κατάσταση και τερματίζει τη λειτουργία της, αλλιώς εκτελεί ξανά τον υπολογισμό P. □

Σημαντική παρατήρηση: Ο χρόνος που θα χρειαστεί η M' στην γενική της μορφή για να τερματίσει μπορεί να υπολογιστεί με κάποια πολύ "χονδρικά" φράγματα τα οποία στη συνέχεια δεν μας δίδουν το επιθυμητό αποτέλεσμα. Εμείς θα εξετάσουμε μόνον περιπτώσεις όπου η M **δεν** μεταβάλλει την τιμή του r και το μήκος της εξόδου του αλγορίθμου $P(\hat{i}, \hat{r}, \vec{x})$ **φράσσεται από το μήκος** $l = \|\hat{i}, \hat{r}, \vec{x}\|$ της εισόδου του. Επιπλέον υποθέτουμε ότι η χρονική πολυπλοκότητα της M είναι $O(l^q)$. Τότε

$$\Upsilon_{M'}(\hat{r}, \vec{w}) \leq r \cdot C_M(l^q),$$

άρα η χρονική πολυπλοκότητα της M' φράσσεται από $O(r \cdot l^q)$. Αν επιπλέον $r \leq O(l^s)$ τότε η χρονική πολυπλοκότητα της M' είναι $O(l^{q+s})$, δηλαδή πολυωνυμική ως προς το μήκος της λέξης εισόδου.

Πρόταση 3.3.3. *Έστω αλγόριθμος $Q(\vec{x}, \vec{y}, \vec{z})$ της μορφής*

$$\textbf{while} \quad \Phi(\vec{x}, \vec{y}) \quad \textbf{do} \quad P(\vec{y}, \vec{z}),$$

όπου το Φ αντιπροσωπεύει τον έλεγχο μιας σχέσης και το P έναν αλγόριθμο. Έστω επίσης μηχανές Turing M_1, M_2 που υλοποιούν τους αλγορίθμους Φ και P αντίστοιχα. Τότε υπάρχει μηχανή Turing M η οποία να υλοποιεί τον αλγόριθμο Q.

Απόδειξη. Έστω ότι οι M_1, M_2 έχουν k_1, k_2 ταινίες αντίστοιχα. Κατασκευάζουμε την μηχανή M με $2 \cdot (k_1 + k_2)$ ταινίες και με αντίστοιχο τρόπο όπως στην παραπάνω περίπτωση. □

Σχόλιο: Η χρονική πολυπλοκότητα μιας τέτοιας μηχανής δεν μπορεί να υπολογιστεί a priori αλλά μόνο μελετώντας τον συγκεκριμένο αλγόριθμο που υλοποιεί. Στη γενική της μορφή μια τέτοια μηχανή μπορεί να μην τερματίζει. Όμως οι περιπτώσεις που θα εξετάσουμε θα τερματίζουν πάντοτε. Τότε, ο αλγόριθμος μπορεί με κατάλληλες μετατροπές να υλοποιηθεί και από αλγόριθμο της μορφής

$$\textbf{for} \quad i = 0 \quad \textbf{to} \quad n \quad \textbf{do} \quad Q',$$

και μάλιστα η πολυπλοκότητα **θα είναι πολωνυμική ως προς το μήκος** τής λέξης είσοδου όταν ισχύουν όσα αναφέρθηκαν στην προηγούμενη παρατήρηση. Εδώ χρειαζόμαστε **(α)** η πολυπλοκότητα της Q' να είναι πολυωνυμική, **(β)** η Q' να επαναλαμβάνεται το πολύ όσο ένα πολυώνυμο του μήκους της λέξης εισόδου και **(γ)** το μήκος της λέξης εξόδου της Q' να είναι το πολύ όσο το μήκος της λέξης εισόδου.

3.4 Ύψωση σε δύναμη

Θεώρημα 3.4.1 (Fast exponantiation method). *Έστω φυσικοί αριθμοί m, k. Μπορούμε να υπολογίσουμε το m^k κάνοντας το πολύ $2 \cdot \lfloor \log k \rfloor$ πολλαπλασιασμούς.*

Απόδειξη. Έστω $\overline{k_0 k_1 ... k_l}$ το ανεστραμμένο δυαδικό ανάπτυγμα του k

$$\begin{aligned} k &= \sum_{i=0}^{l} k_i \cdot 2^i \\ m^k &= m^{\sum_{i=0}^{l} k_i \cdot 2^i} = \prod_{i=0}^{l} m^{k_i \cdot 2^i} \end{aligned}$$

Αρχικά θα υπολογίσουμε διαδοχικά τις δυνάμεις m^{2^i} για $0 < i \leq l$, μέσω της αναδρομικής σχέσης $m^{2^{i+1}} = (m^{2^i})^2$. Στην συνέχεια, όπως φαίνεται από την παραπάνω ισότητα, αρκεί να πολλαπλασιάσουμε αυτές τις δυνάμεις που

στην αντίστοιχη θέση του δυαδικού της αναπτύγματος εμφανίζεται το ψηφίο 1, επειδή

$$m^{k_i \cdot 2^i} = \begin{cases} 1 & \text{αν} \quad k_i = 0 \\ m^{2^i} & \text{αν} \quad k_i = 1. \end{cases}$$

□

Το παραπάνω θεώρημα έχει πολύ μεγάλη ισχύ, κυρίως στην περίπτωση που υπολογίζουμε παραστάσεις της μορφής

$$m^r \mod n.$$

Το παραπάνω συμβαίνει γιατί αν κάθε φορά που υψώνουμε στο τετράγωνο το υπολογίζουμε και modulo n. Τότε ο χρόνος υπολογισμού είναι **πολυωνυμικός** ως προς m, r, n διότι το μήκος των ενδιάμεσων λέξεων δεν υπερβαίνει το μήκος των αρχικών. Σημειώνουμε ότι αυτό **δεν συμβαίνει** στη γενική περίπτωση καθότι κάθε φορά που υψώνουμε στο τετράγωνο διπλασιάζεται το μήκος της λέξης και τελικά ο υπολογισμός της δύναμης γίνεται σε εκθετικό χρόνο.

Πρόταση 3.4.2. *Υπάρχει μηχανή Turing M, που αν της δοθούν οι λέξεις $u, v, w \in N$, δίνει σαν έξοδο τη λέξη s, όπου*

$$\mathbf{A}(s) = (\mathbf{A}(u))^{\mathbf{A}(v)} \mod \mathbf{A}(\mathbf{w})$$

σε χρόνο $O((\max\{|u|, |v|, |w|\})^3)$.

Απόδειξη. **(Αλγόριθμος Square and multiply)**. Έστω $l = |u|$. Με v_i συμβολίζουμε το i-οστό γράμμα της λέξης v.

s := 1; y := u;
for i = 0 **to** l **do**
{ **if** $\mathrm{v_i} = 1$ **then do** {s := (s $\cdot_\Lambda$ y) mod_Λw};
y := y $\cdot_\Lambda$ y mod_Λw };

Το περιεχόμενο του βρόχου υλοποιείται σε χρόνο $O((|u| + |w|)^2)$ και εξάγει αποτέλεσμα που έχει μήκος που φράσσεται από το μήκος της εισόδου. Ο βρόχος εκτελείται $O(|v|)$ φορές. Συνεπώς ο παραπάνω αλγόριθμος υλοποιείται από μηχανή Turing που τρέχει σε χρόνο $O((\max\{|u|, |v|, |w|\})^3)$. □

Σχόλιο: Στην παραπάνω διαδικασία δεν διευκρινίσαμε πώς μπορούν να εντοπιστούν τα v_i για τον έλεγχο της συνθήκης $v_i = 1$ του αλγορίθμου. Το πρόβλημα εντοπισμού προκύπτει από το γεγονός ότι έως τώρα διαφορετικές μεταβλητές εισάγονταν σε διαφορετικές ταινίες, ώστε να είναι πιο εμφανής η λειτουργία της μηχανής. Θα μπορούσαμε να πούμε ότι αν είχαμε μια μηχανή Turing που να εκτελεί την εντολή "find $\mathrm{v_i}$", τότε θα μπορούσαμε να αποδείξουμε ευκολότερα ότι ο square and multiply υλοποιείται από μηχανή Turing. Εδώ όμως η κατάσταση είναι αρκετά πιο απλή. Στην αρχή της λειτουργίας της

μηχανής τοποθετούμε τον οδηγό της ταινίας που περιέχει την v στο πρώτο της γραμμα. Στη συνέχεια κάθε φορά που εκτελείται ο έλεγχος $v_i = 1$, μετακινούμε τον οδηγό μια θέση δεξιά. Η ενέργεια αυτή της μηχανής δεν απαιτεί επιπλέον χρόνο.

Παρατήρηση: Με αντίστοιχο επιχείρημα μπορεί να δειχθεί ότι η συνάρτηση $\min\{m^a, n\}$ υπολογίζεται σε πολυωνιμικό χρόνο ως προς την είσοδο, ακριβώς επειδή στην διαδικασία του fast exponentiation δεν θα υπολογιστεί ποτέ λέξη μήκους μεγαλύτερου από $\max\{m, a, 2 \cdot n\}$.

3.5 Ευκλείδειος αλγόριθμος και τετραγωνική ρίζα

Πρόταση 3.5.1. *Υπάρχει μηχανή Turing που, δοθείσης μιας λέξης $u \in N$ εξάγει την λέξη w όπου $A(w) = \lfloor \sqrt{A(u)} \rfloor$.*

Απόδειξη. Ο αλγόριθμος είναι ο παρακάτω:
$\mathrm{s} := 1; \quad \mathrm{t} := \mathrm{u}; \quad \mathrm{d} := (\mathrm{s} + \mathrm{t}) \div_\Lambda 2;$
while $\mathrm{t} - \mathrm{s} >_\Lambda 1$ **do** {**if** $\mathrm{d} \cdot_\Lambda \mathrm{d} >_\Lambda \mathrm{u}$ **then** $\{\mathrm{t} := \mathrm{d}\}$
else $\mathrm{s} := \mathrm{d}; \quad \{$**if** $\mathrm{d} \cdot_\Lambda \mathrm{d} = \mathrm{u}$ **then** break$\}; \quad \mathrm{d} := (\mathrm{s} + \mathrm{t}) \div_\Lambda 2\}$

Ουσιαστικά, η μέθοδος που ακολουθείται είναι μέθοδος διχοτόμησης, όπου κάθε φορά η ποσότητα $t - s$ υποδιπλασιάζεται. Εύκολα αποδεικνύεται ότι το πλήθος των επαναλήψεων του βρόχου είναι $O(|u|)$. Επιπλέον, αν εκτελείται ο υπολογισμός $d \cdot_\Lambda d$, μπορούμε να θεωρήσουμε ότι η λέξη αυτή δεν εξάγεται από τον αλγόριθμο (π.χ. μπορούμε να προσθέσουμε μια εντολή "σβήσε το $d \cdot_\Lambda d$"). οπότε ο βρόχος εξάγει λέξεις που έξουν μήκος το οποίο φράσσεται από το μήκος της εισόδου, και εκτελείται σε χρόνο $O(|u|^2)$. Άρα η χρονική πολυπλοκότητα του αλγορίθμου είναι $O(|u|^3)$. □

Πρόταση 3.5.2. *Υπάρχει μηχανή Turing που αν της δοθούν ως είσοδος δύο λέξεις $u, w \in N$, υπολογίζει τον "μέγιστο κοινό διαιρέτη" $(u, w)_\Lambda$ σε χρόνο $O(\min\{|u|, |w|\}^2 \cdot \max\{|u|, |w|\})$.*

Απόδειξη.

while $0 <_\Lambda \mathrm{w}$ **do** $\{\mathrm{x} := \mathrm{w}; \quad \mathrm{w} := \mathrm{u} \mod_\Lambda \mathrm{w}; \quad \mathrm{u} := \mathrm{x}\}$

Η πρώτη εκτέλεση του βρόχου χρειάζεται $O(|u| \cdot |w|)$ βήματα και εξάγει λέξεις των οποίων το μήκος φράσσεται από το $\min\{|u|, |w|\}$. Για τις επόμενες επαναλήψεις εύκολα μπορεί να δειχθεί ότι το μήκος της εξόδου του βρόχου μειώνεται τουλάχιστον κατά 1 κάθε δύο επαναλήψεις του βρόχου. Άρα χρειάζονται το πολύ $O(\min\{|u|, |w|\})$ επαναλήψεις. Επίσης ο βρόχος εκτελείται σε χρόνο το πολύ $O(\min\{|u|, |w|)$. Άρα η χρονική πολυπλοκότητα του αλγορίθμου είναι $O(\min\{|u|, |w|\}^2 \cdot \max\{|u|, |w|\})$. □

Κεφάλαιο 4

Εξειδικευμένοι αλγόριθμοι

4.1 Πυκνότητα πρώτων αριθμών

Πρόταση 4.1.1 (Nair). *Έστω* $\mathrm{lcm}(m)$ *το ελάχιστο κοινό πολλαπλάσιο των αριθμών* $1, 2, ..., m$ *και* $n \geq 7$. *Τότε*

$$\mathrm{lcm}(n) \geq 2^n.$$

Απόδειξη. Η απόδειξη που παρουσιάζουμε εδώ είναι παρόμοια με την πρωτότυπη απόδειξη του Nair [11].Κεντρικό ρόλο στην απόδειξη παίζουν οι ιδιότητες του ολοκληρώματος

$$I(m,n) = \int_0^1 x^{m-1}(1-x)^{n-m}dx, \qquad 1 \leq m \leq n.$$

Για να υπολογιστεί το ολοκλήρωμα $I(m,n)$, ουσιαστικά αρκεί να βρούμε τον συντελεστή κάθε μονωνύμου που θα ολοκληρωθεί. Υπολογίζοντας αυτούς τους συντελεστές ([19] σελ. 12) καταλήγουμε στην σχέση

$$I(m,n) = \sum_{j=0}^{n-m} (-1)^j \binom{n-m}{j} \frac{1}{m+j}.$$

Το παραπάνω άθροισμα είναι ρητός αριθμός, με παρονομαστή που διαιρεί το $\mathrm{lcm}(n)$.

Επιπλέον το $I(m,n)$ είναι "μικρό". Για να το δείξουμε αυτό παρατηρούμε αρχικά ότι η συνάρτηση $I(m,n)$ σχετίζεται με την **συνάρτηση βήτα του Euler**

$$B(s,t) = \int_0^1 x^{s-1}(1-x)^{t-1}dx, \qquad s,t \geq 0$$

μέσω της σχέσης

$$I(m,n) = B(m, n-m+1).$$

Για την **συνάρτηση βήτα** ισχύει η παρακάτω ταυτότητα [12]:

$$B(s,t) = \frac{t}{\binom{s+t-1}{s-1}}, \qquad s,t \geq 0.$$

Συνεπώς όταν $0 \leq m \leq n$ ισχύει

$$I(m,n) = B(m, n-m+1) = \frac{n-m+1}{\binom{n}{m-1}} = \frac{1}{n\binom{n-1}{m-1}} = \frac{1}{m\binom{n}{m}}$$

Από το παραπάνω προκύπτει ότι $k\binom{2k}{k} \mid \operatorname{lcm}(2k)$, άρα

$$k\binom{2k}{k} \mid \operatorname{lcm}(2k+1).$$

Επιπλέον επειδή

$$(k+1)\binom{2k+1}{k+1} = k\binom{2k+1}{k} = (2k+1)\binom{2k}{k},$$

έπεται ότι

$$(2k+1)\binom{2k}{k} \mid \operatorname{lcm}(2k+1).$$

Επειδή $(k, 2k+1) = 1$, έπεται ότι

$$k(2k+1)\binom{2k}{k} \mid \operatorname{lcm}(2k+1).$$

Τέλος, επειδή $\binom{2k}{k} \geq \frac{2^{2k}}{2k+1}$ έπεται ότι

$$\operatorname{lcm}(2k+1) \geq k2^{2k}, k \geq 1.$$

Άρα

$$\operatorname{lcm}(2k+1) \geq 2^{2k}, \qquad k \geq 2$$

και

$$\operatorname{lcm}(2k+2) \geq \operatorname{lcm}(2k+1) \geq 2^{2k+2}, k \geq 4.$$

Η περίπτωση για $n = 8$ μας δίνει $840 > 256$. Συνεπώς από τα παραπάνω, για κάθε $n \geq 7$ έπεται ότι $\operatorname{lcm}(n) > 2^n$. □

4.2 Εξειδικευμένοι αλγόριθμοι

Ορισμός 4.2.1. *Έστω r, n φυσικοί αριθμοί. Ορίζουμε ως* **τάξη του n ως προς r** *και συμβολίζουμε* $\mathrm{ord}_r n$ *τον ελάχιστο θετικό αριθμό k με την ιδιότητα*[1] $n^k \equiv 1 \mod r$.

Πρόταση 4.2.2. *Έστω φυσικοί αριθμοί n, r με $(n, r) = 1$. Υπάρχει μηχανή Turing, που αν της δωθούν οι λέξεις $u = \mathbf{\Lambda}(n), w = \mathbf{\Lambda}(r)$, δίνει ως έξοδο την λέξη z με $\mathbf{A}(z) = \mathrm{ord}_r n$.*

Απόδειξη. Ο αλγόριθμος είναι ο παρακάτω:
m := 1; x := u;
while $\mathrm{x} \neq 1 \mod_\Lambda \mathrm{w}$ **do** $\{\mathrm{x} := \mathrm{x} \cdot_\Lambda \mathrm{u} \mod_\Lambda \mathrm{w}; \quad \mathrm{m} := \mathrm{m} +_\Lambda 1\}$

Ο αλγόριθμος δεν κάνει κάτι περισσότερο από το να υπολογίζει τις διαδοχικές δυνάμεις $n^m \mod r$, μέχρι να βρεθεί αποτέλεσμα 1. Ο βρόχος εκτελείται σε χρόνο $O(|u| \cdot |w|)$, εξάγει λέξεις μήκους που φράσσονται από το $|w|$ και εκτελείται το πολύ $O(|w|)$ φορές. Άρα η χρονική πολυπλοκότητα του αλγορίθμου είναι $O(|u| \cdot |w|^2)$. □

Λήμμα 4.2.3. *Έστω n φυσικός αριθμός μεγαλύτερος της μονάδας. Υπάρχει $r \geq \max\{3, \lceil \log^5 n \rceil\}$ τέτοιο, ώστε* $\mathrm{ord}_r(n) > \log^2 n$.

Απόδειξη. Όταν $n = 2$ η απόδειξη είναι τετριμμένη: το $r = 3$ επιβεβαιώνει την πρόταση. Έστω λοιπόν $n > 2$. Τότε $\log^5 n > 10$, συνεπώς ικανοποιείται η προϋπόθεση του λήμματος (3.6.2). Έστω $B = \lceil \log^5 n \rceil$ και r ο ελάχιστος φυσικός αριθμός που δεν διαιρεί το γινόμενο

$$n^{\lfloor \log B \rfloor} \cdot \prod_{i=1}^{\lfloor \log^2 n \rfloor} (n^i - 1).$$

Το (r, n) δεν μπορεί να διαιρείται από όλους τους πρώτους παράγοντες του r, καθώς τότε το r θα διαιρούσε το $n^{\lfloor \log B \rfloor}$. Άρα ούτε ο αριθμός $\frac{r}{(r,n)}$ διαιρεί το παραπάνω γινόμενο. Επειδή ο r είναι ο ελάχιστος αριθμός με αυτήν την ιδιότητα, άρα $(r, n) = 1$. Επιπλέον, το r δεν διαιρεί κανένα από τα $n^i - 1$ για $1 \leq i \leq \lfloor \log^2 n \rfloor$. Από τα παραπάνω προκύπτει ότι $\mathrm{ord}_r(n) > \log^2 n$. Τέλος

$$n^{\lfloor \log B \rfloor} \cdot \prod_{i=1}^{\lfloor \log^2 n \rfloor} (n^i - 1) < n^{\lfloor \log B \rfloor + \frac{1}{2} \log^2 n \cdot (\log^2 n - 1)} \leq n^{\lfloor \log^4 n \rfloor} \leq 2^{\lfloor \log^5 n \rfloor} \leq 2^B$$

και από το λήμμα (3.6.2) προκύπτει ότι $r \leq B$. □

[1] Στην περίπτωση όπου $(n, r) > 1$ τέτοιος φυσικός αριθμός δεν υπάρχει οπότε ορίζουμε $\mathrm{ord}_r n = \infty$.

Πρόταση 4.2.4. *Υπάρχει μηχανή Turing, που αν της δοθεί η λέξη u, που αντιστοιχεί στο δυαδικό ανάπτυγμα του αριθμού n, (i) αν για κάθε $1 < a \leq r$ ισχύει $(a, n) = 1$, τότε δίνει ως έξοδο την λέξη $\mathbf{\Lambda}(r)$ που αντιστοιχεί στο r με την ιδιότητα $\mathrm{ord}_{\check{r}}(n) > \log^2 n$, ενώ αν υπάρχει τέοιο a, τότε η έξοδος της μηχανής είναι $COMPOSITE$.*[2] *Η χρονική πολυπλοκότητα αυτής της μηχανής είναι μικρότερη από $O(|u|^{6+\varepsilon})$, για κάθε ε θετικό.*

Η μηχανή Turing που θα περιγράψουμε είναι πρωτότυπη και εκτελεί έναν πολύ εξειδικευμένο υπολογισμό που θα μας χρησιμεύσει στην συνέχεια. Σημειώνουμε πως ο αλγόριθμος αυτός είναι ουσιαστικά επέκταση του αλγόριθμου (3.6.1).

Απόδειξη. Ο αλγόριθμος είναι ο παρακάτω:
$\mathrm{x} := \mathrm{u}; \quad \mathrm{m} := 1; \quad$ **while** $\quad \mathrm{m} \leq_\Lambda \mathbf{\Lambda}(|\mathrm{u}|^2) \quad$ **do**
$\Big\{\mathrm{m} := 1; \quad$ **while** $\quad \mathrm{x} \neq 1 \mod {}_\Lambda \mathrm{r} \quad$ **do** $\quad \big\{\mathrm{x} := \mathrm{x} \cdot_\Lambda \mathrm{u}; \quad \mathrm{m} := \mathrm{m} +_\Lambda 1$
if $\quad \mathrm{x} = \mathrm{u} \mod {}_\Lambda \mathrm{r} \quad$ **then** $\quad$ OUTPUT "COMPOSITE"$\big\}\Big\}$

Ο εσωτερικός βρόχος υπολογίζει τις διαδοχικές δυνάμεις $n^m \mod \check{r}$ μέχρι να βρεθεί αποτέλεσμα ίσο με 1 ή $n \mod \check{r}$. Στην πρώτη περίπτωση η μηχανή συνεχίζει την λειτουργία της αποθηκεύοντας το m, ενώ στη δεύτερη εξάγει αποτέλεσμα "ΣΥΝΘΕΤΟΣ". Πράγματι, αυτό συμβαίνει τότε και μόνον τότε όταν $(n, r) \neq 1$, οπότε το n δεν έχει αντίστροφο στο $\mathbb{Z}_{\check{r}}$. Ο βρόχος εκτελείται σε χρόνο $O(|u| \cdot |r|)$ και εξάγει λέξεις που φράσσονται από το μήκος του r. Το πλήθος επαναλήψεων του βρόχου είναι της τάξης $O(|r|)$ άρα συνολικά χρειάζεται $O(|u| \cdot |r|^2)$ βήματα. Συνεπώς και όλο το περιεχόμενο του εξωτερικού βρόχου χρειάζεται $O(|u| \cdot |r|^2)$ βήματα και μάλιστα εξάγει λέξεις με μήκος που φράσσεται από το $|r|$. Το πλήθος των επαναλήψεων του βρόχου λόγω του λήμματος (3.6.3) δεν ξεπερνά τις $\lceil \log n \rceil^5$. Άρα εκτελείται το πολύ $O(|u|^5)$ φορές. Και πάλι με βάση το λήμμα (3.6.3) ο χρόνος εκτέλεσης του βρόχου είναι μικρότερος από $O(|u|^{1+\varepsilon})$ για κάθε θετικό ε. Άρα ο αλγόριθμος εκτελείται σε χρόνο μικρότερο από $O(|u|^{6+\varepsilon})$ για κάθε θετικό ε. □

Πρόταση 4.2.5. *Υπάρχει μηχανή Turing που δοθείσης μια λέξης u που αντιστοιχεί στο δυαδικό ανάπτυγμα ενός φυσικού αριθμού n, μπορεί να μας πιστοποιήσει αν το n είναι δύναμη ακεραίου, σε χρόνο $O(|u|^6)$.*

Απόδειξη. Στον παρακάτω αλγόριθμο [5] γράφοντας s^t θα εννούμε την λέξη z με $A(z) = A(s)^{A(t)}$.

b:=2;
while $\quad 2^{\mathrm{b}} \leq_\Lambda \mathrm{u} \quad$ **do** $\quad \Big\{\mathrm{a} := 1; \quad \mathrm{c} := \mathrm{u};$
while $\quad 2 \leq \mathrm{c} -_\Lambda \mathrm{a} \quad$ **do** $\quad \Big\{\mathrm{m} := (\mathrm{a} +_\Lambda \mathrm{c}) \div_\Lambda 2; \quad \mathrm{p} := \min_\Lambda\{\mathrm{m}^{\mathrm{b}}, \mathrm{u} +_\Lambda 1\}$

[2] Αυτό μπορεί να γίνει προσθέτοντας μια ειδική τελική κατάσταση, όπως έχει περιγραφεί νωρίτερα.

if p=u **then** {**return** ("perfect power", m,b)}
else {**if** $\mathrm{p} < \mathrm{u}$ **then** $\{\mathrm{a} := \mathrm{m}\}$ **else** $\{\mathrm{c} := \mathrm{m}\}\}\Big\}$

$\mathrm{b} := \mathrm{b} +_\Lambda 1\Big\}$

return (no perfect power)

Από την καθαρά μαθηματική σκοπιά ο παραπάνω αλγόριθμος εκτελεί την εξής διαδικασία: Αρχικά θέτει ως εκθέτη το 2. Στη συνέχεια με την μέθοδο της διχοτόμησης εξετάζει αν υπάρχει ακέραιος που υψωμένος στο τετράγωνο να δίνει ακριβώς n. Αν αυτό δεν συμβαίνει, τότε αυξάνει τον πιθανό εκθέτη κατά ένα και εκτελεί πάλι την ίδια διαδικασία. Σημειώνουμε εδώ ότι τόσο οι πιθανοί εκθέτες, όσο και οι πιθανές βάσεις που εξετάζονται για κάθε εκθέτη είναι το πολύ $\log n$ σε πλήθος.

Από την υπολογιστική σκοπιά: Κατ' αρχάς ο παρακάτω αλγόριθμος:

$$\mathrm{p} := \min\{\mathrm{m}^{\mathrm{b}}, \mathrm{n} +_\Lambda 1\}$$

είναι υλοποιήσιμος από μηχανή Turing σε χρόνο $O(|u|^3)$, με την μέθοδο Fast Exponentiation δεδομένου ότι το p φράσσεται από το $n+1$. Ο εσωτερικός βρόχος του αλγορίθμου εκτελείται λοιπόν σε χρόνο $O(|u|^3)$ και οι επαναλήψεις που κάνει είναι $O(|u|)$. Άρα ο εσωτερικός βρόχος χρειάζεται $O(|u|^4)$ βήματα. Ο εξωτερικός βρόχος επίσης χρησιμοποιείται $O(|u|)$ φορές. Τέλος όλοι οι αλγόριθμοι δίνουν εξόδους που δεν υπερβαίνουν το μήκος της λέξης εισόδου. Συνεπώς ο αλγόριθμος υλοποιείται από μηχανή Turing με χρονική πολυπλοκότητα $O(|u|^5)$. □

Πρόταση 4.2.6. *Υπάρχει μηχανή Turing που αν της δοθούν ως είσοδος οι λέξεις $\hat{n}, \hat{r}$ και η λέξη f που αντιστοιχεί στην ακολουθία των συντελεστών ενός πολυωνύμου $f(X) \in \mathbb{Z}_n(X)/(X^r-1)$, τότε δίνει ως έξοδο την λέξη που περιγράφει τους συντελαστές του πολυωνύμου $[f(X)]^n \mod X^r - 1, n$, σε χρόνο $O(r^2 \cdot \log^3 n)$.*

Απόδειξη. Στην κατασκευή της μηχανής οι συντελεστές του πολυωνύμου θα εισάγονται σε μία ταινία παρεμβάλλοντας ενδιάμεσα το σύμβολο της δίεσης (#) και στο τέλος της λέξης θα γράφεται το σύμβολο τις δίεσης δύο διαδοχικές φορές (##). Αυτό γίνεται για να αποφύγουμε το πλήθος των ταινιών που θα χρειαζόμαστε να εξαρτάται από το r. Συνεπώς στην ταινία που εισάγονται οι συντελεστές του πολυωνύμου στο αρχικό στιγμιότυπο θα διαβάζουμε τη λέξη

$$\triangleright f_0 \# f_1 \# ... \# f_{r-2} \# f_{r-1} \# \#.$$

Επιπλέον, δεδομενου ότι οι συντελεστές των πολυωνύμων υπολογίζονται modulo n, οι αντίστοιχες λέξεις θα έχουν μήκος που φράσσεται από το $|u|$. Εμείς όμως όλους τους συντελεστές θα τους θεωρούμε λέξεις μήκους **ακριβώς** $|u|$, ώστε να αποφύγουμε το ενδεχόμενο κατά την διάρκεια των υπολογισμών να

χρειαστεί να μετακινηθεί όλο το περιεχόμενο μίας ταινίας δεξιά επειδή κάποια λέξη δεν "χορούσε" αναμεσα από τις δύο διέσεις (#) που της αντιστοιχούσαν.

Η διαδικασία που ακολουθείται είναι αντίστοιχη με αυτήν του αλγορίθμου Square and Multiply. Στον αλγόριθμο, όπως εμφανίζεται παρακάτω, οι δείκτες στις μεταβλητές που εκφράζουν πολυώνυμα θα παίζουν το ρόλο της διάκρισης μεταξύ των συντελεστών των πολυωνύμων, ενώ στο u παίζουν τον ρόλο της διάκρισης των ψηφίων του.

$\mathrm{x} := \mathrm{f}; \quad \mathrm{y} := \mathrm{f}; \quad \mathrm{z} := 0; \quad \mathrm{v} := 0;$
for $k = 0$ **to** $|u|$ **do**
$\Big\{$**for** $i = 0$ **to** $r - 1$ **do**
$\Big\{$**for** $j = 0$ **to** $r - 1$ **do**
$\{\mathrm{z}_{(\mathrm{i+j}) \mod \mathrm{r}} := \mathrm{z}_{(\mathrm{i+j}) \mod \mathrm{r}} + \mathrm{x_i} \cdot \mathrm{y_j}\}\Big\}$;
if $\hat{n}_k = 1$ **then**
$\Big\{$**for** $i = 0$ **to** $r - 1$ **do** $\{\mathrm{v_i} := \mathrm{v_i} + \mathrm{z_i}\}\Big\}$;
for $i := 0$ **to** $r - 1$ **do** $\{x_i := z_i\}$;
for $j = 0$ **to** $r - 1$ **do** $\{y_i := z_i\}\Big\}$

Κατ' αρχάς θα υπάρχουν 5 ταινίες που το περιεχόμενό τους θα είναι ακολουθία συντελεστών πολυωνύνων, με τη μορφή που περιγράφτηκε πριν. Η πρώτη ταινία θα είναι αυτή της εισόδου, οι επόμενες τρεις που χρησιμοποιούνται βοηθητικά και η πέμπτη που χρησιμοποιείται για να μας δώσει την έξοδο. Στις τρεις βοηθητικές ταινίες γίνονται οι πράξεις που απαιτούνται για να προσδιοριστούν τα $(f)^{2^i}$, όπως στο αλγόριθμο Square and Multiply. Οι υπόλοιπες ταινίες χρησιμοποιούνται για την διεξαγωγή των υπολοίπων βοηθητικών υπολογισμών.

Ένα ερώτημα που προκύπτει εδώ είναι το πώς μπορεί η μηχανή να αναγνωρίζει κάθε φορά το i-οστό ψηφίο του u ή τον j-οστό συντελεστή του f. Το πρώτο γίνεται ακριβώς όπως και στον αλγόριθμο Square and Multiply. Το δεύτερο είναι κάπως πιο περίπλοκο και αποτελεί συνδυασμό της προηγούμενης μεθόδου και του τρόπου που δομήσαμε την μηχανή που κάνει τον απλό πολλαπλασιασμό δυαδικών αναπτυγμάτων. Αυτό που χρειάζεται να παρατηρήσουμε είναι ότι σε κάθε βήμα που χρειάζεται εύρεση κάποιου j-οστού συντελεστή, ο οδηγός θα διαβάζει τον $(j-1)$-οστό συντελεστή πολυωνύμου. Οπότε το ζητούμενο είναι ο οδηγός της αντίστοιχης ταινίας να μετακινηθεί μία δίεση (#) δεξιά. Εξαίρεση αποτελεί η περίπτωση $j = \breve{w}$. Τότε ο οδηγός μπορεί να βρίσκεται στην θέση $\breve{w}-1$ και θα πρέπει να μετακινηθεί στην αρχή. Αυτό όμως είναι εύκολο να γίνει. Αν περνώντας δεξιά της επόμενης δίεσης (#) διαβάσει πάλι δίεση (#), τότε στη συνέχεια μετακινείται αριστερά μέχρι να αναγνώσει το σύμβολο $\triangleright$.

Τώρα μπορούμε να περάσουμε στην ανάλυση της πολυπλοκότητας. Ο εσωτερικός βρόχος χρειάζεται $O(|u|^2)$ βήματα. Οι τρεις βρόχοι, από τον εξωτερικό

προς τον εσωτερικό, εκτελούνται $|u| = \log n$, r και r φορές αντίστοιχα. Εύκολα παρατηρούμε ότι οι υπόλοιπες "πράξεις" που εκτελεί ο αλγόριθμος χρειάζονται λιγότερο χρόνο από τον υπολογισμό του εσωτερικού βρόχου. Συνολικά λοιπόν η πολυπλοκότητα της μηχανής Turing είναι $O(r^2 \cdot \log^3 n)$. Σημειώνουμε ότι σε όλες τις παραπάνω περιπτώσεις, ο εντοπισμός του ζητούμενου συντελεστή ενός πολυωνύμου διαρκεί λιγότερο από όσο διαρκούν οι υπόλοιπες κινήσεις που γίνονται. □

Κεφάλαιο 5

Πιστοποίηση πρώτων αριθμών

5.1 Το Θεώρημα AKS

Σχόλιο: Έστω $P(X), Q(X) \in \mathbb{Z}_n(X)$. Στην συνέχεια θα λέμε ότι

$$P(X) = Q(X) \qquad \mod (n, X^r - 1),$$

αν τα $P(X)$ και $Q(X)$ αντιστοιχίζονται στο ίδιο στοιχείο του $\mathbb{Z}_n(X)/(X^r - 1)$.

Θεώρημα 5.1.1 (AKS). *Έστω $n, r \in \mathbb{N}$, με $n \geq r$. Αν για κάθε a με $2 \leq a \leq r$ ισχύει $a \not| n$, $\mathrm{ord}_r(n) > \log^2 n$ και για όλα τα a με $1 \leq a \leq \lfloor \sqrt{r} \log n \rfloor$ ισχύει*

$$(X + a)^n = X^n + a \mod (n, X^r - 1),$$

τότε ο n είναι δύναμη πρώτου.

Απόδειξη. Έστω n, r που ικανοποιούν τις προϋποθέσεις του Θεωρήματος AKS. Αρχικά κάνουμε κάποιες παρατηρήσεις και δίνουμε κάποιους ορισμούς:

Αν $\mathrm{ord}_r(n) > 1$, τότε υπάρχει p τέτοιο, ώστε $p|n$ και $\mathrm{ord}_r(p) > 1$. Επίσης αν για $2 \leq a \leq r$ ισχύει $a \not| n$, τότε $p > r$. Τέλος, αφού $(n, r) = 1$, άρα και $(p, r) = 1$. Άρα $n, p \in \mathbb{Z}_r^*$. Συμβολίζουμε $l = \lfloor \sqrt{\phi(r)} \log n \rfloor$. Έστω $Q_r(X)$ το r-οστό κυκλοτομικό πολυώνυμο υπέρ του $\mathbb{Z}_p$. Το $Q_r(X)$ διαιρεί το $X^r - 1$. Επιπλέον το "παραγοντοποιεί" σε ανάγωγους παράγοντες βαθμού $\mathrm{ord}_r(p)$. Έστω $h(X)$ ένας τέτοιος παράγοντας. Το $h(X)$ είναι μονικό ανάγωγο πολυώνυμο, άρα το $F = \mathbb{Z}_p[X]/h(X)$ είναι σώμα.

- Ορίζουμε τα σύνολα

$$I = \{(\frac{n}{p})^i \cdot p^j | i, j \geq 0\} \quad \text{και} \quad P = \{\prod_{a=o}^{l} (X + a)^{\beta_a} | \beta_a \geq 0\}.$$

- Έστω πολυώνυμο $f(X) \in \mathbb{Z}_p(X)/(X^r - 1)$ και $m \in \mathbb{Z}$. Θα λέμε ότι το m είναι ενδοσκοπικό για το πολυώνυμο $f(X)$, άν

$$[f(X)]^m = f(X^m) \qquad \mod (p, X^r - 1).$$

- Ορίζουμε το σύνολο T που περιέχει τα υπόλοιπα όλων των στοιχείων του I, modulo r και το σύνολο G που περιέχει τα υπόλοιπα όλων των πολυωνύμων που ανήκουν στο P, $\mod h(X), p$.

Το T σχηματίζει ομάδα που παράγεται από τα $\frac{n}{p}, p$. Επιπλέον $n \in T$ και $ord_r n > \log^2 n$, άρα $|T| = t > \log^2 n$. Το G είναι προφανώς υποσύνολο του F και εύκολα μπορεί να δειχθεί ότι είναι υποομάδα της F^*.

Η απόδειξη του θεωρήματος θα γίνει σε τρία βασικά βήματα.

Βήμα 1ο: Κάθε στοιχείο του I είναι ενδοσκοπικό για κάθε πολυώνυμο στο P.
Βήμα 2ο: $|G| > n^{\sqrt{t}}$
Βήμα 3ο: Άν το n δεν είναι δύναμη πρώτου, τότε $|G| \leq n^{\sqrt{t}}$.

Βήμα 1ο: Κάθε στοιχείο του I είναι ενδοσκοπικό για κάθε πολυώνυμο στο P.

Από τις υποθέσεις που έχουμε για τα n, r έχουμε ότι

$$\text{για} \quad 1 \leq a \leq l \quad \text{ισχύει} \quad (X+a)^n = X^n + a \mod (n, X^r - 1),$$

άρα

$$\text{για} \quad 1 \leq a \leq l \quad \text{ισχύει} \quad (X+a)^n = X^n + a \mod (p, X^r - 1).$$

Επίσης, επειδή το $\mathbb{Z}_p$ έχει χαρακτηριστική p, έπεται ότι

$$\text{για} \quad 1 \leq a \leq l \quad \text{ισχύει} \quad (X+a)^p = X^p + a \mod (p, X^r - 1).$$

Από τις δύο παραπάνω σχέσεις, εφαρμόζοντας το μικρό θεώρημα του Fermat, προκύπτει ότι

$$\text{για} \quad 1 \leq a \leq l \quad \text{ισχύει} \quad (X+a)^{n/p} = X^{n/p} + a \mod (p, X^r - 1).$$

Παρατηρούμε ότι οι αριθμοί $n, p, n/p$ είναι ενδοσκοπικοί για τα πολυώνυμα $P_a(X) = X - a$ $(1 \leq a \leq l)$.

Λήμμα 5.1.2. *Έστω $m, n \in \mathbb{N}$ και $f(X) \in \mathbb{Z}_p(X)/(X^r - 1)$. Άν τα m, n είναι ενδοσκοπικά για το $f(X)$, τότε και το $m \cdot n$ είναι ενδοσκοπικό για το $f(X)$.*

Απόδειξη.

$$\begin{aligned} [f(X)]^{m \cdot n} &= [f(X^m)]^n \qquad \mod (p, X^r - 1) \\ [f(X^m)]^n &= f(X^{m \cdot n}) \qquad \mod (p, X^{m \cdot r} - 1) \\ &= f(X^{m \cdot n}) \qquad \mod (p, X^r - 1) \end{aligned}$$

□

Λήμμα 5.1.3. *Άν το m είναι ενδοσκοπικό για τα πολυώνυμα $f(X), g(X) \in \mathbb{Z}_p(X)/(X^r-1)$, τότε είναι ενδοσκοπικό και για το $f(X)\cdot g(X)$.*

Απόδειξη.

$$[f(X)\cdot g(X)]^m = [f(X)]^m \cdot [g(X)]^m = f(X)^m \cdot g(X)^m \qquad \mod (p, X^r-1).$$

□

Από τα παραπάνω λήμματα και από το γεγονός ότι οι αριθμοί p και n/p είναι ενδοσκοπικοί για τα πολυώνυμα $P_a(X) = X - a$ $(1 \le a \le l)$, προκύπτει ότι κάθε αριθμός στο σύνολο I είναι ενδοσκοπικός για κάθε πολυώνυμο από το σύνολο P.

Βήμα 2ο: $|G| > n^{\sqrt{t}}$.

Αρχικά θα δείξουμε ότι δύο διαφορετικά πολυώνυμα βαθμού μικρότερου από t στο P αντιστοιχίζονται σε διαφορετικά στοιχεία της G. Έστω $f(X)$ και $g(X)$ δύο τέτοια πολυώνυμα και ας υποθέσουμε ότι

$$f(X) = g(X) \mod (p, h(X)).$$

Έστω ακόμα $m \in T$. Αφού $f(X) = g(X) \mod (p, h(X))$, άρα

$$[f(X)]^m = [g(X)]^m \qquad \mod (p, h(X))$$

και επειδή $m \in T$ έπεται ότι

$$f(X^m) = g(X^m) \qquad \mod (p, h(X)).$$

Έστω το πολυώνυμο $R(X) = f(X) - g(X)$. Το $R(X)$ έχει βαθμό $\deg R < t$. Επιπλέον το X^m είναι ρίζα του πολυωνύμου για κάθε $m \in T$. Επειδή $m \in T$ έπεται $(m, r) = 1$. Αυτό σημαίνει ότι το X^m είναι πρωταρχική r-οστή ρίζα της μονάδας. Άρα το $R(X)$ θα έχει t διαφορετικές ρίζες στο F. Αυτό όμως είναι άτοπο επειδή $\deg R < t$.

Τώρα παρατηρούμε ότι τα πολυώνυμα $X, X+1, ..., X+l$ είναι διαφορετικά ανά δύο διότι

$$l = \sqrt{r}\log n < r < p.$$

Επίσης επειδή $\mathrm{ord}_r p > 1$, έπεται $\deg h > 1$ άρα $X + a \neq 0 \mod h(X), p$. Άρα υπάρχουν τουλάχιστον $l+1$ διακριτά πολυώνυμα βαθμού 1 στην G. Άρα υπάρχουν τουλάχιστον $\binom{t+l}{t-1}$ διακριτά πολυώνυμα βαθμού μικρότερου από

t. Συνεπώς

$$\begin{aligned} |G| &\geq \binom{t+l}{t-1} \\ &\geq \binom{1+\lfloor\sqrt{t}\log n\rfloor+l}{\lfloor\sqrt{t}\log n\rfloor} \\ &\geq \binom{2\lfloor\sqrt{t}\log n\rfloor+1}{\lfloor\sqrt{t}\log n\rfloor} \\ &> 2^{\lfloor\sqrt{t}\log n\rfloor+1} \\ &\geq n^{\sqrt{t}} \end{aligned}$$

Βήμα 3ο: Άν το n δεν είναι δύναμη πρώτου, τότε $|G| \leq n^{\sqrt{t}}$.

Έστω $J = \left\{ \left(\frac{n}{p}\right)^i \cdot p^j \mid 0 \leq i,j \leq \lfloor\sqrt{t}\rfloor \right\}$. Προφανώς $J \subseteq I$. Άν το n δεν είναι δύναμη του p, τότε το J έχει $(\lfloor\sqrt{t}\rfloor+1)^2 > t$ στοιχεία. Άρα τουλάχιστον δύο στοιχεία του J είναι ίσα modulo r. Έστω m,n τα στοιχεία αυτά, με $m > n$. Έχουμε λοιπόν $m = n \mod r$, άρα $X^m = X^n \mod X^r-1$. Έστω $f(X) \in P$. Τότε

$$\begin{aligned} [f(X)]^m &= f(X^m) \quad \mod X^r-1, p \\ &= f(X^n) \quad \mod X^r-1, p \\ &= [f(X)]^n \quad \mod X^r-1, p. \end{aligned}$$

Άρα το $f(X)$ είναι ρίζα του πολυωνύμου $S(X) = X^m - X^n$ στο F. Το $f(X)$ επιλέχθηκε τυχαία, άρα το $S(X)$ έχει τουλάχιστον $|G|$ διακεκριμένες ρίζες στο F. Αν ο βαθμός του $S(X)$ είναι k, άρα

$$|G| \leq k \leq \left(\frac{n}{p}\cdot p\right)^{\lfloor\sqrt{t}\rfloor} \leq n^{\lfloor\sqrt{t}\rfloor}.$$

Από τα παραπάνω λήμματα προκύπτει άμεσα το Θεώρημα AKS. □

5.2 Ο αλγόριθμος AKS

Θεώρημα 5.2.1. *Υπάρχει μηχανή Turing, που αν της δοθεί η λέξη* $\mathbf{\Lambda}(n)$*, πιστοποιεί σε πολυωνυμικό χρόνο αν ο αριθμός* n *είναι πρώτος ή σύνθετος.*

Απόδειξη. **1** **if** το u είναι γνήσια δύναμη ακεραίου **then** {OUTPUT "COMPOSITE"};

2 Βρες το μικρότερο r τέτοιο ώστε $\mathrm{ord}_r(n) > |u|^2$;

3 **for** $a = 1$ **to** r **do**
$\{$**if** $1 < (a, \check{u}) < \check{u}$ **then** {OUTPUT "COMPOSITE"}$\}$;
4 **if** $u \leq \hat{r}$ **then** {OUTPUT "PRIME"};
5 **for** $a = 1$ **to** $\lfloor \sqrt{r} \rfloor |u|$ **do**
$\{$**if** $(X + a)^{\check{u}} \neq X^{\check{u}} + a \mod X^r - 1, \check{u}$ **then**
{OUTPUT "PRIME"}$\}$;
6 OUTPUT "PRIME";

Ο παραπάνω αλγόριθμος είναι παραλλαγή αυτού που δημοσιεύτητκε από τους M. Agrawal, N. Kayal και N. Saxena το 2003 [4]. Οι τροποποιήσεις που έχουν γίνει σε αυτήν την εργασία, εκτός φυσικά από την υλοποίησή του σε μηχανές Turing, είναι ελάχιστες, με σημαντικότερη ίσως τον αλγόριθμο που δόθηκε στο (3.6.4) που μπορεί να αντικαταστήσει τις γραμμές **2** και **3** αυτού του αλγορίθμου.

Ορθότητα του αλγορίθμου: Ο παραπάνω αλγόριθμος πράγματι αποφασίζει αν ο $\check{u}$ είναι πρώτος ή όχι. Έστω ότι ο $\check{u}$ είναι πρώτος. Τότε κανένας από τους ελέγχους που όταν ικανοποιούνται δίνουν έξοδο "COMPOSITE" δεν μπορεί να ικανοποιείται. Άρα αν ο $\check{u}$ είναι πρώτος τότε ο αλγόριθμος δίνει έξοδο "PRIME". Έστω ότι ο $\check{u}$ είναι σύνθετος. Τότε, αν ο αλγόριθμος στους πρώτους ελέγχους δεν τον αναγνωρίσει ως σύνθετο, λόγω του Θεωρήματος AKS θα περάσει τον έλεγχο **5** και ο αλγόριθμος θα τον αναγνωρίσει στον τελικό έλεγχο. Άρα αν ο $\check{u}$ είναι σύνθετος ο αλγόριθμος επιστρέφει "COMPOSITE".

Πολλυπλοκότητα του αλγορίθμου: Κάθε επιμέρους διαδικασία του αλγορίθμου έχουμε δει ότι υλοποιείται από μηχανή Turing σε πολυωνυμικό χρόνο. Συγκεκριμμένα ο έλεγχος **1** εκτελείται σε χρόνο $O(|u|^6)$ (πρόταση 3.6.5). Αντί των **2** και **3** μπορούμε να υλοποιήσουμε τον αλγόριθμο (3.6.4) που εκτελείται ε χρόνο $O(|u|^{6.1})$. Το r που προκύπτει είναι μικρότερο από $\lceil n \rceil = O(|u|^5)$. Ο έλεγχος της $n \geq r$ γίνεται σε χρόνο μικρότερο από $O(|u|)$. Τέλος ο κύριος βρόχος, ο έλεγχος **5** του αλγορίθμου, υλοποιείται από μηχανή Turing σε χρόνο $O(r^{1/2} \cdot |u| \cdot r^2 \cdot |u|^3) = O(|u|^{16.5})$. Άρα και η συνολική πολυπλοκότητα του αλγορίθμου AKS είναι $O(|u|^{16.5})$, δηλαδή **πολυωνυμική** ως προς το μήκος της λέξης εισόδου. □

Βιβλιογραφία

1. Adleman L, Pomerance C, Rumely RS. On distingushing Prime Numbers From Composite Numbers, *Annals of Mathematics* 117, 173-206, 1983

2. Adleman L, Huang MA. Primality Testing And Two Dimensional Abelian Varieties Over Finite Fields, *Springer Verlag* 1992

3. Agrawal M, Kayal N, Saxena N. Primality and Identity Testing via Chinese Remaindering,
unpublished (available at: http://www.cse.iitk.ac.in/users/manindra/algebra/primality_original.pdf)

4. Agrawal M, Kayal N, Saxena N. Primes is in P, *Annals of Mathemaics* 160 2004 no.2 781-793

5. Dietzfelbinger M. *Primality Testing in Polynomial Time,* Springer 2004

6. Goldwasser S, Kilian J. Almost all primes can be quickly certified, *Proc. Annual ACM Symposium on the Theory of Computing*, 316-329, 1986.

7. Hennie F. *Introduction to Computability,* MIT 1977

8. Lewis HR, Papadimitriou CH. *Elements of the theory of Computation,* Prantice Hall 1998

9. Μαγιολαδίτης Μ. Πιστοποίηση Πρώτων Αριθμών, ανέκδοτο
(διαθέσιμο: http://www.math.uoc.gr/ marios/primes.pdf)

10. Miller G. Riemann Hypothesis and test for Primality,*JCSS13* 1976, 300-317

11. M. Nair, On Chebyshev-type inequalities for primes, *Amer. Math. Monthly* 89 (1982), 126-129.

12. Persides S. Mathematical Handbook Part A Ver. 1 ESPI, Athens 2007

13. Pomerance C. Recent developments in primality testing, *The Mathematical Intelligencer*3 1981 no.3 97-105

14. Pratt V. Every Prime has a Succinct Certificate,*SIAM J of Comp* 1975, 214-220

15. Rabin M. Propabilistic algorithms for primality testing, *J of Number Theory* 12 1980 128-138

16. Solovay M, Strassen V. A fast Monte-Carlo Test for Primality,*SIAM J of Comp* 1977 84-85

17. Συγκελάκης Α. Ισοδυναμία μηχανών Turing, ανέκδοτο (διαθέσιμο: http://math.uoc.gr/ags/turing.pdf)

18. Tenenbaum G. *Introduction to analytic and probabilistic mumber theory,* Cambridge Univercity Press 1995