Πτυχιακή Εργασία

ΔΥΣΚΟΛΙΑ vs. ΤΥΧΑΙΟΤΗΤΑ

Στασινός Παναγιώτης Α.Μ. 2007166

Τμήμα Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών

Παράρτημα Σπάρτης

Επιβλέπων καθηγητής : Δρ. Καραγιώργος Γρηγόρης

-2011-

Προς τον προϊστάμενο και τους καθηγητές του τμήματος Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών - Παράρτημα Σπάρτης

Μετά από τέσσερα χρόνια φοίτησης στο τμήμα σας αισθάνομαι την ανάγκη να σας ευχαριστήσω για την υπεύθυνη εργασία σας προκειμένου τώρα που λαμβάνω το πτυχίο μου να νιώθω άρτια καταρτισμένος για να μπορώ να ανταποκριθώ στις απαιτήσεις της αγοράς εργασίας. Η συνεργασία, η απαιτητική προσφορά, η συστηματική κατάρτιση και το πνεύμα υπευθυνότητας ήταν στοιχεία που ενεργοποίησαν και τη δική μου θέληση για την κατά το δυνατόν άριστη κατάρτισή μου.

Θα ήταν παράληψη να μην επισημάνω πράγματα τα οποία ενδεχόμενα να θεωρούνται αυτονόητα και πολύ μικρά που όμως αποτελούν βασικά στοιχεία που θεμελιώνουν και χτίζουν βήμα βήμα τη σχέση του φοιτητή προς το ίδρυμα και το καθηγητικό προσωπικό. Συγκεκριμένα η εκ μέρους του προϊσταμένου του τμήματος καθιέρωση έγκαιρης κατάρτισης προγράμματος εξεταστικής, ανακοινώσεων της σχολής, άμεσης διανομής συγγραμμάτων, η οργάνωση πληροφοριών και γραμματειακής υποστήριξης για την εξυπηρέτηση των φοιτητών και η λειτουργικότητα της βιβλιοθήκης είναι μερικά από αυτά.

Υπερβαίνοντας τον καθωσπρεπισμό επειδή η παράληψη θα ήταν άδικη τολμώ έχοντας περάσει και από άλλο πανεπιστημιακό ίδρυμα και έχοντας συνεργαστεί επί δύο έτη για την εκπόνηση της πτυχιακής αυτής εργασίας, να μην επισημάνω την μεγάλη προσφορά του προϊσταμένου του τμήματος κυρίου Γρηγορίου Καραγιώργου ο οποίος με πατρική στοργή, επιστημονική καθοδήγηση και αγάπη προς τους αυριανούς επιστήμονες καταξιώνει με την παρουσία του το ίδρυμα.

Για την συμβολή του αυτή σε ένδειξη ευγνωμοσύνης του αφιερώνω την πτυχιακή μου εργασία. Όχι επιφανειακά αλλά με ουσία. Απόδειξη ότι για να κάνει δεκτή την πτυχιακή μου με ανάγκασε με τον τρόπο του να προσφύγω στην πολυάριθμη βιβλιογραφία , που μου υπέδειξε , για να με καταστήσει κατά το δυνατόν καλύτερο γνώστη της πληροφορικής επιστήμης.Επίσης τον ευχαριστώ γιατί μαζί με τον κύριο Μπάρδη Γεώργιο με ενθάρρυναν για μεταπτυχιακές σπουδές στην πληροφορική και με τις συστατικές τους επιστολές συνέβαλαν στο να γίνω δεκτός σε μεταπτυχιακό πρόγραμμα.

Ευχαριστώ επίσης τους κυρίους Πανάγο Νικόλαο , Μπάρδη Γεώργιο και Λιαπέρδο Ιωάννη γιατί με τη συμβολή τους τιμούν και καταξιώνουν το ίδρυμα και το έχουν κάνει δημιουργό πραγματικά καταρτισμένων επιστημόνων.

Περιεχόμενα

1 ΠΕΡΙΛΗΨΗ - ΕΙΣΑΓΩΓΗ **5**

1.1 Λέξεις Κλειδιά 6

2 Θεωρία Υπολογισμού **7**

2.1 Αλφάβητα και γλώσσες 7

2.2 Αυτόματα και Turing μηχανές 9

2.3 Οι κλάσεις P και BPP 14

3 Τυχαιότητα **19**

3.1 Kurt Gödel - Θεώρημα Μη Πληρότητας 19

3.2 David Hilbert- Φορμαλισμός των Μαθηματικών 20

3.3 Alan Turing- Το Πρόβλημα Τερματισμού 22

3.4 Τυπικό Αξιωματικό Σύστημα 23

3.5 Αδυναμία-ορισμού-της-τυχαιότητας του Borel 26

4 Ψευδοτυχαία bits και ακολουθίες **28**

4.1 Ορισμοί τυχαίων-ψευδοτυχαίων γεννητόρων 28

4.2 Απαιτήσεις ασφάλειας για τις γεννήτριες 29

4.3 Δημιουργία τυχαίων bit 32

4.4 Δημιουργία ψευδοτυχαίων bit 35

4.5 Στατιστικά tests 35

4.6 Κρυπτογραφικά ασφαλείς ψευδογεννήτριες bit 40

5 Δυσκολία vs. Τυχαιότητα **41**

5.1 Εισαγωγή 41

5.2 Το εγχείρημα των Nisan-Wigderson 45

5.3 Η δυσκολία που απαιτείται για τον γεννήτορα 45

5.4 Διαφορά σε σχέση με τις one-way functions 47

5.5 Βασικό Συμπέρασμα 48

1 ΠΕΡΙΛΗΨΗ - ΕΙΣΑΓΩΓΗ

Η σχέση μεταξύ των διαφορετικών τρόπων υπολογισμού όπως ντετερμινιστικά, μη ντετερμινιστικά ή τυχαιοποιημένα είναι ένα κεντρικό θέμα της θεωρίας της υπολογιστικής πολυπλοκότητας . Ένα πολύ σημαντικό πρόβλημα είναι η σχέση μεταξύ τυχαιοποιημένου και ντετερμινιστικού πολυωνυμικού χρόνου , όπου από τεχνικής άποψης αυτό είναι το ερώτημα εάν P=BPP .

Αρκετοί θεωρητικοί της πολυπλοκότητας φαίνεται να πιστεύουν πως οι κλάσεις P και BPP είναι πράγματι ίσες και αυτή τους η πεποίθηση ενισχύεται από τα βαθιά αποτελέσματα των Naom Nisan και Avi Wigderson οι οποίοι συνέδεσαν το ζήτημα derandomization της κλάσης BPP με την ύπαρξη hard συναρτήσεων[1]. Στόχος της εργασίας αυτής είναι να μελετήσει θεωρητικά χωρίς θεωρήματα , αποδείξεις και αυστηρούς ορισμούς τα αποτελέσματα αυτά των Nisan και Wigderson .

Αρχικά γίνεται μια εισαγωγή σε κάποιες βασικές έννοιες της θεωρίας υπολογισμού τις οποίες θα χρησιμοποιήσουμε αρκετές φορές . Ακολουθεί μια μικρή ιστορική αναδρομή από τα μαθηματικά μέχρι να καταλήξουμε στο παράδοξο αδυναμίας ορισμού της τυχαιότητας . Στην επόμενη ενότητα δίνονται οι ορισμοί των τυχαίων και ψευδοτυχαίων γεννητόρων καθώς και κάποιες απαιτήσεις τις οποίες οφείλουν να πληρούν .

Τέλος, θα γίνει αναφορά στην εργασία των Nisan-Wigderson, "Hardness vs. Randomness "όπου θα μιλήσουμε για τον γεννήτορα τον οποίο κατασκεύασαν , την δυσκολία που απαίτησαν να έχει ο γεννήτορας αυτός καθώς και τα συμπεράσματά τους σχετικά με την αποτελεσματική ντετερμινιστική προσομοίωση των τυχαιοποιημένων αλγορίθμων όπου θα μπορούσε να επιτευχθεί (υπό κάποιες συνθήκες) με την βοήθεια του γεννήτορα αυτού.

1.1 Λέξεις Κλειδιά

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ : γλώσσα αναγνωρίζεται από αυτόματα - Turing μηχανή , οι κλάσεις P και BPP , τυχαίος και ψευδοτυχαίος γεννήτορας , τυχαιότητα , προσεγγιστικός αλγόριθμος , πιθανοκρατικός αλγόριθμος , derandomization

2 Θεωρία Υπολογισμού

[2] Τι είναι αυτό που κάνει κάποια προβλήματα υπολογιστικώς δύσκολα και κάποια άλλα εύκολα; Αυτό είναι το κεντρικό ερώτημα της θεωρίας πολυπλοκότητας . Είναι εντυπωσιακό ότι , παρά την εντατική σχετική έρευνα που έχει πραγματοποιηθεί από το 1965 (περίπου) και μετά , η απάντηση εξακολουθεί να μας είναι άγνωστη . Οι θεωρίες της υπολογισιμότητας και της πολυπλοκότητας συνδέονται στενά μεταξύ τους .

Στη θεωρία υπολογισιμότητας , ο στόχος είναι να χαρακτηρίσουμε τα διάφορα προβλήματα ως επιλύσιμα ή ανεπίλυτα , ενώ η θεωρία πολυπλοκότητας στοχεύει στην ταξινόμηση των επιλύσιμων προβλημάτων σε εύκολα και δύσκολα. Επιπλέον, αρκετές από τις έννοιες που χρησιμοποιούνται στη θεωρία πολυπλοκότητας εισάγωνται από τη θεωρία υπολογισιμότητας. Θα δώσουμε στην ενότητα αυτή αρκετούς βασικούς ορισμούς τους οποίους θα χρησιμοποιήσουμε πολλές φορές στις επόμενες ενότητες.

2.1 Αλφάβητα και γλώσσες

Χρειάζεται λοιπόν στην υποενότητα αυτή να ορίσω δύο βασικές έννοιες , το αλφάβητο και τις γλώσσες[3].

Ας είναι Χ ένα πεπερασμένο σύνολο, το αλφάβητό μας. Τα στοιχεία θα τα ονομάζουμε γράμματα. Αν γράψουμε κάποια γράμματα του Χ το ένα δίπλα στο άλλο σχηματίζουμε μια λέξη από το Χ.

Ας πάρουμε για παράδειγμα το ελληνικό αλφάβητο

$$X=\{ \alpha , \ldots , \omega \}$$

Τότε, οι ψωμι , πεπονι , υπονομος είναι μερικές λέξεις που μπορούμε να σχηματίσουμε. Επίσης, αν πάρουμε το αλφάβητο του Morse X= { $^{-}$, $\cdot$, $|$} τότε τα σήματα Morse δεν είναι παρά λέξεις από αυτό.

Γενικά λοιπόν, αν X είναι ένα οποιοδήποτε **αλφάβητο** , μια λέξη από το X είναι μια πεπερασμένη ακολουθία γραμμάτων του X. Είναι απαραίτητο βέβαια να εισάγουμε την κενή λέξη, δηλαδή τη λέξη χωρίς κανένα γράμμα! Τη λέξη αυτή τη συμβολίζουμε με το ελληνικό γράμμα ε.

Ας συμβολίσουμε τώρα το X^* το σύνολο όλων των λέξεων που μπορούμε να σχηματίσουμε από το αλφάβητο X. Είναι, δηλαδή,

$$X^* = \{ \varepsilon \} \cup \{ x_1 \ldots x_n \, | x_k \in \mathrm{X} \text{ , k=1,2, ... } \}$$

Το μήκος μιας λέξης w το συμβολίζουμε |w |και δεν είναι παρά το πλήθος των γραμμάτων που την απαρτίζουν.

Ας είναι X ένα αλφάβητο. Ένα οποιοδήποτε υποσύνολο του X^* ονομάζεται **γλώσσα** και συμβολίζεται συνήθως με L .

2.2 Αυτόματα και Turing μηχανές

Πρέπει να ξεκαθαρίσουμε από την αρχή ότι το ενδιαφέρον μας για τις διάφορες μηχανές, που θα ορίσουμε , δεν εντοπίζεται στην τεχνολογία τους, αλλά μόνο στις ιδιότητές τους. Με άλλα λόγια δεν μας ενδιαφέρει πώς είναι για παράδειγμα κατασκευασμένο ένα αυτόματο αλλά το τι μπορεί να κάνει.

Ορίζουμε οπότε , το αυτόματο , την Turing μηχανή καθώς και πότε θα λέμε ότι ένα αυτόματο αναγνωρίζει μια γλώσσα και πότε μια Turing μηχανή αναγνωρίζει μια γλώσσα.

Ορισμός : Πεπερασμένο αυτόματο είναι μια πεντάδα (Q , X , δ , q_0 , T) , όπου

1. Q είναι ένα πεπερασμένο σύνολο - τα στοχεία του ονομάζονται καταστάσεις
2. X είναι ένα πεπερασμένο σύνολο , που ονομάζεται αλφάβητο
3. δ : Q x X → Q είναι η συνάρτηση μεταβάσεων
4. $q_0 \epsilon$ Q είναι η εναρχτήρια κατάσταση , αρχική κατάσταση και
5. T ⊆ Q είναι το σύνολο των καταστάσεων αποδοχής , τελικές καταστάσεις

Λέμε ότι η απεικόνιση δ : Q x X → Q περιγράφει τη λειτουργία του αυτόματου A. Η σημασία της είναι η εξής : αν το αυτόματο βρίσκεται στην εσωτερική κατάσταση q και το τροφοδοτήσουμε με ένα γράμμα x από το αλφάβητο X, τότε θα μεταβεί στην εσωτερική κατάσταση δ(q , x) .

Με χρήση επαγωγής επάνω στο μήκος των λέξεων του X, μπορούμε να επεκτείνουμε τη δ σε μια απεικόνιση δ^* : Q x X^* → Q με τον ακόλουθο τρόπο:

$$\delta^* (q , \epsilon) = q$$

$$\delta^* (q , x_1...x_n x_{n+1}) = \delta (\delta^* (q , x_1...x_n), x_{n+1})$$

όπου ϵ είναι η κενή λέξη του X^* , q είναι στοιχείο του συνόλου Q και $x_1, ..., x_{n+1}$ είναι γράμματα του X.

Ας υποθέσουμε τώρα ότι το αυτόματο A βρίσκεται στην αρχική εσωτερική κατάσταση q_0 και ας το τροφοδοτήσουμε με τη λέξη w∈ X^* . Τότε, το A θα μεταβεί στην εσωτερική κατάσταση $\delta^* (q_0 , w)$. Αν συμβεί η $\delta^* (q_0 , w)$, να είναι τελική κατάσταση, τότε λέμε ότι το αυτόματο αναγνωρίζει τη λέξη w .

Το σύνολο όλων των λέξεων που **αναγνωρίζει το αυτόματο** A, συμβολίζεται |A |ή A , ώστε :

$$|A| = \{ w\in X^* | \delta^* (q_0 , w) \in T \}$$

Άρα εάν A είναι το σύνολο όλων των λέξεων που αποδέχεται το αυτόματο M , λέμε τότε ότι το M αναγνωρίζει την γλώσσα A και γράφουμε L(M)=A .

Θα εξετάσουμε τώρα ένα πολύ πιο ισχυρό μοντέλο , που προτάθηκε το 1963 από τον Alan Turing , και λέγεται **μηχανή Turing** (συντομία TM ή M). Η συγκεκριμένη μηχανή μοιάζει αρκετά με το πεπερασμένο αυτόματο , αλλά διαθέτει άπειρη μνήμη την οποία μπορεί να προσπελάζει χωρίς περιορισμούς . Αποτελεί επομένως ένα πολύ πιο πιστό μοντέλο υπολογιστή γενικής χρήσης . Για την ακρίβεια , μια μηχανή Turing μπορεί να κάνει ό,τι και ένας πραγματικός υπολογιστής . Παρόλο αυτά , υπάρχουν προβλήματα που ούτε αυτή η μηχανή μπορεί να επιλύσει . Υπό μία πολύ ρεαλιστική έννοια , τα προβλήματα αυτά βρίσκονται πέρα από τα όρια των υπολογιστικών δυνατοτήτων .

Οι διαφορές ανάμεσα σε ένα πεπερασμένο αυτόματο και μια μηχανή Turing:

1. Μια TM μπορεί όχι μόνο να διαβάσει από την ταινία , αλλά και να γράφει σε αυτήν .

2. Η κεφαλή εγγραφής - ανάγνωσης μπορεί να κινείται τόσο προς τα δεξιά όσο και προς τα αριστερά .

3. Η ταινία είναι άπειρη .

4. Οι ειδικές καταστάσεις αποδοχής και απόρριψης προκαλούν τον άμεσο τερματισμό του υπολογισμού .

Κεντρική θέση στον ορισμό μιας TM κατέχει η συνάρτηση μεταβάσεων δ , η οποία περιγράφει πώς μεταβαίνει η μηχανή από το κάθε βήμα στο επόμενο . Η συνάρτηση αυτή είναι της μορφής $Qx\Gamma \rightarrow Qx\Gamma xA, \Delta$. Με άλλα λόγια , εάν η τρέχουσα κατάσταση της μηχανής είναι η q , το σύμβολο της ταινίας στην τρέχουσα θέση της κεφαλής είναι το α , και δ(q , α) = (r , b , A) , τότε η μηχανή αντικαθιστά στην ταινία το σύμβολο α με το b και μεταβαίνει στην

κατάσταση r . Η τρίτη συνιστώσα της τιμής δ , που είναι είτε A (αριστερά) είτε Δ (δεξιά) , καθορίζει εάν μετά την εγγραφή του b η κεφαλή θα κινηθεί προς τα αριστερά ή προς τα δεξιά .

Ορισμός : Μηχανή Turing είναι μια 7 - άδα (Q , Σ , Γ , δ , q_0 , q_{accept}, q_{reject}) , όπου τα Q , Σ και Γ είναι πεπερασμένα σύνολα , και

1. Q είναι το σύνολο καταστάσεων
2. Σ είναι το αλφάβητο εισόδου , που δεν περιέχει το σύμβολο διαστήματος ⊔
3. Γ είναι το αλφάβητο ταινίας , με $\sqcup \epsilon \Gamma$ και $\Sigma \subseteq \Gamma$
4. $\delta : Qx\Gamma \rightarrow Qx\Gamma x\{A, \Delta\}$ είναι η συνάρτηση μεταβάσεων
5. $q_0 \varepsilon Q$ είναι η εναρκτήρια κατάσταση
6. $q_{accept} \epsilon Q$ είναι η κατάσταση αποδοχής , και
7. $q_{reject} \epsilon Q$ είναι η κατάσταση απόρριψης , και ισχύει ότι $q_{accept} \neq q_{reject}$.

Ας είναι Σ ένα πεπερασμένο (μη κενό σύνολο) αλφάβητο με τουλάχιστον δύο στοιχεία , και ας είναι Σ^* το σύνολο των πεπερασμένων συμβολοσειρών της Σ. Έτσι, μια γλώσσα της Σ είναι ένα υποσύνολο του Σ^* . Κάθε Turing μηχανή έχει ένα αλφάβητο Σ ως είσοδό του. Για κάθε συμβολοσειρά w της Σ^* υπάρχει υπολογισμός από μια Turing μηχανή M η οποία έχει ως είσοδο το w .

Λέμε ότι M δέχεται το w αν αυτός ο υπολογισμός τερματίζει σε μια κατάσταση αποδοχής. Ας σημειωθεί ότι η M δεν δέχεται το w , είτε γιατί ο υπολογισμός καταλήγει σε απορριπτική κατάσταση είτε διότι ο υπολογισμός αδυνατεί να τερματίσει.

Λέμε λοιπόν [4] πως μια **Turing μηχανή αναγνωρίζει την γλώσσα L** με αλφάβητο Σ, το οποίο συμβολίζεται με L(M) και ορίζεται ως εξής :

$$L(M)=\{\ w\in \Sigma^* \mid M \text{ αποδέχεται το } w\ \}$$

Η συλλογή λοιπόν των λέξεων που αποδέχεται η μηχανή M είναι η γλώσσα της M ή η γλώσσα που αναγνωρίζεται από την M , και συμβολίζεται με L(M) .

Με απλά λόγια λοιπόν μια γλώσσα λέγεται (κατά Turing) αναγνωρίσιμη εάν υπάρχει μηχανή Turing που να την αναγνωρίζει .

Ορίζουμε επίσης ως $t_M(w)$ τον αριθμό των βημάτων που κάνει μια μηχανή M με είσοδο w . Επίσης ορίζουμε ως $T_M(n)$ ως το χειρότερο χρόνο εκτέλεσης μιας μηχανή M ως :

$$T_M(n) = \max\ \{\ t_m(w) \mid w\in \Sigma^n\ \}$$

όπου Σ^n είναι το σύνολο όλων των δυνατών συμβολοσειρών του Σ μεγέθους n.

Τέλος, λέμε ότι μια **μηχανή TM τρέχει σε πολυωνυμικό χρόνο** αν υπάρχει k τέτοιο ώστε για όλα τα n , $T_M(n) \leq n^k + k$.

Μπορούμε τώρα στην επόμενη υποενότητα να δώσουμε τους βασικούς ορισμούς των κλάσεων P και BPP .

2.3 Οι κλάσεις P και BPP

Ακόμη και όταν ένα πρόβλημα είναι επιλύσιμο , και άρα μπορεί να αντιμετωπιστεί υπολογιστικά , πιθανόν να είναι πρακτικά ανεπίλυτο , εάν η επίλυσή του απαιτεί εξωπραγματικό χρόνο ή υπερβολικά πολλή μνήμη . Οπότε μας ενδιαφέρει και ο χρόνος εκτέλεσης αλλά και η μνήμη που χρειάζεται μια μηχανή.

Η κλάση P κατέχει κεντρική θέση διότι ο πολυωνυμικός χρόνος είναι μια θεωρητική προσέγγιση για έναν εφικτό υπολογισμό.

Ορισμός : Η κλάση **P** [5] αποτελείται από τις γλώσσες L για τις οποίες υπάρχει ντετερμινιστικός πολυωνυμικού χρόνου αλγόριθμος A τέτοιος ώστε

- $x \in L \Rightarrow A(x)$ accepts.
- $x \notin L \Rightarrow A(x)$ rejects.

Σε ορισμένα προβλήματα , τα λεγόμενα προβλήματα βελτιστοποίησης , μας δίνεται μια συλλογή δυνατών λύσεων και μας ζητείται να βρούμε μεταξύ αυτών μια βέλτιστη λύση . Ωστόσο , πιθανόν στην πράξη να μην είναι απαραίτητη η απολύτα βέλτιστη λύση . Ίσως μία λύση που απλώς προσεγγίζει τη βέλτιστη να επαρκεί , και να είναι πολύ ευκολότερο να βρεθεί .

Όπως υποδηλώνει και η ονομασία του , ένας **προσεγγιστικός αλγόριθμος (approximation algorithm)** είναι σχεδιασμένος έτσι ώστε να βρίσκει τέτοιες προσεγγιστικά βέλτιστες λύσεις [2].

Πιθανοκρατικός αλγόριθμος { probabilistic algorithm } είναι κάθε αλγόριθμος που χρησιμοποιεί εκ κατασκευής το αποτέλεσμα μιας τυχαίας διεργασίας . Κατά κανόνα , ένας τέτοιος αλγόριθμος περιλαμβάνει κάποια εντολή τύπου "ρίψης νομίσματος" και το αποτέλεσμα αυτών των ρίψεων επηρεάζει τη συνέχεια της εκτέλεσής του και επομένως και την έξοδο που παράγει . Κάποιοι τύποι προβλημάτων φαίνεται να επιλύονται ευκολότερα με πιθανοκρατικούς αλγορίθμους απ'ότι με ντετερμινιστικούς .

Πώς είναι όμως δυνατόν μια τυχαία απόφαση που λαμβάνεται μέσω της ρίψης ενός νομίσματος να είναι προτιμότερη από τον ακριβή ή έστω τον προσεγγιστικό υπολογισμό της καλύτερης επιλογής σε κάθε συγκεκριμένη περίσταση ;

Η απάντηση είναι ότι μερικές φορές ο υπολογισμός της βέλτιστης επιλογής απαιτεί υπερβολικό χρόνο , ενώ η προσεγγιστική εκτίμησή της εισάγει στο αποτέλεσμα κάποια μεροληψία που το καθιστά αναξιόπιστο .

Για να μελετήσουμε τους πιθανοκρατικούς υπολογισμούς θα περιγράψουμε πρώτα ένα μοντέλο πιθανοκρατικής TM .

Ορισμός : Πιθανοκρατική TM (probabilistic TM) είναι μια μη ντετερμινιστική TM στην οποία το κάθε μη ντετερμινιστικό βήμα ονομάζεται **κερματοριπτικό (coin-flip step)** και έχει δύο αποδεκτές επόμενες κινήσεις . Σε κάθε κλάδο b του υπολογισμού μιας τέτοιας μηχανής M για είσοδο w αποδίδουμε πιθανότητα

$$\Pr[b]=2^{-k}$$

όπου k το πλήθος των κερματοριπτικών βημάτων κατά μήκος του b . Ορίζουμε επίσης ως πιθανότητα αποδοχής της w από την M την ποσότητα

$$\Pr[\text{ η M αποδέχεται την w }]= \sum_{accepting-b_i} \Pr[b]$$

Οι πιθανοκρατικοί αλγόριθμοι που μας ενδιαφέρουν περισσότερο είναι εκείνοι που παρουσιάζουν καλή χρονική ή χωρική απόδοση . Η χρονική και η χωρική πολυπλοκότητα μιας πιθανοκρατικής TM μετρούνται όπως και στις μη ντετερμινιστικές μηχανές , δηλαδή με βάση τον υπολογιστικό κλάδο που χρησιμοποιεί τον περισσότερο χρόνο ή χώρο , αντίστοιχα .

Ορισμός : Η κλάση **BPP** [5] αποτελείται από τις γλώσσες L για τις οποίες υπάρχει πιθανοκρατικός πολυωνυμικού χρόνου αλγόριθμος A τέτοιος ώστε

- $x \in L \Rightarrow Pr$ [A(x) accepts] $\geq \frac{2}{3}$
- $x \notin L \Rightarrow Pr$ [A(x) accepts] $\leq \frac{1}{3}$

Η επιλογή της σταθεράς $\frac{1}{3}$ στον ορισμό αυτό είναι εν μέρει αυθαίρετη .

Ορισμός : Η κλάση **RP** [5] αποτελείται από τις γλώσσες L για τις οποίες υπάρχει πιθανοκρατικός πολυωνυμικού χρόνου αλγόριθμος A τέτοιος ώστε

- $x \in L \Rightarrow Pr$ [A(x) accepts] $\geq \frac{1}{2}$
- $x \notin L \Rightarrow Pr$ [A(x) accepts] $= 0$

Στο τέλος της ενότητας αυτής θα δώσω τον ορισμό για μια θεμελιώδη έννοια της σύγχρονης θεωρίας της κρυπτογραφίας , τις μονόδρομες (one-way functions) συναρτήσεις. Γιατί εάν δεχθούμε την ύπαρξη μονόδρομων συναρτήσεων μπορούμε να κατασκευάσουμε ασφαλή κρυπτοσυστήματα ιδιωτικού κλειδιού .

Στους παρακάτω ορισμούς υπάρχουν κάποιες έννοιες για τις οποίες παραπέμπω στη βιβλιογραφία [2].

Ορισμός : Μονόδρομη μετάθεση (one-way permutation)[2] είναι οποιαδήποτε μετάθεση f που έχει επιπλέον τις εξής δύο ιδιότητες :

1. Υπολογίζεται σε πολυωνυμικό χρόνο

2. Για κάθε k , κάθε αρκετά μεγάλο n ,και κάθε πολυωνυμικού χρόνου πιθανοκρατική TM M , εάν επιλέξουμε τυχαία μια λέξη w μήκους n και εκτελέσουμε την M για είσοδο f(w) , τότε

$$Pr_{M,w} [\mathrm{M(f(w))=w}] \leq n^{-k}$$

Ο συμβολισμός $Pr_{M,w}$ σημαίνει ότι η πιθανότητα λαμβάνεται επί των τυχαίων επιλογών της M και την τυχαία επιλογή της w .

Μονόδρομη συνάρτηση (one-way function)[2] είναι οποιαδήποτε μακροδιατηρητική συνάρτηση f που έχει επιπλέον τις εξής δύο ιδιότητες :

1. Μπορεί να υπολογιστεί σε πολυωνυμικό χρόνο

2. Για κάθε k κάθε αρκετά μεγάλο n , και κάθε πολυωνυμικού χρόνου πιθανοκρατική TM M , εάν επιλέξουμε τυχαία μια λέξη w μήκους n και εκτελέσουμε την M για είσοδο f(w) , τότε

$$Pr_{M,w} [\mathrm{M(f(w))=y,όπου\ f(y)=f(w)}] \leq n^{-k}$$

3 Τυχαιότητα

Η σειρά που θα ακολουθήσω και νομίζω ταιριάζει καλύτερα να μιλήσω εν συντομία για το τυχαίο είναι η εξής. Αρχικά θα αναφερθώ στο πατέρα της λογικής Kurt Gödel όπου θα γίνει άμεση σύνδεση με τον Alan Turing. Στη συνέχεια θα γίνει μια μικρή αναφορά για την προσπάθεια δημιουργίας ενός τυπικού αξιωματικού συστήματος από τον David Hilbert . Την αποτυχή προσπάθεια αυτή του Hilbert τελικά θα διαδεχθεί ο Alan Turing και θα γίνει αναφορά στον Emil Post για τα τυπικά αξιωματικά συστήματα. Τέλος θα σας παρουσιάσω πως ο ορισμός της τυχαιότητας δεν είναι δυνατός .

3.1 Kurt Gödel - Θεώρημα Μη Πληρότητας

Όπως αναφέρει και ο ίδιος ο Gregory Chaitin [6] ερωτεύτηκε με την πρώτη ματιά το 1958 ένα βιβλίο που έπεσε στα χέρια του . Οι συγγραφείς του ήταν οι Νάγκελ (Nagel) και Νιούμαν (Newman) και ο τίτλος του : Η απόδειξη του Gödel . (Gödel's Proof). Ίσως σε αυτό το βιβλίο να βρισκόταν η πιθανή εξήγηση των δυσκολιών που συναντούσαν οι μαθηματικοί με τους πρώτους αριθμούς : το Θεώρημα της Μη Πληρότητας του Gödel , που λέει ότι κάθε πεπερασμένο σύστημα μαθηματικών αξιωμάτων , κάθε μαθηματική θεωρία είναι μη πλήρης . Πιο συγκεκριμένα , ο Gödel απέδειξε ότι θα υπάρχουν πάντα προτάσεις σχετικές με την αριθμητική , προτάσεις που να αφορούν θετικούς ακέραιους , πρόσθεση , πολλαπλασιασμό - ονομάζονται αριθμοθεωρητικές - , οι οποίες είναι αληθείς αλλά όχι αποδείξιμες!

Μας λέει χαρακτηριστικά μέσα στο βιβλίο του ο συγγραφέας , "δεν ήμουν ηλίθιος που δεν μπορούσα να κατανοήσω την απόδειξη του Gödel ". Ήταν σε θέση να την παρακολουθήσει βήμα προς βήμα , του έμοιαζε όμως σαν να προσπαθούσε να αναμείξει το λάδι με το νερό. Το μυαλό του αντιστεκόταν , λάτρευε τη μη πληρότητα αλλά όχι την απόδειξη του Gödel . Έτσι το πρώτο βήμα για τη δουλειά που ξεκίνησε ο ίδιος δεν ήταν η απόδειξη του Gödel του 1931 αλλά η εναλλακτική προσέγγιση στη μη πληρότητα από τον Alan Turing , το 1936 , στην οποία η ιδέα του υπολογισμού παίζει καθοριστικό ρόλο. Χάρη στον Turing η ιδέα του υπολογισμού , η ιδέα του υπολογιστή , έγινε η νέα δύναμη του μαθηματικού συλλογισμού. Και γιατί ο Hilbert ;

3.2 David Hilbert- Φορμαλισμός των Μαθηματικών

Ο David Hilbert , πριν από έναν περίπου αιώνα , διερεύνησε πρώτος κατά πόσο μπορούμε να μελετήσουμε την ισχύ των μαθηματικών χρησιμοποιώντας τα ίδια τα μαθηματικά. Από αυτή την άποψη του θεωρείται ως ο δημιουργός των μεταμαθηματικών. Δική του ιδέα ήταν ότι για να μπορέσουμε να μελετήσουμε τι μπορούν να επιτύχουν τα μαθηματικά , πρέπει πρώτα να αποσαφηνίσουμε τους κανόνες του παιχνιδιού. Δική του ιδέα ήταν η δημιουργία ενός τυπικού αξιωματικού συστήματος (ή ΤΑΣ) για το σύνολο των μαθηματικών. Θα ήταν ένα σύστημα που θα βοηθούσε να απαλειφθούν όλες οι ασάφειες στη μαθηματική επιχειρηματολογία , ένα σύστημα που θα εξάλειφε κάθε αμφιβολία σχετικά με το αν μια μαθηματική απόδειξη είναι σωστή ή όχι.

Πώς μπορεί να γίνει κάτι τέτοιο ; Τι είναι ένα τυπικό αξιωματικό σύστημα; Η γενική ιδέα είναι ότι μοιάζει πολύ με τα Στοιχεία του Ευκλείδη , μόνο που πρέπει να είμαστε πολύ πολύ πιο σχολαστικοί σχετικά με τις λεπτομέρειες !

Το πρώτο βήμα είναι να δημιουργηθεί μια εντελώς τυπική τεχνητή γλώσσα για τα μαθηματικά. Προσδιορίζουμε το αλφάβητο των συμβόλων που θα χρησιμοποιήσουμε , τη γραμματική , τα αξιώματα , τους κανόνες παραγωγής συμπερασμάτων και έναν αλγόριθμο ελέγχου των αποδείξεων :

Το τυπικό αξιωματικό σύστημα του Hilbert

Αλφάβητο

Γραμματική

Κανόνες συναγωγής

Αλγόριθμος ελέγχου των αποδείξεων

Οι μαθηματικές αποδείξεις οφείλουν να διατυπωθούν σε αυτή τη γλώσσα χωρίς να λείπει τίποτα · η κάθε μικροσκοπική λεπτομέρεια του συλλογισμού πρέπει να είναι στη θέση της. Αρχίζετε με τα αξιώματα και στη συνέχεια εφαρμόζετε τους κανόνες συναγωγής , έναν προς έναν , μέχρι να παράξετε όλα τα θεωρήματα! Αυτό μοιάζει πολύ με μια γλώσσα προγραμματισμού υπολογιστών : αρκετά σαφής ώστε η μηχανή να μπορεί να την ερμηνεύει , να μπορεί να την κατανοεί , αρκετά σαφής ώστε η μηχανή να την ελέγχει. Χωρίς καθόλου αμφισημίες. Χωρίς αντωνυμίες. Χωρίς ορθογραφικά λάθη και με άψογη γραμματική !

3.3 Alan Turing- Το Πρόβλημα Τερματισμού

Η πρώτη πράξη όμως σε αυτό το δράμα , στην παραχμή και πτώση της μαθηματικής βεβαιότητας , ήταν το περίφημο άρθρο του 1936 όπου ο Alan Turing εισήγαγε τον υπολογιστή ως μαθηματική ιδέα , όχι ως τεχνητό δημιούργημα. Και το πιο ενδιαφέρον στοιχείο σε αυτό το άρθρο είναι το περίφημο πρόβλημα τερματισμού του Turing , που αποδεικνύει ότι υπάρχουν πράγματα που κανένας υπολογιστής δεν μπορεί να υπολογίσει , άσχετα με το πόσο έξυπνα θα τον προγραμματίσετε , άσχετα με το πόση υπομονή θα δείξετε μέχρι να πάρετε την απάντηση. Στην ουσία , ο Turing ανακαλύπτει δύο πράγματα : το πρόβλημα τερματισμού και τους μη υπολογίσιμους πραγματικούς αριθμούς.

Εδώ θα μιλήσουμε μόνο για το πρόβλημα τερματισμού. Τι είναι το πρόβλημα τερματισμού ; Είναι το ερώτημα αν ένα απόλυτα αυτόνομο και αυτάρκες πρόγραμμα υπολογιστή , ένα πρόγραμμα χωρίς εισαγωγές και εξαγωγές (input - output) , πρόκειται ποτέ να σταματήσει ; Αν το πρόγραμμα χρειάζεται οποιουσδήποτε αριθμούς , αυτοί πρέπει να δίνονται από το ίδιο το πρόγραμμα , όχι να εισάγονται από τον εξωτερικό κόσμο.

Ο Turing κατάφερε να αποδείξει το εξαιρετικά θεμελιώδες θεώρημα ότι δεν υπάρχει τρόπος να αποφανθούμε εκ των προτέρων και σε πεπερασμένο χρόνο αν το πρόγραμμα του υπολογιστή θα σταματήσει ποτέ. Αν σταματήσει ενδέχεται να το αντιληφθούμε. Το πρόβλημα είναι να αποφασίσουμε πότε να παραιτηθούμε και να πειστούμε ότι δε θα σταματήσει ποτέ. Αλλά δεν υπάρχει τρόπος να κάνουμε κάτι τέτοιο.

Η καταπληκτική ιδέα του Turing ήταν να ορίσει την έννοια της υπολογισιμότητας , της διάκρισης δηλαδή ανάμεσα σε πράγματα που μπορούν και σε πράγματα που δεν μπορούν να υπολογιστούν και στη συνέχεια να συναγάγει τη μη πληρότητα από τη μη υπολογισιμότητα.

Turing: Η μη υπολογισιμότητα συνεπάγεται τη μη πληρότητα!

Εξίσου καταπληκτική όμως και η προσέγγιση του Chaitin ο οποίος συμπεραίνει τη μη πληρότητα από κάτι πιο βαθύ. Στη δική του περίπτωση η τυχαιότητα και όχι η μη υπολογισιμότητα θα είναι αυτή που θα οδηγήσει στη μη πληρότητα.

Gregory Chaitin : Η τυχαιότητα συνεπάγεται τη μη πληρότητα!

3.4 Τυπικό Αξιωματικό Σύστημα

Το επόμενο βήμα σε αυτό το μονοπάτι έγινε το 1944 από τον Emil Post , που υπήρξε καθηγητής στο City College του Πανεπιστημίου της Νέας Υόρκης. Ο Post είχε την έμπνευση και το βάθος σκέψης ώστε να κατανοήσει την ιδέα-κλειδί στην απόδειξη του Turing σχετικά με το ότι η μη υπολογισιμότητα συνεπάγεται τη μη πληρότητα . Διαισθάνθηκε ότι η ουσία του ΤΑΣ έγκειται στην ύπαρξη ενός αλγορίθμου για την παραγωγή όλων των θεωρημάτων , ενός αλγορίθμου που είναι πολύ αργός και δε σταματά ποτέ , αλλά κάποια στιγμή φτάνει σε καθένα από αυτά τα θεωρήματα .

Αυτός ο εκπληκτικός αλγόριθμος παράγει τα θεωρήματα κατά σειρά μεγέθους , όχι όμως κατά τη σειρά μεγέθους της διατύπωσης του κάθε θεωρήματος αλλά κατά τη σειρά μεγέθους της απόδειξης.

Το τυπικό αξιωματικό σύστημα του Hilbert - Turing - Post

Μηχανή που παράγει όλα τα θεωρήματα

ένα προς ένα , σε μια αυθαίρετη σειρά

Με άλλα λόγια , υπάρχει ένας αλγόριθμος , ένα πρόγραμμα υπολογιστή , που μπορεί να εκτελέσει αυτή την εργασία. Αυτό είναι το ουσιαστικό περιεχόμενο της έννοιας του ΤΑΣ : είναι ένα παιχνίδι που μπορεί κάποιος να χρησιμοποιήσει για να μελετήσει τα όρια της τυπικής αξιωματικής μεθόδου. Το κλειδί είναι ότι υπάρχει ένας αλγόριθμος που να παράγει όλα τα θεωρήματα αυτό μας ενδιαφέρει.

Λοιπόν , ποιά είναι η ουσία ; Η ουσία είναι ότι ο στόχος του Hilbert για ένα ΤΑΣ που να περιλαμβάνει όλα τα μαθηματικά είναι ανέφικτος γιατί δε μπορούμε να χωρέσουμε όλη τη μαθηματική αλήθεια μέσα σε ένα ΤΑΣ. Τα μαθηματικά δεν μπορούν να είναι στατικά , πρέπει να είναι δυναμικά , να εξελίσσονται. Είμαστε υποχρεωμένοι να επεκτείνουμε συνεχώς το ΤΑΣ , να του προσθέτουμε νέες ιδέες , νέα αξιώματα. Έτσι στην αντιπαράθεση του Hilbert με τον Henri Poincaré σχετικά με το φορμαλισμό και τη διαισθητικότητα , τοποθετούμαστε πια **οριστικά με το μέρος της διαισθητικότητας** . Ας δούμε και ένα πολύ εύστοχο παράδειγμα.

Τι είναι μια διοφαντική εξίσωση ; Είναι μια αλγεβρική εξίσωση στην οποία οι συντελεστές και οι άγνωστοι οφείλουν να είναι ακέραιοι. Και τι είναι το 10^o πρόβλημα του Hilbert ; Είναι η πρόκληση που απήφθυνε ο Hilbert στην ανθρωπότητα , να αναχαλύψει έναν αλγόριθμο που να προσδιορίζει αν μια διοφαντική εξίσωση είναι επιλύσιμη. Με άλλα λόγια , έναν αλγόριθμο που να αποφαίνεται αν υπάρχουν κάποιοι ακέραιοι αριθμοί που μπορούμε να βάλουμε στη θέση των αγνώστων ώστε να ικανοποιούν την εξίσωση.

Παρατηρήστε ότι αν υπάρχει λύση , μπορούμε να τη βρούμε αντιχαθιστώντας στη θέση των αγνώστων όλες τις δυνατές τιμές ακεραίων αριθμών , αρχίζοντας με μικρούς αριθμούς και σταδιακά ανεβαίνοντας προς τα πάνω. Το ερώτημα είναι να αποφασίσουμε πότε πρέπει να παραιτηθούμε ; Τι μας θυμίζει αυτό ; Ναι πρόκειται ακριβώς για το ίδιο πράγμα που συμβαίνει και με το πρόβλημα τερματισμού του Turing .

Δεν μπορούμε να αποφανθούμε αν μια συγκεκριμένη διοφαντική εξίσωση έχει λύσεις. Γιατί αν μπορούσαμε να το κάνουμε αυτό , θα μπορούσαμε να αποφασίσουμε αν ένα πρόγραμμα υπολογιστή δίνει κάποιο αποτέλεσμα , και αν μπορούσαμε να το κάνουμε αυτό , θα μπορούσαμε να επιλύσουμε το πρόβλημα τερματισμού του Turing .

Άραγε , γιατί όταν έχουμε τη δυνατότητα να αποφανθούμε αν ένα πρόγραμμα υπολογιστή δίνει αποτέλεσμα , θα έχουμε και τη δυνατότητα να αποφασίζουμε αν ένα πρόγραμμα τερματίζει ; Ο λόγος είναι ο ακόλουθος : ένα οποιοδήποτε πρόγραμμα που είτε τερματίζει είτε δεν τερματίζει και δεν παράγει αποτέλεσμα

μπορεί να μετατραπεί σε ένα άλλο που να παράγει το μήνυμα "ετοιμάζομαι να τερματίσω ! "(σε δυαδική μορφή ως ακέραιοι) τη στιγμή που είναι έτοιμο να τερματίσει. Αν το πρόγραμμα είναι γραμμένο σε υψηλού επιπέδου γλώσσα , αυτό είναι εύκολο. Αν είναι γραμμένο σε κώδικα μηχανής , θα χρειαστεί να "τρέξετε "το πρόγραμμα βήμα προς βήμα και να εντοπίσετε πότε είναι έτοιμο να τερματίσει. Συνεπώς , εάν είστε σε θέση να αποφασίσετε αν το τροποποιημένο πρόγραμμα παράγει αποτέλεσμα , τότε είστε σε θέση να αποφασίσετε και εάν το αρχικό πρόγραμμα τερματίζει μας εξηγεί ο συγγραφέας.

Αυτό είναι όλο και όλο : οι διοφαντικές εξισώσεις είναι στην ουσία υπολογιστές . Εκπληκτικό ! Ακόμη , αυτό αποδεικνύει ότι η θεωρία αριθμών είναι δύσκολη ! Αποδεικνύει ότι η μη υπολογισιμότητα και η μη πληρότητα τριγυρνάνε στην καρδιά του προβλήματος , σε διοφαντικές εξισώσεις ηλικίας άνω των δύο χιλιάδων ετών ! Ο ίδιος ο Chaitin μας δείχνει έπειτα στο βιβλίο του ότι και η τυχαιότητα κάπου εκεί κρύβεται...

3.5 Αδυναμία-ορισμού-της-τυχαιότητας του Borel

Τέλος σας παραθέτω τις αναμνήσεις του Gregory Chaitin σχετικά με την ανάλυση του Émile Borel , ενός προβλήματος που εγείρεται αναπόφευκτα με οποιαδήποτε προσπάθεια ορισμού της τυχαιότητας.

Ας υποθέσουμε ότι έχουμε κάποιο τρόπο να διακρίνουμε τους ακέραιους αριθμούς που τα ψηφία τους στο δεκαδικό σύστημα συγκροτούν μια τυχαία ακολουθία , από αυτούς που δεν έχουν αυτή την ιδιότητα.

Ας σκεφτούμε τώρα τον πρώτο N-ψήφιο αριθμό που ικανοποιεί τον συγκεκριμένο ορισμό της τυχαιότητας. Όμως αυτός ο συγκεκριμένος αριθμός είναι μη τυπικός , δεν ικανοποιεί της προϋποθέσεις που τον ορίζουν , αφού τυχαίνει να είναι ο πρώτος που έχει αυτή τη συγκεκριμένη ιδιότητα !

Το πρόβλημα είναι ότι "τυχαίο"σημαίνει "τυπικό", δεν ξεχωρίζει από το πλήθος , δεν έχει διακριτικά χαρακτηριστικά . Αν όμως είμαστε σε θέση να ορίσουμε την τυχαιότητα , τότε η ιδιότητά του να είναι κάτι τυχαίο γίνεται ένα ακόμα χαρακτηριστικό που μπορεί να χρησιμοποιηθεί για να δείξουμε ότι ορισμένοι αριθμοί είναι μη τυπικοί και ξεχωρίζουν από το πλήθος !

Έτσι , με αυτό τον τρόπο έχουμε μια έννοια ιεραρχίας της τυχαιότητας , αυτή με την οποία ξεκινήσαμε , η επόμενη , αυτή που προκύπτει από αυτή , και ούτω καθεξής...Καθεμία από αυτές προκύπτει με βάση τους προηγούμενους ορισμούς της τυχαιότητας ως ένα ακόμα χαρακτηριστικό σαν αυτά που χρησιμοποιούνται για να κατηγοριοποιήσουμε τους αριθμούς !

Το συμπέρασμα του Borel είναι ότι δεν είναι εφικτός ένας οριστικός ορισμός της τυχαιότητας. Δεν είναι εφικτός ένας ορισμός που να τα συμπεριλαμβάνει όλα. Η τυχαιότητα είναι μια έννοια που ξεγλιστράει , δεν μπορείς να την αρπάξεις , κρύβει μέσα της το παράδοξο. Τίθεται θέμα να αποφασίσουμε πόσο απαιτητικοί θα είμαστε. Μπορείτε να αποφασίσετε κάποια στιγμή να πείτε "φτάνει "και σε εκείνο το σημείο να θεωρήσετε ότι έχετε το τυχαίο.

4 Ψευδοτυχαία bits και ακολουθίες

Στην ενότητα αυτή θα γίνει μια εισαγωγή σχετικά με τις γεννήτριες τυχαίων bit αλλά και τις ψευδοτυχαίες γεννήτριες τυχαίων bit και ακολουθιών [7]. Πιο συγκεκριμένα θα δοθούν οι ορισμοί των γεννητόρων αυτών αλλά και οι τρόποι με τους οποίους μπορούμε να τους κατασκευάσουμε. Τέλος θα δούμε κάποιες απαιτήσεις ασφαλείας και πέντε βασικά τεστ που θα πρέπει να περνούν με επιτυχία οι ψευδοτυχαίες γεννήτριες.

4.1 Ορισμοί τυχαίων-ψευδοτυχαίων γεννητόρων

Ας δούμε λοιπόν παρακάτω τον ορισμό μιας γεννήτριας τυχαίων bit .

Ορισμός : Μια **random bit generator - RBG** είναι ένας αλγόριθμος που εξάγει μια ακολουθία στατιστικά ανεξάρτητων και αμερόληπτων δυαδικών ψηφίων.

Παρατήρηση random bits vs. random numbers: μια RBG μπορεί να χρησιμοποιηθεί για να παράγει (ομοιόμορφα κατανεμημένους) τυχαίους αριθμούς. Για παράδειγμα ένας τυχαίος ακέραιος αριθμός στο διάστημα $[o, n]$ μπορεί να επιτευχθεί με τη δημιουργία μιας τυχαίας ακολουθίας bit μήκους (n+1) και έπειτα μετατροπή αυτού σε ακέραιο. Εάν ο ακέραιος που θα προκύψει υπερβαίνει τον αριθμό n τότε μπορούμε να τον απορρίψουμε και να δημιουργήσουμε μια νέα τυχαία ακολουθία bit .

Ωστόσο , η παραγωγή τυχαίων bits είναι μια αναποτελεσματική και δύσκολη διαδικασία. Επιπλέον , μπορεί να είναι πρακτικά αδύνατο με ασφάλεια να μεταφερθεί ένας μεγάλος αριθμός τυχαίων bits εφόσον αυτό απαιτείται από εφαρμογές όπως one-time pad. Σε τέτοιες περιπτώσεις , το πρόβλημα μπορεί να βελτιωθεί με την αντικατάσταση μιας τυχαίας γεννήτριας bit σε μια ψευδοτυχαία γεννήτρια. Δίνεται ο ορισμός μιας ψευδοτυχαίας γεννήτριας τυχαίων bit.

Ορισμός : Μια **γεννήτρια ψευδοτυχαίων bit , pseudorandom bit generator - PRBG** είναι ένας αλγόριθμος ντετερμινιστικός (εδώ σημαίνει ότι αν έχω τον ίδιο αρχικό σπόρο η γεννήτρια θα μου παράγει πάντα το ίδιο αποτέλεσμα - την ίδια ακολουθία) όπου δέχεται σαν είσοδο μια πραγματικά τυχαία δυαδική ακολουθία μήκους k και δίνει σαν έξοδο μια δυαδική ακολουθία μήκους $l \gg k$ η οποία 'φαίνεται'να είναι τυχαία. Η είσοδος στην PRBG καλείται seed (σπόρος) ενώ η έξοδος της ονομάζεται ακολουθία ψευδοτυχαίων bit (output pseudorandom bit generator).

4.2 Απαιτήσεις ασφάλειας για τις γεννήτριες

Η παραγωγή μιας PRBG δεν είναι τυχαία , στην πραγματικότητα ο αριθμός των πιθανών ακολουθιών που εξάγει η PRBG βρίσκεται σε ένα μικρό τμήμα που ονομάζεται $2^k/2^l$ όλων των πιθανών δυαδικών ακολουθιών μήκους l .

Στόχος είναι να ληφθεί μια μικρή και πραγματικά τυχαία ακολουθία και να μετατραπεί σε μια ακολουθία με πολύ μεγαλύτερο μήκος από την αρχική τέτοια ώστε ένας αντίπαλος να μην μπορεί να ξεχωρίσει αν η μεγάλη αυτή ακολουθία είναι προϊόν μιας PRBG ή μιας πραγματικά τυχαίας ακολουθίας μήκους l .

Για να πιστοποιήσουμε όμως ότι οι γεννήτριες αυτές είναι ασφαλείς πρέπει να υπόκεινται σε μια ποικιλία στατιστικών δοκιμασιών για την ανίχνευση ειδικών χαρακτηριστικών που περιμένει κανείς από τυχαίες ακολουθίες. Όπως θα δούμε και στο παράδειγμα παρακάτω , περνώντας μια γεννήτρια αυτούς τους στατιστικούς ελέγχους είναι αναγκαία αλλά όχι επαρκής προϋπόθεση για να είναι ασφαλής.

Παράδειγμα : μια γραμμική γεννήτρια παράγει μια ακολουθία ψευδοτυχαίων $x_1, x_2, x_3, ...$ σύμφωνα με την γραμμική επανάληψη

$$x_n = ax_n - 1 + bmodm, n \succeq 1$$

όπου οι ακέραιοι αριθμοί a,b,m είναι παράμετροι που χαρακτηρίζουν τη γεννήτρια ενώ x_0 είναι το μυστικό seed. Ενώ τέτοιου είδους γεννήτριες χρησιμοποιούνται συχνά για την προσομοίωση πιθανοτικών αλγορίθμων και περνούν τα στατιστικά τεστ παρόλο αυτά είναι προβλέψιμες άρα και εξ ολοκλήρου μη ασφαλείς για κρυπτογραφικούς σκοπούς. Ακόμη και αν δίνεται μια μερική ακολουθία που έχει παράγει η γεννήτρια αυτή , η υπόλοιπη ακολουθία μπορεί να ανακαταστευαστεί ακόμη και αν οι παράμετροι a,b,m είναι άγνωστοι.

Μια ελάχιστη απαίτηση ασφάλειας για μια γεννήτρια ψευδοτυχαίων bit είναι το μήκος k ενός τυχαίου seed να είναι αρκετά μεγάλο έτσι ώστε η αναζήτηση

πάνω από 2^k στοιχείων (ο συνολικός αριθμός των πιθανών σπόρων) να είναι ανέφικτη για τον αντίπαλο. Δύο γενικές απαιτήσεις είναι ότι οι ακολουθίες παραγωγής μιας PRBG :

1. θα πρέπει στατιστικώς να μην ξεχωρίζει από μια αληθινά τυχαία ακολουθία
2. τα bit εξόδου θα πρέπει να είναι απρόβλεπτα σε έναν αντίπαλο με περιορισμένους υπολογιστικούς πόρους (οι απαιτήσεις αυτές δίνονται παρακάτω στους ορισμούς)

Ορισμός : Μια γεννήτρια ψευδοτυχαίων bit λέμε ότι περνάει όλα τα πολυωνυμικού χρόνου στατιστικά τεστ (polynomial-time statistical tests) εάν κανένας αλγόριθμος πολυωνυμικού χρόνου δε μπορεί σωστά να διακρίνει μια output ακολουθία της γεννήτριας και μιας πραγματικά τυχαίας ακολουθίας που έχει το ίδιο μήκος , με πιθανότητα σημαντικά μεγαλύτερη του 1/2.

Ορισμός : Μια γεννήτρια ψευδοτυχαίων bit λέμε ότι περνάει το τεστ του επόμενου bit (next-bit test) εάν δεν υπάρχει αλγόριθμος πολυωνυμικού χρόνου ο οποίος σε είσοδο των πρώτων l bits μια ακολουθίας s , να μπορεί να προβλέψει το (l+1)ο bit με πιθανότητα σημαντικά μεγαλύτερη του 1/2.

Συμπέρασμα : Μια PRBG περνάει το next-bit test αν και μόνο αν περνάει όλα τα polynomial-time statistical tests.

Ορισμός : Μια PRBG που περνάει το next-bit test καλείται κρυπτογραφικά ασφαλής γεννήτρια ψευδοτυχαίων bit (cryptographically secure pseudorandom bit generator - CSPRBG).

4.3 Δημιουργία τυχαίων bit

Μια πραγματικά τυχαία γεννήτρια bits απαιτεί μια φυσική πηγή τυχαιότητας. Το να σχεδιαστεί μια συσκευή ή ένα λογισμικό ώστε να εκμεταλευτεί αυτή την τυχαιότητα και να παράγει μια ακολουθία bit που είναι απαλλαγμένη από "προβλέψεις"και "συσχετίσεις"είναι ένα αρκετά δύσκολο έργο. Επιπλέον , για τις περισσότερες κρυπτογραφικές εμαρφογές , η γεννήτρια δεν πρέπει να υπόκεινται σε παρατήρηση ή χειραγώγηση από τον αντίπαλο. Σε αυτή τη παράγραφο θα ερευνήσουμε μερικές πιθανές πηγές τυχαίων bits.

Οι γεννήτριες που βασίζονται σε φυσικές πηγές τυχαιότητας υπόκεινται σε επιρροές από εξωτερικούς παράγοντες καθώς επίσης και δυσλειτουργίες τους. Είναι επιταχτική λοιπόν η ανάγκη οι συσκευές αυτές να ελέγχονται ανά τακτά χρονικά διαστήματα π.χ. χρησιμοποιώντας στατιστικά τεστ.

(i) Hardware-based generators

Hardware-based random bit generators εκμεταλεύονται την τυχαιότητα που προκαλείται χάρη σε φυσικά φαινόμενα. Οι φυσικές αυτές όμως διεργασίες μπορεί να παράγουν bits που είναι μεροληπτικά ή συσχετίζονται οπότε σε αυτή την περίπτωση πρέπει να υποβάλλονται σε de-skewing τεχνικές , τις οποίες αναφέρω παρακάτω.

Παραδείγματα τέτοιων φυσικών φαινομένων είναι:

1. ο χρόνος που πέρασε μεταξύ εκπομπών σωματιδίων κατά τη διάρκεια μιας ραδιενεργούς αποσύνθεσης

2. ο θερμικός θόρυβος από μια δίοδο ημιαγωγών ή αντίστασης

3. η αστάθεια της συχνότητας ενός ελεύθερου ταλαντωτή

4. το ύψος ενός ημιαγωγού πυκνωτή σε σχέση με την περίοδό του

5. ο στροβιλισμός του αέρα μέσα σε ένα σφραγισμένο δίσκο που προκαλεί τυχαίες διακυμάνσεις στο δίσκο

6. ο ήχος από μια είσοδο μικροφώνου ή βίντεο χάμερας

Οι γεννήτριες που βασίζονται στα δύο πρώτα φαινόμενα πρέπει να κατασκευαστούν εξωτερικά χρησιμοποιώντας τα τυχαία bits και ως εκ τούτου μπορεί να υπόκεινται σε παρατήρηση από τον αντίπαλο. Οι γεννήτριες που βασίζονται σε ταλαντωτές και πυκνωτές μπορούν να κατασκευαστούν σε VLSI συσκευές αλλά μπορούν να περικλείονται σε συσκευασίες από ανθεκτικό υλικό ώστε να προστατεύονται από τους αντιπάλους.

(ii) Softoware-based generators

Ο σχεδιασμός μια γεννήτριας τυχαίων bits σε λογισμικό είναι ακόμη πιο δύσκολο από ότι σε hardware . Τέτοιες διεργασίες μπορεί να βασίζονται :

1. στο ρολόι του συστήματος

2. στο χρόνος που παρήλθε μεταξύ πληκτρολογήσεων ή κινήσεων του ποντικιού

3. στο περιεχόμενο των εισροών/εκροών buffers

4. σε κάποια εισαγωγή του χρήστη

5. system load ή στατιστικά στοιχεία του δικτύου

Η συμπεριφορά αυτών των διεργασιών μπορεί να ποικίλει σημαντικά ανάλογα με διάφορους παράγοντες όπως η πλατφόρμα του ηλεκτρονικού υπολογιστή. Μπορεί επίσης να είναι δύσκολο να αποφευχθεί ένας αντίπαλος από την παρατήρηση ή το χειρισμό αυτών των διαδικασιών.

Για παράδειγμα αν ο αντίπαλος έχει μια γενική ιδέα για το πότε μια τυχαία ακολουθία δημιουργήθηκε μπορεί να μαντέψει το περιεχόμενο του ρολογιού του συστήματος με αρκετά μεγάλη ακρίβεια. Έτσι ένα καλά σχεδιασμένο λογισμικό γεννήτριας τυχαίων bits θα πρέπει να αξιοποιεί και πολλές καλές πηγές τυχαιότητας που είναι διαθέσιμες.

Χρησιμοποιώντας πολλές ασφαλείς πηγές , κάθε πηγή πρέπει να υποβληθεί σε δειγματοληψία και το δείγμα θα πρέπει να συνδυαστεί με μια συνάρτηση που τα κάνει μίξη. Μια τεχνική για να πετύχει αυτό είναι να εφαρμοστεί μια κρυπτογραφική hash συνάρτηση . Σκοπός της τεχνικής αυτής είναι να *"αποστάξει"* τα πραγματικά τυχαία bits από την ακολουθία της κάθε πηγής.

(iii) De-skewing

Μια φυσική πηγή τυχαίων bits μπορεί να παράγει ελλατωματικά outputs bits υπό την έννοια ότι μπορεί να είναι μεροληπτικά ή να συσχετίζονται. Υπάρχουν διάφορες τεχνικές για τη δημιουργία πραγματικά τυχαίων bit ακολουθιών από outputs bits ελλατωματικών γεννητριών , οι τεχνικές αυτές ονομάζονται de-skewing τεχνικές.

Παράδειγμα : Ας υποθέσουμε ότι μια γεννήτρια παράγει προκατειλημμένα αλλά ασυσχέτιστα bits . Ας υποθέσουμε ότι η πιθανότητα για το 1 είναι p και η πιθανότητα για το 0 είναι p-1 όπου p είναι άγνωστη παράμετρος αλλά σταθερή με $0 < p < 1$. Εάν η ακολουθία εξόδου μιας τέτοιας γεννήτριας ομαδοποιείται σε ζευγάρια bits έτσι ώστε ένα ζευγάρι 10 να μετατρέπεται σε 1 , ένα ζευγάρι 01 να μετατρέπεται σε 0 και τα ζευγάρια 00 και 11 να απορρίπτονται , τότε η προκύπτουσα ακολουθία είναι τόσο αμερόληπτη αλλά και ασυσχέτιστη.

4.4 Δημιουργία ψευδοτυχαίων bit

Μια one-way function f μπορεί να χρησιμοποιηθεί για τη δημιουργία ακολουθίας ψευδοτυχαίων bit επιλέγοντας πρώτα ένα τυχαίο seed s και στη συνέχεια εφαρμόζοντας την συνάρτηση αυτή στην ακολουθία των τιμών s,s+1,s+2,....όπου έξοδος είναι η ακολουθία f(s),f(s+1),f(s+2),... .Ανάλογα με τις ιδιότητες της one-way συνάρτησης f μπορεί να είναι αναγκαίο να χρησιμοποιήσουμε μόνο μερικές τιμές της εξόδου f(s+i) προκειμένου να παρακάμψουμε τυχόν συσχετίσεις μεταξύ διαδοχικών τιμών.

Αν και τέτοιες ad-hoc [1] μέθοδοι δεν έχουν αποδειχθεί ότι είναι κρυπτογραφικά ασφαλείς φαίνονται να είναι επαρκείς για τις περισσότερες εφαρμογές.

4.5 Στατιστικά tests

Η υποενότητα αυτή παρουσιάζει κάποια tests για τη μέτρηση της ποιότητας μιας γεννήτριας υποθέτοντας ότι είναι μια γεννήτρια τυχαίων bits . Ενώ είναι αδύνατο να δοθεί μια μαθηματική απόδειξη ότι μια γεννήτρια είναι πράγματι μια γεννήτρια τυχαίων bit , τα test αυτά περιγράφουν εδώ τον εντοπισμό ορισμένων αδυναμιών που μπορεί να έχει μια γεννήτρια. Αυτό επιτυγχάνεται με τη λήψη δείγματος από τα εξαγώμενα της γεννήτριας και υποβάλλοντας τα δείγματα αυτά σε στατιστικές δοκιμές. Κάθε στατιστικό τεστ καθορίζει αν η ακολουθία αυτή έχει ένα συγκεχριμένο χαρακτηριστικό όπου μια πραγματικά τυχαία ακολουθία θα ήταν πιθανόν να παρουσιάσει-έχει .

[1] Στην επιστήμη ad-hoc μέθοδος είναι η προσθήκη ξένων υποθέσεων σε μια θεωρία για να τη σώσει από τυχόν παραποιήσεις

Το συμπέρασμα του κάθε τεστ δεν είναι οριστικό αλλά μάλλον πιθανοτικό-πιθανολογικό. Ένα παράδειγμα ενός τέτοιου χαρακτηριστικού είναι ότι η ακολουθία θα πρέπει να έχει περίπου τον ίδιο αριθμό από 0 και 1. Αν η ακολουθία αποτύχει σε κάποια από τα στατιστικά αυτά τεστ , τότε η γεννήτρια μπορεί να απορριφθεί ως μη τυχαία , εναλλακτικά η γεννήτρια μπορεί να υποβληθεί σε περαιτέρω τεστ. Από την άλλη , αν η ακολουθία περνά όλα τα στατιστικά τεστ , η γεννήτρια είναι αποδεκτή ως τυχαία. Πιο συγκεκριμένα ο όρος "αποδεκτές" πρέπει να αντικατασταθεί από δεν "απορρίπτεται" , αφού το ότι περνάει τα τεστ ορίζει απλώς κάποια πιθανοτικά στοιχεία που παράγει η γεννήτρια, τα οποία στοιχεία αυτά είναι ορισμένα χαρακτηριστικά των τυχαίων ακολουθιών. Η κανονική κατανομή (normal distribution) θεωρείται η σπουδαιότερη κατανομή της Θεωρίας Πιθανοτήτων και της Στατιστικής.

Οι λόγοι [8] που εξηγούν την εξέχουσα θέση της, είναι βασικά δύο:

i) Πολλές τυχαίες μεταβλητές περιγράφονται ικανοποιητικά από την κανονική κατανομή ή περιγράφονται από κατανομές που μπορούν να προσεγγισθούν από την κανονική κατανομή.

ii) Οι ιδιότητες της κανονικής κατανομής αξιοποιούνται στη Στατιστική Συμπερασματολογία. Ουσιαστικά, η κανονική κατανομή, αποτελεί το θεμέλιο της Στατιστικής Συμπερασματολογίας.

Ας δούμε τώρα **πέντε κύρια tests** που επιθυμούμε να ολοκληρώνουν με επιτυχία οι γεννήτριες .

Ας είναι s=$s_0, s_1, s_2, ..., s_{n-1}$ μια δυαδική ακολουθία μήκους n . Παρουσιάζουμε 5 στατιστικά τεστ που χρησιμοποιούνται συνήθως για να διαπιστωθεί εάν η δυαδική ακολουθία s διαθέτει κάποια χαρακτηριστικά που θα είχε και μια πραγματικά τυχαία ακολουθία. Τονίζεται και πάλι πως το αποτέλεσμα του κάθε τεστ δεν είναι οριστικό αλλά πιθανοτικό. Αν μια ακολουθία περνάει και τα πέντε τεστ , δεν δίνεται καμία εγγύηση ότι θα έχει πραγματικά παραχθεί από μια γεννήτρια τυχαίων bit .

(i) Frequency test (monobit test)

Σκοπός του τεστ αυτού είναι να καθορίσει αν ο αριθμός των 0 και 1 στην ακολουθία s είναι περίπου ίδιος , όπως αναμένεται να είναι για μια τυχαία ακολουθία. Ας είναι n_0 , n_1 ο αριθμός των 0 και 1 αντίστοιχα. Το στατιστικό στοιχείο που χρησιμοποιείται είναι :

$$X_1 = \frac{(n_0 - n_1)^2}{n}$$

το οποίο περίπου ακολουθεί την κατανομή X^2 με 1 βαθμό ελευθερίας , εάν n $\geq 10^7$.

(ii) Serial test (two-bit test)

Σκοπός του τεστ αυτού είναι να καθοριστεί αν ο αριθμός των εμφανίσεων των 00 , 01 , 10 και 11 ως υπακολουθίες του s είναι περίπου ίδιος , όπως θα αναμενόταν για μια τυχαία ακολουθία. Ας είναι n_0 , n_1 ο αριθμός των 0 και 1 στο s αντίστοιχα και n_{00} , n_{01} , n_{10} και n_{11} ο αριθμός των 00 , 01 , 10 , 11 στο

s αντίστοιχα. Ας σημειωθεί ότι είναι $n_{00} + n_{01} + n_{10} + n_{11} = (n-1)$ όσο οι υπακολουθίες επιτρέπεται να αλληλοκαλύπτονται. Το στατιστικό στοιχείο που χρησιμοποιείται είναι :

$$X_2 = \frac{4}{n-1}(n^2_{00} + n^2_{01} + n^2_{10} + n^2_{11}) - \frac{2}{n}(n^2_0 + n^2_1) + 1$$

το οποίο περίπου ακολουθεί την κατανομή X^2 με 2 βαθμό ελευθερίας , εάν n ≥ 21.

(iii) Poker test

Έστω m ένας θετικός ακέραιος τέτοιος ώστε $\lfloor \frac{n}{m} \rfloor \geq 5 \cdot (2^m)$ και ας είναι $k = \lfloor \frac{n}{m} \rfloor$. Χωρίστε την ακολουθία s σε k τμήματα μη επικαλυπτόμενα με μήκος m το καθένα και ας είναι n_i ο αριθμός των εμφανίσεων του $i^{ου}$ τύπου ακολουθίας μήκους m , $1 \leq i \leq 2^m$. Το poker test καθορίζει αν οι ακολουθίες μήκους m η καθεμία , εμφανίζονται περίπου τον ίδιο αριθμό φορών στην s όπως θα αναμενόταν και για μια τυχαία ακολουθία. Το στατιστικό στοιχείο που χρησιμοποιείται είναι :

$$X_3 = \frac{2^m}{k}(\sum_{i=1}^{2^m} n^2_i) - k$$

το οποίο περίπου ακολουθεί την κατανομή X^2 με 2^m - 1 βαθμό ελευθερίας . Ας σημειωθεί ότι το poker test είναι μια γενίκευση του frequency test . Για m=1 το poker test δίνει το frequency test .

(iv) Runs test

Σκοπός του τεστ αυτού είναι να καθορισθεί αν ο αριθμός των runs[2] με διαφορετικά μήκη στην ακολουθία s είναι περίπου ίδιος όπως αναμενόταν να είναι για μια τυχαία ακολουθία. Ο αναμενόμενος αριθμός των gaps (or blocks) για μήκος i σε μια τυχαία ακολουθία μήκους n είναι $e_i = (n - i + 3)/2^{i+2}$. Ας είναι το k ίσο με το μεγαλύτερο ακέραιο αριθμό i για τον οποίο $e_i \geq 5$. Ας είναι B_i , G_i ο αριθμός των blocks and gaps αντίστοιχα , για μήκος i μέσα στο s , για κάθε i , $1 \leq i \leq k$.

Το στατιστικό στοιχείο που χρησιμοποιείται είναι :

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - ei)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}$$

το οποίο περίπου ακολουθεί την κατανομή X^2 με $2k - 2$ βαθμό ελευθερίας.

(v) Autocorrelation test

Σκοπός του τεστ αυτού είναι να ελέγξει τις συσχετίσεις μεταξύ της ακολουθίας s και (μη κυκλικών) shifted versions της. Έστω d ένα σταθερός ακέραιος , $1 \leq d \leq \lfloor n/2 \rfloor$.Ο αριθμός των bits μέσα στην s που δεν είναι ίσα με τις s-shifts είναι A(d)=$\sum_{i=0}^{n-d-1} s_i \bigoplus s_{i+d}$ όπου $\bigoplus$ δηλώνει τον τελεστή XOR . Το στατιστικό στοιχείο που χρησιμοποιείται είναι :

$$X_5 = 2(A(d) - \tfrac{n-d}{2})/\sqrt{n-d}$$

το οποίο περίπου ακολουθεί την κατανομή N(0,1) εάν n - d $\geq$ 10 .

[2]Έστω s μια ακολουθία. Ένα run της s , είναι μια υποακολουθία της s που αποτελείται από συνεχόμενα 0 ή συνεχόμενα 1 , τα οποία δεν προηγούνται αλλά ούτε διαδέχονται το ίδιο σύμβολο. Ένα run από μηδενικά ονομάζεται gap ενώ ένα run από άσσους ονομάζεται block

4.6 Κρυπτογραφικά ασφαλείς ψευδογεννήτριες bit

Στις κρυπτογραφικά ασφαλείς γεννήτριες ψευδοτυχαίων bit (CSPRBG) η ασφάλεια της κάθε γεννήτριας εξαρτάται εξ' ολοκλήρου από την υποτιθέμενη επιλυσιμότητα ενός υποκείμενου αριθμο-θεωρητικού προβλήματος. Για παράδειγμα υπάρχουν γεννήτριες κρυπτογραφικά ασφαλείς που χρησιμοποιούν το αριθμο-θεωρητικό πρόβλημα των modular multiplications.

Ο Adi Shamir [9] βασίστηκε στο RSA κρυπτοσύστημα και έδωσε το πρώτο δημοσιευμένο παράδειγμα γεννήτριας ψευδοτυχαίων αριθμών η οποία είναι κρυπτογραφικά αρκετά ασφαλής.

Η RSA κρυπτογράφηση δημοσίου κλειδιού λειτουργεί με modulusN τα οποία απεικονίζουν το μυστικό cleartext M υπό το γνωστό δημοσιοποιημένο κλειδί K στα $M^K(modN)$. Η αντίστοιχη συνάρτηση αποκρυπτογράφησης ανακτά το cleartext παίρνοντας την κ-ιοστή ρίζα (root) του ciphertext (modN).Η κρυπτογραφική ασφάλεια του κρυπτοσυστήματος RSA είναι συνεπώς εξ ορισμού ισοδύναμη με τη δυσκολία υπολογισμού των ριζών modN .

Όταν το N είναι ένας μεγάλος και σύνθετος αριθμός με άγνωστη παραγοντοποίηση τότε το πρόβλημα εύρεσης ρίζας πιστεύεται ότι είναι πολύ δύσκολο, αλλά όταν η παραγοντοποίηση του N (Euler's notient function $\varphi(N)$) είναι γνωστή και το K είναι πρώτος σε σχέση με την $\varphi(N)$ υπάρχει ένας γρήγορος αλγόριθμος για την επίλυσή του.

5 Δυσκολία vs. Τυχαιότητα

Η ενότητα [3] αυτή είναι ουσιαστικά μια περιληπτική θεωρητική μελέτη του paper των Naom Nisan και Avi Widgerson (συμβ. N - W) με τίτλο "Hardness vs. Randomness "[1]. Παρουσιάζεται η κατασκευή των N - W μιας pseudorandom bit generator (PRG) η οποία παίρνει μια σειρά από πραγματικά τυχαία bits και τα μετατρέπει σε μια μεγάλη σειρά η οποία φαίνεται τυχαία σε κάθε αλγόριθμο από μια κλάση πολυπλοκότητας C χρησιμοποιώντας μια αυθαίρετη συνάρτηση , με συγκεκριμένα ορισμένη δυσκολία και συγκεκριμένο χρόνο εκτέλεσης , η οποία είναι hard στην κλάση αυτή C .

Το πιο σημαντικό αποτέλεσμα της γεννήτριας αυτής είναι ότι με τη βοήθειά της, η αποτελεσματική ντετερμινιστική προσομοίωση των τυχαιοποιημένων αλγορίθμων είναι επιτεύξιμη κάτω από ασθενέστερες παραδοχές από ότι ήταν γνωστές μέχρι προηγουμένως . Η αποτελεσματικότητα της προσομοίωσης εξαρτάται από την ισχύ των παραδοχών και ίσως επιτύχει P=BPP . Πιστεύετε με βάση τα αποτελέσματα των N-W πως το χάσμα μεταξύ τυχαιοποιημένης και ντετερμινιστικής πολυπλοκότητας δεν είναι μεγάλο .

5.1 Εισαγωγή

Σύμφωνα με την τρέχουσα κατάσταση των πραγμάτων υπάρχουν προβλήματα υπολογιστικά για τα οποία έχουμε πολυωνυμικούς τυχαιοποιημένους αλγο-

[3] Στην υποενότητα αυτή χρησιμοποιώ συχνά παραπομπές στη βιβλιογραφία για να τονίσω πως οι όποιες απλουστεύσεις των συμπερασμάτων-θεωρημάτων έγιναν με βάση τη βιβλιογραφία

ρίθμους και μονάχα εκθετικού χρόνου ντετερμινιστικούς αλγορίθμους. Δεν γνωρίζουμε πότε η τυχαιότητα μπορεί να βοηθήσει στο βαθμό εκτέλεσης ενός αλγορίθμου και αν ναι σε τι βαθμό. Οπότε μας απασχολεί το ποια είναι η δύναμη των τυχαιοποιημένων αλγορίθμων ή το γεγονός αν είναι πιο ισχυροί από τους ντετερμινιστικούς [10].

Γενικά στόχος είναι να πάρουμε έναν οποιονδήποτε τυχαιοποιημένο αλγόριθμο ο οποίος τρέχει σε πολυωνυμικό χρόνο και να τον προσομοιώσουμε χρησιμοποιώντας έναν ντετερμινιστικό αλγόριθμο , προσπαθώντας να μειώσουμε τον χρόνο εκτέλεσης του ντετερμινιστικού αλγορίθμου. Αυτού του είδους η προσομοίωση ονομάζεται "**derandomization of BPP** " [10].

Πώς κάνεις όμως έναν τυχαιοποιημένο αλγόριθμο ντετερμινιστικό ; [10] Απλώς τρέχεις τον αλγόριθμο αυτό, δίνοντας του ως είσοδο x , χρησιμοποιώντας όλα τα δυνατά κομμάτια μεγέθους r . Δεχόμαστε το x αν η πλειοψηφία των εκτελέσεων δέχεται το x . Αυτή η διαδικασία είναι και η προφανής συσχέτιση μεταξύ τυχαιότητας και χρόνου εκτέλεσης.

Σκοπός επομένως είναι να προσομοιωθεί η κλάση BPP χρησιμοποιώντας λιγότερο χρόνο. Αν μπορέσουμε να προσομοιώσουμε έναν τυχαιοποιημένο αλγόριθμο χρησιμοποιώντας λιγότερα τυχαία bits τότε θα καταφέρουμε να "derandomize "την κλάση BPP . Άρα στόχος είναι να ξεγελάσουμε τους BPP αλγορίθμους [10]. Ωστόσο , προκειμένου να επιτευχθεί αυτό, χρειαζόμαστε μια γεννήτρια που να ξεγελά τα boolean κυκλώματα (σε μέγεθος περίπου ίσο με το μέγεθος του χρόνου λειτουργίας των αλγορίθμων που θέλουμε να ξεγελάσουμε). Αυτό το εμπόδιο θα το αντιμετωπίζουμε συνεχώς μπροστά μας.

Δυστυχώς, μέχρι τώρα δεν υπάρχουν κατασκευές γεννητόρων που να ξεγελούν boolean πολυωνυμικά κυκλώματα. Ίσως βρούμε παρηγοριά στο γεγονός ότι το συστατικό που λείπει είναι το βασικό εμπόδιο που αντιμετωπίζουμε από τη θεωρία πολυπλοκότητας. Δηλαδή , την αποτυχία να αποδείξουμε την ύπαρξη hard functions , συναρτήσεων δηλαδή που δεν μπορούν να υπολογιστούν από πολυωνυμικού μεγέθους κυκλώματα. Λόγω αυτής της δυσκολίας λοιπόν όλα τα παρακάτω αποτελέσματα αποδεικνύουν την ακόλουθη μορφή [10]:

η ύπαρξη μιας **hard συνάρτησης , συνεπάγεται** ⇒

την ύπαρξη ενός **καλού γεννήτορα , συνεπάγεται** ⇒

ότι **BPP** ⊆ **υπο-εκθετικού χρόνου**

Πώς είναι δυνατόν να προσομοιώσουμε τους τυχαιοποιημένους αλγορίθμους όμως ; Ο Yao μας δείχνει πώς να χρησιμοποιούμε μια PG για να πετύχουμε την προσομοίωση [1].

Η προσομοίωση μπορεί να χωριστεί σε δύο στάδια. Αρχικά, ο original randomized αλγόριθμος που χρησιμοποιεί O(t) τυχαίας bits προσομοιώνεται από έναν randomized αλγόριθμο που χρησιμοποιεί $l(t^2)$ τυχαία bits. Αυτό γίνεται απλά τροφοδοτώντας τον original αλγόριθμο με pseudorandom ακολουθίες οι οποίες λαμβάνονται από την γεννήτορα PG , αντί για πραγματικά random bits.

Όσο τα εξαγόμενα του PG μοιάζουν τυχαία σε κάθε κύκλωμα μεγέθους t^2 , και όσο κάθε αλγόριθμος που τρέχει σε χρόνο t μπορεί να προσομοιωθεί από ένα κύκλωμα μεγέθους $l(t^2)$, τα εξαγόμενα του γεννήτορα θα δείχνουν τυχαία στον original αλγόριθμο.

Η πιθανότητα έτσι αποδοχής αυτού του randomized αλγορίθμου θα είναι σχεδόν ίδια με αυτή του αυθεντικού. Έπειτα προσομοιώνουμε αυτόν τον τυχαιοποιημένο αλγόριθμο ντετερμινιστικά, δοκιμάζοντας όλα τα πιθανά τυχαία seeds και παίρνουμε ένα μέσο όρο.

Αυτή είναι και η βασική ιδέα για να πετύχουμε την προσομοίωση. Όσο καλύτερους γεννήτορες έχουμε, τόσο καλύτερες hard συναρτήσεις κατασκευάζουμε, δηλαδή τόσο πιο καλές pseudorandom ακολουθίες παράγουμε και έστι επιτυγχάνουμε την καλύτερη δυνατή προσομοίωση των τυχαιοποιημένων αλγορίθμων.

Επίσης, ο Yao έδειξε πως κάθε one-way permutation μπορεί να χρησιμοποιηθεί για την κατασκευή γεννητόρων οι οποίοι ξεγελούν κάθε πολυωνυμικού χρόνου υπολογισμό. Αυτό το αποτέλεσμα έδωσε την πρώτη ουσιαστική συσχέτιση μεταξύ hardness και randomness.

Έπειτα οι Impagliazzo-Levin-Luby , σύμφωνα με την βοήθεια των παραπάνω κατάφεραν να κατασκευάσουν μια PG βασισμένοι στην υπόθεση ύπαρξης μιας αυθαίρετης one-way συνάρτησης. Οπότε αν υποθέσουμε πως πραγματικά υπάρχει μια one-way permutation είμαστε σε θέση να derandomize την κλάση BPP [10].

Οι N-W πετυχαίνουν το ίδιο αποτέλεσμα χρησιμοποιώντας μια πιο ασθενή υπόθεση για την δυσκολία. Δεν απαιτούν την υπόθεση ύπαρξης ούτε μιας one-way συνάρτησης αλλά ούτε την ύπαρξη μιας one-way permutation. Παίρνουν το ίδιο αποτέλεσμα χρησιμοποιώντας μια αυθαίρετη συνάρτηση σε μια κλάση C , με συγκεκριμένα ορισμένη δυσκολία και χρόνο εκτέλεσης.

5.2 Το εγχείρημα των Nisan-Wigderson

Οι N-W κατασκεύασαν έναν γεννήτορα ο οποίος υπολογίζει την συνάρτηση f , και ακολουθεί η παρακάτω διαδικασία : από μία τυχαία σειρά x_0(τον σπόρο), υπολογίζει την ακολουθία $\{X_i\}$όπου $X_i = \mathrm{f}(X_i)$.Τα εξαγόμενα bits b_i εξαρτώνται από την ακολουθία αυτή. Η καρδιά του εγχειρήματος είναι [1] να δειχθεί ότι αν ένα μικρό κύκλωμα το οποίο δεν μπορεί να ξεγελαστεί από την ακολουθία των bit $\{ b_i \}$, μπορεί να χρησιμοποιηθεί για να υπολογίσει την συνάρτηση f^{-1} διαψεύδοντας όμως έτσι την υποτιθέμενη δυσκολία της.

Δηλαδή, αν δεν καταφέρει η ακολουθία αυτή των $\{ b_i \}$ να ξεγελάσει ένα συγκεκριμένου μεγέθους μικρό κύκλωμα, δείχνουν πως αυτό σημαίνει πως δεν έχουμε έτσι μια πραγματικά hard συνάρτηση. Αλλά αν καταφέρνουμε να ξεγελάμε κάθε τέτοιο μικρό κύκλωμα, τότε έχουμε πράγματι μια hard συνάρτηση. Στη συνέχεια η ντετερμινιστική προσομοίωση του τυχαιοποιημένου αλγορίθμου προχωράει δοκιμάζοντας όλα τα δυνατά seeds X_0.

5.3 Η δυσκολία που απαιτείται για τον γεννήτορα

Η υπόθεση κάτω από την οποία κατασκευάστηκε ο γεννήτορας των N-W είναι η ύπαρξη μιας hard συνάρτησης. Τι απαιτήσεις όμως θέλουν οι N-W με τον όρο hard; Με τον όρο hard θέλουν όχι μόνο η συνάρτηση να μην μπορεί να υπολογιστεί από ένα μικρό κύκλωμα αλλά και να μην μπορεί να προσεγγιστεί από ένα κύκλωμα [1].

Το να προσεγγίσουμε μια συνάρτηση σημαίνει να συμφωνήσουμε με τη συνάρτηση αυτή σε ένα μεγάλο μέρος των εισόδων (δηλαδή σε αρκετές ίδιες εισόδους να συμφωνούν τα εξαγόμενά τους). Κατασκευάζουν λοιπόν το γεννήτορα χρησιμοποιώντας την υπόθεση πως υπάρχει μια συνάρτηση f η οποία δεν μπορεί να προσεγγιστεί από ένα πολυωνυμικό κύκλωμα. Υπάρχουν μάλιστα και δύο βασικές παράμετροι τις οποίες ορίζουν για τα κυκλώματα αυτά. Το μέγεθος των κυκλωμάτων από τα οποία δεν μπορεί να προσεγγιστεί η συνάρτηση αλλά και την ακρίβεια την προσομοίωσης που απαιτείται [1].

Πρόκειται για μια αδύναμη απαίτηση καθώς στον ορισμό της, απαιτεί μόνο τα μικρά κυκλώματα που επιχειρούν να υπολογίζουν την συνάρτηση f , να έχουν αμελητέο-σχεδόν μηδενικό ποσοστό σφάλματος. Θέλουν λοιπόν κανένα μικρό κύκλωμα να μην αποκτά κάποιο σημαντικό πλεονέκτημα στην προσπάθεια υπολογισμού της συνάρτησης f [1]. Αυτό είναι ένα από τα πιο σημαντικά σημεία του γεννήτορα των N-W. Ο γεννήτορας που κατασκεύασαν βασίζεται στη δυσκολία προσέγγισης μιας αυθαίρετης συνάρτησης της κλάσης EXPTIME.

5.4 Διαφορά σε σχέση με τις one-way functions

Η ύπαρξη μιας one-way συνάρτησης είναι ισοδύναμη με την κατασκευή μιας PG η οποία πρέπει να τρέχει σε πολυωνυμικό χρόνο.

Ας εξετάσουμε λίγο το τρόπο με τον οποίο χρησιμοποιούμε την υπόθεση της δυσκολίας [10] του γεννήτορα την οποία απαιτούμε. Χρειαζόμαστε μια συνάρτηση η οποία να είναι "εύκολη ", αφού η γεννήτρια πρέπει να υπολογίσει τη συνάρτηση f. Από την άλλη πλευρά , θα πρέπει να είναι hard συνάρτηση για τα πολυωνυμικά κυκλώματα, το οποίο συζητήσαμε στην παραπάνω υποενότητα.

Αφού ο γεννήτορας των N-W πρέπει να υπολογίσει τη συνάρτηση f οι N-W του επιτρέπουν να τρέχει σε εκθετικό χρόνο στο μέγεθος της εισόδου του. Αυτή είναι άλλωστε και η μεγαλύτερη διαφορά μεταξύ του γεννήτορα των N-W και των γεννητόρων που βασίζονται στην ύπαρξη μιας one-way συνάρτησης. Ο PG που προέρχεται από μια one-way συνάρτηση απαιτεί ο γεννήτορας να είναι υπολογίσιμος σε πολυωνυμικό χρόνο.

Εδώ το γεννήτορα, όχι μόνο τον αφήνουν οι N-W να τρέχει σε εκθετικό χρόνο, αλλά ζητάνε η f να είναι υπολογίσιμη σε εκθετικό χρόνο. Αυτό είναι μια πολύ πιο ασθενής απαίτηση. Αυτή η χαλάρωση επιτρέπει λοιπόν την κατασκευή μιας PG κάτω από ασθενέστερες συνθήκες και είναι πιο βολικό [10] . Η κατασκευή με το τρόπο αυτό αποδίδει μεν λιγότερο αποτελεσματικές PG αλλά η απώλεια αυτή, δεν έχει καμία επιρροή όταν χρησιμοποιούμε αυτή την PG για ντετερμινιστική προσομοίωση των τυχαιοποιημένων αλγορίθμων [1] .

5.5 Βασικό Συμπέρασμα

Τα βασικό λοιπόν συμπέρασμα της παραπάνω γεννήτριας είναι πως χρησιμοποιώντας την υπόθεση ότι υπάρχει μια συνάρτηση η οποία μπορεί να υπολογιστεί σε εκθετικό χρόνο και δεν μπορεί να προσεγγιστεί από πολυωνυμικά κυκλώματα τότε μπορούμε να καταφέρουμε να πετύχουμε τη καλύτερη δυνατή ντετερμινιστική προσομοίωση των τυχαιοποιημένων αλγορίθμων [10]. Πρέπει να δώσουμε μεγάλη προσοχή στη διαφορά των λέξεων "να υπολογιστεί "και "να προσεγγιστεί ".

Μιλώντας κάπως ανεπίσημα [11], αυτό δηλώνει δηλαδή ότι από μια hard συνάρτηση (όπως την απαιτήσαμε παραπάνω) η οποία είναι υπολογίσιμη σε εκθετικό χρόνο , μπορεί κάποιος να κατασκευάσει μια pseudorandom γεννήτρια G η οποία να "επεκτείνει "ένα μικρό σπόρο σε αρκετά bits , του οποίου η έξοδος να μην μπορεί να διακριθεί από μια πραγματικά τυχαία γεννήτρια από ένα μικρό κύκλωμα. Έπειτα χρειάζεται μόνο να απαριθμηθούν όλοι οι σπόροι της γεννήτρια αυτής G , το οποίο παίρνει πολυωνυμικό χρόνο και να τροφοδοτήσουμε τον τυχαιοποιημένο αλγόριθμο που θέλουμε να προσεγγίσουμε με τις εξόδους της γεννήτριας αυτής G . Ως εκ τούτου , θα μπορούσαμε να επιτύχουμε P=BPP .

References

[1] Naom Nisan & Avi Wigderson *Hardness vs. Randomness.* University of California at Berkeley (1994).

[2] Sipser Michael *Εισαγωγή στη θεωρία υπολογισμού.* Ηράκλειο (2009).

[3] Σ.Μποζαπαλίδης *Μαθηματικά και Πληροφορική* Θεσσαλονίκη (1997).

[4] Stephen Cook *The P versus NP Problem*
http : //www.claymath.org/millennium/P_vs_NP/pvsnp.pdf

[5] Salil Vadhan *The Power of Randomness*
http : //people.seas.harvrad.edu/~sali/pseudorandomness/power.pdf

[6] Gregory Chaitin *Μετα-μαθηματικά , τα μυστικά του αριθμού Ω.* (2007).

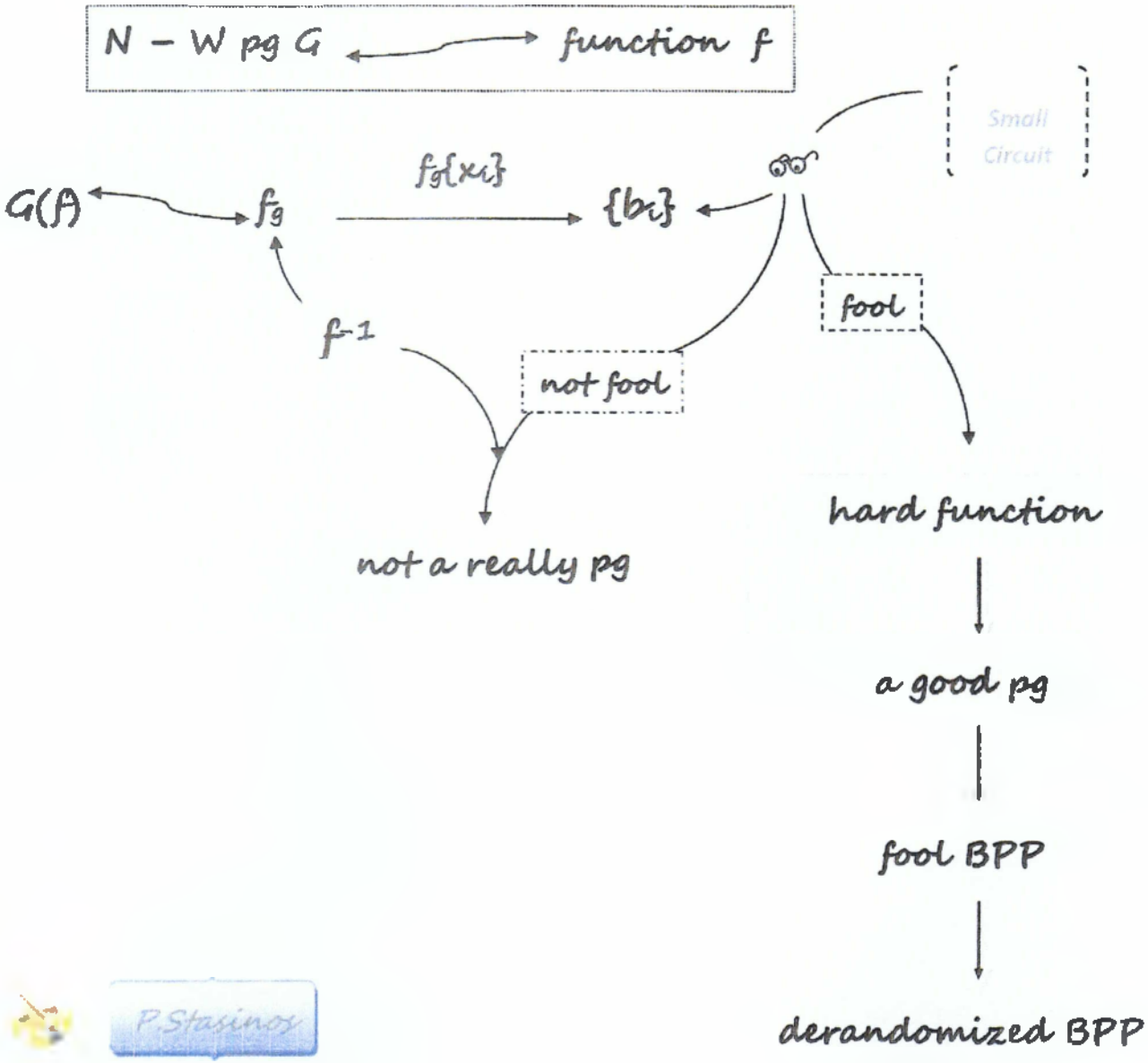
[7] A. Menezes & P. van Oorschot and S. Vanstone *Handbook of Applied Cryptography.* CRC Press (1996).

[8] Παναγόπουλος Γ. *Κανονική Κατανομή www.aua.gr/gpapadopoulos*

[9] Adi Shamir *On the generation of Cryptographically Strong Pseudorandom Sequences.* The Weizmann Institute of Science (1983).

[10] Avi Wigderson *Derandomizing BPP.*
http : //www.math.ias.edu/~avi/BOOKS/rand.pdf

[11] Madhu Sudan *Advanced Complexity Theory.*
http : //people.csail.mit.edu/madhu/ST05/


N – W pg G
function f
Small Circuit
G(A)
f_g
$f_g\{x_i\}$
$\{b_i\}$
f-1
not fool
fool
not a really pg
hard function
a good pg
fool BPP
derandomized BPP
P.Stasinos