Πιθανοθεωρητικοί Αλγόριθμοι: Τυχαίοι Αριθμοί, Δειγματοληψία

Αν. Καθηγητής Π. Λουρίδας

Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας,
Οικονομικό Πανεπιστήμιο Αθηνών
louridas@aueb.gr

Επισκόπηση

Γενικά

- Συνήθως ο παράγοντας τύχη είναι κάτι που προσπαθούμε να ελαχιστοποιήσουμε.
- Θεωρείται καλό να μπορούμε να πούμε ότι «δεν αφήσαμε τίποτε στην τύχη».
- Και όμως, όπως θα δούμε, μπορούμε να λύσουμε σημαντικά προβλήματα ακριβώς χρησιμοποιώντας τον παράγοντα τύχη.

Τυχαιότητα

- Κάτι είναι τυχαίο όταν είναι μη προβλέψιμο (unpredictable).
- Η ρίψη ενός νομίσματος είναι τυχαία, αφού υπάρχουν 50–50 πιθανότητες να έρθει κορώνα ή γράμματα.
- Η *τυχαιότητα* (randomness) είναι η απουσία κανονικότητας σε μια σειρά γεγονότων ή δεδομένων.

Τυχαίο

- Λέμε ότι κάτι είναι *τυχαίο* (random) όταν δεν υπάρχει καμμία κανονικότητα σε αυτό, όπως και αν το παρατηρήσουμε.
- Ο λευκός θόρυβος είναι τυχαίος.
- Οι ζαριές ενός δίκαιου ζαριού είναι τυχαίες.
- Η κίνηση Brown είναι τυχαία.

Δείγματα

- Σε μία δημοσκόπηση ερευνούμε ένα *δείγμα* (sample) του συνολικού πληθυσμού.
- Θέλουμε να έχουμε ένα *αντιπροσωπευτικό δείγμα* (representative sample).
- Αυτό δεν είναι πάντα εύκολο να το βρούμε.
- Για παράδειγμα, το *σφάλμα επιβίωσης* (survivorship bias) συμβαίνει όταν το δείγμα μας περιλαμβάνει μόνο εκείνα τα άτομα που εξακολουθούν να είναι παρόντα.
- Για να αποφύγουμε τέτοια προβλήματα, προσπαθούμε να δουλεύουμε με ένα *τυχαίο δείγμα* (random sample).

Πιθανοθεωρητικοί Αλγόριθμοι

- Για να μπορέσουμε να πάρουμε ένα τυχαίο δείγμα, θα πρέπει να χρησιμοποιήσουμε έναν αλγόριθμο οποίος χρησιμοποιεί κάπου τυχαιότητα.
- Αφού ο αλγόριθμος λειτουργεί σε κάποιο βαθμό τυχαία, θα είναι ένας *πιθανοθεωρητικός αλγόριθμος*.
- Ένας πιθανοθεωρητικός αλγόριθμος φυσικά δεν θα δουλεύει πάντα με τον ίδιο τρόπο.
- Το ερώτημα λοιπόν είναι: μπορούμε να έχουμε πιθανοθεωρητικούς αλγόριθμους οι οποίοι να λειτουργούν καλά και να γνωρίζουμε και πόσες είναι οι πιθανότητες να μη λειτουργούν καλά;
- Μπορούμε να ρυθμίζουμε τις παραμέτρους των αλγορίθμων αυτών ώστε να ελέγχουμε τη συμπεριφορά τους;

Επισκόπηση

Τυχαιότητα στον Υπολογιστή;

- Οι πιθανοθεωρητικοί αλγόριθμοι λειτουργούν με τυχαιότητα, η οποία έχει τη μορφή *τυχαίων αριθμών* (random numbers).
- Αλλά πώς μπορούμε να παράξουμε τυχαίους αριθμούς στον υπολογιστή;
- Προφανώς, ένας ντετερμινιστικός αλγόριθμος αποκλείεται να έχει τυχαίο αποτέλεσμα.

  Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin.—John von Neumann, 1951

Γεννήτριες Τυχαίων Αριθμών

- Τυχαίοι αριθμοί παράγονται από *γεννήτριες τυχαίων αριθμών* (random number generators).
- Οι μόνες γεννήτριες πραγματικά τυχαίων αριθμών (True Random Number Generators, TRNGs) λειτουργούν με βάση κάποια τυχαία φυσική διαδικασία.
- Μπορεί να είναι ένας μετρητής Geiger, ένας ανιχνευτής φωτονίων σε ένα ημιδιάφανο καθρέπτη, ανιχνευτής ραδιοφωνικού θορύβου στην ατμόσφαιρα, κ.λπ.
- Μία γεννήτρια πραγματικά τυχαίων αριθμών μπορεί να ενσωματωθεί σε έναν υπολογιστή.
- Παρ' όλα αυτά, δεν είναι πάντοτε διαθέσιμη, και δεν μπορεί πάντα να παράξει τυχαίους αριθμούς με το ρυθμό που τους χρειαζόμαστε.

Γεννήτριες Ψευδοτυχαίων Αριθμών

- Αν δεν έχουμε μια γεννήτρια πραγματικά τυχαίων αριθμών, πρέπει να χρησιμοποιήσουμε μια *γεννήτρια ψευδοτυχαίων αριθμών* (Pseudorandom Number Generator, PRNG).
- Μία γεννήτρια ψευδοτυχαίων αριθμών παράγει αριθμούς που *φαίνονται* τυχαίοι, ακόμα και αν στην πραγματικότητα δεν είναι.

Ψευδοτυχαίοι Αριθμοί και Ομοιόμορφη Κατανομή

- Μια κοινή απαίτηση είναι οι ψευδοτυχαίοι αριθμοί που παράγονται να ακολουθούν την *ομοιόμορφη κατανομή* (uniform distribution).
- Ένα σύνολο αριθμών ακολουθεί την ομοιόμορφη κατανομή αν ο κάθε ένας από αυτούς εμφανίζεται στο σύνολο με την ίδια πιθανότητα.
- Αυτό όμως δεν είναι αρκετό για τους σκοπούς μας.
- Για παράδειγμα, οι αριθμοί:

$$1, 2, \ldots, 10, 1, 2, \ldots, 10, 1, 2, \ldots, 10, \ldots$$

ακολουθούν την ομοιόμορφη κατανομή, αλλά δεν φαίνονται καθόλου τυχαίοι.

Έλεγχοι Τυχαιότητας

- Για το σκοπό αυτό, θέλουμε οι αριθμοί που παράγονται από μια γεννήτρια ψευδοτυχαίων αριθμών να περνούν συγκεκριμένους *στατιστικούς ελέγχους*.
- Οι έλεγχοι αυτοί διαπιστώνουν κατά πόσο μια ακολουθία αριθμών δεν έχει τα χαρακτηριστικά μιας πραγματικά τυχαίας ακολουθίας αριθμών.

Μέθοδος Γραμμικού Υπολοίπου

- Μια μέθοδος που χρησιμοποιείται εδώ και αρκετά χρόνια είναι η *μέθοδος του γραμμικού υπολοίπου* (linear congruential method).
- Η μέθοδος αυτή παράγει μια σειρά αριθμών με τον τύπο:

$$X_{n+1} = \left(aX_n + c\right) \bmod m, \quad n \geq 0$$

- Η ακολουθία ξεκινά δίνοντάς στον τύπο έναν αρχικό αριθμό X_0 ο οποίος ονομάζεται *σπόρος* (seed).

Αλγόριθμος Γραμμικού Υπολοίπου

Algorithm: Linear Congruential Random Number Generator.

`LinearCongruential`$(x) \longrightarrow r$

Input: x, a number $0 \leq x < m$

Data: m the modulus, $m > 0$

a the multiplier, $0 < a < m$

c the increment, $0 < c < m$

Output: r, a number $0 \leq r < m$

1 $r \leftarrow (a \times x + c) \bmod m$

2 **return** r

Υλοποίηση σε Γλώσσες Προγραμματισμού

- Στις υλοποιήσεις της μεθόδου σε γλώσσες προγραμματισμού, δεν περνάμε το x σε κάθε κλήση.
- Συνήθως προσφέρονται δύο συναρτήσεις.
- Μια συνάρτηση `Seed(s)`, η οποία δίνει τον σπόρο στη μέθοδο.
- Μια συνάρτηση `Random()`, η οποία εκτελεί τη μέθοδο και κρατά το αποτέλεσμα της κλήσης σε μία κρυφή μεταβλητή, ώστε να χρησιμοποιηθεί στην επόμενη κλήση.

Η Σημασία του Σπόρου

- Είναι προφανές ότι οι ψευδοτυχαίοι αριθμοί που θα παραχθούν εξαρτώνται από την αρχική τιμή του x, τον σπόρο.
- Αυτό δεν είναι κακό, γιατί μας επιτρέπουν να μετατρέψουμε ένα πιθανοθεωρητικό πρόγραμμα σε απολύτως ντετερμινιστικό.
- Αυτό είναι χρήσιμο όταν κάνουμε ελέγχους στο πρόγραμμα, οπότε πρέπει να ξέρουμε ακριβώς τι τιμές να περιμένουμε και δεν θέλουμε κάθε φορά να συμπεριφέρεται διαφορετικά.

Οι Παράμετροι a, m, c

- Στο $X_{n+1} = (aX_n + c) \bmod m$ οι τιμές των παραμέτρων a, m, c είναι σημαντικές.
- Η μέθοδος δεν μπορεί να μας δώσει περισσότερους από m ψευδοτυχαίους αριθμούς.
- Αν κάποια στιγμή δώσει έναν αριθμό που τον έχει ήδη δώσει προηγουμένως, θα αρχίσει να επαναλαμβάνει τις προηγούμενες τιμές.
- Στην πραγματικότητα, η μέθοδος έχει μια *περίοδο*.
- Πρέπει να επιλέξουμε τα a, m, c ώστε η περίοδος να είναι όσο μεγαλύτερη γίνεται (αν είναι δυνατόν m) και οι αριθμοί να ακολουθούν την ομοιόμορφη κατανομή.

Οι Παράμετροι a, m, c

- Αν για παράδειγμα δώσουμε: $s = 0$, $m = 10$, $a = 3$, και $c = 3$, θα πάρουμε:

$$3, 2, 9, 0, 3, 2, 9, 0, \ldots$$

- Για να επιτύχουμε τη μέγιστη δυνατή περίοδο, ίση με m, τα a, m, c πρέπει να πληρούν τις ακόλουθες προϋποθέσεις:
  1. Τα m και c πρέπει να είναι πρώτοι μεταξύ τους.
  2. Το $a - 1$ πρέπει να διαιρείται από όλους τους πρώτους παράγοντες του m.
  3. Το $a - 1$ θα πρέπει να διαιρείται με το 4, αν το m διαιρείται με το 4.
- Συνήθως επιλέγουμε κάποιους αριθμούς που είναι γνωστό ότι πληρούν τις προϋποθέσεις, όπως $s = 2^{32}$, $a = 32310901$, c περιττός, και m μια δύναμη του 2.

Εύρος Τιμών

- Η μέθοδος γραμμικού υπολοίπου επιστρέφει αριθμούς μεταξύ του 0 και του $m-1$, δηλαδή στο διάστημα $[0, m-1]$.
- Αν θέλουμε ψευδοτυχαίους αριθμούς με ένα άλλο εύρος τιμών, για παράδειγμα $[0, k]$, μπορούμε να πολλαπλασιάσουμε το αποτέλεσμα με $k/(m-1)$.
- Αν θέλουμε αριθμούς από το 0 μέχρι και το 1, διαιρούμε με το $m-1$.
- Συχνά θέλουμε αριθμούς στο διάστημα $[0, 1)$, οπότε διαιρούμε με το m.

Ο Αλγόριθμος xorshift64*

- Τα τελευταία χρόνια έχουν αναπτυχθεί και άλλοι αλγόριθμοι παραγωγής ψευδοτυχαίων αριθμών οι οποίοι περνούν περισσότερους ελέγχους τυχαιότητας από τη μέθοδο γραμμικού υπολοίπου.
- Ένας από αυτούς, που έχει το πλεονέκτημα να είναι πολύ γρήγορος, είναι ο xorshift64* (διαβάζεται xor shift 64 star).
- Όπως και η μέθοδος του γραμμικού υπολοίπου, δίνει έναν νέο ψευδοτυχαίο αριθμό κάθε φορά που καλείται, με είσοδο τον προηγούμενο ψευδοτυχαίο. Την πρώτη φορά του δίνουμε ένα σπόρο που πρέπει να είναι διαφορετικός από το μηδέν.
- Ο αλγόριθμος xorshift64* παράγει ψευδοτυχαίους αριθμούς των 64 bits.

Ο Αλγόριθμος xorshift64*

Algorithm: xorshift64*.

`XORShift64Star`$(x) \longrightarrow r$

Input: x, a 64-bit integer different than 0

Output: r, a 64-bit number

1 $x \leftarrow x \oplus (x \gg 12)$

2 $x \leftarrow x \oplus (x \ll 25)$

3 $x \leftarrow x \oplus (x \gg 27)$

4 $r \leftarrow x \times 2685821657736338717$

5 **return** r

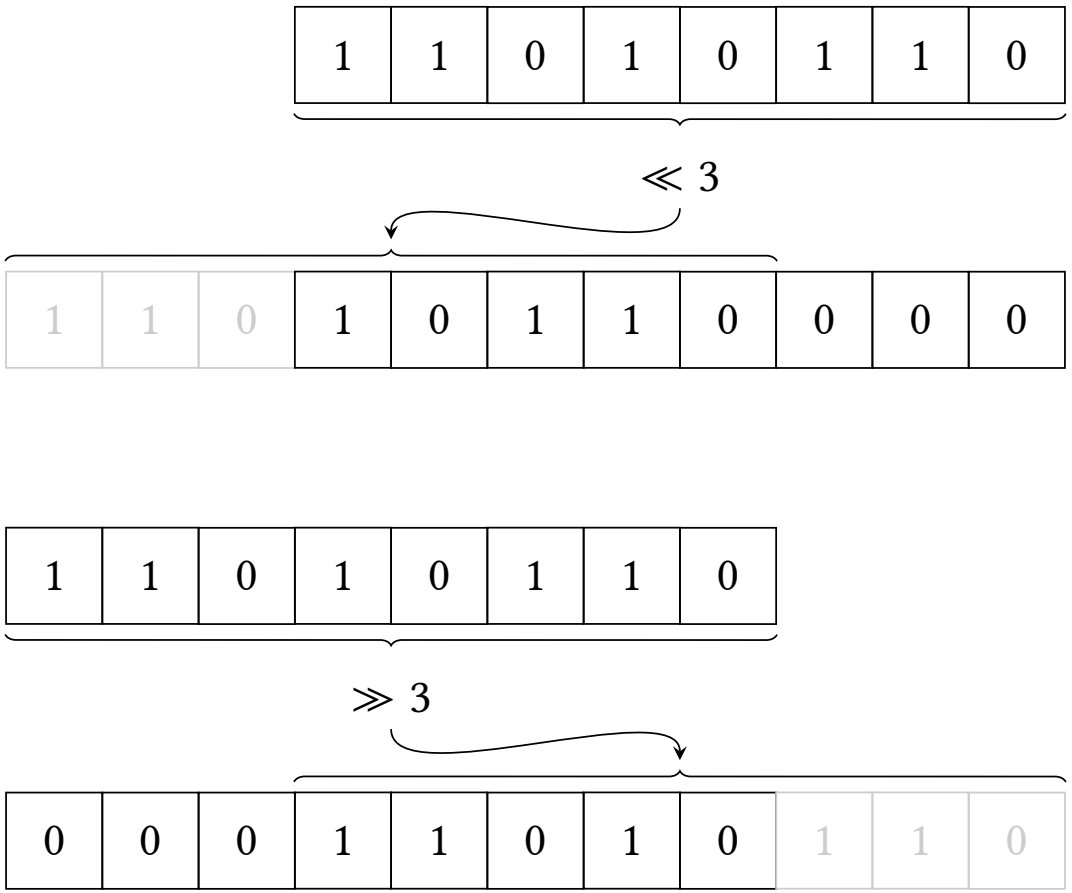
Μαγικοί Αριθμοί

- Ο αριθμός 26858216577736338717 που εμφανίζεται στον αλγόριθμο είναι ένας *μαγικός αριθμός* (magic number).
- Μαγικός αριθμός είναι ένας αριθμός ο οποίος εμφανίζεται σε ένα πρόγραμμα χωρίς να υπάρχει ερμηνεία για την τιμή του.
- Γενικώς μαγικοί αριθμοί πρέπει να αποφεύγονται, ή να τεκμηριώνονται επαρκώς.
- Στη συγκεκριμένη περίπτωση ο αριθμός αυτός έχει επιλεγεί διότι προσδίδει στο αποτέλεσμα καλά χαρακτηριστικά τυχαιότητας.

Τελεστές Ολίσθησης (Shift Operators)

- Ο αλγόριθμος xorshift64* χρησιμοποιεί τους τελεστές ολίσθησης $\ll$ και $\gg$.
- Η παράσταση $x \ll a$ σημαίνει ότι θα ολισθήσουμε τα bits του x προς τα αριστερά κατά a θέσεις, για παράδειγμα $1110 \ll 2 = 1000$.
- Η παράσταση $x \gg a$ σημαίνει ότι θα ολισθήσουμε τα bits του x προς τα δεξιά κατά a θέσεις, για παράδειγμα $1101 \gg 2 = 0011$.

Τελεστές Ολίσθησης (Shift Operators)



Ο Αλγόριθμος xorshift64* και ο Αλγόριθμος xorshift1024*

- Ο αλγόριθμος xorshift64* είναι πολύ γρήγορος.
- Επίσης παράγει αριθμούς με καλά χαρακτηριστικά τυχαιότητας.
- Η περίοδός του είναι $2^{64} - 1$.
- Αν θέλουμε ακόμα μεγαλύτερη περίοδο, μπορούμε να χρησιμοποιήσουμε το μεγάλο αδερφάκι του, τον αλγόριθμο xorshift1024*.

Ο Αλγόριθμος xorshift1024*

Algorithm: xorshift1024*.

`XORShift1024Star`$(S) \rightarrow r$

Input: S, an array of 16 unsigned 64-bit integers

Data: p, a number initially set to 0

Output: r, a 64-bit random number

1 $s_0 \leftarrow S[p]$

2 $p \leftarrow (p + 1)\ \&\ 15$

3 $s_1 \leftarrow S[p]$

4 $s_1 \leftarrow s_1 \oplus (s_1 \ll 31)$

5 $s_1 \leftarrow s1 \oplus (s_1 \gg 11)$

6 $s_0 \leftarrow s_0 \oplus (s_0 \gg 30)$

7 $S[p] \leftarrow s_0 \oplus s1$

8 $r \leftarrow S[p] \times 1181783497276652981$

9 **return** r

Ο Αλγόριθμος xorshift1024*

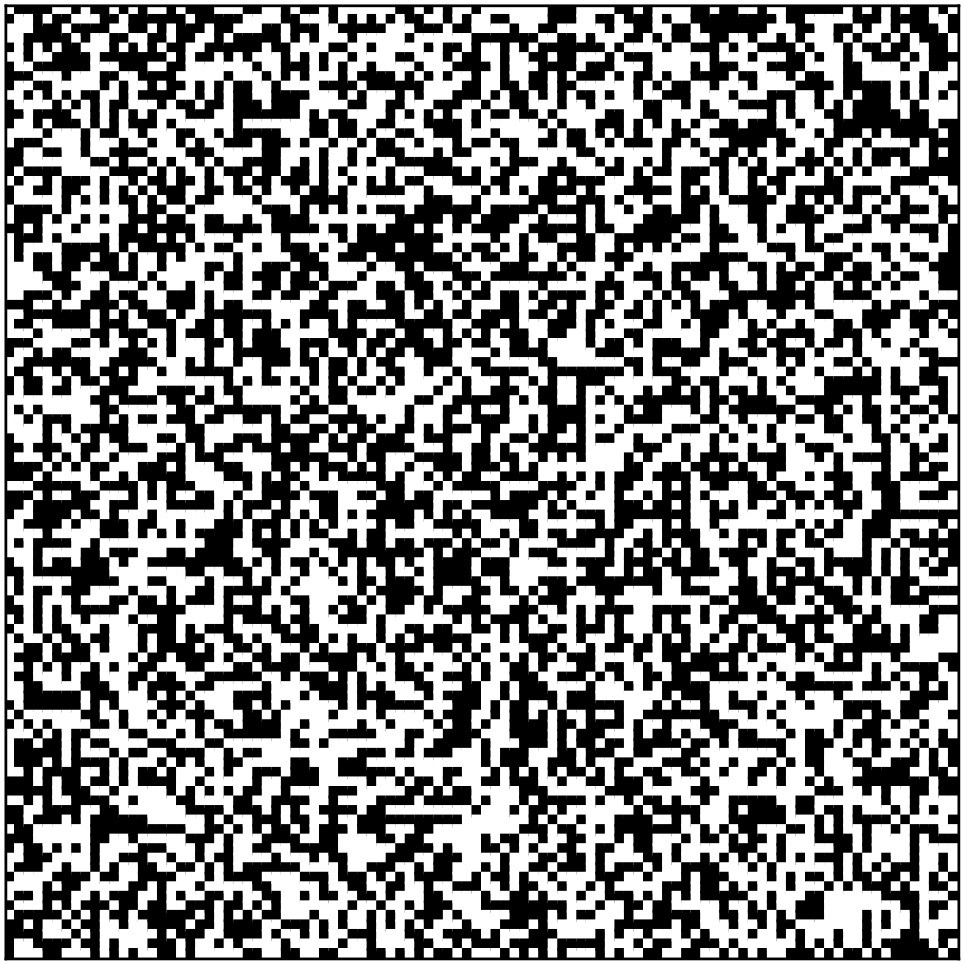
- Συγκρινόμενος με τον xorshift64*, ο xorshift1024* έχει πολύ μεγαλύτερη περίοδο: $2^{1024} - 1$.
- Επιπλέον, οι αριθμοί που παράγει περνούν ακόμα περισσότερους στατιστικούς ελέγχους τυχαιότητας.
- Ο σπόρος του είναι οι αρχικοί 16 αριθμοί στον πίνακα S, οι οποίοι συστήνεται να έχουν παραχθεί από τον xorshift64*. Προσέξτε ότι τα περιεχόμενα του S ($S[p]$) αλλάζουν με κάθε εκτέλεση του αλγόριθμου.
- Το όνομα του αλγόριθμου προκύπτει από το γεγονός ότι χρησιμοποιεί έναν πίνακα 16 θέσεων, η κάθε μία από τις οποίες έχει 64 bits, άρα $16 \times 64 = 1024$.
- Ενδιαφέρον έχει η γραμμή 2 του αλγορίθμου, με την οποία εξασφαλίζεται ότι το p θα λαμβάνει τις τιμές από 0 μέχρι και 15.

Σχεδιασμός Γεννητριών Τυχαίων Αριθμών

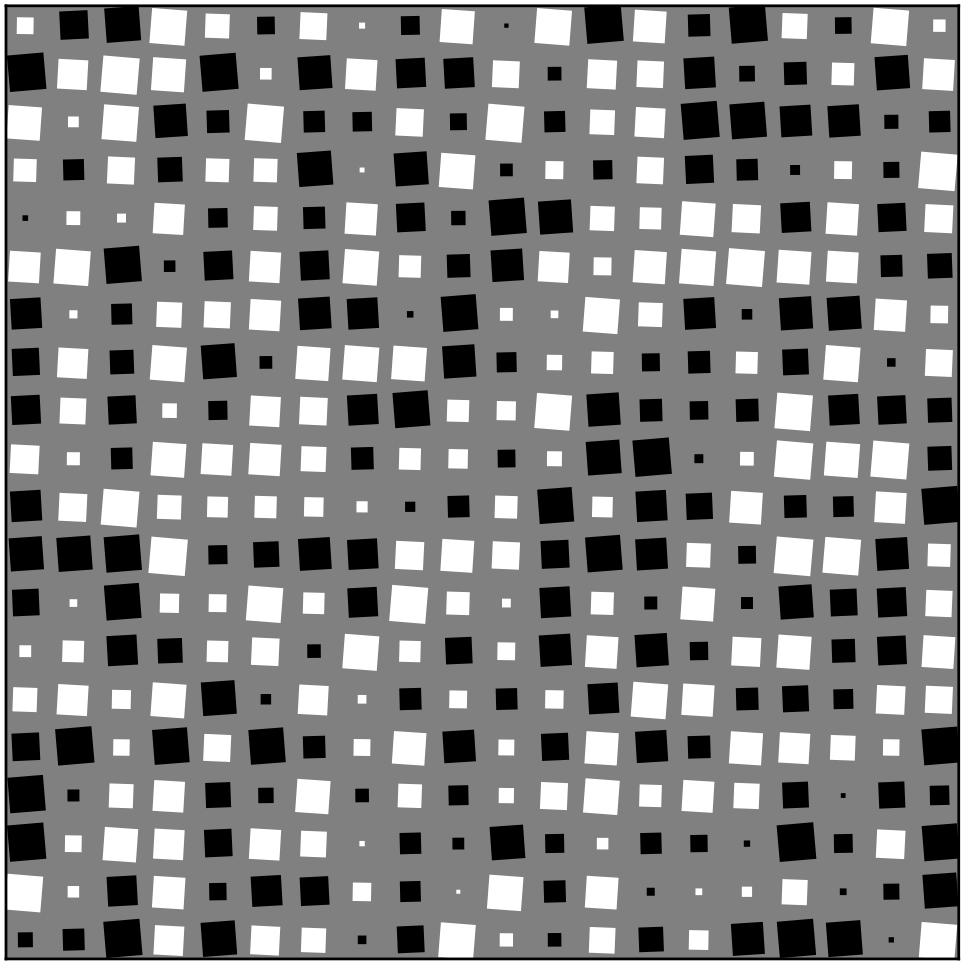
- Όπως μπορούμε να δούμε, η δημιουργία έστω ψευδοτυχαίων αριθμών δεν είναι καθόλου εύκολη υπόθεση.
- Θέλει πολύ δουλειά και εντατικούς ελέγχους.

  random numbers should not be generated with a method chosen at random—Donald Knuth

10.000 Τυχαία Bits



Μια Τυχαία Εικόνα



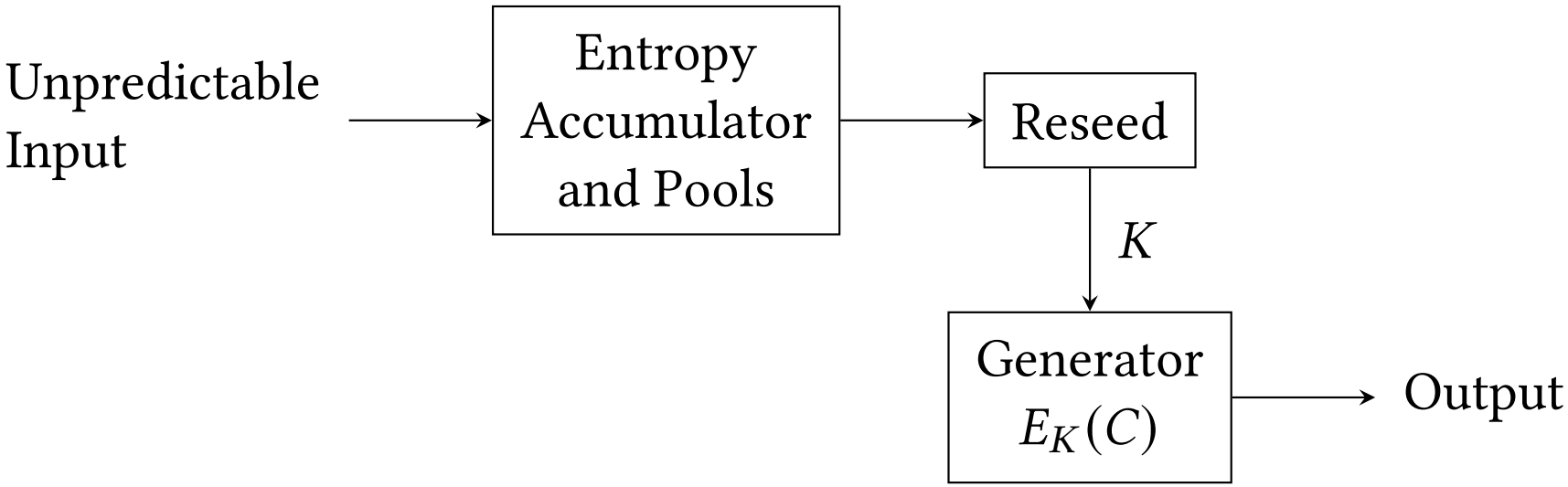
Τυχαίοι Αριθμοί και Κρυπτογραφία

- Οι γεννήτριες τυχαίων αριθμών που έχουμε δει μέχρι τώρα *δεν είναι κατάλληλες για κρυπτογραφική χρήση.*
- Σε πολλά κρυπτογραφικά πρωτόκολλα χρειάζονται τυχαίοι αριθμοί, είτε ως κλειδιά είτε ως μοναδικοί κωδικοί μιας χρήσης (nonces), κ.λπ.
- Για να είναι οι τυχαίοι αριθμοί κατάλληλοι για κρυπτογραφική χρήση πρέπει:
  - Να περνούν στατιστικούς ελέγχους τυχαιότητας.
  - Να αντέχουν σε επιθέσεις: να μην υπάρχει πολυωνυμικός αλγόριθμος ο οποίος να μπορεί να αρχίσει να προβλέπει τους επόμενους αριθμούς, ή που να μπορεί να βρει ποιοι αριθμοί έχουν ήδη παραχθεί.

Κρυπτογραφικά Ασφαλείς Γεννήτριες Ψευδοτυχαίων Αριθμών

- Στην κρυπτογραφία χρησιμοποιούμε ειδικές γεννήτριες τυχαίων αριθμών.
- Οι γεννήτριες αυτές ονομάζονται Κρυπτογραφικά Ασφαλείς Γεννήτριες Ψευδοτυχαίων Αριθμών (Cryptographically Secure Pseudorandom Number Generators—CSPNGs).

Η Γεννήτρια Ψευδοτυχαίων Αριθμών Fortuna



Επισκόπηση

Απλοϊκή Μέθοδος (Λάθος)

- Έστω ότι έχουμε ένα πληθυσμό μεγέθους n και θέλουμε να πάρουμε τυχαία m μέλη του πληθυσμού.
- Μια απλή ιδέα είναι να εξετάσουμε κάθε μέλος και να το βάλουμε στο δείγμα μας με πιθανότητα m/n.
- Αυτό είναι *λάθος*.
- Με τον τρόπο αυτό θα πάρουμε ένα δείγμα μεγέθους m *κατά μέσο όρο*, όχι *κάθε φορά*.

Βασική Ιδέα

- Έστω ότι έχουμε ήδη επιλέξει κάποια αντικείμενα.
- Εξετάζουμε το επόμενο στη σειρά, και θέλουμε να δούμε με ποια πιθανότητα θα το επιλέξουμε.
- Αν έχουμε ήδη επιλέξει πολλά, και μας μένουν πολλά να εξετάσουμε, η πιθανότητα να το επιλέξουμε θα πρέπει να είναι *μικρή*.
- Αν έχουμε επιλέξει λίγα και μας μένουν λίγα να εξετάσουμε, η πιθανότητα να το επιλέξουμε θα πρέπει να είναι *μεγάλη*.
- Άρα θέλουμε η πιθανότητα επιλογής εντός αντικειμένου να προσαρμόζεται αναλόγως με τα πόσα μας μένουν και πόσα έχουμε επιλέξει.

Δειγματοληψία με Επιλογή

- Έστω ότι έχουμε εξετάσει t αντικείμενα και έχουμε επιλέξει k από αυτά για το δείγμα μας.
- Αυτό σημαίνει ότι ακόμα δεν έχουμε εξετάσει $n - t$ αντικείμενα.
- Επίσης πρέπει να προσθέσουμε στο δείγμα μας $m - k$ αντικείμενα.
- Με πόσους δυνατούς τρόπους μπορεί να εξελιχθεί η επιλογή;

Δειγματοληψία με Επιλογή

- Ο αριθμός των τρόπων με τους οποίους μπορεί να εξελιχθεί η επιλογή, έστω w_1, είναι ο αριθμός των τρόπων με τους οποίους μπορούμε να επιλέξουμε $m-k$ αντικείμενα από $n-t$ αντικείμενα:

$$w_1 = \binom{n-t}{m-k}$$

- Σκεπτόμενοι με τον ίδιο τρόπο, αν έχουμε εξετάσει $t+1$ αντικείμενα και έχουμε επιλέξει $k+1$ αντικείμενα, η διαδικασία μπορεί να εξελιχθεί με:

$$w_2 = \binom{n-t-1}{m-k-1}$$

διαφορετικούς τρόπους.

Δειγματοληψία με Επιλογή

- Συνεπώς η πιθανότητα να πάμε από το t και το k στο $t+1$ και το $k+1$ είναι:

$$\frac{w_2}{w_1} = \binom{n-t-1}{m-k-1} \Big/ \binom{n-t}{m-k} = \frac{m-k}{n-t}$$

- Συνεπώς, αν έχουν επιλεγεί τα k αντικείμενα από τα πρώτα t, το $k+1$ αντικείμενο θα πρέπει να επιλεγεί με πιθανότητα $(m-k)/(n-t)$.
- Ο αλγόριθμος που προκύπτει από αυτόν τον κανόνα ονομάζεται *δειγματοληψία με επιλογή* (selection sampling).

Αλγόριθμος Δειγματοληψίας με Επιλογή

Algorithm: Selection sampling.

`SelectionSampling`$(P, m) \rightarrow S$

Input: P, an array containing the items of the population
m the number of items to select

Output: S, an array with m randomly selected items from P

```
1  S ← CreateArray(m)
2  k ← 0
3  t ← 0
4  n ← |P|
5  while k < m do
6      u ← Random(0, 1)
7      if u × (n − t) < (m − k) then
8          S[k] ← P[t]
9          k ← k + 1
10     t ← t + 1
11 return S
```

Συνολική Πιθανότητα Επιλογής Αντικειμένου

- Είδαμε ότι η πιθανότητα να επιλέξουμε το $k+1$ αντικείμενο αφού έχουμε επιλέξει k από τα πρώτα t είναι $(m-k)/(n-t)$.
- Μπορεί να αποδειχθεί ότι η *συνολική πιθανότητα* (overall probability) με την οποία μπορεί να επιλεγεί ένα αντικείμενο είναι ακριβώς m/n.
- Πώς από το $(m-k)/(n-t)$ φτάνουμε στο m/n;
- Το $(m-k)/(n-t)$ είναι *δεσμευμένη πιθανότητα* (conditional probability): είναι η πιθανότητα να επιλεγεί το $k+1$ αφού έχουν επιλεγεί k από τα t. Το m/n είναι η μη δεσμευμένη πιθανότητα να επιλεγεί ένα στοιχείο.

Αριθμός Αντικειμένων που Επιλέγονται

- Αν μας έχουν μείνει να εξετάσουμε $n - t$ αντικείμενα και μας μένουν να επιλέξουμε $n - t$ αντικείμενα, τότε $m - k = n - t$.
- Συνεπώς το $u \times (n - t) < m - k$ θα γίνει $u < 1$ και άρα θα επιλέξουμε σίγουρα το επόμενο αντικείμενο.
- Το ίδιο θα συμβεί και για το επόμενο αντικείμενο, κ.λπ., άρα θα επιλέξουμε όλα τα αντικείμενα μέχρι το τέλος.
- Συνεπώς δεν υπάρχει περίπτωση να τερματίσει ο αλγόριθμος έχοντας επιλέξει λιγότερα από m αντικείμενα.
- Επίσης, αν κάποια στιγμή επιλέξει m αντικείμενα, δεν θα επιλέξει κανένα επιπλέον, αφού το $u \times (n - t) < m - k$ θα γίνει $u \times (n - t) < 0$.
- Συνεπώς αφού ο αλγόριθμος θα επιλέξει τουλάχιστον m στοιχεία και το πολύ m στοιχεία, θα επιλέξει ακριβώς m στοιχεία.

Αριθμός Επαναλήψεων

- Ο μέγιστος αριθμός επαναλήψεων είναι n, αν χρειαστεί να εξετάσουμε όλα τα στοιχεία.
- Συχνά όμως θα έχουμε επιλέξει όσα στοιχεία χρειαζόμαστε πριν φτάσουμε στο τέλος.
- Η πιθανότητα να επιλέξουμε ένα οποιοδήποτε στοιχείο είναι m/n.
- Άρα η πιθανότητα να σταματήσει ο αλγόριθμος *πριν* το τελευταίο στοιχείο είναι $1 - m/n$.
- Αποδεικνύεται ότι κατά μέσο όρο, ο αλγόριθμος σταματά αφού ελέγξει $(n+1)m/(m+1)$ στοιχεία.

Δειγματοληψία με Ταμιευτήρα

- Η δειγματοληψία με επιλογή απαιτεί να γνωρίζουμε το μέγεθος του πληθυσμού.
- Τι μπορούμε να κάνουμε αν δεν γνωρίζουμε το μέγεθος του πληθυσμού;
- Για παράδειγμα, μπορεί να θέλουμε να επιλέξουμε τυχαία έναν αριθμό εγγραφών από ένα αρχείο καθώς το διαβάζουμε από την αρχή προς το τέλος.
- Ή μπορεί να μας δίνονται τα στοιχεία ενόσω ο αλγόριθμος εκτελείται.
- Για το σκοπό αυτό μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο *δειγματοληψίας με ταμιευτήρα* (reservoir sampling).
- Ο αλγόριθμος αυτός είναι ένας online αλγόριθμος.

Δειγματοληψία με Ταμιευτήρα

- Η βασική ιδέα είναι ότι αν θέλουμε ένα δείγμα m στοιχείων, γεμίζουμε έναν ταμιευτήρα με m στοιχεία μόλις τα βρούμε.
- Στη συνέχεια, για κάθε επιπλέον στοιχείο που εξετάζουμε, θα αλλάξουμε, πιθανώς, τα περιεχόμενα του ταμιευτήρα ώστε τα αντικείμενα που βρίσκονται σε αυτόν να περιέχονται με πιθανότητα m/t, όπου t είναι ο αριθμός των αντικειμένων που έχουμε εξετάσει.
- Όταν τελειώσουμε, τα αντικείμενα θα βρίσκονται στον ταμιευτήρα με πιθανότητα m/n, όπου n είναι το μέγεθος του πληθυσμού μας.

Δειγματοληψία με Ταμιευτήρα

- Στην αρχή βάζουμε τα πρώτα m αντικείμενα. Συνεπώς τα αντικείμενα αυτά έχουν όλα επιλεγεί με ίση πιθανότητα $m/t = m/m = 1$.
- Έστω λοιπόν ότι έχουμε εξετάσει t στοιχεία και τα περιεχόμενα του ταμιευτήρα βρίσκονται όλα με την ίδια πιθανότητα μέσα σε αυτόν.
- Θέλουμε να δείξουμε ότι το ίδιο θα συμβαίνει και όταν έχουμε εξετάσει $t + 1$ στοιχεία.

Δειγματοληψία με Ταμιευτήρα

- Όταν εξετάζουμε το $t+1$ στοιχείο, θα το προσθέσουμε στον ταμιευτήρα με πιθανότητα $m/(t+1)$, αντικαθιστώντας ένα από τα στοιχεία του ταμιευτήρα.
- Επιλέγουμε τυχαία το στοιχείο που θα βγάλουμε από τον ταμιευτήρα, άρα κάθε στοιχείο στον ταμιευτήρα έχει πιθανότητα $1/m$ να βγει.
- Επομένως η πιθανότητα να αντικατασταθεί ένα στοιχείο στον ταμιευτήρα είναι $m/(t+1) \times (1/m) = 1/(t+1)$.
- Αντιστρόφως, η πιθανότητα να παραμείνει ένα στοιχείο στον ταμιευτήρα είναι $1 - 1(t+1) = t/(t+1)$. Αφού το στοιχείο ήταν ήδη στον ταμιευτήρα, η πιθανότητα να ήταν εκεί και να παραμείνει εκεί είναι $(m/t) \times t/(t+1) = m/(t+1)$.
- Συνεπώς τόσο το αντικείμενο που μπήκε στον ταμιευτήρα όσο και κάθε αντικείμενο που παρέμεινε σε αυτόν είναι στον ταμιευτήρα με πιθανότητα $m/(t+1)$.

Δειγματοληψία με Ταμιευτήρα

- Εν ολίγοις: όταν είμαστε σε αντικείμενο $t \leq m$, το βάζουμε απλώς στον ταμιευτήρα.
- Όταν είμαστε σε αντικείμενο $t > m$, το βάζουμε στον ταμιευτήρα με πιθανότητα m/t και βγάζουμε από τον ταμιευτήρα ένα αντικείμενο στην τύχη.

Δειγματοληψία με Ταμιευτήρα

Algorithm: Reservoir sampling.

`ReservoirSampling`$(src, m) \rightarrow S$

Input: src, a source of items of the population
m the number of items to select

Output: S, an array with m randomly selected items from src

```
1  S ← CreateArray(m)
2  for i ← 0 to m do
3      S[i] ← GetItem(src)
4  t ← m
5  while (a ← GetItem(src)) ≠ NULL do
6      t ← t + 1
7      u ←RandomInt(1, t)
8      if u ≤ m then
9          S[u − 1] ← a
10 return S
```

Δειγματοληψία με Ταμιευτήρα