Αλγόριθμοι Παραγοντοποίησης Φυσικών Αριθμών

Μάρκος Ραδαίος

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
ΕΚΠΑ

Δεκέμβριος 2024

Περιεχόμενα

Πρώτες σκέψεις

Ας ξεκινήσουμε με κάτι απλό. Μπορούμε να δοκιμάσουμε όλους τους φυσικούς αριθμούς, από το 2 και έπειτα, μέχρι να βρούμε κάποιον διαιρέτη του N. Τι χρονική πολυπλοκότητα έχει αυτό;

Πρώτες σκέψεις

Το παρακάτω θεώρημα μπορεί να μας βοηθήσει.

Θεώρημα

Κάθε σύνθετος αριθμός N έχει τουλάχιστον έναν διαιρέτη p για τον οποίο ισχύει $1 < p \leq \sqrt{N}$.

Πρώτες σκέψεις

Το παρακάτω θεώρημα μπορεί να μας βοηθήσει.

Θεώρημα

Κάθε σύνθετος αριθμός N έχει τουλάχιστον έναν διαιρέτη p για τον οποίο ισχύει $1 < p \leq \sqrt{N}$.

Απόδειξη: Έστω ότι για κάποιον σύνθετο N δεν ισχύει. Τότε όλοι οι διαιρέτες του N θα είναι μεγαλύτεροι του $\sqrt{N}$. Οπότε, αν γράψουμε τον N ως $N = p \cdot q$, θα ισχύει $p > \sqrt{N}$ και $q > \sqrt{N}$. Με πολλαπλασιασμό κατά μέλη παίρνουμε $p \cdot q > \sqrt{N} \cdot \sqrt{N} \Leftrightarrow N > N$, άτοπο.

Πού καταλήγουμε;

Αφού κάθε ημιπρώτος αριθμός N είναι και σύνθετος, θα έχει και κάποιον διαιρέτη p, με $p \leq \sqrt{N}$. Άρα, ο αλγόριθμος θα τερματήσει σε $\sqrt{N}$ βήματα το πολύ και έχει χρονική πολυπλοκότητα $\mathcal{O}(\sqrt{N})$.

Για $N \approx 10^{25}$ χρειαζόμαστε μέχρι και 9 ώρες.

Για $N \approx 10^{30}$ χρειαζόμαστε μέχρι και 116 μέρες (φανταστείτε να γίνει διακοπή ρεύματος...).

Και τώρα;

Πώς θα βελτιώσουμε την λύση μας; Ας σημειώσουμε τα διάφορα κομμάτια της και ας δούμε αν μπορούμε να αλλάξουμε καθένα από αυτά.

1. Ελέγχουμε ορισμένους αριθμούς.
2. Κάνουμε πράξεις modulo για έλεγχο διαιρετότητας.

Βελτίωση του Κομματιού 1

Μπορούμε να ελέγξουμε τον αριθμό 2 και μετά μόνο περιττούς αριθμούς. Ή αντίστοιχα να παραλείπουμε και τα πολλαπλάσια του 3. Αυτή η ιδέα μπορεί να γενικευτεί.

Θεώρημα

Έστω φυσικός αριθμός $N > 1$. Ο ελάχιστος αριθμός p που είναι μεγαλύτερος του 1 και διαιρεί τον N είναι πρώτος αριθμός.

Βελτίωση του Κομματιού 1

Μπορούμε να ελέγξουμε τον αριθμό 2 και μετά μόνο περιττούς αριθμούς. Ή αντίστοιχα να παραλείπουμε και τα πολλαπλάσια του 3. Αυτή η ιδέα μπορεί να γενικευτεί.

Θεώρημα

Έστω φυσικός αριθμός $N > 1$. Ο ελάχιστος αριθμός p που είναι μεγαλύτερος του 1 και διαιρεί τον N είναι πρώτος αριθμός.

Απόδειξη: Αν ο N είναι πρώτος ισχύει εξ' ορισμού. Από εδώ και στο εξής θεωρούμε ότι ο N είναι σύνθετος. Έστω ότι δεν ισχύει το ζητούμενο. Τότε ο p θα είναι σύνθετος. Συνεπώς, θα υπάρχει κάποιος φυσικός αριθμός q με $1 < q < p$, τέτοιος ώστε $q \mid p$. Ισχύει όμως ότι $p \mid N$, άρα και $q \mid N$. Άτοπο, αφού θεωρήσαμε ότι ο p είναι ο ελάχιστος μη τετριμμένος διαιρέτης του N.

Βελτίωση του Κομματιού 1

Μας αρκεί να ελέγχουμε πρώτους αριθμούς. Αυτοί θα είναι πιο λίγοι. Πόσο πιο λίγοι;

Βελτίωση του Κομματιού 1

Μας αρκεί να ελέγχουμε πρώτους αριθμούς. Αυτοί θα είναι πιο λίγοι. Πόσο πιο λίγοι;

Το πλήθος των πρώτων αριθμών μέχρι το $\sqrt{N}$ είναι περίπου $\frac{2\sqrt{N}}{\log N}$ (μέχρι το N είναι περίπου $\frac{N}{\log N}$). Σαφώς λιγότεροι, αλλά πάλι δεν είναι επαρκώς λίγοι και για να τους βρούμε χρειαζόμαστε επιπλέον χρόνο.

Περιεχόμενα

Ώρα για μια ριζική αλλαγή

Ένα αρνητικό της λύσης μας είναι ότι εξαρτάται άμεσα από το μέγεθος του ελάχιστου παράγοντα. Αν κάποιος θέλει να μας δυσκολέψει είναι απολύτως "στο χέρι του": μπορεί απλά να χρησιμοποιήσει κάποιον αριθμό με μεγάλο ελάχιστο παράγοντα.

Για να αποφύγουμε κάτι τέτοιο, μπορεί να μας βοηθήσει η τυχαιότητα, όμως είναι δύσκολο απλά να μαντέψουμε τους ίδιους τους παράγοντες. Ψάχνουμε έναν αριθμό μέσα από $\sqrt{N}$ επιλογές ή δύο αριθμούς μέσα από N επιλογές. Πρακτικά, ψάχνουμε ψύλλους στα άχυρα.

Βελτιώνοντας τις πιθανότητες

Θεώρημα

Αν οι αριθμοί a, b αφήνουν το ίδιο υπόλοιπο όταν διαιρούνται με τον p, τότε ο p διαιρεί το $|a - b|$.

Με βάση αυτό, πλέον δεν θα αναζητούμε συγκεκριμένους, περιορισμένους αριθμούς ενός συνόλου. Αυτό που θα αναμένουμε, θα είναι να τύχουμε το ίδιο υπόλοιπο 2 φορές.

Το πρόβλημα είναι ότι δεν γνωρίζουμε τον p. Γνωρίζουμε, όμως, ότι σε περίπτωση που δύο αριθμοί αφήνουν το ίδιο υπόλοιπο με τον p, θα ισχύει $\gcd(|a - b|, N) \neq 1$ και με αυτόν τον τρόπο θα μπορούμε να βρούμε έναν παράγοντα του N.

Αυτήν την διαδικασία ακολουθεί ο Αλγόριθμος Pollard's Rho: διαλέγει τυχαίους αριθμούς και προσπαθεί να εντοπίσει ένα ζεύγος a, b που θα φανερώσει κάποιον παράγοντα του N, μέσω του $\gcd(|a - b|, N)$.

Το Πρόβλημα των Γενεθλίων

Πόσους φοιτητές πρέπει να διαλέξουμε τυχαία από ένα αμφιθέατρο, ώστε να εξασφαλίσουμε 50% πιθανότητα δύο από αυτούς να έχουν γενέθλια την ίδια μέρα;

Το Πρόβλημα των Γενεθλίων

Πόσους φοιτητές πρέπει να διαλέξουμε τυχαία από ένα αμφιθέατρο, ώστε να εξασφαλίσουμε 50% πιθανότητα δύο από αυτούς να έχουν γενέθλια την ίδια μέρα;

Περίπου 23 φοιτητές.

Το Πρόβλημα των Γενεθλίων

Έστω ότι έχουμε ένα σύνολο p αντικειμένων. Σε κάθε γύρο, επιλέγουμε κάποιο από τα p αντικείμενα στην τύχη. Πόσους γύρους θα χρειαστούμε ώστε να ξανατύχουμε κάποιο αντικείμενο που έχουμε ήδη τύχει σε προηγούμενο γύρο;

Το Πρόβλημα των Γενεθλίων

Έστω ότι έχουμε ένα σύνολο p αντικειμένων. Σε κάθε γύρο, επιλέγουμε κάποιο από τα p αντικείμενα στην τύχη. Πόσους γύρους θα χρειαστούμε ώστε να ξανατύχουμε κάποιο αντικείμενο που έχουμε ήδη τύχει σε προηγούμενο γύρο;

Θα χρειαστούμε κατά μέσο όρο $\sqrt{\frac{\pi}{2} \cdot p}$ γύρους, δηλαδή $\mathcal{O}(\sqrt{p})$. Στην περίπτωσή μας, όπου ο p είναι ο ελάχιστος μη τετριμμένος διαιρέτης του N, θα είναι αντίστοιχα $\mathcal{O}(N^{\frac{1}{4}})$. Συνεπώς, ο Pollard's Rho θα βρίσκει κάποιον παράγοντα του N σε τόσα βήματα, κατά μέσο όρο.

Παραγωγή Ψευδοτυχαίων Αριθμών

Συνήθως προτιμούμε την συνάρτηση $f(x) = (x^2 + c) \mod N$, για διάφορους λόγους.

Επιλέγουμε το c και ξεκινάμε από κάποιο x_0 της αρεσκείας μας. Από εκεί και πέρα, αν έχουμε τον αριθμό x, ο επόμενος ψευδοτυχαίος αριθμός της ακολουθίας δίνεται από το $f(x)$.

Αλγόριθμοι Εύρεσης Κύκλου: Floyd's Algorithm

Ο Αλγόριθμος του Floyd έχει τα εξής βήματα:

1. Ξεκίνα τους δείκτες x και y από την ίδια θέση.
2. Προχώρα τον δείκτη y μία φορά ($y := f(y)$).
3. Προχώρα τον δείκτη x δύο φορές ($x := f(f(x))$).
4. Έλεγξε αν οι δείκτες ταυτίζονται. Αν ναι, τότε τελειώσαμε. Αν όχι, πήγαινε στο βήμα 2.

Ο δείκτης x ονομάζεται λαγός (hare) και ο δείκτης y ονομάζεται χελώνα (tortoise). Για αυτόν τον λόγο, ο αλγόριθμος είναι γνωστός και ως Tortoise and Hare Algorithm.

Αλγόριθμοι Εύρεσης Κύκλου: Brent's Algorithm

Ο Αλγόριθμος του Brent έχει τα εξής βήματα:

1. Ανάθεσε $l := 1$.
2. Ανάθεσε $y := x$.
3. Επανάλαβε την παρακάτω διαδικασία l φορές:
   - Προχώρα τον δείκτη x μία φορά.
4. Επανάλαβε την παρακάτω διαδικασία l φορές:
   - Προχώρα τον δείκτη x μία φορά.
   - Έλεγξε αν οι δείκτες ταυτίζονται. Αν ναι, τότε τελειώσαμε. Αν όχι, συνέχισε.
5. Διπλασίασε το l και πήγαινε στο βήμα 2.

Σε σύγκριση με τον Αλγόριθμο του Floyd χρησιμοποιεί λιγότερους υπολογισμούς τιμών της f. Επίσης, μπορεί να συνδυαστεί με την ακόλουθη τροποποίηση που τον κάνει ακόμα πιο αποδοτικό.

Αλγόριθμοι Εύρεσης Κύκλου: Brent's Algorithm

Ο έλεγχος αν οι δύο δείκτες ταυτίζονται είναι υπολογιστικά ακριβός (σκεφτείτε τι πολυπλοκότητα χρειαζόμαστε για να υπολογίσουμε τον Μέγιστο Κοινό Διαιρέτη δύο αριθμών). Σχεδόν πάντα, ο έλεγχος επιστρέφει αρνητική απάντηση και για αυτόν τον λόγο μπορούμε να τον κάνουμε πιο αραιά, π.χ. κάθε 1024 βήματα. Αντί για 1024 βήματα μπορούμε να χρησιμοποιήσουμε κάποιον άλλον σταθερό αριθμό ή ακόμα και κάποια συνάρτηση ως προς l, όπως το $\sqrt{l}$.

Αλγόριθμοι Εύρεσης Κύκλου: Brent's Algorithm

1. Ανάθεσε $l := 1$ και $q := 1$.
2. Ανάθεσε $y := x$, $max\_leap := \sqrt{l}$ και $x\_snap := x$.
3. Επανάλαβε την παρακάτω διαδικασία l φορές:
   - Προχώρα τον δείκτη x μία φορά.
4. Επανάλαβε την παρακάτω διαδικασία l φορές:
   - Προχώρα τον δείκτη x μία φορά.
   - Αν έχεις συμπληρώσει max_leap βήματα από τον τελευταίο έλεγχο ή βρίσκεσαι στην l-οστή (τελευταία) επανάληψη αυτής της λούπας, τότε έλεγξε αν $\gcd(q, N) > 1$ (αν $\gcd(q, N) > 1$, τότε πήγαινε στο βήμα 6, αλλιώς ανάθεσε $x\_snap := x$), αλλιώς ανάθεσε $q := (q \cdot |x - y|) \mod N$.
5. Διπλασίασε το l και πήγαινε στο βήμα 2.
6. Επανάφερε τον δείκτη x σε x_snap.
7. Για όσο οι δείκτες δεν ταυτίζονται, προχώρα τον δείκτη x μία φορά.

Πολλαπλασιασμός Αριθμών των 128 bits

Για τον υπολογισμό του $f(x)$ χρειαζέται να κάνουμε πολλαπλασιασμούς αριθμών των 128 bits, των οποίων το γινόμενο μπορεί να χρειαστεί μέχρι και 256 bits για να αναπαρασταθεί. Η C, όμως, δεν έχει δικό της τύπο για αριθμούς των 256 bytes.

Από την στιγμή που ψάχνουμε το υπόλοιπο του γινομένου με το N, μπορούμε να αξιοποιήσουμε την ίδια ιδέα με το Binary Modular Exponentiation. Αυτό θα έχει πολυπλοκότητα $\mathcal{O}(\log(\min(a, b)))$.

Πολλαπλασιασμός Αριθμών των 128 bits

Μπορούμε να κάνουμε τις πράξεις σε $\mathcal{O}(1)$ ως εξής:

$$(a_1 \cdot 2^{64} + a_0)(b_1 \cdot 2^{64} + b_0) =$$

$$(a_1 \cdot b_1) \cdot 2^{128} + (a_0 \cdot b_1 + a_1 \cdot b_0) \cdot 2^{64} + a_0 \cdot b_0$$

Μπορούμε να εκφράσουμε το γινόμενο ως δύο μεταβλητές *heavy*, *light* των 128 bits τέτοιες ώστε $result = heavy \cdot 2^{128} + light$ (χρειάζεται προσοχή με τα κρατούμενα!). Κάτι σαν να κόβουμε την δυαδική αναπαράσταση του αποτελέσματος στην μέση.

Για παράδειγμα, για έναν δεκαδικό αριθμό με 10 ψηφία όπως το 6230473920 ισχύει αντίστοιχα:

$$6230473920 = 62304 \cdot 10^5 + 73920$$

Modulo: Μια ιδιαίτερα αργή πράξη

Η πράξη modulo είναι εξαιρετικά σημαντική στην Θεωρία Αριθμών. Παρόλα αυτά, ο υπολογισμός της είναι αρκετά πιο αργός σε σχέση με των άλλων βασικών πράξεων - πρόσθεση, αφαίρεση, πολλαπλασιασμός και δυαδικές πράξεις - εκτός της διαίρεσης με την οποία συνδέεται άμεσα, αφού $a \ mod \ b = a - \lfloor \frac{a}{b} \rfloor \cdot b$.

Stein's Algorithm για τον Υπολογισμό GCD

Αποφεύγει τις πράξεις modulo που κάνει ο Αλγόριθμος του Ευκλείδη. Χρησιμοποιεί αφαιρέσεις και bit shifts.

Βασίζεται σε 4 ιδιότητες:

1. $\gcd(a, 0) = a$
2. $\gcd(2a, 2b) = 2 \cdot \gcd(a, b)$
3. $\gcd(2a, b) = \gcd(a, b)$, αν b περιττός
4. $\gcd(a, b) = \gcd(a, b - a)$, αν a, b περιττοί και $a \leq b$

Όταν τον υλοποιούμε καλό είναι να αποφεύγουμε την αναδρομή και να προτιμούμε επαναληπτικές (iterative) υλοποιήσεις.

Montgomery Multiplication

Κάθε τιμή της f για να υπολογιστεί χρειάζεται μια πράξη modulo. Ισχυριζόμαστε ότι πλέον μπορούμε να βελτιώσουμε την πράξη. Τι άλλαξε;

Montgomery Multiplication

Κάθε τιμή της f για να υπολογιστεί χρειάζεται μια πράξη modulo. Ισχυριζόμαστε ότι πλέον μπορούμε να βελτιώσουμε την πράξη. Τι άλλαξε;

Το δεξί μέλος της πράξης παραμένει σταθερό για τις πράξεις μας. Με την μέθοδο Montgomery Multiplication μεταφέρουμε τους αριθμούς σε έναν νέο χώρο, το Montgomery Space, όπου η πράξη modulo μπορεί να γίνει με bit shifts.

Montgomery Multiplication

Η μέθοδος μετατρέπει κάθε αριθμό a σε $a \cdot r \ mod \ N$ για κάποιο κατάλληλο r, με $r > N$ και $gcd(r, N) = 1$. Στην πράξη, επιλέγουμε το r να είναι δύναμη του 2.

Δύο αριθμούς a, b που έχουν μετατραπεί σε αυτήν την μορφή μπορούμε να κανονικά τους προσθέσουμε/αφαιρέσουμε ($a \cdot r + b \cdot r \equiv (a + b) \cdot r \mod N$). Επίσης, μπορούμε να βρούμε τον gcd τους με το N ($\gcd(a \cdot r, N) = \gcd(a, N)$, διότι $\gcd(r, N) = 1$).

Montgomery Multiplication

Με τον πολλαπλασιασμό τα πράγματα είναι κάπως διαφορετικά. Ισχύει ότι $(a \cdot r)(b \cdot r) \equiv (ab \cdot r) \cdot r \mod N$. Δηλαδή, έχουμε ένα περίσσειο r. Σε τέτοιου είδους σχέσεις, ΔΕΝ μπορούμε να διαιρέσουμε με r. Πρέπει να πολλαπλασιάσουμε με r^{-1}. Η πράξη $x \cdot r^{-1} \mod N$ ονομάζεται Montgomery Reduction και μπορούμε να την εκτελέσουμε χωρίς κάποια πράξη modulo.

Περιεχόμενα

Άλλες Μέθοδοι Παραγοντοποίησης

1. **Μέθοδος Fermat**: Προσπαθεί να εκφράσει το N ως διαφορά τετραγώνων, αφού $a^2 - b^2 = (a+b)(a-b)$ και το δεξιά μέλος δίνει μια παραγοντοποιημένη μορφή.
2. **Pollard's $p-1$**: Βασίζεται στο ότι αν p πρώτος, τότε είναι πολύ πιθανό ο $p-1$ να έχει μικρούς διαιρέτες (λέμε ότι είναι B-powersmooth, για κάποιο μικρό B), σε συνδυασμό με το Μικρό Θεώρημα του Fermat ($a^{p-1} \equiv 1 \mod p$).
3. **Lenstra's Ecliptic Curve Method**: Χρησιμοποιεί ελλειπτικές καμπύλες και το group structure τους για να "φανερώσει" κάποιον παράγοντα του N.

Άλλες Μέθοδοι Παραγοντοποίησης

4. **Quadratic Sieve**: Σε παρόμοια λογική με την μέθοδο του Fermat προσπαθεί να δημιουργήσει μια ισοτιμία της μορφής $a^2 \equiv b^2 \mod N$, λύνοντας ένα XOR σύστημα.
5. **General Number Field Sieve**: Ο πιο αποδοτικός ασυμπτωτικά αλγόριθμος που γνωρίζουμε. Αξιοποιεί πολύ πιο προχωρημένες ιδέες και έννοιες που ξεφεύγουν από άποψη δυσκολίας, όπως δαχτύλιοι αλγεβρικών αριθμών και πεδία.

Βοηθητικοί Σύνδεσμοι

1. Διαδραστικό Web App προς Κατανόηση του Birthday Paradox
2. Visualization του Αλγορίθμου του Floyd για Εύρεση Κύκλου