Gedanken zur (p-1)-Faktorisierungsmethode

Thorsten Reinecke

4. Februar 2000

Zusammenfassung

Das Funktionsprinzip der (p-1)-Faktorisierungsmethode wird erläutert. Die *standard continuation* wird motiviert. Es wird eine Modifikation der Phase 1 vorgestellt, die für lange Läufe der Phase 1 vorteilhaft sein dürfte. Es wird eine entfernte Verwandtschaft zur Faktorisierungsmethode nach Fermat aufgezeigt.

1 Grund-Algorithmus

Der (p-1)-Algorithmus zur Faktorisierung natürlicher Zahlen basiert auf dem sogenannten kleinen Fermatschen Satz:

p prim $\Longrightarrow$ $a^{p-1} \equiv 1 \bmod p$ für $\gcd(a,p) = 1$

Dieser Sachverhalt wird häufig für Vorstufen verschiedener Primzahltests verwendet, in dem man für verschiedene Basen $a^{p-1} \bmod p$ berechnet. Falls der Ausdruck ungleich 1 wird, weiß man, daß die Zahl zusammengesetzt sein muß. Andernfalls "verstärkt" sich die Annahme, daß die Zahl prim sein könnte.

Der kleine Fermatsche Satz läßt sich jedoch auch zur Faktorisierung natürlicher Zahlen einsetzen. Nehmen wir an, p wäre unser gesuchter Primfaktor. Nehmen wir ferner an, wir hätten bereits für kleinere Primfaktoren Probedivision durchgeführt. Dann zerfällt $p-1$ in mindestens zwei Faktoren: $p-1 = 2 \cdot \frac{p-1}{2}$. Mit einiger Wahrscheinlichkeit ist $\frac{p-1}{2}$ sogar noch weiter zerlegbar. Und genau diese Hoffnung hegt man bei der (p-1)-Faktorisierungsmethode. Man rät sukzessive mögliche kleine Faktoren von $p-1$.

Seien $e_1, e_2, e_3, .., e_k$ diese gewählten Zahlen und a eine gewählte Basis, dann wäre $a^{e_1 \cdot \ldots \cdot e_k} \equiv 1 \bmod p$. Dies läßt sich mit den Potenzgesetzen umformulieren zu $(...((a^{e_1})^{e_2})...)^{e_k} \equiv 1 \bmod p$. Nun sieht man auch, daß falsch geratene Faktoren von $p-1$ die Kongruenz trotzdem erhalten, da die Basis a unter den oben genannten Einschränkungen beliebig gewählt werden kann, somit also falsche e_i in die Basis gezogen werden können.

Ein Problem verbleibt allerdings noch: Da wir p nicht kennen, können wir die Kongruenz nicht bilden. Dies stört allerdings wenig, da für eine gegebene Zahl $N = p \cdot r$, deren Faktor p wir suchen, folgendes gilt: $x \equiv y\ (mod\, p) \Longleftrightarrow \exists z:\ x \equiv y + z \cdot p \bmod (p \cdot r)$.

Falls $a^{e_1 \cdot \ldots \cdot e_k} \equiv 1 \bmod p$ gilt, so gilt $a^{e_1 \cdot \ldots \cdot e_k} - 1 \equiv 0 \bmod p$ und somit $a^{e_1 \cdot \ldots \cdot e_k} - 1 \equiv z \cdot p \bmod N$. Wenn nun $z \neq 0$ ist (und dies ist ein sehr wahrscheinlicher Fall!), so erhalten wir durch $\gcd(a^{e_1 \cdot \ldots \cdot e_k} - 1 \bmod N, N)$ den gewünschten Teiler p .

Ein praktikabler Algorithmus ergibt sich daraus, in dem man sich eine Basis a wählt und dann alle Primzahlen (und ggf. Primzahlpotenzen) bis zu einer vorher festgelegten Obergrenze als Exponenten verwendet. In regelmäßigen Abständen testet man dann den ggT von $a-1$ und N. – Der in der Literatur manchmal beschriebene Akkumulationsschritt, bei welchem nach jedem Exponentiationsschritt die neu erhaltene Basis um eins vermindert mit einer Sammelvariablen zu multiplizieren ist, um den ggT-Test anschließend mit der Sammelvariablen durchführen zu können, ist aufgrund des Kleinen Fermatschen Satzes unnötig. Nichts desto trotz kann die Akkumulation in vielen anderen Algorithmen die kostenintensiven ggT-Berechnungen erheblich reduzieren und ist auch bei der Bestimmung multiplikativer, von einander unabhängiger Inversen vorteilhaft einsetzbar.

2 Standard Continuation

Falls der Grundalgorithmus versagt, kann man eine zweite Phase anhängen. Diese zweite Phase ist erfolgreich, wenn $p-1$ durch den Grundalgorithmus bis auf einen Primfaktor zerlegt werden konnte. Dieser übriggebliebene Restfaktor s kann nun ohne teure Exponentiation ermittelt werden, in dem man ein a^q mit einer Primzahl q in der Größenordnung des vermuteten s bestimmt und mittels Multiplikation die Umgebung von q nach s absucht. Hierfür kann man sich eine Tabelle kleiner Potenzen von a anlegen, um die Abstände zwischen den Primzahlen jeweils durch eine Multiplikation überwinden zu können: $a^{q+\Delta} = a^q \cdot a^{\Delta}$. Diese Phase schließt erfolgreich ab, falls $q = s$ und somit $\gcd(a^q - 1 \bmod N, N)$ den Teiler liefert. Man beachte, daß die Basis a nicht die Ursprungsbasis ist, sondern die nach Abschluß des Standard-Algorithmus erhaltene Basis.

Wenn man so will, kann man die Operationen als Ringoperationen auffassen. Phase 1 hat den Ring komprimiert (die Standard-Exponentiation auf den natürlichen Zahlen entspricht der multiplikativen Ring-Operation). In Phase 2 sucht man den komprimierten Ring durch additive Operationen ab.

In der Literatur sind weitere, wesentlich effizientere Methoden für Phase 2 als die hier beschriebene *standard continuation* entwickelt worden.

3 Modifikation der Phase 1

In der Literatur finden sich für die erste Phase eigentlich keine Verbesserungsvorschläge.

Wenn man den (p-1)-Algorithmus nur ergänzend zu anderen Algorithmen einsetzt, dann wird man Phase 1 möglichst kurz halten und darauf spekulieren, den Teiler in Phase 2 zu ermitteln zu können. Falls auch Phase 2 versagt, dann war die Zahl eben nicht für den (p-1)-Algorithmus geeignet. – Diese Vorgehensweise ist durchaus legitim.

Ein möglicher Nachteil des oben beschriebenen Vorgehens liegt allerdings darin, die Phase 1 zu früh abzubrechen. Eine zu große Verlängerung der Phase 1 hingegen kostet unnötige Zeit.

Eine Lösung bestünde darin, die Phase 1 substantiell zu beschleunigen. Dies ist bisher nicht gelungen.

Eine abgeschwächte Verbesserung der Phase 1 wäre erreichbar, wenn man die Phase 1 mit geringen Kosten um Elemente der Phase 2 ergänzen könnte. Die Phase 1 könnte dann mit einer wesentlich höheren Obergrenze durchlaufen werden, weil quasi parallel eine Art "Phase 2" abläuft. Basierend auf dem Grundalgorithmus ist dies tatsächlich möglich und die zusätzlichen Kosten sind mit denen der standard-continuation vergleichbar. Ob sich allerdings die Modifikationen lohnen, bedarf weiterer Untersuchungen. Hier bin ich eher skeptisch, da man die (p-1)-Methode – wenn überhaupt – dann nur ergänzend vor den elliptischen Kurven und/oder dem Quadratischen Sieb einsetzt. Dadurch dominiert von der Bedeutung her die Phase 2.

In einer Standalone-Anwendung des (p-1)-Verfahrens würde hingegen Phase 1 dominieren, da man nicht riskieren könnte, das Verfahren mit Phase 2 erfolglos zu beenden.

Die erforderlichen Modifikationen sind relativ simpel. Wir betrachten die fortlaufenden Werte der Basis als Folge, die durch den Startwert a_0 und eine Folge von Exponenten e_i bestimmt werden: $a_{n+1} := a_n^{e_n}$. Wir definieren ferner $a_i := 1$ für negative i.

Das (p-1)-Standardverfahren sieht dann etwa folgendermaßen aus:

$i := 0$; repeat $i := i + \mathit{step}$ until $\gcd(a_i - 1, N) \neq 1$;

Das modifizierte Verfahren sieht so aus:

$i := 0$; repeat $b := \prod_{j=i+1}^{i+\mathit{step}} a_j - a_{j-\Delta}$; $i := i + \mathit{step}$ until $\gcd(b, N) \neq 1$;

Das modifizierte Verfahren ergänzt das Standardverfahren um zusätzliche Faktoren, da für $\Delta = 1$ z.B.

$a_{n+1} - a_n = a_n^{e_n} - a_n = a_n \cdot (a_n^{e_n - 1} - 1) = a_n \cdot (a_n - 1) \cdot \sum_{i=0}^{e_n - 2} a_n^i$

gilt und somit ein Teiler $a_{n+1} - 1$ lediglich eine Auswertung später entdeckt wird. Hingegen können zusätzliche Faktoren durch den Summenausdruck hinzukommen, die bei jedem Auswertungsschritt variieren. Über die Höhe des Delta kann die Größe des zusätzlichen Faktors eingestellt werden. Dieser dürfte seinerseits mit einiger Wahrscheinlichkeit in weitere Primfaktoren zerfallen. Auf diese Weise erhält man eine Art "Phase 2" bereits innerhalb der Grundphase.

Allgemein gilt:

$a_{n+\Delta} - a_n = a_n^{\prod_{i=n}^{n+\Delta-1} e_i} - a_n = a_n \cdot (a_n^{(\prod_{i=n}^{n+\Delta-1} e_i)-1} - 1) = a_n \cdot (a_n - 1) \cdot \sum_{i=0}^{(\prod_{k=n}^{n+\Delta-1} e_k)-2} a_n^i$

Der obige Term sieht relativ kompliziert aus; tatsächlich geht es jedoch nur um den Nachweis der Tatsache, daß das Verfahren durch Berechnung der linken Seite auch bei anderen Deltawerten zusätzliche Faktoren (rechte Seite) gegenüber dem Standardverfahren produziert.

Die zusätzlichen Kosten der Modifikation der Phase 1 sind relativ gering. Es ist ein Ringpuffer für die Basen anzulegen. Dieser kann als Array der Größe $\Delta + 1$ realisiert werden. Der Zugriff erfolgt dann über die Indices modulo $\Delta + 1$.

4 Verwandtschaft zur Faktorisierungmethode nach Fermat

Nehmen wir an, $N = p \cdot q$ bestehe aus zwei annähernd gleichen Teilern. – Dies ist der einzige Sonderfall, wo die Fermatmethode den anderen Faktorisierungsmethoden deutlich überlegen ist.

Vorausgesetzt, daß p und q große Primzahlen sind, läßt sich auch hier eine modifizierte (p-1)-Methode anwenden.

Wir gehen zunächst von folgender Kongruenz aus:

$a^{(p-1)(q-1)} \equiv 1 \bmod N$

Dies ergibt umgeformt:

$a^{p+q} \equiv a^{N+1} \bmod N$

Nun gilt $p + q \geq \left\lfloor 2 \cdot \sqrt{N} \right\rfloor$ und wir erhalten $a^{p+q-\lfloor 2\cdot\sqrt{N}\rfloor} \equiv a^{N+1-\lfloor 2\cdot\sqrt{N}\rfloor} \bmod N$.

Dies kann man benutzen, um $p + q$ zu bestimmen. Daraus wiederum können p und q folgendermaßen berechnet werden:

Aus den binomischen Formeln ergibt sich: $(p - q)^2 = (p + q)^2 - 4pq = (p + q)^2 - 4N$

also: $p = \frac{(p+q)+(p-q)}{2} = \frac{(p+q)+\sqrt{(p+q)^2-4N}}{2}$

Nun gibt mehrere Alternativen, $p + q$ zu ermitteln:

4.0.1 Zweiermethode

Wir berechnen $2^{N+1-\lfloor 2\cdot\sqrt{N}\rfloor} \bmod N$ und wandern per Division durch 2 abwärts, bis wir 1 erhalten. Dies ist über Shiften der Werte mit geringem Aufwand möglich. Die Anzahl der Shift-Operationen zählen wir und erhalten dadurch den Wert $p + q$.

4.0.2 Tabellen-Methode

Wir wählen ein a und eine Größe step. Wir berechnen für $0 \leq i < step$ eine Tabelle mit $a^{N+1-i-\lfloor 2\cdot\sqrt{N}\rfloor} \bmod N$. Anschließend können wir in sukzessive $a^{k \cdot step}$ berechnen, bis der Wert in der Tabelle gefunden wird.

4.1 Verbesserungsmöglichkeiten

Zunächst einmal ist $p + q$ eine gerade Zahl, was es ermöglicht, die Schrittweiten zu verdoppeln.

Eine weitere Verbesserung läßt sich erzielen, wenn man berücksichtigt, daß man rund die Hälfte der Primzahlen als Teiler von $p + q$ ausschließen kann. Denn $t \mid (p + q) \Leftrightarrow p \equiv -q \bmod t$. Eine anschließende Multiplikation mit p ergibt $p^2 \equiv -N \bmod t$, so daß $-N$ ein quadratischer Rest modulo t sein muß.

4.2 Bewertung

Die Methode funktioniert nur in Spezialfällen: Die zu faktorisierende Zahl muß aus genau zwei Primzahlen bestehen. Akzeptable Laufzeiten ergeben sich nur, wenn beide Primzahlen gleichstellig sind und überdies in einer möglichst langen Kette ihrer führenden Stellen übereinstimmen.

Wenngleich es unwahrscheinlich erscheint, auf diese Weise einen Teiler zu finden, so ist doch für diesen Spezialfall die Verwandtschaft zur Faktorisierungsmethode nach Fermat ein interessanter Aspekt.

Literatur

[1] Riesel, Hans: "Prime Numbers and computer methods for factorization", Boston; Basel; Stuttgart: Birkhäuser, 1985

[2] Peter L. Montgomery: "Speeding the Pollard and Elliptic Curve Methods of Factorization", Mathematics Of Computation, Vol. 48 (177), January 1987, p. 243-264