Rozklad přirozeného čísla na součin prvočísel

Připomeňme, že hledáme co nejrychlejší algoritmus, který dané přirozené číslo N rozloží na součin prvočísel.
Rozdělme náš problém na tři úkoly:

1. **Test na složenost:** Pro dané $N \in \mathbb{N}$ rychle rozhodnout, zda N splňuje nějakou podmínku, která je splněna každým prvočíslem, a tedy rozhodnout, zda je to *určitě číslo složené* anebo zda je to *asi prvočíslo*.
2. **Test na prvočíselnost:** Je-li N asi prvočíslo, *dokázat,* že N skutečně prvočíslem je, nebo to vyvrátit.
3. **Nalezení dělitele:** Je-li N složené, nalézt netriviálního dělitele d čísla N.

Celé rozkládání je pak rekurzivní proces: máme-li dělitele d čísla N, který splňuje $1 < d < N$, opakujeme celý postup pro čísla d a $\frac{N}{d}$.

Metoda pokusného dělení

Zkoušíme dělit N postupně všemi prvočísly 2, 3, 5, 7, 11, ... až do jisté hranice. Pokud jsme schopni to provést pro všechna prvočísla $p \leq \sqrt{N}$, provedeme tím všechny tři úkoly současně.

V případě, kdy N je natolik velké, že dělení N všemi prvočísly $p \leq \sqrt{N}$ by bylo příliš zdlouhavé, můžeme N dělit všemi prvočísly až do jisté hranice, abychom se zbavili malých faktorů (čím je prvočíslo menší, tím větší je pravděpodobnost, že dělí náhodně zvolené přirozené číslo).

Budeme předpokládat, že máme uloženu tabulku prvočísel $p[1] = 2$, $p[2] = 3$, $p[3] = 5$, $p[4] = 7$, ..., $p[k]$. Po vyčerpání této tabulky budeme pokračovat v dělení N čísly z jistých zbytkových tříd (např. čísly dávajícími zbytek 1 nebo 5 po dělení 6, resp. čísly dávajícími zbytek 1, 7, 11, 13, 17, 19, 23 nebo 29 po dělení 30 apod.). Tím sice některá dělení provedeme zbytečně, ale výsledek bude stále správný (testovat, zda číslo, kterým hodláme dělit, je prvočíslo, nemá smysl).

Zvolme horní hranici B, v níž přestaneme dělit, abychom algoritmem neztratili příliš mnoho času.

Algoritmus (Pokusné dělení). *Je dána tabulka prvočísel* $p[1] = 2$, $p[2] = 3$, $p[3] = 5$, $p[4] = 7$, $\ldots$, $p[k]$ *(kde* $k > 3$*), číslo* $B \geq p[k]$*, vektor* $t = [6, 4, 2, 4, 2, 4, 6, 2]$*, číslo* j *tak, že* $j = 0$ *(resp. 1, 2, 3, 4, 5, 6, 7), právě když* $p[k] \equiv 1 \pmod{30}$ *(resp. 7, 11, 13, 17, 19, 23, 29). Pro dané* $N \in \mathbb{N}$ *algoritmus hledá rozklad čísla* N*. Jen největší činitel tohoto rozkladu nemusí být prvočíslo, ale v tom případě může být dělitelný jen prvočísly většími než* B*.*

1. *[Inicializace] Je-li* $N \leq 5$*, pak vytiskni odpovídající rozklad* N *a skonči. Jinak polož* $i \leftarrow -1$, $m \leftarrow 0$.
2. *[Další prvočíslo] Polož* $m \leftarrow m + 1$*. Je-li* $m > k$*, polož* $i \leftarrow j - 1$ *a jdi na 5, jinak polož* $d \leftarrow p[m]$.
3. *[Zkus dělit] Současně spočítej* $r \leftarrow N \bmod d$, $q \leftarrow [\frac{N}{d}]$*. Je-li* $r = 0$*, vytiskni* d *jako činitele v hledaném rozkladu a polož* $N \leftarrow \frac{N}{d}$ *a opakuj krok 3.*
4. *[Prvočíslo?] Je-li* $d \geq q$*, vytiskni* N *jako posledního činitele a zprávu, že rozklad je úplný a skonči. Jinak, je-li* $i < 0$*, jdi na 2.*
5. *[Další dělitel] Polož* $i \leftarrow (i + 1) \bmod 8$, $d \leftarrow d + t[i]$*. Je-li* $d > B$*, vytiskni* N *jako posledního činitele a zprávu, že poslední činitel v rozkladu nemusí být prvočíselný a skonči. Jinak jdi na 3.*

Metoda pokusného dělení

Jestliže v kroku 4 nastane $d \geq q$, už víme, že N není dělitelné žádným prvočíslem $p \leq d$. Navíc

$$N = qd + r < (q+1)d \leq (d+1)d,$$

a tedy $\sqrt{N} < d + 1$. Proto už musí být N prvočíslo.

Pro úplný rozklad N se metoda pokusného dělení hodí jen pro malá N (řekněme $N < 10^8$), protože pro větší N existují lepší metody. Každopádně je užitečná pro odstranění malých faktorů.

Vhodnou tabulkou prvočísel by mohla být tabulka prvočísel menších než 500 000, máme-li na ni dost místa v paměti (je to 41 538 prvočísel). Vhodnější než uložení vlastních prvočísel může být uložení diferencí mezi nimi nebo dokonce poloviny diferencí (diferenci $p[k] - p[k-1]$ můžeme uložit do jednoho bytu pro $p[k] \leq 1\,872\,851\,947$, její polovinu dokonce pro $p[k] \leq 1\,999\,066\,711\,391$).

Obsahuje-li naše tabulka prvočísla aspoň do 500 000, je asi lepší po vyčerpání tabulky v dělení nepokračovat, ale užít jinou metodu.

Testy na složenost

Zvolme nějakou podmínku, které vyhovuje každé prvočíslo a které složená čísla většinou nevyhovují, přičemž je třeba, aby bylo možné podmínku pro dané přirozené číslo rychle ověřit.

Na první pohled se zdá být vhodnou podmínkou Wilsonova věta:

$$\forall n > 1: \qquad n \text{ je prvočíslo } \Leftrightarrow\ (n-1)! \equiv -1 \pmod{n}$$

To je dokonce nutná a dostatečná podmínku prvočíselnosti a byla by tedy nejen testem na složenost, ale také testem na prvočíselnost. Avšak nikdo neví, jak spočítat pro velká N dostatečně rychle číslo $(N-1)! \mod N$.

Výhodnější podmínku dává Fermatova věta, neboť je možné rychle počítat mocniny prvků v libovolné grupě.

Předpokládejme, že je dána grupa $(G, \cdot)$ s neutrálním prvkem 1 a že umíme prvky této grupy uchovávat v paměti a také s nimi počítat (násobit je a počítat jejich inverzní prvky).

Výpočet mocniny v grupě

Algoritmus (Binární umocňování zprava doleva). *Pro dané $g \in G$ a dané celé číslo n algoritmus počítá g^n v grupě $(G, \cdot)$. Proměnné y a z slouží k uchovávání prvků grupy G.*

1. *[Inicializace] Polož $y \leftarrow 1$. Je-li $n = 0$, pak vytiskni y a skonči. Je-li $n < 0$, polož $N \leftarrow -n$, $z \leftarrow g^{-1}$. Jinak polož $N \leftarrow n$, $z \leftarrow g$.*
2. *[Násob?] Je-li N liché, polož $y \leftarrow z \cdot y$.*
3. *[Poloviční N] Polož $N \leftarrow [\frac{N}{2}]$. Je-li $N = 0$, vytiskni y a skonči. Jinak polož $z \leftarrow z \cdot z$ a jdi na 2.*

Důkaz správnosti algoritmu. Vždy před započetím kroku 2 platí $y \cdot z^N = g^n$. Jistě to platilo při prvním vstupu na krok 2.
Označme N', y' a z' nové hodnoty proměnných N, y a z po provedení kroků 2 a 3.
Je-li N sudé, pak $y' = y$, $N' = \frac{N}{2}$, $z' = z^2$, tedy

$$y' \cdot (z')^{N'} = y \cdot z^{2\frac{N}{2}} = y \cdot z^N.$$

Je-li N liché, pak $y' = y \cdot z$, $N' = \frac{N-1}{2}$, $z' = z^2$, tedy

$$y' \cdot (z')^{N'} = y \cdot z \cdot z^{2\frac{N-1}{2}} = y \cdot z^N.$$

Odhad časové náročnosti algoritmu

Grupové násobení se provádí $a + b - 1$ krát, kde a je počet cifer ve dvojkovém zápise čísla n a b je počet jedniček v tomto zápise.
Jistě platí $a + b - 1 \leq 2[\log_2 |n|] + 1$.
Je-li například $G = (\mathbb{Z}/m\mathbb{Z})^\times$, je jedno násobení časové náročnosti $O(\ln^2 m)$, proto celý algoritmus je časové náročnosti $O(\ln^2 m \ln |n|)$.

V předchozím algoritmu jsme procházeli cifry dvojkového zápisu čísla n zprava doleva. Zcela analogicky můžeme tyto cifry procházet ovšem zleva doprava. Musíme však znát polohu „nejlevější“ jedničky v tomto zápise, tj. znát $e \in \mathbb{Z}$ s vlastností $2^e \leq |n| < 2^{e+1}$.

Algoritmus (Binární umocňování zleva doprava). *Pro dané $g \in G$ a dané celé číslo n algoritmus počítá g^n v grupě $(G, \cdot)$. Je-li $n \neq 0$, musí být dáno $e \in \mathbb{Z}$ s vlastností $2^e \leq |n| < 2^{e+1}$.*

1. *[Inicializace] Je-li $n = 0$, pak vytiskni 1 a skonči. Je-li $n < 0$, polož $N \leftarrow -n$, $z \leftarrow g^{-1}$. Jinak polož $N \leftarrow n$, $z \leftarrow g$. Konečně (tj. v obou případech) polož $y \leftarrow z$, $E \leftarrow 2^e$, $N \leftarrow N - E$.*
2. *[Konec?] Je-li $E = 1$, vytiskni y a skonči. Jinak polož $E \leftarrow \frac{E}{2}$.*
3. *[Násob] Polož $y \leftarrow y \cdot y$. Je-li $N \geq E$, polož $N \leftarrow N - E$, $y \leftarrow y \cdot z$. Jdi na 2.*

Důkaz správnosti algoritmu. Vždy před započetím kroku 2 platí $y^E \cdot z^N = g^n$. Jistě to platilo při prvním vstupu na krok 2, kdy $y^E \cdot z^N = z^{2^e} \cdot z^{N-2^e} = z^N = g^n$.

Označme E', N' a y' nové hodnoty proměnných E, N a y po provedení kroků 2 a 3.

Je-li $N < E'$, pak $E' = \frac{E}{2}$, $y' = y^2$, $N' = N$, tedy

$$(y')^{E'} \cdot z^{N'} = (y^2)^{\frac{E}{2}} \cdot z^N = y^E \cdot z^N.$$

Je-li $N \geq E'$, pak $E' = \frac{E}{2}$, $y' = y^2 \cdot z$, $N' = N - E'$, tedy

$$(y')^{E'} \cdot z^{N'} = (y^2 \cdot z)^{\frac{E}{2}} \cdot z^{N-\frac{E}{2}} = y^E \cdot z^N.$$

Vždy před krokem 2 je $0 \leq N < E$. Proto při skončení je $N = 0$.

Porovnání obou algoritmů

Nevýhody oproti předchozímu algoritmu: je třeba předem spočítat e, to však (je-li základ naší poziční soustavy pro uchovávání „velkých“ čísel mocnina 2) je velmi rychlé. Patrně totiž budeme znát pozici nejvyšší nenulové cifry v naší poziční soustavě a pak určení e zabere čas ohraničený konstantou. Zdánlivou nevýhodou je i uchovávání velkého čísla E a výpočet rozdílu $N - E$. Avšak při implementaci budeme uchovávat e a test $N \geq E$ i výpočet $N - E$ provedeme manipulací s bitem obsahujícím e-tou dvojkovou cifru čísla N (zde je podstatné, aby skutečně byl základ naší poziční soustavy mocnina 2).

Výhoda: jedno ze dvou násobení, které se provádí v kroku 3, je vždy proměnnou z, ve které je v průběhu celého výpočtu g (nebo g^{-1} je-li $n < 0$). Je-li tedy například $G = (\mathbb{Z}/m\mathbb{Z})^\times$, v případě, že $g = a + m\mathbb{Z}$ pro $|a|$ menší než je základ naší poziční soustavy, je násobení g pouze lineárního času a ne kvadratického. Pak se tedy v kroku 3 provede jedno násobení řádu $O(\ln^2 m)$ a nejvýše jedno řádu $O(\ln m)$. Celý algoritmus je sice stále časové náročnosti $O(\ln^2 m \ln |n|)$, ale s menší O-konstantou.

Využití umocňování pro test na složenost

Pro test na složenost užijeme malou Fermatovu větu: máme tedy dáno přirozené číslo N, o kterém chceme vědět, zda je to číslo složené. Budeme to vědět jistě, nalezneme-li celé číslo a, $1 \leq a < N$, pro které platí $a^{N-1} \not\equiv 1 \pmod{N}$.
Takové a se nazývá svědek složenosti čísla N. Pokud však pro takové a platí $a^{N-1} \equiv 1 \pmod{N}$, nemůžeme z toho usoudit nic.

Celý algoritmus tedy bude vypadat takto: budeme náhodně volit $a \in \mathbb{Z}$, $1 \leq a < N$, a počítat $a^{N-1} \mod N$. Pokud pro některé a bude splněno $a^{N-1} \not\equiv 1 \pmod{N}$, jsme hotovi a víme, že N je opravdu složené číslo (zmíněné a si můžeme zapamatovat pro případ, že bychom chtěli přesvědčit někoho dalšího o složenosti N). Pokud pro všechna a budeme dostávat $a^{N-1} \equiv 1 \pmod{N}$, po jistém počtu pokusů algoritmus ukončíme a usoudíme, že patrně je N prvočíslo. Jestli je však opravdu N prvočíslo, takto zjistit nemůžeme.

Nevýhodou popsaného algoritmu je, že téměř jistě neodhalí jistý typ složených čísel, nazývaných Carmichaelova čísla.

Carmichaelova čísla

Definice. Složené číslo N se nazývá Carmichaelovo číslo, jestliže pro všechna celá čísla a, která jsou nesoudělná s N, platí $a^{N-1} \equiv 1 \pmod{N}$.

Carmichaelovo číslo by náš algoritmus označil za složené pouze tehdy, kdyby za a zvolil číslo soudělné s N, což je však velmi nepravděpodobné. Přitom platí:

Věta (Alford, Granville, Pomerance). Existuje nekonečně mnoho Carmichaelových čísel.

Příklad. $N = 561 = 3 \cdot 11 \cdot 17$ je Carmichaelovo číslo.

Důkaz. Pro libovolné celé číslo a nesoudělné s 561 z Fermatovy věty dostáváme $a^2 \equiv 1 \pmod{3}$, $a^{10} \equiv 1 \pmod{11}$ a $a^{16} \equiv 1 \pmod{17}$. Protože 2, 10 i 16 jsou dělitelé čísla 560, je 561 Carmichaelovo číslo.

Výhodnější než testovat Fermatovu větu je proto testování následujícího zesílení Fermatovy věty.

Využití zesílení malé Fermatovy věty

Věta. *Pro libovolné liché prvočíslo p a libovolné celé číslo a nedělitelné p platí*

$$a^{\frac{p-1}{2}} \equiv \pm 1 \pmod{p}.$$

Důkaz. Z Fermatovy věty

$$p \mid (a^{p-1} - 1) = (a^{\frac{p-1}{2}} - 1) \cdot (a^{\frac{p-1}{2}} + 1).$$

Protože je p prvočíslo, musí dělit některého z uvedených činitelů.

Příklad. Test, zda $a^{\frac{N-1}{2}} \equiv \pm 1 \pmod{N}$, by mohl vyloučit i 561, neboť

$$5^{280} \equiv 5^{16 \cdot 17 + 8} \equiv 5^8 = 25^4 \equiv 8^4 = 16^3 \equiv (-1)^3 = -1 \pmod{17}$$

a zároveň

$$5^{280} \equiv (5^2)^{140} \equiv 1 (\mathrm{mod}\, 3),$$

a proto 5^{280} není kongruentní modulo 561 ani s 1 ani s -1.

Další zesílení malé Fermatovy věty

Příklad. Test, zda $a^{\frac{N-1}{2}} \equiv \pm 1 \pmod{N}$, neodhalí například $N = 1729 = 7 \cdot 13 \cdot 19$, neboť $\frac{N-1}{2} = 864 = 2^5 \cdot 3^3$ je dělitelné 6, 12 i 18 a tedy z Fermatovy věty plyne, že pro všechna celá čísla a nesoudělná s N platí $a^{\frac{N-1}{2}} \equiv 1 \pmod{N}$.

Věta. *Nechť p je liché prvočíslo. Pišme $p - 1 = 2^t \cdot q$, kde t je přirozené číslo a q je liché. Pak pro každé celé číslo a nedělitelné p buď platí $a^q \equiv 1 \pmod{p}$ nebo existuje $e \in \{0, 1, \ldots, t-1\}$ splňující $a^{2^e q} \equiv -1 \pmod{p}$.*

Důkaz. Z Fermatovy věty

$$p \mid (a^{p-1} - 1) = (a^q - 1) \cdot \prod_{e=0}^{t-1} (a^{2^e q} + 1).$$

Protože je p prvočíslo, musí dělit některého z uvedených činitelů.

Teoretický základ testu Millera a Rabina

Věta. *Nechť $N > 10$ je liché složené číslo. Pišme $N - 1 = 2^t \cdot q$, kde t je přirozené číslo a q je liché. Pak nejvýše čtvrtina z čísel množiny $\{a \in \mathbb{Z};\ 1 \leq a < N,\ (a, N) = 1\}$ splňuje následující podmínku:*

$$a^q \equiv 1 \pmod{N}$$

nebo existuje $e \in \{0, 1, \ldots, t-1\}$ splňující

$$a^{2^e q} \equiv -1 \pmod{N}.$$

Pro dané N algoritmus otestuje podmínku věty pro 20 náhodně zvolených a. Pokud pro některé takové a není podmínka splněna, vytiskne zprávu, že N je složené. Pokud je splněna podmínka pro každé takové a, vytiskne zprávu, že N je asi prvočíslo.

Podle předchozí věty je pravděpodobnost, že bude vytištěna tato zpráva, ačkoli je N složené, menší než 4^{-20}.

Algoritmus Millera a Rabina

Algoritmus (Miller - Rabin). *Pro dané liché $N \geq 3$ algoritmus s vysokou pravděpodobností objeví, že N je složené. Pokud se mu to nepodaří, vytiskne zprávu, že N je asi prvočíslo.*

1. *[Inicializace] Polož $q \leftarrow N-1$, $t \leftarrow 0$. Dokud je q sudé, opakuj $q \leftarrow \frac{q}{2}$, $t \leftarrow t+1$. Polož $c \leftarrow 20$.*
2. *[Zvol a] Pomocí generátoru náhodných čísel zvol náhodně $a \in \mathbb{Z}$, $1 < a < N$. Pak polož $e \leftarrow 0$, $b \leftarrow a^q \mod N$. Je-li $b = 1$, jdi na 4.*
3. *[Umocňuj na druhou] Dokud je $b \neq N-1$ a $e \leq t-2$, opakuj $b \leftarrow b^2 \mod N$, $e \leftarrow e+1$. Je-li $b \neq N-1$, vytiskni zprávu, že N je složené a vytiskni svědka složenosti a. Skonči.*
4. *[Už proběhlo 20 pokusů?] Polož $c \leftarrow c-1$. Je-li $c > 0$, jdi na 2. Jinak vytiskni zprávu, že N je asi prvočíslo a skonči.*

Odhad časové náročnosti. Algoritmus je řádově stejné časové náročnosti jako umocňování v něm použité (předpokládáme, že generování nového a je řádově rychlejší), proto jde o kubickou časovou náročnost.

Testy na prvočíselnost

Víme, že N je asi prvočíslo (například prošlo testem Millera a Rabina).
Chceme *dokázat*, že N skutečně prvočíslem je, nebo to vyvrátit.
Na následující větě je založen $N-1$ test Pocklingtona a Lehmera.

Věta. *Nechť N je přirozené číslo, $N > 1$. Nechť p je prvočíslo dělící $N-1$. Předpokládejme dále, že existuje $a_p \in \mathbb{Z}$ tak, že*

$$a_p^{N-1} \equiv 1 \pmod{N} \quad a \quad \left(a_p^{\frac{N-1}{p}} - 1, N\right) = 1.$$

Nechť p^{α_p} je nejvyšší mocnina p dělící $N-1$. Pak pro každý kladný dělitel d čísla N platí

$$d \equiv 1 \pmod{p^{\alpha_p}}.$$

Důkaz věty Pocklingtona a Lehmera

Věta. Nechť N je přirozené číslo, $N > 1$. Nechť p je prvočíslo dělící $N-1$. Předpokládejme dále, že existuje $a_p \in \mathbb{Z}$ tak, že

$$a_p^{N-1} \equiv 1 \pmod{N} \quad a \quad \left(a_p^{\frac{N-1}{p}} - 1, N\right) = 1.$$

Nechť p^{α_p} je nejvyšší mocnina p dělící $N-1$. Pak pro každý kladný dělitel d čísla N platí $d \equiv 1 \pmod{p^{\alpha_p}}$.

Důkaz. Každý kladný dělitel d čísla N je součinem prvočíselných dělitelů čísla N, větu dokažme pouze pro d prvočíslo. Podle Fermatovy věty platí $a_p^{d-1} \equiv 1 \pmod{d}$, neboť $(a_p, N) = 1$.
Protože $\left(a_p^{\frac{N-1}{p}} - 1, N\right) = 1$, platí $a_p^{\frac{N-1}{p}} \not\equiv 1 \pmod{d}$.
Pro $e = \min\{n \in \mathbb{N};\ a_p^n \equiv 1 \pmod{d}\}$ platí $e \mid d-1$, $e \mid N-1$ a $e \nmid \frac{N-1}{p}$. Kdyby $p^{\alpha_p} \nmid e$, z $e \mid N-1$ by plynulo $e \mid \frac{N-1}{p}$, spor. Je tedy $p^{\alpha_p} \mid e$, a tedy i $p^{\alpha_p} \mid d-1$.

Užití věty Pocklingtona a Lehmera

Věta. *Nechť N je přirozené číslo, $N > 1$. Nechť p je prvočíslo dělící $N - 1$. Předpokládejme dále, že existuje $a_p \in \mathbb{Z}$ tak, že*

$$a_p^{N-1} \equiv 1 \pmod{N} \quad a \quad \left(a_p^{\frac{N-1}{p}} - 1, N\right) = 1. \tag{1}$$

Nechť p^{α_p} je nejvyšší mocnina p dělící $N - 1$. Pak pro každý kladný dělitel d čísla N platí $d \equiv 1 \pmod{p^{\alpha_p}}$.

Důsledek. Nechť $N \in \mathbb{N}$, $N > 1$. Předpokládejme, že můžeme psát $N - 1 = F \cdot U$, kde $(F, U) = 1$ a $F > \sqrt{N}$, přičemž známe rozklad čísla F na prvočinitele. Pak platí:

- jestliže pro každé prvočíslo $p \mid F$ můžeme najít $a_p \in \mathbb{Z}$ splňující (1) z předchozí věty, pak je N prvočíslo;
- je-li N prvočíslo, pak pro libovolné prvočíslo $p \mid N - 1$ existuje $a_p \in \mathbb{Z}$ splňující (1).