Algoritmy okolo teorie čísel

Martin Mareš ⟨*mj@ucw.cz*⟩, 22. 1. 2011

Úvodem

Tento textík rozebírá několik základních algoritmických problémů souvisících s teorií čísel:

- počítání největších společných dělitelů
- řešení lineárních kongruencí
- testování prvočíselnosti (je dáno číslo a máme rozhodnout, zda je prvočíslem)
- faktorizaci (je dáno číslo a máme ho rozložit na součin prvočísel)
- šifrovací algoritmus RSA

Notace.

- *Čísly* budeme obvykle myslet čísla celá nebo přirozená, pokud nebude hrozit nedorozumění, budeme psát pouze „číslo“.
- $a \setminus b$ bude značit, že číslo a je dělitelem čísla b, tedy že existuje c takové, že $ac = b$.
- $\gcd(a, b)$ bude označovat *největšího společného dělitele* (greatest common divisor) čísel a a b, čili největší číslo c takové, že $c \setminus a \wedge c \setminus b$.
- $a \perp b$ bude značit, že a a b jsou *nesoudělná*, tedy $\gcd(a, b) = 1$.
- Často budeme počítat modulo nějaké přirozené číslo $n > 0$, tehdy budeme místo rovnosti psát znaménko $\equiv_n$, které bude značit rovnost modulo n. Pokud bude z kontextu jasné, modulo čím počítáme, budeme psát pouze $\equiv$.
- Časovou složitost algoritmů budeme vyjadřovat vzhledem k N = počtu bitů čísel, se kterými pracujeme, a elementární aritmetické operace budeme považovat za jednotkové. (Při použití dlouhočíselné aritmetiky pak stačí složitost vynásobit složitostí jedné operace, tedy typicky $\mathcal{O}(N^2)$ nebo, pokud použijeme chytřejší algoritmus na násobení, $\mathcal{O}(N \log N)$.)

Než začneme budovat algoritmy, zavedeme si pár pojmů z algebry a teorie čísel a dokážeme pár základních vět:

Definice. *(algebraické minimum)*

- *Algebra* je tvořena nosnou množinou spolu s nějakými operacemi. k-ární operací nazýváme funkci, která k-ticím prvků z nosné množiny přiřazuje opět prvky nosné množiny.
- *Komutativní grupa* (dále jen grupa) je algebra $(G, \cdot, \mathbf{1}, ^{-1})$, kde G je nosná množina, $\cdot$ binární operace nad G, $\mathbf{1}$ nulární operace nad G (čili konstanta), $^{-1}$ unární operace nad G a platí následující axiomy:
  1. $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ *(asociativita)*
  2. $a \cdot b = b \cdot a$ *(komutativita)*
  3. $a \cdot \mathbf{1} = a$ *(prvek* $\mathbf{1}$ *je jednotkový)*
  4. $a \cdot (a^{-1}) = \mathbf{1}$ *(*a^{-1} *je prvek inverzní k* a*)*
- $(H, \cdot, \mathbf{1}, ^{-1})$ je *podgrupou* grupy $(G, \cdot, \mathbf{1}, ^{-1})$ právě tehdy, když $H \subseteq G$ a množina H je uzavřená na všechny tři operace (tj. provedeme-li operaci s prvky z H, musí opět vyjít prvek z H). $(H, \cdot, \mathbf{1}, ^{-1})$ je pak také grupou.
- *Řád grupy* říkáme počtu prvků její nosné množiny.
- Pro prvek a a číslo $n \in \mathbb{Z}$ dále definujeme *mocninu* takto: $a^0 = \mathbf{1}$, $a^{n+1} = a^n \cdot a$, $a^{n-1} = a^n \cdot a^{-1}$. Platí obvyklé vlastnosti mocniny: $a^{m+n} = a^m \cdot a^n$, $a^{mn} = (a^m)^n$, $a^{-n} = (a^n)^{-1}$ apod.
- Prvku g říkáme *generátor* grupy, jestliže se každý prvek nosné množiny dá vyjádřit jako nějaká mocnina g. Grupa je *cyklická,* pokud má generátor.
- Pro libovolný prvek a je $\{a^n \mid n \in \mathbb{Z}\}$ cyklická podgrupa. *Řád prvku* a definujeme jako řád této podgrupy.

Příklad. *(různé příklady grup)*

- $(\mathbb{Z}, +, 0, -)$ (celá čísla spolu s obvyklým sčítáním, nulou a změnou znaménka) tvoří cyklickou grupu (generátory jsou 1 a -1).
- $(2\mathbb{Z}, +, 0, -)$ (sudá celá čísla spolu s obvyklým sčítáním, nulou a změnou znaménka) tvoří podgrupu předchozí grupy, která je také cyklická (rozmyslete si, že každá podgrupa cyklické grupy je cyklická),
- $(\mathbb{Z}_n, +_{\bmod n}, 0, -)$ (čísla 0, 1, …, $n-1$ spolu se sčítáním modulo n, nulou a změnou znaménka) tvoří cyklickou grupu (generátorem je třeba 1; které jsou další?),
- $(\mathbb{Q}, \cdot, 1, ?)$ (racionální čísla s násobením) nemohou tvořit grupu, jelikož k nule neexistuje inverzní prvek,
- $(\mathbb{Q} - \{0\}, \cdot, 1, 1/x)$ (racionální čísla bez nuly s násobením a převrácenou hodnotou) grupu tvoří, není cyklická.
- $(\mathbb{Z}_n - \{0\}, \cdot_{\bmod n}, ?)$ (čísla $1 \ldots n-1$ s násobením modulo n) pro některá n grupou je, pro jiná není, protože obecně nemusí existovat inverzní prvky. Za chvíli prozkoumáme, jak to přesně je.

Věta. *(Lagrangeova)* Pokud má konečná grupa G nějakou podgrupu H, je řád G dělitelný řádem H.

Důkaz: Viz libovolná skripta z algebry. [Idea: Pokryjeme G disjunktními kopiemi H.] ♡

Definice. *Multiplikativní grupa modulo n* $(\mathbb{Z}_n^*, \cdot \bmod n, 1, {}^{-1})$ je tvořena všemi invertibilními prvky v $\mathbb{Z}_n$, tj. takovými $x \in \mathbb{Z}_n$, pro něž existuje $x^{-1} \in \mathbb{Z}_n$ splňující $x \cdot x^{-1} = 1$. K tomu, aby to byla grupa, zbývá nahlédnout, že jednička je invertibilní a součin dvou invertibilních prvků je opět invertibilní ($(a \cdot b)^{-1} = a^{-1} \cdot b^{-1}$, jelikož $(a^{-1} \cdot b^{-1}) \cdot (a \cdot b) = a^{-1} \cdot b^{-1} \cdot b \cdot a = a^{-1} \cdot 1 \cdot a = a^{-1} \cdot a = 1$). Symbol '$\cdot$' již budeme obvykle vynechávat.

Pozorování. Pokud je n prvočíslem, jsou všechny prvky invertibilní – z následující věty plyne, že $x^{-1} \equiv x^{n-2}$.

Věta. *(Malá Fermatova)* Je-li p prvočíslo a $x \perp p$, platí $x^{p-1} \equiv_p 1$.

Důkaz: Můžeme provést například indukcí podle x (pro indukční krok se použije binomická věta), ale vynecháme jej, protože Malá Fermatova věta je triviálním důsledkem Eulerovy věty, již dokážeme za chvíli. ♡

K charakteristice invertibilních prvků modulo obecné n se také zkusíme propracovat. Začněme velmi tradičním algoritmem:

Algoritmus. *Euklidův algoritmus* slouží k výpočtu $\gcd(a, b)$ a je založen na následujících jednoduchých pozorováních:

- $\gcd(a, b) = \gcd(b, a)$
- $\gcd(a, a) = a$
- $\gcd(a, b) = \gcd(a, b - a)$ (dvojice (a, b) a $(a, b - a)$ dokonce sdílejí všechny společné dělitele, takže i gcd)

Kdyby Euklides nepsal své knihy řecky, ale v Pascalu, vypadal by jeho původní algoritmus asi takto:

```
function gcd(a,b:integer):integer;
begin
   while a<>b do
      if a<b then b:=b-a
             else a:=a-b;
   gcd:=a;
end;
```

Snadno ovšem nahlédneme, že opakované odečítání je možno nahradit zbytkem po dělení: (`a=:=b` značíme prohození proměnných `a` a `b`)

```
function gcd2(a,b:integer):integer;
begin
   while a<>0 do begin
      b:=b mod a;
      a=:=b;
      end;
   gcd2:=b;
end;
```

Věta. Euklidův algoritmus má pro N-bitová čísla časovou složitost $\mathcal{O}(N)$.

Důkaz: Nejprve dokážeme složitost pro případ, že a a b jsou dvě po sobě jdoucí Fibonacciho čísla, řekněme $a = F_i$ a $b = F_{i+1}$. Nahlédněme, že $b \bmod a = b - a = F_{i-1}$, čili jedna iterace převede úlohu na tentýž případ s menšími čísly. Proto se po i iteracích algoritmus zastaví.

Obecná a, b zařadíme do Fibonacciho hladin: $L(x)$ bude značit nejmenší i takové, že $F_i \geq x$. Všimneme si, že v každém kroku se $L(a) + L(b)$ sníží aspoň o 1; BÚNO $L(a) \leq L(b)$ (jinak provedeme jednu iteraci naprázdno a čísla tím prohodíme). Pokud je $L(b) > L(a)$, pak $L(b \bmod a) \leq L(a)$ a vše je v pořádku. Je-li $L(b) = L(a)$, tedy $F_i \leq a < b < F_{i+1}$, platí, že $b \bmod a \leq b - a \leq F_{i+1} - F_i = F_{i-1}$, takže $L(b \bmod a) < L(a)$ a součet hladin opět klesne.

Fibonacciho čísla ovšem rostou exponenciálně rychle ($F_n \geq \tau^n - 1$ pro $\tau = (1 + \sqrt{5})/2$), takže $L(a) + L(b) = \mathcal{O}(\log a + \log b) = \mathcal{O}(N)$. Tím je věta dokázána. ♡

Pozorování. Všimněme si ale také, že všechny mezivýsledky, a tím pádem i finální výsledek, jsou lineárními kombinacemi čísel a, b s celočíselnými koeficienty, čili že $\gcd(a, b) = ax + by$ pro nějaké $x, y \in \mathbb{Z}$. Dokonce můžeme algoritmus upravit tak, aby nám x a y našel. Takovému algoritmu se pak často říká Rozšířený Euklidův:

```
function gcd3(a,b:integer; var x,y:integer):integer;
var ax,bx,ay,by,m:integer;
begin
   ax:=1; ay:=0;                { jakou lin.komb. je aktuální a }
   bx:=0; by:=1;                { a jakou aktuální b }
   while a<>0 do begin
      m:=b div a;
      b:=b-m*a;                 { b:=b mod a }
      bx:=bx-m*ax;              { upravíme koeficienty lin.komb. }
      by:=by-m*ay;
```

```
        a=:=b; ax=:=bx; ay=:=by;
        end;
    gcd3:=b;
    x:=bx; y:=by;
end;
```

To nám pomůže k sestrojení algoritmu pro řešení lineárních rovnic modulo n (kongruencí):

Věta. Mějme rovnici $ax \equiv_n b$ (x je neznámá), označme $g = \gcd(a, n)$. Pak tato rovnice má řešení právě tehdy, když $g \setminus b$, a je jej možno nalézt v čase $\mathcal{O}(N)$.

Důkaz: Rovnici můžeme zavedením nové neznámé y upravit na $ax + ny = b$ (pokud se obě strany rovnají modulo n, pak to znamená, že se liší o nějaký násobek n). Pokud je $b = g$, vyřeší ji již popsaný Rozšířený Euklidův algoritmus. Pokud je $b = kg$ pro nějaké k, vyřešíme rovnici nejdříve pro g na pravé straně a řešení pak vynásobíme číslem k. Pokud není b dělitelné číslem g, rovnice nemůže mít řešení, protože levá strana je vždy dělitelná g, zatímco pravá nikdy. ♡

Vraťme se nyní k invertibilním prvkům: a je invertibilní modulo n právě tehdy, když má řešení kongruence $ax \equiv_n 1$. A to již víme, kdy nastává, čili jsme právě dokázali:

Věta. Invertibilní prvky modulo n jsou právě čísla nesoudělná s n.

Definice. Počet čísel mezi 1 a $n-1$ nesoudělných s n budeme značit *Eulerovou funkcí* $\varphi(n)$.

Pozorování. *(vlastnosti funkce φ)*

- $\varphi(n) = |\mathbb{Z}_n^*|$ (to říká předchozí věta)
- $\varphi(p) = p - 1$, je-li p prvočíslo (rovnou z definice)
- $\varphi(p^k) = p^{k-1} \cdot (p-1)$ (soudělná s p^k jsou čísla dělitelná p, takže každé p-té)
- $\varphi(ab) = \varphi(a) \cdot \varphi(b)$, pokud je $a \perp b$ (tehdy je $\gcd(ab, x) = \gcd(a, x) \cdot \gcd(b, x)$, takže $x \perp ab \Leftrightarrow x \perp a \wedge x \perp b$).
- Známe-li faktorizaci čísla n, dokážeme $\varphi(n)$ spočítat v čase $\mathcal{O}(\textit{počet faktorů}) = \mathcal{O}(N)$.

Funkce $\varphi(n)$ také figuruje v následujícím šikovném zobecnění Malé Fermatovy věty:

Věta. *(Eulerova)* Pro libovolné $n > 0$ a $x \perp n$ platí $x^{\varphi(n)} \equiv_n 1$.

Důkaz: Počítejme modulo n. Uvažme posloupnost $x^1, x^2, x^3, \ldots$. V této posloupnosti se určitě vyskytuje jednička, protože možných hodnot x^m je pouze konečně mnoho (nejvýše n). Tedy pro nějaká i, j ($i < j$) je $x^i \equiv x^j$, a proto $x^{j-i} \equiv 1$. Vyberme si nejmenší $m \geq 1$, pro které je $x^m \equiv 1$.

Všimněme si, že čísla x^0, x^1, x^2, …, x^{m-1} tvoří podgrupu multiplikativní grupy $\mathbb{Z}_n^*$ a tato podgrupa má řád m. Proto podle Lagrangeovy věty musí být $m \setminus \varphi(n)$, tedy $\varphi(n) = km$ pro nějaké k. Proto $x^{\varphi(n)} \equiv x^{km} \equiv 1^k \equiv 1$. ♡

Ještě se nám bude hodit následující charakterizace grupy $\mathbb{Z}_n^*$, kterou uvedeme bez důkazu (důkaz naleznete například v Kaleidoskopu teorie čísel od Martina Klazara nebo i ve většině učebnic algebry):

Věta. *(o multiplikativní grupě)* Multiplikativní grupa $\mathbb{Z}_n^*$ je cyklická grupa řádu $n-1$ právě tehdy, když n je prvočíslo.

Část důkazu: Pokud n není prvočíslo, nutně existuje $x < n$ soudělné s n, takže alespoň jeden z prvků $< n$ není invertibilní. Pro prvočíselné n má $\mathbb{Z}_n^*$ řád $n-1$, zbývá dokázat cykličnost.

Zamysleme se ještě nad tím, jak je to v $\mathbb{Z}_n^*$ s odmocninami – kupříkladu pro $n = 5$ je $1^2 = 4^2 = 1$ a $2^2 = 3^2 = 4$, takže 1 a 4 mají dvě odmocniny, zatímco 2 a 3 ani jednu. To není náhoda:

Věta. *(o diskrétní odmocnině)* Pokud je p liché prvočíslo a $x \in \mathbb{Z}_p^*$, pak má x buďto právě dvě odmocniny, nebo nemá žádnou.

Důkaz: Využijeme cykličnosti multiplikativní grupy. Buď g její generátor a g^0, g^1, …, g^{p-2} všechny prvky grupy. Existuje tedy nějaké i takové, že $x = g^i$, a my hledáme y splňující $y^2 = x$, čili j, pro něž $(g^j)^2 = g^i$. Tehdy musí platit $2j \equiv_{p-1} i$. Jelikož $p-1$ je vždy sudé, nemůže pro liché i existovat žádné takové j, zatímco pro sudé i podmínku splňuje $j = i/2$ a $j = i/2 + (p-1)/2$. ♡

Pozorování. *(o odmocninách modulo prvočíslo)* Odmocniny z jedničky tedy musí být 1 a -1 (u těchto dvou to ověříme snadno a již víme, že žádné další neexistují). Jako vedlejší produkt našeho důkazu jsme také zjistili, že odmocninu má právě polovina prvků $\mathbb{Z}_p^*$ (sudé mocniny generátoru) a že $g^{\frac{p-1}{2}} \equiv -1$.

Poznámka. Jelikož podle Malé Fermatovy věty je $x^{p-1} \equiv_n 1$, musí $x^{\frac{p-1}{2}}$ být odmocnina z jedničky, čili buďto 1 nebo -1. A podle toho, která z těchto možností nastane, dokonce snadno poznáme, zda je x odmocnitelné: $(g^{2k})^{(p-1)/2} \equiv g^{k(p-1)} \equiv 1^k \equiv 1$, zatímco $(g^{2k+1})^{(p-1)/2} \equiv g^{k(p-1)} \cdot g^{(p-1)/2} \equiv 1 \cdot (-1) \equiv -1$.

Konečně se nám bude hodit také známá Čínská věta o zbytcích:

Věta. *(Čínská o zbytcích)* Buďte $a_1, \ldots, a_t$ navzájem nesoudělná čísla a $n = a_1 \cdot \ldots \cdot a_t$. Potom pro všechny t-tice $b_1, \ldots, b_t$ takové, že $\forall i\ 0 \leq b_i < a_i$, existuje právě jedno x, $0 \leq x < n$, pro které je $x \bmod a_i = b_i$ pro všechna i. [Jinými slovy číslo ze $\mathbb{Z}_n$ je jednoznačně určeno svými zbytky po dělení $a_1, \ldots, a_t$.]

Důkaz: Nejprve ověříme jednoznačnost. Kdyby x a x' byla nějaká dvě čísla vyhovující podmínkám věty, musí být rozdíl $x - x'$ dělitelný všemi a_i, a tedy i číslem n. To ovšem není možné, pokud jsou $x, x' \in \langle 0, N)$.

Existenci ukážeme následovně: Mějme zobrazení $z : \mathbb{Z}_n \to \mathbb{Z}_{a_1} \times \ldots \times \mathbb{Z}_{a_t}$, které každému číslu přiřadí příslušnou t-tici zbytků. Z jednoznačnosti plyne, že toto zobrazení je prosté. Na druhou stranu jeho definiční obor a obor hodnot jsou stejně velké konečné množiny (obě mají n prvků), takže zobrazení musí být i na. ♡

Důkaz č. 2 (pracnější, ale dává nám algoritmus pro výpočet x): Kdybychom uměli nalézt čísla $z_1, \ldots, z_t \in \mathbb{Z}_n$ taková, že $z_i \equiv_{a_i} 1$ a $z_i \equiv_{a_j} 0$ pro všechna $i \neq j$, měli bychom vyhráno. Můžeme totiž zvolit $x \equiv_n \sum_i b_i z_i$. Potom $x \equiv_{a_j} b_j z_j \equiv_{a_j} b_j$ (členy pro $i \neq j$ jsou nulové).

Kde vzít z_i? Nejprve spočteme $q_i = a_1 \ldots a_{i-1} a_{i+1} \ldots a_t$. To je jistě číslo menší než n, jež je dělitelné všemi a_j pro $j \neq i$, jen modulo a_i dá nějaké nenulové číslo. Pokud jedničku, jsme hotovi, jinak zvolíme multiplikativní inverzi modulo a_i, tedy w_i takové, že $q_i w_i \equiv_{a_i} 1$. Pak stačí položit $z_i \equiv_n q_i w_i$ a jsme hotovi: toto z_i dává modulo a_i jedničku a všemi ostatními a_j je nadále dělitelné.

Jakou má celý výpočet časovou složitost? Všechna q_i zvládneme spočítat v čase $\mathcal{O}(t)$, neboť $q_{i+1} = q_i / a_{i+1} \cdot a_i$. Výpočet každého z_i pak obnáší jedno násobení a jedno řešení lineární kongruence, tedy $\mathcal{O}(N)$ operací. Složení výsledku ze z_i pak stihneme v $\mathcal{O}(t)$. Celkem tedy spotřebujeme čas $\mathcal{O}(tN)$. ♡

Testování prvočíselnosti a faktorizace

Rozhodnout, zda dané číslo x je prvočíslem, můžeme poměrně snadno zkoušením dělitelnosti všemi potenciálními děliteli, nicméně takový algoritmus je vzhledem k délce vstupu (čili počtu bitů čísla x) exponenciální. Zda existuje polynomiální algoritmus, bylo několik desítek let otevřeno. V létě 2002 ovšem prof. Manindra Agrawal spolu se svými dvěma studenty (Neeraj Kayal a Nitin Saxena) publikovali polynomiální algoritmus pracující v čase $\mathcal{O}(n^{6.5})$ (při konstantní složitosti aritmetických operací). Tento algoritmus je poměrně komplikovaný a odhad jeho složitosti vyžaduje dosti netriviální věty z teorie čísel (čtěte: autor tohoto spisku jim věří, ale dokázat by je neuměl). Existuje ovšem celá řada elegantních a rychlých randomizovaných algoritmů pro tento problém, jeden z nich si za chvíli předvedeme.

Naproti tomu o *faktorizaci* čísel, čili rozkladu na součin prvočísel, dosud není známo, zda polynomiální algoritmus existuje, a ani se nepodařilo dokázat nic netriviálního o složitosti tohoto problému. Je jasné, že je v *NP*, ale vesměs se nepředpokládá, že by byl *NP*-úplný. Zatím nejlepší známý randomizovaný algoritmus běží v čase $2^{\mathcal{O}(N^{1/3} \cdot (\log N)^{2/3})}$, čili výrazně lepším, než exponenciálním, ale také o mnoho horším, než polynomiálním.

Pro testování prvočíselnosti sestrojíme algoritmus třídy *co-RP* (Monte Carlo s jednostrannou chybou), tedy algoritmus, který v polynomiálním čase vydá výsledek, který o prvočísle vždy řekne, že je to prvočíslo a pro složené číslo se může s pravděpodobností nejvýše $1/4$ zmýlit. To samo o sobě není moc užitečné, ale pokud algoritmus spustíme k-krát nezávisle a odpovíme, že n je složené, pokud jsme se to dozvěděli při alespoň jednom spuštění, pravděpodobnost chyby klesne na $1/4^k$ a to již pro $k = 100$ zcela postačuje. [Na tomto místě se často uvádí srovnání s pravděpodobností, že váš počítač zasáhne meteorit, ale to je tak trochu podvod, protože zatím nás nezasáhlo tolik meteoritů, abychom takovou pravděpodobnost zvládli realisticky odhadnout; navíc ani nevíme, jestli zásahy jsou opravdu náhodné.]

Nabízí se následující algoritmus: náhodně volit x mezi 2 a $n-1$ a testovat, zda se nestrefíme (meteoritem?) do dělitele zadaného čísla n. Ten má jistě jednostrannou chybu, nicméně její pravděpodobnost je příliš velká: pokud je n součinem dvou velkých prvočísel, má pouze 2 netriviální dělitele, které objevíme s pravděpodobností pouze $2/(n-2)$.

Pomocí Fermatovy věty ale můžeme o číslu dokázat, že je složené, aniž bychom našli konkrétního dělitele:

Algoritmus. *(Fermatův test)* Náhodně zvolíme x mezi 2 a $n-1$ a spočteme $x^{n-1} \bmod n$. Pokud vyjde 1, prohlásíme n za prvočíslo, jinak za číslo složené.

Pozorování. x^k můžeme snadno spočíst v polynomiálním čase: nejprve spočteme x^1, x^2, $x^4 = (x^2)^2$, …, $x^{2^{\lfloor \log_2 n \rfloor}}$, pak k vyjádříme jako součet mocnin dvojky (čili ho rozložíme do dvojkové soustavy) a spočteme $x^k = x^{2^{i_1} + \ldots + 2^{i_l}} = x^{2^{i_1}} \cdot \ldots \cdot x^{2^{i_l}}$. Celkem jsme použili $\mathcal{O}(\log k) = \mathcal{O}(N)$ násobení.

Poznámka. Abychom mohli používat Fermatovu větu, měli bychom si ještě ověřit, že $x \perp n$ (spočítat gcd a pokud je různé od jedné, číslo odmítnout jako složené). Všimněte si ale, že pokud je g společným dělitelem x a n, je $x^{n-1} \bmod n$ také dělitelné g. Proto pro $x \not\perp n$ i původní test číslo prohlásí za složené, takže tato modifikace není potřeba.

Pozorování. Fermatův test tedy pracuje v polynomiálním čase, prvočíslo vždy prohlásí za prvočíslo a složené číslo někdy za složené (najde-li x, pro které je $x^{n-1} \not\equiv 1$ – takovému x se říká *Fermatův svědek* složenosti x), jindy za prvočíslo (pokud se zrovna do svědka nestrefí). Čeká nás ale jedno nemilé překvapení: existují složená čísla, která žádného Fermatova svědka nemají – takovým se říká *Carmichaelova čísla* a je jich nekonečně mnoho. Za domácí úkol si najděte nejmenší Carmichaelovo číslo (je menší než 1 000). Naštěstí ale pokud tento problém nenastane, Fermatových svědků je dostatek:

Věta. Pokud n není ani prvočíslo, ani Carmichaelovo číslo, platí $x^{n-1} \equiv_n 1$ pro nejvýše $n/2$ různých x.

Důkaz: Zvolme $H = \{x \mid x^{n-1} \equiv 1\}$. Všechna čísla v H jsou jistě invertibilní a jejich součin je opět v H. Proto je H podgrupa multiplikativní grupy $\mathbb{Z}_n^*$ a podle Lagrangeovy věty musí pro nějaké k platit $|H| \cdot k = \varphi(n)$. Navíc n není

Carmichaelovo, takže $H \neq \mathbb{Z}_n^*$, a proto $k \geq 2$. „Špatných“ x je tedy $|H| \leq \varphi(n)/2 \leq n/2$. ♡

Zbývá tedy ošetřit Carmichaelova čísla. To bohužel není zrovna jednoduché, ale zabere například následující postup:

Algoritmus. *(Rabinův-Millerův test)*

1. Vygenerujeme náhodné x $(1 < x < n)$.
2. Pokud $x \not\perp n$, odpovíme SLOŽENÉ ($\gcd(x, n)$ je netriviální dělitel).
3. Najdeme t a liché m taková, že $n - 1 = 2^t \cdot m$.
4. Spočteme $b_0 \equiv x^m$.
5. Spočteme $b_1, \ldots, b_t$: $b_{i+1} \equiv b_i^2$ (tudíž $b_t \equiv x^{n-1}$).
6. Pokud je $b_t \not\equiv 1$, odpovíme SLOŽENÉ (x je toho Fermatův svědek).
7. Pokud jsou všechna $b_0, \ldots, b_t \equiv 1$, odpovíme PRVOČÍSLO.
8. Jinak vezmeme nejvyšší i, pro něž $b_i \not\equiv 1$:
   a) Pokud je $b_i \equiv -1$, odpovíme PRVOČÍSLO.
   b) Jinak odpovíme SLOŽENÉ (a x nazveme *Riemannovým svědkem*).

Krok 2 můžeme stejně jako u Fermatova testu vypustit, ale jeho přítomnost nám usnadní analýzu algoritmu.

Na tento algoritmus se můžeme dívat i jinak: nejdříve spočteme x^{n-1} (zajisté mod n). Pokud nevyjde jednička, je n složené podle Fermatova testu. Pokud vyjde a $n - 1$ je sudé, musí být $x^{(n-1)/2}$ odmocninou z jedničky. Již víme, že tyto odmocniny existují modulo prvočíslo jen dvě: 1 a -1. Pokud tedy vyjde něco jiného, n jistě není prvočíslo. Pokud vyjde opět jednička, pokračujeme v odmocňování a počítáme $x^{(n-1)/4}$, $x^{(n-1)/8}$, atd. Zastavíme se, když narazíme na -1 (tehdy odpovíme PRVOČÍSLO), nebo na něco jiného než ± 1, (SLOŽENÉ) nebo když exponent přestane být celočíselný (PRVOČÍSLO). Z této úvahy plyne, že kdykoliv odpovíme SLOŽENÉ, je to pravda.

Důležité ovšem je, že na rozdíl od Fermatova testu se Rabinův-Millerův test nenechá zmást Carmichaelovými čísly a nalezne pro ně dostatek svědků:

Věta. Rabinův-Millerův test testuje prvočíselnost v polynomiálním čase, na prvočísla odpovídá správně a složená čísla prohlašuje za prvočísla s pravděpodobností nejvýše 1/4.

Důkaz: poměrně náročný, naleznete ho například v knize Randomized Algorithms od pánů Motwaniho a Raghavana, případně jeho zjednodušenou verzi (pro konstantu 1/2 namísto 1/4) v jedné z dalších kapitol tohoto spisku. ♡

Poznámka. Za zmínku ještě stojí, že původní Millerův test byl deterministický a pan Miller o něm dokázal, že pokud platí rozšířená Riemannova hypotéza, má každé složené číslo svědka (Fermatova či Riemannova), který je jen logaritmicky velký. Zda je to pravda, to se dosud neví, nicméně pan Rabin později nahlédl, že svědků vždy existuje alespoň $3/4 \cdot n$ a randomizovaný algoritmus byl na světě.

Šifrovací systém RSA

Šifrovací systém RSA (nazvaný podle autorů pánů Rivesta, Shamira a Adlemana) je velmi hezkým a v praxi často používaným příkladem *asymetrické šifry.* To je šifra pracující se dvěma klíči – šifrovacím a dešifrovacím – splňující, že co zašifrujete pomocí šifrovacího klíče, dá se rozšifrovat pouze pomocí dešifrovacího.

Poznámka. Ono „dá se rozluštit pouze . . . “ je samozřejmě definice dosti mlhavá. Mělo by to znamenat, že se znalostí správného klíče lze cokoliv dešifrovat efektivně (řekněme v polynomiálním čase), zatímco bez jeho znalosti to možné není (řekněme je potřeba alespoň exponenciální čas). To je pěkná definice (a pokud budeme měřit složitost pro *průměrný* vstup místo pro nejhorší případ, tak i rozumná), ale bohužel o žádné asymetrické šifře zatím neumíme dokázat, že tuto definici splňuje. Takže si RSA budeme muset předvést bez důkazu, vedeni spíše vírou v jeho solidnost, podloženou tím, že ho zatím nerozlousknul ani nikdo jiný :)

Poznámka. Ani není divu, že dokázat nerozluštitelnost šifry je těžké – kdyby bylo $P = NP$, každou funkci spočítatelnou v polynomiálním čase je možno v polynomiálním čase i invertovat (rozmyslete si, proč to tak je), takže by šifra podle naší definice ani existovat nemohla. Kdyby tedy existovala, uměli bychom dokázat, že $P \neq NP$ a byli bychom slavní.

Definice. *(konstrukce klíče pro RSA)* Zvolme následující parametry:

- p a q – náhodná velká prvočísla (najdeme je např. pomocí Rabinova-Millerova testu),
- $n = pq$ – jejich součin,
- $\varphi(n)$ – Eulerova funkce od n, ze znalosti rozkladu n ji umíme spočítat,
- x, y – náhodný invertibilní prvek modulo $\varphi(n)$ a jeho inverze (vygenerujeme náhodné x mezi 1 a $\varphi(n) - 1$, pomocí rozšířeného Euklidova algoritmu zkusíme spočítat jeho inverzi a nebude-li existovat, x zahodíme a vygenerujeme jiné atd.),
- *Šifrovací klíč* (n, x),
- *Dešifrovací klíč* (n, y).

Algoritmus. *(šifra RSA)* Pro zašifrování používáme funkci $E_{(n,x)}(a) = a^x \bmod n$, pro dešifrování pak $D_{(n,y)}(b) = b^y \bmod n$.

Věta. *(korektnost RSA)* $D_{(n,y)}(E_{(n,x)}(a)) \equiv_n a$. (čili dešifrování je modulo n inverzní k zašifrování)

Důkaz: Nejprve předpokládejme, že $a \perp n$. Tehdy $D(E(a)) \equiv a^{xy} \equiv a^{1+k\varphi(n)} \equiv a \cdot (a^{\varphi(n)})^k \equiv_n a$. (Druhá rovnost platí, jelikož y je inverzí x modulo $\varphi(n)$; $a^{\varphi(n)} = 1$ podle Eulerovy věty.)

Pokud by $\gcd(a, n) \neq 1$, bylo by $p \setminus a$ nebo $q \setminus a$ a Eulerovu větu bychom nemohli použít. Tehdy lze ale důkaz vést jinudy: Nechť $a = pz$ (druhý případ symetricky). Spočtěme a^{xy} mod p: $a^{xy} \equiv (pz)^{xy} \equiv 0 \equiv pz \equiv_p a$. A teď modulo q, využívajíce toho, že $p \perp q$, $z \perp q$, a tedy i $a \perp q$: $a^{xy} \equiv a^{1+k\varphi(n)} \equiv a^{1+k\varphi(p)\varphi(q)} \equiv a \cdot (a^{\varphi(q)})^{k\varphi(p)} \equiv a \cdot 1^{k\varphi(p)} \equiv_q a$. (Opět v hlavní roli Eulerova věta.)

Zbývá dodat, že pokud si jsou libovolná čísla f a g rovna modulo p i modulo q, musí si být rovna i modulo n, což plyne například z Čínské věty o zbytcích. ♡

Pozorování. Všimněte si, že RSA nijak nerozlišuje x a y – libovolný z nich může být šifrovacím klíčem, ten druhý je pak dešifrovacím. Také ho samozřejmě můžeme používat jako *symetrickou šifru,* tj. šifru s jedním jediným tajným klíčem sloužícím jak k šifrování, tak k dešifrování (tehdy je součástí klíče x i y). Navíc je RSA *komutativní šifra* – pokud zprávu zašifrujeme dvěma různými klíči, nezáleží na tom, v jakém pořadí budeme oběma klíči dešifrovat. To vede k jedné velice hezké aplikaci řešící odvěké problémy s výměnou klíčů:

Algoritmus. *(komutativní šifrovací protokol)* Nechť osoba A chce poslat zprávu x osobě B, ale A a B se předem neviděly a nemají možnost si bezpečně vyměnit klíče, zato se dokázaly shodnout na nějaké (klidně veřejně známé) komutativní šifře. Zprávu si mohou předat takto: A zašifruje x svým klíčem, pošle ji B, ten ji zašifruje ještě navíc svým a vrátí ji zpět; A tuto zprávu dešifruje a vrátí ji B. Tou dobou je zpráva zašifrována klíčem B a osoba B ji tedy dokáže dešifrovat. Pro korektnost stačí komutativita šifry, bezpečnost plyne z toho, že zpráva je vždy přenášena ve tvaru zakódovaném aspoň jedním klíčem, který nikdo mimo A a B nezná.

Poznámka. Kdybychom uměli efektivně faktorizovat velká čísla, je RSA samozřejmě snadno rozlousknutelná: jakmile známe faktorizaci n, spočteme si $\varphi(n)$ a pak i multiplikativní inverzi x modulo $\varphi(n)$, což je y.

Ještě jeden test prvočíselnosti

Nakonec předvedeme ještě jeden algoritmus pro pravděpodobnostní testování prvočísel, tentokrát i s důkazem korektnosti. Daní za jednoduchost důkazu ovšem bude to, že náš test může udělat chybu na obě strany: jak prohlásit složené číslo za prvočíslo, tak prvočíslo za složené. Bude fungovat následovně: (n je testované číslo, t počet iterací, na kterém bude záviset pravděpodobnost chyby)

Algoritmus.

1. Pokud je n netriviální mocninou nějakého přirozeného čísla, odpovíme SLOŽENÉ.
2. Vygenerujeme náhodná $a_1, \ldots, a_t \in \mathbb{Z}_n \setminus \{0\}$.
3. Pokud pro nějaké i je $\gcd(a_i, n) \neq 1$, odpovíme SLOŽENÉ.
4. Spočítáme $r_i = a_i^{(n-1)/2} \bmod n$ pro všechna i.
5. Pokud pro nějaké i je $r_i \not\equiv \pm 1$, odpovíme SLOŽENÉ.
6. Pokud pro všechna i je $r_i = 1$, odpovíme SLOŽENÉ.
7. Jinak odpovíme PRVOČÍSLO.

Nejprve si všimneme, že algoritmus běží v polynomiálním čase. Největší společné dělitele a mocniny modulo n už polynomiálně umíme počítat, jediný problematický krok je ten první. V něm ale stačí zkoušet všechny možné exponenty (těch je $\mathcal{O}(\log n) = \mathcal{O}(N)$, jelikož základ je alespoň 2) a pro každý exponent hledat pomocí půlení intervalu odmocninu (opět $\mathcal{O}(N)$ kroků).

Nyní nahlédněme, jak algoritmus probíhá pro prvočísla. První ani třetí krok prvočíslo neodmítnou, pátý také ne (vzpomeňme si na poznámku o odmocninách modulo prvočíslo), jediný problém může nastat v šestém kroku. Již víme, že $r_i = 1$ právě tehdy, má-li a_i druhou odmocninu, a to nastane s pravděpodobností $1/2$. Šestý krok tedy odpoví SLOŽENÉ jen tehdy, když jsou všechna a_i odmocnitelná, pravděpodobnost čehož je $1/2^t$.

Složené číslo naopak prohlásíme za prvočíslo jen tehdy, pokud jsou všechna $a_i \in \mathbb{Z}_n^*$, nalezneme alespoň jedno $r_i \equiv -1$ a všechna ostatní r_j jsou buďto 1 nebo -1. K odhadu pravděpodobnosti tohoto nám poslouží následující lemma:

Lemma. Buď n složené číslo, které není mocninou prvočísla. Nechť pro nějaké $a \in \mathbb{Z}_n^*$ je $a^{(n-1)/2} \equiv_n -1$. Pak množina $S_n = \{x \in \mathbb{Z}_n^* \mid x^{(n-1)/2} \equiv_n \pm 1\}$ obsahuje nejvýše $|\mathbb{Z}_n^*|/2$ prvků.

Důkaz: Podobně jako u Fermatova testu: Všimneme si, že S_n je podgrupa $\mathbb{Z}_n^*$, takže zbývá dokázat, že je to podgrupa netriviální, a použít Lagrangeovu větu. Najdeme číslo b, které nebude ležet v S_n.

Nechť n má prvočíselný rozklad $p_1^{k_1} \cdot \ldots \cdot p_s^{k_s}$. Již víme, že $s \geq 2$. Označme $q = p_1^{k_1}$ a $m = n/q$. Jelikož $q \setminus n$ i $m \setminus n$, musí být pro každý prvek $x \in S_n$ jak $x^{(n-1)/2} \equiv_q \pm 1$, tak $x^{(n-1)/2} \equiv_m \pm 1$ a znaménka obou zbytků jsou stejná.

Kýžené číslo b zvolíme tak, aby pro něj platilo $b \equiv_q a$ a současně $b \equiv_m 1$ (Čínská věta o zbytcích nám jeho existenci zaručuje, jelikož $q \perp m$). Snadno ověříme, že platí:

$$b^{(n-1)/2} \equiv_q a^{(n-1)/2} \equiv_q -1,$$

$$b^{(n-1)/2} \equiv_m 1.$$

Takové b ovšem neleží v S_n, protože jak už jsme pozorovali, pro každý prvek z S_n jsou zbytky po dělení q a m stejné a my jsme si b zvolili tak, aby byly různé. ♡

Náš algoritmus tudíž selže jedině tehdy, když $a_2, \ldots, a_t$ padnou všechna do S_n, a to nastane s pravděpodobností nejvýše $1/2^{t-1}$. Sečteno a podtrženo, dokázali jsme následující větu:

Věta. Prvočíselný test z této kapitoly má při t iteracích pravděpodobnost chyby nejvýše $1/2^{t-1}$.

Rabin a Miller se vrací

O Rabinově-Millerově testu již víme, že prvočíslo vždy prohlásí za prvočíslo a že složené, číslo, které není Carmichaelovo, prohlásí za složené s pravděpodobností alespoň $1/2$. V této kapitole dokážeme, že je to pravda i pro Carmichaelova čísla. Nejprve si připravíme půdu jedním drobným lemmatem:

Lemma. Žádné Carmichaelovo číslo není mocninou prvočísla (druhou nebo větší).

Důkaz: Uvažujme libovolné $n = p^e$, kde p je prvočíslo a $e > 1$. Zvolíme $a = 1 + p^{e-1}$ a podle binomické věty spočteme a^p (vše počítáme v Z_n^*, kam a jistě patří):

$$a^p \equiv (1 + p^{e-1})^p \equiv \binom{p}{0} \cdot 1 \cdot 1 + \binom{p}{1} \cdot 1 \cdot p^{e-1} + \binom{p}{2} \cdot 1 \cdot p^{2(e-1)} + \ldots + \binom{p}{p} \cdot 1 \cdot p^{p(e-1)} \equiv 1$$

(všechny členy mimo nultého jsou totiž dělitelné p^e – prvnímu pomůže kombinační číslo, u ostatních stačí vyšší mocnina p^{e-1}). Proto také $a^n \equiv (a^p)^e \equiv 1$. Tedy $a^{n-1} \equiv a^{-1} \not\equiv 1$, takže a není Carmichaelovo. ♡

Nyní uvažujme, kdy může Rabinův-Millerův test odpovědět, že číslo je prvočíslem. Stane se tak buď v kroku 5 (všechna $b_0, \ldots, b_t$ jsou jedničky, což nastane, kdykoliv $b_0 \equiv 1$) nebo v kroku 8 (nějaké $b_i \equiv -1$ a $b_{i+1} \equiv \ldots \equiv b_t \equiv 1$). Rozebereme postupně oba případy.

Lemma. Buď n Carmichaelovo a $n - 1 = 2^t \cdot m$ jako v algoritmu. Poté existuje alespoň $|\mathbb{Z}_n^*|/2$ čísel $x \in \mathbb{Z}_n^*$, pro něž $b_0 := x^m \not\equiv_n 1$.

Důkaz: Podobnou úvahou založenou na Lagrangeově větě, jako jsme použili u Fermatova testu. Množina $B := \{b \in \mathbb{Z}_n^* \mid b^m \equiv 1\}$ tvoří podgrupu $\mathbb{Z}_n^*$, takže zbývá ukázat, že alespoň jeden prvek $a \in \mathbb{Z}_n^*$ neleží v B.

Buď p nějaký prvočíselný dělitel čísla n. Zvolme $a \in \mathbb{Z}_p^*$, které není modulo p odmocnitelné. Už víme, že takových čísel existuje $(p-1)/2$ a že splňují následující vlastnosti:

$$a^{p-1} \equiv_p 1,$$
$$a^{(p-1)/2} \equiv_p -1.$$

Uvažujme podgrupu $H \subseteq \mathbb{Z}_p$ generovanou prvkem a (tedy množinu $\{a^0, a^1, a^2, \ldots\}$). Z předchozích dvou rovností vyplývá, že řád této podgrupy dělí $p-1$, ale nedělí $(p-1)/2$, takže řád musí být sudé číslo. Pro liché m tedy nemůže platit $a^m \equiv_p 1$, takže ani $a^m \equiv_n 1$. ♡

Nyní se přesuneme ke kroku 8. Z přechozího lemmatu víme, že pro některé volby čísla x v algoritmu je $b_0 \not\equiv 1$. Můžeme proto zvolit i $(0 \leq i < t)$ takové, že $b_{i+1} \equiv x^{2^{i+1}m} \equiv 1$ pro všechna x, ale $b_i \equiv x^{2^i m} \not\equiv 1$ pro alespoň jedno x. Jakmile dokážeme, že $b_i \not\equiv \pm 1$ pro alespoň polovinu z možných x, máme vyhráno.

Lemma. Pro n Carmichaelovo, $n - 1 = 2^t \cdot m$ a i definované podle předchozího odstavce existuje alespoň $|\mathbb{Z}_n^*|/2$ čísel $x \in \mathbb{Z}_n^*$ takových, že $x^{2^i m} \not\equiv_n \pm 1$.

Důkaz: Ještě jednou stejný trik s podgroupou. Tentokrát zvolíme $G := \{x \in \mathbb{Z}_n^* \mid x^{2^i m} \equiv \pm 1\}$, což je evidentně podgrupa $\mathbb{Z}_n^*$, a opět chceme dokázat, že alespoň jeden prvek leží mimo ni.

Z volby i víme, že existuje c, pro něž $c^{2^i m} \not\equiv 1$. Pokud $c^{2^i m} \not\equiv -1$, máme vyhráno, neboť takové c neleží v G. V opačném případě zvolíme nějakou mocninu prvočísla p^e z prvočíselného rozkladu čísla n. Jelikož $c^{2^i m} \equiv_n -1$, musí tato kongruence platit i modulo p^e. Nyní pomocí Čínské věty o zbytcích najdeme d tak, aby splňovalo současně $d \equiv_{p^e} c$ a $d \equiv_{n/p^e} 1$ (zde jsme potřebovali, že n není mocnina prvočísla). Spočítáme-li $(2^i m)$-tou mocninu čísla d modulo jak p^e, tak n/p^e, dostaneme $d^{2^i m} \equiv_{p^e} c^{2^i m} \equiv_{p^e} -1$ a $d^{2^i m} \equiv_{n/p^e} 1$. Proto $d^{2^i m}$ nemůže být modulo n ani 1, ani -1, takže $d \notin G$. ♡