Cryptology

Generátory náhodných čísel a hašovací funkce

Definition 1 (Zanedbatelná funkce)**.** *$\mu : \mathbb{N} \mapsto \mathbb{R}_+$ je zanedbatelná, jestliže pro každý polynom p existuje $n_p \in \mathbb{N}$ takové, že*

$$\forall n \geq n_p : \mu(n) \leq \frac{1}{p(n)}.$$

V této části nás budou zajímat především soubory náhodných veličin $\{X_i\}_{i\in\mathbb{N}}$, X_i nabývá hodnot z $\{0,1\}^{l(i)}$ pro $l(i) : \mathbb{N} \mapsto \mathbb{N}$. Obvykle $l(i) \geq i$, nicméně často budeme uvažovat $l(i) = i$. Pro tyto soubory se budeme snažit zjistit podobnost se soubory $\{U_n\}_{n\in\mathbb{N}}$ příslušných rovnoměrně rozdělených náhodných veličin.

Definition 2 (Blízkost souborů)**.** *Buďtež $\{X_i\}_{i\in\mathbb{N}}$ a $\{Y_i\}_{i\in\mathbb{N}}$ dva soubory náhodných veličin. Pak rozlišujeme tyto stupně blízkosti souborů náhodných veličin:*

- *Soubory jsou identické, pokud pro všechna $n \in \mathbb{N}$ platí $X_n = Y_n$, tj. X_n a Y_n mají stejné rozdělení.*
- *Soubory jsou statisticky blízké, jestliže jejich statistická diference $\Delta_{X,Y}(n)$ je zanedbatelná funkce.*

$$\Delta_{X,Y}(n) = \frac{1}{2}\Sigma_{v\in\{0,1\}^n}|Pr[X_n = v] - Pr[Y_n = v]|$$

- *Soubory jsou výpočetně nerozlišitelné, jestliže pro každý polynomiální pravděpodobnostní algoritmus D je $\delta_{D,X,Y}(n)$ zanedbatelná funkce.*

$$\delta_{D,X,Y}(n) = |Pr[D(X_n) = 1] - Pr[D(Y_n) = 1]|$$

- *Soubory jsou silně výpočetně nerozlišitelné, jestliže pro každý soubor obvodů $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ polynomiální velikosti je $\sigma_{\mathcal{C},X,Y}(n)$ zanedbatelná funkce.*

$$\delta_{D,X,Y}(n) = |Pr[C_n(X_n) = 1] - Pr[C_n(Y_n) = 1]|$$

Definition 3. *Řekneme, že soubor náhodných veličin $\{X_i\}_{i\in\mathbb{N}}$ je (silně) pseudonáhodný, jestliže je (silně) výpočetně nerozlišitelný od souboru $\{U_n\}_{n\in\mathbb{N}}$ náhodných veličin s uniformním rozdělením.*

Definition 4 (jednosměrná funkce)**.** *Funkce $F : \{0,1\}^* \mapsto \{0,1\}^*$ je jednosměrná, jestliže je spočitatelná v polynomiálním čase a je těžko invertovatelná, tj. pro každý pravděpodobnostní polynomiální algoritmus B je funkce*

$$Pr_{x\sim U_n}[B(f(x)) \in f^{-1} \circ f(x)],\ x \sim U_n$$

zanedbatelná v n.

Zvláštním případem jsou *jednosměrné permutace*, prosté jednosměrné funkce zachovávající délku vstupu.

Definition 5 (Těžký bit)**.** *Zobrazení $b : \{0,1\}^* \mapsto \{0,1\}$ je těžký bit funkce f, jestliže platí:*

- *b je spočitatelné v polynomiálním čase*
- *není odhadnutelné s nezanedbatelnou výhodou, tj. pro každý pravděpodobnostní algoritmus B je funkce*

$$|Pr_{x\sim U_n}[B(f(x)) = b(x)] - \frac{1}{2}|$$

zanedbatelná v n.

Je-li f prostá, pak má těžký bit, právě když je jednosměrná. V opačném případě bychom mohli hodnotu bitu $b(x)$ spočíst invertováním $f(x)$. Pokud by naopak bylo možné s nezanedbatelnou pravděpodobností spočíst libovolný bit x_i, pak bychom mohli vzor spočíst s nezanedbatelnou pravděpodobností i celé x.

Pro neprosté funkce je situace složitější, jelikož požadujeme nalezení právě použitého x, nikoliv jiného se stejným obrazem. Pak tedy například $f(x)$ má dva stejně pravděpodobné vzory x a x' a $b(x) \neq b(x')$, pak je odhad nemožný.

Lemma 1. *Pro x délky n označme $b_x(r)$ orákulum splňující*

$$|Pr_{r\sim U_n}[b_x(r) = \langle x, r\rangle] - \frac{1}{2}| \geq \varepsilon(n).$$

Existuje pravděpodobnostní algoritmus A pracující v čase polynomiálním čase v $(\frac{n}{\varepsilon(n)})$ takový, že pro každé x platí

$$Pr[A^{b_x}(1^n) = x] \geq \frac{\varepsilon(n)^2}{2n}.$$

Theorem 2. *Nechť f je libovolná jednosměrná funkce. Definujme funkci g předpisem*

$$g(x, r) := (f(x), r), |x| = |r|.$$

Pak bodový součin vektorů x, r modulo 2 je těžkým bitem funkce g.

Proof. Nechť $\langle x, r\rangle$ není těžký bit funkce g a označme G algoritmus, který to dokazuje a označme γ jeho výhodu:

$$\gamma(n) = |Pr_{x,r\sim U_n}[G(f(x), r) = \langle x, r\rangle] - \frac{1}{2}|.$$

Budiž S_n množina takových x délky n, na kterých je výhoda G alespoň $\gamma(n)/2$. Tato množina má velikost alespoň $\frac{\gamma(n)}{2}2^n$, jinak by výhoda G byla menší než

$$\frac{1}{2^n}\left(\Sigma_{x\in S_n} 1 + \Sigma_{x\notin S_n}\frac{\gamma(n)}{2}\right) < \frac{\gamma(n)}{2} + \frac{\gamma(n)}{2} = \gamma(n)$$

Dle Lemmatu 1 lze pro každé $x \in S_n$ v čase, který je polynomiální v $n/\gamma(n)$ a s pravděpodobností alespoň $\text{poly}(\gamma(n)/n)$ najít x s pomocí $f(x)$ ($b_x(r) = G(f(x), r)$). Protože $x \sim U_n$ leží v S_n s pravděpodobností alespoň $\gamma(n)/2$ a $\gamma(n)$ není zanedbatelná, dostáváme spor s jednosměrností f. □

Definition 6 (Závazek)**.** *Polynomiální pravděpodobnostní algoritmus Z je bitový závazek, jestliže pro každé $m \in \{0, 1\}$ a $x, x' \in \{0, 1\}^*$ platí:*

- *(závaznost)* $Z(x, m) \neq Z(x', 1-m)$
- *(tajnost) Soubory náhodných veličin* $\{Z_{x \sim U_n}(x, 0)\}$ *a* $\{Z_{x \sim U_n}(x, 1)\}$ *jsou výpočetně nerozlišitelné.*

Buď f jednosměrná funkce a b její těžký bit, pak závazkem může být $Z(m, x) := (f(x), m \oplus b(x))$, autorizací k otevření závazku je x.

Definition 7. *(Neuniformně silný) pseudonáhodný generátor je deterministický algoritmus G, který vstup délky n prodlužuje na výstup délky $l(n) > n$ tak, že soubor $\{G(U_n)\}_{n \in N}$ je (silně) pseudonáhodný.*

Theorem 3. *Nechť je f jednosměrná permutace a b její těžký bit. Pak je zobrazení $G : x \mapsto f(x) \parallel b(x)$ pseudonáhodný generátor.*

Proof. Chceme rozlišit $f(x) \parallel b(x)$ od náhodné posloupnosti, tzn. chceme uhodnout $b(x)$ z $f(x)$.

Nechť G není pseudonáhodný generátor a D je algoritmus, který to dokazuje, tj. $|Pr[D(f(x)b(x)) = 1] - Pr[D(y') = 1]| = \varepsilon(n)$, ε nezanedbatelná funkce. Toto lze přepsat do formy

$$|Pr[D(y \parallel b \circ f^{-1}(y)) = 1] - Pr[D(y \parallel \sigma) = 1]| = \varepsilon(n).$$

Protože s pravděpodobností 1/2 platí $\sigma = b \circ f^{-1}(y)$, pak také

$$Pr[D(y \parallel \sigma) = 1] = \frac{1}{2}Pr[D(y \parallel b \circ f^{-1}y) = 1] + \frac{1}{2}Pr[D(y \parallel \overline{b \circ f^{-1}y}) = 1]$$

$$|Pr[D(y \parallel b \circ f^{-1}y) = 1] - Pr[D(y \parallel \overline{b \circ f^{-1}y}) = 1]| = 2\varepsilon(n)$$

Rozlišovač $A(y) := \sigma$, jestliže $D(y \parallel \sigma) = 1$, $A(y) := \overline{\sigma}$ v opačném případě. Úspěšnost A je:

$$Pr[A(y) = b \circ f^{-1}(y)] =$$

$$= Pr[D(y \parallel \sigma) = 1 \wedge \sigma = b \circ f^{-1}(y)] + Pr[D(y \parallel \sigma) = 0 \wedge \overline{\sigma} = b \circ f^{-1}(y)]$$

$$\frac{1}{2}Pr[D(y \parallel b \circ f^{-1}(y)) = 1] + \frac{1}{2}Pr[D(y \parallel \overline{b \circ f^{-1}(y)}) = 0]$$

$$\frac{1}{2}Pr[D(y \parallel b \circ f^{-1}(y)) = 1] + \frac{1}{2}(1 - Pr[D(y \parallel \overline{b \circ f^{-1}(y)}) = 1])$$

$$\frac{1}{2} + \frac{1}{2}(Pr[D(y \parallel b \circ f^{-1}(y)) = 1] - Pr[D(y \parallel \overline{b \circ f^{-1}(y)}) = 1]) = \frac{1}{2} \pm \varepsilon(n)$$

Výhoda A, tj. $|Pr[A(y) = b \circ f^{-1}(y)] - \frac{1}{2}|$, není zanedbatelná funkce, což je spor s předpokladem, že b je těžký bit. □

Theorem 4. *Nechť G je pseudonáhodný generátor s prodlužovací funkcí $l(n) = n + 1$ a nechť l' je libovolný polynom. Definujme zobrazení G' předpisem*

$$G'(s) = \sigma_1(s) \parallel \cdots \parallel \sigma_{l'(|x|)}(s),$$

kde

$$x_0 = s,\ G(x_{i-1}) = x_i \parallel \sigma_i(s),\ i = 1, 2, \ldots, l'(|x|).$$

Pak je G' pseudonáhodný generátor.

Proof. Buď $X_1, \dots, X_{l'(n)}$ nezávislé kopie náhodné proměnné U_1. Budeme chtít ukázat, že $(X_1, \dots)$ neumíme rozlišit od $(\sigma_1(s), \dots)$ s nezanedbatelnou výhodou. Zdůrazněme, že $\sigma_i(s)$ vycházejí ze stejné volby s a jsou tedy závislé. Zadefinujme si hybridní distribuci:

$$H^{(i)} = (X_1, \dots, X_i, \sigma_1(s), \dots, \sigma_{l'(n)-i}(s))$$

Algoritmus D, který rozlišuje $H^{(0)}$ od $H^{l'(n)}$ s nezanedbatelnou výhodou, rozlišuje taktéž s nezanedbatelnou výhodou nějaké $H^{(i)}$ od $H^{(i+1)}$. Předpokládejme, že jsme takové i zvolili ($Pr = 1/l'(n)$). Pro dané $y \parallel \sigma$ necháme rozhodnout

$$(X_1, \dots, X_i, \sigma, \sigma_1(y), \dots, \sigma_{l'(n)-i-1}(y))$$

Je-li $y \parallel \sigma \sim U_{n+1}$, jde o $H^{(i+1)}$. Je-li $y \parallel \sigma = G(s)$ pro $s \sim U_n$, jde o $H^{(i)}$, protože $\sigma = \sigma_1(s)$ a $\sigma_i(y) = \sigma_{i+1}(s)$. $\square$

Interactive proofs

Definition 8 (ITM)**.** *Interaktivní Turingův stroj je vícepáskový pravděpodobnostní TM se vstupem a výstupem, který kromě vstupní pásky (veřejná), výstupní pásky a pracovní pásky obsahuje ještě*

- *dodatečnou vstupní pásku (soukromá)*
- *vstupní komunikační pásku (read-only)*
- *výstupní komunikační pásku (write-only)*
- *stavový bit (1 políčko s 1/0)*

Stroji je přiřazena identita jedna nebo 0. Program pak obsahuje instrukce pouze pro případ, že je stavový bit roven identitě stroje. V takovém případě stroj pracuje, jinak je nečinný.

Definition 9. *Interaktivní systém je dvojice ITM (A, B), které*

- *mají opačnou identitu,*
- *sdílí veřejnou vstupní pásku,*
- *sdílejí stavový bit,*
- *vstupní komunikační páska A je výstupní komunikační páskou B a naopak.*

Konvence: A dokazovatel, B ověřovatel; tj. výstupem interaktivního výpočtu je pouze výstup B a složitost se bere v potaz pouze u ověřovatele.

Definition 10 (IP)**.** $L \in IP \Leftrightarrow \exists$ *interaktivní systém (A, B) t.ž.*

- *(efficiency) poly-time*
- *(completeness) (A, B) přijme $x \in L$ s pr. alespoň $2/3$*
- *(soundness) $x \notin L$, pak pro lib. ITM A^* je pravděpodobnost, že (A^*, B) přijme x, menší než $1/3$.*

Theorem 5. *1.* $BPP \subseteq IP$

2. $NP \subseteq IP$

Proof. 1: $L \in BPP$ a B ho rozhoduje ve smyslu BPP. Pak $(\emptyset, B)$ rozhoduje L ve smyslu IP – výpočet proběhne v jedné fázi.

2: $L \in NP$, A sdělí slovo y (svědek x) B. B ověří, že y je ve svědecké relaci s x (ověření v poly-time, úplnost daná existencí svědka, spolehlivost z neexistence svědka pro $x \notin L$) □

Definition 11 (Grafový neisomorfismus). *GraphNI*=$\{(G_1, G_2)|G_1 \ncong G_2\}$

Grafový neizomorfismus je zřejmě v $PSPACE$ (mohu in-place vyzkoušet všechna řešení).

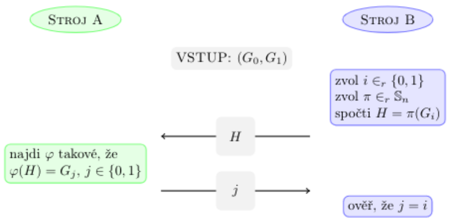
Claim 6. $GraphNI \in IP$

Proof. Uvažme dvě kola uvedené komunikace. Efektivita je zřejmá. Pokud $(G_0, G_1) \in GraphNI$, pokud $\varphi(H) = G_j$, pak platí $i = j$ – B přijme vstup s pravděpodobností 1 (úplnost).

$(G_0, G_1) \notin GraphNI$, tj. grafy jsou izomorfní. Definujme multiset

$$M_i := \{\pi(G_i)|\pi \in S_n\}, \ i = 0, 1.$$

Zřejmě v tomto případě $M_0 = M_1$, tj. A odpoví nezávisle na volbě i. Z uniformity volby i je tedy pr. přijetí vstupu $(\frac{1}{2})^2$, čímž je dokázána spolehlivost.



$\square$

Theorem 7 (Shamir). $IP = PSPACE$

Definition 12. *Interaktivní důkazový systém (A, B) se nazývá důkaz s nulovou znalostí (zero-knowledge proof) pro jazyk L, pokud rozhoduje jazyk ve L ve smyslu Definice 9 a navíc pro každý interaktivní stroj B^* existuje poly-time PTM M takový, že pro každé $x \in L$ platí:*

- $Pr[M(x) = \perp] \leq 1/2$ *(výpočet M selhal)*
- $M(x) \neq \perp \Rightarrow M(x) \sim \sigma_F(A, B^*)(x)$ *pro* $\sigma_F(A, B^*)(x)$ *závěrečný snímek* $B*$ *po výpočtu* (A, B^*) *na* x.

Symbol $\sim$ odpovídá jednomu ze stupňů blízkosti ($\{M(x)\}_{x\in L}$ i $\{\sigma_F(A, B^*)(x)\}_{x\in L}$ soubory náhodných veličin):

- $\sim$ *rovnost souborů – důkaz s dokonale nulovou znalostí (třída PZK),*
- $\sim$ *statistická blízkost – důkaz s téměř dokonalou nulovou znalosti (třída SZK),*
- $\sim$ *výpočetní nerozlišitelnost – důkaz s výpočetně nulovou znalosti (třída CZK).*

Poznamenejme, že M neumí rozhodovat jazyk – simulátor je úspěšný jen pro $x \in L$, pro $x \notin L$ se může chovat libovolně.

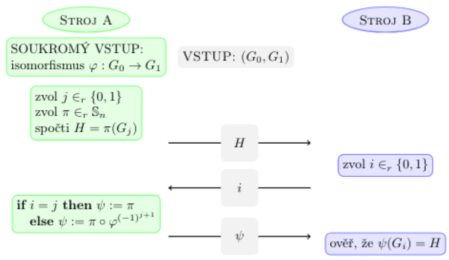
Claim 8. $GraphNI \in PZK$

Proof. Popíšeme simulátor M ověřovatele B^*. Stroj M se může chovat jako B, jen si sám musí generovat zprávy od A. Pokud $\{G_0, G_1\} \in GraphNI$ je simulace A snadná – A vždy pošle j, které je rovnou i, což je hodnota stroji M známá z předchozí simulace. $\square$

Definition 13 (Grafový isomorfismus)**.** *GraphISO*=$\{(G_1, G_2) | G_1 \cong G_2\}$

Claim 9. $GraphISO \in PZK$

Proof. Důkaz spočívá v dvojím opakování popsané komunikace. Stroj B přijme, jestliže v obou kolech $\phi(G_i) = H$. Systém je zřejmě efektivní a úplný.



Předpokládejme nyní, že $(G_0, G_1) \notin GraphISO$, pak ale zpráva H je, bez ohledu na postup A^*, isomorfní nejvýše jednomu z grafů. S pravděpodobností $1/2$ B zvolí i, pro které neexistuje ψ splňující $\phi(G_i) = H$. Ve dvou kolech tedy odmítne s pravděpodobností $3/4$ – spolehlivost.

Nyní je třeba dokázat nulovou znalost. M může simulovat celý výpočet s výjimkou situace $\psi = \pi \circ \varphi^{-1}$, tj. $i \neq j$. S pr. $1/2$ (pr. volby $i \neq j$) simulace selže hned v prvním kole. Při dvou kolech je tato pravděpodobnost $1/4$ – stačí tedy tři opakování pokusu o simulaci, aby se pravděpodobnost dostala nad $1/2$. M tedy opakuje celý postup třikrát a vytiskne $\perp$, pokud všechny pokusy selhaly. Jinak vytiskne výstupní snímek B^* po úspěšném pokusu. $\square$

Computer algebra

Největší společný dělitel

Data: $f,\ g$
Result: $\mathrm{NSD}(f,g)$
$a_0 := f,\ a_1 := g,\ i := 1;$
while $a_i \neq 0$ **do**
  $a_{i+1} := a_{i-1} \bmod a_1;$
  $i{+}{+};$
end
return a_{i-1}

Algorithm 1: Eukleidův algoritmus

Eukleidův algoritmus má dvě nevýhody – koeficienty obvykle vycházejí velké a navíc je aritmetika nad podílovým tělesem pomalá. Naivním přístupem dostáváme sice složitost $O(mc(f)mc(g))$, nicméně reálně v Eukledioвě algoritmu dojde k exponenciálnímu nárůstu koeficientů v průběhu výpočtu. Proto musíme k výpočtu NSD přistupovat obezřetněji.

Definition 14. *Buď* $f = \sum_{i=0}^{n} a_i x^i \in \mathcal{R}[x]$ *polynom. Pak definujeme*

- *obsah* $cont(f) = NSD(a_0, a_1, \ldots, a_n)$,
- *primitivní část* $pp(f) = \sum_{i=0}^{n} \frac{a_i}{cont(f)} x^i$.

Polynom f *nazveme primitivní, jestliže* $cont(f) = 1$.

Data: $f,\ g$
Result: $\mathrm{NSD}(f,g)$
$a := \mathrm{NSD}_{\mathcal{R}}(\mathrm{cont}(f), \mathrm{cont}(g));$
$b := \mathrm{NSD}_{\mathcal{R}[x]}(\mathrm{pp}(f), \mathrm{pp}(g));$
return $a.b$

Algorithm 2: Generický algoritmus pro výpočet NSD

Definition 15 (Pseudodělení)**.**
$f\ pdiv\ g = lc(g)^{degf - defg + 1} f\ div\ g$
$f\ pmod\ g = lc(g)^{degf - defg + 1} f\ mod\ g$

Definition 16 (Podobnost polynomů)**.** *Řekneme, že polynomy* $f, g \in \mathcal{R}$ *jsou podobné* ($f \sim g$)*, jestliže* $\exists k, l \in \mathcal{R} : kf = lg$.

Definition 17. *Posloupností polynomiálních zbytků (PRS) rozumíme každou posloupnost* $f_1, f_2, \ldots, f_k \in \mathbb{R}[x]$ *splňující*

1. $f_1, \ldots, f_{k-1} \neq 0,\ f_k = 0$
2. $degf_i \geq degf_{i-1}$

3. $f_{i+1} \sim (f_{i-1} \text{ pmod } f_i)$

Claim 10. *Buď $\mathcal{R}$ gaussovský obor a $f_1, \ldots, f_k$ nějaká PRS v $\mathcal{R}[x]$. Pak $NSD(f_1, f_2) \sim f_{k-1}$.*

Proof. $\mathcal{Q}$ podílové těleso $\mathcal{R}$, z třetí vlastnosti PRS $f_{i+1} \parallel (f_{i-1} \text{ pmod } f_i) \parallel (f_{i-1} \bmod f_i)$, tj. PRS simuluje, až na podobnost, chod Eukleidova algoritmu v $\mathcal{Q}[x]$, a tedy $f_{k-1} \sim \mathrm{NSD}_{\mathcal{Q}[x]}(f_1, f_2)$. □

Data: f, g primitivní, def$f \geq$deg$g$
Result: NSD(f, g)
$f_1 := f, \; f_2 := g, \; i := 2;$
while $f_i \neq 0$ **do**
  $f_{i+1} = \frac{1}{\alpha_i + 1}(f_{i-1} \text{ pmod } f_i);$
  $i{+}{+};$
end
return $pp(f_{i-1})$

Algorithm 3: NSD pomocí PRS

Nechť $\delta_i = \deg f_i - \deg f_{i+1}$, pak máme následující možnosti voleb α_i:

Eukleidovská PRS	$\alpha_{i+1} = 1$
Primitivní PRS	$\alpha_{i+1} = \mathrm{cont}(f_{i-1} \text{ pmod } f_i)$
Redukovaná PRS	$\alpha_3 = 1, \; \alpha_{i+1} = \mathrm{lc}(f_{i-1})^{\delta_{i-2}+1}, \; i \geq 3$
Subrezultantová PRS	$\alpha_3 = 1, \; \alpha_{i+1} = \mathrm{lc}(f_{i-1})\beta^{\delta_{i-1}^{i-1}}, \; i \geq 3,$ kde $\beta_2 = \mathrm{lc}(f_2)^{\delta_1}, \; \beta_{i+1} = lc(f_{i+1})^{\delta_i}\beta_i^{1-\delta_i}$.

V praxi je Eukleidovská PRS ještě horší, než-li Eukleidův algoritmus, ovšem primitivní a redukovaná PRS se již dostávají do polynomiální složitosti (přesněji $O(n^4)$, $n = \deg f$), pro náhodné polynomy vychází redukovaná varianta cca o 20% lépe.

Theorem 11. *Buď $\mathcal{R}$ gaussovský obor, $\mathcal{Q}$ jeho podílové těleso, $\overline{\mathcal{Q}}$ jeho algebraický uzávěr a f, g dva nekonstantní polynomy z $\mathcal{R}[x]$. Pak NTJE:*

1. *Polynomy f, g jsou soudělné v $\mathcal{Q}[x]$*
2. *$\deg NSD_{\mathcal{R}[x]}(f, g) > 0$*
3. *$\exists u, v \in \mathcal{R}[x]$, $\deg u < \deg f, \deg v < \deg g : uf + vg = 0$*
4. *f, g mají společný kořen v $\overline{\mathcal{Q}}$.*

Proof. $1 \Leftarrow 2$ z Gaussova lemmatu (dělitelnost v $\mathcal{Q}[x]$ implikuje dělitelnost v $\mathcal{R}[x]$)

$1 \Leftrightarrow 4$ zřejmé

$1 \Rightarrow 3$ $d = NSD(f, g), \; u := g/d, \; v = f/d$

$3 \Rightarrow 1$ $uf = gv \Rightarrow f|v \vee f|g \Rightarrow f|g$ $(\deg v < \deg f)$ □

Definition 18 (Sylvesterova matice)**.** *Buď $f = \sum_{i=0}^{n} a_i x^i, \; g = \sum_{i=0}^{m} b_i x^i$ jsou dva nekonstatní polynomy z $\mathcal{R}[x]$. Sylvesterovou matici $M(f, g)$ rozumíme čtvercovou matici velikosti $m + n$ nad $\mathcal{R}$:*

$$M(f,g) = \begin{pmatrix} a_n & 0 & \dots & 0 & b_m & 0 & \dots & 0 \\ a_{n-1} & a_n & & \vdots & b_{m-1} & b_m & & \vdots \\ \vdots & a_{n-1} & & 0 & \vdots & b_{m-1} & & 0 \\ a_0 & \vdots & & a_n & b_0 & \vdots & & b_m \\ 0 & a_0 & & a_{n-1} & 0 & b_0 & & b_{m-1} \\ \vdots & 0 & & \vdots & \vdots & 0 & & \vdots \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & a_0 & 0 & 0 & \dots & b_0 \end{pmatrix}$$

Determinant této matice nazvěme rezultant polynomů f, g ($res(f,g)$).

Theorem 12 (Sylvesterovo kritérium)**.** *Buď $\mathcal{R}$ gaussovský, $\mathcal{Q}$ jeho podílové těleso a $f, g \in \mathcal{R}[x]$ dva nekonstatní polynomy. Pak platí:*

$$(res)(f,g) = 0 \Leftrightarrow f,\ g \text{ jsou soudělné v } \mathcal{Q}[x].$$

Proof. Řešení M(f,g) odpovídá 3. bodu Věty 11 (po roznásobení dostaneme konvoluci). □

Z Cramérova pravidla (a rozvoje podle posledního sloupce) jsme dokonce schopni dokázat, že

$$\exists u, v \in \mathcal{R}[x], \deg u < \deg f, \deg v < \deg g : uf + vg = \mathrm{res}(f,g).$$

Tuto úlohu je též možno řešit nad konečnými tělesy. Buďto tak, že budeme počítat modulo $p \in \mathbb{P} > \mathrm{NSD}_{\mathbb{Z}}(lc(f), lc(g)).\mathrm{LM}(f,g)$ (Landau-Mignottova mez). Tato mez je ovšem značně neoptimální a proto by bylo vhodné seskládat NSD přes CRT (stačí aby součin prvočísel překročil tuto mez). Komplikací je, že potřebujeme jen *použitelná* prvočísla ($p \nmid \mathrm{NSD}_{\mathbb{Z}}(lc(f), lc(g))$) a šťastná (stupeň NSD v celých číslech i konečném tělese je stejný), tzn. kdykoliv nám vzroste stupeň NSD po přidání prvočísla, musíme započíst výpočet znovu.

Faktorizace polynomů nad konečými tělesy

Definition 19 (Bezčtvercový polynom)**.** *$\mathcal{R}$ gaussovský, $f \in \mathcal{R}[x_1, x_2, \dots, x_n]$ bezčtvercový polynom $\Leftrightarrow$ f není násobek čtverce nekonstatního polynomu.*

Definition 20 (Bezčtvercový rozklad)**.** *$\mathcal{R}$ gaussovský, $f \in \mathcal{R}[x_1, x_2, \dots, x_n]$, pak $f = h_1 h_2^2 \dots h_k^k$ je bezčtvercový rozklad $\Leftrightarrow$ $\forall i \neq j : h_i, h_j$ nesoudělná a h_i bezčtvercová.*

Lemma 13. *Buď $\mathbb{F}_q$ konečné těleso charakteristiky p, $f \in \mathbb{F}_q[x]$ t.ž. $f' = 0$. Pak $\exists g \in \mathbb{F}_q[x] : f = g^p$.*

Theorem 14. *$f \in \mathcal{F}_{p^n}[x]$ primitivní polynom*

1. *f bezčtvercový $\Leftrightarrow$ NSD$(f, f') = 1$*
2. *$f = \Pi_1^n h_i^i$ bezčtvercový rozklad f, pak*

$$\text{NSD}(f, f') = \Pi_{i=2, p\nmid i} h_i^{i+1} \Pi_{i=1, p|i} h_i^i.$$

Proof. 1 $\Leftarrow$: $f = g^2h \Rightarrow f' = 2gg'h + g^2h' \Rightarrow g|\text{NSD}(f, f')$.

1 $\Rightarrow$: $\exists$ irreducibilní $\pi \in \mathbb{F}_{p^n}[x] : \pi|f \wedge \pi|f'$, deg$\pi > 1$ (f primitivní). Takže platí $f = \pi g \Rightarrow f' = \pi' g + \pi g \Rightarrow \pi|\pi'$ (spor s lemmatem) nebo $\pi|f'$ (spor s bezčtvercovostí).

2 : $f' = \sum_{j,p\nmid j} h_j^{j-1} j\Pi_{i\neq j} h_i^i h_j'$, $g := \Pi_{p\nmid i} h_i^{i-1}\Pi_{p|j} h_j^j|\text{NSD}(f, f')$. Potřebujeme ověřit, že $\text{NSD}(\sum_{p\nmid j} h_j' j\Pi_{i\neq j,p\nmid i} h_i, \Pi h_i)$ je nesoudělné s $\text{NSD}(f, f')$. Pro spor nechť existuje $\pi \in \mathbb{F}_{p^n}[x]$ ireducibilní, které dělí oba polynomy, tj. $\exists! i : \pi|h_i$ a z druhé strany $\pi|h_i'$, což je dle 1 spor s bezčtvercovostí h_i. □

Data: $f \in \mathbb{F}_q[x]$ primitivní, nekonstantní
Result: bezčtvercová faktorizace $f = \Pi h_i^i$
if $f' = 0$ **then**
| goto 3;
end
$f_1 =$NSD(f, f'); $g_1 = f/f_1$; $j = 1$;
while $deg\, g_j > 0$ **do**
| $g_{i+1} =$NSD(g_j, f_j);
| $f_{j+1} = f_j/g_{j+1}$;
| $h_j = g_j/g_{j+1}$;
| $j{+}{+}$;
| $f = f_j$
end
finish:
if $deg f \neq 0$ **then**
| $h_p, \ldots, h_{l(p)}$ =recur$(\sqrt[p]{f})$
end
return $\frac{u}{\Pi h_i^i} h_1, h_2, \ldots, h_{j-1}$

Algorithm 4: bezčtvercová faktorizace $O(\deg(f)^3 l^2(q))$

Claim 15. *$f \in \mathbb{F}_q[x]$ monický, bezčtvercový, nekonstantní; $h \in \mathbb{F}_q[x]$ nekonstantní t.ž. $h^q \equiv_f h$. Pak $f = \Pi_{a\in\mathbb{F}_q} NSD(f, h - a) := A$.*

Proof. $a \neq b \Rightarrow$NSD$(h - a, h - b) = 1$, $f = \Pi g_i$ ireducibilní rozklad (g_i BÚNO monické).

$x^q - x = \Pi(x - a) \Rightarrow \Pi(h - a) = h^q - h \equiv_f 0$ (dosazovací homomorfismus) $\Rightarrow f|\Pi(h - a), \forall g_i \exists! j : g_i|h - a_i$. □

Claim 16. *$Q \in \mathbb{F}_q^{n\times n}$ jejíž sloupce tvoří koef. polynomů*

$$1, x^q \ mod\ f, x^{2q} \ mod\ f, \ldots, x^{(n-1)q} \ mod\ f.$$

Pak $\sum a_i x^i \in \mathbb{F}_q[x] \in \{h | h^q - h \equiv_f 0, deg\, h < deg\, f\} \Leftrightarrow Qa = a$.

Proof. $h^q = \sum a_i x^{qi} \Rightarrow h^q \bmod f = \sum(a_i x^{qi} \bmod f) = \sum a_i \sum q_{ji} x^j = \sum\sum q_{ji} x^j a_i$. Hledám taková a_i, že tato suma dá $\sum a_i x^i$, tj. po dosazení do matice se úloha přeloží do hledání vlastního vektoru (první sloupec obsahuje jen jednu 1, takže pro vl. číslo 1).

Dimenze Ker$(Q - I_n)$ pak dokonce odpovídá počtu faktorů f. □

Data: $f \in \mathbb{F}_q[x]$ bezčtvercový, monický, nekonstantní
Result: $g_1, \dots, g_m$ ireducibilní t.ž. $f = \Pi g_i$
Sestav matici Q;
Najdi bázi $\mathrm{Ker}(Q - I_n) \Rightarrow 1 = h_1, \dots, h_m$;
$i = 2; F = \{f\}$;
while $|F| < m$ **do**
  foreach t *in* F **do**
    $t = \Pi_{a \in \mathbb{F}_q} \mathrm{NSD}(t, h_i - a)$ rozklad $t \Rightarrow$ nahraď t v F;
  end
  i++;
end
return F

Algorithm 5: Belekampův algoritmus $O(\deg(f)^3 l^2(q) q)$

Gröbnerovy báze

Definition 21 (Monom)**.** *Prvek* $\Pi x_i^{e_i} \in \mathbb{K}[x_1, \dots, x_n]$, $e_1, \dots, e_n \in \mathbb{N}_0$*, se nazývá monom (monočlenů). Množinu všech monomů označme* $\mathbb{T}^n = \{x^\alpha | \alpha \in \mathbb{N}_0^n\}$.

Definition 22. *Přípustné uspořádání* $\mathbb{T}^n$ *je lineární uspořádání* $<$ *splňující:*

1. $1 < x^\alpha,\ \alpha \in \mathbb{N}_0^n \backslash \{0\}$,
2. $\forall x^\alpha, x^\beta, x^\gamma : x^\alpha < x^\beta \Rightarrow x^\alpha x^\gamma < x^\beta x^\gamma$.

- $<_{lex}$: lexikografické uspořádání
- $<_{deglex}$: $x^\alpha <_{deglex} x^\beta \Rightarrow \sum a_i < \sum b_i \vee (\sum a_i = \sum b_i \wedge x^\alpha <_{lex} x^\beta)$
- $<_{degrevlex}, <_{DRL}$: $x^\alpha <_{DRL} x^\beta \Rightarrow$
  $\sum a_i < \sum b_i \vee (\sum a_i = \sum b_i \wedge \exists j \forall i > j : a_j > b_j \wedge a_i = b_i)$.

$<_{DRL}$ je velmi efektivní při výpočtu Gröbnerových bazí. Term v polynomu f, který je vzhledem ke zvolenému uspořádání největší značíme $lt(f)$.

Definition 23. *Nechť* $f, g, h \in \mathbb{K}[x_1, \dots, x_n]$, $<$ *přípustné uspořádání* $\mathbb{T}^n$*,* f *se redukuje na* h *modulo* g *v jednom kroku*

$$(f \to^g h) \Leftrightarrow \exists term\ X\ \ v\ f\ a\ lt(g)|X,\ \ h = f - \frac{X}{lt(g)} g.$$

Analogicky pro $\{f_1, f_2, \dots, f_z\} = F \subset \mathbb{K}[x_1, \dots, x_n] \backslash \{0\}$

$$(f \to_+^F h) \Leftrightarrow f \to^{f_{i_1}} h_{i_1} \to^{f_{i_2}} \cdots \to^{f_{i_s}} h.$$

Definition 24. $f_1, f_2, \dots, f_s, r \in \mathbb{K}[x_1, \dots, x_n] \backslash \{0\}$, $<$ *přípustné uspořádání* $\mathbb{T}^n$*. Pak* r *je redukovaný vzhledem k* $\{f_i\} \Leftrightarrow$ *žádný term v* r *není dělitelný prvkem z množiny* $\{lm(f_i)\}$.

Definition 25 (**Gröbnerova báze**)**.** $I \subseteq \mathbb{K}[x_1, \dots, x_n]$ *ideál. Pak množina* $G = \{g_1, \dots, g_t\} \subseteq I \backslash \{0\}$ G *je Gröbnerovou bází ideálu* I*, jestliže platí*

$$\forall 0 \neq f \in I \exists g_i \in G : lm(g_i) | lm(f).$$

Data: $f, f_1, f_2, \ldots, f_s \in \mathbb{K}[x_1, \ldots, x_n] \backslash \{0\}$, $<$ přípustné uspořádání $\mathbb{T}^n$
Result: $u_1, u_2, \ldots, u_s, r \in \mathbb{K}[x_1, \ldots, x_n] \backslash \{0\}$ t.ž. $f = \sum u_i f_i + r$, r red. vzhledem k $\{f_i\}$ a $lm(f) \geq max\{lm(u_i)lm(f_i)\}$
$u_i = 0$; $r = 0$; $h := f$;
while $h \neq 0$ **do**
  if $\exists i : lm(f_i) | lm(h)$ **then**
    $h := h - \frac{lt(h)}{lt(f_i)} f_i$;
    $u_i := u_i + \frac{lt(h)}{lt(f_i)}$;
  end
  else
    $h := h - lt(h)$;
    $r := r + lt(h)$;
  end
end
return $i_1, \ldots, u_s, r$

Algorithm 6: dělení se zbytkem

$Lt(S) := (ln(f) | f \in S) \ldots$ ideál generovaný množinou

Theorem 17. $0 \neq I \subseteq \mathbb{K}[x_1, \ldots, x_n]$ *ideál,* $G = \{g_1, \ldots, g_t\} \subseteq I \backslash \{0\}$ *a* $<$ *přípustné uspořádání* $\mathbb{T}^n$*. Pak NPJE:*

1. *G je Gröbnerova báze*
2. $\forall f \in \mathbb{K}[x_1, \ldots, x_n] : f \in I \Leftrightarrow f \to_+^G 0$
3. $\forall f \in \mathbb{K}[x_1, \ldots, x_n] : f \in I \Leftrightarrow \exists u_1, \ldots, u_t \in \mathbb{K}[x_1, \ldots, x_n] : f = \sum u_i g_i$, $lm(f) = max\{lm(u_i)lm(g_i)\}$
4. $Lt(I) = Lt(G)$

Proof.

$1 \Rightarrow 2$	$\Leftarrow$	platí vždy
	$\Rightarrow$	$f \to_+^G r$ (redukované), $f \in I \Rightarrow f - r \in I \Rightarrow r \in I$
		pokud $r \neq 0$, pak spor s redukovaností r
$2 \Rightarrow 3$	$\Leftarrow$	platí vždy ($G \subseteq I$)
	$\Rightarrow$	Algo 6, kdyby ostrá nerovnost, pak LHS je LK monočlenů menších než $lm(f) \Rightarrow$ spor
$3 \Rightarrow 4$		$Lt(G) \subseteq Lt(I), 0 \in Lt(I) \Rightarrow f = \sum u_i g_i$
		$lm(f) = max\{lm(u_i)lm(g_i)\} \Rightarrow \exists i : lm(f) = lm(u_i)lm(g_i) \in Lt(g)$
$4 \Rightarrow 1$		$lm(f) = \sum lm(g_i)v_i$, v_i jakosoučet termů $\to$ roznásobíme

$\square$

Definition 26. $G \subseteq \mathbb{K}[x_1, \ldots, x_n] \backslash \{0\}$ *konečná, pak G je Gröbnerova báze $\Leftrightarrow$ G je Gröbnerova báze (G).*

Theorem 18. $G \subseteq \mathbb{K}[x_1, \ldots, x_n] \backslash \{0\}$*, pak G je Gröbnerova báze, jestliže*

$$\forall f \in \mathbb{K}[x_1, \ldots, x_n] : (f \to_+^g r_1, f \to_+^G r_2 \Rightarrow r_1 = r_2).$$

Proof. $\Rightarrow$: $r_1 - r_2 \in (G) \Rightarrow$ lze ho zredukovat, ale $lm(r_1 - r_2)$ je obsažené v r_1 nebo $r_2 \Rightarrow$ spor s redukovaností.

$\Leftarrow$: Tvrzení $g \to_+^G r$ redukované $\Rightarrow g - cXg_i \to_+^G r$ pro $X \in \mathbb{T}^n$ a $c \in \mathbb{K}$.

$r = 0 : f = \sum u_i g_i$, u_i LK monočlenů $\to f = \sum c_j X_j g_{i,j}$

$f \to_+^G \Rightarrow f - c_j X_j g_{i,j} \to_+^G r \ \ldots 0 \to_+^G r \Rightarrow r = 0$ □

Definition 27. *Buď* $0 \neq f, g \in \mathbb{K}[x_1, \ldots, x_n], <$ *přípustné uspořádání* $\mathbb{T}^n$, $L =$*NSN*$(lm(f), lm(g))$. *Pak definujeme* S*-polynom* f *a* g*:*

$$S(f,g) = \frac{L}{lt(f)} f - \frac{L}{lt(g)} g$$

Theorem 19 (Buchberger)**.** *Nechť* $G = \{g_1, \ldots, g_s\} \subseteq \mathbb{K}[x_1, \ldots, x_n] \backslash \{0\}$, $<$ *přípustné uspořádání* $\mathbb{T}^n$. *Pak* G *je Gröbnerova báze právě tehdy, když* $\forall i \neq j$: $S(g_i, g_j) \to_+^G 0$.

Proof. $\Rightarrow$: G Gröbnerova báze $(G) \Rightarrow \forall f \in (G) : f \to_+^G 0$

$\forall i, j : S(f_i, f_j) \in (f_i, f_j) \subseteq (G)$.

$\Leftarrow$: dokážeme 3, ze všech vyjádření $f = \sum u_i g_i$ zvolíme t.ž. $max\{lm(u_i)lm(f_i)\}$ bylo nejmenší možné. □

Data: $\mathbb{F} = \{f_1, \ldots, f_s\} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ nenulový, $<$ přípustné uspořádání $\mathbb{T}^n$
Result: Gröbnerova báze (F)
$G := F; \mathcal{G} = \{(f_i, f_j | 1 \leq i \leq j \leq s\}$;
while $\mathcal{G} \neq \emptyset$ **do**
  vyber $(f, g) \in \mathcal{G}$ a odeber je z $\mathcal{G}$;
  h t.ž. $S(f, g) \to_+^G h$;
  if $h \neq 0$ **then**
    $G := G \cup \{h\}$;
    $\mathcal{G} := \mathcal{G} \cup \{(h, u) | u \in \mathcal{G}\}$;
  end
end
return G

Algorithm 7: Buchbergerův algoritmus

Řešení soustav polynomiálních rovnic

Soustava $\ldots p_1 = 0$, $p_2 = 0, \ldots, p_k = 0$; $p_i \in \mathbb{K}[x_1, \ldots, x_n]$. Spočtu Gröbnerovu bázi z ideálu generovaného $\{p_1, p_2, \ldots, p_k\}$. Postupně hledám řešení prvků báze s nejmenším počtem proměnných a postupně propaguji dále.

Factorization

Factorization

Definition 28 (Factorization)**.** *Given a number N, find a number $r \in \mathbb{Z}_N$ such that $r|N$ and r is not a unit.*

```
Data: N composite number
Result: r non-trivial divisor of N
i = 2;
while i ≤ √N do
    if i|N then
        break;
    end
    i++;
end
return i
```

Algorithm 8: Trial division ($O(\sqrt{N})$)

Pollard's rho

- complexity $O(\sqrt{p})$ - p is the smallest factor of N
- used to factor ninth Fermat number ($2^{2^9} + 1$)
- generate a pseudorandom sequence x_i mod N - $x_0, g(x_0), g(g(x_0))....$ (g is most commonly $x^2 + 1$ mod N, or $x^2 - 1$ mod N)
- "rho" shape – M – preperiod, T – period
- the sequence is closely connected to another sequence: x_i mod p
- by birthday paradox, there should be a collision (cycle) in x_i mod p in $O(\sqrt{p})$ steps
- if the collision is in x_i mod p and not in x_i mod N, then a factorization is found
- cycle is then found by Floyd, or Brent cycle finding algorithm
- Floyd's algorithm (tortoise and hare)
- utilizes that $\exists i : x_i = x_{2i}$ ($i = T\lceil \frac{M}{T} \rceil$ if $M > 0$, else $i = T$)
- $i \leq T + M$
- algorithm:
  - compute two sequences: x_i, y_i
  - $x_0 = y_0$
  - $x_{i+1} = g(x_i)$ (tortoise) $y_{i+1} = g(g(y_i))$ (hare)

  - after each step compute $gcd(|x_i - y_i|, N)$ - if $gcd \neq 1, N$ - success, if $gcd = N$ - try again with different settings

- Brent's algorithm
- $l(i) = 0$ if $i = 0$, else $2^{\lfloor log_2(i) \rfloor}$
- utilizes that $\exists i : x_{l(i)-1} = x_i$
- algorithm:
  - compute one sequence: x_i (uses y_i, but no need to compute it)
  - compares x_{2^i-1} with x_j for all $2^i \leq j \leq 2^{i+1} - 1$
  - $x_0 = y_0$
  - $x_{i+1} = g(x_i)$
  - if $i + 1$ is a power of two: $y_{j+1} = x_i$, $x_{i+1} = g(x_i)$
- this way it is too slow (too many iterations)
- $\exists i : y_i = y_{l(i)-1}$ and $3/2l(i) \leq i < 2l(i)$
- smallest such $i = 2^{\lceil log_2(max(M+1,T)) \rceil} + T\lceil \frac{l(M)+1}{T} \rceil - 1$ if $M > 0$ or $T > 1$, else $i = 3$
- now using Brent's algorithm should be about 36% faster than Floyd's (in one step computing only one evaluation of g)
- another speedup - instead of gcd in every step, multiply $x_i - y_i$ together mod N, and then every 100 (or other number) steps compute gcd (gcd the slowest part of the algorithm) (if fail (in 100 steps combines multiple factor - try last 100 steps properly))

Pollard's p-1

Definition 29 (power/smooth)**.**
$B \in \mathbb{N}$, $n \in \mathbb{N}$ *is B-smooth* $\Leftrightarrow \forall p \in \mathbb{P} : p|n \Rightarrow p \leq B$
$B \in \mathbb{N}$, $n \in \mathbb{N}$ *is B-powersmooth* $\Leftrightarrow \forall p \in \mathbb{P} \forall a \in \mathbb{N} : p^a|n \Rightarrow p \leq B$

$$\text{Largest } B\text{-powersmooth number } e_B := \mathrm{NSN}(1, \ldots, B)] = \prod_{p \leq B; p \in \mathbb{P}} p^{\lfloor \log_p B \rfloor}.$$

If N composite, $p|N$ such that $p - 1$ is B-powersmooth then

$$\forall a \in \mathbb{Z} : NSD(a, p) = 1 \Rightarrow a^{p-1} \equiv_p 1 \Rightarrow a^{e_B} \equiv_p 1$$

and thus $\mathrm{GCD}(a^{e_B} - 1, N) > 1$.

N is composite, $p|N$ and $p = c.p_1$ for c B-powersmooth and $p_1 \in \mathbb{P}, B_1 \leq p_1 \leq B_2$.

$$a \in \mathbb{Z}_N : a^{e_{B_1}} \bmod N =: b; GCD(b - 1, N) = 1,$$

$$\forall q \in \mathbb{P} \cap \{B_1 + 1, \ldots, B_2\} : GCD(b^q - 1, N) =?$$

We'll exploit the fact that differences between two consecutive primes are rather small.

Data: N composite, factor base $p_1, \ldots, p_n \leq B$
Result: non-trivial divisor or failure
$a \in \mathbb{Z}_N \backslash \{0, 1\}$; $//a = 2$
$d = NSD(a, N)$;
if $d > 1$ **then**
  return d;
end
for $i = 1$ *to* k **do**
  $e = \lfloor \log_{p_i} B \rfloor$;
  $a = a^{p_i^e} \bmod N$;
  $d = GCD(a - 1, N)$;
  if $d > 1$ **then**
    return d;
  end
end
if $d = 1$ *or* $d = N$ **then**
  return fail;
end
else
  return d;
end

Algorithm 9: Pollard $(p-1)$ $(O(B\log B\log^2 N))$

Data: N composite, factor base $p_1, \ldots, p_n \leq B$,
$q_1, \ldots, q_l \in \{B_1 + 1, \ldots, B_2\}$, $d_i = q_{i+1} - q_i$
Result: non-trivial divisor or failure
$a \in \mathbb{Z}_N \backslash \{0, 1\}$;
$b = a^{e_{B_1}} \bmod N$;
if $GCD(b - 1, N) > 1$ **then**
  goto label;
end
for d_i **do**
  $b_d = b^{q_i}$;
end
$b = b^{q_1} \bmod N$;
for $i = 1$ *to* $l - 1$ **do**
  if $GCD(b - 1, N) > 1$ **then**
    break;
  end
  $b = b.b_{d_i}$;
end
label: **if** $GCD(n - 1, N) \neq 1, N$ **then**
  return $GCD(b - 1, N)$
end
else
  return fail;
end

Algorithm 10: 2-phase Pollard $(p-1)$

ECM

- similar to $p-1$ method – groups $\mathbb{Z}_p^*$ is substituted by elliptic curve group
- elliptic curve: $y^2 = x^3 + ax + b$ – Weierstrass form (field characteristic $\neq 2, 3$ – requires $gcd(6, N) = 1$)
- addition of $P, Q = R$ on curve – third point intersecting curve on line given by P, Q is reflected around x axis (or point at infinity – $y = \infty$)
- formulas(nontrivial):
  - $P \neq Q$: $s = \frac{y_P - y_Q}{x_P - x_Q}$, $x_R = s^2 - x_P - x_Q$, $y_R = s(x_P - x_R) - y_P$
  - x_R can be deduced by putting secant line and elliptic curve into the same equation and Vieta's formula, y_R is computed from line definition
  - $P = Q$: $s = \frac{3x_P^2 + a}{2y_P}$, $x_R = s^2 - 2x_P$, $y_R = s(x_P - x_R) - y_P$
  - formula can be deduced by taking the homogeneous polynomial of elliptic curve, and doing implicit derivation
- on an "elliptic curve modulo N" we try to compute $e_B P$ for some point P if we fail – we could not find an inverse mod N – we have a factor
- useful for finding "small" ($\leq 10^{30}$) factors
- starting point
  - choose random curve, choose y_P, compute x_P as a square root mod N X as hard as factorization
  - $b = 1$, $P = [0, 1]$
  - random point on random curve – choose x_P, y_P, a randomly, compute b
- speedup – parallel computation on multiple curves
  - need curves $y^2 = x^3 + a_i x + 1$ (different curves, different a_i) $P_i = [0, 1]$
  - prime power by prime power we compute $e_B P$ in a parallel way
  - speedup – parallel inverse mod N – 1 gcd, 3(number of curves) multiplications mod N
  - input d_i ($1 \leq i \leq M$) – values to be inverted
  - $c_i = \prod_{j \leq i} d_j$
  - find u, v: $uc_M + vN = 1$
  - for all i: $b_i = uc_{i-1}$, $u = ud_i$ (b_i are inverses)
  - if algorithm fails, we found factor
- complexity $L_p[1/2, \sqrt{2}]$

Dixon's factorization

- basis for the following two algorithms
- finding relations $(x, y) \in \mathbb{Z}$: $x^2 \equiv y \bmod N$
- try to find relations such that $\prod y_i = y^2 \in \mathbb{Z}$
- using $x^2 = x_1^2 x_2^2 ... x_n^2 \equiv y^2 \bmod N$ try to factor N $((x+y)(x-y)$ – hopefully get nontrivial decomposition)
- choose a factor base $\{-1, 2, ...p_{max}\}$ of size $|B|$, generate relations (x, y), and try to factor y in factor base – keep smooth relations (factorization successful)
- repeat until $< |B|$ $(= t)$ smooth relations
- generate a $t \times |B|$ matrix of exponents mod 2
- solve it – find a linear combinations of rows that gives zero – product of such y_i is a square

CFRAC

- uses continuous fractions expansion of $\sqrt{N}$ to generate relations
- continued fraction
  - generic continued fraction $[x_0, x_1 ... x_n] \in \mathbb{Q}(x_0, ...x_n)$
  - definition: P_n, $Q_n \in \mathbb{Z}[x_1, ..x_n]$ polynomial coefficients from $\mathbb{Z}_2$, $P_{-2} = 0, P_{-1} = 0, Q_{-2} = 1, Q_{-1} = 0, P_n = x_n P_{n-1} + P_{n-2}, Q_n = x_n Q_{n-1} + Q_{n-2}$
  - $[x_0, ...x_n] = \frac{P_n}{Q_n}$ for $n \geq 0$
  - proof of the above (idea) – rewriting $x_n = x_n + 1/x_{n+1}$ and basic rewriting of expressions
  - Lemma: $\forall n \geq -1$ $P_n Q_{n-1} - P_{n-1} Q_n = (-1)^{n+1}$
  - proof: by induction $n = -1$ easy, then $\begin{pmatrix} x_0 & 1 \\ 1 & 0 \end{pmatrix} ... \begin{pmatrix} x_n & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} P_n & P_{n-1} \\ Q_n & Q_{n-1} \end{pmatrix}$, and taking determinants completes the proof
  - Lemma: define $p_n = P_n(a_0, ...a_n), q_n = Q_n(a_0, ...a_n)$ – values of P_n, Q_n at $(a_0 ... a_n)$, then sequence $(\frac{p_n}{q_n})$ converges
  - proof: $p_n q_{n-1} - p_{n-1} q_n = (-1)n + 1$ is equivalent to $\frac{p_n}{q_n} - \frac{p_{n-1}}{q_{n-1}} = (-1)^{n+1} \frac{1}{q_n q_{n-1}}$ – q_n is increasing, so the sequence converges (Leibniz criterion?)
  - definition: value of continued fraction $[a_0 ...]$ is $\lim_{n \to \infty} \frac{p_n}{q_n}$, $\frac{p_n}{q_n}$ is called the n–th convergent of $[a_0 ...]$
  - every $\alpha \in \mathbb{R} \setminus \mathbb{Q}$ can be expressed as continued fraction – proof: just start building the fraction ($a_0 = \lfloor \alpha \rfloor$, $\alpha_1 = (\alpha - a_0)^{-1}$, $a_1 = \lfloor \alpha_1 \rfloor$...)

- continued fraction expansion of $\sqrt{N}$
  - if N is not a square, then $\sqrt{N}$ has infinite expansion as continued fraction
  - lemma: $N \in \mathbb{N} \forall n \exists R_n, S_n \in \mathbb{Z}(S_n \neq 0)$: $\alpha_n = \frac{R_n+\sqrt{N}}{S_n}$ (α_n as defined above), where $R_0 = 0, S_0 = 1$, $R_{n+1} = a_n S_n - R_n$, $S_{n+1} = \frac{N-R_{n+1}^2}{S_n}$
  - proof (by induction): $n = 0$ holds, then rewrites $\alpha_{n+1} = (\frac{R_n+\sqrt{N}}{S_n} - a_n)^{-1}$, and then simple algebraic rewritings
  - still need to prove that $S_n \in \mathbb{Z}$ (induction): $n = 0$ ok, need to show $S_n | N - R_{n+1}^2$ $N - R_{n+1}^2 = N - (a_n S_n - R_n)^2 = N - R_n^2 - S_n(...)$ – sufficient to show, that $S_n | N - R_n^2$, $N - R_n^2 = S_n S_{n-1}$ (definition of S_n), therefore if $S_0...S_n \in \mathbb{Z}$ then $S_{n+1} \in \mathbb{Z}$
- CFRAC relations
  - lemma: $\sqrt{N} \notin \mathbb{Q}$, then $\forall n \in \mathbb{N}_0 p_{n-1}^2 - N q_{n-1}^2 = (-1)^n S_n$ (p_i, q_i, S_i defined above)
  - proof: $\sqrt{N} = [a_0, ..., \alpha_n]$ (α_n – plugging the residual irrational value into the continued fraction) $= \frac{\alpha_n p_{n-1}+p_{n-2}}{\alpha_n q_{n-1}+q_{n-2}} = \frac{\frac{R_n+\sqrt{N}}{S_n}p_{n-1}+p_{n-2}}{\frac{R_n+\sqrt{N}}{S_n}q_{n-1}+q_{n-2}} = \frac{R_n p_{n-1}+\sqrt{N}p_{n-1}+p_{n-2}S_n}{R_n q_{n-1}+\sqrt{N}q_{n-1}+q_{n-2}S_n}$
  - this gives $(R_n p_{n-1} + p_{n-2}S_n)1 + p_{n-1}\sqrt{N} = N q_{n-1} 1 + (R_n q_{n-1} + q_{n-2}S_n)\sqrt{N}$,
  - as 1 and $\sqrt{N}$ are $\mathbb{Q}$–linearly independent, we get $p_{n-1} = R_n q_{n-1} + q_{n-2}S_n$ (multiply by p_{n-1}) and $N q_{n-1} = R_n p_{n-1} + p_{n-2}S_n$ (multiply by q_{n-1})
  - by summing the multiplied equations we get: $p_{n-1}^2 - N q_{n-1}^2 = S_n(p_{n-1}q_{n-2} - p_{n-2}q_{n-1}) = (-1)^n S_n$ (by definition)
- CFRAC
  - choose a factor base $\{-1, 2, 3...p_max\}$
  - generate continued fraction convergents of $\sqrt{N}$, try to factor S_n in the factor base, if so, add relation $(p_{n-1}, (-1)^n S_n)$
  - afterwards linear phase and factorization
- Pell's equation
  - want integral solutions of $x^2 - Ny^2 = 1$
  - Prihoda said – for state exams you need to know the following: solutions exists, and can be found using continuous fractions

Quadratic sieve

(tonelli-shanks)

- Tonelli–Shanks algorithm (square root mod odd p)

- idea:
  - $\mathbb{Z}_p^* \approx \mathbb{Z}_{p-1} \approx \mathbb{Z}_{2^e} \times \mathbb{Z}_l$ (l odd)
  - we have (but cannot compute) $\phi : \mathbb{Z}_p^* \to \mathbb{Z}_{2^e} \times \mathbb{Z}_l$
  - $z \in \mathbb{Z}_p^* : o(z) = 2^l \Leftrightarrow \phi(z) = (\alpha, 0)$ α even
  - $a \in \mathbb{Z}_p^* : a$ is a square $\Leftrightarrow \phi(a) = (\alpha, \beta)$ α odd
  - $\phi(a^l) = (\alpha, 0)$ α odd, $\phi(z) = (\beta, 0)$ β even
  - $k : k\beta = \alpha \bmod 2^e$ k must be even
  - $a^l = z^k \Rightarrow a = a^{l+1} z^{-k}$ – power are even, can divide by 2
  - z can be found as a l–th power of a quadratic nonresidue (can be checked easily)
  - finding k is finding a discrete logarithm in $\mathbb{Z}_{2^e}$ – Pohlig–Hellman
- Quadratic sieve
  - introduces sieving part into Dixon's factorization
  - generating relations: $Q(x) = (x - \lfloor\sqrt{N}\rfloor)^2 - N$ – relation $((x - \lfloor\sqrt{N}\rfloor), Q(x)$
  - take factor base ($p_{max} \approx e^{1/2\sqrt{ln(N)lnln(N)}}$), sieving interval $\{-M...M\}$ ($M \approx e^{\sqrt{ln(N)lnln(N)}}$)
  - trying to find $x \in \{-M...M\}$: $Q(x)$ can be factored in factor base
  - sieving removes operations $Q(x) div\ p$ if $p \nmid Q(x)$ – using Tonelli–Shanks algorithm finds solution to $Q(x) \equiv 0 \bmod p$
  - algorithm:
    * choose factor base B, size of sieving interval M
    * for $i \in \{-M...M\}$: $\{P[i] = Q(i)\}$
    * for all primes in factor base:
      · find $C = \{c \in \mathbb{Z}_p | Q(c) = 0 \bmod p\}$ – using Tonelli–Shanks algorithm
      · for $c \in C$: for $i \in \{-\lceil M/p\rceil, \lceil M/p\rceil\}$: $P[c+ip] /= p^{v_p(P[c+ip)}$ (divide by highest possible power of p)
      · this is the speedup due to sieving
    * for $i \in \{-M...M\}$ if $P[i] = \pm 1$ add relation $(i - \lfloor\sqrt{N}\rfloor, Q(i) \bmod N)$
    * linear phase and solution

Discrete logarithm

formulation - given group G, its generator g and $h \in G$, find $x \in \mathbb{N} : g^x = h$

in general, the problem is hard (solution require time $O(\sqrt{|G|})$)), however for some groups more efficient algorithms exist - $(\mathbb{Z}_n, +)$ - reduces to inverting - euclidean algorithm, index calculus where applicable

Bruteforce

- try all possibilities
- item complexity $O(|G|)S$

Pohlig-Hellman reduction

$|G| = \prod p_i^{e_i} \Rightarrow$ we may solve the problem in subgroups and then use CRT to get the solution.

input : G of order $n = p^e$, $g \in G$ generator and $h \in G$
output: $x \in \mathbb{Z}_n : g^x = h$
$x_0 = 0$;
$\gamma = g^{p^{e-1}}$;
for $k \in \mathbb{Z}_e$ **do**
    $h_k = (g^{-x_k} h)^{p^{e-1-k}}$;
    find d_k: $\gamma^{d_k} = h_k$;
    $x_{k+1} = x_k + p^k d_k$
end
return x_e

Algorithm 11: Pohlig-Hellman: one subgroup

Use this approach for all primes with values $g_i = g^{n/p_i^{e_1}}$ and $h_i = h^{n/p_i^{e_i}}$, combine through CRT.

Baby-step, giant-step

Time-memory tradeoff – $O(\sqrt{|G|})$ memory, $O(\sqrt{|G|})$ time.

input : Cyclic group G of order n, α generator, $\beta \in G$
output: $x : \alpha^x = \beta$ or fail
$m = \lceil \sqrt{n} \rceil$;
for $0 \le j < m$ **do**
    $T[\alpha^j] = j$;
end
$a = \alpha^{-m}$;
$\gamma = \beta$;
for $0 \le i < m$ **do**
    if $\gamma \in T$ **then**
        return $im + T[\gamma]$
    end
    else
        $\gamma = \gamma.a$
    end
end
return fail

Algorithm 12: Baby-step, giant-step

Index calculus

- only for some groups - needs decomposition into small elements (e.g. prime numbers))
- multiple phases
- 0. phase - choose a size of a factor base - $\{-1, 2, 3, 5, ..., p_r\}$
- 1. phase
  - find linear relations between elements of factor base, and power of the generator g ($g^k \bmod q = (-1)^{e_0} 2^{e1} 3^{e_2} ... p_r^{e_r}$)
  - exponents of the relations are saved as rows of an extended matrix (k is right side)
  - find enough relations, so the matrix has full rank
- 2. phase - using linear algebra transform the exponent matrix into row reduced echelon form - get logarithms of elements of factor base
- 3. phase
  - for different s we try to decompose in the factor base $g^s h \bmod q = (-1)^{f_0} 2^{f_1} ... p_r^{f_r}$
  - if we find a decomposition, then $x = f_0 log(-1) + f_1 log(2) + ... + f_r log(p_r) - s$
- size of factor base is important - too small -¿ hard to find relations and 3. phase also hard
- too big -> 2. phase takes too long
- expected time (assuming optimal size of factor base) $L_n[1/2, \sqrt{2} + o(1)]$
- only the 3. phase depends on h -> for a given group G and generator g it is possible to precompute and store row reduced echelon exponent matrix and then compute logarithms quickly -¿ logjam attack on 512-bit Diffie-Hellmann