Číselné algoritmy

Generické algoritmy pro výpočet DLP

24. 5. 2021

MPQS

Doposud jsme pracovali s kvadratickým sítem, kde byly relace generovány jedním polynomem $Q(x)$.
P. Montgomery navrhl variantu kvadratického síta, které využije více polynomů. Tento návrh umožňuje lepší paralelizaci kvadratického síta, dále je možno zkrátit prosívací interval. Nevýhodou je pak čas strávený inicializací polynomů, navíc pro každý polynom a každé prvočíslo z faktorizační báze hledáme kořeny polynomu modulo p.

Polynomy pro MPQS

Uvažujeme polynomy tvaru $ax^2 + 2bx + c$, kde $a \in \mathbb{N}$, $b, c \in \mathbb{Z}$, $b^2 - ac \in \mathbb{N}$, $N \mid b^2 - ac$.
Pak $aQ(x) = a^2x^2 + 2abx + ac = (ax + b)^2 - (b^2 - ac)$. Pokud I_Q je prosívací interval polynomu Q, dostáváme relace tvaru

$$(az + b, aQ(z)), z \in I_Q$$

Necht' $M \in \mathbb{N}$ je takové, že $I_Q = \{\lfloor -\frac{b}{a} \rfloor - M, \ldots, \lfloor -\frac{b}{a} \rfloor + M\}$.
Hodnoty a, b, c se snažíme nastavit tak, aby $\min_{z \in I_Q} aQ(z)$ a $\max_{z \in I_Q} aQ(z)$. měly zhruba stejnou absolutní hodnotu.
Protože je $\min_{z \in I_Q} aQ(z) \approx -(b^2 - ac)$,
$\max_{z \in I_Q} aQ(z) \approx (aM)^2 - (b^2 - ac)$, rovnost

$$(b^2 - ac) \approx (aM)^2 - (b^2 - ac)$$

vede k volbě $a \approx \frac{\sqrt{2(b^2 - ac)}}{M}$. Na takto voleném prosívacím intervalu je maximální hodnota $|Q(z)|$ asi $M\sqrt{\frac{b^2 - ac}{2}}$. Pokud a, b, c volíme tak, aby $N = b^2 - ac$, jsou hodnoty $|Q(z)|$ na I_Q srovnatelné s hodnotami $(x + \lfloor \sqrt{N} \rfloor)^2 - N$ na intervalu $\{-M, \ldots, M\}$.

Inicializace polynomu pro MPQS

- Pro dané M udávající délku prosívacího intervalu zvolíme $a \in \mathbb{P}$, $a \approx \frac{\sqrt{2N}}{M}$, $(\frac{N}{a}) = 1$
- určíme $b \in \mathbb{Z}_a$ tak, aby $b^2 \equiv N \pmod{a}$
- dopočteme $c = \frac{b^2 - N}{a}$
- prvočíslo a přidáme do faktorizační báze

MPQS pár poznámek

Jak bylo zmíněno, pro každý polynom Q a každé prvočíslo p z faktorizační báze potřebujeme najít $\{c \in \mathbb{Z}_p \mid Q(c) \equiv 0 \pmod{p}\}$.
Pokud máme vždy $b^2 - ac = N$, je výhodné spočíst nejprve odmocninu z N v $\mathbb{Z}_p$ a pomocí této odmocniny dopočíst kořeny jednotlivých polynomů.
Dále se využívá inicializace více polynomů se stejným vedoucím koeficientem. Postup upravíme tak, že a volíme ve tvaru $a = p_1 p_2 \dots p_t$, kde $2 < p_1 < p_2 < \cdots < p_t$ a $(\frac{N}{p_i}) = 1$ pro každé p_i.
Kongruence $b^2 \equiv N \pmod{a}$ má pak 2^t různých řešení v $\mathbb{Z}_a$, každé z nich lze použít pro inicializaci polynomu pro MPQS.

Problém diskrétního logaritmu

Dána je konečná cyklická grupa G a její generátor g. Prozatím budeme předpokládat, že známe $n = |G|$. Pro $h \in G$ označme $\mathrm{dlog}_g(h)$ prvek množiny $\{0, 1, \ldots, n-1\}$, pro který je

$$h = g^{\mathrm{dlog}_g(h)}.$$

Snadno se přesvědčíme, že

a) $\forall h_1, h_2 \in G$ je $\mathrm{dlog}_g(h_1 h_2) = \mathrm{dlog}_g(h_1) + \mathrm{dlog}_g(h_2) \bmod n$

b) $\forall h \in G, \forall z \in \mathbb{Z}$ je $\mathrm{dlog}_g(h^z) = z\,\mathrm{dlog}_g(h) \bmod n$

Redukce Pohlig - Hellman, úvod

Při znalosti úplné faktorizace n se problém výpočtu $\mathrm{dlog}_g(-)$ redukuje na výpočet $\mathrm{dlog}_{g^u}(-)$ v podgrupách G prvočíselného řádu.
Připomeňme, že cyklická grupa řádu n má pro každé $d \mid n$ právě jednu podgrupu řádu d. Tato podgrupa je generovaná $g^{\frac{n}{d}}$.

Redukce Pohlig-Hellman, první část

Předpokládejme, že $n = u.v$, kde $u, v \in \mathbb{N}$ a $\mathrm{NSD}(u, v) = 1$.
Pak $\mathrm{dlog}_g(h)$ lze vyjádřit pomocí $\mathrm{dlog}_{g^u}(h^u)$ a $\mathrm{dlog}_{g^v}(h^v)$.
Výpočet problému diskrétního logaritmu v G se převede na výpočet diskrétního logaritmu v podgrupě řádu v a v podgrupě řádu u.
Postup:

Najdi $a, b \in \mathbb{Z}$ tak, aby $au + bv = 1$ (rozš. Euklid alg.)

Spočti $\ell_u := \mathrm{dlog}_{g^u}(h^u)$, $\ell_v := \mathrm{dlog}_{g^v}(h^v)$

$\mathrm{dlog}_g(h) := (u\ell_u a + v\ell_v b) \bmod n$

Korektnost:
$h = h^{au+bv} = (h^u)^a(h^v)^b = ((g^u)^{\ell_u})^a((g^v)^{\ell_v})^b = g^{u\ell_u a+v\ell_v b}$

Poznámka ke složitosti

Při analýze algoritmů tohoto typu je zvykem oddělovat grupové operace a bitové operace (nevíme, jak náročná je pro konkrétní grupu jedna grupová operace).
V našem postupu počítáme h^u, g^u, h^v, g^v při užití rychlého umocnění uděláme $O(\max(\log(u), \log(v)))$ grupových operací. Dále počítáme a, b v čase $O(\log(u)\log(v)) \subseteq O(\log^2(n))$. Protože koeficienty a, b mají vlastnost $|a| \leq v, |b| \leq u$, lze výpočet $(u\ell_u a + v\ell_v b) \bmod n$ provést v čase $O(\log^2(n))$. Další bitové operace uděláme při výpočtu rychlého umocnění, ty ale nezvýší asymptotický odhad.

Pohlig-Hellmanova redukce, druhá část

Předpokládejme, že $n = p^e$, kde $p \in \mathbb{P}$ a $e \in \mathbb{N}$. Výpočet $\mathrm{dlog}_g(h)$ lze provést pomocí e výpočtů $\mathrm{dlog}_{g^{p^{e-1}}}(-)$, kde $\langle g^{p^{e-1}} \rangle$ je (jediná) podgrupa G řádu p.
Princip: $\ell = \mathrm{dlog}_g(h)$ hledáme ve tvaru

$$\ell = \sum_{i=0}^{e-1} b_i p^i, b_0, b_1, \ldots, b_{e-1} \in \{0, 1, \ldots, p-1\}$$

Postupně odhalujeme $b_0, b_1, \ldots, b_{e-1}$

Jak spočíst b_0

Vyjdeme ze vztahu

$$h = g^{\ell} = g^{\sum_{i=0}^{e-1} b_i p^i},$$

který umocníme na p^{e-1}:

$$h^{p^{e-1}} = g^{p^{e-1}\ell} = g^{\sum_{i=0}^{e-1} b_i p^{i+e-1}} = (g^{p^{e-1}})^{b_0},$$

protože $g^{bp^j} = (g^{p^j})^b = 1$ pro každé $b \in \mathbb{Z}$ a $j \geq e$.
Z toho dostáváme

$$b_0 = \mathrm{dlog}_{g^{p^{e-1}}}(h^{p^{e-1}})$$

Jak spočíst b_{i+1} z $b_0, b_1, \ldots, b_i$

Předpokládejme, že jsme spočetli $b_0, b_1, \ldots, b_i$ pro $i < e-1$ a chceme spočíst b_{i+1}.
Postup je podobný jako pro výpočet b_0: Vyjdeme ze vztahu

$$h = g^{\ell},$$

který umocníme na $p^{e-1-i-1}$:

$$h^{p^{e-1-i-1}} = g^{p^{e-1-i-1}\ell} = g^{\sum_{t=0}^{e-1} b_t p^{t+e-1-i-1}} = g^{\sum_{t=0}^{i+1} b_t p^{t+e-1-i-1}}$$

$$(g^{p^{e-1}})^{b_{i+1}} = (hg^{-\sum_{t=0}^{i} b_t p^t})^{p^{e-i-2}}$$

Odtud již dostaneme

$$b_{i+1} = \mathrm{dlog}_{g^{p^{e-1}}}((hg^{-\sum_{t=0}^{i} b_t p^t})^{p^{e-i-2}})$$

Baby-Step Giant-Step

Algoritmus založen na principu 'time-memory tradeoff'.
V cyklické grupě $G = \langle g \rangle$ řádu n chceme pro $h \in G$ určit $\ell = \mathrm{dlog}_g(h)$.
Označme $m := \lceil \sqrt{n} \rceil$. Protože $\ell \in \{0, \ldots, n-1\}$ existují $a, b \in \{0, \ldots, m-1\}$ takové, že $\ell = am + b$.
Pak $g^\ell = h \Rightarrow g^{am+b} = h \Rightarrow g^b = (g^{-m})^a h$.

Algoritmus Baby-Step Giant-Step

Vstup: $h \in G, g \in G$, $n = |G|$; grupa $G = \langle g \rangle$ zadána nějakými parametry.

Výstup: $\ell \in \{0, 1, \ldots, n-1\}$ takové, že $h = g^\ell$

0: $m := \lceil \sqrt{n} \rceil$

1: (baby steps) Pro $i = 0, 1, 2, \ldots, m-1$ spočti g^i. Prvky ulož do paměti tak, aby vznikl seznam, ve kterém lze rychle vyhledávat.

2: $S := g^{-m}$, $H := h$

3: **for** $a = 0$ **to** $m-1$ **do**

zjisti zda $H \in \{g^i \mid i \in \{0, 1, \ldots, m-1\}\}$. Pokud ano, urči $b \in \{0, \ldots, m-1\}$, pro které je $H = g^b$ a jdi do bodu 4.
$H := S.H$

4. **return** $\ell = am + b$

Korektost: Pokud jdeme do bodu 4., je $H = g^{-ma}h = g^b$, a tedy také $h = g^{am+b}$.

Protože m je dostatečně velké, lze každé číslo z množiny $\{0, 1, \ldots, n-1\}$ vyjádřit ve tvaru $am + b, a, b \in \{0, 1, \ldots, m-1\}$, algoritmus proto vždy z cyklu 3 vyskočí.

Pár slov ke složitosti

Spočtěme grupové operace algoritmu:

- g^0 - zavolání konstanty je formálně taky operace v grupě
- v kroku 1. počítáme $g^i := g^{i-1}.g$, pro $i = 1, \ldots, m-1$ - jedna grupová operace. Celkem $m-1$ násobení.
- Výpočet S v kroku 2.: $S = (g^{m-1}.g)^{-1}$ - jedno násobení a jedna inverze
- V každé iteraci kroku 3. se provede jedno násobení. Na násobení v poslední iteraci ale již nedojde. V kroku 3. se proto provede nejvýše $m-1$ operací

Počet grupových operací provedených algoritmem je proto nejvýše $1 + m - 1 + 2 + m - 1 = 2m + 1$. Lze ukázat, že u generických algoritmů pro počítání DLP nelze očekávat výrazně lepší asymptotickou složitost.

S bitovými operacemi je to složitější: V kroku 4 pocítáme ℓ v čase $O(\log^2(m))$. Dále je třeba uložit prvky $\{g^i \mid i \in \{0, 1, \ldots, m-1\}\}$ do paměti ve struktuře, ve které půjde efektivně vyhledávat.

Pokud použijeme třídění typu Quicksort s průměrnou složitostí $O(m\log(m))$ budeme v každé iteraci vyhledávat v čase $O(\log(m))$

Paměťová náročnost

Nevýhodou algoritmu je jeho paměťová náročnost, potřebujeme uložit $\lceil \sqrt{n} \rceil$ prvků do paměti. Existují různé mechanismy (trie, hashovací funkce), které umožňují trochu snížit nároky algoritmu na paměť.

Baby-Step Giant-Step bez znalosti řádu

Pokud neznáme řád $|G|$ můžeme za m volit nějaký jeho dolní odhad (např. $m = 1$).
V případě, že algoritmus neuspěje, hodnotu m zdvojnásobíme a pustíme algoritmus znovu.
Toto opakujeme tak dlouho, dokud nenajdeme výstup.
Podívejme se na odhad počtu volaných operací v G pro takovýto algoritmus.

$$M := \lceil \log(\sqrt{|G|}) \rceil$$

Předpokládejme, že postupně zkoušíme $m = 0, 1, 2, 4, 8, \ldots$. Počet operací pro $m = 2^i$ odhadneme jako $2^{i+1} + 1$. Pak

$$\sum_{i=0}^{M} 2^{i+1} + 1 = (2^{M+2} - 2) + (M + 1) =$$

$$4 * 2^M + M - 1 \leq 4 * 2^{\log(\sqrt{|G|})+1} + M - 1 \leq 8\sqrt{|G|} + M\,.$$

Počet grupových operací volaných touto variantou je tedy stále $O(\sqrt{|G|})$.

Pollardova ρ-metoda pro diskrétní logaritmus

Pamět'ovou náročnost algoritmu Baby-step Giant-step odstraňuje Pollardova ϱ-metoda.
Metoda vyžaduje zobrazení $f : G \to G$, kterým lze generovat dostatečně dlouhé 'náhodné' posloupnosti prvků G (podobně jako u faktorizační metody Pollard ϱ). Pokud bychom mohli uplatnit narozeninový paradox, bude střední hodnota grupových operací provedených v algoritmu $O(\sqrt{|G|})$.

Příklad zobrazení f

Předpokládejme, že $G = \langle g \rangle$, na vstupu je prvek h a chceme určit $\mathrm{dlog}_g(h)$.
Dále předpokládáme, že grupa je rozdělena na disjunktní sjednocení tří zhruba stejně velkých množin I, II, III. Tedy $G = I \dot\cup II \dot\cup III$.
Pro $x \in G$ definujeme $f(x)$ takto:

- $f(x) := hx$ pro $x \in I$
- $f(x) := x^2$ pro $x \in II$
- $f(x) := gx$ pro $x \in III$

Příklad: Pro $G = \mathbb{Z}_p^*$, kde $p \in \mathbb{P}$ bychom mohli volit

$$I := \{1, 2, \ldots, \lfloor \frac{p-1}{3} \rfloor\},$$

$$II := \{\lfloor \frac{p-1}{3} \rfloor + 1, \lfloor \frac{p-1}{3} \rfloor\} + 2, \ldots, \lfloor \frac{2(p-1)}{3} \rfloor\},$$

$$III := \{\lfloor \frac{2(p-1)}{3} \rfloor + 1, \lfloor \frac{2(p-1)}{3} \rfloor + 2, \ldots, p-1\}.$$

Princip algoritmu Pollard ϱ

Algoritmus generuje posloupnost $x_0, x_1, x_2, \cdots \in G$ takovou, že $x_{i+1} = f(x_i)$ pro každé $i \in \mathbb{N}_0$. Pro každé i dále počítá $(a_i, b_i) \in \mathbb{Z}_n \times \mathbb{Z}_n$ takovou, že

$$x_i = g^{a_i} h^{b_i} .$$

Pokud volíme $x_0 = g$, definujeme $(a_0, b_0) := (1, 0)$. Máme-li spočteno (a_i, b_i), pro funkci f z předchozího slajdu definujeme (a_{i+1}, b_{i+1}) předpisem

- $(a_{i+1}, b_{i+1}) := (a_i, b_i + 1 \bmod n)$ pro $x_i \in I$
- $(a_{i+1}, b_{i+1}) := (2a_i \bmod n, 2b_i \bmod n)$ pro $x_i \in II$
- $(a_{i+1}, b_{i+1}) := (a_i + 1 \bmod n, b_i)$ pro $x_i \in III$

Kolize

Cílem je najít $i \neq j \in \mathbb{N}_0$ pro které je $x_i = x_j$. Pak máme

$$g^{a_i} h^{b_i} = g^{a_j} h^{b_j} \Rightarrow g^{a_i - a_j} = h^{b_j - b_i} .$$

Pokud je $\mathrm{NSD}(b_j - b_i, n) > 1$, algoritmus skončí neúspěšně.
V opačném případě najdeme $c \in \mathbb{Z}$ tak, aby
$c(b_j - b_i) \equiv 1 \pmod{n}$. Potom

$$h = (h^{b_j - b_i})^c = (g^{a_i - a_j})^c = g^{c(a_i - a_j)} .$$

Takže

$$\mathrm{dlog}_g(h) = c(a_i - a_j) \bmod n .$$

Pokud algoritmus skončí neúspěchem, je možno začít s jinou hodnotou $x_0 = g^{a_0} h^{b_0}$, kde a_0, b_0 lze volit náhodně.
Podobně jako při faktorizačním algoritmu hledáme kolize ve speciálním tvaru. Například Pollard-Floyd hledá kolize typu $x_i = x_{2i}$.

To je pro dnešek vše

Děkuji za pozornost.