Číselné algoritmy

Pollardova ϱ-metoda

8. 3. 2021

Složitost základních aritmetických operací

Při odhadech složitosti algoritmů budeme předpokládat využití klasických algoritmů pro výpočet aritmetických operací. Jejich složitost viz skripta *Počítačová algebra* str. 25.
Máme-li $x, y \in \mathbb{Z}$, x má n cifer, y má n cifer, přičemž $n \geq m$, složitost aritmetických operací je

- $x \pm y$ složitost $O(n)$
- $x.y$ složitost $O(mn)$
- $x \operatorname{div} y$, $x \bmod y$ složitost $O(m(n-m+1)) \subseteq O(n^2)$
- $\mathrm{NSD}(x, y)$ složitost $O(mn) \subseteq O(n^2)$.

Příklad: $N \in \mathbb{N}$, $x, y \in \mathbb{Z}_N$ časovou složitost výpočtu $xy \bmod N$ budeme uvažovat jako $O(\log^2(N))$.
Spočteme $x.y$ v čase $O(\log^2(N))$ a pak $xy \bmod N$. Pokud je $xy > N$, bude tento výpočet v čase
$O(\log(N)(2\log(N) - \log(N) + 1)) = O(\log^2(N))$.

Prime number theorem

Pro $r \in \mathbb{R}$, $r > 2$ označme $\pi(r) := |\mathbb{P} \cap (0, r\rangle|$.

Věta

(Haddamard, de la Vallée Poussin 1896)

$$\lim_{r \to \infty} \pi(r)/(r/\ln(r)) = 1$$

Typicky budeme hodnotu $\pi(r)$ aproximovat jako $r/\ln(r)$ (při použití v asymptotických odhadech složitosti algoritmů je to v pořádku). Ve skutečnosti by nám stačil slabší výsledek $\pi(r) \in O(r/\ln(r))$, jehož důkaz je výrazně snažší.

Zkusmé dělení

Vstup: $N \in \mathbb{N}$ složené

Výstup: vlastní dělitel N

1. $i := 2$
2. **while** $i \nmid N$ **do**
   $i := i + 1$
3. **return** i

Odhad složitosti (v nejhorším případě) $O(\sqrt{N}\,\log^2(N))$.
Někdy se může hodit odhad složitosti vyjádřený i jinak než jen na délce vstupu. Pokud p je nejmenší prvočíselný dělitel N, je odhad složitosti $O(p\,\log^2(N))$.

Zkusmé dělení 2

U některých algoritmů budeme předpokládat, že mají k dispozici předpočítanou dostatečně dlouhou posloupnost prvočísel.

Vstup: $N \in \mathbb{N}$, složené

Výstup: vlastní dělitel N

k dispozici: $p_1, p_2, \ldots, p_k$ posloupnost všech prvočísel $\leq \sqrt{N}$

1. $i := 1$
2. **while** $p_i \nmid N$ **do**
   $i := i + 1$
3. **return** p_i

Pokud budeme uvažovat počet prvočísel $\leq \sqrt{N}$ zhruba $\sqrt{N}/\ln(\sqrt{N})$ dostaneme odhad složitosti tvaru $O(\sqrt{N}\,\log(N))$.

Pollard-ϱ, úvod

- J. Pollard: *A Monte-Carlo method for factorization*, BIT **15**(1975), 331-334
- V roce 1981 pomocí této metody faktorizováno $F_8 = 2^{2^8} + 1$ (J. Pollard, R. Brent)

Posloupnosti generované funkcí

Mějme konečnou množinu $S \neq \emptyset$ a zobrazení $f : S \to S$. Uvažme posloupnost prvků $s_0, s_1, \dots \in S$ generovanou takto:

- $s_0 \in S$ je zvolen
- zbylé prvky dané rekurentním vztahem $s_{i+1} = f(s_i)$, $i \in \mathbb{N}_0$.

Pro takovéto posloupnosti definujeme pojem *perioda* a *preperioda* posloupnosti.

Definice

1. *Preperioda posloupnosti* $s_0, s_1, \dots$ *je nejmenší* $M \in \mathbb{N}_0$ *takové, že existuje* $M' > M$ *splňující* $s_M = s_{M'}$.
2. *Perioda posloupnosti* $s_0, s_1, \dots$ *je nejmenší* $T \in \mathbb{N}$ *takové, že* $s_M = s_{M+T}$, *kde* M *je preperioda posloupnosti.*

Pozorování

V posloupnosti $s_0, s_1, \dots \in S$ *s preperiodou* M *a periodou* T *platí* $s_i = s_j$ *právě když* $i = j$ *nebo* $M \leq i, j$ *a* $T | i - j$.

Aplikace v Pollardově ϱ-metodě

Pollardova ϱ-metoda potřebuje zobrazení $f : \mathbb{Z}_N \to \mathbb{Z}_N$, pro které platí

1. Pro každé $x \in \mathbb{Z}_N$ lze efektivně spočíst $f(x)$.
2. f je kompatibilní s $\bmod p$, kde $p \in \mathbb{P}$ je dělitel N (tedy existuje funkce $\overline{f} : \mathbb{Z}_p \to \mathbb{Z}_p$ splňující $\overline{f}(x \bmod p) = f(x) \bmod p$ pro každé $x \in \mathbb{Z}_N$)
3. Je-li p nejmenší prvočíselný dělitel N potřebujeme 'náhodné chování' funkce $\overline{f}$.

Obvykle se volí polynomy ze $\mathbb{Z}_N$, typicky stupně 2. Například

$$f : x \mapsto x^2 + 1 \bmod N$$

(některé volby se nedoporučují např. $f = x^2 \bmod N$, $f = x^2 - 2 \bmod N$)
Zobrazení $\overline{f}$ lze definovat jako $f \bmod p$.

Idea Pollardovy ϱ-metody

Na vstupu je složené N, označme $p \in \mathbb{P}$ dělitele N.
Pomocí vhodné funkce $\overline{f} : \mathbb{Z}_p \to \mathbb{Z}_p$ generujeme posloupnost $\overline{s_0}, \overline{s_1}, \cdots \in \mathbb{Z}_p$ výše uvedenou metodou tj platí $\overline{s_{i+1}} = \overline{f}(\overline{s_i})$ pro každé $i \in \mathbb{N}_0$.
Snažíme se najít kolizi, tj dvojici indexů $i \neq j \in \mathbb{N}_0$ pro kterou je $\overline{s_i} = \overline{s_j}$. Tuto kolizi pak zkusíme využít pro nalezení faktorizace N.
Základní problém: Neznáme p. Proto generujeme posloupnost $s_0, s_1, \cdots \in \mathbb{Z}_N$ pomocí vhodného zobrazení $f : \mathbb{Z}_N \to \mathbb{Z}_N$ (tedy s_0 volíme, zbytek počítáme ze vztahu $s_{i+1} = f(s_i)$).
Pokud položíme $\overline{s_i} := s_i \bmod p$, bude platit $\overline{s_{i+1}} = \overline{f}(\overline{s_i})$. Tyto prvky sice neznáme, ale jsme schopni detekovat, zda náhodou neplatí $\overline{s_i} = \overline{s_j}$:

$$\overline{s_i} = \overline{s_j} \Rightarrow \mathrm{NSD}(s_i - s_j, N) > 1$$

'Náhodné chování' zobrazení $\overline{f}$ poslouží k heuristickému odhadu složitosti. Pokud bychom mohli použít narozeninový paradox, očekáváme, že v průměrném případě kolize nastane asi po $O(\sqrt{p})$ krocích.

Naivní algoritmus Pollard ϱ

Vstup: $N \in \mathbb{N}$ složené

Výstup: Vlastní dělitel N, nebo neúspěch

1. Zvolíme vhodné f např. $f(x) = x^2 + 1 \bmod N$
2. Zvolíme náhodně $s_0 \in \mathbb{Z}_N$ (často též $s_0 := 2$)
3. for $j = 1$ to N do:

   $s_j := f(s_{j-1})$
   for $i = 0$ to $j - 1$ do:

   $d := \mathrm{NSD}(s_i - s_j, N)$
   if $d > 1$ then GOTO 4.
4. if $d < N$ then return d

   else return neúspěch

Kde je třeba algoritmus vylepšit

Pokud očekáváme, že lze použít narozeninový paradox, bude v průměrném případě počet iterací vnějšího cyklu $O(\sqrt{p})$. Pro uložení prvků posloupnosti bychom potřebovali tedy $O(\sqrt{p}\log(N))$ bitů.
Co je ale ještě horší je časová složitost: Pro $j = 1, 2, \ldots$ počítáme v j-té iteraci j krát $\mathrm{NSD}(s_i - s_j, N)$. Pokud je počet iterací vnějšího cyklu $O(\sqrt{p})$, máme $O(p)$ výpočtů NSD a algoritmus vychází hůř než zkusmé dělení.

Floydova metoda hledání cyklů

Tvrzení

Máme posloupnost $s_0, s_1, \cdots \in S$ *s preperiodou* M *a periodou* T. *Pak existuje* $0 < i \leq M + T$ *takové, že* $s_i = s_{2i}$

Důkaz: Hledáme $i > 0$, pro které bude platit $i \geq M$ a $T \mid 2i - i$, neboli hledáme kladný násobek periody, který bude $\geq M$. Pro $M = 0$ volíme $i = T$, pro $M > 0$ volíme $\lceil \frac{M}{T} \rceil T$.

Algoritmus Pollard-Floyd

Vstup: $N \in \mathbb{N}$ složené

Výstup: Vlastní dělitel N, nebo neúspěch

1. Zvolíme vhodné f např. $f(x) = x^2 + 1 \bmod N$
2. Zvolíme náhodně $s_0 \in \mathbb{Z}_N$ (často též $s_0 := 2$)
3. $s := f(s_0)$, $s' := f(s)$
4. **while** $(\mathrm{NSD}(s - s', N) = 1)$ **do**
   $s := f(s), s' := f(f(s'))$
5. **if** $\mathrm{NSD}(s - s', N) < N$ **then return** $\mathrm{NSD}(s - s', N)$
   else return neúspěch

Poznámka: V případě neúspěchu lze počítat s jinou hodnotou s_0 případně s jinou volbou f.

Pár slov ke složitosti

V rámci jedné iterace dochází ke třem výpočtům funkce f (čas $O(\log^2(N))$) a jednomu výpočtu NSD (čas $O(\log^2(N))$). Rozhodující pro složitost algoritmu je počet iterací cyklu. Označme p nejmenší prvočíselný faktor N. Předpokládejme, že platí následující předpoklad:
Pro zvolené s_0 (případně i f) označme M a T předperiodu a periodu posloupnosti $s_0 \bmod p, s_1 \bmod p, s_2 \bmod p, \ldots$, kde $s_{i+1} = f(s_i)$. Uvažme $M + T$ jako náhodnou veličinu. Pak střední hodnotu $M + T$ lze interpretovat jako střední hodnotu délky náhodně vybrané posloupnosti prvků $\mathbb{Z}_p$, které vybíráme tak dlouho, dokud se nějaký člen posloupnosti neobjeví podruhé.
S tímto předpokladem dostaneme pro $p \to \infty$ střední hodnotu $M + T \approx \sqrt{\frac{\pi p}{2}}$.
Složitost algoritmu v průměrném případě by pak byla $O(\sqrt{p}\log^2(N))$, kde p je nejmenší prvočíselný dělitel N. V závislosti na délce vstupu lze pak odhad napsat ve tvaru $O(N^{\frac{1}{4}}\log^2(N))$.

Drobné vylepšení

V praxi je výpočet $\mathrm{NSD}(s - s', N)$ typicky náročnější než výpočet součinu v $\mathbb{Z}_N$.
Algoritmus lze modifikovat tak, aby k výpočtu NSD docházelo jenom v každé k-té iteraci, kde k volíme (např. $k = 100$).
Zavedeme proměnnou x, inicializujeme na $x := 1$.
V každé iteraci pak provedeme $x := x(s - s') \bmod N$.
V každé k-té iteraci provedeme $\mathrm{NSD}(x, N)$. Pokud vyjde 1, nastavíme $x := 1$ uložíme hodnoty s, s' do paměti a pokračujeme dále.
Pokud ne, dohledáme, kde nastala kolize.

Konec druhé přednášky

Domácí úkol k Pollardově ϱ-metodě bude zveřejněn koncem týdne.
`http://artax.karlin.mff.cuni.cz/~ppri7485/algoritmy/`