Dokumentace zápočtového programu z Programování II (NPRG031)

Testování prvočíselnosti

David Pěgřímek

http://davpe.net

Úvodem

V různých oborech (například v kryptografii) je potřeba zjistit, zda je číslo (v praxi velké) prvočíslem. Existuje řada prvočíselných testů, já jsem si vybral Rabinův-Millerův. Původní (Millerova) verze testu je deterministická, ale závisí na zatím nedokázané zobecněné Riemannově hypotéze [1]. Později byl test pozměněn na pravděpodobnostní panem Rabinem [2]. Pokud test odpoví, že testované číslo je složené, má vždy pravdu. Odpoví-li, že je prvočíslo, je pravděpodobnost $1/4^k$, že číslo je ve skutečnosti složené (k je počet iterací testu a lze jej v programu nastavit). Algoritmus jsem implementoval pro dlouhá čísla, s maximálně 50 ciframi.

Poznámka: V této verzi programu je k dispozici i slabší, Fermatův test, který lze zapnout v parametru programu (viz. níže).

Pro uživatele

Program se standardně přeloží pomocí příkazu `make`. Ve složce *bin* je navíc přeložený program pro systém Windows s názvem *primality_test.exe*.

Parametry programu

Program bez parametrů

Po spuštění bez parametrů, program čte čísla ze standardního vstupu. Uživatel zadává kladná přirozená čísla menší než 10^{51}. Program poté odpoví 1, pokud zadané číslo je prvočíslo, nebo 0 pokud je složené. Čtení čísel je zastaveno ukončením vstupu (tzv. `EOF` - End of File).

Nepovinné parametry programu

bin/primality_test [*vstupní soubor*] [*výstupní soubor*] [*iterace*] [*test*]

vstupní soubor
Jméno souboru, ze kterého bude program načítat čísla k testování. Pro načítání ze standardního vstupu vložte místo jména souboru pomlčku.

výstupní soubor
Jméno souboru, do kterého bude program ukládat výsledky testů. Pro ukládání do standardního vstupu vložte místo jména souboru pomlčku.

iterace
Číslo mezi 1 až 999 které značí, kolikrát se prvočíselný test provede. Větší počet iterací znamená větší pravděpodobnost, že test správně rozezná prvočíslo od složeného čísla. Velký počet iterací ovšem zpomalí čas testování. Výchozí hodnota je 10.

test
Písmeno určující, který algoritmus pro testování prvočíselnosti se použije. Zatím je k dispozici Rabinův-Millerův test (písmeno **r**) a Fermatův test (písmeno **f**). Výchozí hodnota je „r“.

Možná zlepšení

Pro zvýšení rychlosti programu by se hodilo implementovat aritmetické operace (násobení a modulení) jinými, než školními algoritmy. Rovněž by se dala ušetřit i paměť – třeba pomocí statické alokace (a použití tzv. arena allocator) a chytřejšího uložení binárního čísla.

Chybové návratové hodnoty

V případě, že se vyskytne známý problém, program skončí s nenulovou návratovou hodnotou a vypíše chybovou hlášku. Zde je pro úplnost seznam chybových návratových hodnot a popis chyby.

kód	*Popis chyby*
1	Číslo zadané k otestování obsahuje nečíselný znak (např. "+12", "1 000", "3 ").
2	Číslo zadané k otestování má více než 50 cifer.
3	V parametru *vstupní soubor* je zadán soubor, který nelze otevřít (např. neexistuje).
4	V parametru *výstupní soubor* je zadán soubor, který nelze otevřít.
5	V parametru *iterace* je zadáno příliš velké číslo (lze zadat číslo mezi 1 až 999).
6	V parametru *iterace* je zadáno číslo obsahující nečíselný znak.
7	Parametr *test* musí být buď písmeno „f“, nebo „r“.

Testovací data

Ve složce *tests* jsou k dispozici čtyři soubory s testovacími daty.

carmichael.in	Patnáct složených, carmichaelových čísel.
composites.in	Patnáct složených čísel.
primes_small.in	Deset prvočísel o nejvýše 10 cifrách.
primes_big.in	Deset prvočísel s více než 10 a nejvýše 50 ciframi.

Pro programátory

Teorie za testováním prvočíselnosti

Malá Fermatova věta: Nechť p je prvočíslo, pak pro číslo $a \in [1, p)$, které je s p nesoudělné platí

$$a^{p-1} \equiv 1 \pmod{p}$$

Z této věty se dá vyvodit jednoduchý pravděpodobnostní test prvočíselnosti.

Fermatův test prvočíselnosti:

Zvolíme počet iterací `iterace` a číslo n, které chceme otestovat.
for $i \leftarrow 1$ **to** `iterace` **do**
  Vygenerujeme náhodné $a \in [1, n)$
  if $a^{n-1} \neq 1 \pmod{n}$ **then**
    Číslo n nesplňuje MFV a nemůže tedy být prvočíslo (a je toho Fermatův svědek).
    return složené
  end if
end for
Nenašli jsme žádného svědka, n je pravděpodobně prvočíslo.
return prvočíslo

Časová složitost testu je $O(kN^2)$ kde N je počet bitů čísla n a k je počet iterací. Tento test však není všemocný, problém mu dělají takzvaná Carmichaelova čísla, která nesprávně prohlašuje za prvočísla.

Carmichaelovo číslo: Složené číslo n, pro které platí

$$a^{n-1} \equiv 1 \pmod{n}$$

pro všechna $a \in [1, p)$, která jsou s n nesoudělná.

Věta o diskrétní odmocnině: Je-li p liché prvočíslo, pak $a \in [1, p)$ má buď právě dvě odmocniny, nebo žádnou.

Důsledek: Číslo 1 má právě dvě odmocniny: 1 a $n - 1$.

Pomocí tohoto důsledku a následujícího lemmatu lze zobecnit Malou Fermatovu větu.

Lemma: Nechť n je liché přirozené číslo. Pak $n - 1$ lze napsat jako $2^s \cdot t$, kde t je rovněž liché přirozené číslo.

Zobecněná Malá Fermatova věta: Je-li n je liché prvočíslo, pak podle lemmatu víme že $n - 1 = 2^s \cdot t$ pro liché t. Nechť $a \in [1, n)$, pak pro posloupnost

$$a^{\frac{n-1}{2}} = a^{2^{s-1} \cdot t},\ a^{2^{s-2} \cdot t},\ \ldots,\ a^t \pmod{n}$$

platí: buď jsou všechny členy posloupnosti rovny 1, nebo existuje k, že k-tý člen posloupnosti je roven $n - 1$ a všechny členy posloupnosti v pořadí před k jsou rovny 1.

Nyní podle této věty vylepšíme Fermatův prvočíselnostní test. Tento test se jmenuje Rabinův-Millerův a jeho předností je, že mu nedělají potíže Carmichaelova čísla.

Rabinův-Millerův test prvočíselnosti:

Zvolíme počet iterací `iterace` a číslo n, které chceme otestovat.
for $i \leftarrow 1$ **to** `iterace` **do**
  Nejprve provedeme Fermatův test.
  Vygenerujeme náhodné $a \in [1, n)$.
  if $a^{n-1} \neq 1 \pmod{n}$ **then**
    return složené
  end if
  for $x \leftarrow 2^{s-1} \cdot t$ **to** t **do**
    if $a^x = n - 1 \pmod{n}$ **then**
      Odmocnina z 1 je $n - 1$. Zastavíme iteraci, neboť nevíme nic o odmocnině z $n - 1$.
      `break`
    else if $a^x \neq 1 \pmod{n}$ **then**
      Odmocnina z 1 není ani $n - 1$ ani 1, tedy n je určitě složené.
      return složené
    end if
    $x = x/2$
  end for
end for
Nenašli jsme žádného svědka, n je pravděpodobně prvočíslo.
return prvočíslo

Stejně jako u Fermatova testu, pokud test odpoví složené, má vždy pravdu. Ale jak moc má pravdu, když odpoví prvočíslo?

Věta: Odpoví-li Rabinův-Millerův algoritmus na číslo n prvočíslo, pak pravděpodobnost, že n je složené je nejvýše $1/4^k$ kde k je počet iterací testu.

Časová složitost testu je $O(kN^3)$ kde k je počet iterací a N je počet bitů testovaného čísla n.

Implementace aritmetických operací

V programu jsou implementovány aritmetické operace s velkými čísly v binární soustavě. Tyto operace jsou sčítání, odčítání, násobení, dělení a modulení pomocí školních algoritmů. Dále jsou podporovány operace bitový posun doleva a doprava (v konstantním čase). Poslední operací je modulární umocňování, neboli počítání $a^n \pmod m$, která využívá následujících dvou tvrzení.

Tvrzení: Pro přirozená čísla a, b a přirozené číslo $m \neq 0$ platí:

$$a \cdot b \bmod m = (a \bmod m \cdot b \bmod m) \bmod m$$

Tvrzení (Mocnění pomocí čtverců): Pro přirozené číslo n a číslo x platí:

$$x^n = \begin{cases} 1 & \text{pro } n = 0 \\ x \cdot \left(x^{\frac{n-1}{2}}\right)^2 & \text{pro } n \text{ liché} \\ \left(x^{\frac{n}{2}}\right)^2 & \text{pro } n \text{ sudé} \end{cases}$$

Algoritmus počítá $a^n \pmod m$ v čase $O(N)$ kde N je počet bitů n (aritmetické operace považuji za konstantní).

Modulární umocňování:

```
modexp (a, n, m):
if n > 0 then
  if n je liché then
    return (a· modexp(a, n − 1, m)) mod m
  else
    b = modexp(a, n/2, m)
    return b^2 mod m
  end if
else
  return 1
end if
```

Reference

[1] G. Miller. Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Sciences*, 13(3):300-317, 1976

[2] M. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128-138, 1980