Identificarea numerelor prime

Subțirică Florin-Ioan, 322CD

Facultatea de Automatică și Calculatoare
Universitatea Politehnică din București

Abstract. Scopul acestei lucrări este o comparație între doi algoritmi populari de identificare a numerelor prime - Fermat, respectiv Miller-Rabin - care reprezintă de fapt un test probabilistic pentru a determina dacă un număr este compus sau probabil prim.

Keywords: Fermat · Miller-Rabin · numere prime · numere compuse · numere întregi

1 Introducere

1.1 Descrierea problemei rezolvate

Un **număr natural** p > 1 se numește prim dacă : p | ab atunci p | a sau p | b, unde a, b sunt numere naturale. Aceasta este o proprietate esențială a numerelor prime, iar cele două definiții sunt echivalente pentru inelul ($\mathbb{Z}$, +, ·). Un număr prim are exact doi divizori pozitivi: numărul 1 și numărul în sine. Acești divizori sunt improprii. Un număr prim este deci nefactorizabil. Cel mai mic număr prim este 2; în afară de 2 toate numerele prime sunt numere impare. Există o infinitate de numere prime, fapt demonstrat de Euclid în Antichitate.

Opusul noțiunii de număr prim este cel de număr compus. Un număr compus este un număr întreg pozitiv care are cel puțin un divizor pozitiv în afară de 1 și el însuși. Prin definiție, orice număr întreg mai mare de 1 este fie număr prim, fie număr compus. Se poate scrie ca produs de numere prime mai mici ca el, fiind deci factorizabil. Este astfel un multiplu al altor numere de modul mai mic decât el, acestea putând fi chiar prime.

1.2 Exemple de aplicații practice pentru problema aleasă

// TODO corectare

1. Folosim și ne bazăm în mod constant pe numere prime pentru securitatea era cibernetică;
2. Proprietatea matematică ciudată a primelor este folosită în criptare și decriptare;
3. Ele sunt utilizate în generarea de coduri de corectare a erorilor utilizate în telecomunicații. Ei se asigură că mesajul este trimis și primit cu corecție automată;

4. Primele acționează ca bază pentru crearea algoritmilor de criptare cu cheie publică;
5. Sunt folosite pentru tabelele hash;
6. Ele sunt, de asemenea, utilizate pentru generarea de numere pseudoaleatoare;
7. Primele sunt, de asemenea, utilizate în proiectarea mașinilor cu rotor. Un număr este fie prim, fie coprim față de numărul de pe alt rotor dintr-un rotor. Acest lucru ajută la generarea ciclului complet înainte de a repeta orice poziție posibilă a rotorului;
8. Primele sunt folosite în calcul în sistemul de criptare RSA.

1.3 Specificarea soluțiilor alese

Pentru a rezolva problema dată, am ales următorii algoritmi:
// TODO Nu ai expus concret teorema lui Fermat / cea de la Miller-Rabin

1. Metoda Fermat : Dacă un anumit număr este prim, atunci această metodă returnează întotdeauna 'true'. Dacă numărul dat este compus (sau non-prim), atunci poate returna 'true' sau 'false', dar probabilitatea de a produce rezultate incorecte pentru compus este scăzută și poate fi redusă făcând mai multe iterații.

   Time complexity: O(k Log n) : De reținut că funcția putere durează O(Log n).
   Auxiliary Space: O(1) : Această metoda poate eșua chiar dacă creștem numărul de iterații (k mai mare). Există câteva numere compuse cu proprietatea că pentru fiecare a < n, mcd(a, n) = 1 și $a^{n-1} \equiv 1 \pmod{n}$. Astfel de numere se numesc numere Carmichael. Testul de primalitate al lui Fermat este adesea folosit dacă este necesară o metodă rapidă pentru filtrare, de exemplu în faza de generare a cheii a algoritmului criptografic al cheii publice RSA.

2. Miller-Rabin : Această metodă este o metodă probabilistă (asemenea metodei Fermat), dar este în general preferată în detrimentul metodei lui Fermat. Testul Miller-Rabin implementează două modificări ale funcției PSEUDO-PRIME. Prima este alegerea mai multor valori selectate aleatoriu, în detrimentul folosirii unei singure baze de bază. A doua modificare constă într-o teoremă importantă a teoriei numerelor.

   Time Complexity: O(k*logn)
   Auxiliary Space: O(1)

1.4 Specificarea criteriilor de evaluare alese pentru validarea soluțiilor

Metrici de performanță. Cele mai importante metrici la evaluarea performanței ale unui algoritm de identificare a numerelor prime sunt următoarele:

– timpului și memoria utilizată în timpul testării numerelor prime;
– acuratețea programului, deoarece acești algoritmi au șanse (destul de mici) de a produce rezultate incorecte.

Testarea algoritmilor. // TODO de completat cu detalii despre teste După implementarea efectivă a algoritmilor, aceștia sunt testați pentru a rezolva eventualele erori și se calculează valorile teoretice ale complexității timpului și memoriei folosite.
Fiecare algoritm va fi testat de mai multe ori pentru fiecare set de date, pentru a măsura o performanță medie pentru fiecare categorie și o medie generală mai precisă.
Din acestea ar trebui să putem afla și acuratețea fiecărui algoritm.

Pentru a fi mai ușor de urmărit progresul algoritmilor la fiecare rulare, voi realiza tabele, care descriu dependența timp/memorie și acuratețe, cât și grafice (dreptă de regresie) pentru dependențele din tabele.

2 Prezentarea soluțiilor

2.1 Descrierea modului în care funcționeaza algoritmii aleși

2.2 Analiza complexitații soluțiilor

2.3 Prezentarea principalelor avantaje și dezavantaje pentru soluțiile luate în considerare

3 Evaluare

3.1 Descrierea modalitații de construire a setului de teste folosite pentru validare

3.2 Specificațiile sistemului de calcul pe care am rulat testele (procesor, memorie disponibilă)

System: ASUS TUF Gaming A15 FA506IU

Hardware

– CPU: AMD Ryzen 7 4800H 2.9 GHz nominal, overclock 4.2 GHz
– GPU: NVIDIA GeForce GTX 1660 Ti Mobile 6GB GDDR6
– RAM: 16GB DDR4 @ 3200MHz
– Storage: 1TB SSD Kingston

Software

– OS: Arch Linux $x86_64$
– Kernel: $6.0.12 - arch1 - 1$
– IDE: VisualStudio Code $x86_64$, code $1.74.2 - 1$

3.3 Ilustrarea, folosind grafice/tabele, a rezultatelor evaluării soluțiilor pe setul de teste

3.4 Interpretarea, succintă, a valorilor obținute pe teste. Dacă apar valori neașteptate, încercați să oferiți o explicație

4 Concluzii

// Incearca sa pui articolele originale in care au fost descrisi algoritmii. // Ar fi bine sa pui referinte in text unde folosesti informatiile respective. // Cand ai referinte web, e bine sa pui si data accesarii (informatiile se pot schimba in timp).

References

1. https://www.geeksforgeeks.org/primality-test-set-1-introduction-and-school-method/
2. https://www.geeksforgeeks.org/primality-test-set-2-fermet-method/
3. https://www.geeksforgeeks.org/primality-test-set-3-miller-rabin/
4. https://en.wikipedia.org/wiki/Prime_number
5. https://en.wikipedia.org/wiki/Composite_number