CS 333 - Algorithm Analysis

Project Progress Report

Probabilistic Algorithm For Testing Primality[3]

İsmail Can Yagmur
dept. Computer Science
Özyeğin University
Istanbul, Turkey
can.yagmur@ozu.edu.tr

Betül Seyhan
dept. Computer Science
Özyeğin University
Istanbul, Turkey
betul.seyhan@ozu.edu.tr

Göksel Verep
dept. Computer Science
Özyeğin University
Istanbul, Turkey
goksel.verep@ozu.edu.tr

***Abstract*—From a theoretical perspective, prime numbers are building blocks of whole numbers according to number theorists. On the other hand, prime numbers have been used in a variety of crucial applications such as encryption of the data. Hence, finding prime numbers is important to discover. However, determining for a given arbitrarily large integer whether it is prime number or not is not an easy task. In this paper, instead of deterministic algorithms which need relatively large computations, practical probabilistic algorithm is presented. If the algorithm states that a given number is composite, then the result is always true, but when it states the number is a prime, there is a relatively small deterministic probability of error. The main advantage of this algorithm is that the test of primality of n requires in the worst case "$c(\log_2 n)^2$" steps where c is about 100. Thus, very large numbers are practical to use in this test.**

***Index Terms*—number theory, primality, probabilistic, encryption**

I. Introduction

Testing primality is an important algorithmic problem. Determining primality has occupied computer scientists and mathematicians for centuries. The algorithms that have been proposed for determining whether a given number is prime or composite are generally restricted to special cases or taking a lot of time and effort and especially not applicable for large numbers. In this paper, a practical probabilistic algorithm is presented for testing large numbers in an arbitrary form for primality. The result of the algorithm is always true when it asserts that a given number is composite. On the other hand, there exists provably small probability error when the algorithm asserts that a given number is prime.

Thus far various algorithms are proposed for testing primality. Some of these algorithms are deterministic and some of them are probabilistic. Although deterministic algorithms always give true results for composite and prime numbers, generally their run time complexity is not polynomial. One of the most famous and fastest deterministic algorithms is AKS primality test which is created by Agrawal, Kayal, and Saxena. AKS algorithm is a polynomial time algorithm and commonly used in cryptology. Probabilistic tests are much faster but not always accurate. Fermat primality test, Miller-Rabin primality test, ,Frobenius primality test and Baillie–PSW primality test are probabilistic tests. Miller-Rabin is the most commonly used probabilistic algorithm since tests are easy to implement and have an efficient run time complexity.

In Section II, we discuss the fundamentals of Probabilistic Algorithm for Testing Primality and the mathematical idea behind the algorithm. In Section III, we discuss the algorithm details and how the implementations of the probabilistic algorithms are done for testing large numbers primality. In Section IV, we analyze the test result according to runtime efficiency and correctness. In Section V, we summarize the main aim of the paper and list open questions and possible future work.

II. Background

A. Problem Statement

Determining whether a integer is prime or not is one of the most important problems for centuries. Nevertheless, all deterministic methods that test the primality of integers are not applicable for very big integers practically because even the new version of AKS[1] primality test,the fastest deterministic method for big arbitrary integers n, has complexity $O((logn)^6)$. Miller-Rabin probabilistic algorithm intends to test primality of big arbitrary numbers practically. The result of algorithm is always true when it states that integer is composite and it has small probability of error when it states that integer is prime. The complexity of Miller-Rabin algorithm can be pushed to $O(klog^3n)$ using repeated squaring multiplication which makes it applicable for big integers.[2].

B. Definitions and Notations

Throughout this paper,

1) (a, b) denotes the greatest common divisor(g.c.d) of the integers a,b
2) $res(a, b)$ denotes the least non negative residue of a when divided by b
3) $b|a$ denotes the fact that b divides a, i.e., $res(a, b) = 0$
4) Let n and b be integers. $W_n(b)$ denotes,
    a) $1 \leq b < n$;
    b) i) $b^{n-1} \not\equiv 1 \pmod{n}$ or
        ii) $\exists i s.t. 2^i|(n-1)$ and $1 < (b^{(n-1)/2^i} - 1, n) < n$

The denoted equation is called that such an integer b a witness to the compositeness of n
5) $c(S)$ denotes that the number of elements in the set S
6) E_n denotes the set of all c where $1 \leq c < n$, $(c, n) = 1$
7) ϕ denotes Euler's function

III. The Probabilistic Method

A. The Algorithm

Given a positive odd integer $n > 4$, an integer k is chosen by the tester to determine the desired reliability of the test. Let $n = 2^l * m + 1$, where m is odd positive integer. The following process repeated k times. A random integer b is picked in the range $[1, n-1]$. Then, $x \equiv b^m \mod n$ is computed. If x is equivalent to 1 or $n-1$, then n is said to be **"strong probable prime to base b"**. Hence, b is not **witness to the compositeness to n** so next random integer can be picked to do the next iteration of the process. Otherwise, the following process is repeated $l-1$ times. Compute $x \equiv x^2 \mod n$. If $x = 1$, n is definitely composite so terminate the process. If $x = n-1$, then n is strong probable prime to base b; therefore, next iteration of the process can be executed. If repeating $m-1$ times is executed without jumping to somewhere else, n is definitely composite number; therefore, the process can be terminated. If repeating k times is executed without returning a value, n is a prime number with a probability of error rate $1/4^k$; therefore the process is terminated.

Pseudocode

```
Require: Odd integer $n > 4$ to be tested for primality
Require: $k$, a parameter that determines the reliability
Ensure: $n = 2^l * m + 1$ where $m$ is odd integer
  LOOP:
  repeat
    Pick a random number $b$ in the range [2, n - 1]
    $x \leftarrow b^m \mod n$
    if $x = 1$ or $x = n - 1$ then
      do next LOOP
    end if
    repeat
      $x \leftarrow x^2 \mod n$
      if $x = 1$ then
        return COMPOSITE
      end if
      if $x = n - 1$ then
        do next LOOP
      end if
    until $l - 1$ times
    return COMPOSITE
  until k times
  return PRIME
```

B. The Implementation

Write here ...

...

...

...

IV. Discussion

Write here ...

...

...

...

Proof of Theorem 1

Write here ...

...

...

...

A. The Analysis of the Algorithm

Write here ...

...

...

...

B. The Analysis of the Implementation

Write here ...

...

...

...

C. Experimental Results

Write here ...

...

...

...

V. Conclusion And Future Work

A. Conclusion

As a result, our analyzes and experimental results shows that Miller-Rabin probabilistic algorithm for primality test provides faster results for very big integers with small change of error when we compare against deterministic primality test algorithms. After our analyzes and results, there are open questions about primality test algorithms:

1) Is it possible to improve Miller - Rabin algorithm to reduce the complexity $O(klog^3n)$?
2) Is it possible to improve Miller- Rabin algorithm to reduce the possibility of error ?
3) Is probabilistic algorithms or deterministic algorithms always preferable against each other or is there any break point to change our algorithm preference ?

B. Future Work

We are planning to analyze Miller-Rabin algorithm using FFT-based multiplication (Harvey-Hoeven alogrithm) which can push the running time down to $O(klogn)$.

References

[1] Hendrik W Lenstra Jr and Carl B Pomerance. “Primality testing with Gaussian periods”. In: *Journal of the European Mathematical Society* 21.4 (2019), pp. 1229–1269.

[2] *Miller–Rabin primality test*. Wikipedia. URL: en . wikipedia . org / wiki / Miller - Rabin_primality_test. Accessed 17.04.2022.

[3] Michael O Rabin. “Probabilistic algorithm for testing primality”. In: *Journal of number theory* 12.1 (1980), pp. 128–138.

Appendix

Write here ...

...

...

...