MILLER-RABIN PRIMALITY TEST ALGORITHM

Ericka Joy R. Formanes
College of Science – Bachelor of Science in Computer Science
Technological University of the Philippines, Manila, Philippines

Contents

Introduction

The Miller-Rabin primality test is a widely used probabilistic algorithm that determines whether a given number is prime or composite. It is named after its inventors, Gary L. Miller and Michael O. Rabin. This test provides a fast and efficient way to check the primality of large numbers, making it a valuable tool in number theory and cryptography.

The algorithm is based on the concept of witnesses. A witness is a number that can potentially reveal the compositeness of the tested number. The test works by selecting random witnesses and performing a series of modular exponentiations and checks. If a witness fails any of the checks, the tested number is proven to be composite. However, if all witnesses pass the checks, the number is considered probably prime, with a very high confidence level.

Unlike deterministic primality tests that guarantee absolute certainty, the Miller-Rabin test is probabilistic. This means that it can occasionally produce a false positive result, classifying a composite number as prime. However, the likelihood of such errors can be made astronomically small by using a sufficient number of random witnesses.

The Algorithm

The Miller-Rabin primality test algorithm is a probabilistic algorithm used to determine whether a given number is prime or composite. It is an efficient and widely used algorithm for primality testing. Moreover, the algorithm offers a higher probability of accuracy compared to simpler tests like Fermat's little theorem. Here is an overview of the algorithm:

Input: n (n is the number to be tested for primality)
Output: whether n is prime or not

1. Let $n - 1 = 2^s d$, where $d \epsilon N$ and $s \epsilon N$
   Choose a random integer a with $2 \leq a \leq n - 2$

2. Compute
   $$X \equiv a^d (mod n)$$
   If $X \equiv \pm 1 (mod n)$ Then end the algorithm with message “n is probably prime”.
   If s=1 then end the algorithm with message “n is definitely not prime”.
   Otherwise set r=1 and go to step 3

3. Compute
   $$X \equiv a^{2^r d} (mod n)$$
   If $x \equiv 1 (mod n)$ then end the algorithm with message “n is definitely not prime”.
   If $x \equiv -1 (mod n)$ then end the algorithm with message “n is probably prime”.
   Otherwise set r=r+1 and go to step 4

4. If r=s-1, then go to step 5, otherwise go to step 3

5. Compute
   $$X \equiv a^{2^{s-1} d} (mod n)$$
   If $x \neq -1 (mod n)$ then terminate the algorithm with “n is definitely not prime”.
   If $x \equiv -1 (mod n)$ then terminate the algorithm with “n is probably prime”.

The algorithm uses modular exponentiation to compute the values of X at each step and checks for congruence with ±1 modulo n. If the congruence conditions are satisfied or if the algorithm reaches the termination conditions, a conclusion is made about the primality of n.

Time Complexity of the Miller-Rabin Primality Test

The time complexity of the Miller-Rabin primality test depends on the number of iterations/rounds performed. The more rounds, the higher the probability of correctly identifying whether a number is prime or composite. The time complexity is typically considered as $O(k \log^3 n)$, where k is the number of iterations and n is the input number being tested for primality.

Pseudocode of the Algorithm

The Rabin-Miller primality test algorithm can be written in pseudocode as follows. The parameter k determines the accuracy of the test. The greater the number of rounds, the more accurate the result.

Input #1: n > 2, an odd integer to be tested for primality
Input #2: k, the number of rounds of testing to perform
Output: “composite” if n is found to be composite, “probably prime” otherwise

```
let s > 0 and d odd > 0 such that n − 1 = 2^s d  # by factoring out powers of 2 from n − 1
repeat k times:
   a ← random(2, n − 2)  # n is always a probable prime to base 1 and n − 1
   x ← a^d mod n
   repeat s times:
      y ← x^2 mod n
      if y = 1 and x ≠ 1 and x ≠ n − 1 then # nontrivial square root of 1 modulo n
         return “composite”
      x ← y
   if y ≠ 1 then
      return “composite”
return “probably prime”
```

Step-By-Step Numerical Example of the Algorithm

Given a number n, check if it is prime or not using Rabin – Miller primality test.

Given input: n = 17, k = 3

Step 1: Compute for d and s such that $2^s d = n - 1$.

n - 1 = 16
$2^s d \rightarrow 2^4 1 = 16$
16 = 16, TRUE
Thus, s = 4 and d = 1.

Step 2: Repeat the following steps for kth time or 3 times/rounds:

1st Iteration:
Generate a random number a between 2 and n - 2. Let's say a = 7.
Calculate $x = (a^d) \text{ mod } n = (7^1) \text{ mod } 17 = 7 \text{ mod } 17 = 7$.
Repeat s = 4 times:
Calculate $y = (x^2) \text{ mod } n = (7^2) \text{ mod } 17 = 49 \text{ mod } 17 = 15$.
Since $y = 15 \neq 1$, continue to the next iteration.
Since $y \neq 1$, proceed to the next round.

2nd Iteration:
Generate a new random number a. Let's say a = 5.
Calculate $x = (a^d) \text{ mod } n = (5^1) \text{ mod } 17 = 5 \text{ mod } 17 = 5$.
Repeat s = 4 times:
Calculate $y = (x^2) \text{ mod } n = (5^2) \text{ mod } 17 = 25 \text{ mod } 17 = 8$.
Since $y = 8 \neq 1$, continue to the next iteration.
Since $y \neq 1$, proceed to the next round.

3rd Iteration:
Generate a new random number a. Let's say a = 3.
Calculate $x = (a^d) \text{ mod } n = (3^1) \text{ mod } 17 = 3 \text{ mod } 17 = 3$.
Repeat s = 4 times:
Calculate $y = (x^2) \text{ mod } n = (3^2) \text{ mod } 17 = 9 \text{ mod } 17 = 9$.
Since $y = 9 \neq 1$, continue to the next iteration.
Since $y \neq 1$, proceed to the next round.

Since we have completed the required number of rounds and none of the conditions that n is “composite” were met, we conclude that *n* or *17 is "probably prime"* based on the Miller-Rabin primality test.

Advantages and Disadvantages

The Miller-Rabin test offers several advantages and some disadvantages as shown below:

When is it ***ADVANTAGEOUS*** to use Miller-Rabin Primality Test?

1. The algorithm can be advantageous when testing high number for primality.
2. Due to its advantage in speed when compared to other primality tests, Miller Rabin test will be the test of choice for various cryptographic applications.
3. The Miller-Rabin test finds extensive use in cryptographic protocols, such as RSA encryption.
4. The random nature of the Miller-Rabin test makes it effective in identifying most composite numbers.

When is it ***DISADVANTAGEOUS*** to use Miller-Rabin Primality Test?

1. If a deterministic primality test is required, the Miller-Rabin test falls short.
2. If the input set includes numbers that could be Carmichael numbers, additional checks or alternative primality tests should be employed to handle such cases.

ADVANTAGES OF MILLER-RABIN PRIMALITY TEST

- The Miller-Rabin test is efficient for primality testing, especially for large numbers.
- Although the Miller-Rabin test is a probabilistic algorithm, it provides a high probability of correctly identifying composite numbers.
- The algorithm is relatively straightforward to implement compared to other complex primality tests.

DISADVANTAGES OF MILLER-RABIN PRIMALITY TEST

- In spite of the fact that the Miller-Rabin test is quick, there exist composite numbers n for which this test fails for 1/4 of all bases coprime to n.
- It is a probabilistic algorithm. It can offer a high probability of accuracy, but it still has the chance to get it wrong. The accuracy is not 100% guaranteed.
- The Miller-Rabin test is not effective for identifying Carmichael numbers, which are composite numbers that pass the test for all possible bases.

In conclusion, the Miller-Rabin primality test offers advantages and limitations. It is advantageous for testing large numbers and is commonly used in cryptographic applications due to its speed. The algorithm has a high probability of correctly identifying composite numbers and is relatively easy to implement. However, it is not deterministic so it can occasionally produce false positives, making it unsuitable for certain scenarios. Additional checks or alternative tests may be needed for specific number sets, especially for Carmichael numbers. Nonetheless, the Miller-Rabin test remains an efficient and widely used method for primality testing in various fields.

Conclusion

The Miller-Rabin primality test is an efficient probabilistic algorithm used to determine if a number is prime or composite. Its advantages include speed, especially for large numbers, making it ideal for applications like RSA encryption. The test offers a high probability of correctly identifying composite numbers by utilizing random witnesses and multiple rounds of testing. It is relatively easy to implement due to its straightforward algorithmic steps. However, the test has limitations. As a probabilistic algorithm, it can occasionally produce false positives, classifying composite numbers as probably prime. Deterministic primality tests are more suitable when absolute certainty is required. Additionally, the Miller-Rabin test is ineffective for identifying Carmichael numbers, which can pass the test for all bases. To handle such cases, alternative tests or additional checks are needed.

In conclusion, the Miller-Rabin test provides efficiency, accuracy, and ease of implementation, making it a valuable tool for primality testing. However, its probabilistic nature and limitations regarding Carmichael numbers should be considered in practical applications.

References

1. Wikipedia (n.d.). Miller–Rabin primality test. Retrieved from https://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test
2. GeeksforGeeks (2022). Primality Test | Set 3 (Miller–Rabin). Retrieved from https://www.geeksforgeeks.org/primality-test-set-3-miller-rabin/
3. Keith Conrad (2011). THE MILLER–RABIN TEST. Retrieved from https://kconrad.math.uconn.edu/blurbs/ugradnumthy/millerrabin.pdf
4. Ginni (2022). What are the Miller-Rabin Algorithm for testing the primality of a given number? Retrieved from https://www.tutorialspoint.com/what-are-the-miller-rabin-algorithm-for-testing-the-primality-of-a-given-number
5. Rajesh Pavuluru (n.d.). Miller-Rabin. Retrieved from https://cs.indstate.edu/~rpavuluru/Abstract.pdf