COMPARISON STUDY OF FERMAT, SOLOVAY-STRASSEN AND MILLER-RABIN PRIMALITY TEST USING MATHEMATICA 6.0

Ega Gradini[1]

ABSTRACT

This paper presents three primality tests; Fermat test, Solovay-Strassen test, and Rabin-Miller test. Mathematica software is used to carry out the primality tests. The application of Fermat's Litle Theorem as well as Euler's Theorem on the tests was also discussed and this leads to the concept of pseudoprime. This paper is also discussed some results on pseudoprimes with certain range and do quantitative comparison. Those primality tests need to be evaluated in terms of its ability to compute as well as correctness in determining primality of given numbers. The answer to this is to create a source codes for those tests and evaluate them by using Mathematica 6.0. Those are Miller-Rabin test, Solovay-Strassen test, Fermat test and Lucas-Lehmer test. Each test was coded using an algorithm derived from number theoretic theorems and coded using the Mathematica version 6.0. Miller-Rabin test, SolovayStrassen test, and Fermat test are probabilistic tests since they cannot certainly identify the given number is prime, sometimes they fail. Using Mathematica 6.0, comparison study of primality test has been made and given the Miller- Rabin test as the most powerful test than other.

Keywords: *Primality Test, Fermat Test, Solovay-Strassen Test, Miller-Rabin Test, Prime Number*

[1] Ega Gradni, Dosen Prodi Pendidikan Matemtika – STKIP Bina Bangsa Getsempena, Jalan Tgk Chik Di Tiro, Peuniti, Banda Aceh, Telepon 0651-33427, Email: ega@stkipgetsempena.ac.id

1. CRYPTOGRAPHY AND PRIME TESTING

Prime number plays an important role in the RSA (Rivest, Shamir & Adler) cryptography. Until now, there is no valid formula to produce prime number, one of the recent technologies is to determine primality or compositeness of an integer given. Primality testing is the process to test whether or not a given number *n* is a prime. Until now, primality testing is one of the fundamental problems concerning prime numbers. It becomes more important since prime number's applications in some area, such as cryptography, detections of error in coding, and information security, especially communication and network security. Ability of primality tests has always been a centre of discussion. Those tests need to be evaluated in terms of its ability to compute as well as correctness in determining primality of given numbers. The answer to this is to create a source codes for those tests and evaluate them. These days there are many programming languages, but it requires sound understanding, and this takes up so much times just to learn how to use and get your job done. Mathematica has the solution to this problem. It is a user friendly software and do not takes up so much time to learn.

Theorem 1 Euclid Prime Number Theorem

There are infinite numbers of prime numbers [Eynden].

Proof :

Suppose that $p_1, p_2, \ldots, p_k$ are all primes. Consider the number

$$n = p_1 p_2 \ldots p_k + 1. \qquad (1)$$

If it is prime, then it is a new prime. Otherwise, it has a prime factor q. If q were one of the primes p_i for $i = 1, 2, \ldots, k$, then

$$q \mid (p_1 p_2 \ldots p_k) \qquad (2)$$

and since

$$q \mid (p_1 p_2 \ldots p_k + 1) \qquad (3)$$

q would divide the difference of these numbers, namely 1, which is impossible. So q cannot be one of the p_i for $i = 1, 2, \ldots, k$, and must therefore be a new prime (Eynden, 2001).

In this paper three primality tests were being carried out using the help of Mathematica software to assess and compare their ability. Since Mathematica very easy to learn and the command is very simple. Mathematica has so many build-in functions that can carry out many technical tasks like counting digit number, solving congruency and modulo problems, where this function is so much needed in carrying out the tests that we will see later. Mathematica also has a build-in function to count number of primes less than an integer and many more functions related to number theoretical concept. In this work there are four primality tests source code that has been designed using

Mathematica. Those are Miller-Rabin test, Solovay-Strassen test, Fermat test and Lucas-Lehmer test. Each test was coded using an algorithm derived from number theoretic theorems [Anderson] and coded using the Mathematica version 6.0. Miller-Rabin test, Solovay-Strassen test, and Fermat test are probabilistic tests since they cannot certainly identify the given number is prime, sometimes they fail. This is due to the Fermat's little theorem and Euler's theorem which does not work in both ways [Jones].

Theorem 2 Fermat Little Theorem (Mcintosh, 2007)

If n is a prime number and a is an integer, then

$$a^p \equiv a \pmod{n} \qquad (1)$$

Furthermore, if the greatest common divisors of a and n is 1, then

$$a^{(n-1)} \equiv 1 \pmod{n} \qquad (2)$$

Proof:

Consider the first $n-1$ positive multiples of a, that is, the integer $a, 2a, 3a, \ldots, (n-1)a$. None of this number is congruent modulo n to neither any other nor zero. If it happened,

$ra \equiv sa \pmod{n}$, $1 \leq r < s \leq n-1$ (1)

then a could be cancelled to gives $r \equiv s \pmod{n}$, which is impossible. Therefore, the previous set of integer must be congruent modulo n to 1, 2, 3, …, $n-1$, taken in some order. Multiplying all these congruent together, we find that

$a.2a.3a\ldots (n-1)\, a \equiv 1.2.3\ldots.(n-1) \pmod{n}$ (2)

then

$a^{(n-1)}\,(n-1)! \equiv (n-1)! \pmod{n}$ (3)

by cancelling $(n-1)!$ from both sides of the preceding congruence, this is possible since $n \nmid (n-1)!$, gives $a^{(n-1)} \equiv 1 \pmod{n}$ (Dorsey, 1999). Therefore these theorems do not work in both ways but the tests assumed that it works. That is why pseudoprime exists. The source code of each test using Mathematica 6.0 is shown in section 2, 3, and 4 respectively.

In section 5, the comparisons of the three tests are discussed, pseudoprime including carmichael number as absolute pseudoprime, Euler pseudoprime in Solovay-Strassen test, and strong pseudoprime that produced by Miller-Rabin test are also discussed, where we can see how Mathematica software has helped carrying out the tests without spending much effort on the programming task. Lastly section 6 concludes.

2. CODING FERMAT TEST USING MATHEMATICA 6.0

The following source codes are derived from an algorithm obtained mainly from

Fermat's theorem, Euler's Theorem and other related theorems from number theory to test whether or not the given number is a prime number. First, the first 100 integers set as the input. The output will come up as a prime or a composite. After this is done we can enlarge the input range to the first 1000 and until 10,000. Some build-in functions of Mathematica 6.0 is also used to carry out some technical tasks, like finding the exact number of primes less than an integer, tabulating prime numbers, pseudoprimes, strong pseudoprime and Carmichael numbers obtained from the output of the following source codes, drawing and plotting facilities are also been used to facilitate our tasks. The build-in Mathematica commands, "Prime Q[*integer]"* and "PrimePi[*integer*] are used to compare the list of primes and pseudoprimes and to compute percentage of pseudoprimes produced by each test. Due to space limitation, we are unable to show all tables and figures. To get a full picture of this project, we suggest one refers to Gradini (2009).

Fermat test is developed from Fermat Little Theorem and then become one of the probabilistic prime testing. According to Fermat's Little Theorem, if n is prime and GCD (a, n) = 1 then $a^{n-1} \equiv 1 \pmod{n}$. If n is not prime, it is not necessary true that $a^{n-1} \equiv 1 \pmod{n}$, but there still a possibility. According to Herman and Soltys (2008), all primes pass the Fermat test for all $a \in \mathbb{Z}$. Fermat's theorem also can be used to test compositeness of a number. For a given integer n, choose some integer a with GCD (a, n) = 1 and compute $r \equiv a^{n-1} \pmod{n}$. If the modulo n computation give the results not equal to 1, n is composite. Otherwise, n probably prime, in other words, n can be prime or composite.

2.1 Algorithm of Fermat Test

Input: an integer $n \geq 3$.

Output: n is prime or n is composite

1. Choose random integer a with $2 \leq a \leq n-1$ and GCD (a, n) = 1.
   - 1.1 Compute $r \equiv a^{n-1} \pmod{n}$.
   - 1.2 If $r \neq 1$, n is composite, otherwise n is prime number.
2. If GCD (a, n) $\neq 1$, then n is composite.

Here, when GCD (a, n) $\neq 1$, n is certainly composite because it is implies that n has another divisors beside 1 and itself, which is contradict with definition of prime number. If GCD (a, n) = 1, n can be composite or prime number.

2.2 Source Code of Fermat Test

By coding the Algorithm into *Mathematica* (6.0 version), here is the source code:

```
a=__;
n=__;
If [2≤a≤n-1,If [GCD [a,n]==1,
r =PowerMod [a,n-1,n];
If[r≠1,n "is composite",n "is Prime"],n "is
composite "],"cannot be proceed, pick a any
integer 2≤a≤n-1"]
```

Once this is executed, if the input is a prime number then it will tells you that ***n is a***

prime. Otherwise it will gives you a message ***n is a composite.*** Take notes, there are numbers which also identified as prime numbers even though they are not. These numbers are called pseudoprimes. This is due again to the Fermat's theorem. This source code can also identify Carmichael numbers (absolute pseudoprimes). Carmichael number is a pseudoprime for all bases *a*. From the output produced by this test, we have tabled list of primes for every base *a* which are less than 10,000. The build-in Mathematica command, "PrimeQ[*integer]"* is used to verify the primality of the test output and from here we can identify pseudoprimes, then the command, "PrimePi[10,000]", is used to compute percentage of pseudoprimes. The smaller the percentage is, the better the ability of the test is, as this indicates the accuracy of the test. To look for Carmichael numbers, we just need to look out for common pseudoprimes for each base *a*. Here, we restrict the choice of the base *a* from 2 until 20 only. For a complete output list of pseudoprimes and carmichael numbers refer to [Gradini].

3. CODING SOLOVAY-STRASSEN TEST USING MATHEMATICA 6.0

The Solovay-Strassen primality testing was the first test popularized by the advent of public key cryptography, in particular RSA cryptosystem. Solovay-Strassen test developed from Euler test combine with Jacobi symbol. Recall that Jacobi symbols $\left(\frac{a}{n}\right)$ is equivalent to the Legendre symbols when n is prime.

Definition 1 Euler's Criterion [Jones and Jones]

If n is an odd prime and GCD (a ,n) = 1, then for all integer a,

$$\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n} \qquad (1)$$

Definition 2 Euler Witness

Let n be an odd composite integer and a be integer in interval [1, n-1]. If either Greatest Common Divisor (GDD)

$$(a, n) > 1 \text{ or } \left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2} \pmod{n} \qquad (1)$$

then a is called an Euler witness for n. Otherwise, if GCD (a, n) = 1 and

$$a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \pmod{n} \qquad (2)$$

then n is said to be an Euler pseudo-prime to the base a [Bektas].

3.1 Algorithm of Solovay-Strassen Test

Input: an odd integer $n \geq 3$.

Output: n is prime or n is composite.

1. Choose a random integer a with $2 \leq a \leq n-1$ and GCD (a, n) = 1.
   - 1.1 Compute $r \equiv a^{(n-1)/2} \pmod{n}$.
   - 1.2 If $r \neq 1$ and $r \neq n-1$, then

i. Compute Jacobi symbol, $S = \left(\frac{a}{n}\right)$.

ii. If $r \not\equiv S \pmod{n}$, n is composite.

iii. Otherwise, n is prime .

1.3 Otherwise, n is prime.

2. If GCD $(a, n) \neq 1$, n is composite.

On the algorithm, n must be an odd integer, since for n is even the remainder computation cannot be done. This happens since it needs to calculate $r \equiv a^{(n-1)/2} \pmod{n}$ but when n even number, n-1 will become odd number, thus $\frac{n-1}{2}$ is not integer. For some n and a, the computation does not have integer solution. Example for this problem will be shown next. Same as Fermat test, when GCD $(a, n) \neq 1$, n is composite number.

3.2 Source code of Solovay-Strassen test

By coding the algorithm above onto *Mathematica* (Version 6.0), the source code is

```
a=__;
n=__;
If[2≤a≤n-1 ∧ OddQ[n]==True,If
[GCD[a,n]==1,
 r=PowerMod[a,(n-1)/2,n];
 If [r≠ 1∧ r≠n-1,
  s=JacobiSymbol [a,n];
If [ r≠Mod[s,n],n "is Composite",n "is Prime"],
n "is Prime"],n "is composite"],"cannot be proceed,
pick 2 ≤a≤n-1"]
```

Solovay-Strassen test is usually hard to implement since it involves Jacobi symbol computation but still Mathematica managed to compute this. Here the same things happen as the one in Fermat test. But here instead of using the Fermat theorem, we have used the Euler theorem to determine primality and it also works one way. In this test the pseudoprimes is called Euler pseudoprimes and based on the output list, carmichael number does not exist here as there is no common pseudoprime for each base a. For a complete output list of Euler pseudoprimes, refer to [Gradini].

4. CODING MILLER - RABIN TEST USING MATHEMATICA 6.0

Miller-Rabin test also known as strong pseudo-prime test, since sometimes some composite numbers passes this test. It comes as a works of G.L. Miller, based on the Riemann Hypothesis. Later, M.O. Rabin modified it in 1976.

Theorem 3 Miller-Rabin Test [Burton].
Suppose n is an odd prime. Let

$$n - 1 = 2^s r \qquad (1)$$

where r is odd and $s \geq 1$. Let a any integer with $1 < a < n - 1$, such that GCD $(a, n) = 1$.. Then satisfy either

$$a^r \equiv 1 \pmod{n} \text{ or } a^{2^j r} \equiv -1 \pmod{n} \qquad (2)$$

for some j on interval $[0, s - 1]$.

Theorem 4 Miller-Rabin –Selfridge Test [Yan].

Let p be a prime, then

$x^2 \equiv 1 \pmod{p}$ (1)

if and only if

$x \equiv \pm 1 \pmod{p}$ (2)

4.1 Algorithm of Miller-Rabin Test

Input: Integer $n \geq 3$.

Output: *n* is prime or *n* is composite.

1. If *n* is even, there are much easier test available, otherwise continue to the next step.
2. Choose *a* random integer *a* with $2 \leq a \leq n-1$ and GCD $(a, n) = 1$.
3. Write $n - 1 = 2^s r$ where *r* is odd.
4. Compute $b \equiv a^r \pmod{n}$.

4.1 If $b \neq 1$ and $b \neq n - 1$ then do the following

i. $j \leftarrow 1$

ii. while $j \leq s-1$ and $b \neq n - 1$, do the following steps

a. Compute $b \leftarrow b^2 \pmod{n}$.

b. If $b = 1$ then *n* is composite.

iii. $j \leftarrow j+1$

iv. If $b \neq n - 1$, then *n* is composite, else *n* is prime.

4.2 Else, *n* is prime.

4.2 Source Code of Miller-Rabin Test

By coding this algorithm into *Mathematica* (6.0 version), the source code is:

```
n = __;
a = __;
k = FactorInteger [n - 1]
s = k[[1, 2]];
r = (n - 1)/(2^s);
OddQ[r]==True;
If [EvenQ[n] == True ∧ GCD[a, n]≠ 1, n "is even, Another suitable test available", b = PowerMod[a, r, n];
    If[b ≠ 1 ∧ b ≠ n - 1, j = 1;
        While[j ≤ s - 1∧ b ≠ n - 1, b = Mod[b^2, n];
        If[b == 1, n "is comp"]; j++];
            If[b ≠ n - 1, n "is composite", n "is Prime"], n "prime"]]
```

Writing Miller-Rabin source code is not as easy as the last two tests. In this test pseudoprimes are called strong pseudoprimes as the only pseudoprimes exist are those of odd pseudoprimes that passes the last two tests. Here the list of pseudoprimes is much shorter, when compared to the earlier ones, that means less fake primes and this makes Miller-Rabin test a much better test then the last two. In this test, no carmichael number is produced. For a complete list of strong pseudoprimes refer to[Gradini].

5. COMPARISON BETWEEN FERMAT'S TEST, SOLOVAY-STRASSEN TEST, AND MILLER RABIN TEST.

Running through all the source codes indicated in the earlier section, we managed to make lists of prime numbers, differentiate pseudoprimes and strong pseudoprimes, as well as carmichael numbers. For pseudoprimes and carmichael numbers, we have used some build-in Mathematica functions to help us out in determining the exact number of primes, such that we can make comparison with the outputs from the source code. Most tables and figures pertaining to distributions of pseudoprimes, strong pseudoprimes and carmichael numbers are omitted due to space limitation. Comparisons between Fermat test, Solovay-Strassen test, and Miller-Rabin test using Mathematica 6.0 are as follows:

1. There are absolute pseudoprimes (carmichael numbers) in Fermat test, but not on the other tests.

2. Miller-Rabin test is better in testing primality than the other primality tests. It is because the rest often wrong in identifying integer *n*. By the output obtained, Fermat test and Solovay-Strassen test have more pseudoprimes than Miller-Rabin test. From the test output list, we have deduced that for testing prime numbers, Fermat test has the biggest number of error, while Miller-Rabin test has the smallest value of error.

3. Strong Pseudoprimes are fewer than pseudoprime produced by the Fermat and Solovay-Strassen test. Mathematica has produced the number of pseudoprimes that exist in integers $\leq$ 10,000 with base a, $2 \leq a \leq 20$. It is learnt from the output that strong pseudoprime is fewer than Euler and Fermat pseudoprime. In turn, Euler pseudoprime is fewer than Fermat pseudoprime. Refer to table 1 for the list of percentage of pseudoprimes. The smaller the percentage the better the test is.

Table 5.1: Percentage of Fermat, Euler and strong pseudoprime.

a	Pseudoprime (%)		
	Fermat	**Euler(Solovay-stras**	**Strong**
2	1.79%	1.14%	0.41%
3	1.87%	0.98%	-
4	3.82%	1.79%	0.81%
5	1.55%	0.73%	-
6	2.20%	1.38%	0.65%
7	1.22%	0.98%	-
8	5.70%	3.18%	1.06%
9	3.99%	1.79%	-
10	2.44%	1.30%	0.49%

11	2.28%	1.14%	-
12	2.69%	1.38%	0.73%
13	2.03%	1.14%	-
14	2.60%	1.79%	0.65%
15	1.55%	0.73%	-
16	5.13%	3.83%	2.93%
17	2.03%	1.14%	-
18	3.25%	1.95%	1.06%
19	3.09%	2.04%	-
20	2.60%	1.63%	0.65%

4. From the Mathematica programming, we managed to compute the error of Miller-Rabin test which is less than 1 / 4, where Fermat and Solovay-Strassen is bigger than 1 / 4. Now it can be determined that the Miller-Rabin test has the least probability of error in identifying primes $\leq 10{,}000$. Fermat and Solovay-Strassen test has the probability of error slightly greater than Millerrabin test.
5. From the figure 5.1, we have put all the three output of probabilistic tests in a chart and compare the number of prime produced by each test in the range of the first 500 integers, followed by the first 1000 integers until the first 10,000 integers. We have also incorporated the number of prime produced using the build-in Mathematica function, PrimePi[*integer]*, for each specified range, where this is considered as the exact number of primes. It appears that, the Miller-Rabin test has the nearest number of prime to the exact one produced by the build-in Mathematica command. Thus this indicates that Miller-Rabin test has the least pseudoprimes compared to the two tests, and the Solovay-Strassen test has less pseudoprimes than the pseudoprimes produced by the Fermat test. Therefore, this makes the Miller-Rabin test superior to Fermat and Solovay tests. It also appears that Solovay- Strassen test is better than Fermat test.

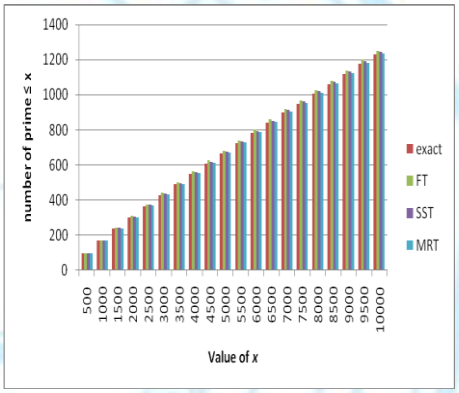

Figure 5.1 the number of prime <10,001 by Fermat, Solovay-Strassen, and Miller-Rabin test comparing to exact number prime using Mathematica 6.0.

6. CONCLUSIONS

Using Mathematica 6.0 for the primality tests, we have managed to conduct technical tasks and compare the ability of the four primality tests discussed earlier. These technical tasks were usually done using programming languages which take very long time to do the source code. For example programming language C or C++ and other

programming languages that requires deep understanding and time consuming.

Technically speaking, using Mathematica, we have managed to see that the Miller-Rabin primality test is better than Fermat and Solovay-Strassen probabilistic primality test as it produces less pseudoprimes when compare to the two of them. Now, we can actually forecast that the same trend could also be seen when a much higher level of programming language is being used to carry out these four primality tests. This is to say that, we do not need to spend so much time and effort learning programming languages that takes ages to produce the same results.

This project has actually taken an advantage of the technology offered by the Mathematica software. Without this software, we might not be able to produce such results, which is very fast in terms of producing lists of meaningful output, tabulates and interprets output data .The Mathematica source codes together with some build-in functions used in this project are suitable to be used in teaching and learning processes, especially in teaching number theory at the university level or another area that need contribution of them.

BIBLIOGRAPHY

Agrawal, Maninda & Kayal, Neeraj & Saxena, Nitin. (2003). *Prime is in P.* http://www.csc.iitk.ac.in/news/primality-v3.ps

Anderson, James A & Bell, James M.(1996). *Number Theory with application.* New Jersey: Prentice Hall

Bektas, Attila. (2005). *Probabilistic Primality Test.* Master's thesis. The middle East Technical University.

Borneman, Kristen.(2007). *Mersenne Primes and Lucas-Lehmer Test.* [on-line], http://www.mattfind.com/Lucas-Lehmer test for mersenne primes.

Burton, David M.(2002). *Elementary Number Theory*. 5th edn. New York: McGraw Hill.

Ega, Gradini, Project report of MSc in Teaching of Mathematics, Universiti Sains Malaysia, 2009.

Jones, Gareth A & Jones, Mary J. (1998). *Elementary Number Theory.* London: SpringerVerlag.

Kumandari, Ramanujachary & Romero,Christina.(1998). *Number Theory with Computer Application.* New Jersey: Prentice- Hall.

Mollin, Richard A.(1998). *Fundamental Number Theory with application.* Florida: CRC Press.

Yan,Song, Y.(2000).*Number Theory for computing.* New York: Springer-Verlag. [11] Weis stein, Eric W. (2009). *Rabin-Miller Strong Pseudoprime Test.* [online], http://mathworld.wolfram.com/RabinMillerStrong PseudoprimeTest.htm1