Comparison study for Primality testing using Mathematica

Hailiza Kamarulhaili & Ega Gradini

hailiza@cs.usm.my

School of Mathematical Sciences, Universiti Sains Malaysia, Minden 11800 Penang, MALAYSIA

Abstract: *In this paper, we present four primality testings. 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 are also discussed and this leads to the concept of pseudoprime. We also discussed some results on pseudoprimes with certain range and do quantitative comparison and in addition to this, we discussed the existence of carmichael numbers and Mersenne primes.*

1 Introduction

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.

In this project four primality tests were being carried out using the help of Mathematica software to assess and compare their ability. We find 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]. Fermat Little Theorem says that if *n* is a prime then $a^n \equiv a(\bmod n)$. Euler's Theorem also says that if *n* is a prime then $\left(a/n\right) \equiv a^{n-1/2} \bmod n$. These theorems do not guarantee the primality of *n* even if *n* satisfies the congruency. Therefore these theorems do not work in both ways but the tests assumed that it works. That is why pseudoprime exists.

Lucas-Lehmer test is a deterministic test, where the accuracy of the test cannot be argued. However, the computation time is a bit too long compared to the three tests. In Lucas-Lehmer test, we deal with Mersenne numbers and the task was to determine whether or not a given Mersenne number is a Mersenne prime. The source code of each test using Mathematica 6.0 is shown in section 2.

In section 3, the comparisons of the four 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 4 concludes.

2 Coding the primality tests 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, we set the first 100 integers 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. We show the source codes here as they are the main task for this project. We also used some build-in functions of Mathematica 6.0 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, "PrimeQ[*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 [6].

2.1 Source Code of Fermat Test

```
a=__; *( the base a, an integer greater than or equal 2 )*
n=__;*( any integer, the first 100 integers )*
If [2≤a≤n-2,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-2"]
```

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 [6].

2.2 Source code of Solovay-Strassen test

```
a=__; *( the base a, an integer greater than or equal 2 )*
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 [6].

2.3 Source Code Of Miller-Rabin Test

```
n = __;
a = __; *( the base a, an integer greater than or equal 2 )*
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 [6].

2.4 Source Code of Lucas-Lehmer Test

```
p=__;
If [PrimeQ[p]==True,
S[1]=4;
n=2^p-1;
 For [i=1,i<=p-1,i++,
   S[i+1]= Mod[(S[i])^2-2,n];]

  For [i=1,i≤ p-1,i++,Print["S[",i,"]=",S[i]]];
If[S[p-1]==0,n "is a Mersenne Prime",

 n "is Composite"],"p must be prime number"]
```

This test is a bit different from the other three tests that we have seen earlier. This test is to determine whether a given Mersenne number is a Mersenne prime. A Mersenne number is a number which is of form $2^p - 1$ for *p* integer. If the number $2^p - 1$ is a prime then *p* must also be prime [Jones]. It looks like, this test prefer an input *p* to be prime. Even so, that does not guarantee that the Mersenne number $2^p - 1$ produced from this test would also be a Mersenne prime. But anyway, we do not have to worry about that as we can input any number and the test will take care of the rest. Take notes that, we have adopted the build-in Mathematica function "PrimeQ[*p*] in the source code. The S[*i*] function in the above source code, computes the congruency task as in the Lucas- Lehmer Theorem [7].

Unlike the last three tests, this test is a deterministic test and therefore there is no pseudoprimes listed as an output of this test. In other words, this test is reliable in determining the primality of a given number, even though the scope of the prime might be smaller compared to others. The base point is, we can be 100% certain on the output of this test. However this test takes more times compared to the rest. For a complete list of Mersenne primes produced from this test, again refer to [6].

3 Comparison between Fermat's Test, Sollovay-Strassen Test, Miller Rabin Test and Lucas-Lehmer 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, Miller-Rabin test, and Lucas-Lehmer 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. Strong Pseudoprimes are fewer than pseudoprime produced by the Fermat and Solovay-Strassen test. Recall that Euler pseudoprime produced by Solovay-Strassen test and Strong pseudoprime produced by Miller-Rabin 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. Mathematica also has enable us to get distribution list of Fermat, Euler and strong pseudoprime. Refer to table 3.1 for the list of percentage of pseudoprimes. The smaller the percentage the better the test is.

Table 3.1 : Percentage of Fermat, Euler and strong pseudoprime.

a	**Pseudoprime (%)**		
	Fermat	**Euler(Solovay-strassen)**	**Strong (Miller-Rabin)**
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%

3. 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.
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 Miller-rabin test.
5. From the figure 3.1, 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.

6. Lucas-Lehmer test produced the largest prime numbers, where the other tests do not able to do that. This is supported by the work done under the GIMPS (great Internet Mersenne Prime Search). GIMPS has produced the largest Mersenne prime which is $2^{43112609}-1$, and it consists of 12978189 digits. It was calculated in 2008. Using Mathematica, we have managed to list up mersenne primes up to *p* = 1279, which is the mersenne prime $2^{1279}-1$ which consists of 392 digits. Since this test is a deterministic test, so the issue of pseudoprime and carmichael number does not exist. Therefore the surety of this test is 100%. Nevertheless this test is estimated to take longer times to compute a bigger number when compared to the other three tests.

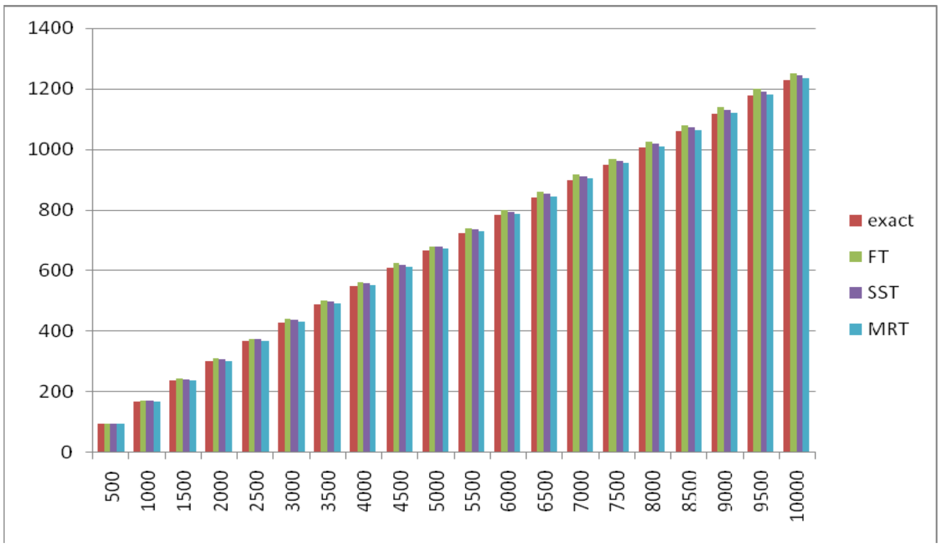

Figure 3.1 the number of prime <10,001 by Fermat, Solovay-Strassen, and Miller-Rabin test comparing to exact number prime using MATHEMATICA 6.0.

4 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. As for Lucas-Lehmer test, since it is a deterministic test, so it is 100% certain that no mistakes on determining the primality of a number, nevertheless this test is slower than the others. However, credit has to be given to contribution of Lucas-Lehmer test, since its ability to produce the largest prime numbers. 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.

Acknowledgement

This work is part of a project under Universiti Sains Malaysia RU (Research University) grant, 2009

Bibliography

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

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

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

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

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

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

[7] Jones, Gareth A & Jones, Mary J. (1998). *Elementary Number Theory*. London: Springer-Verlag.

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

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

[10] Yan,Song, Y**.**(2000).*Number Theory for computing*. New York: Springer-Verlag.

[11] Weisstein**,** Eric W. (2009). *Rabin-Miller Strong Pseudoprime Test.*[online], http://mathworld.wolfram.com/Rabin-MillerStrongPseudoprimeTest.htm1