

On Primality Testing

By

Murat TEPELİ

**A Dissertation Submitted to the
Graduate School in Partial Fulfillment of the
Requirements for the Degree of**

MASTER OF SCIENCE

**Department: Computer Engineering
Major: Computer Software**

**İzmir Institute of Technology
İzmir, Turkey**

September, 2000

We approve the thesis of Murat TEPELİ

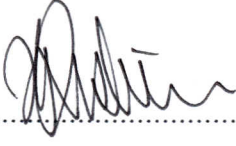


22.09.2000

Assoc.Prof.Dr.Ahmet KOLTUKSUZ

Supervisor

Department of Computer Engineering



22.09.2000

Prof.Dr.Halis PÜSKÜLCÜ

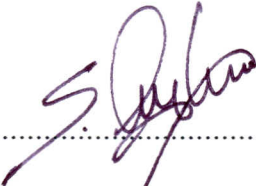
Department of Computer Engineering



22.09.2000

Asst.Prof.Dr.Tuğkan TUĞLULAR

Department of Computer Engineering



22.09.2000

Prof.Dr.Sitki AYTAÇ

Head of Department

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor *Associate Professor Ahmet KOLTUKSUZ* for his invaluable guidance, supports and continuous inspiration in every single step of my thesis.

I also would like to thank my colleagues and friends for sharing their opinions and giving valuable advises.

Last but not least, to my dear wife *Feray TEPELİ* and my family for their patience and continuous support during my long study hours.

ABSTRACT

In this study, prime numbers and primality, which is one of the most important topics in number theory is analyzed.

Subject of primality of a number has been the focus of many scientific studies and several different theories has been developed for many years. Based on these theorems, primality of large numbers has been investigated. There are also computer based algorithms to test large numbers.

In this work, five different testing methods have been studied and computer programs have been developed. Best method was determined by comparing the test results from different methods.

ÖZ

Bu çalışmada sayılar teorisinin çok önemli konularından biri olan asal sayılar ve asallık testleri incelenmiştir.

Çok eski zamanlardan beri bir tamsayının asal olup olmadığı matematikçilerin ilgisini çekmiş ve çeşitli teoremler ortaya atılmıştır. Yıllar geçtikçe bu teoremler baz alınarak büyük sayıların asal olup olmadığı incelenmiştir. Günümüzde bilgisayar yardımı ile çok büyük sayıları test eden algoritmalar mevcuttur.

Bu çalışmada, bu test sonuçlarından beş tanesi incelenmiş ve programları yazılmıştır. Test sonuçları neticesinde daha iyi performansa sahip olan metod belirlenmiştir.

TABLE OF CONTENTS

LIST OF TABLES.....	ix
LIST OF FIGURES.....	x
CHAPTER 1. DEFINITIONS AND SOME PROPERTIES OF PRIME NUMBERS.....	1
1.1 Introduction.....	1
1.2 Prime and Composite Numbers.....	1
1.3 The Number of the Prime Numbers.....	2
1.4 Primality of Number One.....	5
1.5 Greatest Common Divisor.....	6
1.6 Definition of the Order of mod n.....	6
1.7 The Fundamental Theorem of Arithmetic.....	6
1.8 Perfect Numbers.....	7
1.8.1 Theorem I.....	7
1.8.2 Theorem II.....	8
1.9 Mersenne Numbers.....	9
1.10 Relatively Prime.....	9
1.11 The Euler Φ – Function.....	10
1.12 Congruences.....	10
CHAPTER 2. SOME METHODS FOR PRIMALITY TESTING.....	12
2.1 Introduction.....	12
2.2 The Sieve of Eratosthenes.....	12
2.3 Wilson’s Method.....	12
2.4 Lucas’ Method.....	13

2.5 Proth's Method.....	14
2.6 Pepin's Method.....	16
2.6.1 Fermat Numbers.....	16
2.6.2 Pepin's Theorem.....	18
2.7 Miller-Rabin's Probabilistic Method.....	18
CHAPTER 3. DETAILS OF THE SELECTED TESTING METHODS.....	20
3.1 Introduction.	20
3.2 Special Library (M I R A C L).....	20
3.3 Selected Testing Methods.....	21
3.3.1 The Sieve of Eratosthenes.....	21
3.3.2 Wilson's Method.....	23
3.3.3 Proth's Method.....	24
3.3.4 Miller – Rabin's Method.....	24
CHAPTER 4. THE PERFORMANCE INTERPRETATION OF THE SELECTED METHODS.....	26
4.1 Introduction.	26
4.2 Selected Numbers.....	28
4.3 Discussion of the Results.....	31
CHAPTER 5. CONCLUSIONS.....	34
SUMMARY.....	35
ÖZET.....	36
BIBLIOGRAPHY.....	37
APPENDIX A. LIST OF SPECIAL FUNCTIONS(MIRACL.H).....	A1

APPENDIX B.
C CODES OF THE SELECTED PRIMALITY TESTING METHODS.. B1

APPENDIX C.
TIME RESULTS OF THE SAMPLE NUMBERS..... C1

LIST OF TABLES

Table 4.1	Largest Known Primes	27
Table 4.2	Some Relatively Small Primes	27

LIST OF FIGURES

Figure 3.1 The Sieve of Eratosthenes	22
Figure 4.1 Graphic of The Time Spent of The Sieve of Eratosthenes	32
Figure 4.2 Graphic of The Time Spent of An Improvement of The Sieve of Eratosthenes	32
Figure 4.3 Graphic of The Time Spent of The Wilson's Testing Method	32
Figure 4.4 Graphic of The Time Spent of The Proth's Testing Method	33
Figure 4.5 Graphic of The Time Spent of The Miller-Rabin's Testing Method	33

Chapter 1

DEFINITIONS AND SOME PROPERTIES OF PRIME NUMBERS

1.1 Introduction

The aim of this chapter is to explain some of the mathematical terms that will be mentioned in the next chapters and to provide the definitions and theorems that are important for primality testing.

1.2 Prime and Composite Numbers

Suppose that a, b and c are integers and $ab = c$ then it is said that; a and b are called *factors* or *divisors* of c and c is called a *multiple* of a and b . For instance the number 3 is a factor of 12 since $3 \cdot 4 = 12$; the number 12 is a multiple of 3.

If a number d is a factor of a number c , it shall be denoted by notation of $d|c$. If d is not a factor of c , then $d \nmid c$.

Now, prime and composite numbers can be defined.

A positive integer p greater than 1 is called *prime* if it has no positive factors other than 1 and p . A positive integer greater than 1 that is not a prime is called *composite*. A composite number n has positive factors other than 1 and n .

A composite number is a product of several factors, such as $15=3 \cdot 5$; or $16=2 \cdot 8$. A prime p is characterized by the fact that its only possible factorization apart from the order of the factors is $p=1 \cdot p$. Every composite number can be written as a product of *primes*, such as $16=2 \cdot 2 \cdot 2 \cdot 2$.¹

Theorem

Every composite number has got a prime factor.²

Proof

This theorem can be proved by means of induction.
Let n be a composite number. Then

$$n = n_1 n_2$$

where n_1 and n_2 are positive integers both of them are less than n . If either n_1 or n_2 is a prime, the theorem is proved. If n_1 is not prime, then

¹ Anthony J. Pettofrezzo, Donald R. Byrkit, *Elements of Number Theory*, p.23.

² *ibid*, p.24.

$$n_1 = n_3 n_4$$

where n_3 and n_4 are positive integers both of them are less than n_1 . If either n_3 or n_4 is a prime, the theorem is proved. If n_3 is not prime, then

$$n_3 = n_5 n_6$$

where n_5 and n_6 are positive integers both of them are less than n_3 . In general, after k steps the following is obtained.

$$n_{2k-1} = n_{2k+1} n_{2k+2}$$

where n_{2k+1} and n_{2k+2} are positive integers both of them are less than n_{2k-1} . Since

$$n > n_1 > n_3 > n_5 > \dots > n_{2k-1} > 0$$

for any value of k , the process have to terminate; that is, since there are only a finite number of composite integers less than n , there must exist an n_{2k-1} , for some value of k , that is a prime. Thus, it is concluded that every composite number has got a prime factor. ³

1.3 The Number of the Prime Numbers

In this section, different proofs show that there exists infinitely many primes.

Theorem

There exist infinitely many prime numbers. ⁴

Euclid's Proof

Suppose that $p_1 = 2 < p_2 = 3 < \dots < p_r$ are all the primes.

Let $P = p_1 p_2 \dots p_r + 1$ and let p be a prime dividing P ; then p cannot be any of $p_1, p_2, \dots, p_r = 1$, which is impossible. Thus this prime p is still another prime, and p_1, p_2, \dots, p_r would not be all the primes.

Euclid's proof is very simple; however, it does not give any information about the new prime, only that it is at most equal to the number P , but it may well be smaller. ⁵

³ Anthony J. Pettofrezzo, Donald R. Byrkit, *Elements of Number Theory*, p.25.

⁴ Paulo Ribenboim, *The Book of Prime Number Records*, p.3.

⁵ *ibid*, p.4.

Kummer's Proof

Suppose that there exist only finitely many primes $p_1 < p_2 < \dots < p_r$. Let $N = p_1 p_2 \dots p_r > 2$. The integer $N - 1$, being a product of primes, has a prime divisor p_i in common with N ; so, p_i divides $N - (N-1) = 1$, which is absurd! ⁶

Euler's Proof

Suppose that p is any prime, then $1/p < 1$; hence; the sum of geometric series is

$$\sum_{k=0}^{\infty} \frac{1}{p^k} = \frac{1}{1 - (1/p)}$$

Similarly, if q is another prime, then

$$\sum_{k=0}^{\infty} \frac{1}{q^k} = \frac{1}{1 - (1/q)}$$

Multiply these equalities;

$$1 + \frac{1}{p} + \frac{1}{q} + \frac{1}{p^2} + \frac{1}{pq} + \frac{1}{q^2} + \dots = \frac{1}{1 - (1/p)} \times \frac{1}{1 - (1/q)}$$

Explicitly, the left-hand side is the sum of the inverses of all natural numbers of the form $p^h q^k$ ($h \geq 0, k \geq 0$), each counted only once, because every natural number has a unique factorization as a product of primes. This simple idea is the basis of the proof. ⁷

Suppose that p_1, p_2, \dots, p_r are all the primes. For each $i = 1, \dots, n$

$$\sum_{k=0}^{\infty} \frac{1}{p_i^k} = \frac{1}{1 - (1/p_i)}$$

Multiplying these n equalities, one obtains

$$\prod_{i=1}^n \left(\sum_{k=0}^{\infty} \frac{1}{p_i^k} \right) = \prod_{i=1}^n \frac{1}{1 - (1/p_i)},$$

⁶ Paulo Ribenboim, *The Book of Prime Number Records*, p.4.

⁷ *ibid*, p.7.

..., -4, -3, -2, -1, 0, 1, 2, 3, 4, ...

According to this list integers can be divided into four groups;

- The NUMBER $\Rightarrow 0$
- The UNIT $\Rightarrow -1, 1$
- The PRIME NUMBERS $\Rightarrow -5, -3, -2, 2, 3, 5, \dots$
- The COMPOSITE NUMBERS $\Rightarrow -8, -6, -4, 4, 6, 8, \dots$

1.5 Greatest Common Divisor (gcd)

If $k|a$ and $k|b$, then k is called a **common divisor**, or **common factor**, of a and b . The largest positive integer g that divides the values of each of two integers a and b is called the **greatest common divisor** of a and b and it is denoted by (a, b) or $gcd(a, b)$.¹³

1.6 Definition of the Order of mod n (ord)

The smallest number $k > 0$ such that

$$a^k \equiv 1(\text{mod } n)$$

is called the **order of a mod n**. It is written $k = ord_n a$.¹⁴

1.7 The Fundamental Theorem of Arithmetic

An integer $n > 1$ is either a prime or can be expressed as a product of primes.¹⁵

For example; $30 = 2 \cdot 3 \cdot 5$

$$143 = 11 \cdot 13$$

¹³ Anthony J. Pettofrezzo, Donald R. Byrkit, *Elements of Number Theory*, p.34.

¹⁴ Peter Giblin, *Primes and Programming*, p.120.

¹⁵ Peter Giblin, *Primes and Programming*, p.1.

1.8 Perfect Numbers

A positive integer n is called *perfect number* if it is equal to the sum of all its positive divisors other than itself; that is $\sigma(n) = 2n$.¹⁶

For example;

$$6 \Rightarrow 1 + 2 + 3 + 6 = 6 * 2 = 12 \Rightarrow \sigma(6) = 6 * 2 = 12$$

$$28 \Rightarrow 1 + 2 + 4 + 7 + 14 + 28 = 28 * 2 \Rightarrow \sigma(28) = 28 * 2 = 56$$

496 and 8128 are the other perfect numbers.

Every number equal to $1 + 2 + 3 + \dots + 2^{n-1}$, that equals $2^n - 1$, is represented in binary notation by n consecutive 1's. Therefore,

$$P_1 = 2(2^2 - 1),$$

$$P_2 = 2^2(2^3 - 1),$$

$$P_3 = 2^4(2^5 - 1),$$

and

$$P_4 = 2^6(2^7 - 1).$$

Note that the factor of the form $2^p - 1$ in P_1, P_2, P_3, P_4 are prime numbers, like this;

$$2^2 - 1 = 3$$

$$2^3 - 1 = 7$$

$$2^5 - 1 = 31$$

$$2^7 - 1 = 127$$

1.8.1 Theorem I

An even integer is a perfect number if it is of the form $2^{p-1}(2^p - 1)$ where $2^p - 1$ is a prime.¹⁷

¹⁶ Anthony J. Pettofrezzo, Donald R. Byrkit, *Elements of Number Theory*, p.65.

¹⁷ *ibid*, p.67.

Proof I

Let $n = 2^{p-1}(2^p - 1)$ where $2^p - 1$ is a prime. Then;

$$\begin{aligned}\sigma(n) &= (1 + 2 + 2^2 + \dots + 2^{p-1})[1 + (2^p - 1)] \\ &= \frac{2^p - 1}{2 - 1} * 2^p \\ &= 2^p(2^p - 1) \\ &= 2n\end{aligned}$$

Hence, n is an even perfect number. That is an even integer is a perfect number is it is of the form $2^{p-1}(2^p - 1)$ where $2^p - 1$ is a prime. ¹⁸

For example;

$$33.550.336 = 2^{12}(2^{13} - 1) \Rightarrow 2^{13} - 1 = 8191$$

8191 is a prime number.

1.8.2 Theorem II

If a number of the form $2^p - 1$ is a prime, then p is a prime number. ¹⁹

Proof II

The equivalent statement that is p is not a prime is proved, then $2^p - 1$ is not a prime. Let $p=rs$, a composite number ($r>1$ and $s>1$). Then

$$\begin{aligned}2^p - 1 &= 2^{rs} - 1 \\ &= (2^r)^s - 1 \\ &= (2^r - 1)[(2^r)^{s-1} + (2^r)^{s-2} + \dots + 1]\end{aligned}$$

Since $r>1$, then $2^r - 1 > 1$. Thus $2^p - 1$ is a composite number. This is a contradiction, so if a number of the form $2^p - 1$ is a prime, then p is a prime number. ²⁰

¹⁸ Anthony J.Pettofrezzo, Donald R.Byrkit, *Elements of Number Theory*, p.67.

¹⁹ Kenneth H.Rosen, *Discrete Mathematics and Its Applications*, p.125.

²⁰ ibid, p.125.

1.9 Mersenne Numbers

A number of form $2^p - 1$, where p is prime, is called a *Mersenne Number* and will be denoted by M_p . The following five Mersenne numbers are some of known primes are called *Mersenne Primes*; ²¹

$$M_2, M_3, M_5, \dots, M_{4253}, M_{4423}, \dots$$

Theorem

A prime number cannot be a perfect number. ²²

Proof

Let p be any prime. Then

$$\sigma(p) = p + 1$$

If p is a perfect number, then

$$\sigma(p) = 2p \text{ and } 2p = 1 + p \Rightarrow p = 1$$

Since $p \geq 2$ then p is not a perfect number; So a prime number cannot be a perfect number. ²³

1.10 Relatively Prime

Two integers a and b , which are not both zero, are called **relatively prime** if their greatest common divisor unity; that is, if $(a, b) = 1$. For example, 22 and 15 are relatively prime integers, although neither integer is a prime.

More generally, the $n \geq 2$ integers a_1, a_2, \dots, a_n , which are not all zero, are called relatively prime if their greatest common divisor unity; that is, if $(a_1, a_2, \dots, a_n) = 1$. Furthermore, if $(a_i, a_j) = 1$ for each $i \neq j$ and $i, j = 1, 2, \dots, n$, then the $n \geq 2$ integers a_1, a_2, \dots , and a_n are called **pairwise relatively prime**.

For example, the three integers 5, 10, 13 are relatively prime since $(5, 10, 13) = 1$; however, the three integers are not pairwise relatively prime since $(5, 10) \neq 1$. ²⁴

²¹ Anthony J. Petofofrezzo, Donald R. Byrkit, *Elements of Number Theory*, p.70.

²² *ibid*, p.71.

²³ *ibid*, p.71.

²⁴ *ibid*, p.43.

1.11 The Euler Φ - Function

The number of positive integers less than or equal to a positive integer n and relatively prime to n shall be denoted by $\Phi(n)$.

For example, $\Phi(12) = 4$ since the four positive integers 1, 5, 7 and 11 are less than 12 and relatively prime to 12.

In a similar manner, it can be shown that

$\Phi(1) = 1$	$\Phi(2) = 1$	$\Phi(3) = 2$
$\Phi(4) = 2$	$\Phi(5) = 4$	$\Phi(6) = 2$
...		
$\Phi(10) = 4$	$\Phi(11) = 10$	$\Phi(12) = 4$
...		

This interesting and extremely important number-theoretic function $\Phi(n)$ is called *Euler Φ -function*. This function is sometimes also called as the *indicator of n* and sometimes called as the *totient of n* .²⁵

Theorem

If p is prime number, then

$$\Phi(p) = p - 1$$
²⁶

Proof

If p is prime number, each of the $p-1$ positive integers less than p is relatively prime to p . Now p is not relatively prime itself. Hence $\Phi(p) = p - 1$.²⁷

1.12 Congruences

The solutions of several problems – especially primality testing problems – in number theory depend on congruences. Consider a positive integer m . If a and b are two integers such that;

²⁵ Anthony J. Pettofrezzo, Donald R. Byrkit, *Elements of Number Theory*, p.71.

²⁶ *ibid*, p.73.

²⁷ *ibid*, p.73.

$m \mid (a-b)$ then a is said to be *congruent* to b modulo m ; this will be denoted $a \equiv b(\text{mod } m)$

If $m \nmid (a-b)$ then a is said to be *incongruent* to b modulo m and will be denoted by

$$a \not\equiv b(\text{mod } m) \quad ^{28}$$

For instance; $7 \equiv 2(\text{mod } 5)$, $31 \equiv -2(\text{mod } 3)$ and $16 \not\equiv 9(\text{mod } 4)$

²⁸ Kenneth H. Rosen, *Discrete Mathematics and Its Applications*, p.119.

Chapter 2

SOME METHODS FOR PRIMALITY TESTING

2.1 Introduction

There are many primality testing methods in number theory. Mathematicians have studied this topic for a long time. Currently many research activities are going on either to develop or to find a new testing method.

The aim of this thesis is not to test all of the testing methods. Some of the methods that are suitable for computers are selected. Some testing methods cannot be applied as computer software, because the number of steps of such methods cannot be determined. Hence, the following testing methods have been selected and C codes for comparing them have been developed. Some algorithms are slow, some of them are not suitable for large numbers and some of them are quite fast but probabilistic. Therefore the selected algorithms that are detailed below are the most feasible ones in terms of computer programming and hardware limitations.

2.2 The Sieve of Eratosthenes

This method is the easiest and well-known method among the primality testing methods.

Consider the following number set;

$$2, 3, 4, 5, 6, \dots, n$$

Starting first 2; The number two is the first prime because it has got no factors other than 1 and 2. Then all of the numbers those can be divided by 2 (such as 4,6,8,10,...,2x (all even numbers) is ignored.

Lets continue with the number 3. It is the second prime number because of above reason. Then all of numbers those can be divided by 3 (such as 6,9,12,15, ... 3x) is ignored.

The other numbers can be tested with the same method. Finally some numbers will stay in the list. According to sieve algorithm, these numbers are primes. ¹

2.3 Wilson's Method

This method depends on a theorem that is called *Wilson theorem*. According to this theorem;

¹ Peter Giblin, *Primes and Programming*, p.55.

If a number is prime; it means that $(n-1)! \equiv -1 \pmod{n}$.²

For instance; $n = 3 \Rightarrow n - 1 = 3 - 1 = 2$

$$\Rightarrow 2! = 2 \cdot 1 = 2$$

$$\Rightarrow 2 \equiv -1 \pmod{3} \quad \text{Hence } n=3 \text{ is prime.}$$

$$n = 13 \Rightarrow n - 1 = 13 - 1 = 12$$

$$\Rightarrow 12! = 12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \dots 3 \cdot 2 \cdot 1 = 479001600$$

$$\Rightarrow 479001600 + 1 = 479001601 \Rightarrow 479001601/13 = 36846277$$

$$\Rightarrow 479001600 \equiv -1 \pmod{13} \quad \text{Hence } n=13 \text{ is prime.}$$

$$n = 6 \Rightarrow n - 1 = 6 - 1 = 5 \Rightarrow 5! = 120 \Rightarrow 120 \not\equiv -1 \pmod{6}$$

Hence $n = 6$ is not prime.

2.4 Lucas' Method

There are three Lucas' testing methods. First and second were discovered by Lucas. The last one was discovered by Lehmer.

Test I

This testing method was discovered by Lucas in 1876. It says:

Let $n > 1$. Assume that there exists an integer $a > 1$ such that:

$$(i) \quad a^{n-1} \equiv 1 \pmod{n}$$

$$(ii) \quad a^m \not\equiv 1 \pmod{n}, \text{ for } m=1, 2, \dots, n-2.$$

Then n is a prime number.³

Test II

This testing method was discovered by Lucas in 1891. It says:

Let $n > 1$. Assume that there exists an integer $a > 1$ such that:

$$(i) \quad a^{n-1} \equiv 1 \pmod{n}$$

$$(ii) \quad a^m \not\equiv 1 \pmod{n}, \text{ for every } m < n, \text{ such that } m \text{ divides } n-1.$$

Then n is a prime number.⁴

² Evangelos Kranakis, *Primality and Cryptography*, p.41.

³ Paulo Ribenboim, *The Book of Prime Number Records*, p.37.

⁴ *ibid*, p.37.

Test III

In 1927, Lehmer made Lucas' testing method more practical. It says:

Let $n > 1$. Assume that for every prime factor q of $n-1$ there exists an integer $a > 1$ such that:

$$(i) \quad a^{n-1} \equiv 1 \pmod{n}$$

$$(ii) \quad a^{(n-1)/q} \not\equiv 1 \pmod{n}$$

Then n is a prime number. ⁵

In this theorem
 n : the number that will be tested
 a : the integer that will be fitted $a^{n-1} \equiv 1 \pmod{n}$
 q : the integer that will be fitted $q \mid (n-1)$

For instance, $n = 3$; $a = 5$; $q = 2 \Rightarrow n - 1 = 3 - 1 = 2$

$$\Rightarrow 5^2 = 25 \equiv 1 \pmod{3}$$

$$\Rightarrow (q = 2) \mid (n - 1)$$

$$\Rightarrow a^{(n-1)/q} = 5^{(3-1)/2} = 5^{2/2} = 5^1 = 5 \not\equiv 1 \pmod{3}$$

Hence 3 is prime.

Similarly, $n = 7$; $a = 4$; $q = 3 \Rightarrow n - 1 = 7 - 1 = 6$

$$\Rightarrow 4^{7-1} = 4^6 = 4096 \equiv 1 \pmod{7}$$

$$\Rightarrow (q = 3) \mid (n - 1)$$

$$\Rightarrow a^{(n-1)/q} = 4^{(7-1)/3} = 4^{6/3} = 4^2 = 16 \not\equiv 1 \pmod{7}$$

Hence 7 is prime.

2.5 Proth's Method

This method can be used to verify the primality of integers of specific forms. Proth's test is concerned with numbers of the form $k2^n + 1$. ⁶

If n is prime, according to Proth's test this lemma must become true.

⁵ Paulo Ribenboim, *The Book of Prime Number Records*, p.38.

⁶ Evangelos Kranakis, *Primality and Cryptography*, p.49.

Lemma

Let $n = ab + 1 > 1$, where $0 < a \leq b + 1$. Assume that for any prime divisor p of b there exist an integer x such that $x^{n-1} \equiv 1 \pmod{n}$ and $\gcd(x^{(n-1)/p} - 1, n) = 1$. Then n is prime.⁷

Proof

Assume on the contrary that n is not prime and let q be a prime factor of n which is $\leq \sqrt{n}$. Consider, for every prime factor p of b there exists an integer X_p such that;

$$\text{ord}_q(X_p) \mid (n-1) \quad \text{and} \quad \text{ord}_q(X_p) \nmid \frac{n-1}{p},$$

where $\text{ord}_q(X) =$ least t such that $x^t \equiv 1 \pmod{q}$. Let p^k be the largest power of prime p such that $p^k \mid b$.

Then $\text{ord}_q(X_p) = sp^k$, for some integer s . Considering the prime factorization of b and using the last idea one can find an integer x such that $\text{ord}_q(x) = b$.

It follows that $q - 1 \geq b$ and thus;

$$q^2 \geq (b+1)^2 \geq a(b+1) = ab + a \geq n$$

In particular, $q^2 = n$, $a = 1$ and $a = b + 1$; which is contradiction.⁸

The Proth Theorem can be proved by means of the lemma that is mentioned in section 2.5.1.

Proth Theorem

Assume $3k$, $k \leq 2^n + 1$ and $3 < 2^n + 1$. Then the followings are equivalent;

- i) $k2^n + 1$ is prime.
- ii) $3^{k2^{n-1}} \equiv -1 \pmod{(k2^n + 1)}$ ⁹

⁷ Evangelos Kranakis, *Primality and Cryptography*, p.50.

⁸ *ibid*, p.50.

⁹ Peter Giblin, *Primes and Programming*, p.129.

Proof

This can be proved by using the following values and the lemma that is mentioned in section 2.5.1.

$$a = k, b = 2^n + 1, x = 3. \quad {}^{10}$$

2.6 Pepin's Method

One has to examine Fermat Numbers to get a better understanding of Pepin's method.

2.6.1 Fermat Numbers

If a number can represent a special form, it can be easily tested whether it is prime or composite number.

The numbers of the form $2^m + 1$ were considered long years ago. If $2^m + 1$ is a prime, then m must be of the form $m = 2^n$, so it is a *Fermat Number*

$$F_n = 2^{2^n} + 1. \quad {}^{11}$$

The Fermat numbers;

$$F_0 = 3$$

$$F_1 = 5$$

$$F_2 = 17$$

$$F_3 = 257$$

$$F_4 = 65537$$

...

are prime numbers. Fermat believed and tried to prove all Fermat numbers are primes. Since F_5 has 10 digits, in order to test its primality, it would be necessary to have a table of primes up to 100,000 (this was unavailable to Fermat) or to derive and use some criterion for number to be a factor of Fermat number.

¹⁰ Peter Giblin, *Primes and Programming*, p.129.

¹¹ Paulo Ribenboim, *The Book of Prime Number Records*, p.71.

For $a = n \Rightarrow$ Assume that it is true for $a = n$;

Let all primes p ; $n^p \equiv n \pmod{p}$

By induction; it must be true for $a = n+1$

Let all primes p ; $(n+1)^p \equiv n+1 \pmod{p}$

Hence; the theorem is true for every natural number a .¹⁵

2.6.2 Pepin's Theorem

Let $F_n = 2^{2^n} + 1$ (with $n \geq 2$) and $k \geq 2$. Then the following conditions are equivalent.

(i) F_n is prime and $(k/F_n) = -1$

(ii) $k^{(F_n-1)/2} \equiv -1 \pmod{F_n}$ ¹⁶

2.7 Miller-Rabin Probabilistic Testing Method

In 1975, Miller proposed a primality test. To formulate Miller-Rabin test, which involves the congruences used in the definition of strong pseudoprimes. First the definition of pseudoprime will be expressed.

Pseudoprime

According to Fermat's Little Theorem;

If p is a prime number and if a is an integer, then

$$a^p \equiv a \pmod{p}. \text{ In addition;}$$

If p does not divide a , then

$$a^{p-1} \equiv 1 \pmod{p}.$$

¹⁵ Kenneth H. Rosen, *Discrete Mathematics and Its Applications*, p.145.

¹⁶ Paulo Ribenboim, *The Book of Prime Number Records*, p.71.

Chapter 3

DETAILS OF THE SELECTED TESTING METHODS

3.1 Introduction

Mathematicians have studied primality testing for years. When someone found a number and said it was a prime, they proved that by means of known theorems. These numbers were quite small and applicable for known theorems. In spite of theorems known to them, they could not try large numbers of which they suspected that it could be prime easily. That was because of the lack of technology known as computers.

However, with the aid of the computers it is now possible to try out large numbers for their primality. The goal is always to find a larger prime number than the previous one.

Although the hardware constitutes the main factor in prime number search, the software also takes its share in this effort.

The following steps are typical in prime number hunting:

- Determine how large the number would be in terms of digit size, file size etc.,
- Try to put this number in a specific form (i.e. Mersenne, Fermat etc.),
- Select the appropriate algorithm by the above criteria (deterministic/probabilistic, Wilson, Sieve, Lucas etc.),
- Decide upon the programming language (parallel languages, C, Pascal, Fortran etc.),
- Determine whether any specific library will be needed or not (Miracl, Rth etc.),
- Select a hardware (PC, Super computer, any other specific architecture like number crunchers).

3.2 Special Library (M I R A C L)

This is a shareware C library that is distributed on the Internet. MIRACL is a highly efficient and portable Multiprecision Integer and Rational Arithmetic C/C++ Library. Its main area of application is in the implementation of Public Key Cryptography systems and protocols (<ftp://ftp.compapp.dcu.ie/pub/crypto/miracl.zip>) and hence in the usage for primality testing. MIRACL library has been extensively used in this research.

3.3 Selected Testing Methods

The following primality test methods are selected and explained below:

- The Sieve of Eratosthenes,
- An improvement of the Sieve of Eratosthenes,
- Wilson's Test,
- Proth's Test,
- Miller-Rabin Test.

3.3.1 The Sieve of Eratosthenes

This method is the easiest and well-known method between the primality testing methods. Eratosthenes (276-190 B.C.E.) discovered the following algorithm, called the Sieve of Eratosthenes.¹

Start with the integers ≥ 2 in a list:

The first step is to "sieve by 2", remove all multiples of 2 besides 2 itself. The first row of crosses does this. The smallest unsieved number > 2 is now 3, and we sieve by 3, remove all multiples of 3 besides 3 itself. The smallest unsieved number > 3 is now 5 and repeat by sieving with 5. In this way, all composite numbers are sieved out and what remain are primes. Notice that in the example above, containing numbers up to 28, sieving by 2, 3, and 5 leaves only primes. In fact, the first composite that would be left is $49 = 7^2$, since any smaller number has a prime factor $\leq \lfloor \sqrt{48} \rfloor < 7$, and therefore ≤ 5 .

Sieving by 2, 3, 5 and 7 will remove all composites $< 11^2 = 121$.

The sequence of smallest numbers which remain after the succession of sievings are consecutive primes starting with 2. (This is evident from the example above, but if you want to see it formally then any induction argument is required. It is certainly true after one sieving, since 2 and 3 are the number in question. Assume that the first k sievings are by the first k primes $2, 3, \dots, p_k$ and that the smallest unsieved number $> p_k$ is p_{k+1} , the $(k+1)$ st prime. Now sieve by p_{k+1} , remove all multiples of p_{k+1} besides p_{k+1} itself. The next prime, p_{k+2} , will certainly not be sieved out, but all remaining composites between p_{k+1} and p_{k+2} will be, since they are divisible by some prime $< p_{k+2}$. This completes the induction.)

When sieving by a prime $p > 2$, the number $2p, 3p, \dots, (p-1)p$ will have gone at an earlier stage

¹ Kenneth H. Rosen, *Discrete Mathematics and Its Applications*, p.342.

3.3.2 Wilson's Method

An integer $p > 1$ is prime if and only if $(p-1)! \equiv -1 \pmod{p}$.

$$P \text{ PRIME} \Leftrightarrow (P-1)! \equiv -1 \pmod{P} \quad ^4$$

Proof

Let p is a prime number. By Fermat's theorem, the congruence;

$$x^{p-1} \equiv 1 \pmod{p},$$

$$x^{p-1} - 1 \equiv 0 \pmod{p},$$

has $p-1$ incongruent solutions $1, 2, 3, \dots, p-1$. Therefore,

$$x^{p-1} - 1 \equiv (x-1)(x-2)(x-3)\dots[x-(p-1)] \pmod{p}.$$

Now, any value of x satisfies this congruence. If x is let as 0, then

$$-1 \equiv (-1)^{p-1} \cdot 1 \cdot 3 \cdot \dots \cdot (p-1) \pmod{p};$$

that is, $(-1)^{p-1} \cdot (p-1)! + 1 \equiv 0 \pmod{p}$.

Since p is prime, p is either an odd number or equal to 2.

If p is odd, then

$$(-1)^{p-1} = 1 \text{ and}$$

$$(p-1)! + 1 \equiv 0 \pmod{p};$$

If $p = 2$, then

$$(-1)^{p-1} = -1, -1 \equiv 1 \pmod{2}, \text{ and}$$

$$(p-1)! + 1 \equiv 0 \pmod{p}.$$

Thus, $(p-1)! \equiv -1 \pmod{p}$ is valid for any prime p .⁵

⁴ Peter Giblin, *Primes and Programming*, p.102.

⁵ Anthony J. Pettofrezzo, Donald R. Byrkit, *Elements of Number Theory*, p.125.

3.3.3 Proth's Method

Proth found a new primality test method in 1878. This method is concerned with numbers of the form $k2^n + 1$. The Proth's theorem is;

Assume $3k$, $k \leq 2^n + 1$ and $3 < 2^n + 1$. Then the followings are equivalent;

- i) $k2^n + 1$ is prime.
- ii) $3^{k2^{n-1}} \equiv -1 \pmod{(k2^n + 1)}$

Proth's Theorem gives a method that is very fast at determining that a number is prime. However, it is very slow in determining that a number is composite, because it has to check every possible base b from 2 to n .

Proth's Theorem is probably a good place to begin when looking for primes within a certain range, because we can check a small number of bases b quickly for one n and then move on to another n .

If we only have one n to test, and n happens to be composite, the algorithm will run for a long time, and we might be wise to try a different algorithm.⁶

3.3.4 Miller - Rabin's Method

This probabilistic method depends on the Miller's testing method. Miller's testing method is;

Let $n-1 = 2^r m$ where $r \geq 1$ and m is odd. Let $\gcd(b, n) = 1$. If either $b^m \equiv \pm 1 \pmod{n}$ or $b^{2^k} \equiv -1 \pmod{n}$ for some $k \leq r$, then n passes Miller's Test to base b .

In addition, Miller-Rabin probabilistic testing method is;

Let n be an odd positive integer and let b_i for $i=1, \dots, k$ be such that $1 < b_i < n-1$ and $\gcd(b_i, n) = 1$. If n passes Miller's test for all bases b_i , then the probability that n is prime is at least $1-1/4^k$.⁷

Let the same example in the previous chapter.
 $N=19$ with a base $b=2$. With these values Miller's test is started.

⁶ Kenneth H. Rosen, *Discrete Mathematics and Its Applications*, p.215.

⁷ *ibid*, p.151.

Let $n-1=2^r m$ where $r \geq 1$ and m is odd. Therefore $r=1$ and $m=9$. Since $2^9 \equiv -1 \pmod{n}$, n passes Miller's test for base $b=2$, and we know that the probability that 19 is prime is at least $1-1/4=0,75$.

So with 20 succesful iterations of i from 1, ..., 20 we can be confident that a number n is prime, and also know for sure that the probability that n is not prime is less than $1/1000000000$.

THE PERFORMANCE INTERPRETATION OF THE SELECTED METHODS

4.1 Introduction

The compared performances of the selected primality testing methods will be presented in this chapter.

The selected methods are;

- The Sieve of Eratosthenes,
- An improvement of the Sieve of Eratosthenes,
- Wilson's Test,
- Proth's Test,
- Miller - Rabin Test.

The performance of these methods according to the time that they find the result in whether the number is prime or composite will be tested.

A file that is called "NUMBER.DAT" has got a number text that will be tested for primality.

When the selected methods are compared, the same number _in number.dat_ to all primality testing methods is given. Then the time results get be stored. The time may be quite different according to the primality testing method.

Another important criteria is the hardware. The specification of the hardware which is used is given below.

Processor	: Intel Celeron 400 Mhz
Cache Memory	: 128 KB
Main Memory	: 64 MB SDRAM
HardDisk Capacity	: 4.3 GB

For the present, the biggest known prime is $2^{6972593}-1$. This number has got 2098960 digits. Top ten list are shown in the following table. ¹

¹ <http://www.utm.edu/research/primes/largest.html#biggest>.

Table 4.1 Largest Known Primes by the number of digits.

Number	# Of digits	Discoverer	Year
$2^{6972593}-1$	2098960	Hajratwala, Woltman, Kurowski	1999
$2^{3021377}-1$	909526	Clarkson, Woltman, Kurowski	1998
$2^{2976221}-1$	895932	Spence, Woltman	1997
$2^{1398269}-1$	420921	Armengaud, Woltman	1996
$2^{1257787}-1$	378632	Slowinski, Gage	1996
$2^{859433}-1$	258716	Slowinski, Gage	1994
$2^{756839}-1$	227832	Slowinski, Gage	1992
$167176^{32768}+1$	171153	Yves Gallot	2000
$1697192^{557557}+1$	167847	Scott, Gallot	2000
$3026273252^{530101}+1$	159585	Nash, Dunaieff, Burrowes, Jobling,	1999

The following table shows some of prime numbers that are relatively small.

Table 4.2 Some relatively small primes. ²

# Of digits	Number
10	5915587277
20	4673257461630679457
30	590872612825179551336102196593
40	2425967623052370772757633156976982469681
50	29927402397991286489627837734179186385188296382227
60	470287785858076441566723507866751092927015824834881906763507
70	5850725702766829291491370712136286009948642125131436113342815786444567
80	34263233064835421125264776608163440537925705997962346596977803462033841059628723
90	463199005416013829210323411514132845972525641604435693287586851332821637442813833942427923
100	2074722246773485207821695222107608587480996474721117292752992589912196684750549658310084416732550077

³ <http://www.utm.edu/research/primes/small.html>

The following numbers have been selected for testing purposes. These numbers were chosen due to their number of digits. The composite numbers' were selected randomly. The prime numbers' were obtained from The University of Tennessee at Martin Prime Numbers Research Group.³

4.2 Selected Numbers

Type One

I.	7621	(4 digits)
II.	8833	(4 digits)
III.	8513	(4 digits)
IV.	10037	(5 digits)
V.	3367900313	(10 digits)
VI.	2860486317	(10 digits)
VII.	12764787846358441471	(20 digits)
VIII.	71755440315342536877	(20 digits)
IX.	590872612825179551336102196593	(30 digits)
X.	280829369862134719390036617061	(30 digits)

Type Two

I.	54987541231012456465487987654132654568741532457817
II.	48705091355238882778842909230056712140813460157899
III.	378348910233465647859184421334615532543749747185321634086219
IV.	574876541556778984542315454987666541234557898465415657795415
V.	4237080979868607742750808600846638318022863593147774739556427943294937
VI.	6545767954162063434645465467898941131546746756549951951654575793232101
VII.	1475998436180202124541047592810166939534879181170570911737412942705186 1355011151
VIII.	9865165432130546546765765403464654649056210306547598798316201654654024 6566321547
IX.	3744134716258549582697068030722592021313993868294978362774711172160447 34280924224462969371
X.	3703326004509526488023456099083350582733994873563592630385840178271946 36172568988257769603
XI.	5202642720986189087034837832337828472969800910926501361967872059486045 713145450116712488685004691423
XII.	2193992993218604310884461864618001945131790925282531768679169054389241 527895222169476723691605898519

The lengths of the type two numbers are as follows:

I.	50 digits	II.	50 digits
III.	60 digits	IV.	60 digits
V.	70 digits	VI.	70 digits
VII.	80 digits	VIII.	80 digits
IX.	90 digits	X.	90 digits
XI.	100 digits	XII.	100 digits

³ <http://www.utm.edu/research/primes/index.html>

Type Three

- I. 34313513654346313143111532341337644560245602432601424365424506324276
13245414305321566143465613146301314346564345651345613345656461316313
14633101632245176324604256575438132761483765894320817890960947594059
78045987846643814322071352140762154438032650342478930483274621843281
33207237632210632106312043211433211345326113324653274891537290708932
168932403213220606244652133

- II. 34510334464303246246364626445176768466789137806748646132103142153213
45431351431345313156511613451135031341534613346134365547337433133074
57345462454766244632116323163236032456424365463241345218352789107813
31530221078361307310893494647132143261362163261234506324066246634665
42664246516620424613266264264798267929284601249062496529864294062490
69254924860924466254623643613406537816589475989476984946654637649645
60696100319301436035674462604247156374964846394632189508965531805636
48624782307540768943403031I300176362476254613242517302486435073313013
24651023451636234662436432460123460134166437146465480385943015380322
60225186544568655246683506815328315083163254833562841436748943694336
49335486531732417341423652332036536483589433789133906394363694033653
407334473247643247432454732420978236085972308596230987632285978976283
76324765327654326578932658932658932658932658932765832492604268092469

- III. 2434343213164364130303442346542636543453345433453315335645546635617335
1673350670536674353664833546862556840625486027462454766244732415764504
7436860554696578504093453602144325470322417532421730352467365471306785
7296776604624974982077924647624547465207624570462454765240474321703421
7302401746524724677246577465277624546365160781904466540606436604083533
2681327502450176526548953028435260183258016526504865265083258162658425
6654825364802353246246364626445176768466789137806748646132103142153213
4543135143134531315651161345113503134153461334613436554733743313307457
3454624547662446321163231632360324564243654632413452183527891078133153
0221078361307310893494647132143261362163261234506324066246634665426642
4651662042461326626426479826792928460124906249652986429406249069254924
8609244662546236436134065378165894759894769849466546376496456069610031
9301436035674462604247156374964846394632189508965531805636486247823075
40768943403031I3001763624762546132425173024864350733130132465102345163
6234662436432460123460134166437146465480385943015380322602251865445686
5524668350681532831508316325483356284143674894369433649335486531732417
3414236523320365364835894337891339063943636940336534073344732476432474
3245473242097823608597230859623098763228597897628376324765327654326578
932658932658932658932658932765832492604268092467

- IV. 2434343213164364130303442346542636543453345433453315335645546635853356
1733516733506705366743536648335468625568406254860274624547662447324157
6558383450274474368605546965785040934536021443254703224175324217303524
6736547138556830678574527472296776604624974982077924647624547465207624

5378164275589475989476984946654637649645606961003193014319664964796543
6856603567446260475425742471563749648463946321895089655318056361648624
782307945407685689433640303147532475001763262476254613242517302486436
5073313013246511023451258663623466243643752744752460123460134166437146
4653646245432645462341966451676245694547465577446254746215457652454765
2475480385947630153803226022515198654456865524809668350681532831508316
3254833562841436748094369433649335486531173241734142365238332036536483
5894337891339063943636940033653407334473247643924743245473242097822537
3608597230859623098763225546965785040934536021443254703224175324217303
5246736547138556830678574527472296776604624974982077924647624547465207
6245704624547652402854743281703421730427227440174652472467724657746527
7624546365160781904466540583606436660408353326812473224775024501765265
4895302843526018325801652650486538526508360325816265842566548272536247
4802353246246364626445176768466789137805674864639366132103142153213454
3135143134574531315651161345113503134153461333584613436554579847337433
1330745734546247689547662447263211632316323603245642463583654633446241
3457894218352789107813315302210783617953073102424789349464713214358261
3647654342163261232724450632406624663466542664246516620447962461326626
2474264798386542632792928460124257490624965298642940624906925492486092
4466267945462362475264583132145743859789746283763247653276543261957893
2658932658932658923843436465434965893276583265660649260426809246321646
4654651634357476576576765767616511494165743576576574654006656765764698
7987654967986547132478931

The lengths of the type two numbers are as follows:

- I. 368 digits
- II. 887 digits
- III. 1310 digits
- IV. 1725 digits
- V. 3389 digits

4.3 Discussion of the Results

- For type one numbers generally the best method in terms of time spent is Miller-Rabin. However, one should notice that when numbers get larger Wilson test starts to fail due to the limitations of MIRACL library which does not support the factorial higher than 80000.
- For types two and three numbers the above findings are the same. But in these numbers one peculiar aspect is the time spent to get the results that tends to grow rapidly when deterministic methods used namely the Sieve of Eratosthenes and Proth. Therefore, for quick results it seems that the probabilistic method is best.

- The comparative time vs. the number of digits for prime numbers graphs of each utilized method are given in figures 4.1 – 4.5. Time spent for the efficiency of primality testing of each method can be seen directly from below graphs.

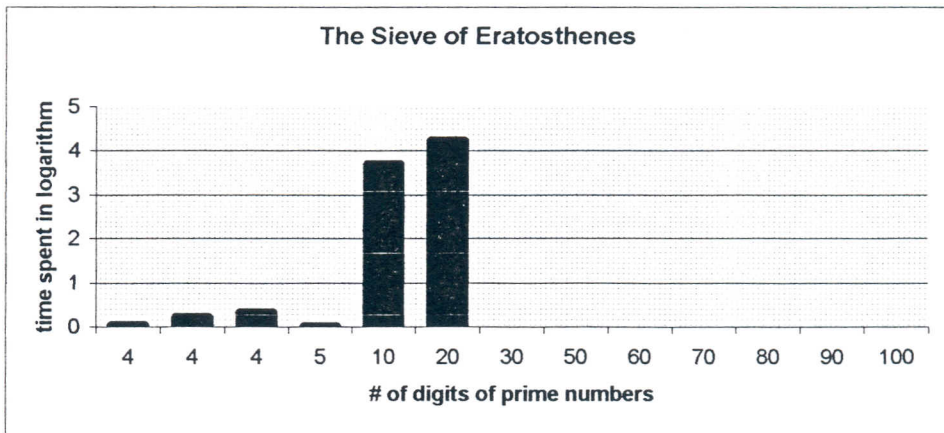


Figure 4.1 The Sieve of Eratosthenes.

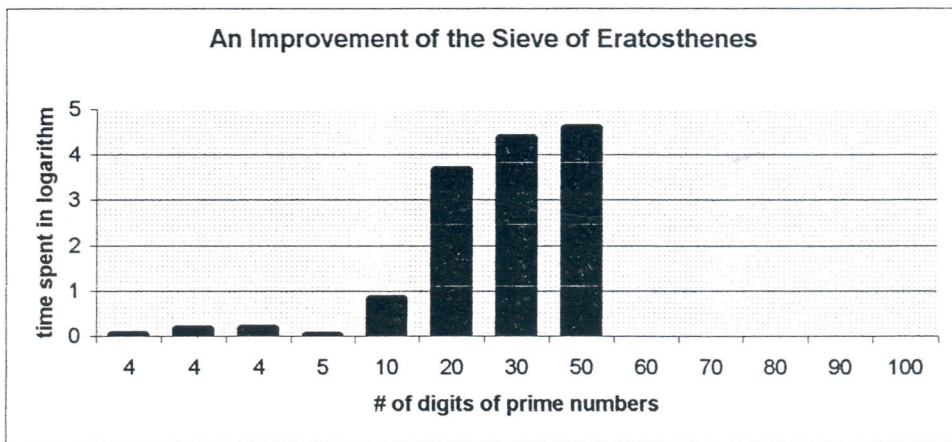


Figure 4.2 An Improvement of the Sieve of Eratosthenes.

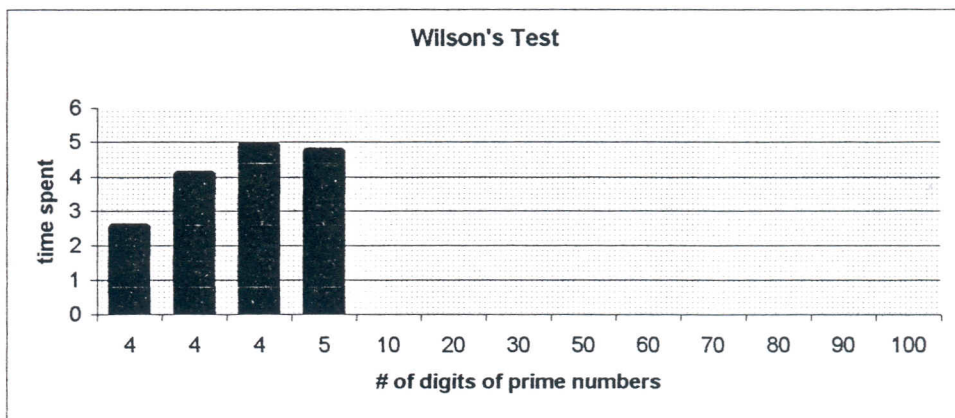


Figure 4.3 Wilson's Testing Method.

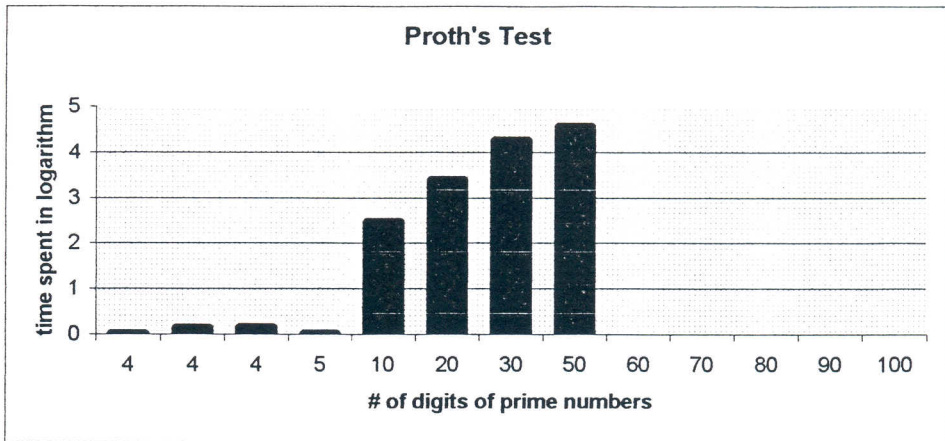


Figure 4.4 Proth's Testing Method.

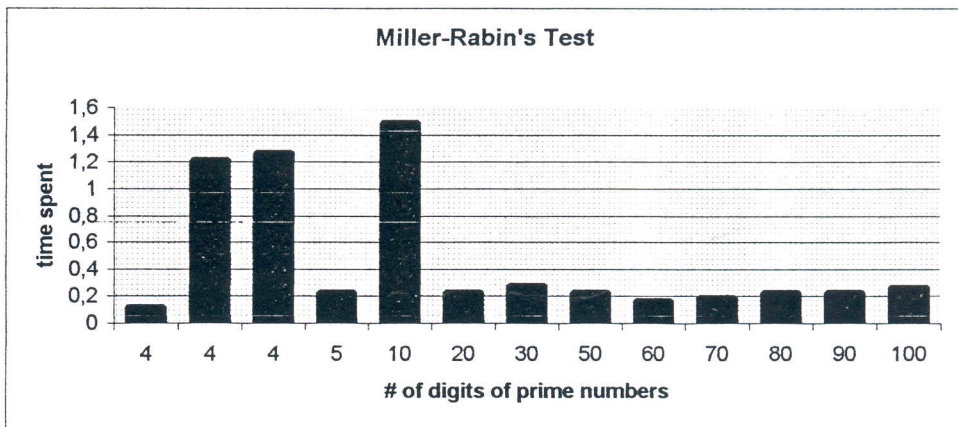


Figure 4.5 Miller-Rabin's Testing Method.

Chapter 5

CONCLUSIONS

In this study, some selected primality testing methods were examined and the best method selected.

Some of these testing methods are applied to programming easily. But sometimes one cannot convert these mathematical terms to a computer software when the big numbers are concerned. In order to test a big number, it is necessary to use some special functions or libraries. For the present work a special C library for big integer operations used.

By looking at the results, deterministic methods are slow. In contrast with that, probabilistic methods are fast.

In future work, one will want to create a special library that supports parallel operations for big numbers. By using this library, new methods can be utilized for primality testing.

SUMMARY

In this study, prime numbers and primality testing methods have been expressed and five of these methods were examined. In addition of the mathematical theorems and proofs of the primality testing methods, computer programs were developed.

Thru the developed software, some selected numbers were tested and results were compared. It is found that the probabilistic testing methods are superior to deterministic ones in terms of time spent in primality testing.

ÖZET

Bu çalışmada asal sayılar ve asallık test metotları üzerinde durulmuş, bu metotlardan beş adedi incelenmiştir. Bu test metotlarının matematiksel teoremlerinin ve ispatlarının yanında, bilgisayar programları yazılmıştır.

Geliştirilen programlar sayesinde bazı seçilmiş sayılar test edilmiş ve sonuçları karşılaştırılmıştır. Asallık testi sırasında harcanan zaman açısından, olasılıksal test metotlarının deterministik metotlardan daha üstün olduğu bulunmuştur.

BIBLIOGRAPHY

DUDLEY, Underwood, "*Elementary Number Theory*", W.H.Freeman and Company, San Francisco, 1969.

GIBLIN, Peter, "*Primes and Programming An Introduction to Number Theory with Computing*", Cambridge University Press, Great Britain, 1993.

KRANAKIS, Evangelos, "*Primality and Cryptography*", Jhon Willey & Sons Ltd., Great Britain, 1986.

PETTOFREZZO, Anthony J., BYRKIT, Donald R., "*Elements of Number Theory*", Prentice Hall Inc., USA, 1970.

RIBENBOIM, Paulo, "*The Book of Prime Number Records*", Second Edition, Springer – Verlag New York Inc., USA, 1989.

RIBENBOIM, Paulo, "*The Little Book of Big Primes*", Springer – Verlag New York Inc., USA, 1991.

ROSEN, Kenneth H., "*Discrete Mathematics and Its Applications*", Third Edition, McGraw – Hill Inc. International Editions Mathematics Series, Singapore, 1998.

SHAMUS, Software Ltd., "*MIRACL C Library*", Version 4.2, Shamus Software Ltd., (<ftp://ftp.compapp.dcu.ie/pub/crypto/miracl.zip>), Dublin Ireland, 1999.

APPENDIX A

LIST OF SPECIAL FUNCTIONS (M I R A C L . H)

- **big** : A new data type in miracl.h library.
- **cinstr** : Input a number from a character string.
Function :

```
int  cinstr(x, s)
big  x;
char *s;
```
- **otnum** : Output a big number to the screen or to a file
Function :

```
int  otnum(x, f)
big  x;
FILE *f;
```
- **compare** : Compare two big numbers.
Function :

```
int  compare(x, y)
big  x, y;
```
- **fmodulo** : Find the remainder when one number is divided by another.
Function :

```
void fmodulo(x, y, z)
big  x, y, z;
```
- **add** : Adds two big numbers.
Function :

```
void add(x, y, z)
big  x, y, z;
```
- **mirsys** : Initialize the MIRACL number of digits and base
Function :

```
miracl *mip=mirsys(500,10);
```
- **mirvar** : Initialises a big/flash variable by reserving a suitable number of memory locations for it.
Function :

```
flash mirvar(iv)
int  iv;
```
- **premult** : Multiplies a big number by an integer
Function :

```
void premult(x, n, z)
int  n
big  x, z;
```

- `nroot` : Extracts lower approximation to a root of a big number.

Function :

```
BOOL nroot(x, n, z)
big x, z;
int n;
```


APPENDIX B

C CODES

B.1 The Sieve of Eratosthenes

```
/*
 *   THE SIEVE OF ERATOSTHENES
 *
 *   MURAT TEPELİ
 */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <miracl.h> /* include MIRACL system */
FILE *dat;
int main()
{
    time_t t1, t2;
    big nf;
    int n;
    unsigned long i=0;
    big nx;
    big n0;
    big n1;
    big fraction;
    char *s;
    char c;
    mirsys(290000,10); /* 100000 digits per "big" */
    nf=mirvar(1); /* initialise "big" variable
nf=1 */
    nx=mirvar(2);
    n0=mirvar(0);
    n1=mirvar(1);
    fraction=mirvar(0);
    printf("\tsieve prime number test program\n");
    printf("\tthe number to be tested (number.dat)=");

    s=malloc(300000);
    if(s==NULL){ printf("not enough memory\n");
exit(1); }
    if( (dat=fopen("number.dat","r"))!=NULL) {
        while( !feof(dat) ) {
            s[i++]=fgetc(dat);
        }
        fclose(dat);
    } else { printf("\tno input file\n"); exit(1);}
    cinstr(nf, s);
    otnum(nf,stdout);
}
```

```

printf("\ttesting primality...");
while(compare(nf,nx)){
    fmodulo(nf, nx, fraction);
    if(compare(fraction, n0)==0) {
        printf("\r\tSIEVE: this is NOT prime!
\n");
        return;
    }
    add(nx, n1, nx);
}
printf("\r\tSIEVE: the given number is prime
\n");
return 0;
}

```

B.2 The Sieve of Eratosthenes - II

```

/*
 *   AN IMPROVEMENT OF THE SIEVE OF ERATOSTHENES
 *
 *   MURAT TEPELİ
 */
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <string.h>
#include <miracl.h> /* include MIRACL system */
FILE *dat;
int main()
{
    time_t t1, t2;
    big nf;
    int n;
    unsigned long i=0;
    big nx;
    big n0;
    big n1;
    big nr;
    big npr;
    big fraction;
    char *s;
    char c;
    mirsys(290000,10); /* 100000 digits per "big" */
    nf=mirvar(1); /* initialise "big" variable
nf=1 */
    nx=mirvar(2);
    n0=mirvar(0);
    n1=mirvar(1);
    nr=mirvar(1);
    npr=mirvar(3);
    fraction=mirvar(0);

```

```

printf("\tsieve prime number test program\n");
printf("\tthe number to be tested (number.dat)=");
s=malloc(300000);
if(s==NULL){ printf("not enough memory\n");
exit(1); }
if( (dat=fopen("number.dat","r"))!=NULL){
while( !feof(dat) ) {
s[i++]=fgetc(dat);
}
fclose(dat);
} else { printf("\tno input file\n"); exit(1);}
cinstr(nf, s);
otnum(nf, stdout);
printf("\ttesting primality...");
nroot(nf, 2 ,nr);
while(compare(nr,nx)){
fmodulo(nf, nx, fraction);
if(compare(fraction, n0)==0) {
printf("\r\tSIEVE: this is NOT prime!
\n");
return;
}
add(nx, n1, nx);
}
printf("\r\tSIEVE: the given number is prime
\n");
return 0;
}

```

B.3 Wilson 's Method

```

/*
* WILSON 'S TESTING METHOD
*
* MURAT TEPELİ
*/
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <miracl.h> /* include MIRACL system */
FILE *dat;
int main()
{
big nf;
int n, i;
big nx;
big n1;
char *s;
mirsys(290000,10);
nf=mirvar(1); /* initialize "big" variable
nf=1 */

```

```

    nf=mirvar(1);      /* initialize "big" variable
nf=1 */
    nx=mirvar(5);
    sx=mirvar(1);
    n1=mirvar(1);
    n2=mirvar(2);
    nu=mirvar(1);
    nm=mirvar(1);
    printf("\tProth's prime number test program\n");
    s=malloc(300000);
    if(s==NULL){ printf("not enough memory\n");
exit(1); }
    if( (dat=fopen("number.dat","r"))!=NULL){
        while( !feof(dat) ) {

            s[i++]=fgetc(dat);
        }
        fclose(dat);
    } else { printf("no input file\n"); exit(1);}
    cinstr(nx,s);
    subtract(nx,n1,nf);
        printf("\tthe number to be tested=");
    otnum(nx, stdout);
    printf("\ttesting primality...");
    for(i=2; i<32767 ; i++){
        convert(i, sx);
        if(compare(sx, nf)>=0) { printf("\r\tthis is NOT
prime\n"); break;}
        if(subdiv(nf,i,nu)==0) break;
    }
    gprime(LIMIT);
    for (k=1; k<LIMIT;k++)
    {
        convert(k, sx);
        if(compare(sx, nu)>=0) { printf("\r\tno prime
divisor!\n"); break;}
        if(subdiv(nu, mip->PRIMES[k], nm)==0) break;
    }

    powmod(n2,nf,nx,nu);
    if(compare(nu,n1)==0) {

        subdiv(nf, mip->PRIMES[k], nu);
        expint(2, size(nu), nm);
        subtract(nm,n1,nx); // nm--;
        egcd(sx, nx, nu);
        if(compare(nu,n1)==0) printf("\r\tPROTH: this
is prime\n");

        else printf("\r\tPROTH: this is not prime\n");
    }
}

```



```

    else printf("\r\tPROTH: this is not prime\n");
    return 0;
}

```

B.5 Miller-Rabin's Method

```

/*
 * MILLER-RABIN 'S TESTING METHOD
 *
 * MURAT TEPELİ
 */
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <miracl.h> /* include MIRACL system */
FILE *dat;
int main()
{
    time_t t1, t2;
    big nf;
    int n, i=0;
    big nx;
    big n1;
    char *s;
    time(&t1);
    mirsys(290000,10);
    nf=mirvar(1); /* initialize "big" variable
nf=1 */
    nx=mirvar(5);
    n1=mirvar(1);
    printf("\tPrime number test program\n");
    s=malloc(300000);
    if(s==NULL){ printf("not enough memory\n");
exit(1); }
    if( (dat=fopen("number.dat","r"))!=NULL) {
        while( !feof(dat) ) {
            s[i++]=fgetc(dat);
        }
        fclose(dat);
    } else { printf("no input file\n"); exit(1);}

    cinstr(nx,s);
    printf("\tthe number to be tested (number.dat)=");
otnum(nx, stdout);
    printf("\ttesting primality...");
    if(isprime(nx)) printf("\r\tthis number is
prime\n"); else printf("\r\tNOT prime!\n");
    exit(1);
    while (n>1) premult(nf,n--,nf); /* nf=(n-1)!=(n-
1)*(n-2)*....3*2*1 */
    fmodulo(nf, nx, nf);
}

```

```
    subtract(nx, n1, nx);  
    if(compare(nx, nf)==0) printf("WILLSON: this is a  
prime number\n");  
    else printf("WILLSON: this is not prime\n");  
    return 0;  
}
```

APPENDIX C

TIME RESULTS OF THE SAMPLE NUMBERS

