

Implementation and Analysis of Different Primality Testing Algorithms

By

Sriharitha Kuchipudi

Agenda

- Introduction
- Applications
- Primality test Algorithm
- Example
- Analysis
- Conclusion

Introduction

- What is prime?

The number which has exactly two distinct natural number divisors: 1 and itself.

- What is primality?

The property of being prime is primality.

- Primality Test:

- The process of proving a number is prime.
- Some prove number is prime.
- Some prove number is composite.

Applications of Prime

- Building blocks of the positive integers
- In cryptography
- Hash Functions
- Cicada

Primality Tests

- Mille-Rabin Primality test
- Solovay-Strassen Primality test
- Lucas-Lehmer Primality test
- Lucas Primality test
- Proth's Test
- Gordon's algorithm
- Maurer's algorithm for generating provable primes.

Miller-Rabin Primality Test

MILLER-RABIN(n, t)

INPUT: an odd integer $n \geq 3$ and $t \geq 1$

OUTPUT: an answer “prime” or “composite” .

- Write $n-1 = 2^s \cdot r$ such that ‘ r ’ is odd.
- For i from 1 to t do the following:
 - Choose a random integer a , $2 \leq a \leq n-2$.
 - Compute $y = a^r \bmod n$.
 - If $y \neq 1$ and $y \neq n-1$ then do the following:
 - $j \leftarrow 1$.
 - While $j \leq s-1$ and $y \neq n-1$ do the following:
 - Compute $y \leftarrow y^2 \bmod n$.
 - If $y = 1$ then return(“composite”).
 - $j \leftarrow j+1$.
 - If $y \neq n-1$ then return(“composite”).
- Return(“prime”).

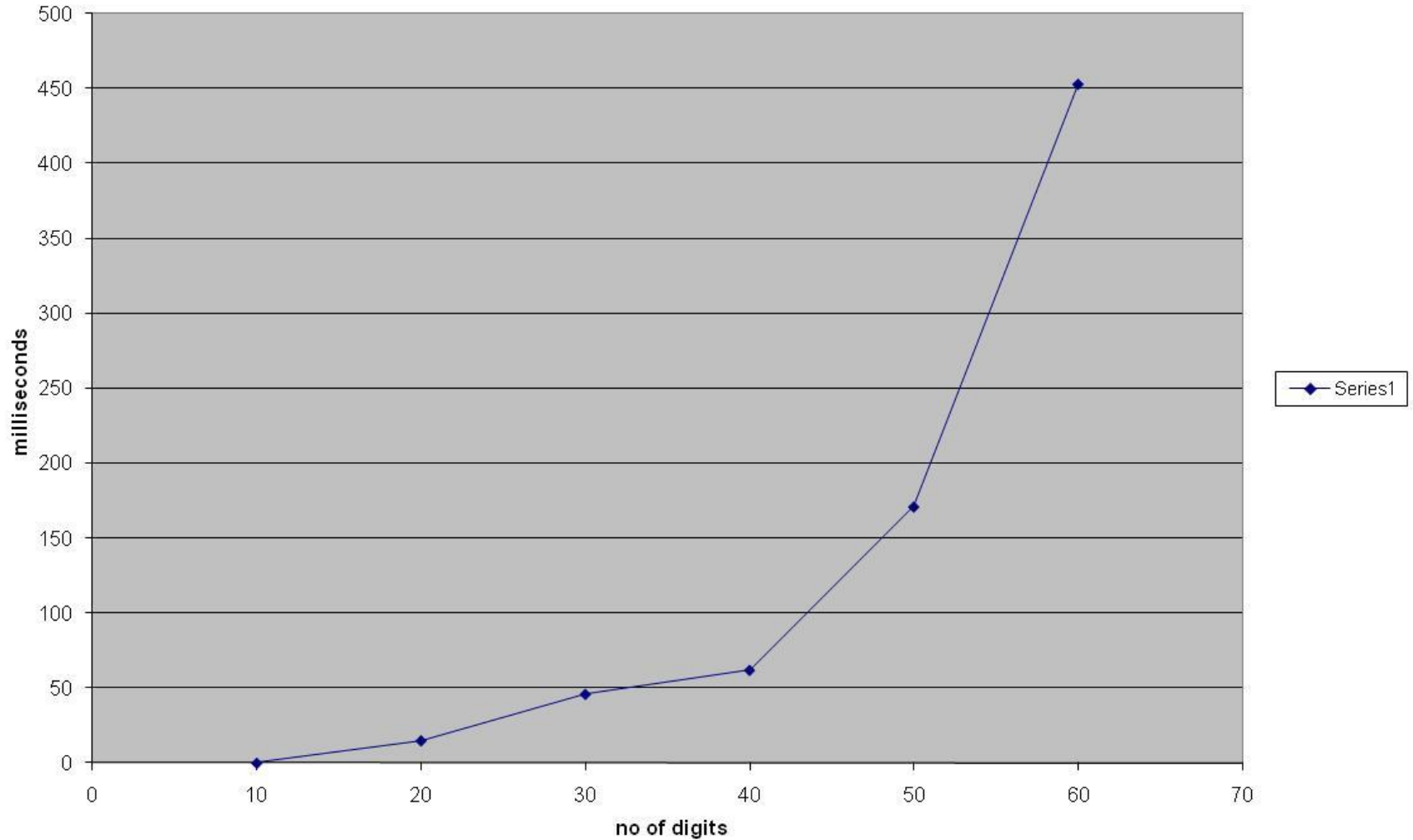
Example

Suppose we wish to determine if $n = 11$ is prime.

- Then $n-1 = 11-1 = 10$.
- $10 = 2 \cdot 5$ where it is of the form $2^s \cdot r$ $s=1$ and $r=5$
- Now we can choose a random number 'a' between 2 and 9.
- Let's take $a=3$ then $y = 3^5 \bmod 11$
- $y=1$. As $y \neq 1$ and $y \neq n-1$ so the given number 'n' is not composite.

Miller-Rabin graph

Miller-Rabin primality test



Solovay-Strassen Primality Test

SOLOVAY-STRASSEN (n,t)

INPUT: an odd integer $n \geq 3$ and $t \geq 1$.

OUTPUT: an answer “prime” or “composite” to the question: “Is n prime?”

- For i from 1 to t do the following:
 - 1.1 Choose a random integer a, $2 \leq a \leq n-2$.
 - 1.2 Compute $r = a^{(n-1)/2} \bmod n$
 - 1.3 If $r \neq 1$ and $r \neq n-1$ then return(“composite”).
 - 1.4 Compute the Jacobi symbol $s = (a/n)$
 - 1.5 If $r \neq s \pmod n$ then return (“composite”).
- Return(“prime”).

Jacobi symbol

- The **Jacobi symbol** is a generalization of the Legendre symbol.
- Let p be an odd prime and a an integer. The Legendre symbol (a/p) is defined as:

$$(a/p) = \begin{cases} 0, & \text{if } a \equiv 0 \pmod{p} \\ 1, & \text{if } a \not\equiv 0 \pmod{p} \text{ and for some integer } x \\ & a \equiv x^2 \pmod{p} \\ -1, & \text{if there is no such } x \end{cases}$$

- Let $n \geq 3$ be odd with prime factorization

$$n = p_1^{e_1} * p_2^{e_2} * \dots$$

Then the Jacobi symbol (a/n) is defined to be

$$(a/n) = (a/p_1)^{e_1} * (a/p_2)^{e_2} * \dots$$

- Observe that if n is prime, then the Jacobi symbol is just the Legendre symbol.

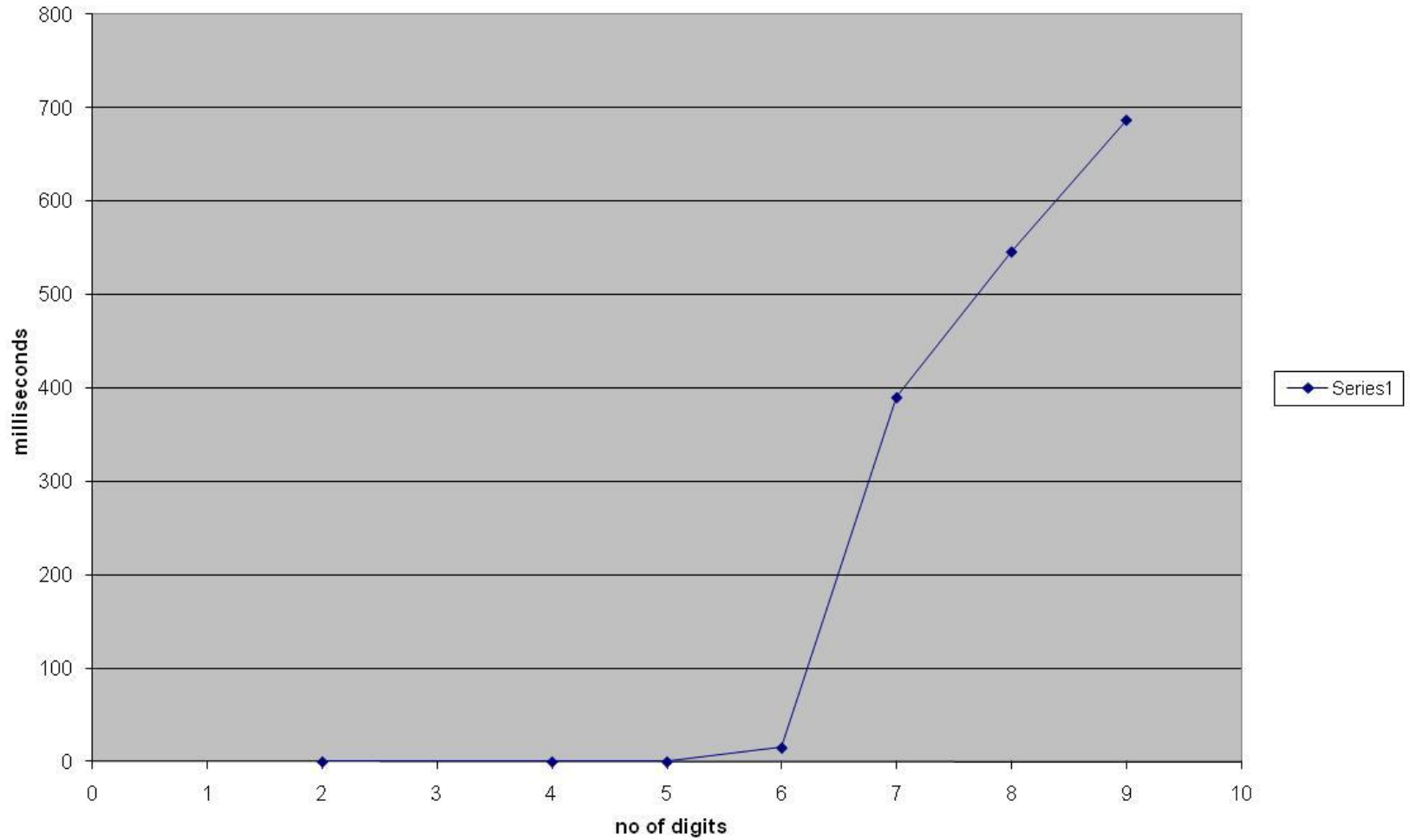
Example

Suppose we wish to determine if $n = 7$ is prime or composite.

- Then $n-1 = 6$. Let $a = 4$.
- Compute $r = a^{(n-1)/2} \bmod 7 = 4^3 \bmod 7 = 1$.
- As $r = 1$ it is not composite .
- Jacobi symbol $s = 1$
- $r = s$ it is prime.

Solovay-Strassen graph

Solovay-Strassen primality test



Proth's Test

- Proth's theorem is a primality test for proth numbers
- If a number is of the form $k \cdot 2^n + 1$, where k is odd and n is positive integer then it is a proth number.
- It states that if 'p' is a proth number of the form $k \cdot 2^n + 1$ with k odd and $k < 2^n$ then if for some integer a ,

$$a^{(p-1)/2} \equiv -1 \pmod{p}$$

Then 'p' is called proth prime.

Example

- For $p = 13$, where 13 is a proth number.
- That is $13 = 2^2 * 3 + 1$ where $n=2$ and $k=3$
- For $a = 5$, $a^{(p-1)/2} = 15625$
- $5^6 + 1 = 15626$ is divisible by 13, so 13 is prime

Lucas Test

- Lucas test is a primality test for a natural number ‘n’ and it requires prime factors of $n-1$.

Input: $n > 2$, an odd integer to be tested for primality; k ,

Output: *prime* if n is prime, otherwise *composite* or *possibly*

- determine the prime factors of $n-1$.

- LOOP1: repeat k times:

pick a randomly in the range $[2, n - 1]$ if $a^{n-1} \neq 1 \pmod{n}$ then return *composite* otherwise

- LOOP2: for all prime factors q of $n-1$:

if $a^{(n-1)/q} \neq 1 \pmod{n}$

if we did not check this equality for all prime factors of $n-1$ then do next LOOP2

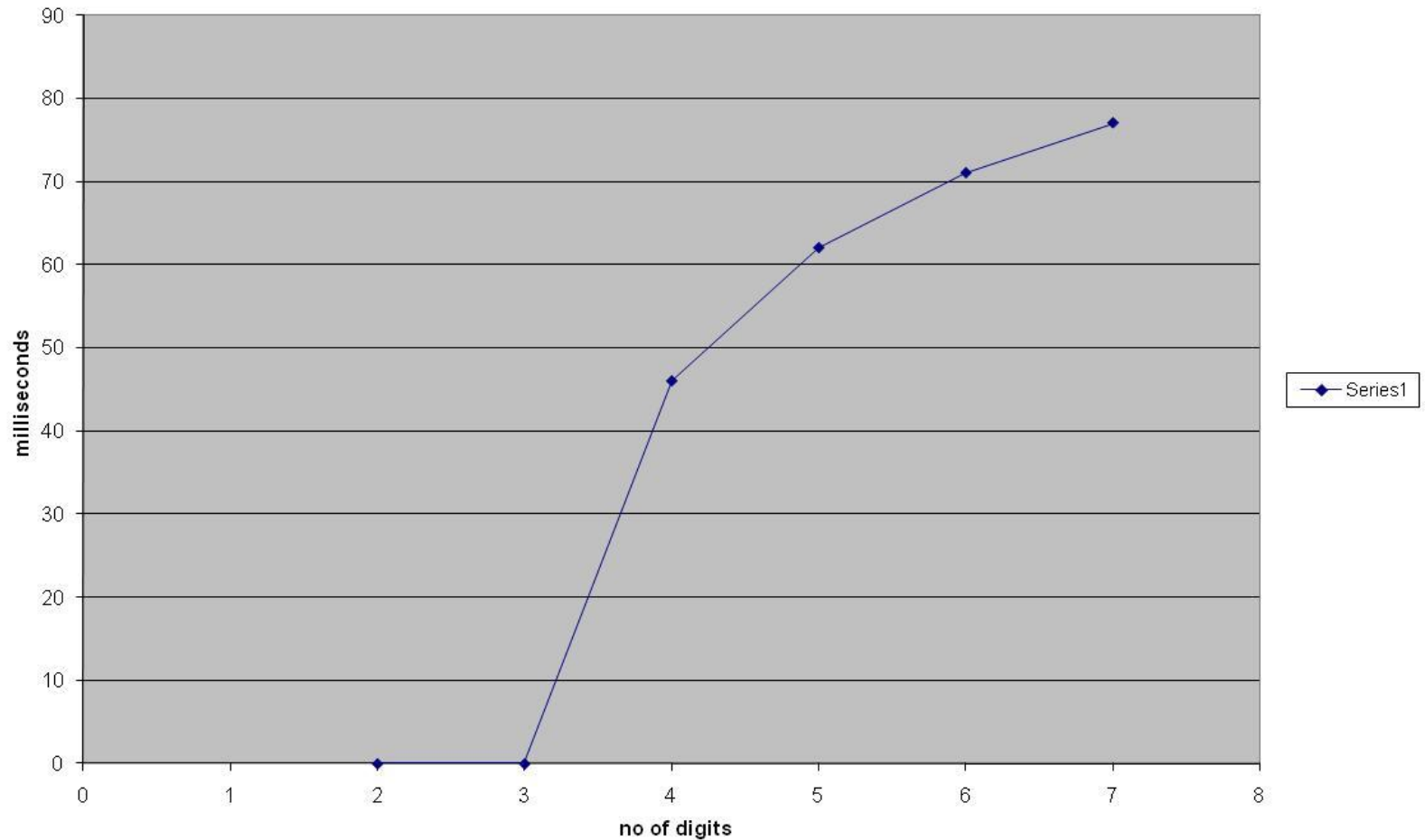
- otherwise return *prime* otherwise do next LOOP1 return *possibly composite*..

Example

- Suppose we wish to determine if $n = 5$ is prime or composite.
 - Let $a = 3$. $a^{n-1} \pmod n = 3^4 \pmod 5 = 1$.
 - As it is not composite we will enter loop 2
 - Here prime factor of $n-1$ i.e., 4 is 2.
 - $n-1/2 = 4/2 = 2$.
 - $a^{n-1/q} \pmod n = 3^2 \pmod 5$ which is not equal to 1.
 - Therefore given number is prime.

Lucas graph

Lucas primality test



Lucas-Lehmer Primality Test

- Lucas-Lehmer test is the primality test for Mersenne numbers.
- Mersenne number is a positive integer that is one less than a power of two.

$$M = 2^s - 1$$

INPUT: a Mersenne number $n = 2^s - 1$ with $s \geq 3$.

OUTPUT: an answer “prime” or “composite” to the question: “Is n prime?”

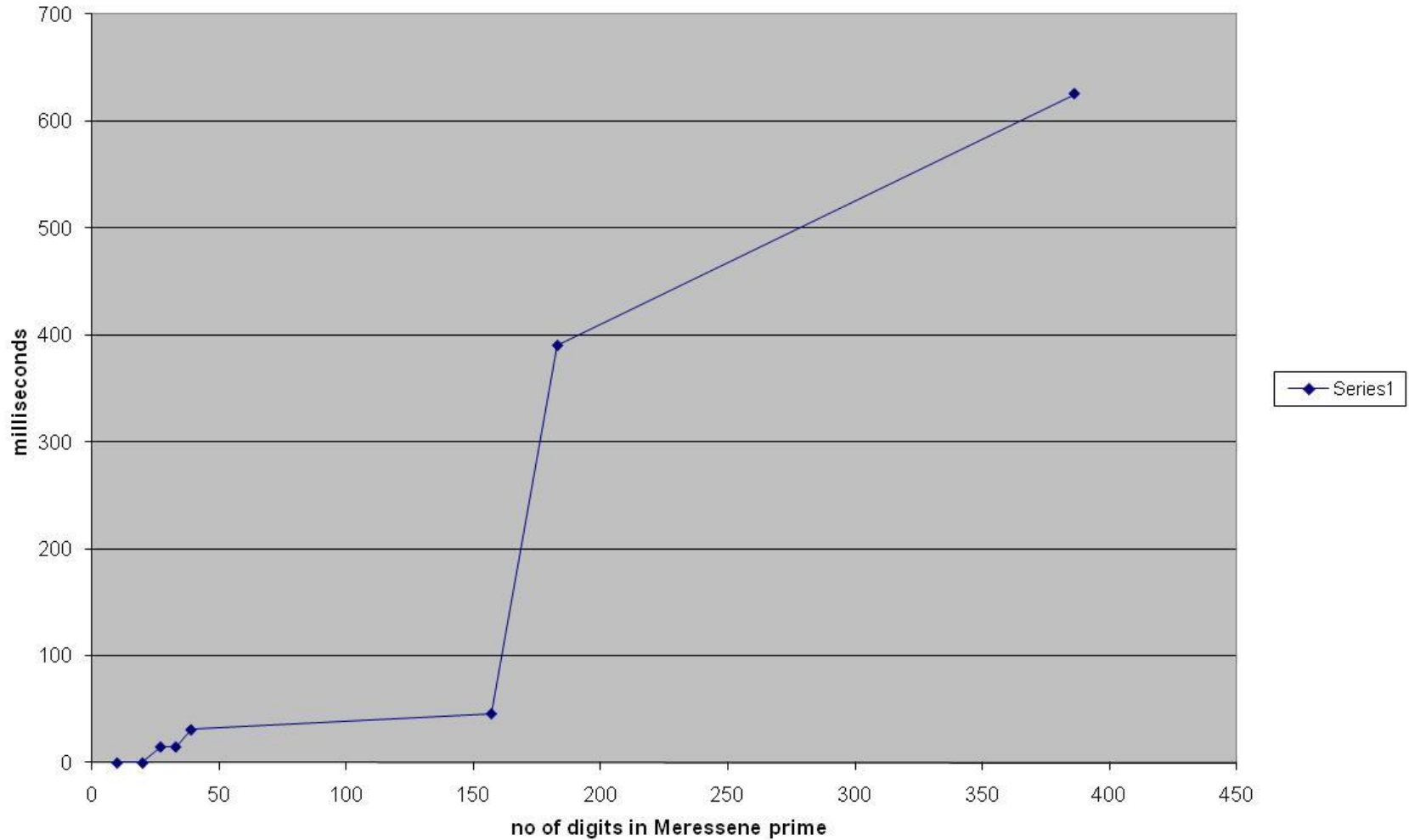
- Use trial division to check if s has any factors between 2 and \sqrt{s} . If it does, then return (“composite”).
- Set $u \leftarrow 4$.
- For k from 1 to $s-2$ do the following: Compute $u \leftarrow (u*u-2) \bmod n$.
- If $u = 0$ then return (“prime”). Otherwise, return (“Composite”).

Example

- Suppose we wish to determine if $M = 7$ is a mersenne prime or not.
- $7 = 2^3 - 1$. Therefore M is a mersenne number and $s=3$.
- Now $u = (4*4) - 2 \pmod{7} = 14 \pmod{7} = 0$
- The number is mersenne prime

Lucas-Lehmer graph

Lucas-Lehmer



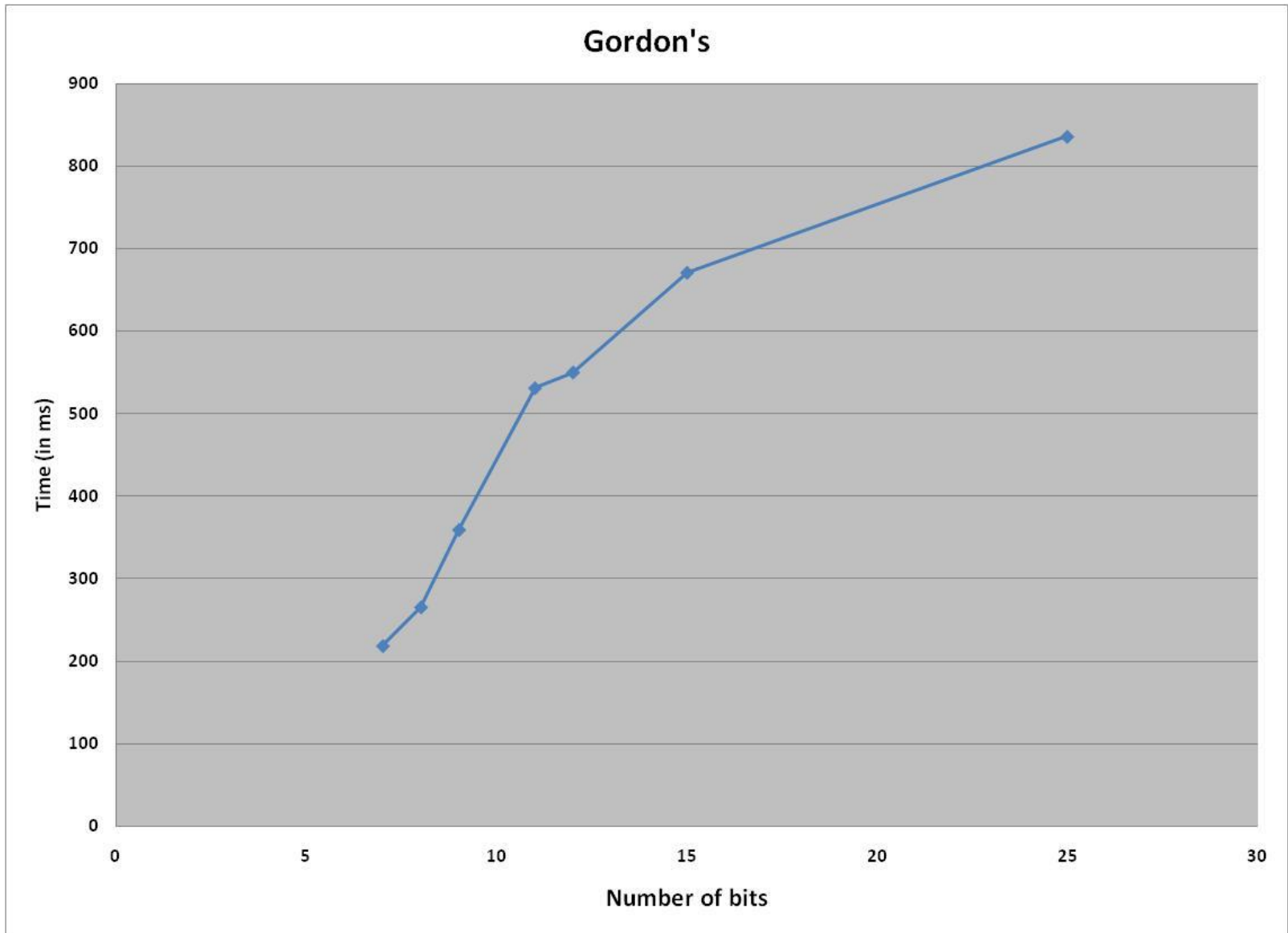
Gordon's algorithm

- Gordon's algorithm generates strong primes.
- A prime number p is said to be a *strong prime* if integers r , s , and t exist such that the following three conditions are satisfied:
 - $p - 1$ has a large prime factor, denoted r ;
 - $p + 1$ has a large prime factor, denoted s ; and
 - $r - 1$ has a large prime factor, denoted t .

Gordon's algorithm

- Output: a strong prime p is generated.
 - Generate two large random primes 's' and 't' of roughly equal bit length
 - Select an integer i_0 . Find the first prime in the sequence $2^{it} + 1$, for $i = i_0, i_0 + 1, \dots$. Denote this prime by $r = 2^{i_0 t} + 1$.
 - Compute $p_0 = 2 \pmod{r}^{s-1}$.
 - Select an integer j_0 . Find the first prime in the sequence $p_0 + 2^j r^s$, for $j = j_0, j_0 + 1, \dots$. Denote this prime by $p = p_0 + 2^{j_0} r^s$.
 - Return(p).

Gordon's graph



Maurer's algorithm

- Maurer's algorithm (Algorithm 4.62) generates random *provable primes that are almost* uniformly distributed over the set of all primes of a specified size.
- A **provable prime** is an integer that is either constructed to be prime or is calculated to be prime using a primality-proving algorithm

Maurer's algorithm

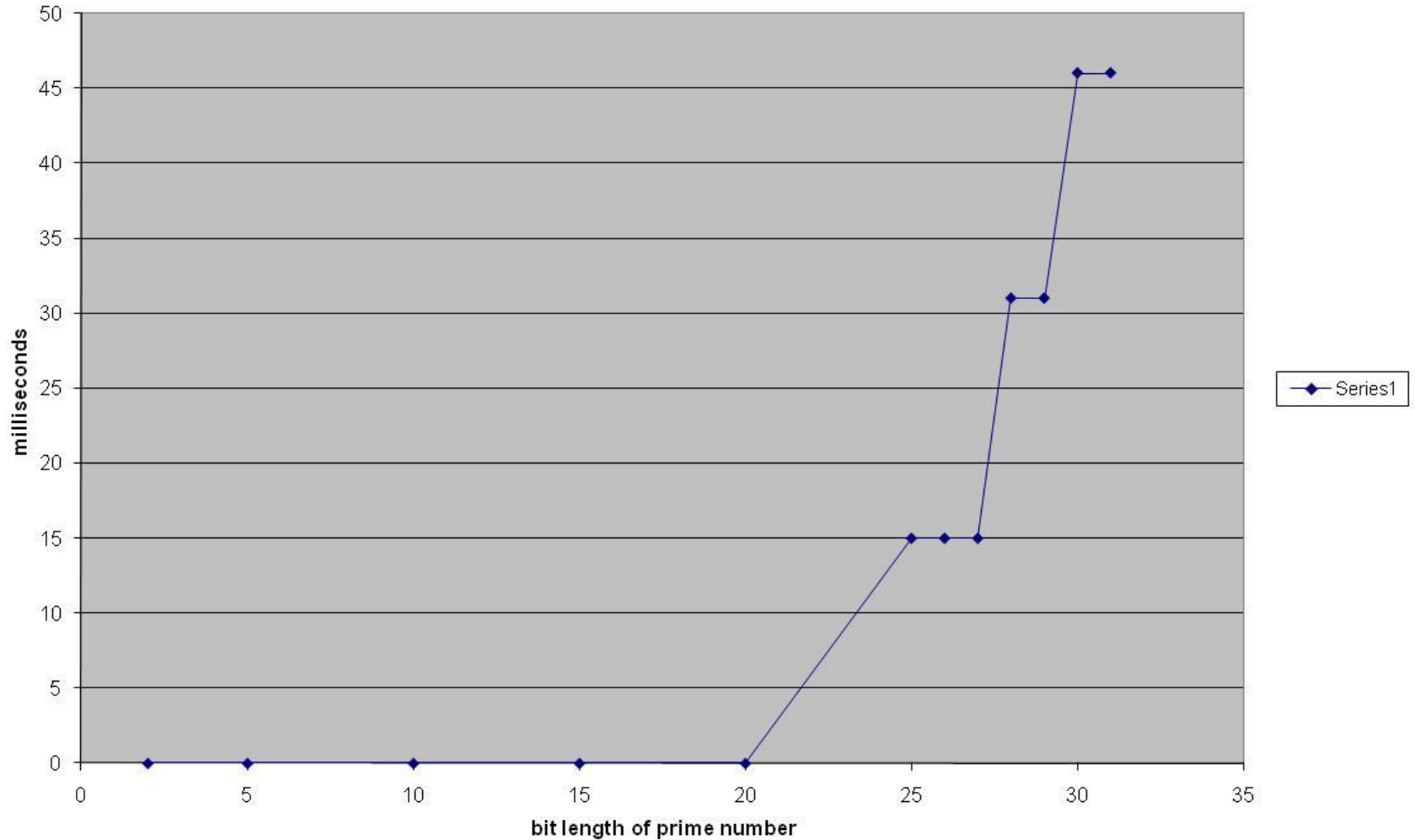
- $\text{PROVABLE_PRIME}(k)$
INPUT: a positive integer k .
OUTPUT: a k -bit prime number n
- If $k \leq 20$ then repeatedly do the following:
 - 1.1 Select a random k -bit odd integer n .
 - 1.2 Use trial division by all primes less than \sqrt{n} to determine whether n is prime.
 - 1.3 If n is prime then return(n).
- Set $c \leftarrow 0.1$ and $m \leftarrow 20$
- Set $B \leftarrow c * k * k$
- If $k > 2m$ then repeatedly do the following: select a random number s in the interval $[0,1]$, set $r \leftarrow s$, until $(k - r * k) > m$. Otherwise (i.e. $k \leq 2m$),
- set $r \leftarrow 0.5$.

Maurer's algorithm

- Compute $q \leftarrow \text{PROVABLE_PRIME}([r*k]+1)$.
- Set $I \leftarrow (/2q)$.
- Success $\leftarrow 0$.
- While (success = 0) do the following:
 - Select a random integer R in the interval $[I+1, 2I]$ and set
 $n \leftarrow 2Rq + 1$.
 - Use trial division to determine whether n is divisible by any prime number $< B$.
 - If it is not then do the following:
 - Select a random integer a in the interval $[2, n-2]$.
 - Compute $b \leftarrow a^2 \pmod n$.
 - If $b = 1$ then do the following:
 - Compute $b \leftarrow a^{b-1} \pmod n$ and $d \leftarrow \text{gcd}(b-1, n)$.
 - If $d = 1$ then success $\leftarrow 1$.
- Return (n) .

Maurer's graph

Maurers algorithm



Conclusion

- There are infinite number of primes and every day they discover a new prime.
- Target one is 4,053,946 digits
- No real formula or algorithm will find all the primes

Questions?