SinkrOn
JURNAL & PENELITIAN TEKNIK INFORMATIKA
Journal Publications & Informatics Engineering Research
Volume 3, Number 2, April 2019
DOI : https://doi.org/10.33395/sinkron.v3i1.211
e-ISSN : 2541-2019
p-ISSN : 2541-044X

The Comparison of Methods for Generating Prime Numbers between The Sieve of Eratosthenes, Atkins, and Sundaram

Muhammad Khoiruddin Harahap
Politeknik Ganesha
Medan, Indonesia
choir.harahap@yahoo.com

Nurul Khairina
Universitas Medan Area
Medan, Indonesia
nurulkhairina27@gmail.com

Abstract **—** Prime numbers are unique numbers. Prime numbers are numbers that only have a dividing factor consisting of numbers 1 and the number itself. The prime numbers from 1 to n that are relatively small can be generated manually, but a prime number generator algorithm is needed to generate prime numbers on a large scale. This study compares three prime number generator algorithms, namely: The Sieve of Eratosthenes, The Sieve of Atkins, and The Sieve of Sundaram. These three sieve algorithms have their own differences in generating prime numbers. The Sieve of Eratosthenes uses a simpler method by crossing multiples of prime numbers and marking them as non-prime numbers. The Sieve of Atkins uses several requirements for quadratic equations and modulus in determining prime numbers. The Sieve of Sundaram has an algorithm similar to The Sieve of Atkins, but there are requirements for linear equations to determine prime numbers. This study aims to see a comparison of these three algorithms in terms of accuracy and speed in generating prime numbers on a large scale. The results of this study indicate The Sieve of Eratosthenes, Atkins and Sundaram algorithms can generate large numbers of prime numbers with good accuracy, this was tested by the Fermat Primality Test Algorithm. The conclusion that can be drawn from this study, The Sieve of Eratosthenes have a faster time to generate prime numbers on a large scale than the other two algorithms.

Keywords — prime number; eratosthenes; atkins; sundaram

I. INTRODUCTION

In computer science, a discussion of prime numbers is a simple matter that is enough to attract the attention of researchers. One of the uses of prime numbers is to generate public keys and private keys on cryptography. Prime numbers are unique numbers. According to mathematical theory, the uniqueness of number 2 is where number 2 is the first and only prime number (Pande, 2015).

The sieve algorithm is an algorithm that searches for prime numbers starting from 1 to n (Sorenson, 2015). There are several sieve algorithms, including The Sieve of Eratosthenes, The Sieve of Atkins, The Sieve of Sundaram, and Mersenne Prime Number (Maltare & Chudasama, 2016). Besides the prime number generator algorithm, there is also a special algorithm for testing prime numbers such as Fermat Little Theorem, Miller-Rabin, Solovoy-Strassen and so on (Tarafder & Chakroborty, 2019).

There are several previous studies that are relevant to this study. Research conducted by (Kochar & Dheeraj Puri Goswami, 2016), do a comparison of two prime number generator algorithms, namely The Sieve of Eratosthenes and The Sieve of Sundaram. The Primality Test is carried out with several algorithms such as Miller-Rabin, Fermat, Solovoy-Strassen, and Frobenius. From the test results, it can be seen that the performance of The Sieve of Eratosthenes is better than The Sieve of Sundaram.

Research conducted by (Apdilah, Harahap, & Khairina, 2017) perform prime generation with the Mersenne Prime Number algorithm and perform testing with Miller-Rabin Primality Test. This research can

SinkrOn
JURNAL & PENELITIAN TEKNIK INFORMATIKA
Journal Publications & Informatics Engineering Research
Volume 3, Number 2, April 2019
DOI : https://doi.org/10.33395/sinkron.v3i1.211
e-ISSN : 2541-2019
p-ISSN : 2541-044X

produce prime numbers on a large scale. This large prime number is useful for RSA cryptographic algorithms, which can increase the security of public keys and private keys.

Research conducted by (Rajput & Bajpai, 2019) conduct research on probabilistic and deterministic Primality Test algorithms such as Fermat, AKS, Miller-Rabin, and Solovay Strassen on prime numbers on a large scale. The results showed that the AKS (Agrawal-Kayal-Saxena Test) is the most efficient primality test algorithm among others.

In this study, researchers had an interest in generating large numbers of prime numbers with the algorithms of The Sieve of Eratosthenes, The Sieve of Atkins, and The Sieve of Sundaram, and tested their accuracy using the Fermat Primality Test Algorithm. The test results will be displayed in the form of tables and graphs.

II. LITERATURE REVIEW

A. *Fermat Primality Test Algorithm*

Testing of prime numbers can use several algorithms, namely by testing prime numbers with Fermat or using the Miller Rabin algorithm. In this discussion only using prime number testing algorithms using Fermat's Theory. Testing prime numbers with the Fermat algorithm is a probability, so it needs to be tested several times until then it can be ascertained that the prime number is deterministic.

$$a^{(n-1)} \bmod n \equiv 1 \text{ , where } 1 < a < n \text{ ….. (1)}$$

a = random number
n = prime number

If the modulo results are ≠ 1, then n is not a prime number (Agrawal, 2006).

Example: Test number 37, whether prime or not prime. We take a = 2, n = 37, then :

$$2^{(37-1)} \equiv 1 \pmod{37} = 2^{36} \bmod 37 = 1.$$

Because the modulo results are 1, then number 37 is a prime number.

B. *The Sieve of Eratosthenes*

The following is the process of the Sieve of Eratosthenes algorithm in generating prime numbers:

Step - 1 :

Make a list of numbers from 1 to n

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Step - 2 :
Mark the first prime number in number 2 (Pande, Prime Generating Algorithms by Skipping Composite Divisors, 2014), then mark and cross all numbers that are multiples of number 2 (E.O'Neill, 2009).

	2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~
11	~~12~~	13	~~14~~	15	~~16~~	17	~~18~~	19	~~20~~
21	~~22~~	23	~~24~~	25	~~26~~	27	~~28~~	29	~~30~~
31	~~32~~	33	~~34~~	35	~~36~~	37	~~38~~	39	~~40~~
41	42	43	~~44~~	45	~~46~~	47	~~48~~	49	~~50~~

Step - 3 :
Mark the next prime number in number 3 then mark and cross all numbers which are multiples of 3.

	2	**3**	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	25	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	35	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	49	~~50~~

Step - 4 :
Repeat steps - 2 and step - 3 until all multiples of numbers are crossed.

	2	**3**	~~4~~	**5**	~~6~~	**7**	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	49	~~50~~

Step – 5 :
Prime numbers are numbers from 2-n that are not crossed during the multiple deletion process of the previous number (Apdilah, Harahap, & Khairina, 2017)

	2	**3**		**5**		**7**			
11		13				17		19	
		23						29	
31						37			
41		43				47		49	

The series of prime numbers between 2 - 50 are : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 49

SinkrOn
JURNAL & PENELITIAN TEKNIK INFORMATIKA
Journal Publications & Informatics Engineering Research
Volume 3, Number 2, April 2019
DOI : https://doi.org/10.33395/sinkron.v3i1.211
e-ISSN : 2541-2019
p-ISSN : 2541-044X

C. The Sieve of Atkins

Following is the process of The Sieve of Atkins algorithm in filtering prime numbers:

Step – 1 :
Determine the search for prime numbers of numbers from 1 – n, where in this study we will generate prime numbers from 1- 50, so that n = 50.

Step – 2 :
Mark numbers 2, 3 and 5 as prime numbers.

Step – 3 :
Calculate the equation based on the following 3 blocks:

$4x^2 + y^2 \bmod 4 => 1 \text{ or } 5 \rightarrow 4x^2 + y^2 < n$....(2)
$3x^2 + y^2 \bmod 6 => 1 \rightarrow 3x^2 + y^2 < n$ …… (3)
$3x^2 – y^2 \bmod 12 => 11 \rightarrow 0 < 3x^2 – y^2 < n$ .. (4)

Formula 2			Formula 3			Formula 4		
x	y	n	x	y	n	x	y	n
1	1	5	1	1	4	1	1	2
1	2	8	1	2	7	2	1	11
1	3	13	1	3	12	2	2	8
1	4	20	1	4	19	2	3	3
1	5	29	1	5	28	3	1	26
1	6	40	1	6	39	3	2	23
2	1	17	2	1	13	3	3	18
2	2	20	2	2	16	3	4	11
2	3	25	2	3	21	3	5	2
2	4	32	2	4	28	4	1	47
2	5	41	2	5	37	4	2	44
3	1	37	2	6	48	4	3	39
3	2	40	3	1	28	4	4	32
3	3	45	3	2	31	4	5	23
			3	3	36	4	6	12
			3	4	43			
			4	1	49			

Step – 4 :
In this step, there are two steps, namely:
a. Delete the value of n which the module results do not match with the requirements of modulo formulas 2, 3 and 4.
b. Delete numbers that are not square-free numbers

- ***Explanation of square-free numbers:***

Square-free numbers are integer integers that cannot be divided by perfect squares other than 1. That is, the main factorization has exactly one factor for each prime that appears in it. As an example, $10 = 2 \cdot 5$ is quadratic free, but $18 = 2 \cdot 3 \cdot 3$ is not square-free numbers, because 18 can be divided by 9 = 2.

Can be seen in the table below, numbers that are **not square-free numbers** are **25, 45, 49**.

Proof that 3 numbers below are not square-free are:

- 25 => is divided by 25 which is the square of 5. Example → $25 : 5^2 = 1$
- 45 => is divided by 45 which is the square of 3. Example → $45 : 3^2 = 1$
- 49 => is divided by 49 which is the square of 7. Example → $49 : 7^2 = 1$

$4x^2 + y^2 \bmod 4 = 1$ or 5	$3x^2 + y^2 \bmod 6 = 1$	$3x^2 – y^2 \bmod 12 = 11$
5 mod 4 = 1	~~4 mod 6 = 4~~	~~2 mod 12 = 2~~
~~8 mod 4 = 0~~	7 mod 6 = 1	11 mod 12 = 11
13 mod 4 = 1	~~12 mod 6 = 0~~	~~8 mod 12 = 8~~
~~20 mod 4 = 0~~	19 mod 6 = 1	~~3 mod 12 = 3~~
29 mod 4 = 1	~~28 mod 6 = 4~~	~~26 mod 12 = 2~~
~~40 mod 4 = 0~~	~~39 mod 6 = 3~~	23 mod 12 = 11
17 mod 4 = 1	13 mod 6 = 1	~~18 mod 12 = 6~~
~~20 mod 4 = 0~~	~~16 mod 6 =4~~	11 mod 12 = 11
~~25~~	~~21 mod 6 = 3~~	~~2 mod 12 = 2~~
~~32 mod 4 = 0~~	~~28 mod 6 = 4~~	47 mod 12 = 11
41 mod 4 = 1	37 mod 6 = 1	~~44 mod 12 = 8~~
37 mod 4 = 1	~~48 mod 6 =0~~	~~39 mod 12 = 3~~
~~40 mod 4 = 0~~	~~28 mod 6 = 4~~	~~32 mod 12 = 8~~
~~45~~	31 mod 6 = 1	23 mod 12 = 11
	~~36 mod 6 = 0~~	~~12 mod 12 = 0~~
	43 mod 6 = 1	
	~~49~~	

Step – 5 :
After step - 4, list the remaining selection numbers:

$4x^2 + y^2 \bmod 4 = 1$ or 5	$3x^2 + y^2 \bmod 6 = 1$	$3x^2 – y^2 \bmod 12 = 11$
5 mod 4 = 1	7 mod 6 = 1	11 mod 12 = 11
13 mod 4 = 1	19 mod 6 = 1	23 mod 12 = 11
29 mod 4 = 1	13 mod 6 = 1	11 mod 12 = 11
17 mod 4 = 1	37 mod 6 = 1	47 mod 12 = 11
41 mod 4 = 1	31 mod 6 = 1	23 mod 12 = 11
37 mod 4 = 1	43 mod 6 = 1	

Step – 6 :
Delete repeated numbers in step number list - 5, and re-create a new list.

$4x^2 + y^2 \bmod 4 = 1$ or 5	$3x^2 + y^2 \bmod 6 = 1$	$3x^2 – y^2 \bmod 12 = 11$
5	7	11

SinkrOn
JURNAL & PENELITIAN TEKNIK INFORMATIKA
Journal Publications & Informatics Engineering Research
Volume 3, Number 2, April 2019
DOI : https://doi.org/10.33395/sinkron.v3i1.211
e-ISSN : 2541-2019
p-ISSN : 2541-044X

13	19	23
29	31	47
17	43	
41		
37		

Arrange the prime numbers obtained in this step from the smallest to the largest number.

Here is the intermediate prime number 1- 50 :
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47

D. The Sieve of Sundaram

The process of The Sieve of Sundaram algorithm is as follows:

Step – 1 :
Determine the value of n as the range of prime numbers. We suppose n = 102. Make a number list according to the formula : (n-2)/2 = (102-2)/ 2, where there are 50 numbers in the list:

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Step – 2 :
Mark and cross out the numbers that are the result of:

$$i + j + 2ij \leq n \ldots\ldots\ (5)$$

where the test value i and j the first time simultaneously starts from number 1.

Selection 1:

i	j	i + j + 2*i*j
1	1	4
1	2	7
1	3	10
1	4	13
1	5	16
1	6	19
1	7	22
1	8	25
1	9	28
1	10	31
1	11	34
1	12	37
1	13	40
1	14	43
1	15	46
1	16	49

The above selection iteration will stop until the value of j < 50.

The Result of Selection 1 :

1	2	3	~~4~~	5	6	~~7~~	8	9	~~10~~
11	12	~~13~~	14	15	~~16~~	17	18	~~19~~	20
21	~~22~~	23	24	~~25~~	26	27	~~28~~	29	30
~~31~~	32	33	~~34~~	35	36	~~37~~	38	39	~~40~~
41	42	~~43~~	44	45	~~46~~	47	48	~~49~~	50

Step -3 :
Repeat step-2 to cross out numbers that are not prime numbers, and ignore the number of results **i + j + 2*i*j** that same from the selection results obtained in the previous stage.

Selection 2 :

i	j	i + j + 2*i*j
2	1	7 => repeat
2	2	12
2	3	17
2	4	22 => repeat
2	5	27
2	6	32
2	7	37 => repeat
2	8	42
2	9	47

The Result of Selection 2 :

1	2	3	~~4~~	5	6	~~7~~	8	9	~~10~~
11	~~12~~	~~13~~	14	15	~~16~~	~~17~~	18	~~19~~	20
21	~~22~~	23	24	~~25~~	26	~~27~~	~~28~~	29	30
~~31~~	~~32~~	33	~~34~~	35	36	~~37~~	38	39	~~40~~
41	~~42~~	~~43~~	44	45	~~46~~	~~47~~	48	~~49~~	50

Step - 4 :
Repeat step-2 to cross out numbers that are not prime numbers, and ignore the number of results **i + j + 2*i*j** that same from the selection results obtained in the previous stage.

Selection 3 :

i	j	i + j + 2*i*j
3	1	10 => repeat

SinkrOn
JURNAL & PENELITIAN TEKNIK INFORMATIKA
Journal Publications & Informatics Engineering Research
Volume 3, Number 2, April 2019
DOI : https://doi.org/10.33395/sinkron.v3i1.211
e-ISSN : 2541-2019
p-ISSN : 2541-044X

3	2	17 => repeat
3	3	24
3	4	31 => repeat
3	5	38
3	6	45

The Result of Selection 3 :

1	2	3	~~4~~	5	6	~~7~~	8	9	~~10~~
11	~~12~~	~~13~~	14	15	~~16~~	~~17~~	18	~~19~~	20
21	~~22~~	23	~~24~~	~~25~~	26	~~27~~	~~28~~	29	30
~~31~~	~~32~~	33	~~34~~	35	36	~~37~~	~~38~~	39	~~40~~
41	~~42~~	~~43~~	44	~~45~~	~~46~~	~~47~~	48	~~49~~	50

Step -5 :
Repeat step-2 to cross out numbers that are not prime numbers, and ignore the number of results **i + j + 2*i*j** that same from the selection results obtained in the previous stage.

Selection 4 :

i	**j**	**i + j + 2*i*j**
4	1	13 => repeat
4	2	22 => repeat
4	3	31 = > repeat
4	4	40 => repeat
4	5	49 => repeat

At this stage of selection, where i = 4 and j = $1 \leq j \leq 5$, it looks like all numbers have been repeated, so the iteration stops at step - 4.

Step – 6 :
Prime numbers are searched by formulas 2*x + 1, where x is the result of number selection in step - 3.

Last Selection Number	**2*x + 1**
1	3
2	5
3	7
5	11
6	13
8	17
9	19
11	23
14	29
15	31
18	37
20	41
21	43
23	47
26	53
29	59
30	61
33	67
35	71
36	73
39	79
41	83
44	89
48	97
50	101

Between prime numbers 1 – 102 :
3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, and 101 (Pruitt & Shannon, 2018).

III. RESULT AND DISCUSSION

The comparison made in this discussion is to test accuracy and speed to generate a large prime number. The comparison of accuracy in determining the accuracy of the three methods is tested by the Fermat method with a certain range of results can be seen as follows:

No	**Method**	**Range**	**Total Prime**	**Invalid (Fermat test)**
1	Eratosthenes	100	25	0
		1000	168	0
		1000000	78498	0
2	Atkins	100	25	0
		1000	168	0
		1000000	78498	0
3	Sundaram	100	25	0
		1000	168	0
		1000000	78498	0

The speed comparison of the three methods to generate the prime numbers is as follows:

No	**Method**	**Range**	**Milli second**
1	Eratosthenes	1000	0.0
		1000000	1.84856009483337
		10000000	19.5828666687011

SinkrOn
JURNAL & PENELITIAN TEKNIK INFORMATIKA
Journal Publications & Informatics Engineering Research
Volume 3, Number 2, April 2019
DOI : https://doi.org/10.33395/sinkron.v3i1.211
e-ISSN : 2541-2019
p-ISSN : 2541-044X

2	Atkins	1000	0.01612544059753
		1000000	3.39212107658386
		10000000	99.6193344593048
3	Sundaram	1000	0.0
		1000000	3.88802647590637
		10000000	26.3855686187744

IV. CONCLUSION AND SUGGESTION

A. Conclusion

The conclusions of this study are as follows:

1. The Sieve of Eratosthenes, Atkins and Sundaram algorithms can generate large numbers of prime numbers with good accuracy, this was tested by the Fermat Primality Test Algorithm.
2. The Sieve of Eratosthenes, Atkins and Sundaram algorithms generate the same number of primes in each test of the range 100, 1000, and 1000000 numbers.
3. From the speed, ratio generates prime numbers in three number range groups, a The Sieve of Eratosthenes have a faster time to generate prime numbers on a large scale than the other two algorithms.

B. Suggestion

The suggestion for this research in the future is to be able to test the accuracy of a prime number generator with other primes testing algorithms such as Rabin Miller and Solovay-Strassen.

V. REFERENCES

Agrawal, M. (2006). Primality Tests Based on Fermat's Little Theorem. In *Distributed Computing and Networking* (pp. 288-293). Guwahati: Springer, Berlin, Heidelberg.

Apdilah, D., Harahap, M. K., & Khairina, N. (2017). Generating Mersenne Prime Number Using Rabin Miller Primality Probability Test to Get Big Prime Number Cryptography in RSA. *International Journal of Information System & Technology*, 1-7.

E.O'Neill, M. (2009). The Genuine Sieve of Eratosthenes. *Journal of Functional Programming (JEP)*, 95-106.

Kochar, V., & Dheeraj Puri Goswami, M. A. (2016). Contrast Various Test For Primality. *International Conference on Accessibility to Digital World (ICADW)* (pp. 1-6). Guwahati: IEEE.

Maltare, N., & Chudasama, C. (2016). Experimenting Large Prime Numbers Generation in MPI Cluster. *International Congress on Information and Communication Technology, Advances in Intelligent Systems and Computing* (pp. 61-66). Gujarat: Springer Singapore.

Pande, N. A. (2014). Prime Generating Algorithms by Skipping Composite Divisors. *International Journal of Computer Science & Engineering Technology (IJCSET)*, 935-940.

Pande, N. A. (2015). Refinement of Prime Generating Algorithms. *International Journal of Innovative Science, Engineering & Technology (IJISET)*, 21-24.

Prime Sieve Using Binary Quadratic Forms2003*Mathematics of Computation* 1023-1030

Pruitt, K., & Shannon, A. G. (2018). Modular Class Primes in the Sundaram Sieve. *International Journal of Mathematical Education in Science and Technology*, 1-4.

Rajput, J., & Bajpai, A. (2019). Study on Deterministic and Probabilistic Computation of Primality Test. *International Conference on Sustainable Computing in Science, Technology & Management (SUSCOM)* (pp. 2206-2212). Jaipur : India: Elsevier SSRN.

Sorenson, J. P. (2015). Two compact incremental prime sieves. *LMS J. Comput. Math.*, 675–683.

Tarafder, A. K., & Chakroborty, T. (2019). A Comparative Analysis of General, Sieve-of-Eratosthenes and Rabin-Miller Approach for Prime Number Generation. *International Conference on Electrical, Computer and Communication Engineering (ECCE)* (pp. 1-4). Bangladesh: IEEE.