Scientific Research Publishing

Applied Mathematics, 2026, 17(11), 715-719
https://www.scirp.org/journal/am
ISSN Online: 2152-7393
ISSN Print: 2152-7385

A Note on Generation of Prime Numbers

Fleur T. Tehrani

Department of Electrical and Computer Engineering, California State University, Fullerton, Fullerton, CA, USA
Email: ftehrani@fullerton.edu

How to cite this paper: Tehrani, F.T. (2026) A Note on Generation of Prime Numbers. *Applied Mathematics*, **17**, 715-719.
https://doi.org/10.4236/am.2026.1711039

Received: September 10, 2026
Accepted: November 10, 2026
Published: November 13, 2026

Copyright © 2026 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution-NonCommercial International License (CC BY-NC 4.0).
http://creativecommons.org/licenses/by-nc/4.0/

Open Access

Abstract

A modified sieving algorithm is presented to generate prime numbers. The procedure is designed to be less complex and potentially less computationally intensive than other methods. This algorithm is iterative and is error-free.

Keywords

Computer Algorithm, Deterministic Methods, Prime Numbers

1. Introduction

A positive integer that is not the product of two other positive integers is called a prime number. Prime numbers have numerous applications, including in various technological fields. Examples of their applications include encryption and cybersecurity for protecting sensitive information, and error-correcting codes used in digital communication systems.

There have been many attempts to find a formula for generating prime numbers, but none have been found to date. Prime numbers are found by using algorithmic procedures, and some of those procedures are not error-free [1] [2].

The purpose of this paper is to present a modified sieving algorithm for generating prime numbers, which is designed to produce primes without error. The algorithm is iterative and designed to reduce computational time.

2. Methods

Prime numbers are not limited, but it is known that the number of primes less than a number n approaches a limit as n becomes large:

$$\pi(n) \sim \frac{n}{\log n} \tag{1}$$

where $\pi(n)$ is the number of prime numbers less than n. The difference between

$\pi(n)$ and the fraction on the right side of the above expression approaches zero as n tends to infinity [3].

$$\lim_{n\to\infty}\left[E(n)=\left(\pi(n)-\frac{n}{\log n}\right)\right]=0 \tag{2}$$

It is also known that primes less than a number n, need to be undividable by integers up to $\sqrt{n}$ to be prime numbers [4].

The procedure presented in this paper is designed to be deterministic and less complex than some other methods. This modified procedure has the potential to reduce computational time compared to other sieving techniques [5] [6]. In computer algorithms, it generally takes less time to find out whether a number is an integer multiple of a set of smaller numbers by iteratively omitting multiples than to eliminate the integer multiples of each individual prime separately, from beginning to end. To make the procedure more efficient, the procedure is performed in two stages; in the first stage, all the primes up to $\sqrt{n}$ (where n is the maximum number) are found, and in the second stage, the numbers greater than $\sqrt{n}$ are iteratively compared with subgroups of primes found in the first stage to find all prime numbers. These are done to reduce the computational intensity of the algorithm.

Procedure

A flowchart of the procedure is shown in **Figure 1(a)** and **Figure 1(b)**.

As shown in **Figure 1(a)**, the maximum number, n, is read at the beginning, the square root of n is found, and the integer part of $\sqrt{n}$ is called K1. Then the algorithm checks all odd integers less than K1 against several initial primes (e.g., 3, 5, and 7) and places those numbers that are not multiples of this initial set of primes in file 1.

As shown in **Figure 1(b)**, after the initial file 1 is formed, the procedure continues by checking each number in that file to see whether it is an integer multiple of any of the smaller numbers in the file. If it is, the number is discarded, and if not, it is kept in the file. Then the numbers in file 1 are counted, and the initial set of primes (e.g., 2, 3, 5, 7) is added to the file.

At this point, file #1 contains all primes that are less than $\sqrt{n}$ (K1). Then the numbers in file #1, starting from number 3, are subdivided into groups of several primes (e.g., groups of 3 numbers each). The next part of the procedure begins by checking every odd integer greater than K1 and up to n, versus each group of numbers in file #1. If they are found to be an integer multiple of any number in the group, they are discarded. This procedure is iterative, with numbers checked against each set in file #1, and the remaining group keeps getting smaller as multiples are found and discarded. The final remaining file is called file #2 and is added to file 1 to form the final file of prime numbers less than n. The numbers in the file are counted and then printed.

An example print-out of primes generated by this procedure, less than 1000, is shown in **Figure 2**. This relatively small sample is presented to show that the results produced by the proposed method are accurate and error-free.

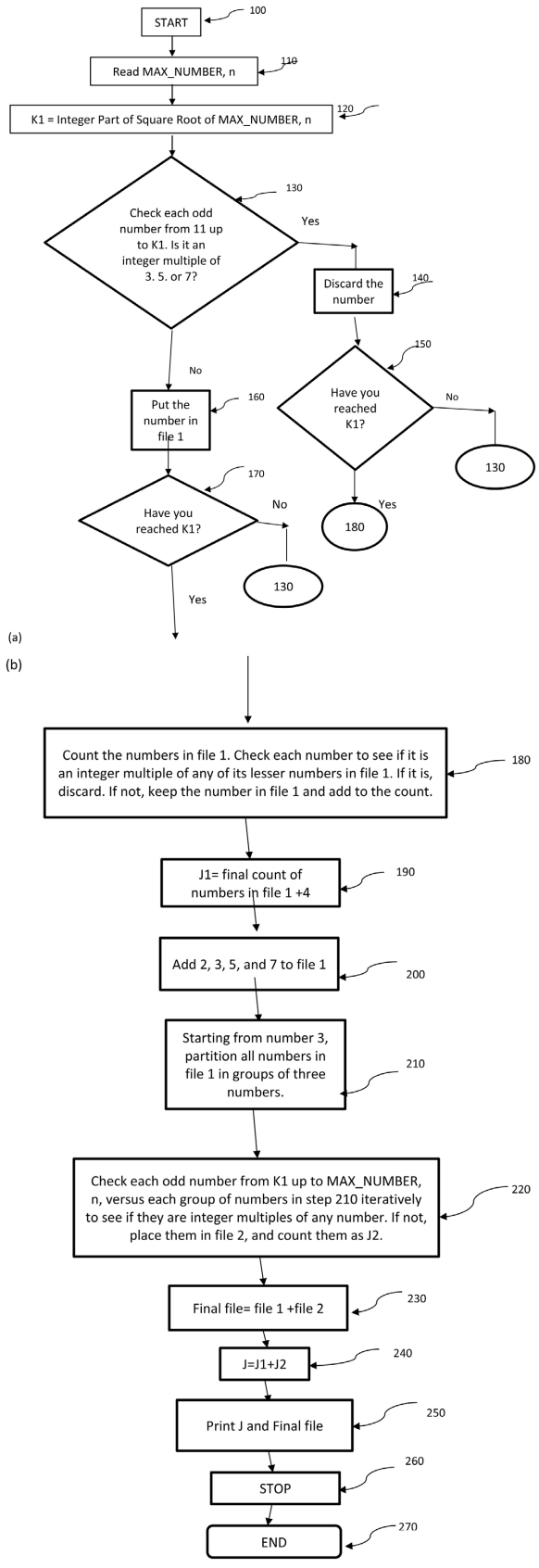


Figure 1. Flowchart of the procedure.

```
2   3  5  7  11 13 17
19  23 29 31 37 41 43
47  53 59 61 67 71 73
79  83 89  97 101 103 107
109 113 127 131 137 139 149
151 157 163 167 173 179 181
191 193 197 199 211 223 227
229 233 239 241 251 257 263
269 271 277 281 283 293 307
311 313 317 331 337 347 349
353 359 367 373 379 383 389
397 401 409 419 421 431 433
439 443 449 457 461 463 467
479 487 491 499 503 509 521
523 541 547 557 563 569 571
577 587 593 599 601 607 613
617 619 631 641 643 647 653
659 661 673 677 683 691 701
709 719 727 733 739 743 751
757 761 769 773 787 797 809
811 821 823 827 829 839 853
857 859 863 877 881 883 887
907 911 919 929 937 941 947
953 967 971 977 983 991 997
```

Figure 2. An example set of prime numbers produced by the flowchart algorithm.

The procedure presented here describes a modified sieving method that does not include complex mathematical functions. It is designed to be error-free, and the built-in modifications are intended to reduce computational intensity, especially as the maximum number increases. Table 1 compares the computational times of this method, referred to as "Flowchart Algorithm", and the widely used sieve of Eratosthenes for $n = 10^3$, 10^5, and 10^6. As seen in this table, the time efficiency of this method, compared with the sieve of Eratosthenes, increases as n increases.

Table 1. Benchmark results—comparison of the execution times of the Flowchart Algorithm with the Algorithm of Eratosthenes for different maximum numbers.

Maximum Number	Prime Count	Exec. Time of Flowchart Algorithm (ms)	Exec. Time of Eratosthenes Algorithm (ms)
1000	**168**	**0.058**	**0.067**
100,000	**9592**	**6.539**	**7.552**
1,000,000	**78,498**	**75.485**	**78.826**

3. Conclusion

Prime numbers have many applications in mathematics and other fields of science and technology. Efficient production of prime numbers has been the focus of research by many mathematicians and computer scientists. The procedure presented in this paper is a modified sieving method designed to generate primes

without error, and its iterative nature is intended to reduce computational time.

Acknowledgements

I want to thank Mr. Vadim Chissek, the Electrical and Computer Engineering Consultant at California State University, Fullerton, for helping verify the algorithmic results.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Grosswald, E. (1966) Topics from the Theory of Numbers. The Macmillan Company, 113-116.

[2] Kraft, J.S. and Washington, L.C. (2015) Elementary Number Theory. CRC Press.

[3] Grandall, R. and Pomerance, C. (2005) Prime Numbers, A Computational Perspective. 2nd Edition, Springer, 121-131.

[4] Giblin, P. (1993) Primes and Programming, an Introduction to Number Theory and Computing. Cambridge University Press.

[5] Greaves, G. (2013) Sieves in Number Theory. Springer, 1-18.

[6] Wagstaff Jr., S.S. (2013) The Joy of Factoring. American Mathematical Society, 191-198.