A FEW TRICKS TO COUNT PRIME NUMBERS EVEN FASTER

Jacek Piątkowski [1], Artur Jakubski [2], Robert Perliński [3]

[1,2,3]Department of Computer Science, Czestochowa University of Technology, Czestochowa, Poland
[1]jacek.piatkowski@icis.pcz.pl, [2]artur.jakubski@icis.pcz.pl, [3]robert.perlinski@icis.pcz.pl

Keywords: **algorithm analysis, computational number theory, primality testing, prime number generation**

Prime numbers have fascinated mathematicians, philosophers, and researchers since ancient times. Among the many, many researchers who have explored this topic, we find the names of some of the greatest – such as Eratosthenes, Euler, Lagrange, Legendre and Riemann [1, 3, 5]. In our times, prime numbers form the foundation of cryptography, secure communication, banking, and telecommunications, because while they are easy to find (even large ones), but it is difficult to factor the product of two large prime numbers.

In this paper, we present practical solutions for accelerating the calculation of prime numbers, which we have utilized in the secure and distributed data processing protocols described in [2].

In the initial stage of the research, three algorithms were implemented to test the prima-lity of natural numbers: the Optimized Trivial Algorithm (**Tr**) – Square Root Trial Division – effective for small numbers, the Miller-Rabin algorithm [5] (**M-R**), and the deter-ministic Miller-Rabin algorithm (**M-Rd**) that are used in crypto-graphy for very large prime numbers. The problem under investigation was to determine the time of generating a vector of N consecutive prime numbers (**PN**s) greater than N, for eight number ranges - $N = \{10^2, 10^3, 10^4, 10^5, 10^6, 10^{7,} 2*10^7, 3*10^7\}$.

Based on the obtained results (see Fig. 1.a), it was found that for relatively small values of N – smaller than approximately $5*10^3$ – the **Tr** algorithm is the most effective (due to its simplicity) and at the same time the fastest. The advantage of the **M-R** and **M-Rd** algorithms becomes visible for values of N greater than 10^5, while the generation time of a vector of 100 thousand **PN**s greater than 100 thousand can be considered negligible.

Algorithms modifications

Although each algorithm for testing the primality of natural numbers, in its general form, assumes the possibility of checking any number N, testing even

numbers makes no sense at all because the result is known in advance. To find the required set of prime numbers, it is sufficient to (variant **A**) start with an odd number and check subsequent values incremented by 2. This simple modification reduces the time required to generate the **PN**s vector by nearly half – see Fig.1.b.

In the next stage of testing (variant **B**), the thesis was formulated that the M-Rd algorithm can be accelerated if it - in its entire formula - is run if the candidate odd number x is divisible by one of the values y = {3, 5, 7, 11, 13}. Although the y-values are prime numbers, from a practical point of view they are not of interest to us.

Based on this assumption, a second variant of the deterministic Miller-Rabin algorithm was implemented: **M-Rd2**, in which the following instruction was added:

```
if((x%3u==0)||(x%5u==0)||(x%7u==0)||(x%11u==0)||(x%13u==0))
  return false;
```

Modification (B) once again reduced the runtime of the M-Rd algorithm by

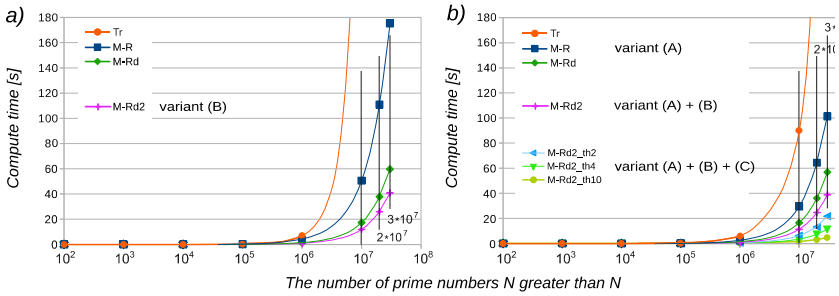


Fig.1. Computational times for prime number vectors.
Algorithms: Tr, M-R, M-Rd, and M-Rd2.
AMD Ryzen 9 3900X (12 cores, 24 threads) 2,2/3,8 GHz, Ubuntu 20.04 64 bit

approximately 30% – see Fig. 1.

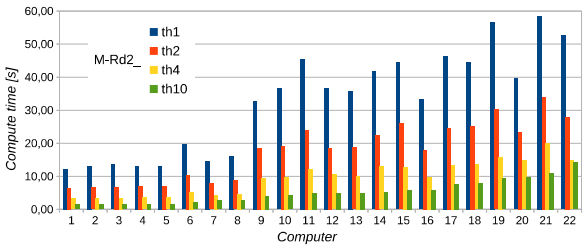
All of the tests described above involved algorithms implemented as single-threaded. In the next phase of the research, it was decided to investigate whether and to what extent the generation of prime number vectors could be accelerated using multithreaded algorithms.

Based on the general prime number theorem, which states that there are approximately x prime numbers in the range from x to $M = x*ln(x)$ [6][7], it is sufficient to divide this range into k intervals, in which k threads will perform separate calculations (variant **C**). For our purposes, we determined the end of the search range for the $N = 3*10^6$ prime numbers greater than N using the following formula:

$$M = N * (1{,}175 * \ln(N) + 0{,}136)$$

Finally, the time required to generate a prime number vector using the deterministic Miller-Rabin algorithm (**M-Rd)** was reduced from approximately 60 seconds (see Fig. 1.a) to less than 2 seconds (see Fig. 1.b.M-Rd2_th10).

We ran the multithreaded variant of the **M-Rd** algorithm on 22 different computers with different processors – from the x86 family (Intel, AMD) and ARM (Apple Mx, Cortex-925).



1	ASUS, 48 × AMD Ryzen Threadripper 7960X 24-Cores 4,2 GHz, 125,3 GiB RAM, Windows 11 Pro 24H2	12	Laptop ASUS ROG Strix G732L, Intel (R) i9-10980HK @4,7 – 4,9 GHz (TURBO), 8 cores, 16 threads, RAM 32GB, Windows 11 Pro 64 bit
2	ASUS, 48 × AMD Ryzen Threadripper 7960X 24-Cores 4,2 GHz, 125,3 GiB RAM, Ubuntu 24.04.3 LTS	13	Laptop ASUS ROG Strix G732L, Intel (R) i9-10980HK @3,1 GHz, 8 cores, 16 threads, RAM 32GB, Windows 10 Pro 64 bit
3	ASUS, 48 × AMD Ryzen Threadripper 7960X 24-Cores 4,2 GHz, 125,3 GiB RAM, Kubuntu 24.04	14	DESKTOP, Intel(R) Core(TM) i9-10980XE, 18 cores, 36 threads, CPU @ 3.00GHz, RAM 64GB, Windows 10 64-bit
4	Gigabyte TRX50 AERO, 48 × AMD Ryzen Threadripper 7960X 24-Cores, 128 GB DDR5-4800 RAM, Ubuntu 24.04	15	Laptop ASUS ROG Strix G531GV, Core i7-9750H @4,1 GHz, 6 cores, 12 threads, RAM 16GB, Windows 10 Home 64 bit
5	ASUS, 48 × AMD Ryzen Threadripper 7960X 24-Cores 4,2 GHz, 125,3 GiB RAM, Ubuntu 22.04.5 WSL – Windows 11 Pro 24H2	16	MacBook Pro, i7-1068NG7, 2.3 GHz, 4 rdzenie 8 wątków, RAM 32GB 3733 LPDDR4X, macOS Sequoia 15.4.1
6	DGX Spark, 20 core ARM Cortex-925, 128 GB, Ubuntu 24.04.4 LTS	17	DESKTOP-S5VA9V2, Intel(R) i7-6700K CPU @ 4.00GHz, RAM 32,0 GB VM UBUNTU
7	MacBook Air 13'', Apple M4, RAM 16GB, macOS Sequoia 15.4.1	18	DESKTOP-S5VA9V2, Intel(R) i7-6700K CPU @ 4.00GHz, RAM 32,0 GB Win10
8	MacBook Pro 14'', Apple M3 Pro, RAM 18GB, macOS Sequoia 15.3.2	19	Laptop ASUS G551, Intel(R) i7-4710HQ CPU @ 2,5 GHz, 4 cores, 8 threads, 16 GB RAM, win 10 home 64 bit
9	ASUSTeK PRIME X570-P, AMD Ryzen 9 3900X bits: 64 (12 rdzeni 24 wątki) 2,2/3,8 GHz, RAM 64GB, Linux Mint 20 Ulyana base: Ubuntu 20.04 64 bit	20	Laptop HP ENVY X360 14, Intel, i7-10510U @1,8 GHz, 4 cores, 8 threads, RAM 16GB, Windows 10 Home 64 bit
10	DESKTOP Intel® Core™ i9-10900, 32 GB Ram, Ubuntu	21	Laptop HP ENVY x360, Intel(R) i7-10510U CPU @ 1,8 GHz, 4 cores, 8 threads, 16GB RAM, win 10 home 64 bit
11	DESKTOP 2x Intel(R) Xeon(R) Gold 6246 CPU @ 3.30GHz, 384 GB RAM, Linux Debian 11	22	DESKTOP FUJITSU_YLRR desk, Intel(R) i5-4590 CPU 3,30 GHz, 4 cores, 4 threads, RAM 8GB, Windows 10 pro, 64 bit.

Fig. 2. The comparison of the execution times of the tested algorithms on different computers.

Based on the results presented in Fig. 2, it is clear that the low-power ARM processors of small mobile computers matched the performance of multi-core desktop processors (AMD Threadriper series).

References

[1] Goldstein, Larry J, A history of the prime number theorem, The American Mathematical Monthly, Taylor & Francis, 1973.

[2] Jakubski A, Piątkowski J, Perliński R., Efficient vote encoding and counting protocols based on the Chinese remainder theorem and multiparty computation, 16th Conference on Mathematical Modeling in Physics and Engineering, 2025.

[3] Koukoulopoulos, Dimitris, The distribution of prime numbers, American Mathematical Soc., 2019.

[4] Rabin, Michael O, Probabilistic algorithm for testing primality, Journal of number theory, Elsevier 1980.

[5] Wells, David, Prime numbers: the most mysterious figures in math, Turner Publishing Company, 2011.

[6] Arias de Reyna, Juan, and Jérémy Toulisse. "The *n*-th prime asymptotically." Journal de théorie des nombres de Bordeaux 25.3 (2013): 521-555.

[7] Axler, Christian. "New estimates for the n^{th} prime number." arXiv preprint arXiv:1706.03651 (2017).