Notes on Number Theory and Discrete Mathematics
Print ISSN 1310–5132, Online ISSN 2367–8275
Vol. 25, 2019, No. 3, 198–206
DOI: 10.7546/nntdm.2019.25.3.198-206

On the software computation of the formulae for the n-th prime number

Dimitar G. Dimitrov

Faculty of Mathematics and Informatics
Sofia University
5 James Bourchier Str., Sofia, Bulgaria
e-mail: `dgdimitrov@fmi.uni-sofia.bg`

Received: 27 January 2019 **Revised:** 14 September 2019 **Accepted:** 16 September 2019

Abstract: Many formulae for calculating the n-th prime number exist. In this paper, a comparison of the computation time of different existing formulae is made.
Keywords: Prime number, Arithmetic formula, Comparison, Software computation.
2010 Mathematics Subject Classification: 11A41, 11A25, 11-04.

1 Introduction

A primary focus of number theory is the study of prime numbers. The problem of finding a formula for calculating the n-th prime number is very old and has always been of interest to researchers.

Following are some well-known formulae:

$$p_n = 1 + \sum_{m=0}^{2^n} \left\lfloor \left\lfloor \frac{n}{\sum_{j=1}^{m} \left\lfloor \cos^2 \pi \frac{(j-1)!+1}{j} \right\rfloor} \right\rfloor^{1/n} \right\rfloor \tag{1}$$

$$p_n = 1 + \sum_{m=0}^{2^n} \left\lfloor \left\lfloor \frac{n}{1 + \sum\limits_{j=2}^{m} \left\lfloor \frac{\sin^2 \frac{\pi\{(j-1)!\}^2}{j}}{\sin^2\left(\frac{\pi}{j}\right)} \right\rfloor} \right\rfloor^{1/n} \right\rfloor \tag{2}$$

$$p_n = 1 + \sum_{m=0}^{2^n} \left\lfloor \left\lfloor \frac{n}{1 + \sum\limits_{j=2}^{m} \left\lfloor \frac{(j-1)!+1}{j} - \left\lfloor \frac{(j-1)!}{j} \right\rfloor \right\rfloor} \right\rfloor^{1/n} \right\rfloor \tag{3}$$

$$p_n = \left\lfloor 1 - \frac{1}{\log 2} \log\left(-\frac{1}{2} + \sum_{d|P_{n-1}} \frac{\mu(d)}{2^d - 1} \right) \right\rfloor \tag{4}$$

where μ is the Möbius function and P_{n-1} is the product of all previously calculated primes $p_1, ..., p_{n-1}$.

Formulae (1), (2) are defined by Willans [10], (3) is a modification which uses an alternative formula for the prime-counting function π introduced by J. Minač [10], and (4) is a recurrent formula by J. M. Gandhi [6].

In [2–5] K. Atanassov introduced the following 10 formulae:

$$p_n = \sum_{i=0}^{C(n)} sg\left(n - \sum_{j=2}^{i} \overline{sg}(j - 1 - \varphi(j)) \right) \tag{5}$$

$$p_n = \sum_{i=0}^{C(n)} sg\left(n - \sum_{j=2}^{i} \overline{sg}(\psi(j) - j - 1) \right) \tag{6}$$

$$p_n = \sum_{i=0}^{C(n)} sg\left(n - \sum_{j=2}^{i} \overline{sg}(\sigma(j) - j - 1) \right) \tag{7}$$

$$p_n = \sum_{i=0}^{C(n)} sg\left(n - \sum_{j=2}^{i} fr\left(\frac{j}{(j-1)!} \right) \right) \tag{8}$$

$$p_n = \sum_{i=0}^{C(n)} sg\left(n - \sum_{j=2}^{i} \overline{sg}(j - \eta(j))\right) \tag{9}$$

$$p_n = \sum_{i=0}^{C(n)} sg\left(n - \sum_{j=2}^{i} \left\lfloor \frac{1}{\delta(j)} \right\rfloor\right) \tag{10}$$

$$p_n = \sum_{i=0}^{C(n)} sg\left(n - \sum_{j=2}^{i} \left\lfloor \frac{1}{\tau(j) - 1} \right\rfloor\right) \tag{11}$$

$$p_n = \sum_{i=0}^{C(n)} sg\left(n - \sum_{j=2}^{i} \left\lfloor \frac{1}{\overline{\omega}(j)} \right\rfloor\right) \tag{12}$$

$$p_n = \sum_{i=0}^{C(n)} sg\left(n - \sum_{j=2}^{i} \left\lfloor \frac{1}{\Omega(j)} \right\rfloor\right) \tag{13}$$

$$p_n = \sum_{i=0}^{C(n)} sg\left(n - \pi_{\mathcal{P}}(i)\right) \tag{14}$$

where $n = \prod_{i=1}^{k} p_i^{\alpha_i}$ is the canonical representation of a natural number $n > 1$,

$$C(n) = \left\lfloor \frac{n^2 + 3n + 4}{4} \right\rfloor,$$

$$sg(x) = \begin{cases} 0, & \text{if } x \le 0 \\ 1, & \text{if } x > 0 \end{cases},$$

$$\overline{sg}(x) = \begin{cases} 0, & \text{if } x \neq 0 \\ 1, & \text{if } x = 0 \end{cases},$$

$$\varphi(n) = \prod_{i=1}^{k} p_i^{\alpha_i - 1}(p_i - 1),$$

$$\psi(n) = \prod_{i=1}^{k} p_i^{\alpha_i - 1}(p_i + 1),$$

$$\sigma(n) = \prod_{i=1}^{k} \frac{p_i^{\alpha_i + 1} - 1}{p_i - 1},$$

$$fr\left(\frac{p}{q}\right) = \begin{cases} 0, & \text{if } \dfrac{p}{\gcd(p,q)} = 1 \\ 1, & \text{else} \end{cases},$$

$$\eta(n) = \sum_{i=1}^{k} \alpha_i p_i,$$

$$\delta(n) = \prod_{i=1}^{k} \alpha_i p_1^{\alpha_1} ... p_{i-1}^{\alpha_{i-1}} p_i^{\alpha_{i-1}} p_{i+1}^{\alpha_{i+1}} ... p_k^{\alpha_k},$$

$$\tau(n) = \prod_{i=1}^{k} (1+\alpha_i),$$

$$\overline{\omega}(n) = \begin{cases} \min(\alpha_1, ..., \alpha_k)k, & \text{if } k > 1 \\ 1, & \text{if } k = 1 \end{cases},$$

$$\Omega(n) = \sum_{i=1}^{k} \alpha_i,$$

$$\pi_{\mathcal{P}}(n) = \begin{cases} 0, & \text{if } n < 2 \\ \pi_{\mathcal{P}}(n-1) + \mathcal{P}(n), & \text{else} \end{cases},$$

$$\mathcal{P}(n) = \begin{cases} 1, & \text{if } n \text{ is prime} \\ 0, & \text{if } n \text{ is composite} \end{cases}.$$

We will also include some recently developed formulae. S. M. Ruiz introduced the following three formulae [11, 12]:

$$p_n = 1 + \sum_{k=1}^{\lfloor 2n\log n + 2 \rfloor} \left(1 - \left\lfloor \frac{1}{n} \sum_{j=2}^{k} \left(1 + \left\lfloor \frac{1}{j}\left(2 - \sum_{s=1}^{j} \left(\left\lfloor \frac{j}{s} \right\rfloor - \left\lfloor \frac{j-1}{s} \right\rfloor \right)\right) \right\rfloor \right) \right\rfloor \right), \tag{15}$$

$$p_n = 1 + \sum_{k=1}^{\lfloor 2n\log n + 2 \rfloor} \left(1 - \left\lfloor \frac{1}{n} \sum_{j=2}^{k} \left\lfloor \frac{\operatorname{lcm}(1,2,...,j)}{j \cdot \operatorname{lcm}(1,2,...,j-1)} \right\rfloor \right\rfloor \right), \tag{16}$$

$$p_n = \lfloor n\log n \rfloor + \sum_{k=\lfloor n\log n \rfloor}^{\lfloor n\log n + n(\log(\log n) - 0.5) + 3 \rfloor} \left(1 - \left\lfloor \frac{1}{n} \sum_{j=2}^{k} \left\lfloor \frac{\operatorname{lcm}(1,2,...,j)}{j \cdot \operatorname{lcm}(1,2,...,j-1)} \right\rfloor \right\rfloor \right). \tag{17}$$

The newest formula in the present research was developed by I. Kaddoura and S. Abdul-Nabi [7]:

$$p_n = 3 + 2\left\lfloor n\log n\right\rfloor - \sum_{x=7}^{\lfloor 2n\log n\rfloor + 2} \left\lfloor \frac{4 + \sum_{j=1}^{\lfloor \frac{x-1}{6}\rfloor} \lfloor S(6j+1)\rfloor + \sum_{j=1}^{\lfloor \frac{x+1}{6}\rfloor} \lfloor S(6j-1)\rfloor}{n} \right\rfloor, \tag{18}$$

where

$$S(x) = -\frac{\sum_{k=1}^{\left\lfloor \frac{\lfloor\sqrt{x}\rfloor}{6}\right\rfloor + 1} \left(\left\lfloor \left\lfloor \frac{x}{6k+1} \right\rfloor - \frac{x}{6k+1} \right\rfloor + \left\lfloor \left\lfloor \frac{x}{6k-1} \right\rfloor - \frac{x}{6k-1} \right\rfloor \right)}{2\left(\left\lfloor \frac{\lfloor\sqrt{x}\rfloor}{6} \right\rfloor + 1 \right)}.$$

2 Methodology

All of the listed formulae have been implemented in C++ strictly following their definitions. No optimizations that change their definitions have been made.

The time used for calculating some of the mathematical functions such as $\mu(d)$ greatly depends on the specific implementations chosen. A list of all implementation details follows.

Some formulae directly or indirectly check whether a given natural number is prime or composite. An efficient AKS primality test [1] has been chosen. The algorithm requires polynomial time.

The following pseudocode represents the algorithm used to calculate the Möbius function [8]:

```
function MOEBIUS($n$)                                    ▷ Where $n \in \mathbb{N}$ and $n > 0$
    if $n = 1$ then return 1
    end if
    if $n = 2$ then return -1
    end if
    $p \leftarrow 0$
    for all $i \in \{2\} \cup \{m \in \mathbb{N} | m \geq 3 \land m \bmod 2 = 1 \land m^2 \leq n\}$ do
        if $n$ mod $i$ = 0 then
            $n \leftarrow n/i$
            $p \leftarrow p + 1$
            if $n$ mod $i$ = 0 then return 0
            end if
        end if
    end for
    if $p$ mod 2 = 0 then
        return -1
    else
```

```
        return $1$
    end if
end function
```

The following is a pseudocode for finding the prime factorization of a natural number n [9]:

```
function DECOMPOSE($n$)                                   ▷ Where $n \in \mathbb{N}$ and $n > 1$
    $k \leftarrow 0$
    for all $i \in \{2\} \cup \{m \in \mathbb{N} | m \geq 3 \wedge m \bmod 2 = 1 \wedge m^2 \leq n\}$ do
        $c \leftarrow 0$
        while $n \bmod i = 0$ do
            $c \leftarrow c + 1$
            $n \leftarrow n/i$
        end while
        if $c > 0$ then
            $result[k] \leftarrow (i, c)$
            $k \leftarrow k + 1$
        end if
    end for
    if $n > 2$ then
        $result[k] \leftarrow (n, 1)$
        $k \leftarrow k + 1$
    end if
    return $result$
end function
```

The number x^n, where $x \in \mathbb{Z}$ and $n \in \mathbb{N}$, is calculated using the following well known fast exponential algorithm:

$$x^n = \begin{cases} 1, & \text{if } n = 0 \\ x.x^{n-1}, & \text{if } n \text{ is odd} \\ (x^{\frac{n}{2}})^2, & \text{if } n \text{ is even} \end{cases},$$

The greatest common divisor (gcd) of two positive natural numbers a and b is calculated using the Stein's binary GCD algorithm [13]. It is very efficient compared to the conventional Euclidean algorithm. This algorithm is also used in the implementation of the least common multiple function used in (16) and (17).

The source code was compiled with the gcc compiler on Ubuntu 16.04 LTS operating system. Testing of the formulae was performed on a desktop computer with an Intel i7 CPU.

1024-bit integer data types from the Boost Multiprecision Library were used (`uint1024_t` and `int1024_t`) so the implementation can handle large intermediate values such as factorials and powers of two. For example, the largest integer for which factorial can be calculated using 32 bits is only 12, using 128 bits – 34, and so on. As a result, larger data types are needed even for relatively small values of n.

Formula	$n = 4$	$n = 9$	$n = 10$	$n = 20$	$n = 30$
(1)	0.000187	3.256814	27.839838	time out	time out
(2)	0.000170	3.429308	28.946541	time out	time out
(3)	0.000135	3.396208	28.492246	time out	time out
(4)	0.000039	1.055240	23.841273	time out	time out
(5)	0.000039	0.000388	0.000578	0.008815	0.044728
(6)	0.000038	0.000383	0.000579	0.008401	0.044321
(7)	0.000041	0.000492	0.000839	0.010981	0.055538
(8)	0.000167	0.005568	0.008969	0.605112	7.757162
(9)	0.000027	0.000333	0.000451	0.007046	0.036736
(10)	0.000030	0.000385	0.000545	0.008474	0.043577
(11)	0.000025	0.000285	0.000404	0.006202	0.032856
(12)	0.000019	0.000264	0.000371	0.005800	0.030580
(13)	0.000016	0.000229	0.000335	0.005386	0.027922
(14)	0.000006	0.000012	0.000014	0.000067	0.000128
(15)	0.000166	0.004771	0.008037	0.116386	0.578732
(16)	0.003338	0.210815	0.418192	15.278138	time out
(17)	0.000508	0.026317	0.040497	1.371085	10.674773
(18)	0.000039	0.000441	0.000656	0.004996	0.017033

Table 1. Results for very small values of n

Each combination of a formula and a value of n was executed 10 times and then the average time in seconds was calculated.

3 Results

Table 1 shows the measured times for small values of n. A time out was set to 60 seconds.

As expected, formulae (1)$-$(4) which use sums with 2^n summands, as well as (8), (16) and (17) are significantly slower than the other formulae. Also for these formulae with increasing n, the number of digits required for intermediate numbers increases drastically and therefore calculations will be slower. For that reason, these formulae were excluded from further testing with larger values. Results can be seen in Table 2. Time out was set to 300 seconds.

It is important to note that the presented results are large because 1024-bit integer types were used. This was needed because, as already pointed, some formulae contain factorials and powers of two which require a large number of digits even for small values of n. If we use standard 32 or 64-bit integers, calculations will be faster. Table 3 shows the results when 64-bit data types are used.

Formula	$n = 100$	$n = 200$
(5)	7.611714	167.108847
(6)	7.499903	163.355355
(7)	9.037541	188.893689
(9)	6.611778	148.642090
(10)	7.491476	163.455926
(11)	6.104673	140.214233
(12)	5.820569	135.284282
(13)	5.541861	130.774057
(14)	0.001620	0.008684
(15)	50.872570	time out
(18)	0.494740	3.383555

Table 2. Results for larger values of n

Formula	$n = 10$	$n = 100$	$n = 200$	$n = 1000$	$n = 20000$
(5)	0.000030	0.309244	7.135392	time out	time out
(6)	0.000024	0.307875	7.115837	time out	time out
(7)	0.000030	0.371489	8.190629	time out	time out
(9)	0.000022	0.275384	6.708046	time out	time out
(10)	0.000036	0.322498	7.475469	time out	time out
(11)	0.000021	0.275238	6.627389	time out	time out
(12)	0.000021	0.258818	6.370391	time out	time out
(13)	0.000020	0.271519	6.572727	time out	time out
(14)	0.000002	0.000054	0.000321	0.027147	113.856200
(18)	0.000049	0.022168	0.158724	14.874442	time out

Table 3. Results for implementation with 64-bit data types

4 Conclusion

The time needed for calculating each formula greatly depends on software implementation details such as data types and algorithms for formulae like $\sin$, $\log$, μ, etc. The present research shows that Atanassov's formula (14) is the fastest one at least for values of n up to 20000. It inherits its efficiency as an immediate consequence of applying the AKS prime testing formula to find $\pi_{\mathcal{P}}(i)$. The most recent formula (18) developed by Kaddoura and Abdul-Nabi is also fast, and it took second place.

References

[1] Agrawal, M., Kayal., N., & Saxena, N. (2004). PRIMES is in P, *Annals of Mathematics*, 160 (2), 781–793.

[2] Atanassov, K. (2001). A new formula for the n-th prime number, *Comptes Rendus de l'Academie Bulgare des Sciences*, 54 (7), 5–6.

[3] Atanassov, K. (2009). A remark on an arithmetic function. Part 3, *Notes on Number Theory and Discrete Mathematics*, 15 (4), 23–27.

[4] Atanassov, K. (2013). A formula for the n-th prime number, *Comptes Rendus de l'Academie bulgare des Sciences*, 66, 4, 503–506.

[5] Atanassov, K., *Formulas for the n-th prime number*, Unpublished manuscript.

[6] Gandhi, J. (1971). Formulae for the nth prime, *Proc. Washington State Univ. Conf. on Number Theory*, Washington State Univ., 96–101.

[7] Kaddoura, I., & Abdul-Nabi S. (2012). On Formula to Compute Primes and the nth Prime, *Applied Mathematical Sciences*, 6, 76, 3751–3757.

[8] Mahapatra, S. (2018). *Program for Mobius Function*, GeeksForGeeks.org, Available online at: `https://www.geeksforgeeks.org/program-mobius-function/`.

[9] Rath B. (2019). *Efficient program to print all prime factors of a given number*, GeekForGeeks.org, Available online at: `https://www.geeksforgeeks.org/print-all-prime-factors-of-a-given-number/`.

[10] Ribenboim, P. (1995). *The New Book of Prime Number Records*, Springer, New York.

[11] Ruiz, S. M. (2000). A functional recurrence to obtain the prime numbers using the Smarandache Prime Function, *Smarandache Notions J.*, Vol. 11, p. 56.

[12] Ruiz, S. M. (2005). A new formula for the nth prime. *Smarandache Notions Journal*, Vol. 15.

[13] Stein, J. (1967). Computational problems associated with Racah algebra, *Journal of Computational Physics*, 1 (3), 397—405.