Notes on Number Theory and Discrete Mathematics
Print ISSN 1310–5132, Online ISSN 2367–8275
2022, Volume 28, Number 2, Pages 147–158
DOI: 10.7546/nntdm.2022.28.2.147-158

On wide-output sieves

Alessandro Cotronei

Department of Mathematics and Statistics,
University of Tromsø
N-9037 Tromsø, Norway
e-mail: a.cotronei@uit.no

Received: 9 November 2021 **Revised:** 17 February 2022
Accepted: 9 March 2022 **Online First:** 23 March 2022

Abstract: We describe and compare several novel sieve-like methods. They assign values of several functions (i.e., the prime omega functions ω and Ω and the divisor function d) to each natural number in the considered range of integer numbers. We prove that in some cases the algorithms presented have a relatively small computational complexity. A more detailed output is indeed obtained with respect to the original Sieve of Eratosthenes.
Keywords: Sieve theory, Sieve of Eratosthenes.
2020 Mathematics Subject Classification: 11N35, 11N36.

1 Introduction

Sieve theory is one of the most used way to find primes since the third century B.C.E. Despite faster methods and modifications are available [3, 9, 14], this old method is still in wide use today because of the simplicity of its implementation. Howevere there is a relevant downside in particular from its use: the big number of repeated operations.

We will start by showing an algorithm that solves this problem and introduce several novel ways of *sieving* natural numbers with a wider output (hence the title). We will show also several other algorithms that work in a similar way.

Sections of this paper are organized with respect to the output for each algorithm. We refer to the original sieve in the following form (Algorithm 1), that uses $O(n \log\log n)$ basic operations (i.e., additions) and $O(n)$ bits of storage [12].

Algorithm 1 Sieve of Eratosthenes

```
1: Input: $n$.
2: Consider an array of natural numbers $1, 2, \dots, n-1, n$.
3: $p \leftarrow 2$.
4: while $p \leq \sqrt{n}$ do
5:     Mark the lowest unmarked number of the list $p$ (2 in the first case) as prime.
6:     for $i$ multiple of $p$ not exceeding $n$ do
7:         Mark $i$ Composite.                    ▷ Alternatively cancel out $i$.
8: Output: Characteristic Prime function or equivalently, all prime numbers $\leq n$.
```

We discuss several characteristics for $n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$ with p_i distinct primes [8]:

- the little omega prime function $\omega(n) = k$. $\omega(1) = 0$;
- the big omega prime function $\Omega(n) = a_1 + a_2 + \cdots + a_k$. $\Omega(1) = 0$;
- the divisor function $d(n) = (a_1 + 1)(a_2 + 1) \cdots (a_k + 1)$. $d(1) = 0$.

We make also use of several other functions as the prime counting functions $\pi(x)$ and the generalized prime counting function $\pi_k^*(x)$, defined as the number of prime numbers $\leq x$ (respectively, positive integers $\leq x$ that can be written as product of not necessarily distinct of k prime numbers) [11]. We use some theorems as the Prime Number Theorem in the form: $\pi(x) = O\left(\frac{x}{\log x}\right)$, and a generalization in the form: $\pi_k^*(n) = O\left(\frac{x(\log\log x)^{k-1}}{(k-1)!\log x}\right)$ [11].

We also make use of Abel's Identity [11]:

$$\sum_{y \leq x} a(n) f(n) = A(x)f(x) - A(y)f(y) - \int_y^x A(t)f'(t)dt,$$

where $A(x) = 0$ if $x < 1$, f has continuous derivative over the considered interval and $A(x) = \sum_{n \leq x} a(n)$.

We give several estimates in terms of the big O notation [11] for computational complexity and bit use (memory consumption) for the algorithms.

We split this paper in three sections, each of them describes algorithms with a different function (respectively, ω, Ω, d) as output.

2 Method with output ω

2.1 First method

The first method uses a slight modification in the original sieve to evaluate the funcion $\omega(n)$ for each number in an interval of consecutive natural numbers that includes 1. It has the same computational cost with the algorithm from which it derives (if we consider the addition step as expensive as the *canceling out* step of the original algorithm). At the same time, the memory required for the implementation is higher because the algorithm stores integer numbers (corresponding to the ω function) instead of logical values (sometimes zeroes and ones). This will be true for each algorithm we present.

We want to remark that this algorithm does not discard the repeated operations of *canceling out* as in the original Sieve of Eratosthenes:

Algorithm 2 Modified Sieve of Eratosthenes 1 (With output ω)

```
1: Input: $n$.
2: Consider an array with indices in the natural numbers $2, 3, \ldots, n-1, n$ and an array of natural
   numbers and size $n$, we call this $A$.
3: $a \leftarrow 0$.
4: while $p \leq \sqrt{n}$ do
5:     Mark the lowest number of the list $p$ such that $a(p) \in \{0, 1\}$ (2 in the first case) as prime.
6:     for $i$ multiple of $p$ not exceeding $n$ do
7:         $a(i) \leftarrow a(i) + 1$.
8: Output: $A \equiv \omega\ \forall l \in \{1, 2, \ldots, n\}$.
```

It is clear that at the end of the algorithm the array a is coinciding with the function ω because every number of the array is increased by one (instead of being marked as true or false or canceled like in the original sieve) for each prime number lower or equal.

This method still uses $O(n \log\log n)$ additions as it does not contain modification with respect to the original algorithm, but a bigger amount of memory has to be assigned. We have to store $\omega(k)\ \forall i \in 1, 2, \ldots, n$ so the required maximum amount is $n \left\lceil \log_2 \left(\max_{1 \leq k \leq n} \omega(k) \right) \right\rceil$ that can be bounded by $O\left(n \log\left(\frac{\log(n)}{\log\log(n)}\right)\right)$ where we have used the fact that $\omega(n)$ is at most $O\left(\frac{\log(n)}{\log\log n}\right)$, [15].

If we assume that that memory allocation is without unused bits for each stored value, then the number of bits used is: $\sum_{k=2}^{n} \lceil \log_2 (\omega(k)) \rceil$. It is not our interest to give a more precise estimate for this quantity since an actual implementation of the corresponding method would not be practical.

3 Methods with output Ω

3.1 Second method

In this procedure, we determine for each number the corresponding Ω function by multiplying numbers with known Ω for prime numbers (so that the Omega value is increased by one by this procedure). We mainly work with intervals of natural numbers between consecutive powers of two. We call these sets I_n:

Definition 3.1 (I_n)**.** *We define the set I_n as $\{2^n, 2^n + 1, \ldots, 2^{n+1} - 2, 2^{n+1} - 1\}$ or equivalently $\{x \in \mathbb{N} : 2^n \leq x < 2^{n+1} - 1\}$.*

Remark 3.1. *It is obvious that $a \in I_n \iff \lfloor \log_2 a \rfloor = n$. Hence in particular $I_0 = \{1\}$, $I_1 = \{2, 3\}$.*

The following proposition gives a hint on how multiplied numbers between intervals behave:

Proposition 3.1. *If $m \in I_q$ and $n \in I_r$, then $mn \in I_{q+r} \cup I_{q+r+1}$.*

Proof. From the Hypotheses, we have $2^q \leq m < 2^{q+1}$ and $2^r \leq n < 2^{r+1}$. If follows that $2^{q+r} \leq mn < 2^{q+r+2}$ and the latter is equivalent to the thesis. □

We will use the following Lemma to obtain numbers with m prime factors:

Lemma 3.1 (Main Lemma for Method 2)**.** *If $m \in I_r$, m has k prime factors ($k \leq r$) and $q = Q(r,k)$, one of the following two statements holds:*

- *there exists a prime number in $p \in I_s, s \leq q$ and a number with $k-1$ prime factors $t \in I_{r-s}$ such that $pt = m$;*
- *there exists a prime number in $p \in I_s, s \leq q$ and a number with $k-1$ prime factors $t \in I_{r-s-1}$ such that $pt = m$.*

Proof. Let $m \in I_r$, m with k prime factors ($2 \leq k \leq r$) and let $q = Q(r,k)$.

We want to prove first that it has at least a prime factor $p \in I_s, s \leq q$. First of all we have that $l = \max\{p \text{ prime} : p \leq 2^{\frac{r+1}{k}}\}$, in particular we have $p \leq 2^{\frac{r+1}{k}}$ and if we take $\bar{p} = \min\{s \text{ prime}, s > p\}$, we have from the definition of $p, \bar{p} > 2^{\frac{r+1}{k}}$. If by absurd the smallest prime of m is $\bar{p}$, we have $m = p_1 \cdot \ldots \cdot p_k \geq \bar{p}^k > 2^{\frac{r+1}{k} \cdot k} = 2^{r+1}$. We have obtained an absurd since m would not belong anymore to the interval of the Hypotheses.

To prove the remaining part of the Lemma, we first notice that if we use the Fundamental Theorem of Arithmetic, it is obvious that if $pt = m$, then t must have exactly $k-1$ prime factors.

The final part of the lemma follows from Proposition 3.1. □

We have also the following definition:

Definition 3.2 (Q function)**.** *Let $m, n \in \mathbb{N}$, we define the function $Q(m,n)$ as follows: let $l = \max\{p \text{ prime} : p \leq 2^{\frac{m+1}{n}}\}$, then there exists unique $r \in \mathbb{N}$ such that $l \in I_r$, we define $Q(m,n) := r$ or equivalently (Remark 3.1) $Q(m,n) := \lfloor \log_2 l \rfloor$.*

We can now enunciate the following Algorithm 3:

Algorithm 3 Sieve-like method (with output Ω)

```
1: Input: $n$.
2: Consider a vector $B$ of size $2^{n+1}$.
3: for i=1 to n do
4:     for j=i-1 to 2 do                          ▷ This step must be skipped for $i = 1$.
5:         $r \leftarrow Q(i,j)$.
6:         for k=1 to r do
7:             For each $p$ prime in $I_k$ and for each $t$ with $j-1$ factors in $I_{i-k}$, store $pt$ as number
               with $j$ factors.                  ▷ Result is in $I_i$ or $I_{i+1}$.
8:         Store all remaining number in the interval $I_i$ as primes.
9: Output: $B \equiv \Omega$ for all numbers $< 2^n$ and some numbers m: $2^n \leq m < 2^{n+1}$.
```

If we assume that step 7 takes only one operation as set multiplication, the computational complexity is in the same order of magnitude with respect to:

$$\begin{aligned}\sum_{i=2}^{n}\sum_{j=i-1}^{2} 1+\sum_{k=1}^{\left\lfloor\frac{i+1}{j}\right\rfloor} 1 &\sim \sum_{i=2}^{n}\sum_{j=i-1}^{2}\left(1+\frac{i+1}{j}\right)\\ &\leq \sum_{i=2}^{n}(i-2+(i+1)\log(i))\\ &= O(n^2\log n),\end{aligned}$$

where we used again Abel's summation formula to evaluate the last sum.

This computational cost is very convenient, since the algorithm assigns the function ω to all the numbers between 1 and 2^n. In practical configurations, the implementation could be much more expensive since set multiplication could be not feasable for regular computers. If we consider Prime Number Theorem and its generalization concerning $\pi_k^*(x)$ (see the Introduction) to estimate roughly the number of prime numbers and numbers with distinct prime factors in the proper intervals (in particular we estimate the prime numbers in I_k by approximating them by $\pi(2^{k+1})-\pi(2^k)$ and numbers with $j-i$ prime factors in I_{i-k} in a similar way), we obtain:

$$\sum_{i=2}^{n}\sum_{j=i-1}^{2} 1+\sum_{k=1}^{\left\lfloor\frac{i+1}{j}\right\rfloor}\frac{2^i(k-1)}{(\log 2)^2(k^2+k)(j-2)!}\left(\frac{(\log\log 2^{i-k+1})^{j-2}}{i-k+1}-\frac{(\log\log 2^{i-k})^{j-2}}{i-k}\right).$$

Having this sum really difficult to evaluate, we estimate a bigger sum with the use of several majorizations:

$$\sum_{i=2}^{n}\sum_{j=i-1}^{2} 1+\sum_{k=1}^{\left\lfloor\frac{i+1}{j}\right\rfloor}\frac{2^i(k-1)}{(\log 2)^2(k^2+k)(j-2)!}\left(\frac{(\log\log 2^{i-k+1})^{j-2}}{i-k+1}\right)$$

$$\begin{aligned}&\sim \sum_{i=2}^{n}\sum_{j=2}^{i-1} 1+\frac{2^i}{(\log 2)^2(j-2)!}\sum_{k=1}^{\left\lfloor\frac{i+1}{j}\right\rfloor}\frac{1}{ki}\left(\log(i\log 2)\right)^{j-2}\\ &\sim \sum_{i=2}^{n}\sum_{j=2}^{i-1} 1+\frac{2^i}{(\log 2)^2(j-2)!}\sum_{k=1}^{\left\lfloor\frac{i+1}{j}\right\rfloor}\frac{1}{ki}\left(\log i\right)^{j-2}\\ &\overset{*}{\sim} \sum_{i=2}^{n}\left((i-2)+\frac{2^i}{i}\sum_{j=2}^{i-1}\frac{(\log i)^{j-2}}{(j-2)!}(\log(i+1)-\log j)\right)\\ &\overset{**}{\sim} \sum_{i=2}^{n}\left((i-2)+\frac{2^i}{i}\sum_{j=2}^{i-1}\frac{(\log i)^{j-1}}{(j-2)!}\right)\\ &\sim \sum_{i=2}^{n}\left(i-2+2^i\right)=O(2^{n+1}),\end{aligned}$$

we have used in particular, among other formulas, the logarithmic approximation of the harmonic series (*, see [11]) and Taylor expansion for the function e^x (**, see [1]).

We want to remark that if we consider $m = 2^n$, the computational complexities are respectively $O\left((\log_2 m)^2 \log\log_2 m\right)$, or equivalently $O\left((\log m)^2 \log\log m\right)$ and $O(2m)$, those amounts are smaller than the corresponding complexities in the Sieve of Eratosthenes. However this algorithm uses more multiplications, that can be more intrinsically expensive then additions, as used in the original sieve.

The memory consumption is given by the sum of the memory consumption for storing the values of Ω and, conveniently, the number of the interval interval I_n for each number. Both of these values can be approximated by $2^{n+1}\log_2(n+1)$, i.e., the number of numbers to be stored times the maximum number of bits needed to store the corresponding Ω function (respectively, the interval), at most $n+1$, see Proposition 3.4 (respectively, Remark 3.1).

The overall memory consumption is therefore proportional to $O(2^{n+1}\log(n+1))$. Again if we consider $m = 2^n$, the overall memory consumptions are proportional to $O\left(m\log\left(1+\log_2 m\right)\right)$ and $O\left(m\log\log m\right)$.

A few properties can make the implementation of the algorithm even more efficient. We suggest some of them, but other ideas could go in the direction of using the additional Ω values in the biggest interval computed by the algorithm to evaluate the function in the whole interval or avoiding redundant calculations by checking the range of the operations before they are performed:

Proposition 3.2. *If $m+1 \leq 2n$, then $Q(m,n) = 1$.*

From the hyphothesis we have $\dfrac{m+1}{n} \leq 2$ and $2^{\frac{m+1}{n}} \leq 2^2 = 4$.

Therefore $l = \max\{p \text{ prime} : p \leq 2^{\frac{m+1}{n}} \leq 4\}$ can contain only the primes 2 and 3, both belonging to I_1. We have reached the thesis.

The following proposition is a generalization of the previous one and can be proved similarly.

Proposition 3.3. *If $m+1 \leq n(t+1)$ then $Q(m,n) \leq t$.*

This proposition can avoid some calculations for the numbers with n not necessarily distinct factors in I_n:

Proposition 3.4. *Each set I_n contains exactly 2 numbers with n prime factors (numbers m such that $\Omega(m) = n$), namely 2^n and $2^{n-1}\cdot 3$. The same set I_n contains no numbers with at least $n+1$ prime factors.*

Proof. By use of the Fundamental Theorem of Arithmetic we notice that 2^n is the smallest number with the required characteristics because the existence of a smaller number with this properties would imply the existence of a prime number smaller than 2.

If we now notice that $2^{n-1}\cdot 3$ is another number with the required number of factors in the correct interval, we must prove that the interval cannot contain numbers with factors bigger than 3 repeated more then once and factors bigger than 5. But this is a consequence of the following inequalities that consider $p_i, i \in \{1,\dots,n\}$ as (not necessarily distinct) prime numbers:

$$\begin{aligned} p_1 \cdot p_2 \cdot \ldots \cdot p_n &\geq 2^{n-2}\cdot 3\cdot 3 \\ &= 2^{n-2}\cdot 9 > 2^{n-2}\cdot 8 = 2^{n+1}, \\ p_1 \cdot p_2 \cdot \ldots \cdot p_n &\geq 2^{n-1}\cdot 5 > 2^{n-2}\cdot 4 \\ &= 2^n. \end{aligned}$$

$\square$

We now give some tables that show some data on how numbers are repetedly counted as output of the algorithm (as $12 = 2 \cdot 6 = 3 \cdot 4$, where both 2 and 3 are indeed in I_1). In particular, Table 1 shows some numbers with repeated output, together with number of repetitions and corresponding Ω, Figure 1 shows graphically the number of repetitions in the output for numbers up to 2000.

Numbers with repeated output	12	18	24	30	36	42	45	48	50	54	60	63	66	70	72	75	78	84	90	96	98	102	105	108	110	114	120	126	130
Number of repetitions	2	2	2	2	2	2	2	2	2	2	2	2	2	3	2	2	2	2	2	2	2	2	3	2	2	2	2	2	2
Corresponding Ω	3	3	4	3	4	3	3	5	3	4	4	3	3	3	5	3	3	4	4	6	3	3	3	5	3	3	5	4	3

Table 1. Repeated output from Algorithm 2

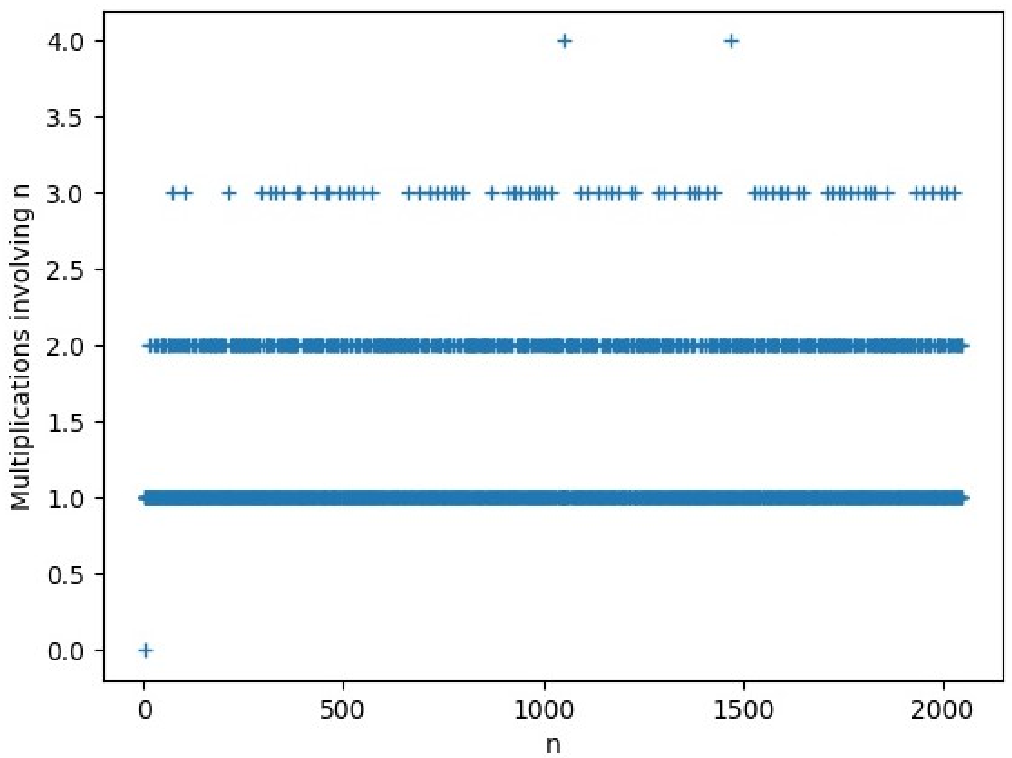

Figure 1. Number of times that n is output of Algorithm 2

Table 2 shows the smallest numbers with number repetitions between 1 and 6. For reference in the original Sieve of Eratosthenes, the smallest numbers are given by the primorial numbers $\left(\prod_{p \leq x} p\right)$ and are smaller than the considered number for this algorithm.

Smallest number with higher number of repetitions	Number of repetitions	Corresponding Ω
2	1	1
12	2	3
70	3	3
1050	4	5
182490	5	6
1771770	6	7

Table 2. Smallest numbers with amount of repetitions from Algorithm 2

3.2 Third and fourth method

In this section we describe some modifications of Eratosthenes' method that rely on the calculation of the number of prime divisors for each number in the interval $1, \dots, n$. The output is again the function Ω:

Algorithm 4 Modified sieve of Eratosthenes 2 (with output Ω)

```
1: Input: $n$.
2: Consider an array of natural numbers and size $n$, we call this $C$.
3: $a \leftarrow 0$.
4: $p \leftarrow 2$.
5: while $p \leq \sqrt{n}$ do
6:     Mark the lowest number of the list $p$ such that $a(p) \in \{0, 1\}$ (2 in the first case) as prime.
7:     Evaluate the multiplicity of $p$ for each number (which is multiple of $p$) up to at least $n$
       ($\left\lfloor \frac{n}{p} \right\rfloor$ numbers).
8:     for $i$ multiple of $p$ do
9:         $a(i) \leftarrow a(i) +$ multiplicity of $p$ in $i$.
10: Output: $C \equiv \Omega$ for all numbers $\leq n$.
```

By construction, this method yields as output the Ω function for all the numbers in the array. We need a convenient method to evaluate the multiplicity $v_p(n)$ of p in n and complete the algorithm. One possible way is to use the following properties of $v_p(n)$:

Proposition 3.5. $v_p(n) = 1$ *for each* p *prime,* n *multiple of* p *such that* $n < p^2$.

Proof. This is obvious as a multiple of p has $v_p(n) \geq 1$ and $v_p(n)$ cannot exceed 2 as $n < p^2$. □

Proposition 3.6. $v_p(n) = p(n + p^l)$ *for each* p *prime,* $l, n \in \mathbb{N}$, $n < p^l$.

Proof. Let $v_p(n) = l$, then there exists b coprime with p such that $n = p^l b$. If now we consider $n + p^k = p^l b + p^k = p^l(b + p^{k-l})$ and since the last multiplicand is coprime with p in any case with $n < p^l$, we have the thesis. □

The following algorithm follows from the last propositions. We can generate indeed the prime multiplicity of p for each multiple of p by copying properly some values. A small number of other operations is needed:

Algorithm 5 Prime Multiplicity Generation 1

```
1: Input: $p, n$.
2: Consider an array $A$ of natural numbers and size $\left\lfloor \frac{n}{p} \right\rfloor$.
3: $b \leftarrow \lfloor \log_p(n) \rfloor$.
4: Assign 1 to the first $p-1$ entries of $A$.    ▷ Unless they exceed the size of A, in that case $A \leftarrow 1$ and exit.
5: for $i$ from 2 to $b$ do
6:     Assign the correct value ($\log_p s$) to the first unassigned entry $s$ of $A$.
7:     Copy the first assigned numbers in the interval $[1, p^i]$ $p$ times.
8:     Delete the last number of the array A.
9: Copy the first assigned numbers in the interval $[1, p^b]$ until the number $\left\lfloor \frac{n}{p} \right\rfloor$ is reached.
10: if $\left\lfloor \frac{n}{p} \right\rfloor = p^l$ for some $l \in \mathbb{N}$ then
11:     Change the last entry of the array to $l$.
12: Output: Multiplicity of $p$ for all multiples of $p$ smaller than $n$.
```

Another possibility, that relies on the multiplication of prime powers of every prime numbers in the interval by numbers which are not divisible by the prime number itself and store them with the exponent of the respective powers is the following:

Algorithm 6 Prime Multiplicity Generation 2

```
1: Input: $p, n$.
2: $b \leftarrow \lfloor \log_p(n) \rfloor$.
3: Consider an array $\bar{A}$ of natural numbers and size $\left\lfloor \frac{n}{p} \right\rfloor$.
4: for $i$ from 1 to $b$ do    ▷ This step could be empty.
5:     for $j$ not multiple of $p$, $p^{i-1}j \leq \left\lfloor \frac{n}{p} \right\rfloor$ do
6:         $\bar{A}(p^{i-1} * j) = i$.
7: Output: Multiplicity of $p$ for all multiples of $p$ smaller than $n$.
```

The computational complexity of the incomplete algorithm *Modified Sieve of Eratosthenes 2* is again in the order of $O(n \log \log n)$, as the cost of Eratosthenes' method. The two methods can be overlapped. In the first case (using *Prime Multiplicity Generation 1* as step 7) the additional complexity is given by:

$$\begin{aligned}\sum_{p\le n}\left(1+\sum_{i=2}^{\lfloor \log_p(n)\rfloor}(2+p)\right) &= \sum_{p\le n}1+(2+p)(\lfloor \log_p(n)\rfloor-1)\\ &= \sum_{p\le n}\left(-1-p+(2+p)\lfloor \log_p(n)\rfloor\right)\\ &= O\left(\frac{n^2}{\log n}\right),\end{aligned}$$

which makes the first part complexity of the algorithm negligible. This estimate has been made by using Abel's summation formula and Prime Number Theorem [2]. In particular the dominant term of the sum is $p\lfloor \log_p(n)\rfloor$. In the second case (Second algorithm as step 7), the computational complexity is again:

$$\begin{aligned}\sum_{p\le n}\left(1+\sum_{i=1}^{\lfloor \log_p(n)\rfloor}\sum_{j\ne kp, p^{i-1}j\le\lfloor\frac{n}{p}\rfloor}1\right) &\sim \sum_{p\le n}\left(1+\frac{np}{p-1}\sum_{i=1}^{\lfloor \log_p(n)\rfloor}\frac{1}{p^i}\right)\\ &\overset{***}{\sim}\sum_{p\le n}\left(1+\frac{np}{p-1}\left(\frac{1-\left(\frac{1}{p}\right)^{(1+\log_p n)}}{1-\frac{1}{p}}-1\right)\right)\\ &\sim \sum_{p\le n}\left(2+\frac{-p+1}{p}-\frac{1}{n}\right)=O\left(\frac{n^2}{\log n}\right).\end{aligned}$$

We have used an identity concerning geometric series (***), several approximations and again Abel's Identity. The amount of bits used is given by the logarithm of the maximum size of Ω in the considered interval times the dimension of the interval. If we consider $\Omega(n)=O(\log n)$ [15], the final bit consumption is $O(n\log\log n)$. These final estimates take into account the fact that the memory used from *Modified Sieve of Eratosthenes 2* is at most in the same order of magnitude.

4 Methods with output *d*

We can modify Algorithm 4 so that the output is the function d, we only change step 9 so the final output is multiplied instead of being summed:

Algorithm 7 Modified Sieve of Eratosthenes 3 (With output d)

```
1: Input: $n$.
2: Consider an array of natural numbers and size $n$, we call this $C$.
3: $a \leftarrow 0$.
4: $p \leftarrow 2$.
5: while $p \leq \sqrt{n}$ do
6:     Mark the lowest number of the list $p$ such that $a(p) \in \{0, 1\}$ (2 in the first case) as prime.
7:     Evaluate the multiplicity of $p$ for each number (which is multiple of $p$) up to at least $n$ $\left(\left\lfloor \frac{n}{p} \right\rfloor \text{numbers}\right)$.
8:     for $i$ multiple of $p$ do
9:         $a(i) \leftarrow a(i) \cdot (1+\text{multiplicity of } p \text{ in } i)$.
10: Output: $C \equiv d$ for all numbers $\leq n$.
```

The computational complexity is comparable with those used from the original method. The maximum memory consumption is given by Gronwall's approximation [7] $d(n) = O(n \log \log n)$ and it is in the same order of magnitude with $O(n \log n)$.

5 Conclusions

The shown methods can evaluate several functions in a way that the primality of any number is obtained, together with other informations.

Of course, we can have algorithms that yield only prime numbers as output, as in classical sieve theory, by considering that prime numbers are characterized by $\omega(p) = 1$, $\Omega(p) = 1$ and $d(p) = 2$. In the end we have achieved additional precision at a bigger cost for memory consumption. Modern machines will be able to front this cost more and more easily.

We consider that a first relevant step could be to find a fast and parallel implementation for the algorithms, as already done with the classical counterparts [5].

We furthermore hypothesize that the methods shown in this paper could extend already existing theories that rely on basic sieve theory [4, 10, 13], or even theory of factorization [6].

These methods, however, are not published with the aim to be the best possible, but to give new directions in the study of sieve theory.

Acknowledgements

The author of this paper wishes to thank Roberto Amato (University of Messina), Francesco Belardo (University of Naples), Enzo Maria Li Marzi (University of Messina) and Thomas Slawig (University of Kiel) for their valuable mentorship. The author wishes to thank also Daniela Giumbo (University of Messina) and the anonymous referees for the valuable suggestions that led to the improvement of this manuscript.

References

[1] Abramowitz, M., & Stegun, I. A. (1970). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, New York.

[2] Apostol, T. (1976). *Introduction to Analytic Number Theory*, Undergraduate Texts in Mathematics, Springer-Verlag.

[3] Atkins, A. O. L., & Bernstein, J. (2004). Prime Sieves using Binary Quadratic Forms. *Mathematics of Computation*, 73(246), 1023–1030.

[4] Cox, D. N. (2008). Visualizing the Sieve of Eratosthenes. *Notices of the American Mathematical Society*, 55(5), 579–582.

[5] Da Costa, C. M. C., Sampaio, A., & Barbosa, J. G. (2014). Distributed Prime Sieve in Heterogeneous Computer Clusters. *Proceedings of the 14th International Conference on Computational Science and Its Applications*, Lecture Notes in Computer Science, Vol. 8582. Springer, Cham, 592–606.

[6] De Saint Guilhem, C. D., Makri, E., Rotaru, D., & Tanguy, T. (2021). The Return of Eratosthenes: Secure Generation of RSA Moduli using Distributed Sieving. *Proceedings of 2021 ACM SIGSAC Conference on Computer and Communications Security*, 594–609.

[7] Gronwall, T. H. (1913). Some asymptotic expressions in the theory of numbers. *Transactions of the American Mathematical Society*, 14, 113–122.

[8] Hardy, G. H., & Wright, E. M. (2006). *An Introduction to the Theory of Numbers*, 6th edition. Oxford University Press.

[9] Hinz, J. G. (2003). An application of algebraic sieve theory. *Archiv der Mathematik*, 80, 586–599.

[10] Li, B., Maltese, G., Costa-Filho, J. I., Pushkina, A. A., & Lvovsky, A. I. (2020). Optical Eratosthenes' sieve for large prime numbers. *Optics Express*, 28(8), 11965–11973.

[11] Nathanson, M. B. (2000). *Elementary Methods in Number Theory*, Springer-Verlag, New York.

[12] Sorenson, J. (1990). *An Introduction to Prime Number Sieves*, Computer Sciences Technical Report No. 909, Department of Computer Sciences, University of Wisconsin–Madison, January 2, 1990.

[13] Tytus, J. B. (2004). *The Random Sieve of Eratosthenes*, Science Direct Working Paper No. S1574-0358(04)70421-9.

[14] Ufuoma, O. (2019). A New and Simple Prime Sieving Technique for Generating Primes Ending with a Given Odd Digit. *Asian Research Journal of Mathematics*, 15(5), 1–11.

[15] Weisstein, Eric W. *Distinct Prime Factors.* From MathWorld–A Wolfram Web Resource. `https://mathworld.wolfram.com/DistinctPrimeFactors.html`