Mathematical Subject Classification: 11N25, 11S40
UDC 511

A. V. Lelechenko
I. I. Mechnikov Odessa National University

PARITY OF THE NUMBER OF PRIMES IN A GIVEN INTERVAL AND ALGORITHMS OF THE SUBLINEAR SUMMATION

Лелеченко А. В. Парність кількості простих чисел на заданному інтервалі та алгоритми сублінійного підсумовування. Пропонується алгоритм визначення парності кількості простих чисел на $[a, b] \subset [x, 2x]$, де $b-a \le x^{1/2+c}$ та $c \in (0, 1/2]$, за $O(x^{\max(c,7/15)+\varepsilon})$ операцій. Алгоритм базується на сублінійних методах підсумовування, розробка котрих становить основну частину статті. Доведено теорему щодо сублінійного підсумовування широкого классу мультиплікативних функцій.
Ключові слова: алгоритмічна теорія чисел, функція розподілу простих чисел, підсумовування мультиплікативних функцій, сублінійне підсумовування.

Лелеченко А. В. Четность количества простых чисел на заданном интервале и алгоритмы сублинейного суммирования. Предлагается алгоритм определения четности числа простых на отрезке $[a, b] \subset [x, 2x]$, где $b - a \le x^{1/2+c}$ и $c \in (0, 1/2]$, за $O(x^{\max(c,7/15)+\varepsilon})$ шагов. Алгоритм основан на сублинейных методах суммирования, разработка которых составляет основную часть статьи. Доказана теорема о сублинейном суммировании широкого класса мультипликативных функций.
Ключевые слова: вычислительная теория чисел, функция распределения простых чисел, суммирование мультипликативных функций, сублинейное суммирование.

Lelechenko A. V. Parity of the number of primes in a given interval and algorithms of the sublinear summation. An algorithm to determine the parity of the number of primes in an interval $[a, b] \subset [x, 2x]$, where $b - a \le x^{1/2+c}$ and $c \in (0, 1/2]$, in $O(x^{\max(c,7/15)+\varepsilon})$ steps is proposed. The algorithm is based on methods of the sublinear summation, which the primary part of the paper is devoted to. A theorem on the sublinear summation of a wide class of multiplicative functions is proven.
Key words: computational number theory, prime-counting function, summation of multiplicative functions, sublinear summation.

Introduction. How many operations are required to find any prime $p > x$ (not necessary the closest) for given x?

A direct approach is to apply AKS primality test [1], which was improved by Lenstra and Pomerance [5] to run in time $O(\log^{6+\varepsilon} x)$, on consecutive integers starting with x. Such method leads to an algorithm with average complexity $O(\log^{7+\varepsilon} x)$, because in average we should run AKS $\log x$ times before a next prime encounters.

But in the worst case available estimates of the complexity are much bigger; they depend on upper bounds of the gaps between primes. The best currently known result on the gaps between primes is by Baker, Harman and Pintz: for large enough x there exists at least one prime in the interval

$$[x, x + x^{0.525+\varepsilon}].$$

Thus we obtain that the worst case of an algorithm may need up to

$$O(x^{0.525+\varepsilon}) \gg x^{1/2}$$

operations.

One can propose another algorithm, which is distinct from the pointwise testing. Suppose that there is a test, which allows to determine whether a given interval $[a,b] \subset [x,2x]$ contains at least one prime in $A(x)$ operations. Then (starting with interval $[x,2x]$) we are able to find a prime $p > x$ in $A(x)\log x$ operations using a dichotomy.

A test to determine whether a given interval contains at least one prime can be built atop Lagarias—Odlyzko formula for $\pi(x)$ [6], which provides an algorithm with $O(x^{1/2+\varepsilon}) \gg x^{1/2}$ complexity. See [8] for more detailed discussion.

In [8] Tao, Croot and Helfgott offer a hypothesis that there exists an algorithm to compute $\pi(x)$ in $O(x^{1/2-c+\varepsilon})$ operations, where $c > 0$ is some absolute constant. This implies that a prime $p > x$ can be found in $O(x^{1/2-c+\varepsilon}) \ll x^{1/2}$ steps. Authors prove the following weaker theorem [8, Th. 1.2].

Theorem 1 (Tao, Croot and Helfgott, 2012)**.** *There exists an absolute constant $c > 0$, such that one can (deterministically) decide whether a given interval $[a,b]$ in $[x,2x]$ of length at most $x^{1/2+c}$ contains an* odd number *of primes in time $O(x^{1/2-c+o(1)})$.*

The aim of our paper is to prove the following result.

Theorem 2. *Let $[a,b] \subset [x,2x]$, $b-a \le x^{1/2+c}$, c is arbitrarily constant such that $0 < c \le 1/2$. Then a parity of $\#\{p \in [a,b]\}$ can be determined in time*

$$O(x^{\max(c,7/15)+\varepsilon}).$$

Main Results.

1. The general summation algorithm. Consider the summation

$$\sum_{n \le x} f(x),$$

where f is a multiplicative function, from the complexity's point of view.

Generally speaking, a property of the multiplicativity does not impose significant restrictions on pointwise computational complexity. Multiplicative functions can be both easily-computable (e. g., $f(n) = n^k$ for every k) and hardly-computable: e. g.,

$$f(p^\alpha) = \begin{cases} 2, & \text{if there are } p^\alpha \text{ consecutive zeroes in digits of } \pi \\ 1, & \text{otherwise.} \end{cases}$$

Luckily the vast majority of multiplicative functions, which have applications in the number theory, are relatively easily-computable.

Definition 1. *A multiplicative function f is called* easily-computable, *if for any prime p, integer $\alpha > 0$ and real $\varepsilon > 0$ the value of $f(p^\alpha)$ can be computed in time $O(p^\varepsilon \alpha^m)$ for some absolute constant m, depending only on f.*

Example 1. The (two-dimensional) divisor function $\tau_2(p^\alpha) = \alpha + 1$, the (two-dimensional) unitary divisor function $\tau_2^*(p^\alpha) = 2$, the totient function $\varphi(p^\alpha) = p^\alpha - p^{\alpha-1}$, the sum-of-divisors function $\sigma(p^\alpha) = (p^{\alpha+1} - 1)/(p-1)$, the Möbius function $\mu(p^\alpha) = [\alpha < 2](-1)^\alpha$ are examples of easily-computable multiplicative functions for any $m > 0$.

Example 2. Let $a(n)$ be the number of non-isomorphic abelian groups of order n. Then $a(p^\alpha) = P(\alpha)$, where $P(n)$ is a number of partitions of n. It is known [4, Note I.19], that $P(n)$ is computable in $O(n^{3/2})$ operations. Thus function $a(n)$ is an easily-computable multiplicative function with $m = 3/2$.

The number of rings of n elements is known to be multiplicative, but no explicit formula exists currently for $\alpha \geqslant 4$. See OEIS [9] sequences A027623, A037289 and A037290 for further discussions.

Example 3. The Ramanujan tau function τ_R is a rare example of an important number-theoretical multiplicative function, which is not easily-computable. The best known result is due to Charles [2]: a value of $\tau_R(p^\alpha)$ can be computed by p and α in $O(p^{3/4+\varepsilon} + \alpha)$ operations.

Surely pointwise product and sum of easily-computable functions are also easily-computable ones. The following statement shows that the Dirichlet convolution

$$(f \star g)(n) = \sum_{d|n} f(d) g(n/d)$$

also saves a property of easily-computability.

Lemma 1. *If f and g are easily-computable multiplicative functions, then*

$$h := f \star g$$

is also easily-computable.

Proof. By definition of easily-computable functions there exists m such that $f(p^\alpha)$ and $g(p^\alpha)$ can be both computed in $O(p^\varepsilon \alpha^m)$ time.

By definition of the Dirichlet convolution

$$h(p^\alpha) = \sum_{a=0}^{\alpha} f(p^a) g(p^{\alpha-a}).$$

This means that computation of $h(p^\alpha)$ requires

$$\sum_{a=0}^{\alpha} O(p^\varepsilon a^m + p^\varepsilon (\alpha - a)^m) \ll p^\varepsilon \alpha^{m+1}$$

operations.

Firstly, consider a trivial summation algorithm: calculate values of function pointwise and sum them up. For an easily-computable multiplicative function the majority of time will be spend on the factoring numbers from 1 to x one-by-one. But no

```
sum(ff, x) =
  Σ = 0
  A ← {k}_{k=1}^{x}
  B ← {1}_{k=1}^{x}
  for prime p ≤ √x
    F ← {ff(p, α)}_{α=1}^{log x/ log p}
    for k ← p, 2p, . . . , ⌊x/p⌋p
      α ← max{a | p^a|k}
      A[k] ← A[k]/p^α
      B[k] ← B[k] · F[α]
  for n ← 1, . . . , x
    if A[n] ≠ 1 ⇒ B[n] ← B[n] · ff(n, 1)
  for n ← 1, . . . , x
    Σ ← Σ + B[n]
  return Σ
```

Listing 1: Pseudocode of Algorithm M. Here $ff(p,\alpha)$ stands for the routine that effectively computes $f(p^\alpha)$.

polynomial-time factoring algorithm is currently known; the best algorithms (e. g., GNFS [10]) have complexities about

$$\exp\left((c+\varepsilon)(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}\right),$$

which is very expensive.

We propose a faster general method like the sieve of Eratosthenes. We shall refer to it as to *Algorithm M.*

Algorithm M. Consider an array A of length x, filled with integers from 1 to x, and an array B of the same length, filled with 1. Values of $f(n)$ will be computed in the corresponding cells of B.

For each prime $p \le \sqrt{x}$ cache values of $f(p), f(p^2), \ldots, f(p^{\lfloor \log x/\log p\rfloor})$ and take integers

$$k = p, 2p, 3p, \ldots, \lfloor x/p \rfloor p$$

one-by-one; for each of them determine α such that $p^\alpha \parallel k$ and replace $A[k]$ by $A[k]/p^\alpha$ and $B[k]$ by $B[k]\cdot f(p^\alpha)$.

After such steps cells of A contain 1 or primes $p > \sqrt{x}$. So for each n such that $A[n] \ne 1$ multiply $B[n]$ by $f(A[n])$.

Now array B contains computed values of $f(1), \ldots, f(n)$. Sum up its cells to end the algorithm.

Algorithm M can be encoded in pseudocode as it is shown in Listing 1.

Note that (similarly to the sieve of Eratosthenes) instead of the continuous array of length x one can manipulate with the set of arrays of length $\Omega(\sqrt{x})$. Inner cycles can be run independently of the order; they can be paralleled easily. Also one can compute several easily-computable functions simultaneously with a slight modification of Algorithm M.

Lemma 2. *If f is an easily-computable multiplicative function then Algorithm M runs in time $O(x^{1+\varepsilon})$.*

Proof. The description of Algorithm M shows that its running time is asymptotically lesser than

$$\sum_{p\le\sqrt{x}} p^\varepsilon \sum_{\alpha\le\log x/\log p} \alpha^m + \sum_{p\le\sqrt{x}} \frac{x}{p} + \sum_{\sqrt{x}<p\le x} p^\varepsilon \ll x^{1+\varepsilon}.$$

2. The fast summation.

Definition 2. *We say that function f* sums up with the deceleration a, *if function $F(x)=\sum_{n\le x} f(x)$ can be computed in $O(x^{a+\varepsilon})$ time.*

Denote the deceleration of f as $\operatorname{dec} f$. Notation $\operatorname{dec} f = a$ means exactly that there exists a method to sum up function f with the deceleration a (not necessarily there is no faster method).

Example 4. Lemma 2 shows that any easily-computable multiplicative function sums up with the deceleration 1.

Example 5. Function $f(n)=n^k$, $k\in\mathbb{Z}_+$, sums up in time $O(1)$, because there is an explicit formula for $F(x)$ using Bernoulli numbers. Thus its deceleration is equal to 0. Note that Dirichlet series of f is $\zeta(s-k)$, including case $\zeta(s)$ when $k=0$.

One can check that the same can be said about $f(n)=\chi(n)n^k$, where χ is an arbitrary multiplicative character modulo m. We just split $F(x)$ into m sums of powers of the elements of arithmetic progressions. In this case Dirichlet series equals to $L(s-k,\chi)$.

Example 6. The characteristic function of k-th powers, $k\in\mathbb{N}$, sums up in $O(1)$ trivially, so its deceleration equals to 0. Dirichlet series of such function is $\zeta(ks)$.

Consider now f such that $f(n^k)=\chi(n)$ and $f(n)=0$ otherwise, where χ is a multiplicative character. Then

$$\sum_{n=1}^{\infty} \frac{f(n)}{n^s} = L(ks,\chi).$$

Such function f also sums up in $O(1)$, because $F(x)=\sum_{n\le x^{1/k}} \chi(n)$ (see Example 5).

Generally, if function f has Dirichlet series $\mathcal{F}(s)$ and function g has Dirichlet series $\mathcal{F}(ks)$ then $\operatorname{dec} g = (\operatorname{dec} f)/k$.

Example 7. Consider Mertens function $M(x) := \sum_{n\le x}\mu(n)$. In [3] an algorithm of computation of $M(x)$ is proposed with time complexity $O(x^{2/3}\log^{1/3}\log x)$ and memory consumption $O(x^{1/3}\log^{2/3}\log x)$. We obtain $\operatorname{dec}\mu = 2/3$.

Note that Dirichlet series of μ equals to $1/\zeta(s)$.

One can see that a function μ_k such that $\mu_k(n^k)=\mu(n)$ and $\mu_k(n)=0$ otherwise sums up with the deceleration $2/(3k)$. Its Dirichlet series is $1/\zeta(ks)$.

Example 8. In [8] an algorithm of computation of $T_2(x) := \sum_{n\le x}\tau_2(n)$ in $O(x^{1/3+\varepsilon})$ time is described. Another algorithm with the same complexity may be found in [7], accompanied with detailed account and pseudocode implementation. Thus $\operatorname{dec}\tau_2 = 1/3$.

Theorem 3. *Let f and g be two easily-computable multiplicative functions, which sums up with decelerations $a := \operatorname{dec} f$ and $b := \operatorname{dec} g$ such that $a+b<2$. Then $h := f\star g$ sums up with the deceleration*

$$\operatorname{dec} h = \frac{1-ab}{2-a-b}.$$

Proof. Let

$$F(x) := \sum_{n\le x} f(n), \quad G(x) := \sum_{n\le x} g(n), \quad H(x) := \sum_{n\le x} h(n).$$

By definition of the Dirichlet convolution

$$H(x) = \sum_{n\le x}\sum_{d_1d_2=n} f(d_1)g(d_2) = \sum_{d_1d_2\le x} f(d_1)g(d_2).$$

Rearrange items:

$$\sum_{d_1d_2\le x} = \sum_{\substack{d_1\le x^c\\ d_2\le x/d_1}} + \sum_{\substack{d_1\le x/d_2\\ d_2\le x^{1-c}}} - \sum_{\substack{d_1\le x^c\\ d_2\le x^{1-c}}},$$

where an absolute constant $c\in(0,1)$ will be defined below in (2). Now

$$H(x) = \sum_{d\le x^c} f(d)G\left(\frac{x}{d}\right) + \sum_{d\le x^{1-c}} g(d)F\left(\frac{x}{d}\right) - F(x^c)G(x^{1-c}). \tag{1}$$

As far as we can calculate $f(1),\ldots,f(x^c)$ with Algorithm M in $O(x^{c+\varepsilon})$ steps, we can compute the first sum at the right side of (1) in time

$$O(x^{c+\varepsilon}) + \sum_{d\le x^c} O\left(\frac{x}{d}\right)^{b+\varepsilon} \ll x^{b+\varepsilon}\sum_{d\le x^c} d^{-b-\varepsilon} \ll$$
$$\ll x^{b+\varepsilon}x^{c(1-b-\varepsilon)} \ll x^{c+b(1-c)+\varepsilon}.$$

Similarly the second sum can be computed in $O(x^{1-c+ac+\varepsilon})$ operations. The last item of (1) can be computed in time $O(x^{ac+\varepsilon} + x^{b(1-c)+\varepsilon})$.

It remains to select c such that $c+b(1-c) = 1-c+ac$. Thus

$$c = \frac{1-b}{2-a-b}, \tag{2}$$

which implies the deceleration $(1-ab)/(2-a-b)$.

Example 9. Function $\sigma_k(n)$ maps n into the sum of k-th powers of its divisors. Thus $\sigma_k(n) = \sum_{d|n} d^k$, which is the Dirichlet convolution of $f(n)=n^k$ and $\mathbf{1}(n)=1$. So Example 5 and Theorem 3 shows that $\operatorname{dec}\sigma_k = 1/2$.

Example 10. Consider $r(n) = \#\{(k,l) \mid k^2 + l^2 = n\}$. It is well-known that $r(n)/4$ is a multiplicative function, and $\frac{1}{4}R(x) := \sum_{n\le x} r(n)/4$ is the number of integer points in the first quadrant of the circle of radius $\sqrt{x}$. Then $R(x)$ can be naturally computed in $O(x^{1/2})$ steps, so $\operatorname{dec} r = 1/2$.

Dirichlet series of $r(n)/4$ equals to $\zeta(s)L(s,\chi_4)$, where χ_4 is the single non-principal character modulo 4. This representation shows that $r(\cdot)/4 = \chi_4 \star \mathbf{1}$. Thus Example 5 together with Theorem 3 gives us another way to estimate the deceleration of r.

Example 11. By Möbius inversion formula for the totient function we have

$$\varphi(n) = \sum_{d|n} d\mu(n/d).$$

This representation implies that $\operatorname{dec}\varphi = 3/4$ (see Example 7 for $\operatorname{dec}\mu$). Jordan's totient functions have the same deceleration, because

$$J_k(n) = \sum_{d|n} d^k \mu(n/d).$$

Theorem 4. *Let f be an easily-computable multiplicative function. Consider*

$$f_k := \underbrace{f \star \cdots \star f}_{k \text{ factors}}.$$

Then

$$\operatorname{dec} f_k = 1 - \frac{1 - \operatorname{dec} f}{k}.$$

Proof. Follows from iterative applications of Lemma 1 and Theorem 3 and from the identities

$$\frac{1-a^2}{2-2a} = 1 - \frac{1-a}{2},$$
$$\frac{1 - a(k+a-1)/k}{2 - 1 + (1-a)/k - a} = 1 - \frac{1-a}{k+1}.$$

Example 12. For the multidimensional divisor function τ_k representations

$$\tau_{2k} = \underbrace{\tau_2 \star \ldots \star \tau_2}_{k \text{ factors}},$$
$$\tau_{2k+1} = \underbrace{\tau_2 \star \ldots \star \tau_2}_{k \text{ factors}} \star \mathbf{1}$$

imply that by Example 8 and Theorem 4 function τ_{2k} sums up with the deceleration $1 - 2/(3k)$, and τ_{2k+1} with the deceleration $1 - 2/(3k+2)$.

In other words

$$\operatorname{dec}\tau_k = \begin{cases} 1 - 4/(3k), & k \text{ is even}, \\ 1 - 4/(3k+1), & k \text{ is odd}. \end{cases} \tag{3}$$

Considering

$$\tau_{-k} = \underbrace{\mu \star \cdots \star \mu}_{k \text{ factors}},$$

we obtain by Example 7 and Theorem 4 that $\operatorname{dec}\tau_{-k} = 1 - 1/(3k)$.

Theorems 3 and 4 cannot provide the deceleration lower than 1/2 even in the best case. To overcome this barrier we should develop better instruments.

Theorem 5. *Let f and g be two easily-computable multiplicative functions, which sums up with decelerations $a := \operatorname{dec} f$ and $b := \operatorname{dec} g$ such that $a + b < 2$. Let*

$$h(n) := \sum_{d_1^{k_1} d_2^{k_2} = n} f(d_1) g(d_2). \tag{4}$$

Then h sums up with the deceleration

$$\operatorname{dec} h = \frac{1 - ab}{(1-a)k_2 + (1-b)k_1}.$$

Proof. Following the outline of the proof of Theorem 3 we obtain identity

$$H(x) = \sum_{d \le x^{c/k_1}} f(d) G\left(\sqrt[k_2]{x/d^{k_1}}\right) + \sum_{d \le x^{(1-c)/k_2}} g(d) F\left(\sqrt[k_1]{x/d^{k_2}}\right) - \\ - F(x^{c/k_1}) G(x^{(1-c)/k_2}).$$

Thus we need $y(x)$ operations to calculate $H(x)$, where

$$y(x) \ll \sum_{d \le x^{c/k_1}} \left(\frac{x}{d^{k_1}}\right)^{b/k_2} + \sum_{d \le x^{(1-c)/k_2}} \left(\frac{x}{d^{k_2}}\right)^{a/k_1} + \\ + x^{ac/k_1} + x^{b(1-c)/k_2} \ll \\ \ll x^{b/k_2 + (1 - bk_1/k_2) \cdot c/k_1} + x^{a/k_1 + (1 - ak_2/k_1) \cdot (1-c)/k_2} + \\ + x^{ac/k_1} + x^{b(1-c)/k_2}.$$

Substitution

$$c = \frac{(1-b)k_1}{(1-a)k_2 + (1-b)k_1}$$

completes the proof.

In terms of Dirichlet series identity (4) means that

$$\mathcal{H}(s) = \mathcal{F}(k_1 s) \mathcal{G}(k_2 s)$$

where

$$\mathcal{F}(s) = \sum_{n=1}^{\infty} \frac{f(n)}{n^s}, \quad \mathcal{G}(s) = \sum_{n=1}^{\infty} \frac{g(n)}{n^s}, \quad \mathcal{H}(s) = \sum_{n=1}^{\infty} \frac{h(n)}{n^s}.$$

One can prove (similarly to Lemma 1) that convolutions of form (4) save a property of the easily-computability.

Example 13. Function τ_2^* sums up with the deceleration 7/15, because

$$\tau_2^*(n) = \sum_{d^2 | n} \mu(d) \tau_2(n/d^2).$$

Example 14. As soon as

$$\tau_2^2(n) = \sum_{d^2|n} \mu(d)\tau_4(n/d^2),$$

we obtain $\operatorname{dec} \tau_2^2 = 5/9$.

The discussion in Examples 5, 6, 7 leads to the following general statement.

Theorem 6. *Let f be a multiplicative function such that*

$$\sum_{n=1}^{\infty} \frac{f(n)}{n^s} = \prod_{m=1}^{M_1} \zeta(k_m s)^{\pm 1} \prod_{m=1}^{M_2} z_m(l_m s - n_m), \tag{5}$$

where each of z_m is either ζ or $L(\cdot, \chi)$, M_1, M_2, $k_m, l_m, n_m \in \mathbb{N}$. Then f sums up in sublinear time: its deceleration is strictly less than 1.

Theorem 6 clearly shows that the concept of fast summation can be easily generalized over various quadratic fields. Following theorem is an example of such kind of results.

Theorem 7. *Consider the ring of Gaussian integers $\mathbb{Z}[i]$. Let*

$$\mathfrak{t}_k \colon \mathbb{Z}[i] \to \mathbb{Z}$$

be a k-dimensional divisor function on this ring. Let

$$\mathfrak{T}_k(x) := \sum_{N(\alpha) \le x} \mathfrak{t}_k(\alpha),$$

where $N(a+ib) = a^2 + b^2$. Then $\mathfrak{T}_k(x)$ can be computed in sublinear time.

Proof. It is well-known that

$$\frac{1}{4} \sum_{\alpha \in \mathbb{Z}[i]} \frac{\mathfrak{t}_k(\alpha)}{N^s(\alpha)} = \zeta^k(s) L^k(s, \chi_4) = \sum_{n=1}^{\infty} \frac{f(n)}{n^s},$$

where

$$f(n) := \sum_{N(\alpha)=n} \mathfrak{t}_k(\alpha).$$

But by Theorem 4

$$\operatorname{dec} \underbrace{\chi_4 \star \cdots \star \chi_4}_{k \text{ factors}} = 1 - 1/k.$$

By (3) we obtain that for even k

$$\operatorname{dec} f = \frac{1 - (1 - 1/k)\bigl(1 - 4/(3k)\bigr)}{1/k + 4/(3k)} = 1 - \frac{4}{7k}$$

and for odd k

$$\operatorname{dec} f = \frac{1 - (1 - 1/k)\bigl(1 - 4/(3k+1)\bigr)}{1/k + 4/(3k+1)} = 1 - \frac{4}{7k+1}.$$

3. Proof of the Theorem 2. The proof follows the outline of the proof of [8, Th. 1.2], but uses improved bound for the complexity of the computation of

$$T_2^*(x) := \sum_{n\le x} \tau_2^*(n).$$

Proof. Trivially we have

$$\sum_{a\le n\le b} \tau_2^*(n) = T_2^*(b) - T_2^*(a-1).$$

As soon as $\tau_2^*(n) = 2^{\omega(n)}$, where $\omega(n) = \sum_{p|n} 1$, all summands in the left side are divisible by 4, beside those, which corresponds to $n = p^j$. Moving to the congruence modulo 4, we obtain

$$2 \sum_{j=1}^{O(\log x)} \#\left\{p \in \left[a^{1/j}, b^{1/j}\right]\right\} \equiv T_2^*(b) - T_2^*(a-1) \pmod 4.$$

As far as $a > x$ and $b - a \le O(x^{1/2+c})$, then for $j > 1$ interval $\left[a^{1/j}, b^{1/j}\right]$ contains $O(x^c)$ elements; thus all such summands can be computed in $O(x^{c+\varepsilon})$ steps using AKS primality test [1]. The right side of the congruence is computable in $O(x^{7/15+\varepsilon})$ operations due to Example 13.

The discussion above shows that the desired quantity

$$\#\{p \in [a,b]\} \equiv \frac{T_2^*(b) - T_2^*(a-1)}{2} - \\ - \sum_{j=2}^{O(\log x)} \#\left\{p \in \left[a^{1/j}, b^{1/j}\right]\right\} \pmod 2$$

can be computed in $O(x^{\max(c,7/15)+\varepsilon})$ steps.

Conclusion. Further development of algorithms of the sublinear summation (e. g., summation of μ in arithmetic progressions) will lead to the generalization of Theorem 6 over broader classes of functions. Also one can investigate summation of f such that its Dirichlet series is infinite, but sparse product of form (5).

1. **Agrawal M.** PRIMES is in P [text] / M. Agrawal, N. Kayal, N. Saxena // Annals of Mathematics. – 2004. – Vol. 160, no. 2. – P. 781–793.

2. **Charles D. X.** Computing the Ramanujan tau function [text] / D. X. Charles // The Ramanujan Journal. – 2006. – Vol. 11, no. 2. – P. 221–224.

3. **Deléglise M.** Computing the summation of the Möbius function [yext] / M. Deléglise, J. Rivat // Exp. Math. – 1996. – Vol. 5, no. 4. – P. 291–295.

4. **Flajolet P.** Analytic combinatorics [text] / P. Flajolet, R. Sedgewick. – [S. l.] : Cambridge University Press, 2009. – 824 p.

5. **Lenstra Jr. H. W.** Primality testing with Gaussian periods H. W. Lenstra Jr., C. Pomerance. – 2011. – nov. – URL: http://www.math.dartmouth.edu/ carlp/aks041411.pdf.
6. **Lagarias J. C.** Computing $\pi(x)$: An analytic method [text] / J. C. Lagarias, A. M. Odlyzko // Journal of Algorithms. – 1987. – Vol. 8, no. 2. – P. 173–191.
7. **Sladkey R. A** Successive approximation algorithm for computing the divisor summatory function [text] / R. Sladkey. – 2012. – URL: http://arxiv.org/pdf/1206.3369v1.
8. **Tao T.** Deterministic methods to find primes [text] / T. Tao, E. Croot III, H. Helfgott // Math. Comp. – 2012. – Vol. 81, no. 278. – P. 1233–1246.
9. **The on-line** encyclopedia of integer sequences [text] / Ed. by N. J. A. Sloane. – [S. l. : s. n.]. – URL: http://oeis.org.
10. **The development** of the number field sieve [text] / Ed. by A. K. Lenstra, H. W. Lenstra. – [S. l.] : Springer Verlag, 1993. – Vol. 1554 of Lecture Notes in Mathematics.