Approximating the Number of Prime Factors Given an Oracle to Euler's Totient Function

Yang Du
Departments of EECS, CSE Division, University of Michigan, Ann Arbor, MI, USA

Ilya Volkovich
Computer Science Department, Boston College, Chestnut Hill, MA, USA

Abstract

In this work we devise the first efficient *deterministic* algorithm for approximating $\omega(N)$ – the number of prime factors of an integer $N \in \mathbb{N}$, given in addition oracle access to Euler's Totient function $\Phi(\cdot)$. We also show that the algorithm can be extended to handle a more general class of additive functions that "depend solely on the exponents in the prime factorization of an integer"[1]. In particular, our result gives the first algorithm that approximates $\omega(N)$ without necessarily factoring N. Indeed, all the previously known algorithms for computing or even approximating $\omega(N)$ entail factorization of N, and therefore are either randomized [12, 9] or require the Generalized Riemann Hypothesis (GRH) [10].

Our approach combines an application of Coppersmith's method for finding non-trivial factors of integers whose prime factors satisfy certain "relative size" conditions of [11], together with a new upper bound on $\Phi(N)$ in terms of $\omega(N)$ which could be of independent interest.

2012 ACM Subject Classification Mathematics of computing → Number-theoretic computations; Theory of computation → Problems, reductions and completeness; Theory of computation → Computational complexity and cryptography

Keywords and phrases Euler's Totient Function, Integer Factorization, Number of Prime Factors, Derandomization

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.17

Acknowledgements The authors would like to thank the anonymous referees for their detailed comments and suggestions on the previous version of the paper.

1 Introduction

The Fundamental Theorem of Arithmetic states that each $N \in \mathbb{N}$ admits a unique factorization into a product of powers of prime numbers $N = p_1^{\alpha_1} \cdot \ldots \cdot p_k^{\alpha_k}$. The *integer factorization problem* asks to compute those prime factors given N as an input. In addition to being a central problem in algorithmic number theory, integer factorization has a direct application to cryptography as all known approaches to break the RSA cryptosystem involve integer factorization. Indeed, there is no known efficient algorithm for the problem and it is believed to be computationally hard. In light of this hardness, a large body of work [10, 4, 14, 8, 11, 6, 7] has been dedicated to the study of the computational complexity of integer factorization when in addition the algorithm is given oracle access to some function $f : \mathbb{N} \to \mathbb{N}$ that provides "useful information" about N. On the practical side, oracles can model extra information on N obtained by means of side-channel attacks.

One such line of work considers integer factorization algorithms that are given oracle access to Euler's Totient function $\Phi(\cdot)$ (see Definition 12 for a formal definition). In a seminal work [10], Miller has shown that one can efficiently factor integers with an oracle to Φ. More

[1] This class and terminology were introduced and studied by Shallit & Shamir in [13].

precisely, the result is an efficient *deterministic* factorization algorithm, however its correctness relies on the Generalized Riemann Hypothesis (GRH). Subsequently, Long [9] and Rabin [12] replaced the GRH assumption with randomness, thus obtaining (unconditional) efficient *randomized* factorization algorithms. It has since been an open question to derandomize these algorithms unconditionally. In this paper we make another step towards the resolution of this problem.

1.1 Previous Results

Landau [8] and Woll [14] have shown that one can efficiently compute the square-free part of an integer[2] given an oracle access to $\Phi(\cdot)$. Building on this result, Żralek [15] and later on Hittmeir & Pomykała [6] achieved another milestone by exhibiting a reduction of integer factorization to the computation of Φ in deterministic subexponential-time. Morain et al. [11] applied Coppersmith's method (see e.g. [5]) to find non-trivial factors of integers whose prime factors satisfy certain "relative size" conditions.

1.2 Our Results

We take a slightly different approach: can we learn some "useful information" about N, given oracle access to $\Phi(\cdot)$? Formally, given an integer $N \in \mathbb{N}$ as an input, we would like to compute $f(N)$ for some "interesting" function $f : \mathbb{N} \to \mathbb{N}$ that can be efficiently computed given the complete factorization of N. A particular example of such a function is $\omega(N)$, which is defined as the number of prime factors of N. Our main result is an approximation algorithm for $\omega(N)$.

▶ **Theorem 1.** *There exist an efficient deterministic algorithm that given $N \in \mathbb{N}$ as an input, outputs an integer L satisfying: $\omega(N) \leq L \leq 3^{\omega(N)}$, given in addition oracle access to $\Phi(\cdot)$.*

▶ Remark 2. Although, $\omega(N) \leq \log N$, computing the actual value of $\omega(N)$ is *believed* to be as hard as factoring N (see e.g. [3, 1]).

▶ Remark 3. Our result is non-trivial when $\omega(N) = O(1)$ or more generally, when $\omega(N) = o(\log \log N)$.

To put our result in context, consider a decision version of the problem: given $N \in \mathbb{N}$ and oracle access to $\Phi(\cdot)$ (as before), decide if $\omega(N) = k$ for a **fixed** $k \in \mathbb{N}$. For $k = 1$ the problem corresponds to primality testing, which can trivially be solved given $\Phi(N)$ since $\Phi(N) = N - 1$ if and only if N is prime[3]. The case $k = 2$ is handled by a falklore result (See Lemma 13 for more details). For $k \geq 3$ the problem is still open.
It is also important to point out that one of the results of [6] provides an efficient deterministic factorization algorithm for N with $\omega(N) = O(1)$, if **in addition** the algorithm given the prime factorization of $\Phi(N)$. To the best of our knowledge, there is no known efficient deterministic algorithm to compute the prime factorization of $\Phi(N)$, even given oracle access to $\Phi(\cdot)$.

In [13], Shallit & Shamir introduced and studied the class of functions that "depend solely on the exponents in the prime factorization of an integer" (exponent-dependent functions, for short). They also showed that $d(N)$ - the number of positive divisors of N, is complete for this class under deterministic Turing reductions. More formally,

[2] See definition 10 for a formal definition.
[3] The breakthrough result of [2] gives an efficient deterministic primality test without $\Phi(N)$.

▶ **Lemma 4** ([13]). *Let $f : \mathbb{N} \to \mathbb{N}$ be an exponent-dependent function that can be efficiently (and deterministically) computed given the factorization of N. Then there exists an efficient deterministic algorithm that given $N \in \mathbb{N}$ as an input, outputs $f(N)$, given in addition oracle access to $d(\cdot)$.*

We observe that $\Phi(N)$ together with $\omega(N)$ are hard for the class of exponent-dependent functions.

▶ **Observation 5.** *For $f : \mathbb{N} \to \mathbb{N}$ as above, there exists a deterministic algorithm that given $N \in \mathbb{N}$ as an input, outputs $f(N)$, given in addition oracle access to $\Phi(\cdot)$ and $\omega(\cdot)$.*

As an important corollary, we obtain that if one can compute $\omega(N)$ **exactly**, given oracle access to $\Phi(\cdot)$ then $\Phi(\cdot)$ is (by itself) hard for the class of exponent-dependent functions. Such a result would constitute another important milestone on route to derandomization of the results of [10, 12, 9]. We show that a similar statement holds w.r.t approximations for *additive* functions: i.e. $f(N \cdot M) = f(N) + f(M)$ for coprime M and N (see Definition 14 for more details).

▶ **Theorem 6.** *Let $f : \mathbb{N} \to \mathbb{N}$ be an additive exponent-dependent function that can be efficiently (and deterministically) computed given the factorization of N. Then there exist an efficient deterministic algorithm that given $N \in \mathbb{N}$ as an input, outputs an integer L satisfying: $f(N) \leq L \leq 3^{f(N)}$, given in addition oracle access to $\Phi(\cdot)$.*

As an instantiation, we obtain an approximation algorithm for $\Omega(N)$ - the number of prime factors of N with multiplicity (see Example 15 for more details).

▶ **Corollary 7.** *There exist an efficient deterministic algorithm that given $N \in \mathbb{N}$ as an input, outputs an integer L satisfying: $\Omega(N) \leq L \leq 3^{\Omega(N)}$, given in addition oracle access to $\Phi(\cdot)$.*

Finally, we note that our main result is obtained using the following new upper bound on $\Phi(N)$ in terms of $\omega(N)$ which could be of independent interest.

▶ **Theorem 8.** *Let $N \in \mathbb{N}$ and let $k = \omega(N)$. Then $\Phi(N) \leq \left(\sqrt[k]{N} - 1\right)^k$.*

1.3 Techniques

In this section we give the outline of the proof of our main result.

First, we give a new upper bound on $\Phi(N)$ in terms of $k = \omega(N)$. Next, we show that there exists an efficient procedure that finds a non-trivial factor of N, if N has a "small" factor. Alternatively, if the procedure fails, then N cannot have a small factor. We complement this result by showing that if N still satisfies the upper bound for a "much" larger value of k then N must have a small factor.

Based on the above, our main algorithm operates as follows: given $N \in \mathbb{N}$ it first attempts to find a non-trivial factor D of N. If it succeeds, then the algorithm proceeds recursively on N/D and D. Otherwise (i.e. if the procedure fails), as discussed above, N cannot have a small factor. This, in turn, implies that the largest value of k satisfying the upper bound cannot be "much larger" than $\omega(N)$ as otherwise N must have had a small factor.

1.4 Organization

The paper is organized as follows: we start by some basic definitions and notations in Section 2. In that section we also prove Observation 5 and Theorem 6 as these follow immediately from the definitions. In Section 3 we give our main algorithm and a new upper bound on $\Phi(N)$ in terms of $\omega(N)$, thus proving Theorems 1 and 8. We show some numerical examples in Section 4. Finally, we conclude with discussion and open questions in Section 5.

2 Preliminaries

Let $\mathcal{P}$ denote the set of all primes. The Fundamental Theorem of Arithmetic states that each integer $N \in \mathbb{N}$ has a unique prime factorization. That is, N can be uniquely written as $N = \prod_{i=1}^{k} p_i^{\alpha_i}$ such that for all i: $p_i \in \mathcal{P}$ and $\alpha_i \in \mathbb{N}$.

▶ **Definition 9** (Roughness). *Let $r \in \mathbb{N}$. A number $N \in \mathbb{N}$ is called* r-rough *if all its prime factors are greater then or equal to r. That is, $\forall i : p_i \geq r$.*

▶ **Definition 10** (Radicals). *We define the* square-free part *or* radical *of N as* $\mathrm{rad}(N) \triangleq \prod_{i=1}^{k} p_i$. *$N$ is called* square-free *or* radical *iff $\forall i : \alpha_i = 1$. In other words, N is radical iff $N = \mathrm{rad}(N)$. We define the set $\mathbb{SQF} \subseteq \mathbb{N}$ as the set of all square-free integers.*

▶ **Definition 11** (Square-free decomposition). *A* square-free decomposition *of $N \in \mathbb{N}$ is a factorization of N as $N = N_1^1 N_2^2 N_3^3 \dots N_\ell^\ell$ such that $\gcd(N_i, N_j) = 1$ and $N_i \in \mathbb{SQF}$.*

By The Fundamental Theorem of Arithmetic, each integer admits a *unique* square-free decomposition. Furthermore, observe that $\ell \leq \log N$. In addition, $\mathrm{rad}(N) = N_1 N_2 N_3 \cdots N_\ell$.

▶ **Definition 12** (Euler's Totient Function). *$\Phi(N) : \mathbb{N} \to \mathbb{N}$ is defined as* $\Phi(N) \triangleq \prod_{i=1}^{k} p_i^{\alpha_i - 1} \cdot (p_i - 1)$.

▶ **Lemma 13.** *Below are some useful properties and facts.*

1. $\Phi(N) \leq N - 1$. *Equality holds if and only if N is prime.*
2. $\dfrac{\Phi(N)}{N} = \dfrac{\Phi(\mathrm{rad}(N))}{\mathrm{rad}(N)}$.
3. *Folklore: Let N be a product of two distinct primes $N = pq$. There exists an algorithm that given N and $\Phi(N)$, outputs p and q, in time* $\mathrm{polylog}(N)$.
   A sketch of the proof appears in Section A.
4. *[14, 8]: There exists an algorithm that given N and oracle access to $\Phi(\cdot)$, outputs the square-free decomposition of N, in time* $\mathrm{polylog}(N)$.

▶ **Definition 14** ([13]). *We call a function $f : \mathbb{N} \to \mathbb{C}$* exponent-dependent *if it depends solely on the exponents in the prime factorization N.*
In addition, we say that f is additive, *if $f(N \cdot M) = f(N) + f(M)$ for all coprime $N, M \in \mathbb{N}$.*

▶ **Example 15.** Prominent examples include the following functions.

1. $\Omega(N) \triangleq \sum_{i=1}^{k} \alpha_i$ - the number of prime factors of N with multiplicity.
2. $\omega(N) \triangleq k$ - the number of prime factors of N **without** multiplicity.
3. $\mu : \mathbb{N} \to \{-1, 0, 1\}$ - Möbius Function:

$$\mu(N) = \begin{cases} 0 & \text{if } N \text{ is not square-free} \\ (-1)^k & \text{if } N \text{ is a product of } k \text{ (distinct) primes} \end{cases}$$

4. $d(N) \triangleq \prod_{i=1}^{k} (\alpha_i + 1)$ - the number of positive divisors of N.

Here, $\Omega(N)$ and $\omega(N)$ are additive.

We remark that there is no known polynomial-time algorithm for computing any of the above functions. Indeed, they are *believed* to be as hard as (complete) integer factorization [3, 1]. Several relations between these (and other) functions have been established in [10, 13, 4, 14, 8, 11]. In particular, in [13] it was shown that $d(N)$ is complete for the class of exponent-dependent functions under Turing reductions, by showing that an oracle to $d(\cdot)$ can be used to compute $e(N) \triangleq \{\alpha_1, \ldots, \alpha_k\}$ - the **multiset** of exponents in the prime factorization of N. We observe that $e(N)$ can be easily constructed from a square-free decomposition of an integer N, given oracle access to $\omega(\cdot)$.

▶ **Observation 16.** *Let $N = N_1^1 N_2^2 N_3^3 \cdots N_\ell^\ell$ be the square-free decomposition of N. Then $e(N) = \{(i, \omega(N_i)) \mid \omega(N_i) > 0\}$. That is, $e(N)$ contains all the values $w(N_i)$ greater than 0, where each $w(N_i)$ appears i times.*

Observation 5 follows by combining the above observation with Part 4 of Lemma 13. The following is another observation immediate from the definition.

▶ **Observation 17.** *Let $N = N_1^1 N_2^2 N_3^3 \cdots N_\ell^\ell$ be the square-free decomposition of N and let $f : \mathbb{N} \to \mathbb{N}$ be an exponent-dependent additive function. Then $f(N) = \sum_{i=1}^{\ell} \omega(N_i) \cdot f(2^i)$.*

Theorem 6 follows by combining the above with Part 4 of Lemma 13 and Theorem 24.

▶ **Lemma 18.** *We have three inequalities that will be used in the later sections.*

1. *AM-GM Inequality. Let $x_1, \ldots, x_m \geq 0$: Then the arithmetic mean is greater then or equal to the geometric mean of these numbers:*

$$\frac{x_1 + \cdots + x_m}{m} \geq \sqrt[m]{x_1 \cdots x_m}.$$

2. *Multivariate Bernoulli Inequality. Let $0 \leq \varepsilon_1, \ldots, \varepsilon_m \leq 1$, then it follows that*

$$\prod_{i=1}^{m}(1 - \varepsilon_i) \geq 1 - \sum_{i=1}^{m} \varepsilon_i$$

3. *For any positive integer k, and $0 \leq x \leq \frac{1}{2k}$ and $\ell \geq 2k$, we have*

$$(1 - x)^\ell \leq 1 - kx.$$

Proof. We will show the proof of Part 3 of Lemma 18. Since $1 - x$ is less then 1 and $\ell \geq 2k$, then $(1 - x)^\ell \leq (1 - x)^{2k}$, we now only need to show that $(1 - x)^{2k} \leq 1 - kx$. We know that $(1-x)^{2k}$ is a convex function on $[0, 1]$ because the second derivative $2k(2k-1)(1-x)^{2k-2} > 0$. In addition, it is clear that both side of the inequality is evaluated to be 1 at $x = 0$, and the first derivative of $(1 - x)^{2k}$ is less then the first derivative of $1 - kx$ at $x = 0$. Therefore, we only need to show $(1 - x)^{2k} \leq 1 - kx$ is established at $x = \frac{1}{2k}$, which is

$$\left(1 - \frac{1}{2k}\right)^{2k} \leq e^{-1} \leq \frac{1}{2} = 1 - k \cdot \frac{1}{2k}.$$

We can see that when k goes from 1 to infinity, the left hand side approached to $1/e$ from the bottom, so it is always less then $1/2$. ◀

3 Algorithm and Technical Results

In this section we prove our main result (Theorem 1) and give a new upper bound on $\Phi(N)$ in terms of $\omega(N)$, thus proving Theorem 8. We first give the upper bound for the case of square-free integers.

▶ **Lemma 19.** *Let $N \in \mathbb{SQF}$ and let $k = \omega(N)$. Then* $\frac{\Phi(N)}{N} \leq \left(1 - \frac{1}{\sqrt[k]{N}}\right)^k$

Proof. Let $N = p_1 \cdots p_k$. Then by applying AM-GM inequality (Lemma 18) we get:

$$\left(\frac{\Phi(N)}{N}\right)^{1/k} = \sqrt[k]{\prod_{i=1}^{k}\left(1-\frac{1}{p_i}\right)} \leq \frac{\sum_{i=1}^{k}\left(1-\frac{1}{p_i}\right)}{k} = 1 - \frac{1}{k}\sum_{i=1}^{k}\frac{1}{p_i} \leq 1 - \sqrt[k]{\prod_{i=1}^{k}\frac{1}{p_i}} = 1 - \frac{1}{\sqrt[k]{N}}.$$

◀

Next, we extend the upper bound to arbitrary N. Theorem 8 follows from the next corollary.

▶ **Corollary 20.** *Let $N \in \mathbb{N}$ and let $k = \omega(N)$. Then* $\frac{\Phi(N)}{N} \leq \left(1 - \frac{1}{\sqrt[k]{\mathrm{rad}(N)}}\right)^k \leq \left(1 - \frac{1}{\sqrt[k]{N}}\right)^k$

Proof. Apply Lemma 19 on $\mathrm{rad}(N)$ together with Lemma 13, Part 2. ◀

Next, we show that there exists an efficient procedure that finds a non-trivial factor of N, if N has a small factor. Alternatively, if the procedure fails, then N cannot have a small factor. Our result relies on the following result of [11] that uses Coppersmith's method to find non-trivial factors of integers whose prime factors satisfy certain relative size conditions.

▶ **Lemma 21** ([11])**.** *Suppose $\omega(N) \geq 3$ and there exists $1 \leq r < \omega(N)$ such that*

$$\alpha_r \geq 2\sum_{i=r+1}^{\omega(N)} \alpha_i,$$

where $\alpha_i \triangleq \log_N(p_i)$. Then there exists an algorithm that given N and $\Phi(N)$ recovers the factor $D = \prod_{i=1}^{r} p_i$, in time $\mathrm{polylog}(N)$.

▶ **Theorem 22.** *Let $N \in \mathbb{SQF}$ and suppose N has a factor smaller then $N^{\frac{1}{3^{\omega(N)-1}}}$. Then there exists an algorithm that given N and $\Phi(N)$ as input, finds a non-trivial factor of N, in time* $\mathrm{polylog}(N)$.

Proof. Let $N = p_1 \cdots p_k$ with $p_1 > p_2 > \ldots > p_k$, and $\omega(N)$ is hence equal to k. Observe that the premises of the claim are equivalent to $p_k \leq N^{\frac{1}{3^{k-1}}}$, namely $\alpha_k \leq \frac{1}{3^{k-1}}$. It is sufficient to show that N satisfies that premises of Lemma 21. We suppose for contradiction that for all $1 \leq r \leq k-1$,

$$\alpha_r < 2\sum_{i=r+1}^{k} \alpha_i. \tag{1}$$

We claim that it implies that for all $1 \leq i \leq k-2$, we have $\alpha_{k-i} < 2 \cdot 3^{i-1} \cdot \alpha_k$. We prove this by induction. The base case $\alpha_{k-1} < 2\alpha_k$ follows directly from Equation 1 when we set $r = k-1$. Now, assume that for all $1 \leq i \leq m$, $\alpha_{k-i} < 2 \cdot 3^{i-1} \cdot \alpha_k$ is established, then for $i = m+1$ we have:

$$\alpha_{k-m-1} < 2 \sum_{i=k-m}^{k} \alpha_i < 2 \left[\sum_{i=1}^{m} \left(2 \cdot 3^{i-1} \cdot \alpha_k\right) + \alpha_k \right] = 2 \left[2 \cdot \frac{3^m - 1}{3-1} + 1 \right] \alpha_k = 2 \cdot 3^m \cdot \alpha_k.$$

Therefore,

$$1 = \alpha_1 + \cdots + \alpha_k < 2\left(3^{k-2} + 3^{k-1} + \ldots + 1\right)\alpha_k + \alpha_k = \left(2 \cdot \frac{3^{k-1} - 1}{3-1} + 1\right)\alpha_k = 3^{k-1} \cdot \alpha_k \leq 1$$

leading to a contradiction. ◀

We complement the above result by showing that (under certain technical conditions) if N satisfies the bound of Lemma 19 with a value of k "much" larger then $\omega(N)$ then N must have a small factor.

▶ **Lemma 23.** *Let $N \in \mathbb{SQF}$ and suppose that N is $(2\omega(N))$-rough. Furthermore, suppose that $\Phi(N)^{1/\ell} \leq N^{1/\ell} - 1$ for some $\ell \geq 2\omega(N)$. Then N has a factor smaller then $N^{1/\ell}$.*

Proof. As before, let $N = p_1 \cdots p_k$ with $p_1 > \cdots > p_k$ so $\omega(N) = k$. By the premises, $p_k \geq 2k$. Next, since $\Phi(N)^{1/\ell} \leq N^{1/\ell} - 1$ we have

$$\frac{\Phi(N)}{N} \leq \left(1 - \frac{1}{N^{1/\ell}}\right)^{\ell}.$$

By the Multivariate Bernoulli Inequality in Lemma 18 part 2, we have

$$\frac{\Phi(N)}{N} \geq 1 - \sum_{i=i}^{k} \frac{1}{p_i} \geq 1 - \frac{k}{p_k}.$$

Therefore, by combining the above we obtain:

$$\left(1 - \frac{k}{p_k}\right)^{1/\ell} \leq 1 - \frac{1}{N^{1/\ell}}.$$

By Lemma 18 part 3, since $\frac{1}{p_k} \leq \frac{1}{2k}$ and $\ell \geq 2k$ we have that:

$$1 - \frac{1}{p_k} \leq \left(1 - \frac{k}{p_k}\right)^{1/\ell}.$$

Therefore,

$$1 - \frac{1}{p_k} \leq 1 - \frac{1}{N^{1/\ell}} \qquad \Longrightarrow \qquad p_k \leq N^{1/\ell}.$$ ◀

Given $N \in \mathbb{N}$, our main algorithm operates as follows: first, it invokes the procedure in Theorem 22 attempting to find a non-trivial factor D of N. If it succeeds, then the algorithm proceeds recursively on N/D and D. Otherwise (i.e. if the procedure fails), as discussed above, N cannot have a small factor. This, in turn, implies that the largest value of k satisfying the bound in Lemma 19 has to be at most $3^{\omega(N)}$ as otherwise, by Lemma 23 N must have had a small factor.

▶ **Theorem 24.** *There exist an algorithm that given $N \in \mathbb{SQF}$ as an input, outputs an integer L satisfying: $\omega(N) \leq L \leq 3^{\omega(N)}$, given in addition oracle access to $\Phi(\cdot)$, in time* $\mathrm{polylog}(N)$. *The description of the algorithm is given below in Algorithm 1.*

Algorithm 1 Approximation of number of factors (ANF).

Function: $\mathrm{ANF}(N)$
Input: $N \in \mathbb{SQF}$ and oracle access to $\Phi(\cdot)$.
Output: An integer L satisfying: $\omega(N) \leq L \leq 3^{\omega(N)}$.
`// N is prime`
if $\Phi(N) = N - 1$ **then return** 1;
if *N has 2 prime factors* **then return** 2; / Invoking the algorithm from Lemma 13 Part 3.
for $M = 1$ **to** $2 \log N$ **do**
  if $M \mid N$ **then return** $\mathrm{ANF}(M) + \mathrm{ANF}(N/M)$;
Invoke the algorithm from Theorem 22 on N and $\Phi(N)$. Let D be the output;
if $D \mid N$ **then return** $\mathrm{ANF}(D) + \mathrm{ANF}(N/D)$;
`// Otherwise:`
$L = \max_{\ell} \Phi(N) \leq \left(N^{1/\ell} - 1\right)^{\ell}$;
return L;

Proof. The proof is by induction on $\omega(N)$. The base cases, i.e. $\omega(N) = 1, 2$, follow from Lemma 13, Parts 1 and 3. Now suppose $\omega(N) \geq 3$. We consider a few cases.

1. N has a factor $M \leq 2 \log N$.
   Observe that $M, N/M \in \mathbb{SQF}$. In addition, as $\omega(N) = \omega(M) + \omega(N/M)$ we have that $\omega(M), \omega(N/M) < \omega(N)$. Therefore, by the induction hypothesis:

   $$\omega(M) \leq \mathrm{ANF}(M) \leq 3^{\omega(M)} \text{ and } \omega(N/M) \leq \mathrm{ANF}(N/M) \leq 3^{\omega(N/M)}.$$

   Consequently:

   $$\mathrm{ANF}(N) = \mathrm{ANF}(M) + \mathrm{ANF}(N/M) \leq 3^{\omega(M)} + 3^{\omega(N/M)} \leq 3^{\omega(M)} \cdot 3^{\omega(N/M)} = 3^{\omega(N)}.$$

   and

   $$\mathrm{ANF}(N) = \mathrm{ANF}(M) + \mathrm{ANF}(N/M) \geq \omega(M) + \omega(N/M) = \omega(N).$$

2. The algorithm from Theorem 22 produces a factor D such that $D \mid N$.
   The claim follows by repeating the argument from the previous case.
3. $L < 2\omega(N)$. Then clearly, $L \leq 3^{\omega(N)}$.
4. W.l.o.g none of the above conditions hold.
   As $\omega(N) \leq \log N$ and N has no factors less then $2 \log N$, it follows that N is $(2\omega(N))$-rough. By Lemma 23, N has a factor smaller then $N^{1/L}$. On the other hand, by Theorem 22, N has no factors smaller then $N^{\frac{1}{3^{\omega(N)-1}}}$. Consequently, $L \leq 3^{\omega(N)-1} \leq 3^{\omega(N)}$.
   Finally, by Lemma 19: $\omega(N) \leq L$. ◀

Theorem 1 follows by combining the above Theorem with Part 4 of Lemma 13.

4 Examples

We use Algorithm 1 (ANF) to find an upper bound for the number of prime factors of a square-free integer N, given in addition $\Phi(N)$.

4.1 Example 1

Let us find the number of prime factors of

$$N = 50000000479987 = 3005349551 \times 127 \times 131,$$

where $\alpha_1 = 0.691869, \alpha_2 = 0.154557, \alpha_3 = 0.153574$. The algorithm will first find that it does not satisfy the base case, then iterate from $M = 1$ to $2 \log N (= 91)$. After that, it will invoke the algorithm from Theorem 22, since $\alpha_1 \geq 2(\alpha_2 + \alpha_3)$. That algorithm will return $p_1 = 3005349551$. Then it will return $\mathrm{ANF}(16637) + \mathrm{ANF}(3005349551) = 3$, since both will satisfy the base cases.

4.2 Example 2

We will find the number of prime factors of

$$N = 504203634089 = 389 \times 331 \times 283 \times 137 \times 101,$$

where $\alpha_1 = 0.221314$, $\alpha_2 = 0.215322$, $\alpha_3 = 0.209508$, $\alpha_4 = 0.182585$, $\alpha_5 = 0.171271$. The algorithm will first find that it does not satisfy the base case, then iterate from $M = 1$ to $2 \log N (= 54)$. Since none of M-s divides N, it will then invoke the algorithm from Theorem 22. As no $1 \leq r < 5$ satisfies

$$\alpha_r \geq 2 \sum_{i=r+1}^{5} \alpha_i,$$

the algorithm will compute the approximation: i.e. return the largest ℓ satisfying $\Phi(N)^{1/\ell} \leq N^{1/\ell} - 1$. As $\Phi(N) = 491059008000$, the algorithm outputs $L = 6 < 3^5$.

5 Discussion & Open Questions

In this paper we give the first efficient *deterministic* algorithm that approximates $\omega(N)$, given oracle access to $\Phi(\cdot)$. In addition, in our approach we attempt to compute/approximate $\omega(N)$ "directly" rather than factoring N first (although we may eventually end up succeeding to factor N). Below are some open questions.

- The immediate open question is, of course, to compute $\omega(N)$ exactly or at least improve the approximation ratio. As was noted, we believe that the exact computation of $\omega(N)$, given oracle access to $\Phi(\cdot)$, would constitute an important milestone. See discussion after Observation 5 for more details.
- Can we extend this result to other "interesting" functions that can be efficiently computed given the complete factorization of N?
- In particular, can we devise an efficient algorithm for computing the Möbius function $\mu(N)$? Given the results of [8, 14], this would be equivalent to determining the parity of $\omega(N)$.

References

1 L. M. Adleman and K. S. McCurley. Open problems in number theoretic complexity, II. In *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*, pages 291–322, 1994. doi:10.1007/3-540-58691-1.

2 M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

3 E. Bach. Intractable problems in number theory. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 77–93, 1988. doi:10.1007/0-387-34799-2.

4 E. Bach, G. L. Miller, and J. Shallit. Sums of divisors, perfect numbers and factoring. *SIAM J. Comput.*, 15(4):1143–1154, 1986. doi:10.1137/0215083.

5 D. Coppersmith. Finding a small root of a univariate modular equation. In *Advances in Cryptology - EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165. Springer, 1996. doi:10.1007/3-540-68339-9.

6 M. Hittmeir and J. Pomykala. Deterministic integer factorization with oracles for euler's totient function. *Fundam. Inform.*, 172(1):39–51, 2020. doi:10.3233/FI-2020-1891.

7 J. Kim, I. Volkovich, and N. X. Zhang. The power of leibniz-like functions as oracles. In *The 15th International Computer Science Symposium in Russia, CSR*, volume 12159 of *Lecture Notes in Computer Science*, pages 263–275. Springer, 2020. doi:10.1007/978-3-030-50026-9.

8 S. Landau. Some remarks on computing the square parts of integers. *Inf. Comput.*, 78(3):246–253, 1988. doi:10.1016/0890-5401(88)90028-4.

9 D. L. Long. Random equivalence of factorization and computation of orders. Technical Report 284, Princeton University, 1981.

10 G. L. Miller. Riemann's hypothesis and tests for primality. *J. Comput. Syst. Sci.*, 13(3):300–317, 1976. doi:10.1016/S0022-0000(76)80043-8.

11 F. Morain, G. Renault, and B. Smith. Deterministic factoring with oracles. *CoRR*, abs/1802.08444, 2018. arXiv:1802.08444.

12 M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of number theory*, 12(1):128–138, 1980.

13 J. Shallit and A. Shamir. Number-theoretic functions which are equivalent to number of divisors. *Inf. Process. Lett.*, 20(3):151–153, 1985. doi:10.1016/0020-0190(85)90084-5.

14 H. Woll. Reductions among number theoretic problems. *Inf. Comput.*, 72(3):167–179, 1987. doi:10.1016/0890-5401(87)90030-7.

15 B. Zralek. A deterministic version of pollard's p-1 algorithm. *Math. Comput.*, 79(269):513–533, 2010. doi:10.1090/S0025-5718-09-02262-5.

A Missing Proofs

Sketch of the proof of Part 4 of Lemma 13. Consider the following quadratic equation

$$f(x) \stackrel{\Delta}{=} x^2 - (N - \Phi(N) + 1)x + N = 0.$$

Observe that $f(x) = (x - p)(x - q)$. Hence, the solutions of the equation are exactly p and q. ◀