Computing the prime-counting function

The Meissel-Lehmer Method

Jean-Luc Portner

Semester project

Computation in Algebra and Number Theory
thought by
David Alexander Loeffler

Department of Mathematics
ETH Zürich
June 2022

1 Introduction

As prime numbers have fascinated mathematicians for centuries counting them arises as a natural question. For the longest time the only known way to find the number of primes smaller than $x \in \mathbb{N}$ $\pi(x)$ was to calculate all primes up to x and count them. For this the fastest method was the sieve of Eratosthenes[1] which will be described in section 1 and has a runtime of $O(x \log\log x)$.

At the beginning of the 19th century Legendre discovered the first way to count primes without having to calculate them all. His method will be described in section 2 and still had the downside of needing to calculate many non-zero terms making it unpractical.

The first efficient algorithm was described by the astronomer E.D.F. Meissler at the end of the 19th century. The importance was that his method made the space used economical through reducing the terms in the sum. Through his method Meissler managed to calculate $\pi(10^8$ correctly and $\pi(10^9)$ with an error of 56. Subsequently many suggested improvements to Meissler's method although none carried out the calculations.

At the middle of the 20th century with the rise of digital computers D.H. Lehmer picked up Meissler's Method and extended and simplified it. He implemented it on an IBM 701 and managed to calculate $\pi(10^{10})$ (with and error of 1) Subsequently we will present his method and some improvements made by ...

1.1 Asymptotic runtime analysis and the RAM

In order to do asymptotic runtime analysis we need to decide on a computational model to use. The classical model of a touring machine (TM) is not practical in our analysis as the time for a random read or write i.e. retrieving or writing values at random places on the tape scales with the total space used. That is the read/write head has to be moved from the current position to the position where the desired value should be read from/written to and back. Thus we consider the model of a *random access machine*, short RAM. The difference to a touring machine is that a RAM does not have a sequential tape but uses an unlimited amount of registers which can be addressed by integers. Moreover computations with these addresses are allowed. With this every memory location can be accessed in constant time. This model also fits better the modern day computers whose memory supports near constant read and write times.

In our discussions the need for a RAM arises in the sieving procedures. As these only work efficiently with random read and write access.

describe big o-notation

1.2 The sieve of Eratosthenes

[2] The sieve of Eratosthenes was first described in the work of Nicomedes (280-210 BC) entitled Introduction to Arithmetic and is probably the best known method of finding primes.

It is based on the following observation. Let $n \in \mathbb{N}$ be a composite number. Then there exists $d, d' \in 1, \ldots, n$ with $d \leq d'$ such that $d \cdot d' = n$. If $d > \sqrt{n}$. Then

$$n = d \cdot d' > \sqrt{n} \cdot \sqrt{n} = n \lightning .$$

Thus every composite number has a divisor $d \leq \sqrt{n}$ and in particular every composite number is divisible by a prime $p \leq \sqrt{n}$.

To now find all primes up to x we have the following algorithm:

[1] Actually this method was known already long before Eratosthenes, he only added the name "sieve" to it in the 3rd century

[2] From grad texts in Maths C1 67

- Write down all numbers from 1 to x.
- Cross out 1.
- Let d be the smallest number on the list whose multiples have not already been eliminated.
  1. If $d > \sqrt{x}$ stop.
  2. Else cross out all multiples of d greater equal d and repeat step 3.

The remaining non crossed out numbers are then the prime numbers up to x.

Example 1.1. An example of this process can be seen in the figure below.

Trivially the procedure ends after maximally $\sqrt{x}$ steps and thus is in $O(\sqrt{x})$. Importantly the only arithmetic operation needed is addition. Hence in reality the sieve is quicker than some other methods which have a better runtime however use multiplication too which has a bit wise complexity of $n^{\log_2 3}$ and thus can be slower.

The runtime bound on the sieve of Eratosthenes can be improved as follows: We assume that crossing out a number can be done in $O(1)$.

Notice that the d's in our algorithm are exactly the prime numbers. Further note that crossing out all multiples of d then takes exactly $\frac{x}{d}$ many steps. If p is the biggest prime $\leq \sqrt{x}$ then the loop is executed

$$\frac{x}{2} + \frac{x}{3} + \frac{x}{5} + \ldots + \frac{x}{p} = x \sum_{\substack{p \leq \sqrt{x} \\ p \text{ prime}}} \frac{1}{p}$$

times.

Using the fact proved by Euler that the reciprocal sum of primes grows with $O(\log\log x)$ and writing down of the primes can be done in time $O(x)$ we get the following theorem

Theorem 1.2. *Finding all primes up to x can be done in time*

$$O(x \log\log x).$$

Often we will also use the following result that $\log(x) \in O(x^\varepsilon)$ for $\varepsilon > 0$ which follows directly from l'Hopital's rule:

$$\lim_{x\to\infty} \frac{\log(x)}{x^\varepsilon} = \lim_{x\to\infty} \frac{x^{-1}}{\varepsilon x^{\varepsilon-1}} = \lim_{x\to\infty} \frac{1}{\varepsilon x^\varepsilon} = 0.$$

Thus sieving can be done in time $O(x^{1+\varepsilon})$.

More recently sieving algorithms have been found which run in sublinear time. More precisely Paul Pitchard found a sieving algorithm in [] with arithmetic complexity of $\Theta(\frac{n}{\log\log n}$.

1.3 Legendre's Method

In 1808, A.M. Legendre expanded on Eratosthenes sieve and gave a more analytic method which we will now describe.

Then we have the following fact:

Fact 1.3. $\phi(x, a) = \pi(x) - \pi(a) + 1$

This follows as ϕ

Also works with starting at d^2. also counts number of different prime factors a number has

2 The Meissel-Lehmer Method

2.1 General Facts

Definition 2.1. Let us denote the primes $2, 3, 5, \ldots$ numbered in increasing order by $p_1, p_2, \ldots$. For $a \geq 1$ let

$$\phi(x, a) = |\{n \leq x \mid p \mid n \Rightarrow p > p_a\}|$$

that is the *partial sieve function* counting the numbers $\leq x$ with no prime factors $\leq p_a$. Moreover we define

$$P_k(x, a) = \left| \left\{ n \leq x \mid n = \prod_{j=1}^{k} p_{m_j}, m_j > a \text{ for } 1 \leq j \leq k \right\} \right|$$

that is the kth partial sieve function counting the numbers $\leq x$ with exactly k prime factors $\geq p_a$. We extend this definition to $P_0(x, a) = 1$. Then as $\phi(x, a)$ also counts 1 we get

$$\phi(x, a) = \sum_{k=0}^{\infty} P_k(x, a)$$

where the sum has only finitely many non zero terms as every number $\leq x$ has a finite number of prime factors. Importantly the numbers with only one prime factor are the primes. Thus $P_1(x, a) = \pi(x) - a$ and we get the following formula:

$$\pi(x) = \phi(x, a) + a - 1 - \sum_{k \geq 2} P_k(x, a).$$

Now if we take p to be the biggest prime smaller equal to $x^{1/j}$. i.e. $p_{\pi(x^{1/j})}$. Then any $n \leq x$ can have at most j many prime factors bigger than p. Thus we conclude that $P_k(x, \pi(x^{1/j})) = 0$ for all $k \geq j$.

We therefore get the following formula: For $a = \pi(x^{1/j})$ it holds that

$$\pi(x) = \phi(x, a) + a - 1 + \sum_{2 \leq k < j} P_k(x, a).$$

Different choices for j now result in the different methods developed over time. A value of $j = 2$ for example yields Legendre's Method:

$$\pi(x) = \phi(x, a) - a + 1.$$

A general Method of Meissel Lehmer type can thus be split into multiple parts: In a first step one has to calculate the value of $\phi(x, a)$. For this the following recurrence is essential:

Lemma 2.2.

$$\phi(x, a) = \phi(x, a - 1) - \phi(\frac{x}{p_a}, a - 1).$$

Proof. Notice that $z \in \{y \leq x \mid p \mid x \Rightarrow p > p_a\}$ if and only if $z \in \{y \leq x \mid p \mid x \Rightarrow p > p_{a-1}\}$ and $p_a \not| z$. The first part of this condition is equal to $z \in \phi(x, a-1)$. Full filling the second condition can be reformulated as $z \in \phi(x, a-1) \setminus \{z \mid p_a \mid z\}$. This now can easily be seen to be equal to $z \in \phi(x, a-1) \setminus \{p_a x \mid p \mid x \Rightarrow p > p_{a-1}\}$. which finally is equal to $z \in \phi(x, a-1) \setminus \phi(\frac{x}{p_a}, a-1)$. Taking the absolute value yields the desired result. □

Through repeated application of this recurrence one can build a structure similar to a binary tree. Each node of the tree is represented by a term $\pm\phi(\frac{x}{n}, b)$ for some n and p. Each parent

node has two children and the sum of each "level" of the tree is equal to $\phi(x,a)$. If some branches are cut early i.e. the recurrence is not applied anymore to that branch, then the sum over all the leaves is equal to $\phi(x,a)$. Notice also that every node in the tree can be uniquely described by the tuple (n,b) where $n=\prod_{k=1}^{r} p_{a_k}$ with $a \geq a_1 > \ldots > a_r \geq b+1$. That is the pair (n,b) is associated with the term $(-1)^r \phi(x/n,b)$.

The tricky part when building this tree is to decide when to stop applying the recurrence to a node and calculate its value. For this different methods of Meissel-Lehmer type use different rules to optimize the runtime, called truncation rules- This part of the calculation is by far the most complex and time consuming. Thus nearly all advancements have been made here.

The second part of the calculation is to find the values of the $P_k(x,a)$. This is in general way easier then the previous step as simple explicit formulas exist. We will show them for $k=2$ and 3.

To calculate P_2 we use the following which holds whenever $a \leq \pi(x^{1/2})$:

$$P_2(x,a) = |\{n \mid n \leq x, n = p_b p_c \text{ with } a < b \leq c\}| \tag{1}$$

$$= \sum_{j=a+1}^{\pi(x^{1/2})} \left| \left\{ n \mid n \leq x, n = p_j p_k \text{ with } j \leq k \leq \pi\left(\frac{x}{p_j}\right) \right\} \right| \tag{2}$$

$$= \sum_{j=a+1}^{\pi(x^{1/2})} \left(\pi\left(\frac{x}{p_j}\right) - j + 1 \right) = \binom{a}{2} - \binom{\pi(x^{1/2})}{2} + \sum_{j=a+1}^{\pi(x^{1/2})} \pi\left(\frac{x}{p_j}\right) \tag{3}$$

where the second equality follows as for $n = p_j p_k$ we have $x \geq n = p_j p_k$ thus implying $p_k \leq \frac{x}{p_j}$ which translates to the inequality for the indices. The third inequality follow just from enumerating and the fourth from the Gaussche Summenformel.

For P_3 the following formula follows:

We are now ready to present and compare different Meissel-Lehmer-Methods.

2.2 Lehmer's original implementation

Lehmer was the first to implement the method on a computer. In his method he mostly chose a value of $a = \pi(x^{1/3})$ thus $P_k(x,a) = 0$ for $k \geq 3$ however some calculations were also carried out for $a = \pi(x^{1/4})$ which adds the term of $P_3(x,a)$. For the computation of P_3 he used a short precalculated table of $\pi(y)$ for small values of y which he stored in memory. In contrast for P_2 no such table is viable as the values of y can get quite big. Thus a modified version of Erastothenes sieve was used whose values where stored on magnetic tape.

The mathematically interesting part of the calculation however lies in finding the value of $\phi(x,a)$. For this Lehmer suggested the following truncation rule:

Definition 2.3 (Truncation rule L)**.** A node $\pm\phi(x/n,b)$ will not be split if one of the following holds

(i) $x/n < p_b$
(ii) $b = c(x)$ for a very slowly growing function $c(x)$.

Lehmer originally chose $c = 5$ for his computations.

He computed the leaves using the following formulas:

Lemma 2.4. *When applying the Truncation rule L the leaves can be calculated as follows:*

- *For leaves of type (i) it holds that $\phi(y,b) = 1$ if $y < p_b$.*

- *For leaves of type (ii) we have*

$$\phi(y,b) = \left\lfloor \frac{y}{Q} \right\rfloor \phi(Q,b) + \phi(y - \left\lfloor \frac{y}{Q} \right\rfloor Q, b)$$

where $b = c(x)$, $Q = \prod_{i \leq c(x)} p_i$ and the values of $\{\phi(y,b) \mid 1 \leq y \leq Q\}$ have been precomputed.

Proof. For the first formula we have: Let $1 < x \leq y$. Then as $y \leq p_b$ the prime factors of x are also smaller than p_b. Thus no x satisfies $p \mid x \Rightarrow p > p_b$ for p a prime number. Therefore $|\{x \leq y \mid p \mid x \Rightarrow p_b\}| = 1$ as only 1 is contained in this set. This yields the desired result of $\phi(y,b) = 1$.

The second formula follows as follows: Consider $x \in \mathbb{N}$ such that $Q\lambda \leq x \leq Q(\lambda+1)$. Then we can write x as $Q\lambda + r$ where r is not divisible by Q. Now as Q is divisible by p_i for $1 \leq i \leq b$ we have that $p_i \mid x$ if and only if p_i divides r.

Let us write $y = Q \cdot \lambda + r$. Then $\lambda = \left\lfloor \frac{y}{Q} \right\rfloor$ and we can compute the following:

$$\begin{aligned}
|\{x \leq y \mid p \mid x \Rightarrow p > p_b\}| &= \sum_{\mu=0}^{\lambda} |\{x \in \mathbb{N} \mid Q\mu \leq x < Q(\mu+1), p \mid x \Rightarrow p > p_b\}| \\
&\quad + |\{x \in \mathbb{N} \mid Q\mu \leq x \leq r, p \mid x \Rightarrow p > p_b\}| \\
&= \sum_{\mu=0}^{\lambda} |\{x < Q \mid p \mid x \Rightarrow p > p_b\}| + |\{x \leq r \mid p \mid x \Rightarrow p > p_b\}| \\
&= \lambda \cdot \phi(Q,b) + \phi(r,b).
\end{aligned}$$

where in the first equality we just dissected the set and used that the resulting sets are disjoint. In the second equality we used the above showed. And in the final equality we used the definition of ϕ as well as that Q is divisible by $p < p_b$ and therefore the strict inequality $x < Q$ can be changed to $x \leq Q$. By now plugging in the values of λ and r in dependence of y and Q we get the desired result. □

For an exact description of the implementation of this truncation rule and calculation the reader can consult []. The big disadvantage of Lehmer's truncation rule is that it is rather space inefficient e.g. the calculation of $\phi(10^{10}, 65)$ lead to more than 3 million leaves which needed to be calculated. Asymptotically the number of nodes in the tree is roughly $\frac{1}{24}a^4 = \Omega(x/\log^4(x))$. Lehmer himself states that "this is a good example of how one can substitute time for space with a high-speed computer."

2.3 The extended Meissel-Lehmer Method

This method was developed by Lagarias, Miller and Odlyzko. Its big advantage time and space efficient. To achieve this they chose a value of $a = \pi(x^{1/2})$ and split the computation of P_2 into batches of size $x^{2/3}$. The big change however was in the truncation rule they used for calculating $\phi(x,a)$ which significantly reduced the amount of leaves:

Definition 2.5 (Truncation rule T)**.** Do not split a node labelled $\pm\phi(x/n, b)$ if either of the following holds:

(i) $b = 0$ and $n \leq x^{1/3}$ or
(ii) $n > x^{1/3}$.

Leaves of type (i) are called *ordinary leaves* and of (ii) are called *special leaves.*

The following lemma shows the great reduction in the number of leaves that can be achieved with this new rule:

Lemma 2.6. *When using truncation rule T to calculate $\phi(x, a)$ with $a = \pi^{1/3}$ the resulting binary tree has at most $x^{1/3}$ ordinary leaves and at most $x^{2/3}$ special leaves.*

Proof. from paper □