Summing $\mu(n)$: an even faster elementary algorithm

Lola Thompson
(joint work with
Harald André Helfgott)

Universiteit Utrecht

The Möbius Function

Def Let $\mu(n) = \begin{cases} (-1)^{\omega(n)} & \text{if } n \text{ is } \square\text{-free} \\ 0 & \text{otherwise} \end{cases}$

$\omega(n) \leftarrow$ # of distinct prime factors of n

Ex $\mu(p) = -1 \ \forall p$ prime

$\mu(4) = \mu(2^{2}) = 0$

$\mu(6) = \mu(2 \cdot 3) = (-1)^{2} = 1$

$\mu(24) = \mu(2^{3} \cdot 3) = 0$

The Mertens Function

Def The function

$$M(N) = \sum_{n \le N} \mu(n)$$

is called the Mertens Function.

Ex $M(6) = \mu(1) + \mu(2) + \mu(3) + \mu(4) + \mu(5) + \mu(6)$

$$= 1 + (-1) + (-1) + 0 + (-1) + 1$$

$$= \boxed{-1}$$

Our goal: Compute $M(N)$ as quickly as possible, using as little space as possible.

Why Compute $M(N)$?

* Testing conjectures

  E.g., Until when is $|M(N)| \leq \sqrt{N}$ true?

  ↳ known to be false for extremely large N (Odlyzko-te Riele)

* For N large, asymptotic expressions for $M(N)$ are good. For bounded N, they are not (Computations are preferable).

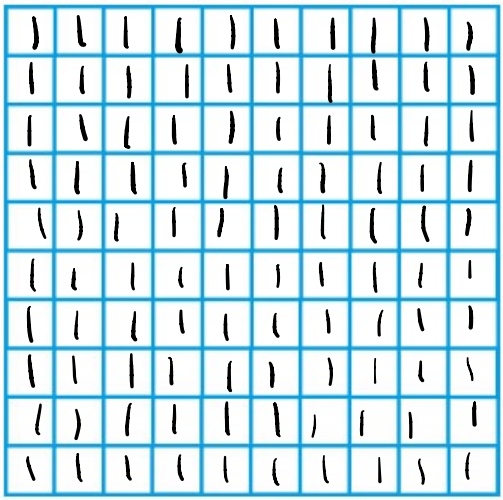
Naive approach: Compute $\mu(n)$ for each $n \leq N$ and sum.

Time: at least $\Theta(N \cdot (\text{Avg time it takes to compute } \mu(n)))$

A Slightly Less Naive Method

We can use the Sieve of Eratosthenes to compute $\mu(n)$ for $1 \le n \le N$ in time $\Theta(N \log\log N)$ and space $\Theta(N)$.

Step ①: Start w/ a "1" in every square:



Step ②: Flip the signs on the even entries, & make every other even entry 0.

1	-1	1	0	1	-1	1	0	1	-1
1	0	1	-1	1	0	1	-1	1	0
1	-1	1	0	1	-1	1	0	1	-1
1	0	1	-1	1	0	1	-1	1	0
1	-1	1	0	1	-1	1	0	1	-1
1	0	1	-1	1	0	1	-1	1	0
1	-1	1	0	1	-1	1	0	1	-1
1	0	1	-1	1	0	1	-1	1	0
1	-1	1	0	1	-1	1	0	1	-1
1	0	1	-1	1	0	1	-1	1	0

even integers

Step ③ Flip the signs on the multiples of 3, making every third multiple of three 0.

1	-1	-1	0	1	1	1	0	0	-1
1	0	1	-1	-1	0	1	0	1	0
-1	-1	1	0	1	-1	0	0	1	1
1	0	-1	-1	1	0	1	-1	-1	0
1	1	1	0	0	-1	1	0	1	-1
-1	0	1	0	1	0	-1	-1	1	0
1	-1	0	0	1	1	1	0	-1	-1
1	0	1	-1	-1	0	1	1	1	0
0	-1	1	0	1	-1	-1	0	1	0
1	0	-1	-1	1	0	1	-1	0	0

0 = multiples of 3

Step ④: Flip the signs on the multiples of 5, making every fifth multiple of five 0.

1	-1	-1	0	-1	1	1	0	0	1
1	0	1	-1	1	0	1	0	1	0
-1	-1	1	0	0	-1	0	0	1	-1
1	0	-1	-1	-1	0	1	-1	-1	0
1	1	1	0	0	-1	1	0	1	0
-1	0	1	0	-1	0	-1	-1	1	0
1	-1	0	0	-1	1	1	0	-1	1
1	0	1	-1	0	0	1	1	1	0
0	-1	1	0	-1	-1	-1	0	1	0
1	0	-1	-1	-1	0	1	-1	0	0

multiples of 5

Step ④: Flip the signs on the multiples of 7, making every seventh multiple of seven 0.

1	-1	-1	0	-1	1	-1	0	0	1
1	0	1	1	1	0	1	0	1	0
1	-1	1	0	0	-1	0	0	1	-1
1	0	-1	-1	1	0	1	-1	-1	0
1	-1	1	0	0	-1	1	0	0	0
-1	0	1	0	-1	0	-1	-1	1	0
1	-1	0	0	-1	1	1	0	-1	-1
1	0	1	-1	0	0	-1	1	1	0
0	-1	1	0	-1	-1	-1	0	1	0
-1	0	-1	-1	-1	0	1	0	0	0

$\square$ = multiple of 7

* Since the next prime is $11 > \sqrt{100}$, we would seem to be finished... except that some of these entries are wrong!

What's going on?

Ex

$\mu(13) = -1$

$\mu(17) = -1$

$\mu(19) = -1$

1	-1	-1	0	-1	1	-1	0	0	1
1	0	1	1	1	0	1	0	1	0
1	-1	1	0	0	-1	0	0	1	-1
1	0	-1	-1	1	0	1	-1	-1	0
1	-1	1	0	0	-1	1	0	0	0
-1	0	1	0	-1	0	-1	-1	1	0
1	-1	0	0	-1	1	1	0	-1	-1
1	0	1	-1	0	0	-1	1	1	0
0	-1	1	0	-1	-1	-1	0	1	0
-1	0	-1	-1	-1	0	1	0	0	0

other wrong entries w/ a prime factor > 7

□ = multiple of 7

Notice that there is at most one prime missing in each factorization.

To get around this, we can store the product of primes that we've found for each entry, so we know if there is a prime factor $> \sqrt{N}$ missing.

1	2	3	0	5	6	7	0	0	10
1	0	1	14	15	0	1	0	1	0
21	2	1	0	0	2	0	0	1	30
1	0	3	2	35	0	1	2	3	0
1	42	1	0	0	2	1	0	0	0
3	0	1	0	5	0	3	2	1	0
1	2	0	0	5	6	1	0	3	70
1	0	1	2	0	0	7	6	1	0
0	2	1	0	5	2	3	0	1	0
7	0	3	2	5	0	1	0	0	0

0 = missing one prime factor

Final Step: Flip all of the signs of the entries

1	2	3	0	5	6	7	0	0	10
11	0	13	14	15	0	17	0	19	0
21	22	23	0	0	26	0	0	29	30
31	0	33	34	35	0	37	38	39	0
41	42	43	0	0	46	47	0	0	0
51	0	53	0	55	0	57	58	59	0
61	62	0	0	65	66	67	0	69	70
71	0	73	74	0	0	77	78	79	0
0	82	83	0	85	86	87	0	89	0
91	0	93	94	95	0	97	0	0	0

$\mu(n) = 1$

$\mu(n) = -1$

$\mu(n) = 0$

To save space, we can use a Segmented Sieve:

70	71	72	73	74	75	76	77	78	79

Look at segments of length $\sqrt{N}$ and check for divisibility by primes up to $\sqrt{N}$.

<u>Space</u>: $\Theta(\sqrt{N})$

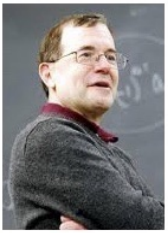
One more way to save space...

<u>Theorem</u> (Helfgott, 2020)

One can construct all primes $p \leq N$ in time $\Theta(N \log N)$ and space $\Theta(N^{1/3}(\log N)^{2/3})$.

Less Naive Methods

① Combinatorial Methods





First Steps: Meissel (1870s), Lehmer (1959)

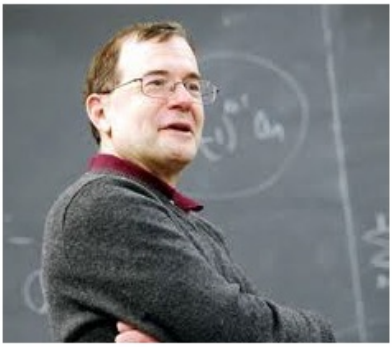
Improved by Lagarias-Miller-Odlyzko (1985)

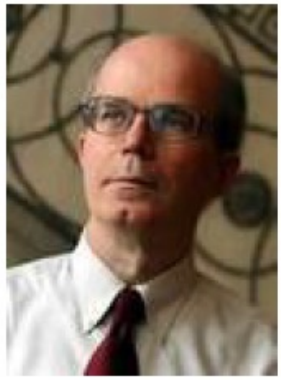
& Deléglise-Rivat (1996)

Main Idea: Use number theoretical identities to break $M(N) = \sum_{n \le N} \mu(n)$ into shorter sums. Compute the short sums once & use them many times.

Time: about $\Theta(N^{2/3})$

② Analytic Methods





Lagarias - Odlyzko (1987)

<u>Main Idea</u>: Can write $M(N)$ as sums over the zeroes of the Riemann Zeta function. There are ∞ly many zeros, but one can truncate & round. If error is $< \frac{1}{2}$, the result is exact.

Time: $\mathcal{O}(N^{1/2+\varepsilon})$ in theory

(nontrivial to implement (Platt, 2012) & slower than combinatorial methods in practice)

Our Goal:

Formulate a combinatorial algorithm that

* improves on the previous time bound of $\Theta(N^{2/3})$
* uses as little space as possible
* is practical to implement on a computer.

Theorem (Helfgott, T., 2021)

One can compute $M(N)$ in

time $\Theta_\varepsilon(N^{3/5}(\log N)^{3/5+\varepsilon})$ and

$\Theta(N^{3/5}\log N)$ using Helfgott's Sieve

Space $\Theta(N^{3/10}(\log N)^{13/10})$

$\Theta(N^{1/5}(\log N)^{5/3})$ using Helfgott's Sieve

Combinatorial Algorithms: general approach

Start w/ an identity:

$$M(N) = 2M(\sqrt{N}) - \sum_{n \le N} \sum_{\substack{m_1 m_2 n_1 = n \\ m_1, m_2 \le \sqrt{N}}} \mu(m_1)\mu(m_2)$$

($k=2$ case of Heath-Brown; also follows from Vaughan or Could use Möbius Inversion)

Swapping the order of summation:

$$M(N) = \underbrace{2M(\sqrt{N})}_{\substack{\text{Compute in} \\ \text{time } O(\sqrt{N}) \\ \text{naively}}} - \boxed{\sum_{m_1, m_2 \le \sqrt{N}} \mu(m_1)\mu(m_2) \left\lfloor \frac{N}{m_1 m_2} \right\rfloor}$$

* Choose a parameter $v \le \sqrt{N}$ and split into cases:

① $m_1, m_2 \le v$

② m_1 or $m_2 > v$

To obtain time $O(N^{2/3})$: ← Deleglise-Rivat, Lagarias-Miller-Odlyzko, etc.

Take $\nu = N^{1/3}$.

- Case ① ($m_1, m_2 \leq \nu$) is the easy case - use a segmented sieve.
- Case ② (m_1 or $m_2 > \nu$) takes more work.

What we do instead:

take $\boxed{\nu \approx N^{2/5}}$.

Larger $\nu \Rightarrow$ Case ① is the hard case now.

↖ This will be the focus of the rest of my talk

Case ② is easier.

How we handle Case ①

Want to compute:

$$\sum_{m_1, m_2 \le v} \mu(m_1)\mu(m_2)\left\lfloor \frac{N}{m_1 m_2} \right\rfloor$$

Case ① → $m_1, m_2 \le v$

$m_1 \to m$, $m_2 \to n$ from now on

Spit $[1,v] \times [1,v]$ into nbhds

$U = I_x \times I_y$ around points (m_0, n_0)

$I_x = [m_0 - a, m_0 + a)$, $I_y = [n_0 - b, n_0 + b)$

Applying a local linear approximation:

$$\frac{N}{mn} = \frac{N}{m_0 n_0} + C_x(m - m_0) + C_y(n - n_0) + \underbrace{ET_{Quad}}_{\text{Quadratic ET}}$$

$$C_x = \frac{-N}{m_0^2 n_0}, \quad C_y = \frac{-N}{m_0 n_0^2}$$

Small provided that U is small

If there were no floor functions...

↑ or quad ET

$$\sum_{(m,n)\in I_x \times I_y} \mu(m)\mu(n)\frac{N}{mn}$$

$$= \sum_{(m,n)\in I_x \times I_y} \mu(m)\mu(n)\left(\frac{N}{m_0 n_0} + c_x(m-m_0) + c_y(n-n_0)\right)$$

$$\overset{\circledast}{=} \left(\sum_{m\in I_x} \mu(m)\left(\frac{N}{m_0 n_0} + c_x(m-m_0)\right)\right)\cdot\sum_{n\in I_y}\mu(n)$$

separation of variables

$$+\left(\sum_{n\in I_y}\mu(n)c_y(n-n_0)\right)\cdot\sum_{m\in I_x}\mu(m)$$

* Use segmented sieve to compute $\mu(m)$ for $m\in I_x$ and $\mu(n)$ for $n\in I_y$.

* Computing the sum $\circledast$ takes time $\Theta(\max(a,b))$ and negligible space.

How to handle $\lfloor\ \rfloor$:

Notice that computing

Variables are separated

$$S_0 := \sum_{(m,n)\in I_x \times I_y} \mu(m)\mu(n)\left(\left\lfloor \frac{N}{m_0 n_0} + c_x(m-m_0)\right\rfloor + \lfloor c_y(n-n_0)\rfloor\right)$$

is the same as above.

what we have from our Linear Approx:

$$S_1 := \sum_{(m,n)\in I_x \times I_y} \mu(m)\mu(n)\left(\left\lfloor \frac{N}{m_0 n_0} + c_x(m-m_0) + c_y(n-n_0)\right\rfloor\right)$$

Can't be separated

what we actually want:

$$S_2 := \sum_{(m,n)\in I_x \times I_y} \mu(m)\mu(n)\left\lfloor \frac{N}{mn}\right\rfloor$$

Notice that

$$\lfloor A+B \rfloor - (\lfloor A \rfloor + \lfloor B \rfloor) = \begin{cases} 0 \text{ if } \{A\}+\{B\} < 1 \\ 1 \text{ otw} \end{cases}$$

So the difference between a term in S_1 & a term in S_0 is either 0 or 1.

(Same for the terms in S_2 vs S_1)

<u>Idea</u>: Let

$$L_0(m,n) = \left\lfloor \frac{N}{m_0 n_0} + c_x(m-m_0) \right\rfloor + \left\lfloor c_y(n-n_0) \right\rfloor$$

$$L_1(m,n) = \left\lfloor \frac{N}{m_0 n_0} + c_x(m-m_0) + c_y(n-n_0) \right\rfloor$$

$$L_2(m,n) = \left\lfloor \frac{N}{mn} \right\rfloor$$

We show that $L_2 - L_1$ and $L_1 - L_0$ Can be Computed quickly

We approximate c_y by a rational #

On each nbhd $I_x \times I_y$

$$\frac{a_0}{q}, \quad q \le Q = 2b$$

Such that $\delta := c_y - \frac{a_0}{q}$ satisfies

$$|\delta| \le \frac{1}{qQ}$$

Thus,

$$\left| c_y(n-n_0) - \frac{a_0(n-n_0)}{q} \right| \le \frac{1}{2q}$$

(can find such an $\frac{a_0}{q}$ using continued fractions)

↳ Time $\Theta(\log b)$

* Now our task is to show that

$L_2(m,n) = L_1(m,n)$ except in at most

Solve a linear equation (mod q)

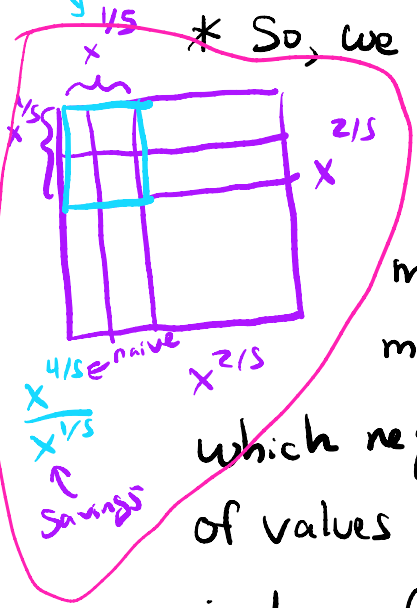
2 "bad" congruence classes (mod q)

(Same for $L_1(m,n)$ vs $L_0(m,n)$)

* In the case where m or n is in a "bad" residue class (mod q), we show that $L_2 - L_1$, $L_1 - L_0$ are char. functions of intervals (or unions of intervals)

To find them, we solve a quadratic equation

at worst



* So, we just need to compute a table of

$$\sum_{\substack{m \equiv a \text{ (bad)} \pmod q \\ m \in I}} \mu(m)$$

which requires pre-computing a table of values of $\mu(m)$, which can be done in time $\Theta(b)$ & space $\Theta(b \log b)$.

So we are saving a factor of "a" compared w/ the naive method

Computations

We wrote our algorithm in C^{++} and ran it on an 80-Core machine at the Max Planck Institute for Mathematics:

x	$M(x)$	x	$M(x)$
10^{17}	-21830254	2^{68}	2092394726
10^{18}	-46758740	2^{69}	-3748189801
10^{19}	899990187	2^{70}	9853266869
10^{20}	461113106	2^{71}	-12658250658
10^{21}	-3395895277	2^{72}	9558471405
10^{22}	-2061910120	2^{73}	-6524408924

Kuznetsov

Hurst

These match with Kuznetsov* & Hurst's Computations using Deleglise & Rivat's algorithm

*Except for a possible sign error in Kuznetsov for $x = 10^{21}$.

Thank you!