Sumar $\mu(n)$: un algoritmo elemental más rápido

Lola Thompson
(trabajo conjunto con
Harald Andrés Helfgott)

Universiteit Utrecht

La función Möbius

Def Sea $\mu(n) = \begin{cases} (-1)^{\omega(n)} & \text{Si } n \text{ es libre de cuadrados} \\ 0 & \text{Si no} \end{cases}$

$\omega(n)$ ← # de factores primos distintos de n

Ej $\mu(p) = -1 \ \forall p$ primo

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

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

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

La función Mertens

Def La función

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

Se llama la función de Mertens.

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

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

$= \boxed{-1}$

Nuestro objectivo: Calcular $M(N)$ lo más rápido posible, utilizando el menor tiempo y espacio posible.

Por qué Calcular $M(N)$?

* Probar conjeturas

  E.j., Hasta cuando es $|M(N)| \leq \sqrt{N}$ verdad?

  ↳ Odlyzko - te Riele
  falsa para N extremadamente grande

* Para N grande, las expresiones asintóticas para $M(N)$ son buenas. Para N acotada, no son (los cálculos son preferibles).

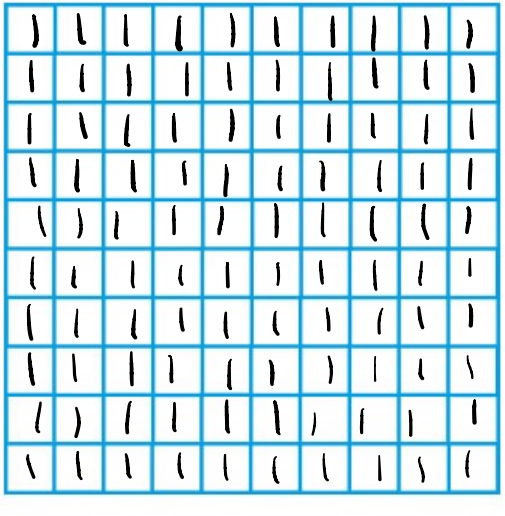
Enfoque Ingenuo: Calcular $\mu(n)$ para cada $n \leq N$ y Sumar.

Tiempo: Por lo menos $\mathcal{O}(N \cdot (\text{tiempo promedio para calcular } \mu(n)))$

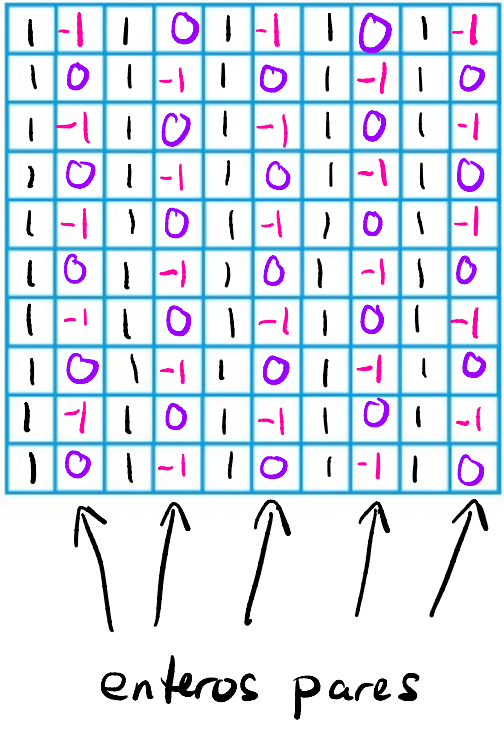
Un método menos ingenuo

Podemos usar la criba de Eratóstenes para calcular $\mu(n)$ para $1 \leq n \leq N$ en el tiempo $\Theta(N \log\log N)$ y espacio $\Theta(N)$.

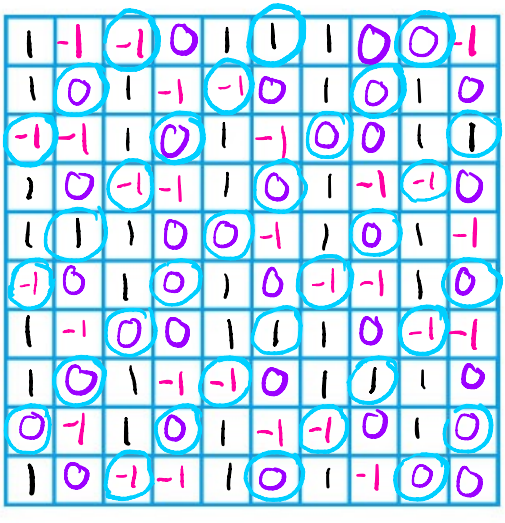
Paso ①: Comenzamos con un "1" en cada cuadrado:



Paso ②: Invertimos los signos en las entradas pares y hacemos que todas las demás entradas pares sean 0.



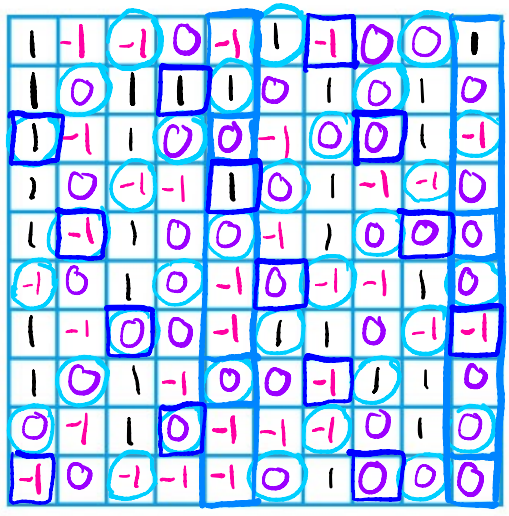
Paso ③ Invertimos los signos en los múltiplos de 3, haciendo que cada tercer múltiplo de tres sea 0.



0 = múltiplos de 3

Paso ④: Invertimos los signos en los múltiplos de 5, haciendo que cada quinto múltiplo de cinco sea 0.

Paso ⑤: Invertimos los signos en los múltiplos de 7, haciendo que cada Séptimo múltiplo de Siete sea 0.



□ = múltiplo de 7

* Porque el próximo primo es $11 > \sqrt{100}$,

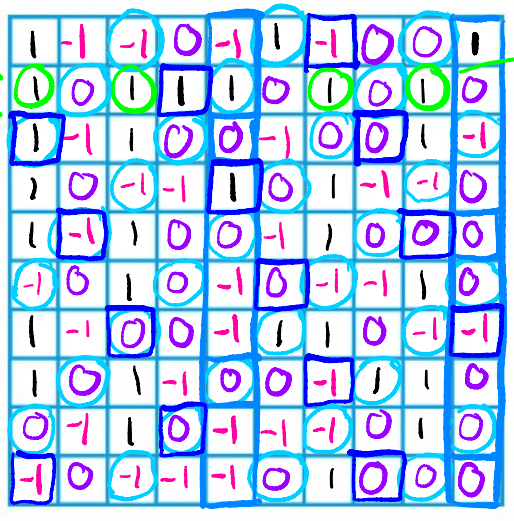
Parece que hemos terminado...

excepto que algunas de estas entradas

Son incorrectas!

Qué esta pasando???

Ej
$\mu(11) = -1$ ←



→ $\mu(19) = -1$

otras entradas incorrectas con un factor primo > 7

Aviso: a todas las entradas de la tabla le falta como máximo un factor primo!

Para evitar esto, podemos almacenar el producto de los números primos que hemos encontrado para cada entrada, de modo que sepamos si falta un factor primo $> \sqrt{N}$.

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 = falta un factor primo

Último paso: invertimos todos los signos de las entradas

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

$\mu(n) = 1$

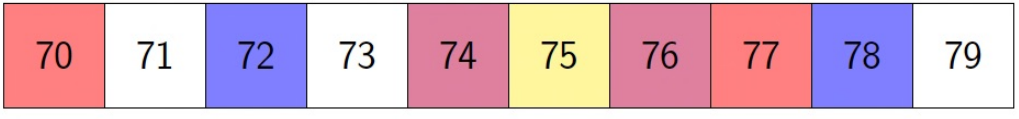
$\mu(n) = -1$

$\mu(n) = 0$

tiempo $\Theta(N \log\log N)$

espacio $\Theta(N)$

Para ahorrar espacio, podemos utilizar una criba segmentada.



Miramos segmentos de longitud $\sqrt{N}$ y verificamos la divisibilidad por números primos hasta $\sqrt{N}$.

Espacio : $\Theta(\sqrt{N})$

Una forma más de ahorrar espacio...

Teorema (Helfgott, 2020)

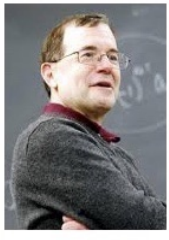
Se puede construir todos los primos $p \leq N$ en tiempo $\Theta(N \log N)$ y espacio $\Theta(N^{1/3} (\log N)^{2/3})$.

↑ mas lento

↑ menos espacio

Métodos menos ingenuos

① Métodos Combinatorios





Primeros pasos: Meissel (1870s), Lehmer (1959)

Mejorado por Lagarias-Miller-Odlyzko (1985)

y Deléglise-Rivat (1996)

Idea Principal: Usamos identidades aritméticas para expresar

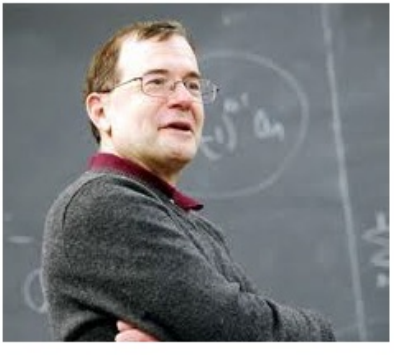
$$M(N) = \sum_{n \leq N} \mu(n)$$

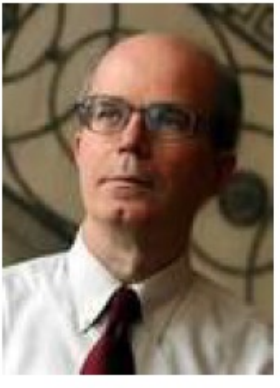
en términos de sumas más cortas.

Calculamos las sumas cortas una vez y las usamos muchas veces.

Tiempo: Aproximadamente $\mathcal{O}(N^{2/3})$

② Métodos analíticos





Lagarias - Odlyzko (1987)

Idea Principal: Podemos escribir $M(N)$ como sumas sobre los ceros de la función zeta de Riemann. Hay infinitos ceros pero podemos truncar y redondear.

Tiempo: $\Theta(N^{1/2+\varepsilon})$ en teoría

No trivial de implementar. Más lento que los métodos combinatorios en la práctica.

Nuestro objetivo

Formular un algoritmo Combinatorio que:

* mejora el límite de tiempo anterior de $O(N^{2/3})$
* utiliza el menor espacio posible
* es práctico de implementar en una computadora

Teorema (Helfgott, T., 2021)

Nosotros podemos calcular $M(N)$ en

tiempo $O_\varepsilon(N^{3/5}(\log N)^{3/5+\varepsilon})$ y $\rightarrow O(N^{3/5}\log N)$ con la criba de Helfgott

espacio $O(N^{3/10}(\log N)^{13/10})$ $\rightarrow O(N^{1/5}(\log N)^{5/3})$ con la criba de Helfgott

Algoritmos combinatorios: enfoque general

Empezamos con una identidad:

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

($k=2$ caso de Heath-Brown; también se sigue de Vaughan o podría usar la inversión de Möbius)

Intercambiando el orden de la suma:

$$M(N) = 2M(\sqrt{N}) - \sum_{m_1, m_2 \le \sqrt{N}} \mu(m_1)\mu(m_2) \left\lfloor \frac{N}{m_1 m_2} \right\rfloor$$

Calcular en tiempo $\Theta(\sqrt{N})$ ingenuamente

* Elegimos un parámetro $V \le \sqrt{N}$ y dividimos en casos:

① $m_1, m_2 \le V$

② m_1 o $m_2 > V$

Para obtener tiempo $\mathcal{O}(N^{2/3})$: ← Deleglise-Rivat, Lagarias-Miller-Odlyzko, etc.

Sea $v = N^{1/3}$.

- Caso ① $(m_1, m_2 \leq v)$ es el caso fácil- utilizamos la criba segmentada.
- Caso ② $(m_1$ o $m_2 > v)$ requiere más trabajo.

Lo que hacemos en su lugar:

Sea $\boxed{v \approx N^{2/5}}$

v mas grande $\Rightarrow$ Caso ① es el caso difícil ahora

↖ este será el tema Central del resto de mi charla

Caso ② es mas fácil

Cómo manejamos el caso ①

Queremos calcular:

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

Caso ① → $m_1, m_2 \leq v$

$m_1 \to m$, $m_2 \to n$ en lo sucesivo

Dividimos $[1, v] \times [1, v]$ en vecindades

$U = I_x \times I_y$ alrededor de puntos (m_0, n_0)

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

Aplicamos una aproximación lineal local:

$$\frac{N}{mn} = \frac{N}{m_0 n_0} + C_x(m - m_0) + C_y(n - n_0) + \underbrace{ET_{Quad}}_{\text{término de error cuadrático}}$$

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

↑ Pequeño si U es Pequeño

Si no hubiera funciones de piso

Σ o ET_{Quad}

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

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

$$\overset{(*)}{=} \left(\sum_{m\in I_x} \mu(m)\left(\frac{N}{m_o n_o} + c_x(m-m_o)\right)\right)\cdot\left(\sum_{n\in I_y}\mu(n)\right)$$

Separación de variables →

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

* usamos la criba segmentada para calcular $\mu(m)$ para $m\in I_x$ y $\mu(n)$ para $n\in I_y$.

* Calcular la suma (*) requiere tiempo $\Theta(\max(a,b))$ y espacio insignificante.

Cómo manejar $\lfloor \ \rfloor$:

Nótese que la computación

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

variables están separadas

es la misma que arriba.

Lo que tenemos de nuestra aprox. lineal:

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

no se puede separar

Lo que realmente queremos:

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

la parte fraccionaria

Observamos que

$$\lfloor A+B \rfloor - (\lfloor A \rfloor + \lfloor B \rfloor) = \begin{cases} 0 & \text{si } \{A\}+\{B\} < 1 \\ 1 & \text{si no} \end{cases}$$

Entonces la diferencia entre un término en S_1 y un término en S_0 es 0 o 1.

(Lo mismo para los términos en S_2 vs S_1)

Idea: Sea

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

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

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

Mostramos que $L_2 - L_1$ y $L_1 - L_0$

Se puede calcular rápidamente.

Aproximamos c_y por un # racional

en cada vecindad $I_x \times I_y$

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

tal que $\delta := c_y - \frac{a_0}{q}$ satisface

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

Así,

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

(Podemos encontrar tal $\frac{a_0}{q}$ usando fracciones continuas)

↳ Tiempo $\mathcal{O}(\log b)$

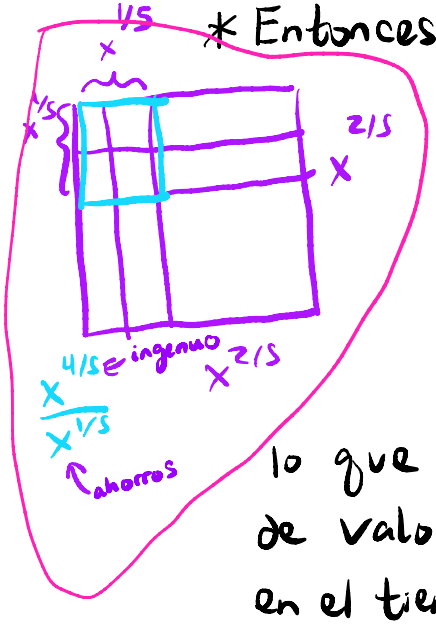
* Ahora nuestra tarea es mostrar que $L_2(m,n) = L_1(m,n)$ excepto en a lo más 2 clases de congruencia malas (mód q)

Resolver una ecuación lineal (mód q)

(Lo mismo para $L_1(m,n)$ vs $L_0(m,n)$)

* En el caso de que m o n esté en una clase de residuo malo, mostramos que $L_2 - L_1$, $L_1 - L_0$ son funciones características de intervalos (o uniones de intervalos)

↖ Para encontrarlas, resolvemos una ecuación cuadrática



* Entonces, solo necesitamos calcular una tabla de

$$\sum_{\substack{m \in I \\ m \equiv a \pmod g}} \mu(m)$$

(malo)

lo que requiere precalcular una tabla de valores de $\mu(m)$, que se puede hacer en el tiempo $\mathcal{O}(b)$ y espacio $\mathcal{O}(b \log b)$

↖ por lo que ahorramos un factor "a" en comparación con el método ingenuo

Cálculos

Escribimos nuestro algoritmo en C++ y lo ejecutamos en una máquina de 80 núcleos en el instituto Max Planck.

x	$M(x)$
10^{17}	-21830254
10^{18}	-46758740
10^{19}	899990187
10^{20}	461113106
10^{21}	-3395895277
10^{22}	-2061910120
10^{23}	62467771689

x	$M(x)$
2^{68}	2092394726
2^{69}	-3748189801
2^{70}	9853266869
2^{71}	-12658250658
2^{72}	9558471405
2^{73}	-6524408924
2^{74}	-6336351930
2^{75}	-4000846218

Hurst

Kuznetsov

H-T '21

estos Concuerdan con los cálculos de Kuznetsov* y Hurst (usando el algoritmo de Deleglise y Rivat)

* excepto por un posible error de signo de Kuznetsov en $x = 10^{21}$.

¡Gracias!