Optimisation in Computational Number Theory

Ben Lowe

2024

Notation

Definition

$\{a, b, c\}$ - The set containing a, b, c.
$x \in X$ - x is an element of the set X.
$\exists$ - There exists.
$\forall$ - For all.
$\mathbb{N}$ - The set of natural numbers, $\{1, 2, 3, ...\}$
$\mathbb{Z}$ - The set of integar numbers, $\{..., -2, -1, 0, 1, 2, ...\}$

For Example

$\forall a \in \mathbb{N}$ and $b \in \mathbb{Z}$, then $\exists c \in \mathbb{Z}$ such that $ab = c$.
For all a in the natural numbers and b in the integers their exists some c in the integers such that $ab = c$.

Primality

Definition

Let $a, b \in \mathbb{N}$, a divides b if and only if $\exists n \in \mathbb{N}$ such that $b = na$. We write $a \mid b$.

Primality

Definition

$P \in \mathbb{N}$ is prime if and only if $\forall a \in \mathbb{N}$ with $n \neq 1, P$ then $a \nmid P$.

Primality

Proposition

$P \in \mathbb{N}$ is prime if $\forall a \in \mathbb{N}$ such that $1 < a \leq \sqrt{P}$ then $a \nmid P$.

Primality

Proof.

By way of contradiction we suppose that both P is not prime and $\forall a \in \mathbb{N}$ st $a < \sqrt{N}$ then $a \nmid P$. Since P is not prime we have some $b \nmid P$ so $\exists n$ such that $bn = P$. Either, b or n are less than $\sqrt{P}$ so we can let $a = b$ and we have a contradiction. Or then must both be greater than $\sqrt{P}$. In this case we have $bn > \sqrt{P}n > \sqrt{P}\sqrt{P} = P$ so $bn > P$, this means $bn \neq P$ which is a contradiction. Therefore, P must be prime. $\square$

Primality

Definition

For $a, b \in \mathbb{Z}$, we say $a \equiv b \mod n$ if and only if $a - b \mid n$.

Primality

Corollary

P is prime if $\forall a \in \mathbb{N}$ *such that* $1 < a \leq \sqrt{P}$ *implies* $P \not\equiv 0 \mod a$.

Primality

Proof.
We suppose $\forall a \in \mathbb{N}$ such that $1 < a \leq \sqrt{P}$ then $P \not\equiv 0 \mod a$.
Then by the definition of modular arithmetic $a - 0 = a \nmid P$.
Therefore by proposition 1.1 P is prime. $\square$

Primality

Algorithm 1 Divisibility Test

function PRIME(n)
    if $n \leq 1$ **then**
        return FALSE ▷ 1 and numbers ≤ 1 are not prime
    end if
    for $i \in \mathbb{N}, 1 < i < \sqrt{n}$ **do**
        if $n \equiv 0 \mod i$ **then**
            return FALSE ▷ n is divisible by i and therefore not prime
        end if
    end for
    return TRUE
end function

Primality

Algorithm 2 Divisibility Test Sieve

function PRIME-SIEVE(n):
    Array<bool> *prime*[*n*]
    for $i \in \{0, ..., n\}$ **do**
        $prime[i] \leftarrow Prime(i)$
    end for
    return primes
end function

Primality

Definition

A composite number is a number $a \in \mathbb{N}$ that is not prime.

Primality

Proposition

Composite numbers can be factored into primes.

Primality

Proof.

Base case: For $k = 4$ and primes $p_1 = 2$, $p_2 = 2$. It is clear $p_1 p_2 = k$. Inductive case: Suppose composite numbers less that k can be factored into primes. Consider $k + 1$ is not prime so there exists $a \in \mathbb{N}$ such that $a \mid k + 1$ so $k + 1 = an$. since $a \geq 1$ clearly both $a < k$ and $n < k$ so by the hypothesis a and n can be factored into primes $a = p_1 p_2 ... p_m$ and $n = q_1 q_2 ... q_w$. Therefore, $k + 1 = p_1 p_2 ... p_m q_1 q_2 ... q_w$ can be factored into primes. $\square$

Primality

Algorithm 3 Sieve Of Eratosthenes

```
function ERATOSTHENES(n):
    Array<bool> prime[n] ← True
    prime[0] ← False
    prime[1] ← False
    for i ∈ {2, 3, ..., n} do
        for p ∈ {2, 3, ..., √i} do
            if p = True then        ▷ Boolean logic omitted for clarity.
                if i ≡ 0 mod p then
                    prime[i] ← False        ▷ i is not prime.
                end if
            end if
        end for
    end for
    return prime
end function
```

Linear Sieves

Proposition

$P \in \mathbb{N}$ is prime or has a perfect square factor if the following conditions hold:

- $4x^2 + y^2 = p$ *has an odd number of solutions when* $p \equiv 1 \mod 12$ *or* $z \equiv 5 \mod 12$.
- $3x^2 + y^2 = p$ *has an odd number of solutions when* $p \equiv 7 \mod 12$.
- $P \not\equiv 3 \mod 4$ *and* $3x^2 - y^2 = p$ *has an odd number of solutions when* $z \equiv 11 \mod 12$.

Linear Sieves

Algorithm 4 Sieve of Atkin

```
function ATKINS(n)
    Array<bool> prime[n] ← False
    for x ∈ {1, 2, .., √n} do
        for y ∈ {1, 2, .., √n} do
            z ← 4x^2 + y^2
            if z ≤ n & (z ≡ 1 mod 12 or z ≡ 5 mod 12) then
                prime[z] ← ¬prime[z]
            end if
            z ← 3x^2 + y^2
            if z ≤ n & z ≡ 7 mod 12 then
                prime[z] ← ¬prime[z]
            end if
            z ← 3x^2 − y^2
            if x > y & z ≤ n & z ≡ 11 mod 12 then
                prime[z] ← ¬prime[z]
            end if
        end for
    end for
    prime[2] ← True
    prime[3] ← True
    prime[5] ← True
    return prime
end function
```

Linear Sieves

Algorithm 5 Remove Perfect Squares

```
function REMOVE(prime)
    for i ∈ {k²|k ∈ {1, 2, ..., n}} do
        if prime[i] = True then
            for j ∈ {k ∈ ℕ|ki² < n} do
                prime[ji²] ← False
            end for
        end if
    end for
    return prime
end function
```

Multi-Threading

Algorithm 6 Atkins Sieve Kernel

function ATKINSKER(x,y, n)
  $z \leftarrow 4x^2 + y^2$
  if $z \leq n$ & $(z \equiv 1 \mod 12$ or $z \equiv 5 \mod 12)$ **then**
    $prime[z] \leftarrow \neg prime[z]$
  end if
  $z \leftarrow 3x^2 + y^2$
  if $z \leq n$ & $z \equiv 7 \mod 12$ **then**
    $prime[z] \leftarrow \neg prime[z]$
  end if
  $z \leftarrow 3x^2 - y^2$
  if $x > y$ & $z \leq n$ & $z \equiv 11 \mod 12$ **then**
    $prime[z] \leftarrow \neg prime[z]$
  end if
end function

Multi-Threading

Algorithm 7 Remove Perfect Squares Kernel

---

function $\textsc{RemoveKer}$(i, j, n)
  $i\_squared \leftarrow i^2$
  if $i\_squared \times j < n$ **then**
    $prime[i\_squared \times j] \leftarrow False$
  end if
end function

Frame Title

Algorithm 8 Multi-threaded Sieve of Atkin

> **function** MULTIATKIN(n)
>
>   Array<bool> $prime[n] \leftarrow False$ ▷ Allocate in GPU cache
>
>   **for** $x \in \{1, 2, .., \sqrt{n}\}$ **do**
>
>   Dispatch $\sqrt{n}$ AtkinsKer
>
>   **end for**
>
>   **for** $i \in \{k^2 | k \in \{1, 2, ..., \sqrt{n}\}\}$ **do**
>
>   Dispatch $\sqrt{n}$ RemoveKer
>
>   **end for**
>
>   **return** prime ▷ After loading the array back into the CPU cache
>
> **end function**

Multi-Threading

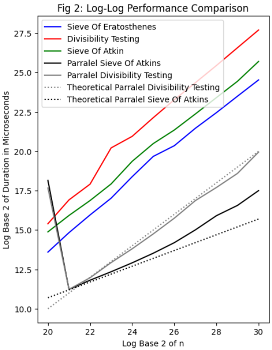

Figure: log-log plot of the duration of different sieve methods.