Primality Testing

by

Benjamin Chen

Waterloo, Ontario, Canada, 2022

Table of Contents

Chapter 1

Introduction

We first give a definition of prime numbers and composite numbers. [2]

Definition 1. *For $a, b \in \mathbb{Z}$ we say that a* **divides** *b (or that a is a* **factor** *of b, or that b is a* **multiple of** *a, and we write $a \mid b$, when $b = ak$ for some $k \in \mathbb{Z}$.*

Definition 2. *Let n be a positive integer. We say that n is a* **prime number** *when $n \geq 2$ and n has no factor $a \in \mathbb{Z}$ with $1 < a < n$. We say that n is* **composite** *when $n \geq 2$ and n is not prime, that is when n does have a factor $a \in \mathbb{Z}$ with $1 < a < n$.*

The study of primes are essential in number theory, as we have the following fundamental theorem: [2]

Theorem 1. *(The Fundamental Theorem of Arithmetic, or The Unique Factorization Theorem) Every integer $n \geq 2$ can be written uniquely in the form $n = \prod_{k=1}^{\ell} p_k = p_1 p_2 \cdots p_\ell$ where $\ell \in \mathbb{Z}^+$ and the p_k are primes with $p_1 \leq p_2 \leq \cdots \leq p_\ell$.*

That is, every integer can be uniquely expressed as a product of primes. Then for a question on an arbitrary integer, k, a possible way to divide and conquer the problem is to decompose the integer into a product of primes, and consider the question on individual primes, and later piecing the results on individual primes together to form the desired result.

However, the problem with the approach is that there are infinitely many prime numbers such that when the arbitrary number, k, approaches infinity, the prime cases we need to consider also approach infinity. We give the following theorem: [1]

Theorem 2 (Euclid)**.** *There are infinitely many prime numbers.*

Proof. Assume that there are finitely many primes $p_1, \ldots, p_n$ say, and consider

$$m = p_1 \cdots p_n + 1$$

Then m can be written as a product of primes, and $p_k \mid m$ for some k with $1 \leq k \leq n$. Hence $p_k \mid m - p_1 \cdots p_n$, and $p_k \mid 1$, which is a contradiction. □

This, in other words, means that for a general solution to a problem, we cannot rely on some pre-calculated prime table to help us quickly decompose the number and precalculated results for the problem.

Thus, one way to make this approach practical is to find a quick way to determine whether a number is prime or not.

1.1 Naïve Primality Test - Trial Division

This is probably the most simplest way to determine whether a number is a prime or not. The idea is simple.

For an given number n, we check if it is divisible by any integer from 2 to $\sqrt{N}$.

This way of determining the primality follows directly from the definition of prime numbers. Note that here we don't need to go to N because if there is a factor a of $n = ab$ such that $a > \sqrt{N}$, then $b < \sqrt{N}$. We would have detected the factor earlier.

1.1.1 Implementation

Maple Implementation

```
NaivePrimeTest := proc(n::integer)::boolean;

local i, TimeUsed, answer;

description "Naive Primality Test";
```

```
   answer := true;
   TimeUsed := time[real]();

   if n < 2 then answer := false;
   else
      for i from 2 to floor(sqrt(n)) do
         if irem(n, i) = 0 then
            answer := false;
            break;
         end if;
      end do;
   end if;
   print("Time used:", time[real]() - TimeUsed);
   return answer;
   end proc
```

Mathematica Implementation

```
NaivePrimeTest[num_] :=
 num >= 2 && (Do[
    If[Divisible[num, i], Return[False, And]], {i, 2,
     Floor[Sqrt[num]]}]; True)
```

1.2 Naïve Primality Test Improved

We first show that every prime number greater than 3 can be written in the form of $6k \pm 1$. [3]

Theorem 3. *Every prime number greater than 3 can be written in the form of* $6k \pm 1$.

Proof. Every number can be expressed in the form of $(6k + i)$ for $i = -1, 0, 1, \cdots 4$ for some integer k.
However, 2 divides $(6k + 0), (6k + 2), (6k + 4)$, 3 divides $(6k + 3)$.
This leaves us with $(6k \pm 1)$, in which contains all the prime numbers. □

This gives us a slightly better way to test the primality of a number.

> We first check whether a number is divisible by 2 or 3, then we check all the numbers in the form of $6k \pm 1 \leq \sqrt{n}$.

1.2.1 Implementation

Maple Implementation

```
NaivePrimeTestImproved := proc(n::integer)::boolean;
local i, TimeUsed, answer, k;
description "Naive Primality Test";
answer := true;
if n < 2 then answer := false;
elif n = 2 or n = 3 then
   return true;
elif irem(n, 2) = 0 or irem(n, 3) = 0 then
   return false;
else
   for k to floor(1/6*sqrt(n) + 1/6) do
      if irem(n, 6*k + 1) = 0 or irem(n, 6*k - 1) = 0 then
         answer := false; break;
      end if;
   end do;
end if;

return answer;
end proc
```

Mathematica Implementation

```
NaivePrimeTestImproved[num_] :=
 (num == 2) || (num ==
    3) || (num >= 2 && ! Divisible[num, 2] && !
     Divisible[num, 3] && (Do[
      If[Divisible[num, i] || Divisible[num, i + 2],
```

```
Return[False, Or]], {i, 5, Floor[Sqrt[num]] - 2, 6}]; True))
```

1.3 Wilson's Theorem

Before we go into more sophisticated algorithms, let us not forget the simple but impractical primality test by Wilson's Theorem.

Theorem 4 (Wilson's Theorem)**.** *For $n \in \mathbb{N}$, n is a prime if and only if*

$$(n-1)! \equiv -1 \bmod n$$

This test is straightforward but practically useless as the running time is in the order of $O(n!)$.

1.3.1 Implementation

Maple Implementation

```
WilsonTheorem := proc(n::integer)::boolean;
   description "Wilson's Theorem Primality Test";
   return evalb(mods((n - 1)!, n) = -1);
end proc
```

Mathematica Implementation

```
   WilsonTheorem[n_] := Mod[(n - 1)!, n] == Mod[-1, n]
```

1.4 The Fermat Test

First, we need to recall several important theorems in number theory.[4]

Theorem 5 (Fermat's little theorem)**.**

$$a^{N-1} \equiv 1 \bmod N$$

for a prime N and any $a \in \mathbb{Z}$ which is coprime to N.

This theorem follows from the fact that the multiplicative group $\mathbb{Z}_N^*$ has an order of $\phi(N) = N - 1$ when N is prime.

If we loosen the restriction such that N is not a prime, we can define the order $\text{ord}_N(a)$ of a modulo N as the smallest integer $k \geq 1$ such that $a^k \equiv 1 \bmod N$.

The following theorem, Euler's theorem, generalized Fermat's little theorem:[4]

Theorem 6 (Euler's Theorem)**.**

$$a^{\phi(N)} \equiv 1 \bmod N$$

for any positive integer N and $a \in \mathbb{Z}$ which is coprime to N.

We have a property that $\text{ord}_N(a) \mid \phi(N)$.

Since testing even numbers for primality is very easy. We shall turn our attention to odd integers. We briefly describe an algorithm below:[4]

Algorithm 1 (Fermat Test)**.**
Input: An odd integer $N \geq 5$.
Output: Either "composite" or "possibly prime".

1. choose $a \in \{2, \cdots, N-1\}$ uniformly at random
2. compute $b = a^{N-1} \operatorname{rem} N$
3. If $b \neq 1$ then return "composite" else return "possibly prime"

It would be very nice if checking primality is this easy. If the above algorithm returns "composite", then it is indeed a composite number. This is guaranteed by the Fermat's little theorem. But if the above algorithm returns "possibly prime", then we don't know if it is really a prime or not. To see why, consider the following subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$:

$$L_N = \left\{u \in (\mathbb{Z}/N\mathbb{Z})^* \mid u^{N-1} = 1\right\}$$

If N is prime, then $L_N = (\mathbb{Z}/N\mathbb{Z})^*$. If $L_N \neq (\mathbb{Z}/N\mathbb{Z})^*$. Then by Lagrange's theorem, we know that $|L_N| \leq \frac{1}{2}|(\mathbb{Z}/N\mathbb{Z})^*|$. If a is chosen to be in $(\mathbb{Z}/N\mathbb{Z})^* \setminus L_N$, then the Fermat test will return "composite". We call this a (and $a \bmod N$) a **Fermat witness** to the compositeness of N. Similarly, if $a \in L_N$, we call this a (and $a \bmod N$) a **Fermat liar** for N.

This method has a serious flaw, that is, there exists composite numbers N such that $a^{N-1} \operatorname{rem} N = 1$ for all a. That is, $L_N \neq (\mathbb{Z}/N\mathbb{Z})^*$ but N is not a prime. These composite numbers are called **Carmichael numbers**.

1.4.1 Proof of Correctness

We claim the following.[4]

> If N is prime, then Algorithm 1 returns "possibly prime". If N is composite and not a Carmichael number, then it returns "composite" with probability at least 1/2.

Proof.

- If N is prime:

  The correctness of this algorithm is guaranteed by Fermat's little theorem.

- If N is not prime:

  - If $\gcd(a, N) > 1$:
    If $\gcd(a, N) > 1$, then $\gcd(b, N) = \gcd(a^{N-1}, N) > 1$. We write $b = kg, N = k'g$. Note that $b > N$. We can apply Euclidean algorithm to write $k = qk' + r$. If we multiply both sides by g, then we get $b \operatorname{rem} N = r$. We note that $\gcd(b, N) = \gcd(N, r) > 1$. Hence, $r \neq 1$. The test would return "composite" as expected.
  - If $\gcd(a, N) = 1$:
    If N is composite but not Carmichael. As shown previously, $|L_N| \leq \frac{1}{2}|(\mathbb{Z}/N\mathbb{Z})^*| = \phi(N)/2$. This shows that at least half of the possible choices of a (coprime to N) are Fermat witnesses.

□

1.4.2 Implementation

Maple Implementation

```
FermatTest := proc(n::integer)::string;
local a, b;
description "Fermat Test";
a := rand(2 .. n - 1)();
b := irem(a^(n - 1), n);
if b <> 1 then
   return "composite";
else
   return "possibly prime";
end if;
end proc
```

Mathematica Implementation

```
FermatTest[n_] :=
 If[PowerMod[RandomInteger[{2, n - 1}], (n - 1), n] != 1, "composite",
   "possibly prime"]
```

1.5 Strong Pseudoprimality Test

First, we wish to show the following lemma.

Lemma 1. *If p is prime, $e \in \mathbb{N}, e \geq 2$. Let $N = p^e$ and $a = 1 + p^{e-1}$, then* $\text{ord}_N(a) = p$.

Proof. We recall a property of Galois field, $GF(q), q = p^m$:

$$(x + y)^{p^j} = x^{p^j} + y^{p^j}$$

for $j = 0, 1, \cdots, m$. Hence, we have:

$$\begin{aligned} a^p &\equiv (1 + p^{e-1})^p \\ &\equiv 1 + p^e \\ &\equiv 1 \mod p^e \end{aligned}$$

Since $\text{ord}_N(a)$ is defined to be the smallest integer k such that $a^k \equiv 1 \mod N$. This implies that $\text{ord}_N(a) = 1$ or p. Since $a \not\equiv 1 \mod N$, $\text{ord}_N(a) = p$. □

With this lemma, we can prove the following claim.[4]

Lemma 2. *Any Carmichael number is squarefree.*

Proof. We prove by contradiction. Assume that there is a prime number p which divides the Carmichael number N exactly $e \geq 2$ times. By the Chinese Remainder Theorem, there exists $a \in \mathbb{Z}$ with $a \equiv 1 + p^{e-1} \mod p^e$ and $a \equiv 1 \mod N/p^e$. By Lemma 1, a has order p modulo p^e. Hence, a also has order p modulo N. By the properties of Carmichael number, $a^{N-1} \equiv 1 \mod N$. Then p divides $N-1$. However, by our assumption, p also divides N. This is a contradiction. □

It can be shown that N is a Carmichael number if and only if N is squarefree and $p - 1$ divides $N - 1$ for any prime factors p of N, and that Carmichael numbers are odd and have at least three prime factors. The first three Carmichael numbers are: $561 = 3 \cdot 11 \cdot 17, 1105 = 5 \cdot 13 \cdot 17$, and $1729 = 7 \cdot 13 \cdot 19$.

We propose the following algorithm.[4]

Algorithm 2 (Strong pseudoprimality test)**.**
Input: An odd integer $N \geq 3$.
Output: Either "composite", or "probably prime", or a proper factor of N.

1. choose $a \in \{2, \cdots, N-1\}$ uniformly at random
2. $d = \gcd(a, N)$
   if $d > 1$ then return d
3. write $N - 1 = 2^v m$ with $v, m \in \mathbb{N}, v \geq 1$, and m odd
   Compute $b_0 = a^m \operatorname{rem} N$
   if $b_0 = 1$ then return "probably prime"
4. for $i = 1, \cdots, v$ do $b_i = b_{i-1}^2 \operatorname{rem} N$
5. if $b_v = 1$ then $j = \min\{0 \leq i < v : b_{i+1} = 1\}$ else return "composite"
6. $g = \gcd(b_j + 1, N)$
   if $g = 1$ or $g = N$ then return "probably prime" else return g

1.5.1 Proof of Correctness

We can see that $b_i \equiv a^{2^i m} \bmod N$. Especially, $b_v = a^{2^v m} = a^{N-1} \equiv \bmod N$. We note that if $b_{i-1} = 1$, then $b_i = 1$.

- If N is composite but not Carmichael:

  There is a probability of at least $1/2$ such that a is a Fermat witness for N, $b_v \neq 1$. This case is just the basic Fermat test. The algorithm returns "composite" in step 5.

- If N is a prime:

  Fermat's little theorem guarantees that $b_v = 1$.

  - If $b_0 = 1$:

    Then the algorithm correctly returns "probably prime" in step 3.
  - If $b_0 \neq 1$:

    We have $b_j \neq 1$ and $b_j^2 \equiv b_{j+1} = 1 \bmod N$ in step 6. We use a property in integral domain R which states that a polynomial f has at most $\deg f$ roots in

R. We conclude that the polynomial $x^2 - 1 \in \mathbb{Z}_N[x]$ has at most two zeroes, $1, -1$. This implies that $b_j = N-1$. Hence, $g = \gcd(N, N) = N$. The algorithm also correctly returns "probably prime" in step 6.

- If N is a Carmichael number:

  Let P be the set of prime divisors of N. Since N is squarefree, we have $N = \prod_{p \in P} p$. Consider $I = \left\{ i \mid 0 \leq i \leq v, \forall u \in (\mathbb{Z}/N\mathbb{Z})^*, u^{2^i m} = 1 \right\}$. By the definition of Carmichael number, $v \in I$. Since m is odd, then $(-1)^m = -1 \neq 1$. Hence, $0 \notin I$. We can say that there exists some $l < v$ such that $I = \{l+1, l+2, \cdots, v\}$. Now, consider the following subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$:

  $$G = \left\{ u \in (\mathbb{Z}/N\mathbb{Z})^* \mid u^{2^l m} = \pm 1 \right\} \subseteq (\mathbb{Z}/N\mathbb{Z})^*$$

  We now wish to show that $G \neq (\mathbb{Z}/N\mathbb{Z})^*$. Since $l \notin I$, there exists some $p \in P$ and $b \in \mathbb{Z}$ coprime to p such that $b^{2^l m} \not\equiv 1 \bmod p$. We take some such p and b. The Chinese Remainder Theorem implies that there exists a $c \in \mathbb{Z}$ such that $c \equiv b \bmod p$ and $c \equiv 1 \bmod N/p$. Consequently, $c^{2^l m} \equiv b^{2^l m} \bmod p$ and $c^{2^l m} \equiv b^{2^l m} \equiv 1 \bmod N/p$. If $G = (\mathbb{Z}/N\mathbb{Z})^*$, then $b^{2^l m} = \pm 1 + kN$. Since $b^{2^l m} \equiv 1 \bmod N/p$, then $b^{2^l m} = 1 + kN$. However, this would force that $b^{2^l m} \equiv 1 \bmod p$. A contradiction. Hence, we have $c \bmod N \in (\mathbb{Z}/N\mathbb{Z})^* \setminus G$. With G being a proper subgroup, G has at most $|(\mathbb{Z}/N\mathbb{Z})^*| / 2 = \phi(N)/2$ elements.

  If a is chosen so that $a \bmod N \in (\mathbb{Z}/N\mathbb{Z})^* \setminus G$, then the algorithm will discover a proper divisor of N. Consider the fact that $b_{l+1} \equiv a^{2^{l+1} m} \equiv 1 \bmod N$ implies that for all $p \in P$, $b_{l+1} \equiv a^{2^{l+1} m} \equiv 1 \bmod p$. This means that $a^{2^l m} \bmod p$ is ± 1. Since $a \notin G$, $b_l \bmod N = a^{2^l m} \bmod N \neq \pm 1$. Both the possibilities can occur. From our previous work, we have shown that in step 5 $j = l$. Then, if $b_l = a^{2^l m} \equiv -1 \bmod p$, $b_l + 1 \equiv 0 \bmod p$. That is, $p \mid b_l + 1$. Hence,

  $$g = \gcd(b_l + 1, N) = \prod_{\substack{p \in P \\ a^{2^l m} \equiv -1 \bmod p}} p$$

  is a proper divisor of N. The probability is bounded by $|(\mathbb{Z}/N\mathbb{Z})^* \setminus G| \geq \phi(N)/2$.

1.5.2 Implementation

Maple Implementation

```
numberOfFactor2 := proc(n::integer)
local v, m;
description "It writes n in the form of (2^v) * m";
m := n; v := 0;
while type(m, even) do ++v; m := 1/2*m; end do;
return v, m; end proc;

firstOccurenceIndex := proc(v::Vector, n::integer)::integer;
local i;
i := 1;
while v(i) <> n do ++i; end do;
return i;
end proc;

StrongPseudoprimalityTest := proc(n::integer)
local a, b, d, j, g, b0, v, m;
description "Strong Pseudoprimality Test.
Also known as Miller-Rabin primality test";
a := rand(2 .. n - 1)();
d := igcd(a, n);
if 1 < d then return d; end if;
v, m := numberOfFactor2(n - 1);
b0 := irem(a^m, n);
if b0 = 1 then return "probably prime"; end if;
b := Vector(v + 1, i -> b0^(2^(i - 1)) mod n);
if b(v + 1) = 1 then j := firstOccurenceIndex(b, 1) - 1;
else return "composite"; end if;
g := igcd(b(j) + 1, n);
if g = 1 or g = n then return "probably prime";
else return g;
end if;
end proc;
```

Mathematica Implementation

```
StrongPseudoprimalityTest[n_] :=
 Module[{a = RandomInteger[{2, n - 1}], b, d, v, m, j, b0, g},
  d = GCD[a, n]; If[d > 1, Return[d]];
  m = NestWhile[#/2 &, n - 1, EvenQ];
  v = IntegerExponent[n - 1, 2];
  b0 = PowerMod[a, m, n]; If[b0 == 1, Return["probably prime"]];
  b = Table[PowerMod[b0, 2^(i - 1), n], {i, 1, v + 1}];
  If[b[[v + 1]] == 1, j = First[FirstPosition[b, 1]] - 1,
   Return["composite"]];
  g = GCD[b[[j]] + 1, n];
  If[g == 1 \[Or] g == n, Return["probably prime"], Return[g]];
  ]
```

References

[1] Wentang Kuo. *PMATH 440/740: ANALYTIC NUMBER THEORY NOTES*. 2021.

[2] Stephen New. *Lecture Notes for PMATH 340, Elementary Number Theory*. 2020.

[3] Primality test. Primality test — Wikipedia, the free encyclopedia, 2022. [Online; accessed 09-March-2022].

[4] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3rd edition, 2013.