PRIMALITY TESTING AND FACTORING ALGORITHMS

SANCHAY DEVNATH

1. Introduction

The most common way to do a primality test of a positive integer n, determine whether n is prime, is to see if $z|n$ given prime $z \in (1, \sqrt{n})$—if false the n is prime. If true, all the values of z for which $z|n$ are a part of the prime factorization of n. Here is an easy demonstration of this. Let us say $n = 91$ and we need to determine whether n is prime, and if it is a composite number what is its prime factorization.

First we find $\sqrt{n}$:

$$\sqrt{91} = 9.53939201$$

We then floor this number because any number greater than it would be redundant for our needs

$$\approx 9$$

Now we check if $z|91$ for $z \in (1, \sqrt{91})$ and is prime:

$$2 \to 2 \nmid 91$$

$$3 \to 3 \nmid 91$$

$$5 \to 5 \nmid 91$$

$$7 \to 7|91$$

Now that we have proved that 91 is composite we must find its prime factorization:

$$91 \div 7 = 13$$

Since 13 is prime, the prime factorization of 91 is:

$$91 = 7 \cdot 13$$

Although this technique is deterministic, mostly determines the primality of a number with complete accuracy, it becomes futile when dealing with bigger numbers. For example, determine whether 1348956739 is prime. If we utilize our previous method, we would first determine $\sqrt{n}$.

$$\sqrt{1348956739} = 36728.1464$$

$$\approx 36728$$

Obviously, it is not optimal to check the divisibility between every single prime in $(1, 36728)$ with 1348956739. Furthermore, we will also need to determine if the numbers in $(1, 36728)$ are prime themselves.

We will discuss the nuances of various primality tests like this in this paper, from those that always provide the right answer to those that are not so accurate but far more efficient.

2. Fermat's Primality Test

Fermat's Test was created by Pierre De Fermat in 1640. It is a deterministic test that revolves around the expression:

$$a^p - a$$

In this formula, p is the number whose primarlity we seek to determine and a is any integer in $(1, p)$. Fermat's Test states that if for all values of a, the expression $a^p - a$ is a multiple of p then p is prime.

Proof. This Primality test is derived from Fermat's other theorem called Fermat's little theorem, which states:

$$a^p \equiv a(\mathrm{mod} p)$$

is true for any prime (p) and any integer (a) in (1,p). This is nothing but a different way of saying $p|(a^p - a)$.

Example. Determine the primality of 5.

$$2 \to 2^5 - 2 = 32 - 2 = 30 \to 5|30$$

$$3 \to 3^5 - 3 = 243 - 3 = 240 \to 5|240$$

$$4 \to 4^5 - 4 = 1024 - 4 = 1020 \to 5|1020$$

Since all integers in $(1, 5)$ output a multiple of 5 when plugged into the formula, 5 is obviously prime. However, although Fermat's test is deterministic (like the traditional square root method) it is not fully accurate in computing the primality of larger numbers (specifically past 511).

3. AKS Test

The AKS primality test (also known as Agrawal–Kayal–Saxena primality test and cyclotomic AKS test) was the first unconditional deterministic primality algorithm published by computer scientists: Manindra Agrawal, Neeraj Kayal, and Nitin Saxena at IIT (Indian Institute of Technology Kanpur) in 2002. The main formula of this test is:

$$(x + 1)^p - (x^p + 1)$$

where x is variable and p is the number whose primality needs to be determined. If the coefficients of the simplified expression of this formula are all

multiples of p, then p is prime.

Example 1. determine the primality of 3.

$$(x+1)^3 - (x^3+1)$$
$$x^3 + 3x^2 + 3x + 1 - x^3 - 1$$
$$3x^2 + 3x$$

3 is obviously a multiple of 3, making 3 prime. Example 2. determine the primality of 4.

$$(x+1)^4 - (x^4+1)$$
$$x^4 + 4x^3 + 6x^2 + 4x + 1 - x^4 - 1$$
$$4x^3 + 6x^2 + 4x$$

In this example the coefficients are: 6, and 4. Obviously 6 is not a multiple of 4, making 4 composite.

Proof. Let us first simplify the formula assuming n is prime using the binomial theorem.

$$(x+1)^n - (x^n+1)$$
$$= \sum_{r=0}^{n} \binom{n}{r} x^r - (x^n - 1)$$
$$= \sum_{r=1}^{n-1} \binom{n}{r} x^r$$

Since we are only concerned with whether n divides the coefficients, we can further simplify this expression into:

$$n \left| \binom{n}{r} \right. \; r \in \; [1, 2, 3...n-1]$$

Further simplified we get this:

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$
$$= \frac{n(n-1)(n-2)...(n-r)!}{r!(n-r)!}$$
$$= \frac{n(n-1)(n-2)...(n-r+1)}{r!}$$

we can now group $\frac{(n-1)(n-2)...(n-r+1)}{r!}$ into θ giving us:

$$n \cdot \theta$$

Since this represents the sum of all the coefficients in the original formula, we can use the distributive property to confirm that n is a factor of each of the coefficients. Keep in mind that this would not be possible if n was composite because then it would be factored out by some r value because $r \in [1, 2, 3...n-1]$.

4. Mills Prime Number Generator

Now we will take a different approach towards primes by discussing how we can generate or, with a big enough computer, create one. In 1947, William Harold Mills proved the existence of A based on results of Guido Hoheisel and Albert Ingham on the prime gaps. Its value is approximately 1.30637788386308 06904686144926... (sequence A051021 in the OEIS). This constant is know as Mills' constant (μ). Mills the following expression utilizing this constant:

$$\lfloor \mu^{3^n} \rfloor$$

Plugging in any positive integer for n will give us a prime number.

$$n = 1 \rightarrow \lfloor \mu^3 \rfloor = 2$$
$$n = 2 \rightarrow \lfloor \mu^9 \rfloor = 11$$
$$n = 3 \rightarrow \lfloor \mu^{27} \rfloor = 1361$$

As you can see not all prime numbers can be produced by this expression, instead a specific sequence of numbers (known as Mills primes) are produced. The sequence looks something like this:

$$2, 11, 1361, 2521008887, 16022236204009818131831320183,$$
$$4113101149215104800030529537915953170486139623539 75$$
$$9933135949994882770404074832568499...$$

Clearly the primes escalate into very large numbers. These numbers are often used in cryptography because it becomes harder for a person to decode a private key as the primes used are larger.

5. Factoring Algorithms

We will now discuss the various different ways to find the prime factorization of a number. The ability to quickly determine the prime factorization of a number is one of the most researched areas of study in modern times—specifically for its relation to RSA cryptography. Cryptography relies on having the prime factorization of huge numbers consisting of large primes as its factors. The difficulty in determining the prime factorization of such large numbers is the reason for the rise in cryptography.

The most basic factorization algorithms is known as Direct Search Factorization. In this method we check every integer if $r|n$ for $r \in (1, \sqrt{n})$ is a factor

of n, n being the number whose prime factorization we must determine, and then proceed to factor. This is the method discussed in the Introduction of this paper.

6. Shor's Algorithm

In 1994, Peter Shor had created one of the most profound factoring algorithms of modern times. This algorithm enabled factorization in the polynomial time with the help of quantum computers—in contrast to the exponential time that was commonly accepted. This reduction in time needed for factorization enabled experts to enhance cryptography.

Theorem. If N is semi-prime, a product of two different primes, and x is an integer that satisfies the following:

$$x^2 \equiv 1(\mathrm{mod}N)$$

$$x \not\equiv 1(\mathrm{mod}N) \quad and \quad x \not\equiv -1(\mathrm{mod}N)$$

Then $\gcd(x-1, N)$ and $\gcd(x+1, N)$ are non-trivial factors of N.
This is the classical part of the algorithm, for which one could use classical computers to factor N.

Proof. From $x^2 \equiv 1(\mathrm{mod}N)$ we can get:

$$N|(x^2-1)$$

From which we get:

$$N|(x+1)(x-1)$$

Therefore if:

$$gcd(x-1, n) = 1$$

Then:

$$N|(x+1)$$

However, $x \not\equiv 1(\mathrm{mod}N)$ *and* $x \not\equiv -1(\mathrm{mod}N)$. This means both $\gcd(x-1, n) > 1$ and $\gcd(x+1, n) > 1$, and are both nontrivial.

Example. Find the prime factorization of 15.
We first guess an integer, x, that satisfies the prerequisites. The lowest number that does so is:

$$x = 4$$

The nontrivial factors of 15 are:

$$gcd(4-1, 15) \quad and \quad gcd(4+1, 15)$$

$$= gcd(3, 15) \quad and \quad gcd(5, 15)$$

Which are $3, and\, 5$.
The ability to identify the nontrivial factors of a semi prime contributed to the $O(log(N))$ time complexity of Shor's Algorithm. This was significantly faster than the previous algorithms that ran on $O(log(N^k)$. This advanced

cryptography by enabling computers to generate bigger numbers, while being able to recognize the factorization of those large numbers.

References

[1]Goswami, Debarate, director. Breaking Classical Codesnbsp; Basics of Shor's Algorithm. YouTube, YouTube, 6 Feb. 2017, https://www.youtube.com . Accessed 25 Dec. 2023.
[2]"Primality Test." Wikipedia, Wikimedia Foundation, 19 Nov. 2023, en.wikipedia.org/wiki/Primalitytest.
[3]"Primality Test." From Wolfram MathWorld, mathworld.wolfram.com/PrimalityTest
[4]Woo, Eddie, director. Primality (2 of 2: AKS Test). YouTube, YouTube, 8 Nov. 2014, https://www.youtube.com/watch?v=D7AHbyAlgIAamp;t=1s. Accessed 24 Dec. 2023.