Olof Persson and Dev Patel

12 December 2022

Primality Testing

In the history of mathematics, the discovery of primes was relatively early as Euclid started studying them and even proved that there is an infinite number of primes. At that time, however, it wasn't really clear if knowing about primes would be important or even useful. Since then there have been many strides in our understanding of primes, even though we still have lots of questions about them. These questions and lack of understanding allow for primes to be powerful tools in cryptography today, and so being able to find these primes efficiently is very important since every exchange on the web basically requires the use of large primes for secure communication. With that being said, the idea of primality testing is the use of different algorithms to search through randomly chosen numbers and check if they are prime. Also, the reason that encryption and decryption requires the use of finding primes is that they need to be random and not reused or the encryption could be easily broken since the primes would be reused too often. We conducted our research on this topic by consulting a wide variety of resources all over the internet. This spanned from YouTube videos, to wikipedia pages and even professional publications.

When it comes to how a primality test checks that a number is prime, there are deterministic and probabilistic tests that determine primality in different ways. A deterministic primality test is one that at the end of the test has made sure to calculate that a number is prime and runs through all the cases that it has to have a 100% confidence that a number is prime or composite, thus deterministic tests tend to take longer. The probability tests are ones that are based on taking certain numbers and testing them against n to see if the formula returns that the number is prime or not. These compared numbers are chosen at random hence the probabilistic way of testing. Also, probabilistic tests don't give 100% certainty that a number is prime or not, but the accuracy goes up with trial of a new random number against n.

The most naive approach to primality testing called trial division is based on the definition of primes that every number between 1 and n (2 to n-1) does not divide n, making it a deterministic approach. The trial division algorithm takes all the values between 1 and n and tries to divide n by all of them and if one of them has no remainder then n is a composite number. This is the simplest method, but in the case that n is very large, then there will be a huge amount of divisions to find out if n is prime. Along with that, it can be optimized so that only the square root of n number of divisions is done, because if n is composite than at least one of the numbers must be less than the square root of n, so a divisor of n would be found in searching all the numbers up to the square root of n. This optimized version while faster is still very inefficient as n becomes very large.

The second naive approach to testing if a number is prime uses Wilson's Theorem that states that a positive integer is prime if $(n-1)! \equiv -1 (\mathrm{mod}\ n)$. This means that the factorial of n-1 must be the modular inverse of -1 to n. The problem with this approach is that it requires the computation a factorial so again as n becomes very large this becomes very inefficient even more inefficient than trial division, but like the trial division it is a deterministic approach meaning that the number is guaranteed to be classified correctly.

The next approach doesn't give such certainty as Fermat's Primality Test is probabilistic. Fermat's Primality Test uses the idea of Fermat's Little Theorem to test that the number is prime. Fermat's Little Theorem states that for a number 'a' that is coprime to p: $a^{p-1} \equiv 1 \ (\mathrm{mod}\ p)$. If for the sake of the test that one assumes that p is prime, then any number p-1 should not divide p and should equal this relation. The problem is that this works for composite numbers too, because composite numbers still have numbers they are coprime to. Values of ‘a’ that make the theorem true, but n is actually composite are called Fermat’s Liar and value p is called a Fermat Pseudoprime to base ‘a’. To make this theorem into a primality test, one has to choose/ multiple different ‘a’ values to make sure that modular inverses of a^{p-1} stay equal to 1. The more ‘a’ values chosen the higher certainty one has that the number p is primes. The other problem with using this test is that certain numbers called Carmichael numbers are pseudoprime to all bases of ‘a’, but are still composite numbers. While Carmichael numbers are very rare, it adds on to the already uncertain probabilistic test.

This leads us to the Miller-Rabin primality test. The Miller-Rabin is a probabilistic algorithm that works in polynomial time named after its inventors, Gary L. Miller and Michael O. Rabin, who developed it in 1976, and is partially based on Fermat’s Primality Test. As mentioned before, this test provides an answer attached with a certain probability of it being correct, rather than an answer that is definite. The probability of using an incorrect answer can be significantly minimized by simply running the algorithm multiple times with the use of different inputs/witnesses. Repeating the algorithm results in a higher probability of the result being correct. The chance that a composite is recognized as a prime is 4^-k for k trials. Therefore repeating the algorithm many times almost ensures a correct result. The algorithm returns false in the case that n is composite. It checks if the number is less than 3 and returns false if it is. It also returns false if the number is even. These are simply the initial checks to minimize the time complexity of the algorithm. Otherwise, choose a witness, w, such that 1 is less than or equal to w and w is also less than n - 1. Use this to calculate the remainder, r, which is the remainder when n - 1 is divided by 2^d / n where d is the largest power of 2 that divides n - 1. If w^r is congruent to 1 in mod n, then the number is probably prime. If w^(2^a * r) is congruent to -1 in mod n for any value of ‘a’ in the range 0 <= a < d, then the number is probably prime. If none of the above conditions are met, then the number is probably composite.

However, the possibilities of the Miller-Rabin do not end here. The use of an upper bound can allow the algorithm to be modified to become deterministic. This can be made possible by fixing ‘a’ to specific values in the test. For example, if n can be upperbounded to 341550071728321, the algorithm can be made deterministic by simply testing for ‘a’ values of a=2,3,5,7,11,13,17. So by confirming that n is less than 341550071728321, checking these seven values of ‘a’ can guarantee a correct result. There are many other deterministic variations for the Miller-Rabin algorithm based on this idea.

In summary, the Miller-Rabin algorithm is a probabilistic algorithm for determining whether a given number is prime or composite. It uses a series of witnesses to test the number and provides a high probability of correctness when used multiple times with different inputs. This algorithm can be said to run in O(k*lg(n)). This is because the preparation takes O(lg(n)) time while the testing takes O(k) trials times O(lg(n)) time for modular exponentiation times the time for multiplication. It is a valuable tool for applications such as cryptography, where the ability to quickly and accurately determine whether a large number is prime is important.

Another very interesting primality test to investigate is the AKS primality test, which is the first polynomial time deterministic algorithm for finding primality of numbers. It was created in 2002, and named after its creators Manindra Agrawal, Neeraj Kayal, and Nitin Saxena at the Indian Institute of Technology. The AKS primality test is a significant advancement in the field of primality testing, as it is the first such algorithm that is both polynomial-time and provably correct. This means that it can determine the primality of a number in a reasonable amount of time, and there is a mathematical proof that guarantees its correctness.

An important aspect of the AKS primality test is its reliance on the concept of polynomial congruences. This is a mathematical relationship between two polynomials, where one polynomial is congruent to the other modulo a given value. By using this concept, the AKS primality test is able to make precise and accurate conclusions about the primality of a number. The AKS primality test works by first checking if the number being tested is a trivial case. This includes numbers less than 2, which are not considered prime, and numbers that are divisible by 2 or 3, which can be quickly determined to be composite. If the number passes this initial check, the AKS primality test proceeds to the main part of the algorithm. The main part of the AKS primality test involves three steps. First, it checks if the number being tested is a power of a smaller number. This can be quickly determined by checking if the number is divisible by any smaller number. If it is not a power of a smaller number, the AKS primality test proceeds to the next step. Next, the AKS primality test checks if the number being tested is a product of two or more smaller numbers. This can be determined using the concept of polynomial congruences, where two polynomials are considered congruent if they have the same value mod n. If the number being tested is not a product of two or more smaller numbers, the AKS primality test proceeds to the final step. In the final step, the AKS primality test checks if the number being tested is prime by checking its divisibility properties. This is done using polynomial modular arithmetic, where mathematical operations are performed mod n. Polynomial modulus has two arguments one that is the modulo for the coefficients of every polynomial term and one for the modulo of the exponents (mod(n, X^r-1) is the mod of the coefficients by n and mod of the exponents by r). If the number passes this final check, it is determined to be prime. Generally speaking, the AKS primality test works by first checking for trivial cases, then using polynomial congruences and polynomial modular arithmetic to determine the primality of the number being tested. By using these mathematical techniques, the AKS primality test is able to make accurate determinations about the primality of a number in a reasonable (polynomial) amount of time. The algorithm of AKS primality test in a put together pseudocode form is shown below.

Input: integer $n > 1$.

```
1. Check if $n$ is a perfect power: if $n = a^b$ for integers $a > 1$ and $b > 1$, output *composite*.
2. Find the smallest $r$ such that $\mathrm{ord}_r(n) > (\log_2 n)^2$. (if $r$ and $n$ are not coprime, then skip this $r$)
3. For all $2 \le a \le \min(r, n-1)$, check that $a$ does not divide $n$: If $a|n$ for some $2 \le a \le \min(r, n-1)$, output *composite*.
4. If $n \le r$, output *prime*.
5. For $a = 1$ to $\left\lfloor \sqrt{\varphi(r)} \log_2(n) \right\rfloor$ do
       if $(X+a)^n \neq X^n + a \pmod{X^r - 1, n}$, output *composite*;
6. Output *prime*.
```

Source: https://en.wikipedia.org/wiki/AKS_primality_test

Overall, the AKS primality test is a highly efficient and reliable method for determining the primality of a number. Its use of polynomial modular arithmetic and polynomial congruences allows it to make accurate determinations in a reasonable amount of time, making it a valuable tool in the field of computational number theory.

After analyzing the Miller-Rabin and AKS primality, the two polynomial time algorithms, it would only be wise to compare them. Starting with Miller-Rabin, it is faster than

AKS in that it is a probabilistic algorithm that can become very accurate very quickly. This is because the formula $1/4^k$ for k trials shows the chance of a composite being labeled a prime. If k is equal to 10, then the probability is less than 0.000001. On the other hand, AKS gives the unconditional correctness that a number is composite or prime, but it has a very very high coefficient and only becomes faster and useful when the numbers become unnecessarily big. Unnecessarily big is meant by the fact that the primes needed to be found in today's everyday use for cryptography are too small to justify using AKS since the coefficient still makes the Miller-Rabin and other primality tests quicker. This means that the AKS test is a huge accomplishment in that it is proved that primality testing is a polynomial time problem and not a non-polynomial time problem, and in the future will probably lead to more amazing discoveries. But today, Miller Rabin is one the most used algorithms for its very high probability of success in the numbers that are used in practice, and even deterministic for small enough numbers if they are so desired.

With the advent of modern computing and cryptography, primality testing and efficient primality tests has become a large industry with cybersecurity. With this realization, the trial division and Sieve of Eratosthenes were seen to be very inefficient for finding large primes, so Fermat’s Little Theorem converted to a probabilistic primality test. Even with that, some better algorithms were needed. This is where the Miller-Rabin Primality Test and AKS Primality Test were born. The Miller-Rabin test is a probabilistic test that doesn’t give 100% certainty that a number is prime, but is very fast and the chances of it being incorrect go down significantly with just a small number of trials/witnesses. Unlike the probabilistic Miller-Rabin, the AKS test is the first deterministic polynomial time algorithm that gives certainty that a number is prime. Even though the AKS is novel and powerful in what it proved, it is still too slow for the primes used today, so really fast algorithms like Miller-Rabin are more widely used in application today.

Works Cited

Koswara, I., Chattopadhyay, A., & Boo, C. (n.d.). *Primality testing*. Brilliant Math & Science Wiki. Retrieved December 12, 2022, from https://brilliant.org/wiki/prime-testing/

Lynn, B. (n.d.). *Primality tests*. Number Theory - Primality Tests. Retrieved December 12, 2022, from https://crypto.stanford.edu/pbc/notes/numbertheory/millerrabin.html

Wikimedia Foundation. (2022, October 19). *AKS Primality Test*. Wikipedia. Retrieved December 12, 2022, from https://en.wikipedia.org/wiki/AKS_primality_test

Wikimedia Foundation. (2022, November 16). *Primality Test*. Wikipedia. Retrieved December 12, 2022, from https://en.wikipedia.org/wiki/Primality_test