The AKS Primality Test

By Tammy Terzian

A Project Presented to

The Faculty of the Mathematics Program

California State University Channel Islands

2013

In Partial Fulfillment of the Requirements

for the Degree of Masters of Science

Department of Mathematics

MS PROJECT BY TAMMY TERZIAN
APPROVED FOR THE MATHEMATICS PROGRAM

Jesse Elliott 1/7/13
Dr. Jesse Elliott Date

[illegible] 1/7/13
Dr. Ivona Grzegorczyk Date

[illegible] 1/7/13
Dr. Brian Sittinger Date

APPROVED FOR THE UNIVERSITY

[illegible] 1-7-13
Dr. Gary A. Berg Date

THE AKS PRIMALITY TEST

Title of Item

AKS PRIMALITY TEST

3 to 5 keywords or phrases to describe the item

Tammy Terzian

Author(s) Name (Print)

[illegible] 1/10/13

Author(s) Signature Date

Acknowledgements

This paper would not have been possible without the support of many people. First I would like to thank my advisor, Dr. Jesse Elliott, for the idea of studying the AKS Primality Test and for all the hours he spent helping me understanding and writing my paper. Thanks to Dr. Buhl for stepping in and being my advisor for a semester. His help in getting me to start my writing process is greatly appreciated. Thanks to Dr. Ivona Grzegorczyk and Dr. Brian Sittinger for being on my committee and reading my paper. Thanks to my fellow classmates and friends in the math department who have been a great encouragement. Lastly, thanks to all my family and friends who have supported me along the way.

Abstract

In 2002 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena discovered an algorithm to test a number for primality that is both deterministic and runs in polynomial time. The AKS algorithm hinges on a calculated value they call r, which is defined for a given integer n >1 as the least value for which the order of n modulo r is greater than $\log_2^2 n$. This r has a proven upper bound of $\log_2^5 n$. In this paper, we prove that $2 + \log_2^2 n$ is a lower bound of the value r, and if n is a square, there is a lower bound of $1 + 2\log_2^2 n$. We also present also data suggesting that $3\log_2^2 n$ is a smaller upper bound of r. If this is indeed an upper bound, the AKS Primality Test is shown to have a time complexity of $O(\log_2^{6+\varepsilon} n)$ in bit operations for any small ε greater than 0. Data also suggests a number n is a square if and only if its corresponding r is greater than $3\log_2^2 n$.

Table of Contents

Chapter 1. Introduction

For millennia, mathematicians have known that there is an infinite number of prime numbers dispersed throughout the set of integers with no completely predictable pattern. This makes finding them a difficult task; one which has fascinated the mathematical community for thousands of years, even to this day. In 2002, Manindra Agrawal, Neeraj Kayal, and Nitin Saxena published the paper *Prime is in P* [1], which contained an incredible breakthrough for finding prime numbers. They introduced the AKS Primality Test, which is a deterministic primality test that runs in polynomial time (polynomial in the bit size of n).

Prior to the AKS algorithm there existed tests that were deterministic but not time efficient, such as a naïve method of checking for factors and the Eratosthenes Sieve. There were also probabilistic tests that run in polynomial time, such as the Fermat Test, the Miller-Rabin Test, and the Solovay-Strassen Test. The AKS Primality Test was the first test that was proven to be both deterministic and run in polynomial time. This test also has the appeal that it is based on basic principles from abstract algebra, the most basic of which is the following.

Theorem 2.2.1 (*Fermat's Little Theorem*). *Let p be a prime. For all integers a*

$$a^p \equiv a (\text{mod } p).$$

Fermat's Little Theorem is a good tool for testing primality; unfortunately it is probabilistic. We reformulate the theorem to the form that serves as the fundamental basis for the AKS Primality Test.

Theorem 2.3.3 (*Fermat's Little Theorem for Polynomials*). *A positive integer n is prime if and only if one has*

$$(X + a)^n - X^n + a \text{ in } \mathbb{Z}_n[X]$$

for all positive integers a such that $a < n$ and $\gcd(a,n) - 1$.
We prove this in chapter 2.

This theorem could be used as a deterministic test for primality if all the values of a less than n are checked. However, it turns out that it would be more efficient to use a naïve method of checking for factors up to $\sqrt{n}$. The AKS Theorem stated below makes a profound improvement regarding the above statement. It says that the values of a to be tested can be limited, and the whole congruence can be reduced modulo a polynomial of degree r where r is an appropriate number computable from n.

Theorem 3.1.1 (*AKS Theorem*). *Assume that n and r are positive integers satisfying the following conditions:*

1. $\text{ord}_r(n) > \log_2(n)^2$.
2. $r < n$ *and* $\gcd(a,n) = 1$ *for all* $a \leq r$.
3. $(X + a)^n \equiv X^n + a \pmod{X^r - 1}$ *in* $\mathbb{Z}_n[X]$ *for all integers* a *from* 1 *to* $\lfloor \sqrt{\phi(r)} \log n \rfloor$.

Then n is a power of a prime.

Remark: Note that all logarithms in this paper are base 2 unless otherwise noted.

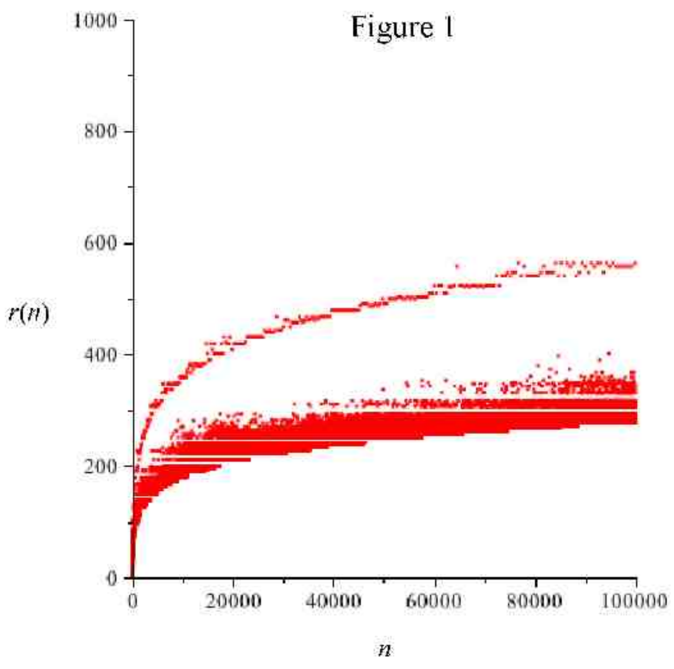
Since to show primality we want n to only be a power of itself, i.e. power one, the AKS Primality Test includes a perfect power test. This eliminates the possibility of n being a perfect power of another number. Thus, if the three conditions in the AKS Theorem hold, then n itself is prime. Combining this information, we can state the AKS Primality Test.

AKS Primality Test.

Input: integer $n > 1$ (a positive integer to test for primality)

```
1.    If ($n = a^b$ for $a$ in $\mathbb{N}$ and $b > 1$), output *composite*.
2.    Find the smallest $r = r(n)$ such that $\text{ord}_r(n) > \log^2 n$.
3.    If $1 < \gcd(a,n) < n$ for some $a \le r$, output *composite*.
4.    If $n \le r$, output *prime*.
5.    For $a = 1$ to $\lfloor \sqrt{\phi(r)} \log n \rfloor$ do
          if $(X + a)^n \not\equiv X^n + a \pmod{X^r - 1}$ in $\mathbb{Z}_n$, output *composite*.
6.    Else, output *prime*.
```

In the paper *Prime is in P*[1], the r value in step 2 is proven to exist and have an upper bound of $\log^5(n)$. This upper bound plays an important role in finding the total run time of the algorithm and proving that the algorithm works. Figure 1 displays the values of n from 1 to 100,000 and their respective r values.



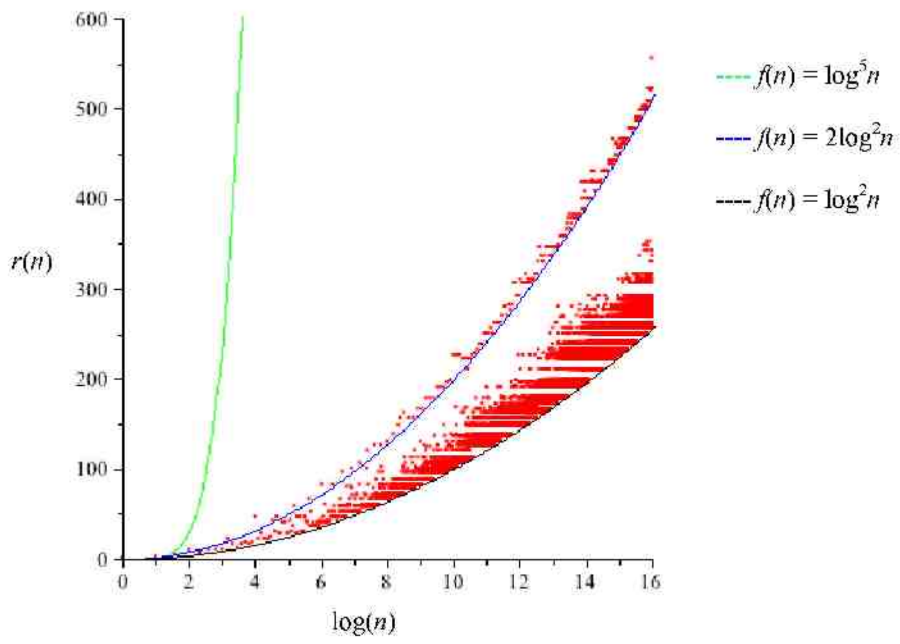
From the graph, it is noticeable that there are two distinct sections. The perfect squares are on top and the non-squares are on the bottom. The following two theorems give lower bounds for $r(n)$, proved in chapter 4.

Theorem 4.5. *A lower bound of* $r(n)$ *is* $2 + \log^2(n)$.

Theorem 4.6. *For any perfect square* $n > 1$, *a lower bound of* $r(n)$ *is* $1 + 2\log^2 n$.

Since the lower bounds are both in logarithmic form, as is the proven upper bound of $\log^5 n$, Figure 2 shows all three bounds and the points in a logarithmic scale.

Figure 2



Although these results regarding lower bounds are interesting, finding an upper bound that is smaller than $\log^5 n$ has more use. The graph in Figure 2 suggests that the lower bound for the squares might be an upper bound for the non-squares. Hence, we formulate the following conjectures.

Conjecture 4.7. *For any positive integer n such that n is not a perfect square,* $r(n) \leq 2\log^2 n$ *except for* $n = 2, 23, 335$.

Now including the perfect squares, it seems reasonable to predict the following.

Conjecture 4.8. *For any positive integer n an upper bound for* $r(n)$ *is* $3\log^2 n$ *for all n*.

We have verified both conjectures for $n \leq 7{,}703{,}162$. Recall that the time complexity of the AKS Primality Test is highly dependent on r. Hence, if the above conjecture is found to be true, the following corollary is also true.

Corollary 4.9. *The time complexity of the AKS algorithm in bit operations is* $O(\log^{6+\varepsilon} n)$ *for a small positive* ε.

Chapter 2. Background

Finding prime numbers may seem like a simple task; one could use the naïve method of testing for divisors or the Eratosthenes' Sieve. Although both methods are deterministic algorithms, when used with large numbers, these algorithms are extremely time consuming. Therefore, people developed more direct ways to test for primality for example the Fermat Test, the Miller-Rabin Test, and the Solovay-Strassen Test. Until the development of the AKS algorithm, these tests were either probabilistic or did not run in polynomial time. The modern way to test for primality is by using methods based on properties of elliptic curves [11].

2.1 Ring $\mathbb{Z}_n$

Definition 2.1.1. *For $n > 2$, let $\mathbb{Z}_n$ be the set $\{0, 1, \ldots, n-1\}$, where addition and multiplication is defined to be addition mod n and multiplication mod n.*

Definition 2.1.2. *The multiplicative group $\mathbb{Z}_n^*$ is the set*

$\{a \in \mathbb{Z}_n | 1 \leq a < n, \text{and} \gcd(a, n) = 1\}$.

Definition 2.1.3. *Let $\varphi(n) = |\mathbb{Z}_n^*|$, the function φ is called the Euler φ function, or Euler totient.*

Example. The group $\mathbb{Z}_6^* = \{1, 5\}$ and $|\mathbb{Z}_6^*| = \varphi(6) = 2$.

The Euler φ function has some noteworthy properties.

(1) $\varphi(n) = n - 1$ if and only if n is prime.

(2) $\varphi(n)$ is even for all $n > 2$.

Definition 2.1.4. *The order of a mod n, denoted* $\mathrm{ord}_n(a)$, *is the smallest positive integer k for which*

$$a^k \equiv 1 \pmod{n}$$

where a and n are relatively prime positive integers.

Note that $\mathrm{ord}_n(a)$ is finite.

Example. $\mathrm{ord}_5(13)$:

$$13^1 \equiv 3 \pmod{5}$$

$$13^2 = 3 \cdot 3 \equiv 9 \equiv 4 \pmod{5}$$

$$13^3 = 13^2 \cdot 13^1 \equiv 4 \cdot 3 \equiv 12 \equiv 2 \pmod{5}$$

$$13^4 = 13^3 \cdot 13^1 \equiv 2 \cdot 3 \equiv 6 \equiv 1 \pmod{5}$$

$$\mathrm{ord}_5(13) = 4$$

2.2 Fermat's Primality Test

The Fermat Test, stated below, is a test based on Fermat's Little Theorem.

Theorem 2.2.1 (*Fermat's Little Theorem*)[3]. *Let p be a prime. For all integers a*

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

Example. Let $p = 11$. The following chart shows the values of a paired with its corresponding $a^{11}(\text{mod } 11)$. The values $a^{11}(\text{mod } 11)$ were computed using successive squaring.

$\boldsymbol{a}$	$\boldsymbol{a^{11}(\textbf{mod } 11)}$
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

This example shows that since 11 is a prime number, Fermat's Little Theorem holds true for all integers a with $1 \leq a < n$. However, notice that the converse of Fermat's Theorem is not a true statement. That is to say that for a less than p, if $a^p \equiv a(\text{mod } p), p$ is not necessarily prime. Observe the following example.

Example. Let $p = 35$. The following chart shows the values of a paired with its corresponding $a^{35}(\text{mod } 35)$. The values $a^{35}(\text{mod } 35)$ were computed using successive squaring.

a	$a^{35} (\bmod 35)$	a	$a^{35} (\bmod 35)$
1	**1**	18	2
2	18	19	24
3	12	**20**	**20**
4	9	**21**	**21**
5	10	22	8
6	**6**	23	32
7	28	24	19
8	22	25	30
9	4	26	31
10	5	27	13
11	16	28	7
12	3	**29**	**29**
13	27	30	25
14	**14**	31	26
15	**15**	32	23
16	11	33	17
17	33	**34**	**34**

Notice the highlighted values for *a* when $a^{35} \equiv a (\bmod 35)$, this may according to Fermat's Little Theorem, lead one to believe that 35 is prime. Looking at *a* = 5, one has $5^{35} \equiv 10 (\bmod 35)$, which shows that 35 is composite. Since 35 = 5 7, it is in fact composite. For larger numbers, however, it might not be possible to recognize the prime divisors quickly. Both 6 and 5 have a formal name. We call 6 a *Fermat liar* for 35, because it leads one to conclude that 35 is prime. Similarly, we call 5 a *Fermat witness* for 35, because it is a witness that 35 is in fact composite [3].

Definition 2.2.2. *For any composite integer n, an integer a is a Fermat liar for n*

$$a^n \equiv a (\bmod n).$$

Definition 2.2.3. *For any composite integer n, an integer a is a Fermat witness for n*

$$a^n \not\equiv a (\bmod n).$$

These observations gives us the following test.

The Fermat Primality Test.

Input: n (a positive integer to test for primality), $k > 1$ (number of iterations to check for primality)

1. Pick an integer a randomly between 1 and $n-1$.
2. If $a^n \not\equiv a (\text{mod } n)$ then return *composite*.
3. Repeat k times, steps 1 and 2.
4. Else return *probable prime*.

This algorithm takes a given positive integer n and randomly picks an integer a between 1 and $n - 1$ to check if Fermat's Little Theorem holds true. If the theorem does not hold true, then n is composite. However, if the theorem does hold true, it is not guaranteed that n is prime, because integer a could be a Fermat Liar. Thus, the algorithm outputs *probable prime*. To assure that the test outputs the most accurate results, the algorithm runs this test k times, which will increase the probability that the outcome is correct. Notice that returning to step 1 allows a to be the number that was already used for testing.

In the previous example when $n = 35$, the Fermat Liars for 35 are 1, 6, 14, 15, 20, 21, 29, and 34. The probability of choosing a Fermat Liar for 35 is $\frac{8}{34} = \frac{4}{17}$. Now assume the test runs two times. The probability of choosing a Fermat Liar both times would be $\frac{4}{17} \cdot \frac{4}{17} = \frac{16}{289}$. If the test repeats k times for the same scenario, the probability of returning *probable prime* for 35 is $\left(\frac{4}{17}\right)^k$, hence, for large k this value quickly approaches zero.

Note that 1 and $n - 1$ are always *Fermat liars* for n. The Fermat Test has a bound for the probability of error for any given composite n. Let F_L be the set of *Fermat liars*. This set is a subgroup of $\mathbb{Z}_n^*$ because it has the multiplicative identity 1 and it is closed under multiplication (since the group is finite, inverses exist).

Theorem 2.2.4. *Given that there is at least one Fermat witness, the probability of Fermat's Primality Test being wrong for a composite number n is less then* $\left(\frac{1}{2}\right)^k$.

Proof. Assume there is at least one Fermat Witness in $\mathbb{Z}_n^*$. Therefore,

$$|F_L| < |\mathbb{Z}_n^*|$$

and

$$|F_L| \text{ divides } |\mathbb{Z}_n^*|.$$

Since $|\mathbb{Z}_n^*| < n - 1$ or $|\mathbb{Z}_n^*| \leq n - 2$,

$$|F_L| \leq \frac{(n-2)}{2}.$$

Therefore, the probability that there is a Fermat Liar given one iteration, is

$$\frac{\frac{n-2}{2} - 2}{n-3} = \frac{(n-6)}{2(n-3)} < \frac{1}{2}.$$

So with *k* iterations, the probability of a Fermat Liar is less than $\left(\frac{1}{2}\right)^k$ □

Recall that Theorem 2.2.4 makes the assumption that there exists at least one *Fermat witness* in the set of $\mathbb{Z}_n^*$. However, for *n* composite there is not always a *Fermat witness* in $\mathbb{Z}_n^*$. We call these values *Carmichael numbers*.

Definition 2.2.5. *A Carmichael number is a composite integer c that satisfies*

$$a^C \equiv a \ (\mathrm{mod}\ c)$$

for any positive integer a < c.

Therefore, a Carmichael number will always return *probable prime* when really it is a composite.

The first five Carmichael numbers are 561, 1105, 1729, 2465, and 2821 [3].

2.3 Polynomial Ring $\mathbb{Z}_n[X]$

Notation 2.3.1. We let $\mathbb{Z}_n[X]$ denote the ring of all polynomials with coefficients in $\mathbb{Z}_n$.

Example. In $\mathbb{Z}_5[X]$, one has

$$8X^6 + 14X + 3 = 3X^6 + 4X + 3 \text{ in } \mathbb{Z}_5[X].$$

Theorem 2.3.2. *Let r be a positive integer, then*

$$X^n = X^{n \bmod r} \pmod{X^r - 1} \text{ in } \mathbb{Z}_n[X].$$

Example. In $\mathbb{Z}_7[X]$, one has

$$12X^{12} + 13X^9 + 3X^2 + 16 \pmod{X^5\text{-}1}$$

$$= 5X^2 + 6X^4 + 3X^2 + 2 \pmod{X^5\text{-}1}$$

$$= 6X^4 + 8X^2 + 2 \pmod{X^5\text{-}1}$$

$$= 6X^4 + X^2 + 2 \pmod{X^5\text{-}1}.$$

Theorem 2.3.3 (*Fermat's Little Theorem for Polynomials*). *A positive integer n is prime if and only if one has*

$$(X + a)^n = X^n + a \text{ in } \mathbb{Z}_n[X]$$

for all positive integers a such that $a < n$ and $\gcd(a, n) = 1$.

Proof.

$\Rightarrow$ Assume n is prime. We must show that $(X + a)^n \equiv X^n + a \pmod{n}$ for all a such that $a < n$ and $\gcd(n, a) = 1$. By the binomial theorem one has

$$(X + a)^n = X^n + \binom{n}{1} X^{n-1}a + \binom{n}{2} X^{n-2}a^2 \dots + \binom{n}{n-1} Xa^{n-1} + a^n.$$

Since n is prime, one has for all integers k such that $1 \leq k < n$,

$$\binom{n}{k} \equiv 0 \pmod{n}.$$

By Fermat's Little Theorem

$$a^n \equiv a \pmod{n}.$$

Therefore, the congruence $(X + a)^n \equiv X^n + a \pmod{n}$ holds.

$\Leftarrow$ Assume n is not prime. Show that $(X + a)^n \not\equiv X^n + a \pmod{n}$.

By the Binomial Theorem

$$(X + a)^n = X^n + \binom{n}{1}X^{n-1}a + \binom{n}{2}X^{n-2}a^2 \ldots + \binom{n}{n-1}Xa^{n-1} + a^n.$$

Let k be a prime divisor of n and let s be a positive integer such that $k^s \mid n$ but $k^{s+1} \nmid n$.

Consider each individual term

$$\binom{n}{k}a^k = \frac{n(n-1)(n-2)\cdots(n-k+1)a^k}{k(k-1)(k-2)\cdots 1}.$$

Let's compare the numerator and the denominator. The integer n is divisible by k^s but not k^{s+1}. Since a is relatively prime to n, the value a^k is not divisible by k. The remaining values $n - 1, n - 2, \ldots, n - k + 1$ are also not divisible by k, since n is divisible by k and there are only $k - 1 < n$ numbers. Therefore, the numerator is divisible by k^s but not k^{s+1}. The denominator is only divisible by k^1. The equation can now be rewritten as

$$\frac{k^s \cdot (\frac{n}{k^s})(n-1)(n-2)\cdots(n-k+1)}{k \cdot (k-1)(k-2)\cdots 1}a^k.$$

Thus, this expression is divisible by k^{s-1} but not by k^s. This means that this expression is not divisible by n. Therefore,

$$\binom{n}{k}a^k \not\equiv 0 \pmod{n},$$

and

$$(X + a)^n \not\equiv X^n + a \pmod{n} \square$$

Note that this proof shows that all monomials of degree k^s are not congruent to 0 mod n.

2.4 Successive Squaring

Successive squaring is the most effective way to compute exponents of the form a^k, where a is a positive integer. We use the process in programming the AKS algorithm later in this paper. It is best explained through an example.

Example. Consider the value 3^{11}.

First convert 11, the exponent, into it's binary form.

$$11 = 1011_2$$

This means $11 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 8 + 0 + 2 + 1$.

Therefore, $3^{11} = 3^{8-0+2+1} = 3^8\, 3^0\, 3^2\, 3^1$, hence all exponents are powers of 2.

Next evaluate each of these values by repeatedly squaring 3. Repeat the squaring process until the digits of the binary number have been exhausted.

$$3^1 = 3$$

$$3^2 = 9$$

$$3^4 = 3^2 \cdot 3^2 = 9 \cdot 9 = 81$$

$$3^8 = 3^4 \cdot 3^4 = 81 \cdot 81 = 6561$$

Now the required values have been calculated.

$$3^{11} = 3^8\, 3^2\, 3^1 = 6561 \cdot 9 \cdot 3 = 177147$$

In the AKS algorithm, the rules of successive squaring are applied to polynomials in the polynomial ring $\mathbb{Z}_n[X]$.

Example. Evaluate $(X + 3)^5$ in $\mathbb{Z}_{11}[X]$. We will apply successive squaring to calculate. Once again convert 5 to binary.

$$5 = 101_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 = 4 + 0 + 1$$

The result is $(X + 3)^{4 \cdot 0 \cdot 1} = (X + 3)^4 (X + 3)^1$.

Now we calculate squares, and reduce the coefficients mod 11.

$$(X + 3)^1 = X + 3$$

$$(X + 3)^2 = X^2 + 6X + 9$$

$$(X + 3)^4 = (X + 3)^2 (X + 3)^2 = (X^2 + 6X + 9)(X^2 + 6X + 9)$$

$$= X^4 + X^3 + 10X^2 + 9X + 4.$$

Hence,

$$(X + 3)^5 = (X + 3)^4 (X + 3)^1 = (X^4 + X^3 + 10X^2 + 9X + 4)(X + 3)$$

$$= X^5 + 4X^4 + 2X^3 + 6X^2 + 9X + 1.$$

As seen above, the result of this process quickly escalates. In order to slow the escalation, the AKS algorithm takes the modulo of these polynomials by a specific polynomial with degree r. To see how this works, look at the following example.

Example.

$$(X + 2)^{11} \pmod{X^4 - 1} \text{ in } Z_{11}[X]$$

From the previous example $11 = 1011_2 = 8 + 0 + 2 + 1$.

At the beginning of squaring, reduce mod $X^4 - 1$ and mod 11 at each step so that it will reduce the size of the polynomial and be more efficient in calculating the result.

$$(X+2)^1 = X+2$$

$$(X+2)^2 = X^2 + 4X + 4.$$

Notice that so far neither the coefficients nor exponents can be reduced.

$$(X+2)^4 = (X+2)^2\,(X+2)^2 = (X^2+4X+4)(X^2+4X+4)$$

$$= X^4 + 8X^3 + 24X^2 + 32X + 16.$$

Since the polynomial is in $\mathbb{Z}_{11}[X]$, the coefficients reduce modulo 11 and we get

$$= X^4 + 8X^3 + 2X^2 + 10X + 5.$$

In addition, working modulo $X^4 - 1$, the exponents can also be reduced.

$$= 1 + 8X^3 + 2X^2 + 10X + 5.$$

Finally, combining like terms we get

$$(X+2)^4 = 8X^3 + 2X^2 + 10X + 6$$

$$(X+2)^8 = (X+2)^4(X+2)^4 = (8X^3 + 2X^2 + 10X + 6)(8X^3 + 2X^2 + 10X + 6)$$

$$= 64X^6 + 32X^5 + 164X^4 + 136X^3 + 124X^2 + 120X + 36.$$

Since the polynomial is in $\mathbb{Z}_{11}[X]$, the coefficients reduce modulo 11, and we get

$$= 9X^6 + 10X^5 + 10X^4 + 4X^3 + 3X^2 + 9X + 3.$$

In addition, working modulo $X^4 - 1$ the exponents can also be reduced

$$= 9X^2 + 10X^1 + 10 + 4X^3 + 3X^2 + 9X + 3.$$

Finally, combine like terms and reduce any coefficients modulo 11 if needed. We get

$$= 4X^3 + 12X^2 + 19X + 13$$

$$= 4X^3 + X^2 + 8X + 2.$$

Now evaluate the original polynomial $(X+2)^{11}$.

$$(X+2)^{11} = (X+2)^8(X+2)^2(X+2)$$

$$= (4X^3 + X^2 + 8X + 2)(X^2 + 4X + 4)(X+2)$$

$$= 4X^6 + 25X^5 + 62X^4 + 94X^3 + 116X^2 + 88X + 16$$

$$= 4X^2 + 3X + 7 + 6X^3 + 6X^2 + 16$$

$$= 6X^3 + 10X^2 + 3X + 1.$$

The following algorithm calculates powers if integers using the method presented by the examples.

Pseudocode. Fast Exponentiation using successive squaring for integers [2].

Input: Monoid M, element a of M, and positive integer n.

```
1.  $u := n$.
2.  $s := a$.
3.  $c := 1$.
4.  While $u \geq 1$ repeat.
5.      if $u$ is odd then $c := c*s$.
6.      $s := s*s$ mod $n$.
7.      $u := u/2$.
8.  Return $c$.
```

The above pseudocode computes a^n in M. However, the focus of this work is on powers of polynomials. Therefore, we made the following adaptations so that the pseudocode works for polynomials.

Pseudocode. Fast Exponentiation of Polynomials

Input: Positive integers n, r, and a

```
1.  $u := n$.
```

```
2.    $s := x + a \pmod{n}$.
3.    $c := 1$.
4.    While $u \geq 1$ repeat.
5.        if $u$ is odd then $c := (c * s \bmod x^r - 1) \bmod n$; $u := u - 1$.
6.        $s := (s * s \bmod x^r - 1) \bmod n$.
7.        $u := u/2$.
8.    Return $c$.
```

The above algorithm computes $(x + a)^n \pmod{X^r - 1}$.

Chapter 3: AKS Primality Test

Pseudocode. AKS Primality Test. [1]

```
Input: Integer $n > 1$

1.  If ($n = a^b$ for $a$ in $\mathbb{N}$ and $b > 1$), output *composite*.
2.  Find the smallest $r$ such that $\text{ord}_r(n) > \log^2 n$.
3.  If $1 < \gcd(a,n) < n$ for some $a \leq r$, output *composite*.
4.  If $n \leq r$, output *prime*.
5.  For $a = 1$ to $\lfloor\sqrt{\phi(r)}\log n\rfloor$ do
        if $(X+a)^n \not\equiv X^n + a \pmod{X^r - 1}$ in $\mathbb{Z}_n[X]$, output *composite*.
6.  Else, output *prime*.
```

3.1 Proof of Correctness

The AKS Primality Test is based on the following theorem.

Theorem 3.1.1 (*AKS Theorem*). *Assume that n and r are positive integers satisfying the following conditions:*

1. $\text{ord}_r(n) > \log^2 n$.
2. $r < n$ *and* $\gcd(a,n) = 1$ *for all* $a \leq r$.
3. $(X^r + a)^n \equiv X^n + a \pmod{X^r - 1}$ *in* $\mathbb{Z}_n[X]$ *for all integers a from* 1 *to* $\lfloor\sqrt{\phi(r)}\log n\rfloor$.

Then n is a power of a prime.

Notice that this theorem looks similar to Theorem 2.3.3. However, Theorem 2.3.3 states that the congruence in part 3 of the AKS Theorem will hold true for all values of r and a

if and only if n is prime. Note that for some composite numbers, there may be some values of a and r for which this congruence is also true, but it will not hold true for all values of a and r. However, in general it is not possible to check every value for r and a in polynomial time. Therefore, Agrawal, Saxena and Kayal made a great discovery when they limited the set of possible values of a needed to verify primality.

Assuming Theorem 3.1.1, we prove that the AKS Primality Test is correct. The first step of the AKS Primality Test is to check if n is a perfect power of a number other than itself. By the AKS Theorem, n is a perfect power of a prime if the three stated conditions hold. Step 1 rules out the case for which n is a perfect power of another number. Therefore, if the hypothesis of the AKS Theorem applies, and n is not a perfect power, then n must be prime.

In step 2, the algorithm finds the specific r value that will satisfy condition 1 of the AKS Theorem. In order for the AKS Primality Test to run in polynomial time, we need to make sure that there is a small enough upper bound for r. We also have the following lemma.

Lemma 3.1.2. *There exists an* $r \leq \max\{3, \lceil \log^5 n \rceil\}$ *such that*

$$\mathrm{ord}_r(n) > \log^2 n.$$

Proof. When $n = 2$, the value $r = 3$ works. So, assume $n > 2$, then

$$\lceil \log^5 n \rceil \geq \lceil \log^5 3 \rceil > 10.$$

Therefore, the $\max\{3, \lceil \log^5 n \rceil\}$ is $\lceil \log^5 n \rceil$.

By the AKS theorem if $n > 3$, there exists an $r \leq \lceil \log^5 n \rceil$ such that $\mathrm{ord}_r(n) > \log^2 n$.

Consider the smallest integer r that does not divide the product

$$n^{\lfloor \log B \rfloor} \prod_{i=1}^{\lfloor \log^2 n \rfloor} (n^i - 1),$$

where $B = \lceil \log^5 n \rceil$. Observe that the largest value k for any integer of the form $m^k \leq B$, where $m > 2$, is $k \leq \lfloor \log B \rfloor$. Since it is desired that $r \leq B$, any prime divisor of r is going to have a power of at most $\lfloor \log B \rfloor$. Therefore, since r does not divide $n^{\lfloor \log B \rfloor}$, there must exist at least one prime divisor of r that does not divide gcd(r,n). Thus, the quotient $\frac{r}{\gcd(r,n)}$ also does not divide the above product. Since r is chosen to be the smallest r that does not divide the product, it follows that gcd(r,n) is 1. Also, since r does not divide any of the values $(n^i - 1)$ for $1 \leq i \leq \log^2 n$, one has $\mathrm{ord}_r(n) > \log^2 n$.

Finally,

$$n^{\lfloor \log B \rfloor} \prod_{i=1}^{\lfloor \log^2 n \rfloor} (n^i - 1) < n^{\lfloor \log B \rfloor + \frac{1}{2} \log^2 n (\log^2 n - 1)} < n^{\log^4 n} \leq 2^{\log^5 n}$$

And $2^{log^5 n} \leq$ the lcm of the first $\log^5 n$ numbers.

Therefore,

$$r \leq \lceil log^5 n \rceil.$$

In step 3, the algorithm checks the integers less than or equal to r to see if they have any common factor with n, i.e. if there is a relatively small divisor of n. If a common divisor is found, then n is composite and step 3 returns composite.

Step 4 checks if n is less than or equal to r. If n is less than r, then n must be prime, otherwise step 3 would have found n to be composite.

Step 5 checks the final condition of the AKS Theorem. If for any value of a the condition doesn't hold true, by Theorem 2.3.3 and the AKS Theorem n is composite. If for all the specified values of a, the condition holds true then by the AKS Theorem n is prime. This proves that the primality test is correct, assuming theorem 3.1.1.

Now, in order to prove Theorem 3.1.1, we assume conditions 1 through 3 of the theorem hold. We must show that n is a power of a prime.

From condition 1, $\text{ord}_r(n) > 1$. This means there exists a prime divisor p of n such that $\text{ord}_r(p) > 1$. This implies $n, p \in \mathbb{Z}_r^*$.

Since for $q|m$, $a \equiv b \pmod{m}$ in $\mathbb{Z}_m \Rightarrow a \equiv b \pmod{q}$ in $\mathbb{Z}_q$, then

$$(X + a)^n \equiv X^n + a \pmod{X^r - 1} \text{ in } \mathbb{Z}_p[X]$$

for all $1 \leq a \leq \lfloor\sqrt{\phi(r)}\log n\rfloor$.

By Theorem 2.4.3, since p is prime

$$(X + a)^p \equiv X^p + a \pmod{X^r - 1} \text{ in } \mathbb{Z}_p[X]$$

for all $1 \leq a \leq \lfloor\sqrt{\phi(r)}\log n\rfloor$.

Both n and p are *introspective numbers for the polynomial* $f(X) = X + a$ as defined below.

Definition 3.1.3. For a polynomial $f(X)$ and a positive integer m, we say that m is *introspective for* $f(X)$ if

$$[f(X)]^m \equiv f(X^m) \pmod{X^r - 1} \text{ in } \mathbb{Z}_p[X].$$

The following two lemmas describe properties of introspective numbers.

Lemma 3.1.4. *Introspective numbers are closed under multiplication.*

Proof. Let m and n be introspective for $f(X)$. We must show

$$[f(X)]^{mn} \equiv f(X^{mn}) (\text{mod } X^r - 1) \text{ in } \mathbb{Z}_p[X].$$

Since m is introspective for $f(X)$, we have

$$[f(X)]^{mn} \equiv [f(X^m)]^n (\text{mod } X^r - 1) \text{ in } \mathbb{Z}_p[X].$$

Since n is also introspective for $f(X)$, the variable X is replaced with X^m to get

$$[f(X^m)]^n \equiv f(X^{mn}) (\text{mod } X^{mr} - 1) \text{ in } \mathbb{Z}_p[X].$$

Since $X^r - 1$ divides $X^{mr} - 1$ in $\mathbb{Z}_p[X]$, the above outcome can be rewritten as

$$[f(X^m)]^n \equiv f(X^{mn}) (\text{mod } X^r - 1) \text{ in } \mathbb{Z}_p[X].$$

Therefore,

$$[f(X)]^{mn} \equiv f(X^{mn}) (\text{mod } X^r - 1) \text{ in } \mathbb{Z}_p[X]$$

Lemma 3.1.5. *The set of polynomials for which m is introspective, is closed under multiplication.*

Proof. Let $g(X)$ and $f(X)$ be polynomials for which m is introspective. Then

$$[f(X)g(X)]^m \equiv [f(X)]^m [g(X)]^m \equiv g(X^m) f(X^m) (\text{mod } X^r - 1) \text{ in } \mathbb{Z}_p[X]$$

Let I be the set

$$I = \{n^i p^j | i, j \geq 0\}.$$

The set I is a set of introspective numbers for all polynomials in the set

$$P = \{\textstyle\prod_{a=0}^{l} (X + a)^{e_a} | e_a \geq 0\}.$$

Notice that I and P are only sets. These sets will be used to define two groups. The reason for defining these two new groups is to find an upper and lower bound on the size of the groups. This is essential to reaching a contradiction in the proof. Let G be the group defined by

$$G = \{i \bmod r \mid i \in I\}.$$

This group consists of the residues of I mod r. By doing this, a group is created because $n, p \in \mathbb{Z}^*_r$: it is the subgroup of $\mathbb{Z}^*_r$ generated by n and p. It follows that G is also generated by n/p and p. Thus we may replace I with the set

$$I = \left\{\left(\frac{n}{p}\right)^i (p)^j \middle| i, j \geq 0\right\}.$$

The order of this group we denote by t. Note that since $\operatorname{ord}_r(n) > \log^2 n$ and $t \geq \operatorname{ord}_r(n)$, we have $t > \log^2 n$.

Now, make the set P into a group by deriving their inverses. Let the polynomial $h(x)$ be an irreducible factor of the rth cyclotomic polynomial in $\mathbb{Z}_p[X]$. The polynomial $h(x)$ has order $\operatorname{ord}_r(p)$ (See Lemma A.1 of the appendix). Let us take the set P modulo $h(x)$ in $\mathbb{Z}_p[X]$; this will give inverses both the coefficients and exponents. Notice that the group H is generated by $X, X + 1, X + 2, \ldots, X + \lfloor\sqrt{\phi(r)}\log n\rfloor$ in the field $F = \mathbb{Z}_p[X]/h(x)$. This group can be thought of as the group of all the residues of polynomials in P mod $h(x)$.

Now let's consider the lower and upper bound of the size of group H.

Lemma 3.1.6. $|H| \geq \binom{t + \lfloor\sqrt{\phi(r)}\log n\rfloor}{t - 1}$.

Proof. Since $h(X)$ is a factor of the rth cyclotomic polynomial $Q_r(X)$, then X is a primitive rth root of unity in F. First, observe that the polynomials in the group P with degree less than t as defined before, are mapped to distinct elements in H. Let $f(X) \neq g(X) \in P$ and let the degree of f and g be less than t. Suppose $f(X) = g(X)$ in the field F. We show that $f(X) \neq g(X)$ by finding a contradiction.

Let $m \in I$, then

$$[f(X)]^m = [g(X)]^m \text{ in } F.$$

Since m is introspective for f and g, and $h(x)$ divides $X^r - 1$, then

$$f(X^m) = g(X^m).$$

This implies that X^m is a root of the polynomial

$$Q(Y) = f(Y) - g(Y) \text{ for all } m \in G.$$

Since G is a subgroup of $\mathbb{Z}_r^*$, $\gcd(m,r) = 1$ and thus each such X^m is a primitive rth root of unity. Since there are t elements in G, there are t distinct roots of $Q(Y)$ in F. However, $g(X)$ and $f(X)$ were picked to have a degree less than t. This is a contradiction of degrees. Therefore, $f(X) \neq g(X)$ in F.

Since $\lfloor\sqrt{\phi(r)}\log n\rfloor < \sqrt{r}\log n < r$ and $p > r$, we know that the integers 1 through $\lfloor\sqrt{\phi(r)}\log n\rfloor$ are distinct in F. So, the elements $X, X+1, X+2, \ldots X+\lfloor\sqrt{\phi(r)}\log n\rfloor$ are all distinct in F. Now observe the polynomials with degree less than $t - 1$. The polynomial

$$f(X) = a_{t-1}X^{t-1} + a_{t-2}X^{t-2} + \ldots a_2X^2 + a_1X^1 + a_0X^0$$

such that $0 \leq a_i \leq \lfloor\sqrt{\phi(r)}\log n\rfloor$,

$$f(X) = \sum_0^{\lfloor\sqrt{\phi(r)}\log n\rfloor} a_i x^i.$$

The total possible set to choose from has $(\lfloor\sqrt{\phi(r)}\log n\rfloor + 1) + (t - 1)$ elements, which is $t + \lfloor\sqrt{\phi(r)}\log n\rfloor$. Now choose $t - 1$ of these elements. Therefore,

$$|H| \geq \binom{t + \lfloor\sqrt{\phi(r)}\log n\rfloor}{t-1}.$$

Having found a lower bound, now we find an upper bound.

Lemma 3.1.7. *If n is not a power of p, the size of H is bounded above as follows:*

$$|H| \leq n^{\sqrt{t}}.$$

Proof. Consider the following subset

$$I = \left\{\left(\frac{n}{p}\right)^i p^j \middle| 0 \leq i, j \leq \lfloor\sqrt{t}\rfloor\right\}.$$

If n is not a power of p, then number of distinct elements in I is equal to $\left(\lfloor\sqrt{t}\rfloor + 1\right)^2 > t$. Since the order of G is t, at least 2 numbers in I must be congruent modulo r. Let these be m_1 and m_2 with $m_1 > m_2$. So, one has

$$X^{m_1} \equiv X^{m_2} \pmod{X^r - 1}$$

in $\mathbb{Z}_p[X]$. Let $f(X)$ be in P. Then

$$[f(X)]^{m_1} \equiv f(X^{m_1}) \equiv f(X^{m_2}) \equiv [f(X)]^{m_2} \pmod{X^r - 1}$$

in $\mathbb{Z}_p[X]$. Thus,

$$[f(X)]^{m_1} = [f(X)]^{m_2}$$

in F. Thus, $f(X)$ in H is a root of $Q'(Y) = Y^{m_1} - Y^{m_2}$ in F.

Since $f(X)$ is arbitrary in H, the polynomial $Q'(Y)$ has at least $|H|$ roots in F. The degree of $Q'(Y)$ is

$$m_1 \leq \left(\frac{n}{p} \cdot p\right)^{\lfloor\sqrt{t}\rfloor} \leq n^{\sqrt{t}}.$$

This shows that $|H| \leq n^{\sqrt{t}}$.

Finally, we prove the AKS Theorem (Theorem 3.1.1).

Proof. $|H| \geq \binom{t+l}{t-1}$

$\geq \binom{l+1+\lfloor\sqrt{t}\log n\rfloor}{\lfloor\sqrt{t}\log n\rfloor}$ since $t > \sqrt{t}\log n$

$\geq \binom{2\lfloor\sqrt{t}\log n\rfloor+1}{\lfloor\sqrt{t}\log n\rfloor}$ since $l > \lfloor\sqrt{t}\log n\rfloor$

$\geq 2^{\lfloor\sqrt{t}\log n\rfloor+1}$ since $\lfloor\sqrt{t}\log n\rfloor > \lfloor\log^2 n\rfloor \geq 1$

$\geq n^{\sqrt{t}}$

By lemma 3.2.4, $|H| \leq n^{\sqrt{t}}$ if n is not a power of p. Therefore, $n = p^k$ for some $k > 0$.

3.2 Proof of Time Complexity

Theorem 3.2.1. *The AKS Algorithm has a time complexity of* $O(\log^{21/2-\varepsilon} n)$.

Proof. Let us observe each specific step of the algorithm and estimate the function describing the bit operations.

Step 1 checks to see if n is a perfect power. This step has been proven to take $O(\log^{3-\varepsilon} n)$.

Step 2 finds the smallest r such that $\text{ord}_r(n) > \log^2 n$. Testing r number of values to see if $n^k \not\equiv 1 \pmod{r}$ for all $k \leq \log^2 n$ for a particular r will take at most $O(\log^{2+\varepsilon} n)$ multiplications modulo r. So, this step will take $O(r \log^{2+\varepsilon} n)$. Since there is an upper bound for r, namely $\log^5 n$, the total time complexity for this step is $O(\log^{7+\varepsilon} n)$.

Step 3 calculates the greatest common divisor between n and a for $1 < a \leq r$. This step will calculate the greatest common divisor r different times. Each greatest common divisor computation has a time complexity of $O(\log n)$. Therefore, this step has a time complexity of $O(r \log n)$, or $O(\log^6 n)$.

Step 4 checks to see if $n \leq r$. The time complexity of this is O(log n).

Step 5 checks the congruency

$$(X + a)^n \not\equiv X^n + a \pmod{X^r - 1}$$

for a from 1 to $\lfloor \sqrt{\phi(r)} \log n \rfloor$. For the equation, $\lfloor \sqrt{\phi(r)} \log n \rfloor$ different equations will be checked. Each equation requires O(log n) multiplications of degree r polynomials with coefficients size O(log n). So, each equation can be verified in $O(r \log^{2+\varepsilon} n)$ bit operations. This means the total run time of step 5 is $O(r\sqrt{\phi(r)} \log^{3+\varepsilon} n) = O(r^{3/2} \log^{3+\varepsilon} n)$ $= O(\log^{21/2+\varepsilon} n)$.

Since the last step dominates all the other time complexities, it is therefore the time complexity of the AKS algorithm as written in *Primes is in P*.

3.3 Programming The AKS algorithm

We have written an executable code to implement the AKS Primality Test following a series of pseudocodes described below. The Maple code for each step can be found in Appendix.

In step 1, the number n is checked to see if it is a perfect power of a prime.

Pseudocode. Perfect Power Test

Input: n (a positive integer to test for perfect power)

```
1.    Set $b := 2$.
2.    Set $l := 1$, and $u := n$.
3.    Set $a := \left\lfloor \frac{l+u}{2} \right\rfloor$.
4.    If $a = 1$ or $a = u$, set $b := b+1$, $l := 1$, $u := n$.
5.    If $2^b > n$, return *not a perfect power*.
6.    If $a^b = n$, return *perfect power*.
7.    If $a^b > n$, set $u := a$; otherwise, set $l := a$.
8.    Return to step 3.
```

Step 2 finds the smallest r such that $\text{ord}_r(n) > \log^2 n$. There are two parts in programming this step. The first is finding the order function, and the second is finding the specific value of r.

Pseudocode. Multiplicative Order

Input: n and r (such that $\gcd(n,r)=1$) output $\text{ord}_r(n)$

```
1.    $i = n \bmod r$.
2.    $w = 1$.
3.    If $i = 1$ then return $w$, break.
4.    Else $i = (i*n) \bmod r$  $w = w+1$.
5.    Return to step 3.
```

Pseudocode. The *r* value

Input: *n* (the value for which you want to calculate *r*)

```
1.    $r := 2$.
2.    If $\gcd(r,n)=1$ and $\mathrm{ord}(n,r) > \log^2 n$ then return $r$; break.
3.    $r := r + 1$.
4.    Return to step 2.
```

Chapter 4 Research and Results

In this chapter we will present further results related to the AKS Primality Test presented in Chapter 3.

We proved that the r value has an upper bound of $\log^5(n)$. This implies that $n \le \lceil \log^5 n \rceil$. Lemma 4.1 follows when we solve for n.

Lemma 4.1. *Step* 4 *may be omitted from the AKS algorithm for* $n >$ 5,690,034.

Proof. Assume the algorithm outputs *prime* in step 4.Since $r \le \lceil \log^5 n \rceil$ and step 4 checks $n \le r$, it follows that

$$n \le \lceil \log^5 n \rceil.$$

So, by the solving for n, one has

$$n \le 5{,}690{,}034 .$$

Consider the following graphs.

Figure 3

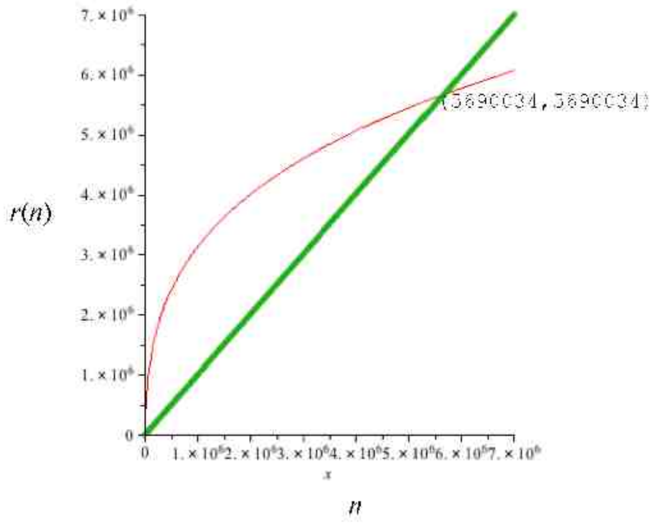


Figure 3 shows the line $f(n) = n$, and $f(n) = \log^5 n$. We see that the point of intersection is (5690034,5690034). This shows that 5,690,034 is theoretically the highest possible number for which $r(n)$ is larger than n. However, actual computer calculations showed the following results.

Theorem 4.2. *The largest positive integer n for which $r(n) > n$ is* 81.

Proof. After running the following algorithm, the output proves theorem 4.2.

Below we present the Maple code for checking all values $n < 5{,}690{,}034$ to see if $r(n) > n$.

```
For n from 2 to 5690034 do
        if n < r(n) then print(n, r(n));
        end if;
end do;
```

After executing the above Maple code, we received all possible outputs in the following table.

2, 3	10, 17	18, 29	26, 29	38, 47
3, 5	11, 13	19, 23	27, 29	40, 47
4, 11	12, 17	20, 23	28, 41	41, 47
5, 7	13, 19	21, 23	29, 41	45, 47
6, 11	14, 17	22, 25	30, 41	49, 67
7, 11	15, 19	23, 43	33, 43	51, 53
8, 11	16, 47	24, 31	35, 37	64, 83
9, 23	17, 23	25, 47	36, 59	81, 83

Lemma 4.3. *The smallest positive integer n for which $n > r(n)$ is* 31.

Proof. By inspection, the above output shows this is true.

Since the results above show us that the largest value for which $r(n) > n$ is 81 and not 5,690,034, the AKS algorithm can be rewritten as follows.

Improved AKS Algorithm.

Input: Integer $n > 81$.

```
1.  If ($n = a^b$ for $a$ in $\mathbb{N}$ and $b>1$), output composite
2.  Find the smallest $r$ such that $\mathrm{ord}_r(n) > \log^2 n$
3.  If $1 < \gcd(a,n) < n$ for some $a \le r$, output composite
4.  For $a = 1$ to $\lfloor\sqrt{\phi(r)}\log n\rfloor$ do
        If $((X + a)^n \not\equiv X^n + a \pmod{X^r-1, n})$, output composite
5.  Else, output prime
```

Notice that the original step 4 has been omitted from the algorithm by using Theorem 4.2. Another way to simplify the algorithm is by combining step 3 with step 1. The resulting algorithm is below.

Improved AKS Primality Test

Input: Integer $n > 81$

```
1. If $n$ is a perfect power, output composite.
2. $r := 1$.
3.if $1 < \gcd(r,n) <$ n output composite.
4.If $\mathrm{ord}_r(n) > \log^2 n$ then break,
else $r := r + 1$ and return to step 3.
5.For $a = 1$ to $\lfloor\sqrt{\phi(r)}\log n\rfloor$
```

If $(X+a)^n \not\equiv X^n + a \pmod{X^r-1, n}$, output *composite*.

6. Output *prime*.

Since Theorem 4.2 states that the lowest value is 81 and not 5,690,034, it makes one wonder if there is an upper bound on the r values that is less than $\log^5(n)$. The proof in Chapter 3 is based solely on the original paper *Primes is in P* [1] Since the paper was published the following Theorem has been proven.

Theorem 4.4. *For all sufficiently large n, there exists a prime number* $r \leq 8\lceil\log n\rceil^3(\log\log n)^3$ *such that r divides n or (r does not divide n and)* $\mathrm{ord}_r(n) > 4\lceil\log n\rceil^2$.

However the data collected suggests that there is a smaller upper bound. The following graph shows the values from 1 to 100,000 plotted against their r values.

Figure 4

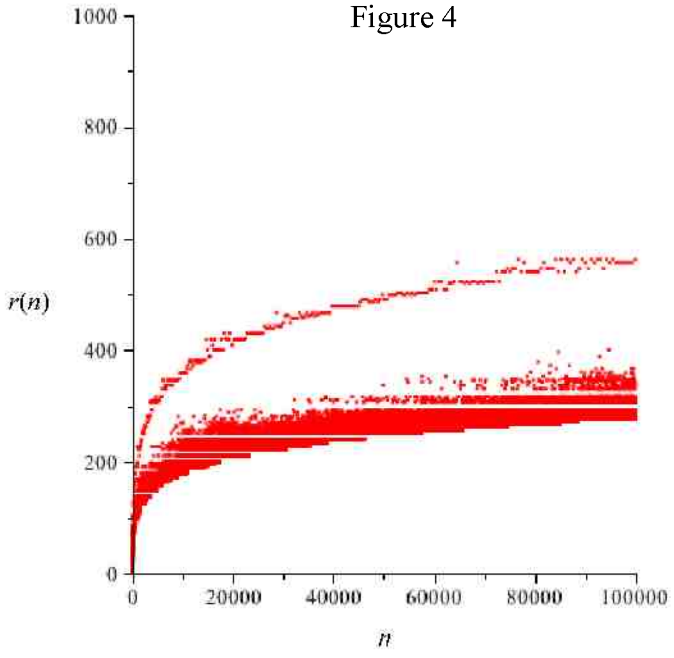


Figure 4 shows that there are two distinct parts in the area of interest. The upper part is made up of the perfect square n values and the lower part is made up of the non-square n values. Both parts have clear lower bounds that we are describing below.

Theorem 4.5. *A lower bound of the $r(n)$ is* $2 + \log^2(n)$.

Proof. The order of n modulo r is at most $r - 1$.

$$\mathrm{ord}_r(n) \leq r - 1$$

$$r \geq \mathrm{ord}_r(n)+1 > 1+\log^2(n)$$

So,

$$r \geq 2 + \log^2(n)_{\Box}$$

Theorem 4.6. *For any perfect square $n > 1$, $r(n) \geq 1 + 2\log^2 n$.*

Proof. Let $n = m^2$ and $r = r(m^2)$, then

$$r \geq 2\log^2 n = 2\log^2 m^2,$$

and

$$\mathrm{ord}_r(m^2) > \log^2 m^2.$$

Note that $\langle m^2 \rangle \subseteq \langle m \rangle \subseteq \mathbb{Z}_r^*$, and that the order of $\mathbb{Z}_r^*$ is $\varphi(r)$. Also the following equation shows the relationship between $\langle m^2 \rangle$ and $\langle m \rangle$.

$$|\langle m^2 \rangle| = \begin{cases} |\langle m \rangle| & \text{if } |\langle m \rangle| \text{ is odd} \\ \frac{1}{2}|\langle m \rangle| & \text{if } |\langle m \rangle| \text{ is even} \end{cases}$$

Look at the case when $\langle m^2 \rangle = \langle m \rangle = \mathbb{Z}_r^*$. If these are equal then the order of $\langle m \rangle$ is odd, because $|\langle m^2 \rangle| = |\langle m \rangle|$. This would imply that $\varphi(r)$ is also odd. However, $\varphi(r)$ is never odd for $r \geq 3$. By this contradiction we see that they are not all equal.

Therefore, $\langle m^2 \rangle$ is a proper subgroup of $\mathbb{Z}_r^*$, so

$$|\langle m^2\rangle \le \frac{1}{2}\varphi(r)$$

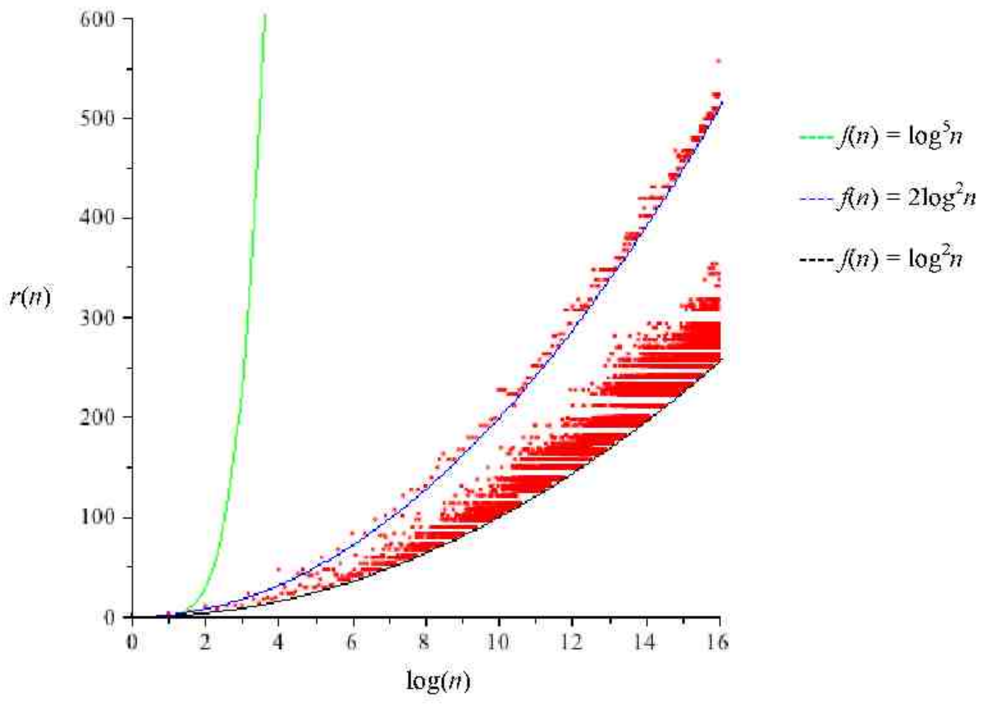
$$\log^2 m^2 < |\langle m^2\rangle| \le \frac{1}{2}\varphi(r) \le \frac{1}{2}(r-1)$$

$$\Rightarrow r > 1 + 2log^2 n \llcorner$$

Since the lower bound is logarithmic and the previously proved upper bound is also logarithmic, it makes sense to change our graph to have a logarithmic scale.

The following graph has the values of log n plotted with its respective r value, along with its upper bound of $\log^5 n$ and lower bounds $2 + \log^2 n$ and $1 + 2\log^2 n$.

Figure 5



Figures 6 and 7 show two graphs, one of the squares and one of the non-square values. It is interesting to notice that when checking primality by factoring naively, one only checks up to the floor of the square root of the number. This means the largest factor is equal to the square root, if the number is a perfect square.

Figure 6 Plot of Perfect Squares

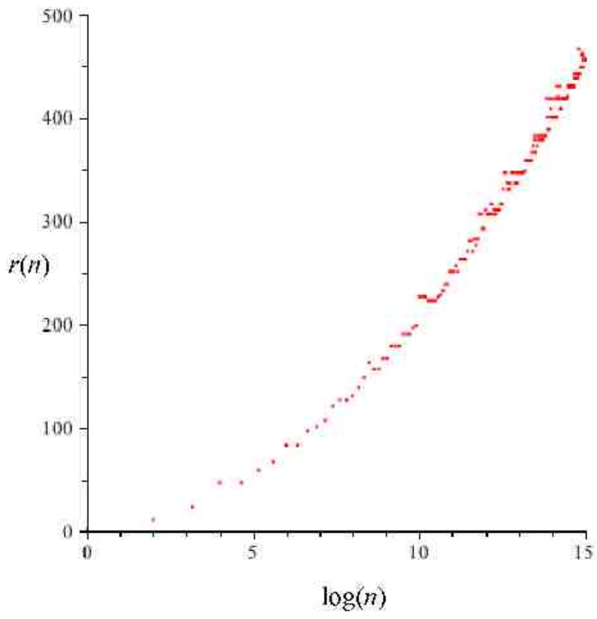
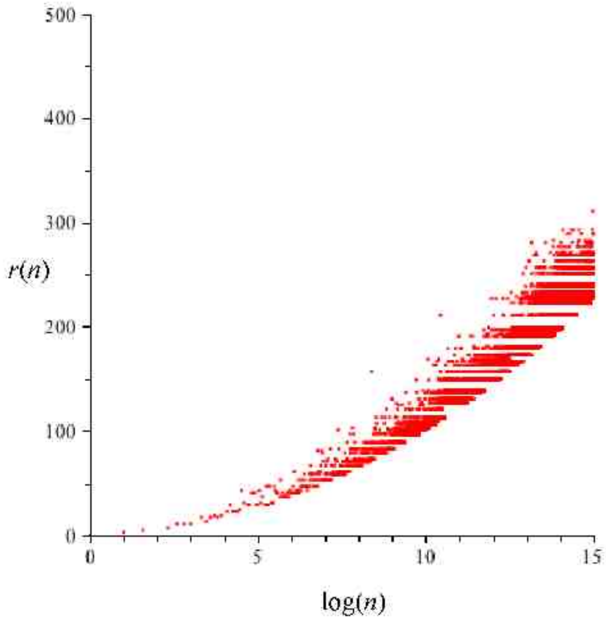


Figure 7 Plot of Non-Squares



Analyzing our results we can make the following two conjectures

Conjecture 4.7. *An upper bound for $r(n)$ is $3\log^2 n$ for all positive integer n.*

Conjecture 4.8. *For any integer n such that n is not a perfect square, $r(n) \leq 2\log^2 n$, except n = 2, 23, and 335.*

The above conjectures have been verified for $n \leq 7{,}703{,}162$.

If Conjecture 4.8 is indeed true, then the following corollary will also be true regarding the time complexity of the AKS algorithm.

Corollary 4.9. *The complexity of the AKS algorithm is* O($\log^{6-\varepsilon} n$) *in bit operations.*

Proof. Recall the previous proof of the time complexity of the AKS algorithm. Notice that only steps 2, 3, and 5 of the AKS algorithm are dependent on the r value. So to prove this corollary one only needs to look at those three steps.

Step 2 finds the smallest r such that $\mathrm{ord}_r(n) > \log^2 n$. Testing r number of values to see if $n^k \not\equiv 1 \pmod{r}$ for all $k \le \log^2 n$ for a particular r will take at most O($\log^{2+\varepsilon} n$) multiplications modulo r. So, this step will take O($r \log^{2+\varepsilon} n$). Since there is an upper bound for r, namely $2\log^2 n$, the total time complexity for this step is O($\log^{4+\varepsilon} n$).

Step 3 calculates the greatest common divisor between n and a for $1 < a \le r$. This step calculates the greatest common divisor r different times. Each greatest common divisor computation has a time complexity of O($\log n$). Therefore, this step has a time complexity of O($r \log n$), or O($\log^3 n$).

Step 5 checks the congruency

$$(X + a)^n \not\equiv X^n + a \pmod{X^r - 1}$$

for a from 1 to $\lfloor \sqrt{\phi(r)} \log n \rfloor$. For this equation we need to check $\lfloor \sqrt{\phi(r)} \log n \rfloor$ different equations. Each equation requires O($\log n$) multiplications of degree r polynomials with coefficients size O($\log n$). Hence, each equation can be verified in time O($r \log^{2+\varepsilon} n$) steps. This means the total run time of step 5 is O($r\sqrt{\phi(r)} \log^{3+\varepsilon} n$) = O($r^{3/2} \log^{3-\varepsilon} n$) = O($\log^{6-\varepsilon} n$).

Since step 5 is still the dominant time complexities, it is therefore the time complexity of the AKS algorithm□

Chapter 5: Future Work

In this paper, we proved that there are lower bounds on the r value of the AKS Primality Test for both the perfect squares and the non-squares. Namely, a lower bound of the r values is $\log^2 n$. If n is a square there is a lower bound of $2\log^2 n$. We also showed by graphical representations that there appears to be an upper bound of $2\log^2 n$ on the r values.

Future possible improvement of this work could be to program the pseudocode in a more efficient programming language. This would give us the opportunity to generate longer output sets and see if the conjectures still hold, possibly proving that both conjectures are true. It would also be interesting to generalize the results so they would apply to any algebraic group.

Bibliography

[1] M. Agrawal, N. Kayal, and N. Saxena, PRIMES is in P, Ann. of Math. **160**, 2002, 781-793.

[2] J. Elliott, *Lectures on Abstract Algebra*, unpublished class notes, 2010.

[3] M. Dietzfelbinger, *Primality Testing in Polynomial Time*, Springer-Verlag, Berlin, 2004.

[4] R. Crandall and C. Pomerance, *Prime Numbers: A computational Perspective*, Springer Science + Business Media, Inc, New York, 2005.

[5] Bressoud and S. Wagon, *A course in computational Number Theory*, Key Publishing, New York, 2000.

[6] Q. Cheng, "Primality Proving Via One Round In ECPP and One Iteration In AKS." Available from the World Wide Web: <http://www.cs.ou.edu/%7Eqcheng/paper/aksimp.pdf>

[7] R. Crandall and J. Papadopoulos, On the implementation of AKS-class primality tests University of Maryland College Park, March 18, 2003.

[8] R. Brent, Primality testing and integer factorization, Australian Academy of Science, 1991, 14-26.

[9] F. Bornemann, PRIMES is in P: A Breakthroough for "Everyman" Notices of the AMS, Volume 50, Number 5, 2003.

[10] Nicodemi, Olympia, Melissa A. Sutherland, and Gary W. Towsley. *An introduction to abstract algebra: with notes to the future teacher*. Upper Saddle River, NJ: Pearson Prentice Hall, 2007.

[11] Koblitz, Neal. *A course in number theory and cryptography*. New York: Springer-Verlag, 1987.

Appendix

Maple Code for Successive Squaring

```
multiplication := proc (n, r, a)
        u := n;
        s := mod(x + a, n);
        c := 1;
        while 1 ≤ u do
                if type(u, odd) then
                        c := mod(rem(expand(c*s), x^r-1, x), n);
                        u := u-1
                end if;
                s := mod(rem(expand(s*s), x^r-1, x), n);
                u := (1/2)*u;
        end do;
    return c;
    end proc:
```

Maple code for Perfect Power Test

```
perfpow := proc (n) b := 2; l := 1; u := n;
    while 2^b <= n do
        a := floor((1/2)*l+(1/2)*u);
        if a = l or a = u then b := b+1; l := 1; u := n; end if;
        if a^b = n then print(true); break; end if;
        if n < a^b then u := a; else l := a; end if
    end do;
    if a^b <> n then print(false) end if;
end proc;
```

Maple Code for Finding the order of *n* modulo *r*

```
ord := proc (x, y) i := mod(x, y);
    for w do if i = 1 then w; break;
        else i := mod(i*x, y);
        end if;
    end do;
end proc:
```

Maple Code for finding the r value

```
r := proc (n) for r from 2 do
          if gcd(r, n) = 1 then
                    if evalf(log[2](n)^2) < ord(n, r) then r; break:
                    end if;
          end if;
          end do ;
end proc;
```

AKS Primality Test

```
f := 0;
if perfpow(n) then f := 1; print(composite) end if;
if f = 0 then for r from 2 do if n > gcd(r, n) and gcd(r, n) > 1 then f := 1;
  print(composite)
          else if ord(n, r) > evalf(log[2](n)^2) then break; end if; end if; end do;
           end if;
if f = 0 then for a to 2*ceil(r^.5)*ceil(log[2](n)) do if multiplication(n, r, a) <>
x^(`mod`(n, r))+a then f := 1; print(composite) end if end do end if; if f = 0 then
print(prime) end if;
```

Lemma A.1. $h(x)$ has degree $\text{ord}_r(p)$.

Proof: Let $Q_r[x]$ be the rth cyclotomic polynomial in F_p.

Let $h(x)$ be an irreducible factor of $Q_r[x]$ and let k be the degree of $h(x)$. Show that $k = \text{ord}_r(p)$.

Since $h(x)$ is irreducible it follows that $F_p[x]/h(x)$ is a field with order p^k. Notice that a multiplicative subgroup of $F_p[x]/h(x)$ is cyclic with a generator $g(x)$. Now,

$$g(x)^p \equiv g(x^p)$$

$$g(x)^{p^d} \equiv g\left(x^{p^d}\right)$$

$$g(x)^{p^d} \equiv g(x)$$

$$g(x)^{p^d - 1} \equiv \mathbf{1}.$$

Thus the order of g(x) is $p^k - 1$ and since the order of any element divides the order of the group one has $(p^k - \mathbf{1})|(p^d - 1)$. Therefore, k must divide d. Since $h(x)$ divides (X^r – 1) and

$$X^r \equiv \mathbf{1} \in F_p[X]/h(x).$$

Thus, the order of X is r, and since r is prime and x is not congruent to 1, r divides (p^k – 1) or in other words $p^k \equiv 1 \bmod r$. Thus, $d|k$ so k must equal d. Therefore, each irreducible polynomial much be ord$_r$(p) □