2005-06-29

An Exposition of the Deterministic Polynomial-Time Primality Testing Algorithm of Agrawal-Kayal-Saxena

Robert Lawrence Anderson
Brigham Young University - Provo

AN EXPOSITION OF THE DETERMINISTIC POLYNOMIAL-TIME PRIMALITY TESTING ALGORITHM OF AGRAWAL-KAYAL-SAXENA

by

Robert Lawrence Anderson

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mathematics
Brigham Young University
August 2005

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Robert Lawrence Anderson

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	David A. Clark, Chair
Date	Jasbir S Chahal
Date	Darrin M Doud

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Robert Lawrence Anderson in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements, (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

_____ _____

Date David A. Clark
Chair, Graduate Committee

Accepted for the Department _____

Tyler J. Jarvis
Graduate Coordinator

Accepted for the College _____

G. Rex Bryce, Associate Dean
College of Physical and Mathematical Sciences

Abstract

I present a thorough examination of the unconditional deterministic polynomial-time algorithm for determining whether an input number is prime or composite proposed by Agrawal, Kayal and Saxena in their paper [1].

All proofs cited have been reworked with full details for the sake of completeness and readability.

Contents

Introduction

This work is intended to provide enough detail that an advanced undergraduate with background in Abstract Algebra and Number Theory should be able to comprehend the construction of this important result on primality without consulting other sources.

The references in the paper have been verified and all proofs have been checked and we have added references to standard literature in algebra, number theory, and computational complexity theory.

Computational complexity theory is part of the theory of computation dealing with the resources required during computation to solve a given problem. The most common resources are time (how many steps does it take to solve a problem) and space (how much memory does it take to solve a problem).

In this theory, the class **P** consists of all those decision problems that can be solved on a deterministic sequential machine in an amount of time that is polynomial in the size of the input; the class **NP** consists of all those decision problems whose positive solutions can be verified in polynomial time given the right information, or equivalently, whose solution can be found in polynomial time on a non-deterministic machine.

The class **RP** (Randomized Polynomial-Time) consists of all those decision problems solvable by an **NP** machine such that:

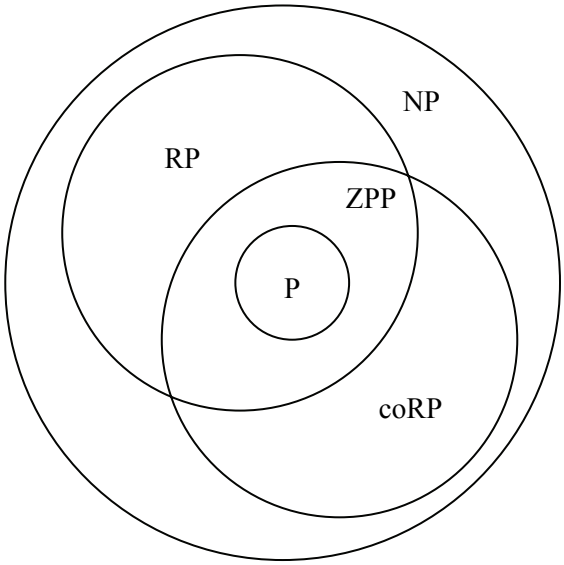
- If the answer is 'yes,' at least 1/2 of computation paths accept.
- If the answer is 'no,' all computation paths reject.

The class **coRP** is the complement of **RP** and consists of all those decision problems solvable by an **NP** machine such that:

- If the answer is 'yes,' all computation paths accept.

- If the answer is 'no,' at least 1/2 of computation paths reject.

The class **ZPP** (Zero-Error Probabilistic Polynomial-Time) is defined to be the intersection of **RP** and **coRP**. It is the class of problems solvable by randomized algorithms that always return the correct answer, and whose expected running time (on any input) is polynomial. The relationship of these classes is given by the following diagram:



Arguably, the biggest open question in theoretical computer science concerns the relationship between those two classes:

Is **P** equal to **NP**?

A $1,000,000 USD prize has been offered for a correct solution. Garey[2] is an excellent reference on the theory of **NP**-completeness for those interested in pursuing this prize.

The historical classification of the problem of testing whether an input integer is prime roughly follows this timeline:

Year	Class	By Whom
pre–1975	***coNP***	Trivial. A single factor of n is a short certificate that can be used to verify that n is not prime. This can be done in linear time using the Euclidean algorithm.
1975	***NP***	Pratt [3]
1975	$\boldsymbol{NP \cap coNP}$	
1977	***RP***	Solovay-Strassen [4]
1987	***coRP***	Adleman-Huang [5]
1987	***ZPP***	$= \boldsymbol{RP \cap coRP}$
2004	***P***	Agrawal-Kayal-Saxena [1]

The classes ***RP***, ***coRP***, and ***ZPP*** were defined by Gill [6] in 1977.

Given a problem such as determining whether or not a number is prime, the question arises, "What is the best algorithm for doing this?". The difficulty in such a question is in deciding what is meant by "best".

One decision that often comes into play is as to the necessity of a deterministic algorithm. For instance, if you are testing to see if a certain number is prime for the purpose of using it to construct a cryptographic system, the fastest known algorithms are probabilistic ones (e.g., the Miller-Rabin test [7]). Although they provide no guarantee of theoretical correctness, the error bound can be made arbitrarily small so that for all practical considerations they are reliable algorithms. However, this sort of primality testing will not be of class of ***P*** no matter how fast it is.

The AKS algorithm is the first known deterministic algorithm for primality testing that can be proved to run unconditionally in polynomial-time on all inputs.

An Overview of Computational Complexity Theory

A ***deterministic one-tape Turing machine*** (DTM) is computational model consisting of a finite state control, a read-write head, and a tape made up of a two-way infinite sequence of tape squares, labeled $\ldots\ -2,\ -1,\ 0,\ 1,\ 2,\ \ldots$.

Let Σ be a finite set of symbols. Let Σ^* denote the set consisting of the empty string ε and all finite concatenations of elements of Σ. Then L is a ***language*** over Σ^* if $L \subset \Sigma^*$. (e.g., the set of binary representations of integers is a language over $\{0,1\}$).

A ***program*** for a DTM consists of

- a finite set Γ of tape symbols, including a subset $\Sigma \subset \Gamma$ of input symbols and a distinguished blank symbol $b \in \Gamma - \Sigma$;
- a finite set Q of states, including a distinguished start state q_0 and two distinguished halt-states q_Y and q_N;
- a transition function $\delta : \left(Q - \{ q_Y , q_N \}\right) \times \Gamma \ \rightarrow\ Q \times \Gamma \times \{ -1, +1 \}$.

A DTM program M with input alphabet Σ ***accepts*** $x \in \Sigma^*$ if and only if M halts in state q_Y when applied to input n.

The language L_M ***recognized*** by the program M is given by $L_M = \{n \in \Sigma^* : M \text{ accepts } n\}$.

For a DTM program which halts on all inputs $n \in \Sigma^*$, its ***time complexity function*** $T_M : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ is given by:

$$T_M(x) = \max\left\{ m : \begin{array}{l} \text{there is an } n \in \Sigma^* \text{ with } x = \lfloor \log_2 n \rfloor + 1, \text{ such that} \\ \text{the computation of } M \text{ on input } n \text{ takes time } m \end{array} \right\}.$$

Any function $f(x)$ is considered to be $\boldsymbol{O(g(x))}$ (pronounced "big-oh of *g*") if given some function $g(x)$ there exists a constant *c* such that $|f(x)| \leq c \cdot |g(x)|$ for all values of $x \geq 0$, where *x* is defined to be the input length of *n*.

A DTM program which halts on all inputs $n \in \Sigma^*$ is called a ***polynomial time*** DTM program if there exists a polynomial $p(x)$ such that, for all $x \in \mathbb{Z}^+$, $T_M(x) \leq p(x)$. Alternately, a DTM program which halts on all inputs $x \in \Sigma^*$, is called a ***polynomial time*** DTM program if its time complexity function is $O(p(x))$. Again, *x* is defined to be the input length of *n*.

The class ***P*** is the class of languages defined as follows:

$\boldsymbol{P} = \{L:$ there is a polynomial time DTM program M for which $L = L_M\}$

Within this framework, the purpose of the paper is to show that with $M =$ "the AKS algorithm" and $L_M =$ "binary representation of positive integers", then *M* is a polynomial time DTM program with input set $\sum = \mathbb{N} - \{1\}$ and halt states $q_Y =$ "PRIME" and $q_N =$ "COMPOSITE", and time complexity function $T_M(x)$ which is $O(x^{25/2})$.

Notation

We will use $\log n$ to denote the base 2 logarithm instead of $\log_2 n$ or $\lg n$. Base 10 logarithms and natural logarithms will be denoted as $\log_{10} n$ and $\ln n$ respectively.

The notation $\text{ord}_r(a)$ represents the *order of a modulo r*, which is the smallest positive integer *k*, such that $a^k \equiv 1 \pmod{r}$.

The notation $\phi(r)$ will be used to represent *Euler's totient function*, which is defined as the number of positive integers less than or equal to *r* that are relatively prime to *r*.

The notation $f(x) \equiv g(x) \pmod{h(x), p}$ is used throughout to mean that $f(x) = g(x)$ in the ring $\mathbb{Z}_p[x] \big/ (h(x))$. In some cases, *p* will be prime and $h(x)$ will have degree *d* and be irreducible in $\mathbb{Z}_p[x]$, so that $\mathbb{Z}_p[x] \big/ (h(x))$ will be a finite field of order p^d.

Time complexity functions will be written in "big-O" notation. A function $f(x)$ is considered to be $O(g(x))$ (pronounced "big-oh of *g*") if given some function $g(x)$ there exists a constant *c* such that $|f(x)| \le c \cdot |g(x)|$ for all values of $x \ge 0$, where *x* is defined to be the binary input length of *n*.

The function $M(n)$ will be used to represent the time complexity function for multiplication. The fastest known algorithm for multiplying two *n* digit numbers is due to Schönhage and Strassen [17] and has time complexity $O(n \log n \log\log n)$. Ordinary multiplication (the kind you learn in school) has time complexity $O(n^2)$.

Part 1 – Conceptual Foundation

The algorithm is based upon a generalization of Fermat's Little theorem given in the Following lemma:

Lemma 1

Let $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$, and $(a,n) = 1$. Then *n* is prime if and only if

$$(x+a)^n \equiv x^n + a \pmod{n}$$

This says that if after we reduce all of the coefficients of the binomial expansion of $(x+a)^n$ modulo *n* we are left with $x^n + a$, then *n* must be prime, otherwise *n* is not prime.

Proof

We want $n \in \mathbb{N}$, $n \geq 2$ since we are only interested in showing whether or not *n* is a prime. If $n < 2$, then *n* will not be prime by definition. The $\gcd(a,n) = 1$ condition is necessary to use Fermat's Little Theorem which this generalizes.

Instead of proving "a implies b" and then "b implies a", we prove the contrapositive of the second direction (i.e., instead of proving "b implies a" we prove "not a implies not b", which is logically equivalent).

Recall that the binomial expansion of $(x+a)^n$ is given by

$$\sum_{i=0}^{n} \binom{n}{i} x^i a^{n-i} \quad \text{where} \quad \binom{n}{i} = \frac{n!}{i!(n-i)!}.$$

First we will prove that if *n* is prime then $(x+a)^n \equiv x^n + a \mod (n)$.

Suppose n is prime. If $0 < i < n$, then $\gcd(n, i!) = 1$ and $\gcd(n, (n-i)!) = 1$ since both i and $n-i$ are less than n and n is prime. Thus $\frac{n!}{i!(n-i)!} = n \cdot c$, where $c = \frac{(n-1)!}{i!(n-i)!}$ and $c \in \mathbb{Z}$, since $\frac{n!}{i!(n-i)!} \in \mathbb{Z}$ and $i!$ and $(n-i)!$ contain no divisors of n. This gives us that $\binom{n}{i} = \frac{n!}{i!(n-i)!} = n \cdot c \equiv 0 \pmod{n}$ for $0 < i < n$, and hence $\binom{n}{i} x^i a^{n-i} \equiv 0 \pmod{n}$ for $0 < i < n$.

Thus we see that when n is prime, the middle terms of the binomial expansion are all equivalent to zero mod n. The only thing left to show is that $a^n \equiv a \pmod{n}$. By Fermat's Little Theorem, we know that if n is prime, then $a^{(n-1)} \equiv 1 \pmod{n}$ whenever $\gcd(a, n) = 1$. Multiplying both sides by a, we get $a^n \equiv a \pmod{n}$ as desired.

Now we will prove that if $(x+a)^n \equiv x^n + a \pmod{n}$ then n is prime by proving its contrapositive, "if n is not prime, then $(x+a)^n \not\equiv x^n + a \pmod{n}$".

Suppose n is not prime (i.e., composite since $n > 1$). Let q be a prime factor of n and let k be the largest integer such that q^k divides n. We can then write $n = q^k m$ where q does not divide m. Then

$$\binom{n}{q} = \frac{n!}{q!(n-q)!} = \frac{q^k m(n-1)...(n-q+1)}{q(q-1)!} = \frac{q^{k-1} m(n-1)...(n-q+1)}{(q-1)!}.$$

Notice that q does not divide the product $m(n-1)...(n-q+1)$ on the far right, since q does not divide m and the first multiple of q less than n is $n-q$. Thus q^k does not divide $\binom{n}{q}$ since we cannot factor out another q on top to make q^k. Also note that $\gcd(q^k, a^{n-q}) = 1$

since $\gcd(a,n)=1$ and q^k divides n. So the coefficient of x^q in the binomial expansion of $(x-a)^n$, given by $\binom{n}{q}a^{n-q}$, cannot be equivalent to zero mod n since it is not even divisible by q^k, let alone by n.

Thus $\left((x+a)^n-\left(x^n+a\right)\right)\not\equiv 0 \pmod{n}$, since even if all other middle coefficients go to zero, we still have

$$\left((x+a)^n-\left(x^n+a\right)\right)\equiv\left(x^n+\binom{n}{q}a^{n-q}x^q+a^n-x^n-a\right)\pmod{n}\equiv\binom{n}{q}a^{n-q}x^q\pmod{n}.$$

Therefore, $(x+a)^n\not\equiv x^n+a \pmod{n}$. □

In order to reduce the number of coefficients that need to be examined, the authors decided to consider Lemma 1 after reducing modulo the ideal generated by x^r-1 for a sufficiently small r. This way at most $r-1$ coefficients need to be examined.

There is one small problem. Lemma 1 is not generally true modulo (x^r-1). It turns out, however, that if r is chosen to be the smallest positive integer such that the order of n in $\left(\mathbb{Z}/r\mathbb{Z}\right)^{\times}$ is greater than $\log^2 n$, then n can be shown to be a prime power if $(x+a)^n\equiv x^n+a \pmod{x^r-1,\ n}$ for all integer values of a ranging from 1 to $\left\lfloor 2\sqrt{\phi(r)}\log n\right\rfloor$. This will be shown in Lemma 7.

Next we will review some definitions and theorems from finite field theory that will be necessary to show the correctness of the algorithm.

Definition 0.1

Let *n* be a positive integer. The splitting field of $x^n - 1$ over a field *K* is called the n^{th} *cyclotomic field over K* and is denoted by $K^{(n)}$. The roots of $x^n - 1$ in $K^{(n)}$ are called the n^{th} *roots of unity over K* and the set of all these roots is denoted by $E^{(n)}$.

Definition 0.2

Let *K* be a field of characteristic *p* and *n* a positive integer not divisible by *p*. Then a generator of the cyclic group $E^{(n)}$ is called a *primitive* n^{th} *root of unity over K*.

Definition 0.3

Let *K* be a field of characteristic *p*, *n* a positive integer not divisible by *p*, and ζ a primitive n^{th} root of unity over K. Then the polynomial

$$\mathcal{Q}_n(x) = \sum_{\substack{s=1 \\ \gcd(s,n)=1}}^{n} \left(x - \zeta^s\right)$$

is called the n^{th} *cyclotomic polynomial over K*.

Theorem 0.4

Let $K = \mathbb{F}_q$ with $\gcd(q,n) = 1$. Then $\mathcal{Q}_n$ factors into $\phi(n)/d$ distinct monic irreducible polynomials in $K[x]$ of the same degree *d*, $K^{(n)}$ is the splitting field of any such irreducible factor over *K*, and $\left[K^{(n)} : K\right] = d$, where d is the least positive integer such that $q^d \equiv 1 \pmod{n}$.

Proof

Let η be a primitive n^{th} root of unity over *K*. Then $\eta \in \mathbb{F}_{q^k}$ if and only if $\eta^{q^k} = \eta$ and the later identity is equivalent to $q^k \equiv 1 \pmod{n}$. The smallest positive integer for which this holds is $k = d$, and so η is in $\mathbb{F}_{q^d}$ but in no proper subfield thereof. Thus the minimal polynomial of η over *K* has degree *d*, and since η is an arbitrary root of Q_n, the desired result follows. □

Corollary 0.5

$\text{ord}_r(n)$ divides $\phi(r)$.

Proof

In Theorem 0.4, *d* is by definition $\text{ord}_n(q)$, which divides $\phi(n)$. □

Part 2 – The AKS Algorithm

The Agrawal-Kayal-Saxena (AKS) algorithm consists of six steps:

1. Test *n* to see if it is a perfect power.

Output COMPOSITE if it is.

This computationally inexpensive step is necessary to allow us to generalize Lemma 1 to the quotient ring $\mathbb{Z}_n[x] \Big/ \left(x^r - 1\right)$. Bernstein [8] has devised an algorithm to perform such a test in time $O\left(\left(\log n\right)^{\left(1+\frac{\sqrt{\log\log n \log\log\log n}}{\log n}\right)}\right)$, which is essentially linear on the binary length of *n*.

The algorithm is essentially a binary search to find a root z of $z^k y - 1$. The midpoint of an interval *R* containing a root is evaluated. Then depending on the sign of $z^k y - 1$, *R* is replaced with the left or right half *R* and the process is repeated. The *middle half* of the interval is taken if the approximation is too close to zero to be sure of the sign. To save time, the calculation of $z^k y - 1$ at the midpoint is approximated. This method effectively determines the binary representation of z one bit at a time. The algorithm outputs (x, p) where $n = x^p$ if *n* is a perfect power, and $(n,1)$ otherwise.

The proofs of the correctness and time complexity of this algorithm are beyond the scope of this paper and will not be given here. The interested reader may consult Bernstein's paper [8] for a more complete treatment .

The algorithm is as follows:

1. Set $f = \lfloor \log 2n \rfloor$

2. Compute $y \leftarrow \text{nroot}_{3+\lceil f/2 \rceil}(n,1)$. The algorithm for computing $\text{nroot}_j(y,k)$ immediately follows this main algorithm.

3. For each prime number $p < f$:

    a. Set $b = 3 + \lceil f / p \rceil$.

    b. Compute $r \leftarrow \text{nroot}_b(y, p)$.

    c. Find an integer x with $|r - x| \leq 5/8$.

    d. If $x = 0$ or $|r - x| \geq 1/4$, start over with the next prime.

    e. Compute the sign of $n - x^p$:

        i. Set $b \leftarrow 1$.

        ii. Compute $r \leftarrow pow_{b+\lceil \lg 8p \rceil}(x, p)$. The algorithm for computing $\text{pow}_b(r,k)$ is given on the next page.

        iii. If $n < r$, output -1 and stop.

        iv. If $r(1 + 2^{-b}) \leq n$, output 1 and stop.

        v. If $b \geq f$, output 0 and stop.

        vi. Set $b \leftarrow \min\{2b, f\}$. Go back to step ii.

    f. If $n = x^p$, output (x, p) and stop.

    g. Start over with the next prime.

4. Output $(n,1)$.

To compute $\text{nroot}_b(y,k)$ (a floating point number):

1. Find the exponent g satisfying $2^{g-1} < y \le 2^g$.
2. Set $a = \lfloor -g/k \rfloor$, so that $2^a \le y^{-1/k} < 2^{n+1}$.
3. Set $B = \lceil \log(66(2k+1)) \rceil$.
4. Set $z \leftarrow 2^a + 2^{a-1}$, $j \leftarrow 1$.
5. Now $\text{nroot}_j(y,k) = z$. If $j = b$, stop.
6. Compute $r \leftarrow \text{trunc}_B(\text{pow}_B(z,k)\,\text{trunc}_B y)$, where the function $\text{trunc}_b\, r$ is the truncation of r to b bits.
7. If $r \le 993/1024$, set $z \leftarrow z + 2^{a-j-1}$.
8. If $r > 1$, set $z \leftarrow z - 2^{a-j-1}$.
9. Set $j \leftarrow j+1$. Go back to step 5.

To compute $\text{pow}_b(r,k)$ (the b-bit approximate k^{th} power of r):

1. If $k = 1$, output $\text{trunc}_b\, r$ and stop.
2. If k is even, compute $\text{pow}_b(r, k/2)$, output $\text{trunc}_b(\text{pow}_b(r,k/2)^2)$ and stop.
3. Compute $\text{pow}_b(r,k-1)$ and output $\text{trunc}_b(\text{pow}_b(r,k-1)\,\text{trunc}_b\, r)$.

Examples:

$$\text{pow}_b(r,7) = \text{trunc}_b\left(\text{trunc}_b\left(\text{trunc}_b\left(\text{trunc}_b(r)^2\right)\text{trunc}_b(r)\right)^2 \text{trunc}_b(r)\right).$$

$$\mathrm{pow}_b(r,8) = \mathrm{trunc}_b\left(\mathrm{trunc}_b\left(\mathrm{trunc}_b\left(\mathrm{trunc}_b(r)^2\right)^2\right)^2\right).$$

2. Find the smallest *r* such that $\mathrm{ord}_r(n) > \log^2 n$**.**

Lemma 2 guarantees the existence of such an $r \le \lceil \log^5 n \rceil$. Since each *r* can be tested using at most $O(\log^2 n)$ multiplications, this step has time complexity $O(r\log^2 n)M(\log n) = O(\log^7 n)M(\log n)$, where $M(x)$ is the time complexity function for multiplication. So using fast Fourier multiplication, we get a time complexity of $O(\log^8 n \log\log n \log\log\log n)$.

Improving this bound for *r* is the key to improving the expected run time of the algorithm. It is believed that *r* is bounded by $O(\log^2 n)$ – a far better result than the bound of $O(\log^5 n)$ used in this proof. Agrawal [1] claims that this result has actually been proven by Hendrik Lenstra, Jr. and Carl Pomerance in a private communication. I have not been able to verify this since this source is not available to me.

3. Compute $\gcd(n,a)$ **for each** $2 \le a \le r$**.**

Output COMPOSITE if $1 < \gcd(n,a) < n$ **for some** $a \le r$**.**

This is just some basic trial division. Computing the GCD of *r* numbers takes $O(r\log n)$, so using the bound in step 2, this step can be computed in time $O(\log^6 n)$.

4. Output PRIME if $n \le r$.

This is a result of the trial division in step 3. This step takes no time of any consequence. Note that since $r \le \lceil \log^5 n \rceil$, this step is only relevant when $n \le 5,690,034$. In practice, one would generally look up such a small number in a table of primes. Furthermore, reducing the bound on *r* decreases the likelihood of this step occurring. For instance, with $r \le \lceil \log^3 n \rceil$, this step is only relevant when $n \le 982$. There are only 165 primes less than 982, with 983 being the 166th prime.

5. Compute $(x+a)^n \bmod (x^r - 1, n)$ for each $1 \le a \le \left\lfloor \sqrt{\phi(r)} \log(n) \right\rfloor$.

Output COMPOSITE if $(x+a)^n \not\equiv x^n + a \pmod{x^r - 1, n}$ for some *a*.

Lemma 1 guarantees that if the test fails for any *a*, then *n* is composite. On the other hand, if step 5 is satisfied for each *a*, a group can be defined with a lower bound on its order. Lemma 6 will show that *n* must be some power of a prime *p* by showing that on the contrary that the order of this same group would be too small. Step 1 of the algorithm will guarantee that that power is 1, thus making *n* a prime.

In this step there are $\left\lfloor \sqrt{\phi(r)} \log n \right\rfloor$ equations which require $O(\log n)$ multiplications of degree *r* polynomials with coefficients of size $O(\log n)$ to verify. Each equation then takes time $O(r \log^2 n) M(\log n)$ to verify. This gives a time complexity of $O(r\sqrt{\phi(r)} \log^3 n) M(\log n) = O(r^{3/2} \log^3 n) M(\log n) = O(\log^{2½} n) M(\log n)$. This step then represents the time complexity of the algorithm, since it dominates all others. Thus using

even classical multiplication, we get $M(\log n) = O(\log^2 n)$, so that the overall time complexity of the algorithm is $O(\log^{25/2} n)$, which is polynomial on the binary length of n. This allows the categorization of this algorithm in ***P***, the collection of deterministic polynomial-time algorithms after the algorithm is proved to be correct in Theorem 1. Fast Fourier multiplication, gives a time complexity of $O(\log^{23/2} n \log\log n \log\log\log n)$.

6. Output PRIME

We will show in Theorem 1 that the algorithm outputs prime if and only if n is prime.

Part 3 – The Correctness of the AKS Algorithm

The following lemma validates step 2 by setting an upper bound for finding a suitable *r*. It is the principal theorem used to show that the AKS algorithm is polynomial-time.

Lemma 2

There exists an $r \le \max\left\{3,\ \left\lceil \log^5 n \right\rceil\right\}$ such that $\text{ord}_r(n) > \log^2 n$.

The proof uses a Chebyshev type bound on the lcm of the first *n* numbers by Nair [9]. This bound is given with proof in Theorem A2 in the Appendix. The authors opted to use this result instead of the sieve theory result of Fouvry [10] in order to keep the proof as elementary as possible.

Proof

For $n = 2, r = 3$ satisfies the lemma, so we may let $n > 2$. Let $r_1, r_2, \ldots, r_t$ be all integers such that either $\text{ord}_{r_i}(n) \le \log^2 n$ or r_i divides n. The first condition implies that for each r_i there exists $k_i \le \log^2 n$ such that $n^{k_i} \equiv 1 \mod (r_i)$, or in other words, $n^{k_i} - 1 = r_i q$ for some integer *q*, so that r_i divides $\left(n^{k_i} - 1\right)$. Each of the r_i must divide the product $n \cdot \prod_{i=1}^{\lfloor \log^2 n \rfloor} \left(n^i - 1\right)$, since the fact that each k_i is less than or equal to $\log^2 n$ ensures that $n^{k_i} - 1$ is one of the terms of this product. This gives us that

$$\text{lcm}\left(r_1, r_2, \ldots, r_t\right) \le n \cdot \prod_{i=1}^{\lfloor \log^2 n \rfloor} \left(n^i - 1\right).$$

We also know that

$$n \cdot \prod_{i=1}^{\lfloor \log^2 n \rfloor} \left(n^i - 1\right) < n^{\log^4 n}.$$

To prove this, first notice that for $k > 0$ and $x > 1$, $x^{\left(\sum_{i=1}^{k} i\right)+1} > \prod_{i=1}^{k} \left(x^i - 1\right)$. And since $n > 2$, we have

$$n \cdot \prod_{i=1}^{\lfloor \log^2 n \rfloor} \left(n^i - 1\right) < n^{\left(\sum_{i=1}^{\lfloor \log^2 n \rfloor} i\right)+1} = n^{\frac{\left(\lfloor \log^2 n \rfloor\right)\left(\lfloor \log^2 n \rfloor + 1\right)}{2}+1} < n^{\log^4 n}.$$

Thus since $n = 2^{\log n}$, we have that

$$\operatorname{lcm}\left(r_1, r_2, \ldots, r_t\right) \leq n \cdot \prod_{i=1}^{\lfloor \log^2 n \rfloor} \left(n^i - 1\right) < 2^{\log^5 n}.$$

Since $n > 2$, we have $\lceil \log^5 n \rceil > 10$ and we may apply Theorem A2 to get

$$\operatorname{lcm}\left(1, 2, \ldots, \lceil \log^5 n \rceil\right) \geq 2^{\lceil \log^5 n \rceil}.$$

Therefore there must exist an $s < \lceil \log^5 n \rceil$ such that $s \notin \{r_1, r_2, \ldots, r_t\}$. If $(s,n) = 1$ then $\operatorname{ord}_s(n) > \log^2 n$ and we are done. If $(s,n) > 1$, then since *s* does not divide *n* and $(s,n) \in \{r_1, r_2, \ldots, r_t\}$, $r = \dfrac{s}{(s,n)} \notin \{r_1, r_2, \ldots, r_t\}$ and so $\operatorname{ord}_r(n) > \log^2 n$. □

For the remainder of this paper we will assume that the algorithm returns PRIME and show that *n* must be prime. This result will be restated later as Lemma 7.

Let *p* be prime and suppose that $p \mid n$. In order to pass through step 1 without getting an output of COMPOSITE, we must have that $n \neq a^b$ for any positive integers *a* and *b*.

In step 2, no output is given.

In order to pass through step 3 without getting an output of COMPOSITE, we need $\gcd(a,n)=1$ or n for all $a \le r$. Since $p \mid n$, this means that p must be greater than r, otherwise step 3 would have output COMPOSITE. Since p is prime, we have $\gcd(r,p)=1$.

If $n \le r$, then step 3 amounts to trial division and is a complete test on the primality of n. So we can pass over step 4 by assuming that $n > r$ without any loss of generality. Recall that since $r \le \lceil \log^5 n \rceil$, this step is only relevant when $n \le 5,690,034$.

Next we assume that n and p are fixed and set $\ell = \left\lfloor \sqrt{\phi(r)} \log n \right\rfloor$. Step 5 of the algorithm verifies ℓ equations. Since the algorithm does not output COMPOSITE in this step, we have

$$(x+a)^n \equiv x^n + a \pmod{x^r - 1,\ n} \text{ for every } a,\ 0 \le a \le \ell.$$

This implies that

$$(x+a)^n \equiv x^n + a \pmod{x^r - 1,\ p} \text{ for every } a,\ 0 \le a \le \ell. \qquad (1)$$

By Lemma 1, we have

$$(x+a)^p \equiv x^p + a \pmod{x^r - 1,\ p} \text{ for every } a,\ 0 \le a \le \ell. \qquad (2)$$

Since p divides n, from (2) we get

$$(x+a)^n \equiv \left((x+a)^p\right)^{n/p} \equiv \left(x^p + a\right)^{n/p} \pmod{x^r - 1,\ p} \text{ for every } a,\ 0 \le a \le \ell, \quad (3)$$

and from (1) we get

$$(x+a)^n \equiv x^n + a = \left(x^p\right)^{n/p} + a \pmod{x^r - 1,\ p} \text{ for every } a,\ 0 \le a \le \ell. \quad (4)$$

(3) and (4) imply that

$$\left(x^p + a\right)^{n/p} \equiv \left(x^p\right)^{n/p} + a \pmod{x^r - 1,\ p} \text{ for every } a,\ 0 \le a \le \ell.$$

Replacing x^p with x, we get

$$\left(x + a\right)^{n/p} \equiv x^{n/p} + a \pmod{x^r - 1,\ p} \text{ for every } a,\ 0 \le a \le \ell.$$

Thus both n and $\frac{n}{p}$ behave like prime p in the above equation. At this point we give the following definition:

Definition 1

For polynomial $f(x)$ and number $m \in \mathbb{N}$, we say that m is ***introspective*** for $f(x)$ if $\left[f(x)\right]^m \equiv f\left(x^m\right) \pmod{x^r - 1, p}$. □

The next two lemmas show that the set of introspective numbers for a given polynomial is closed under multiplication, and that the set of polynomials for which m is introspective is also closed under multiplication.

Lemma 3

If m and m' are introspective for $f(x)$, then so is $m \cdot m'$.

Proof

By definition, $\left[f(x)\right]^m \equiv f\left(x^m\right) \pmod{x^r - 1, p}$. Thus

$$\left[f(x)\right]^{mm'} \equiv \left[f\left(x^m\right)\right]^{m'} \pmod{x^r - 1, p}. \qquad (1)$$

Since m' is also introspective, replacing x with x^m gives:

$$\left[f\left(x^m\right)\right]^{m'} \equiv f\left(x^{mm'}\right) \pmod{x^{mr}-1, p}.$$

Notice that there exists an injective map

$$\varphi: \ {\mathbb{Z}_p[x]}\Big/{\left(x^r-1\right)} \to {\mathbb{Z}_p[x]}\Big/{\left(x^{mr}-1\right)}.$$

Or in other words, since

$$x^{mr}-1 \ = \ \left(x^r-1\right)\left(\sum_{i=1}^{m} x^{(m-i)r}\right),$$

we can see that x^r-1 divides $x^{mr}-1$, and so the ideal generated by x^r-1 contains the ideal generated by $x^{mr}-1$. Thus any congruence modulo $\left(x^{mr}-1\right)$ must also hold modulo the larger ideal $\left(x^r-1\right)$ in precisely the same way that any number divisible by 6 must also be divisible by 2 or 3. Thus we see that

$$\left[f\left(x^m\right)\right]^{m'} \equiv f\left(x^{mm'}\right) \pmod{x^r-1, p}, \qquad (2)$$

and from (1) and (2) we get

$$\left[f\left(x\right)\right]^{mm'} \equiv f\left(x^{mm'}\right) \pmod{x^r-1, p}. \qquad \square$$

Lemma 4

If m is introspective for $f\left(x\right)$ and $g\left(x\right)$, then m is introspective for $f\left(x\right)\cdot g\left(x\right)$.

Proof

First we have that $\left[f(x)g(x) \right]^m = \left[f(x) \right]^m \left[g(x) \right]^m$. And since m is introspective for both $f(x)$ and $g(x)$, we get $\left[f(x)g(x) \right]^m = f\left(x^m\right) g\left(x^m\right) \left(\text{mod } x^r - 1, p\right)$. □

We next define the following sets:

$$I = \left\{ n^i p^j \mid i, j \geq 0 \right\}$$

$$P = \left\{ \prod_{a=1}^{\ell} (x+a)^{e_a} \mid e_a \geq 0 \right\}$$

Notice that since Step 5 guarantees that $(x+a)^n \equiv x^n + a \pmod{x^r - 1,\ n}$ for $1 \leq a \leq \ell$, it follows immediately from Lemma 4 and Lemma 5 that every $m \in I$ is introspective for every $f(x) \in P$.

Proposition 1

$G = \left\{ i \bmod r \mid i \in I \right\}$ is a multiplicative subgroup of $\left(\mathbb{Z}/r\mathbb{Z} \right)^{\times}$.

Proof

Since G is a finite set, it is sufficient to show that G is a multiplicatively closed subset of the multiplicative group $\left(\mathbb{Z}/r\mathbb{Z} \right)^{\times}$. We can see that $G \subset \left(\mathbb{Z}/r\mathbb{Z} \right)^{\times}$ since $n, p \in \mathbb{Z}^+$. Now let $g = n^i p^j \left(\text{mod } r\right)$, $g' = n^k p^l \left(\text{mod } r\right)$. Then $gg' = n^{ik} p^{jl} \left(\text{mod } r\right) \in G$. □

Proposition 2

Let $\mathcal{Q}_r(x)$ be the r^{th} cyclotomic polynomial over $\mathbb{Z}_p$. Let $h(x)$ be any irreducible factor of $\mathcal{Q}_r(x)$ of degree $\text{ord}_r(p)$. Theorem 0.4 guarantees the existence of such an $h(x)$. Then

$$\mathcal{G} = \left\{ f(x) \bmod \left(h(x), p\right) \neq 0 \mid f(x) \in P \right\} \text{ is a subgroup of } \left(\mathbb{Z}_p[x] \Big/ \left(h(x)\right) \right)^{\times}.$$

Proof

Again it is sufficient to show closure under multiplication since it is a finite set. It is clear that $\mathcal{G}$ is a subset of the multiplicative group since we are taking non-zero residues of polynomials in P. Let $f, g \in \mathcal{G}$. Recall that

$$P = \left\{ \prod_{a=1}^{\ell} (x+a)^{e_a} \;\middle|\; e_a \geq 0 \right\}.$$

Then

$$f = \prod_{a=1}^{\ell} (x+a)^{e_a} \pmod{h(x), p} \;,\; g = \prod_{a=1}^{\ell} (x+a)^{e'_a} \pmod{h(x), p}$$

$$fg = \prod_{a=1}^{\ell} (x+a)^{e_a} \pmod{h(x), p} \prod_{a=1}^{\ell} (x+a)^{e'_a} \pmod{h(x), p}$$

$$= \left(\prod_{a=1}^{\ell} (x+a)^{e_a + e'_a} \right) \pmod{h(x), p} \in \mathcal{G}. \qquad \square$$

Proposition 3

Let $F = \mathbb{Z}_p[x] \big/ \big(h(x)\big)$. If $f(x)$, $g(x)$ are distinct elements of *P*, then they are distinct in *F*.

Proof

We show by way of contradiction that if *f* and *g* are not distinct in *F*, then a polynomial $Q(y)$ with degree less than *t* has *t* distinct roots in *F*, which is impossible.

Let $f(x)$, $g(x)$ be distinct elements of *P*. Suppose $f(x) = g(x)$ in *F*. Let $m \in I$. Then $\big[f(x)\big]^m = \big[g(x)\big]^m$ in *F*. Recall that every $m \in I$ is introspective for every $f(x) \in P$. Since *m* is introspective for *f* and *g*, we have

$$\big[f(x)\big]^m = f\left(x^m\right) \quad \text{in } \mathbb{Z}_p[x] \big/ \left(x^r - 1\right),$$

$$\big[g(x)\big]^m = g\left(x^m\right) \quad \text{in } \mathbb{Z}_p[x] \big/ \left(x^r - 1\right).$$

And since $Q_r(x) \mid x^r - 1$, then $h(x) \mid x^r - 1$.

Thus the equalities above also hold in *F*, and we have that $f\left(x^m\right) = g\left(x^m\right)$ in *F*. This implies that x^m is a root of the polynomial $Q(y) = f(y) - g(y)$ for every $y \in G$. By proposition 1, we have $G \le \left(\mathbb{Z}/r\mathbb{Z}\right)^{\times}$. Thus $\gcd(m, r) = 1$, and each such x^m is a primitive r^{th} root of unity. Thus there will be $t = |G|$ distinct roots of $Q(y)$ in *F*. But the degree of $Q(y)$ must be less than *t*, since $\deg f$, $\deg g < t$ in *F*. □

Lemma 5

Let $\mathcal{G}$ be defined as in Proposition 2. Then $|\mathcal{G}| \geq \binom{t+\ell}{t-1}$.

Proof

Let $F = \mathbb{Z}_p[x] \Big/ \big(h(x)\big)$. Since $h(x)$ is a factor of $\mathcal{Q}_r(x)$, then any root of $h(x)$ is a primitive r^{th} root of unity in F. By Lemma 2, r was chosen such that $\text{ord}_r(n) > \log^2 n$. By Corollary 0.5, $\text{ord}_r(n) \mid \phi(r)$. Thus we have $\text{ord}_r(n) < \phi(r) < r$. Thus $r \log^2 n < r^2$, and $\sqrt{r} \log n < r$.

Recall that $\ell = \left\lfloor \sqrt{\phi(r)} \log n \right\rfloor$. This gives $\ell < r$ and we have already chosen $p > r$. Thus $i \neq j$ in $\mathbb{Z}_p$ for $1 \leq i \neq j \leq \ell$. Thus x, $x+1$, $x+2$, …, $x+\ell$ are all distinct in P. So by Proposition 3, x, $x+1$, $x+2$, …, $x+\ell$ are all distinct in F.

Selecting with repetition $t-1$ of $\ell+1$ distinct objects gives us the number of distinct polynomials of degree $t-1$, and is given by the formula [16]:

$$\frac{\big((\ell+1)+(t-1)-1\big)!}{(t-1)!(\ell)!} = \binom{t+\ell-1}{t-1}.$$

Similarly counting polynomials of degree d, $1 \leq d \leq t-1$, we get

$$\sum_{a=1}^{t-1} \binom{t+\ell-1}{t-a} \geq \binom{t+\ell}{t-1}, \text{ since}$$

$$\binom{t+\ell}{t-1} = \binom{t+\ell-1}{t-1} + \binom{t+\ell-1}{t-2}.$$

Thus there are at least $\binom{t+\ell}{t-1}$ distinct polynomials of degree $< t$ in $\mathcal{G}$. □

Lemma 6

If n is not a power of p then $|\mathcal{G}| \le n^{\sqrt{t}}$.

Proof

Consider $\hat{I} \subset I$ where

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

If n is not a power of p, then by taking $0 \le i \le \left\lfloor \sqrt{t} \right\rfloor$, we see that $\hat{I}$ has at least

$\left(\left\lfloor \sqrt{t} \right\rfloor + 1 \right)^2 > t$ distinct elements, namely

$$\begin{array}{llll} 1 & , p & , p^2 & , \ldots , p^{\left\lfloor \sqrt{t} \right\rfloor} \\ \frac{n}{p} & , \frac{n}{p} \cdot p & , \frac{n}{p} \cdot p^2 & , \ldots , \frac{n}{p} \cdot p^{\left\lfloor \sqrt{t} \right\rfloor} \\ \vdots & & \vdots & \\ \left(\frac{n}{p} \right)^{\left\lfloor \sqrt{t} \right\rfloor} & , \left(\frac{n}{p} \right)^{\left\lfloor \sqrt{t} \right\rfloor} \cdot p & , \left(\frac{n}{p} \right)^{\left\lfloor \sqrt{t} \right\rfloor} \cdot p^2 & , \ldots , \left(\frac{n}{p} \right)^{\left\lfloor \sqrt{t} \right\rfloor} \cdot p^{\left\lfloor \sqrt{t} \right\rfloor}. \end{array}$$

Since $|G| = t$, at least two elements of $\hat{I}$ must be equal mod r. Let these be

m_1 and m_2 , $m_1 > m_2$. And we have,

$$x^{m_1} \equiv x^{m_2} \left(\text{mod } x^r - 1 \right) \qquad (1)$$

Let $f(x) \in P$. Since m_1 is introspective,

$$\left[f(x) \right]^{m_1} \equiv f\left(x^{m_1} \right) \left(\text{mod } x^r - 1, p \right).$$

From (1), we get

$$\left[f(x) \right]^{m_1} \equiv f\left(x^{m_2} \right) \left(\text{mod } x^r - 1, p \right),$$

and since m_2 is introspective,

$$\left[f(x)\right]^{m_1} \equiv \left[f(x)\right]^{m_2} \pmod{x^r - 1, p}.$$

Thus $\left[f(x)\right]^{m_1} = \left[f(x)\right]^{m_2}$ in the field *F*. Therefore, $f(x) \in \mathcal{G}$ is a root of the polynomial $Q'(Y) = Y^{m_1} - Y^{m_2}$ in the field *F*. And since $f(x)$ is an arbitrary element of $\mathcal{G}$, the polynomial $Q'(Y)$ has at least $|\mathcal{G}|$ distinct roots in the field *F*. The degree of $Q'(Y)$ is $m_1 \le \left(\frac{n}{p} \cdot p\right)^{\lfloor\sqrt{t}\rfloor} \le n^{\sqrt{t}}$. Thus $|\mathcal{G}| \le n^{\sqrt{t}}$. □

Lemma 7

If the algorithm returns PRIME, then *n* is prime.

Proof

Suppose that the algorithm returns PRIME. Lemma 5 implies that for $t = |G|$ and $\ell = \left\lfloor \sqrt{\phi(r)} \log n \right\rfloor$:

$$|\mathcal{G}| \ge \binom{t+\ell}{t-1}$$

Since *G* is generated by *n* and *p*, and $o_r(n) > \log^2 n$, we have:

$$t > o_r(n) > \log^2 n$$
$$t^2 > t \log^2 n$$
$$t > \sqrt{t} \log n \ge \left\lfloor \sqrt{t} \log n \right\rfloor.$$

Thus

$$|\mathcal{G}| \geq \binom{\ell+1+\lfloor \sqrt{t}\log n \rfloor}{\lfloor \sqrt{t}\log n \rfloor}.$$

Since *G* is a subgroup of $\mathbb{Z}_r^*$, which has order $\phi(r)$,

$$\phi(r) \geq t \; ; \; \ell = \lfloor \sqrt{\phi(r)}\log n \rfloor \geq \lfloor \sqrt{t}\log n \rfloor.$$

Thus

$$|\mathcal{G}| \geq \binom{2\lfloor \sqrt{t}\log n \rfloor+1}{\lfloor \sqrt{t}\log n \rfloor}.$$

Now $\lfloor \sqrt{t}\log n \rfloor > \lfloor \log^2 n \rfloor \geq 1$ since $t > \log^2 n$, so by Theorem A1 in the Appendix,

$$|\mathcal{G}| > 2^{\lfloor \sqrt{t}\log n \rfloor+1} > 2^{\sqrt{t}\log n} = n^{\sqrt{t}}.$$

By Lemma 6, $|\mathcal{G}| \leq n^{\sqrt{t}}$ if *n* is not a power of *p*. Therefore $n = p^k$ for some $k > 0$. If $k > 1$, then the algorithm will output COMPOSITE in step 1. Therefore, $k = 1$ and $n = p$. □

Theorem 1 (Correctness Theorem)

The algorithm returns PRIME if and only if n is prime.

Proof

Suppose that *n* is prime. Step 1 cannot output COMPOSITE since *n* is not a perfect power. Step 3 cannot output COMPOSITE since $\gcd(a,n) = 1$ or *n* for all $a < r$. Step 5 cannot output COMPOSITE by Lemma 1. Thus the algorithm correctly outputs PRIME in either step 4 or step 6.

The converse was proved in Lemma 7. □

Appendix

Theorem A1

If $n \geq 1$, then $\binom{2n+1}{n} > 2^{n+1}$.

Proof

Expanding the notation, we get:

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

There are exactly n terms in the product in the numerator, so we can rewrite this as:

$$\binom{2n+1}{n} = \prod_{i=1}^{n} \frac{n+i+1}{i} > \prod_{i=1}^{n} \frac{2i+1}{i},$$

since $n \geq 1$. Since each term in this last product is greater than 2, we have:

$$\binom{2n+1}{n} > 2^n.$$

Noticing that $\prod_{i=1}^{3} \frac{2i+1}{i} = \frac{3 \cdot 5 \cdot 7}{1 \cdot 2 \cdot 3} = \frac{35}{2} > 16 = 2^4$, we see that indeed

$$\binom{2n+1}{n} > 2^{n+1}.$$

□

Theorem A2

Let $d_n = \underset{1\le m\le n}{lcm}\left(m\right)$ denote the least common multiple of the first *n* positive integers.

Then for any integer $n \ge 7$, $d_n \ge 2^n$.

Proof

For $1 \le m \le n$, consider the integral, $I = \int_0^1 x^{m-1}\left(1-x\right)^{n-m}$. Using the binomial expansion for $\left(1-x\right)^{n-m}$, we get

$$I = \int_0^1 x^{m-1}\sum_{r=0}^{n-m}\left(-1\right)^r\binom{n-m}{r}x^r = \sum_{r=0}^{n-m}\left(-1\right)^r\binom{n-m}{r}\int_0^1 x^{m+r-1}$$

$$I = \sum_{r=0}^{n-m}\left(-1\right)^r\binom{n-m}{r}\frac{1}{m+r}$$

Since $0 \le r \le n-m$, $m+r \mid d_n$ and we see that $Id_n \in \mathbb{N}$.

On the other hand, integration by parts gives us:

$$I = \int_0^1 x^{m-1}\left(1-x\right)^{n-m}\ ,\ dv = x^{m-1}\ ,\quad u = \left(1-x\right)^{n-m}$$

$$v = \frac{x^m}{m}\ ,\ du = -\left(n-m\right)\left(1-x\right)^{n-m-1}$$

$$I = \underbrace{\frac{x^m\left(1-x\right)^{n-m}}{m}\Bigg|_0^1}_{0} + \frac{n-m}{m}\int_0^1 x^m\left(1-x\right)^{n-m-1},$$

and repeated integration by parts after this manner then yields,

$$I = \frac{\left(n-m\right)!}{n\cdot(n-1)\cdot\ldots\cdot m} = \frac{\left(n-m\right)!\left(m-1\right)!}{n!} = \frac{1}{m\binom{n}{m}}.$$

Alternately, recognizing that I is an Eulerian integral of the 1st kind, we could use the well-known (and more general) identity,

$$B(p,q) = \int_0^1 x^{p-1}(1-x)^{q-1}\,dx = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}.$$

And since for positive integers, $\Gamma(z) = (z-1)!$, we get that

$$I = B(m, n-m+1) = \frac{\Gamma(m)\Gamma(n-m+1)}{\Gamma(n+1)} = \frac{(m-1)!(n-m)!}{n!} = \frac{1}{m\binom{n}{m}}$$

as before.

But we have already shown that $Id_n \in \mathbb{N}$; therefore, $m\binom{n}{m}$ divides d_n for all $1 \le m \le n$. In particular, $n\binom{2n}{n}$ divides d_{2n}.

Notice, however, that

$$\begin{aligned}(2n+1)\binom{2n}{n} &= (2n+1)\frac{(2n)!}{n!\cdot n!} = \frac{(2n+1)!}{n!\cdot n!} \\ &= (n+1)\frac{(2n+1)!}{(n+1)!\cdot n!} = (n+1)\binom{2n+1}{n+1}\end{aligned}$$

so that $(2n+1)\binom{2n}{n}$ divides d_{2n+1}. We also know that $n\binom{2n}{n}$ divides d_{2n+1} since, in general, d_{n-1} divides d_n. And since $\gcd(n, 2n+1) = 1$, we deduce that $n(2n+1)\binom{2n}{n}$ divides d_{2n+1}. Also, since $\binom{2n}{n}$ is the largest of the $2n+1$ terms in the binomial expansion of $(1+1)^{2n}$, we have that

$$d_{2n+1} \geq n(2n+1)\binom{2n}{n} \geq n4^n.$$

Thus, if $n \geq 2$, $d_{2n+1} \geq 2^{2n+1}$ and if $n \geq 4$, $d_{2n+2} \geq d_{2n+1} \geq 2^{2n+2}$. And since, $\{2n+1, 2n+2 \mid n \in \mathbb{Z}, n > 0\} = \mathbb{N}$, we have that $d_N \geq 2^N$, $N \geq 9$.

It can be verified by direct computation that this also holds for $N \geq 7$. □

References

[1] **Agrawal, M.**; **Kayal, N.**; **Saxena, N.** PRIMES is in P. *Ann. of Math. (2)* **160** (2004), no. 2, 781--793. MR2123939

[2] **Garey, M. R.**; **Johnson, D. S.** Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences. *W. H. Freeman and Co., San Francisco, Calif.,* 1979. x+338 pp. ISBN: 0-7167-1045-5 MR0519066 **(80g:68056)**

[3] **Pratt, Vaughan R.** Every prime has a succinct certificate. *SIAM J. Comput.* **4** (1975), no. 3, 214--220. MR0391574 **(52 #12395)**

[4] **Solovay, R.**; **Strassen, V.** A fast Monte-Carlo test for primality. *SIAM J. Comput.* **6** (1977), no. 1, 84--85. MR0429721 **(55 #2732)**

[5] **Adleman, L.; Huang, M.** Recognizing primes in random polynomial time. *Proc. ACM STOC'87*, (1987), 462-470.

[6] **Gill, J.** Computational complexity of probabilistic Turing machines. *SIAM J. Comput.* **6** (1977), no. 4, 675--695. MR0464691 **(57 #4616)**

[7] **Rabin, M. O.** Probabilistic algorithm for testing primality. *J. Number Theory* **12** (1980), no. 1, 128--138. MR0566880 **(81f:10003)**

[8] **Bernstein, D. J.** Detecting perfect powers in essentially linear time. *Math. Comp.* **67** (1998), no. 223, 1253--1283. MR1464141 **(98j:11121)**.

[9] **Nair, M.** On Chebyshev-type inequalities for primes. *Amer. Math. Monthly* **89** (1982), no. 2, 126--129. MR0643279 **(83f:10043)**

[10] **Fouvry, Étienne**. Théorème de Brun-Titchmarsh: application au théorème de Fermat. (French) [The Brun-Titchmarsh theorem: application to the Fermat theorem] *Invent. Math.* **79** (1985), no. 2, 383--407. MR0778134 **(86g:11052)**

[11] **Baker, R. C.**; **Harman, G.** The Brun-Titchmarsh theorem on average. *Analytic number theory, Vol. 1 (Allerton Park, IL, 1995),* 39--103, Progr. Math., 138, *Birkhäuser Boston, Boston, MA,* 1996. MR1399332 **(97h:11096)**

[12] **Lidl, R.**; **Niederreiter, H.** Introduction to finite fields and their applications. Revision of the 1986 first edition. *Cambridge University Press, Cambridge,* 1994. xii+416 pp. ISBN: 0-521-46094-8 MR1294139 **(95f:11098)**

[13] **Apostol, T. M.** Introduction to analytic number theory. Undergraduate Texts in Mathematics. *Springer-Verlag, New York-Heidelberg,* 1976. xii+338 pp. MR0434929 **(55 #7892)**

[14] **Rosser, J. B.**; **Schoenfeld, L.** Approximate formulas for some functions of prime numbers. *Illinois J. Math.* **6** 1962 64--94. MR0137689 **(25 #1139)**

[15] **von zur Gathen, J.**; **Gerhard, J.** Modern computer algebra. *Cambridge University Press, New York,* 1999. xiv+753 pp. ISBN: 0-521-64176-4 MR1689167 **(2000j:68205)**

[16] **Grimaldi, R. P.** Discrete and combinatorial mathematics : an applied introduction. third edition. *Addison-Wesley, Massachusetts*, 1994. xvi+874 pp. ISBN: 0-201-54983-2

[17] **Schönhage, A. ; Strassen, V.** Schnele multiplikation grosser zahlen. (German) [Fast multiplication of large numbers] *Computing*, **7** 1971 281-292