

Derandomization of Primality Testing - A Historical Perspective

Vishnu Vyas

December 14, 2007

1 Introduction

Prime numbers are the building blocks of modern arithmetic. It is a well known fact that the every number which is not prime, can be written as a product of primes, unique upto a permutation. Thus the word *composite number*. An important problem that arises naturally in a variety of fields including cryptography, number theory, and combinatorics is the problem of primality testing - i.e, testing whether a number is prime or not.

This problem, also has a unique place in the development of complexity theory as well. Initially, put in the class of Co-NP problems. Later, due to the efforts of various people was reduced to a problem in Co-RP, i.e, a randomized algorithm for determining whether a number is prime or not, was provided. These algorithms traded in certainty of the answer with efficiency to compute the answer. Finally, in 1992 Adleman and Huang, provided a Las-Vegas algorithm for answering the question whether the number is prime or not.

In late 2006, A deterministic polynomial time algorithm was provided by Agarwal-Kayal-Saxena to answer the same question. In this paper I will elaborate more on the various algorithm and finally discuss the importance of the result.

2 Primality Testing vs Factorization

Primality Testing and Factorization, seem to be closely related on the surface, are in reality very different. As of now, we have efficient algorithms for determining whether a number is prime, however, we still do not have a clear and efficient algorithm for generating the factors of a number. The difference between the two has been known for quite a while, in fact, as early as Gauss commented on the distinction and the importance of these problems to mathematics. We will return to this to get another perspective of the relation between these two problems later in the paper.

3 Primes in Co-NP

The problem of determining whether a number is prime is the dual of another deceptively simple problem - determining whether a number is composite or not. This problem can be easily shown to be in **NP**. Problems which are in **NP** require a certificate which can be used to show whether the number is composite or not. Given a factor of the number, we can easily see if it is composite or not by simply performing one division. Thus, the factor is our certificate. Since, the problem of finding whether a number is prime is complementary to the problem of finding whether a number is composite, the problem of primality testing is in **Co-NP**

4 Primes in NP

We have seen, that it is rather trivial to prove that primes are in **co-NP**. But it can also be shown that Primality testing is also in **NP**. This is achieved by using what are known as Pratt Certificates, first introduced by [1]. To understand pratt certificates, some mathematical preliminaries are needed.

4.1 Fermat's Little Theorem and its converse

Fermat's little theorem[2] is a simple theorem which states that for any prime p and any integer a ,

$$a^p \equiv a \pmod{p} \tag{1}$$

Or in other words, p divides $a^p - a$. Given a few extra conditions, we can write an converse for this theorem. The converse for Fermat's little theorem is also known as Lehmer's theorem[3]. This theorem states that if an integer x is prime to m and $x^{m-1} \equiv 1 \pmod{m}$ and there is no integer $e < m - 1$ for which $x^e \equiv 1 \pmod{m}$, then m is prime. Thus, given such an x one can easily verify whether the number is in fact prime in polynomial time, However generation of such an x is harder and in general cannot be accomplished in polynomial time.

4.2 Non-empty intersection of NP and co-NP

The above detailed result, was also important for complexity theory in that it showed that the classes **NP** and **co-NP** have a non-empty intersection.

5 Primes and Randomized Algorithms

Primality testing has also a long history of being attempted via randomized algorithms. The simplest randomized algorithm for primality testing can be constructed on the basis of Fermat's little theorem [2]. If p is a prime, and for any integer a , we can apply Fermat's little theorem and check whether it is a prime or not. However, this simple test does not have a high probability of

success, as lots of composite numbers also satisfy the above test. We can still repeat the test for various a 's to improve our probability of success. However, there are some numbers which pass this test for all values of X . Infact, there is a countably infinite set of numbers which are not prime, yet can satisfy the above test [4]. These numbers are called Carmichael numbers. However, this test can be fast as all the operations required can be performed modulo p . Also, this test can be used as a very simple test before performing more expensive tests.

In this section we will see some more complicated algorithms that can perform similar operations, with high probability of success.

6 Primes in co-RP

As we can see in the naive test explained above, Primality testing is in co-RP, that is has an one sided error. A much more complicated, but reliable algorithm was provided by [5], which with a high probability can compute wether a number is prime or not.

6.1 The Legendre Symbol, Quadratic Residues and other mathematical preliminaries

If p is a prime and for any integer a , $1 \leq a \leq p-1$, if the congruence $y^2 \equiv a \pmod{p}$ is satisfied then a is known as a quadratic residue of p . The legendre symbol $\left(\frac{a}{p}\right)$ is defined to be $+1$ if a is a quadratic residue of p , -1 if a is not a quadratic residue of p and 0 if p divides a .

It follows from a result of Euler that

$$\left(\frac{a}{p}\right) = a^{p-1/2} \pmod{p} \tag{2}$$

6.2 Solovay-Strassen Algorithm

The equation (2) gives us a simple algorithm for implementing primality testing. If p is a prime, then it follows that p satisfies (2). However, if p is not a prime then definitely

$$\left(\frac{a}{p}\right) \neq a^{p-1/2} \pmod{p} \tag{3}$$

However, there are some composite numbers that also satisfy (2) for some a . These numbers are called *Euler psuedo-prime* with base a . It can also be shown that for composite p , there are atmost $p/2$ a 's, that satisfy (2), and this result is the basis of the Solovay-Strassen Algorithm. The Algorithm is as follows

- Let p be the number being tested for primality
- Randomly chose an integer a where $1 < a < p$
- if p satisfies (3), then output composite, else ouput prime

As, we can see, if the number is composite, the algorithm has a probability of success equal to $\frac{1}{2}$. By repeating the algorithm k times, one can significantly reduce the probability of an error to $\frac{1}{2^k}$. Which is significant considering that for a k of 100, the probability of the error, goes to almost nothing. Algorithms like the Solovay-Strassen algorithm, which have a finite probability of error, are more commonly known as Monte-Carlo Algorithms. In the next section we will see another algorithm by Adelman and Huang, which is a Las Vegas Algorithm.

7 Primes in RP and Las Vegas Algorithms

In the previous section we saw algorithms which allowed for the presence of false positives, In this section we briefly discuss an algorithm, which allows false negatives, i.e, for all composite numbers the following algorithms return composite, however for a small number of prime numbers, the *bad* primes, the algorithm might classify them as composite.

7.1 Elliptic Curve Primality Proving

Various algorithms for primality testing were invented with the use of elliptic curves, including such constructions as [6], [7]. These methods were later extended by [8] to “reduce” the problem of testing whether a prime is p to a harder problem of testing whether a larger prime q is actually a prime. This construction, then can yield highly accurate results. With this, the question of whether a number is prime or not was proved to be in **RP**

7.2 The Las Vegas Algorithm for Primality Testing

Combining the algorithms discussed in the previous sections, one can run these algorithms alternatively and get a new algorithm that uses randomness, yet gives a deterministic output of whether a number is prime or not. These types of algorithms are called as Las Vegas Algorithms.

8 The class ZPP and Las Vegas Algorithms

As we’ve seen, the problem of primality testing is both in **co-RP** and **RP**. This proves that there is a non-empty intersection for the classes **RP** and **co-RP**. This class is called **ZPP**. The algorithms in this class do not have any errors, yet, are able to use randomization to their advantage to produce efficient algorithms.

9 Primes in P - The AKS Algorithm

In late 2006, Manindra Agarwal, Neeraj Kayal and Nitin Saxena of IIT Kanpur, gave a proof of Primality testing that is both Deterministic and in Polynomial

time. This algorithm will be briefly explained in this section. The basis of this algorithm rests on the fact that.

$$(X + A)^N = X^N + A \pmod{N} \quad (4)$$

iff N is prime. The above fact is a simple extension of Fermat's little theorem to polynomials. A simple test of polynomials would be to inspect the binomial expansion of $(x+a)^n$ and inspect all the "middle" terms and see if they are zero. However, the problem with this naive test is it requires $\omega(N)$ operations. The ingenuity of the AKS algorithm comes from the fact that for a suitably chosen r , consider the remainder of the equation (5) when divided by $X^r - 1, \text{ mod } n$.

$$(X + A)^N = X^N + A \pmod{X^r - 1, N} \quad (5)$$

When n is prime, it is trivial to see that this identity holds, however, it is more complicated when n is composite, the proof for which is non-trivial. The algorithm in essence, says that an r can be suitably chosen such that we can achieve a deterministic polynomial time algorithm for primality testing. The algorithm is as follows.

1. Let n be the input number.
2. Test for primality using a fast random test such as Fermat's Little theorem, if it passes the step, go to the next step
3. Find the smallest r such that $\phi_r(n) > \log^2(n)$
4. If $1 < (a, n) < n$ for some $a \leq r$, then output COMPOSITE
5. If $n \leq r$ then output PRIME
6. For a in 1 to $\lfloor \sqrt{\phi(r)} \log n \rfloor$ do
 - if $(X + a)^n \neq X^n + a \pmod{X^r - 1, n}$, output COMPOSITE
7. Output PRIME

9.1 Explanation of the Algorithm

An explanation that appeals to the intuition will be provided in this section. The algorithm outputs prime only under two conditions, in step 5 or in step 7. If it so happened to output PRIME in step 5, and n is not prime, then there must be a non-trivial factor which must have existed in step 4, and thus the algorithm would have output COMPOSITE. So, it can only output PRIME in step 5 if n is prime.

If the algorithm outputs prime in step 7, then n failed to satisfy all the equations in step 6. This can happen only if n is prime, as we have already seen before.

However, if n is composite, the algorithm outputs composite in step 4 or in step 6. It is trivial to see, why the algorithm would output composite in step

4. The case of step 6 is more complicated is omitted here for reasons of brevity. Thus, the algorithm presented here is a deterministic polynomial time algorithm for primality testing.

10 Conclusions

This paper, summarizes many of the most important results in Primality testing and how it played an important part in complexity theory. There is undercurrent theme here, in which the problem of primality testing has led to the proof of a non-empty intersection of the complexity classes **NP** and **co-NP**

Another important result, which can be attributed to developments or work towards primality testing is the existence of the class **ZPP** and development of Las Vegas Algorithm. Again, this is in similar vein to the previous result, where two complementary complexity classes (here, **RP** and **co-RP**) are shown to have a non-empty intersection (in this case the class **ZPP**)

10.1 Primality Testing vs. Factorization

We have already discussed the relationship between the two in section 2. However, it is worth another look, especially from the point of theorem proving. We can draw parallels to Primality Testing and Checking the correctness of a Proof, and Factorization to actually generating a proof. As, we can see that generating a proof for a statement is definitely more harder than checking a given proof. Thus, this gives us an intuitive reason of why primality testing should be in P. This parallel, should serve us well as we still don't have an efficient algorithm for factorization.

We can also, theorize that other pairs of complimentary problems, which share a relationship similar to Factorization and Primality Testing, could also be reduced to polynomial time (atleast one of the two).

References

- [1] Pratt, V., *Every Prime Has a Succinct Certificate*. SIAM J. Comput. 4, 214-220, 1975.
- [2] G.H. Hardy, E.M Wright, *An Introduction to the theory of Numbers*, Oxford University Press, 1979.
- [3] Riesel, H. *Prime Numbers and Computer Methods for Factorization*, 2nd ed. Boston, MA: 96, 1994.
- [4] W.R Alford, A. Granville and C.Pomerance, *There are infinitely many carmicheal numbers*, Annals. Math 140, 703-722, 1994.
- [5] R.M Solovay, V. Strassen, *A fast Monte-Carlo test for primality*. SIAM J. Comput. 6, 84-85.

- [6] S. Goldwasser and J.Kilian, *Almost all primes can be quickly certified* Proc. SOTC. 316-329, 1986.
- [7] A.O.L Atkin and F.Morian, *Elliptic Curves and Primality Proving*. Math. Comp., 61, 29-68, 1993.
- [8] L.Adleman and M.D.A Huang, *Primality Testing and Two Dimensional Abelian vareities over finite fields*, Springer Verlag Notes in Mathematics, 1512, 1992.
- [9] M.Agarwal, N. Kayal and N. Saxena, *Primes in P* Ann. Math., 160, 781-793, 2004.