Story of a Discovery

Somenath Biswas
Department of Computer Science and Engineering,
Indian Institute of Technology Kanpur.

1 August 2002: End of a Quest

On 4 August 2002, a short e-mail message reached the mailboxes of a small group of mathematicians and theoretical computer scientists; the message simply asked for comments on an attached draft paper claiming to provide a deterministic polynomial time algorithm to determine if a number, given as input to the algorithm, is prime or not. The day being a Sunday, the message remained unread in most of the mailboxes. One mathematician of Holland who was then at Berkeley did see the mail on that day itself. The e-mail had come from Manindra Agrawal, whose name was familiar to the computer scientist recipients of the e-mail, but not to the mathematicians. The Dutch mathematician, a leading computational number theorist, was quite used to getting 'solutions' of long-standing open problems of number theory from all kinds of people round the globe, because number theory is an area that fascinates people in general, and whose many celebrated open problems can be stated in the language of school mathematics. The Dutch mathematician must have thought the e-mail to be another one from some naive enthusiast, and yet he could not resist the temptation of having a peek at the attached draft paper– because it was supposed to be about a problem that he himself had been struggling with for more than two decades. Even a cursory glance at the paper was enough to see that it could not be the work of someone mathematically immature. As he examined further, it appeared that the paper had a very original approach, it did not use any of the sophisticated mathematical toolkit whose use is almost mandatory in modern day number theory. The paper was a short one, the proposed decision algorithm was only of thirteen lines, and the correctness proof of the algorithm, the crux of the paper, was detailed in less than three pages. The proof rested on six lemmas, and if their proofs were indeed correct, then it was a great breakthrough. The Dutch mathematician phoned a brilliant young researcher, an Indian, and together they went rigorously through every detail of the paper, sitting in a Berkeley cafe. Finally, after several hours and many cups of coffee, the two were fully convinced that not only was the paper correct in every detail,

but also that it was a gem of a mathematical discovery.

Next day, the Dutch mathematician distributed the paper to some of his fellow researchers; one of whom, after going through the paper, contacted the science correspondent of *The New York Times* informing her of the great breakthrough. She got immediately in touch with Manindra Agrawal, and after a number of e-mail exchanges with him, filed her report. The August 8 issue of *The New York Times* carried prominently on its front page Ms. Robinson's report on the discovery: *New Method Said to Solve Key Problem in Math.* Soon thereafter, all major newspapers around the world also carried the news, thereby making Manindra Agrawal, Neeraj Kayal and Nitin Saxena, the three authors of the draft paper, mathematical celebrities. Curiously, it is some kind of an accident that this great result came about in that summer of 2002, as none of them had planned to stay back for the summer in Kanpur that year to work together on the problem.

Manindra, already a full Professor in the Department of Computer Science at IIT Kanpur would turn thirty-six in August 2002. Neeraj and Nitin had just completed their undergraduate programme in Computer Science in the beginning of May, and were in their early twenties. Manindra had been working on primality testing way back from 1998, sometimes with another colleague, sometimes with undergraduate students, but most often on his own. Although he was not aware of it himself, Manindra had already discovered many of the ingredients of the celebrated result by 2001. Neeraj and Nitin in their BTech project with Manindra as the advisor, had examined certain aspects of the problem, and they had, by the summer of 2002, a very good grasp of the conceptual apparatus the three would finally use.

All the same, when in May 2002 Manindra, Neeraj and Nitin decided to do some more work on the problem, none of them had any clue that the end was so near. As said above, it was by chance that all three were staying back in Kanpur that summer. Manindra was supposed to be in Germany, but at the last minute he cancelled the trip, as he was reluctant to leave behind his wife with two small children. Nitin was expecting a scholarship to study abroad, which somehow did not materialize. Had he been awarded the scholarship, he would certainly have left Kanpur by mid-May to do the many necessary chores for leaving the country in a couple of months time. Neeraj was supposed to be at TIFR, Mumbai that summer to join the PhD programme there, but something had happened and he was feeling too depressed to make any effort to face a new environment.

Perhaps more to take their minds off their disappointments than anything else, Manindra suggested to Neeraj and Nitin that all three together take another look at the primality problem. That was sometime in the second week of May 2002, and some magic started working. Already familiar facts

started tumbling into new places and they could sense that something was emerging. Sometime in the second week of July, one little piece, fairly simple in retrospect, suddenly came to Manindra as he was taking his little daughter to her school. Right then, on the seat of his scooter ambling down a quiet morning IITK road, Manindra knew that they had the final piece of the jigsaw puzzle of the solution of a problem that had remained open since the time of the Eratosthenes.

2 The Challenge that was: Decision Problem for Primes

2.1 Prime numbers

Primes are whole numbers which are greater than 1, and are divisible only by 1 and themselves, 2, 3, 5, 7, 11, 13 are the first six primes. Primes can be thought of as *atoms* for the set of natural numbers, i.e., the set $\{1, 2, 3, 4, \ldots\}$, because every natural number other than 1 is the product of a multiset of primes unique to that number. Another way of saying this is that every natural number has a unique prime decomposition. The set of natural numbers is the cornerstone of mathematics; as Kronecker put it: *'God made the natural numbers, man made the rest'*.[1]

Since the time of the ancient Greeks, primes have fascinated every generation of mathematicians. It was Euclid who proved that there are infinitely many primes, and his proof of this fact is considered one of the all-time gems of mathematics. At first glance, primes appear to be distributed on the number line in an arbitrary fashion: at times two consecutive primes differ only by 2, e.g., 17 and 19, or 71 and 73; and then again, at times, the next prime comes only after a huge gap. However, a closer look yields many fascinating properties; for example, as Gauss and Lengendre had correctly guessed and Hadamard and de la Vallée Poussin later proved, the overall density of primes behaves very nicely as we go down the number line.

2.2 Decision problem, polynomial time algorithm

Undoubtedly, the simplest question about primes one would like to ask is: given a number n, is it a prime? This is the *decision* or the *recognition problem* for primes. What we seek is an *algorithm* that takes as an input

[1] Original in German: *'Die ganzen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk.'* in *Leopold Kronecker* by Heinrich Weber, Jahresberichte D.M.V (1893), pp 5–31.

a number, and decides whether the input number is prime. Of course, one can just invoke the definition: systematically divide n by successive numbers from 2 to $\sqrt{n}$; n is not prime if and only if any of these divisions leaves the remainder zero, that is, if we obtain a proper factor of n. In effect, we are making an *explicit and exhaustive search* for a factor of n. Algorithms such as the one we have just outlined are obvious, naive ones, merely carrying out a systematic search for a solution in the space of all solutions. Algorithms of this nature are unsatisfactory for many reasons, one reason being that whenever the solution space that corresponds to an input is huge, such an algorithm would take too many elementary steps before arriving at its decision for the input. Can we do better for deciding primality?

But then, the question arises, what is *better*? How *good* should the algorithm be for it to be satisfactory? Computational complexity, a sub-area of theoretical computer science which interfaces mathematics and computer science, provides a clear answer to these questions. The measure for *goodness* is taken to be the number of steps the algorithm takes on its input. However, there are infinitely many inputs: in case of a decision algorithm for primality *any number* can be the input to the decision algorithm. The way this is handled is that we view the number of steps the algorithm takes on an input as *a function of the length of representation of the input*. Many inputs can have the same length of representation. For a given length, say l, the definition of the measure of goodness will consider the *maximum* number of steps that the algorithm would take on any input of length l.

We represent numbers by digit strings; a number whose value is n is represented by a string of about $\log n$ digits. Our naive exhaustive search algorithm for deciding primality of n would make, in the worst case, about $\sqrt{n}$ divisions, that is, $10^{(\log n)/2}$ divisions.[2] Thus, for a number with l digits, the number of elementary steps can be as many as $10^{l/2}$. To appreciate why this is not satisfctory, let us consider a concrete case. Assume that the computing device we have can execute a million elementary steps in one second. Then for a number with about 100 digits, the naive algorithm when executed on this computing device can take as long as 10^{12} *centuries* to come up with its answer!

What is said is that a decision algorithm is *good* if there is some *constant* k such that the algorithm takes no more than l^k steps for every input whose size of representation is l, whatever the size l may be. As bigger and bigger inputs are provided to the decision algorithm, the algorithm will of course take more and more steps to come up with its decision, but the growth rate of the steps of a good algorithm is bounded by a polynomial in the size of representation of the input, l^k. Such a decision algorithm is called a

[2]Recall that $\sqrt{n}$ is $n^{\frac{1}{2}}$.

polynomial time algorithm. For a number (whose value is) n, since we require $\log n$ digits to express the number, a polynomial time decision algorithm for primality will have a constant k such that the algorithm will take no more than $(\log n)^k$ steps to come up with its answer, whatever the input n may be. To appreciate why such an algorithm will be much better than the previously considered naive algorithm, suppose we have a primality decision algorithm that takes at most $(\log n)^3$ steps for any number n. Then, the execution of such an algorithm on the computing device of the earlier example will take only about one second on any number with 100 digits.

We are now in a position to state precisely what the challenge was for primality: *to devise a polynomial time algorithm to decide primality.*

The *formal* notion of a polynomial time algorithm is a recent one. However, down the ages whenever mathematcians have looked for an algorithm, their goal has always been what we call now polynomial time algorithms. This intuitive identification of *good* algorithms with polynomial time algorithms arises out of a consideration which is much deeper than mere efficiency of computation. A naive algorithm would explicitly and exhaustively search for a solution. The way a polynomial time algorithm manages to avoid exhaustive search is by making use of some nontrivial *insight* into the nature of the problem at hand. A beautiful example of a polynomial time algorithm which is familiar to all of us is the one discovered by Euclid for computing the gcd (i.e., the greatest common divisor) of two given numbers. Naive use of the definition to compute the gcd of two numbers n_1 and n_2 will require us to systematically try out various numbers to see if it divides evenly both n_1 and n_2, i.e., whether it is a common factor, and then pick the largest common factor as the answer. Euclid's insight was a property of gcd:
The gcd of n_1 and n_2 is the least positive *number that can be expressed as*

$$an_1 + bn_2$$

where both a, b can take any integer value, positive, negative, or zero.
It is this insight that Euclid had exploited to come up with the polynomial time algorithm that we all learn in school for computing gcd. Interestingly, one finds again and again that a good insight which is first discovered in the context of a specific problem, later turns out to be of key importance in contexts that far transcend the original one. Undoubtedly, it is the above insight of Euclid which, centuries later, was crucial to define the notion of an *ideal*, a notion which is of fundamental importance in modern abstract algebra.

One now appreciates why some of the best minds down the ages have been searching for (what we call now) a polynomial time algorithm for deciding primality. It was clear to them that what one needed was an appropriate *insight* into primes, and primes being of such importance to all of mathe-

matics, the insight would have the potential of ushering in great advances in many different areas.

2.3 Hunt becomes intense since 1970s

Since the middle of the 1970s, there have been a number of results that indicated the plausibility of the existence of a polynomial time decision algorithm for primes. Pratt showed in 1975 that for every prime p, there exists a short and easily verifiable proof that p is indeed a prime. One always knew that every composite does have such a short, easily verifiable proof that it is composite: for a composite n, the prover provides a nontrivial factor for n, say n_1, and the verifier carries out the division of n by n_1, and as he would find the remainder to be zero, the verifier is convinced that n is a composite. However, that primes also possess short, easily verifiable proofs was a beautiful revelation. This immediately placed the problem of primality decision amongst a class of problems[3] which researchers did not believe to be inherently hard. Therefore, the belief that this problem would admit a polynomial time algorithm gained currency. In 1976 came Miller's result which proved that primality testing can indeed be done in polynomial time if the so called *Extended Riemann Hypothesis* is true. Although mathematicians do believe the Riemann hypothesis, even its extended version to be true, yet so far the hypothesis, possibly the most important question of mathematics today, has remained unproven. As the correctness of Miller's algorithm rested on an *unproven* hypothesis, it did not settle the question regarding primality, though certainly it was an indication in favour of believing that an unconditional polynomial time decision algorithm does indeed exist. This belief was further strengthened by the discovery of a *randomized* polynomial time primality testing algorithm. Solovay and Strassen in 1977, and subsequently Rabin in 1980 showed that if our computation can make use of random bits (e.g., as would be provided by tossing an unbiased coin), then we can decide primality in polynomial time. One does pay a price though: the answer that such an algorithm gives can be erroneous; and although the error probability can be reduced arbitrarily by using more and more random bits, it cannot be eliminated completely.

The randomized algorithms, particularly the one by Rabin, turned out to be quite acceptable to people who need large primes in real-life.[4] However, for theoreticians these randomized algorithms merely indicated that a *true* polynomial time algorithm must be somewhere round the corner– *all* that one needed was some fresh insight into the nature of primality. Several researchers of the highest calibre from round the world began working on

[3]Technically, this class is known as $\mathbf{NP} \cap \mathbf{co\text{-}NP}$.

[4]Large primes have many practical uses, for example, in secure communication.

the problem trying out every intricate concept and technique of number theory. There was some progress, e.g., Adleman, Pomerance, and Rumely discovered an algorithm that was *almost*[5] polynomial time. Goldwasser and Kilian in 1986 gave another such algorithm. Some progress was seen in the 1990s also, but the goal of a truly polynomial time algorithm remained elusive, in spite of every effort by some of the best minds. Slowly, a feeling came about that primality testing was just too hard a problem for present day mathematics, and that one must await some kind of a breakthrough before a solution could emerge. It was in this setting that Manindra started looking at primality testing, or to put it more correctly, was *led to* the problem.

3 1997–1998: Route to Primality Testing Problem

3.1 Polynomial identity problem

An important open problem of computational complexity is the *identity testing problem for multivariate polynomials*. It was Manindra's interest in this problem that led him to seriously consider the primality testing problem. A multivariate polynomial, in its standard or explicit form, is the sum of a number of *terms*, each term being of the form $bx_1^{i_1}x_2^{i_2}\ldots x_p^{i_p}$, where x_i's are variables, $i_1, i_2, \ldots$'s are non-negative integers, and b, called the coefficient of the term, is some constant element from what is called a commutative ring, a familiar example of which is the set of integers. A simple example of such a polynomial is $x^2 + 2xy + y^2$. The same polynomial could be given in an *implicit* form: $(x + y)^2$. A polynomial is called a *zero polynomial* if the coefficient of each term in the polynomial is zero. The identity testing problem essentially is: given a polynomial, is it a zero polynomial?[6] Of course, if the polynomial is given explicitly, the identity testing problem is trivial: just verify if the coefficient of each term is zero or not. The problem is nontrivial when the polynomial to be tested is given implicitly. A simple example of an implicitly given polynomial which is zero is: $(x + y)(x - y) - x^2 + y^2$. Of course, every implicit representation of a polynomial can be converted to its explicit representation by carrying out some amount of computation. However, the catch is that in general such conversion would take too many steps. As a result, we cannot obtain a polynomial time algorithm for polynomial identity testing simply by converting the given

[5]This algorithm would take no more than $(\log n)^{k \log\log\log n}$ steps for some constant k for any input number n. Although, $\log\log\log n$ grows very very slowly with n, it is not a constant, and therefore, the algorithm is not polynomial time.

[6]More accurately, in the identity testing problem, one is given two polynomials, P and Q, and the question is: are P and Q identically the same? Equivalently, is $P - Q$ a zero polynomial?

polynomial to its explicit representation. The best that is known about the problem is that it can be solved by randomized polynomial time algorithms. The basic fact such algorithms use is that whereas a zero polynomial will evaluate to zero no matter what values are chosen for the variables, a non-zero polynomial will evaluate to a non-zero value for a substantial fraction of possible values that the variables can take.[7] If we choose the values for the variables randomly, the chance is good that if the input polynomial is non-zero, the result of the evaluation of the polynomial on the randomly chosen values will also be non-zero. Thus, a randomized testing algorithm is: Given input polynomial Q in variables $x_1, x_2, \ldots x_m$, randomly choose values for each x_i, and then evaluate the polynomial with these values for the variables. If on evaluation we obtain a non-zero value, we are sure that Q is a non-zero polynomial, and if the evaluation is 0, we conclude that *most probably* Q is a zero polynomial. Note however, that we can never be sure in the latter case– Q might have have been non-zero, but we were unlucky to have picked up values at which Q evaluates to zero.

3.2 Work of Chen and Kao

Chen and Kao, two researchers from Berkeley, came up in 1997 with an interesting new approach to tackle the problem. They considered polynomials over rationals, i.e., each coefficient is a rational number.[8] Since the context is the set of rational numbers, then for using the randomized algorithm of the last paragraph, one would evaluate the input polynomial by plugging in rational values for its variables. What Chen and Kao showed was that if we plug in some carefully chosen *irrational* values into the variables, then the result of the evaluation of the polynomial will be zero if and only if the polynomial is zero. They thus showed that instead of viewing the problem in the given context (in this particular case, the set of rational numbers), the problem can be solved by viewing it in an extended context (here, the extension is done by introducing a few irrationals into the infinite set of rationals).

One cannot do computations with irrationals.[9] Chen and Kao avoided this problem by making use of randomization. Interestingly, they showed that it is possible to use only a fixed amount of randomization and yet make the error probability decrease by using more and more computation. Usually, one needs to use more and more random bits to obtain less and less error.

[7] When we plug in concrete values for the variables of a polynomial, we get a concrete number. This number is called the result of the evaluation of the polynomial for the values chosen for the variables. E.g., $(x+y)^2$ evaluates to 9 for $x = 2, y = 1$.

[8] A rational number is one which can be seen as the ratio of two integers.

[9] Each step of a computation can work with only finite, explicitly given objects, but irrationals have no explicit finite representations.

Thus, Chen and Kao managed to trade (upto a point) randomness with computation.

Because of these two fresh and interesting ideas, Manindra and a colleague of his took immediate notice of Chen and Kao's work and decided to explore further possibilities. In particular, they thought of two possible directions of work: first, extend the Chen-Kao work from the context of rationals to other contexts, in particular, to the context of *finite fields*; second, explore if, for the primality testing problem too, one can trade randomness for computation. Very soon, however, they came to know that a group from MIT had already extended the Chen-Kao work to finite fields. The two colleagues read the preprint of the MIT work. Strangely, neither was disheartened that some other group had already done something they were planning to do. A reason perhaps was that they felt that the MIT work was too closely tied to the *details* of the Chen-Kao work, and not to its essence.

3.3 Two birds with one stone

It was at this time that Manindra took notice of a text-book result:
For every prime p, $(x+1)^p = x^p+1 \pmod p$. In other words, if p is a prime, then the polynomial $(x+1)^p - x^p - 1$ is a zero polynomial in the context of the arithmetic modulo p.[10] Manindra almost immediately realised that this provided a way of reducing the primality testing problem to the polynomial identity testing problem. Because, with a little work one can show that given a number n, the polynomial $(x+1)^n - x^n - 1$ is a zero polynomial $\pmod n$, *if and only if* n is a prime. Therefore, to test if a given nummber n is prime or not, check if $(x+1)^n - x^n - 1$ is a zero polynomial $\pmod n$. This gives us the reduction: if we can solve the polynomial identity testing problem easily (even for univariate polynomials), then we can solve the primality testing problem easily as well.

[10]Ordinary arithmetic essentially is a structure with a set which is the set of integers, and two operations defined on this set, viz., addition and multiplication. In arithmetic modulo n, the set may be thought of as $\{0, 1, \ldots, n-1\}$, and the two operations are addition modulo n and multiplication modulo n. When we add two numbers modulo n, we add the two numbers using ordinary addition, and then divide the resultant by n, and retain the remainder, this being the result of the addition modulo n. For example, $17+15$ modulo 20 is 12. We write this as $17 + 15 = 12 \pmod{20}$. Multiplication modulo n is also defined in a similar way. E.g., $17 \times 15 = 15 \pmod{20}$. It was Gauss who had defined formally the notion of modular arithmetic, and showed that in many ways such arithmetics behave like the familiar arithmetic. We should add here that polynomials in one variable behave very similar to numbers: like numbers, we can add, multiply, subtract and divide one polynomial to/by/from/by another polynomial. Therefore, given two polynomials $p(x)$ and $q(x)$, we can speak of $p(x) \pmod{q(x)}$, the latter is the remainder on dividing $p(x)$ by $q(x)$. Modular arithmetic over polynomials is a notion we will use extensively later.

The reduction further reduced Manindra and his co-worker's interest in the MIT group's extension of the Chen-Kao work, because the MIT extension could not tackle the identity testing problem for the polynomial above.[11] They began examining the specific case of the identity problem right from first principles.

Chen and Kao had extended the context of their concern. The question then was, how does one extend the context of the arithmetic modulo n? When n is a prime, say p, the arithmetic modulo p is a finite field, denoted as F_p,[12] and standard abstract algebra tells us how to extend F_p. Elements of the extension can be seen as those polynomials which are obtained as remainders when we divide general polynomials by a specific polynomial.[13]

Dividing polynomials by polynomials to consider the remainders left– with this idea somewhere at the back of their mind, one afternoon in the latter half of 1998, Manindra and his colleague came upon a very simple idea for testing if a given univariate polynomial is zero or not: suppose the given polynomial $g(x)$ is indeed a zero polynomial. Then no matter which polynomial one uses to divide g with, the remainder will always be zero. On the other hand, if g is not a zero polynomial, then division by *only a few* polynomials will leave the remainder zero.[14] It is easy computationally to obtain the remainder polynomial which results from division of a polynomial $g(x)$ by any small degree *monic* polynomial $h(x)$, in the context of modulo n arithmetic, whatever n may be. Since there are a huge number of monic polynomials for doing the division, the simple observation above led Manindra and his colleague to a randomized test to see if a given univariate polynomial $g(x)$ is a zero polynomial modulo n or not: choose a small degree monic polynomial $h(x)$ randomly, and then obtain the remainder of division of $g(x)$ by $h(x)$. If the remainder is not a zero polynomial, then $g(x)$ is definitely not a zero polynomial. If on the other hand the remainder is a zero polynomial, then *most probably* so is $g(x)$.

Since this randomized algorithm can test if $(x+1)^n - x^n - 1$ is zero modulo

[11] There are two reasons: first, the MIT work could handle polynomials only of a small degree, whereas the above polynomial is of a degree whose value is (exponentially) large. Secondly, in general, arithmetic of modulo n will not be a finite field.

[12] Elements of F_p are the numbers $0, 1, \ldots p-1$, and the two operations of F_p on these numbers are: addition modulo p, and multiplication modulo p.

[13] The specific polynomial has to be an *irreducible* polynomial over F_p, that is a polynomial which does not non-trivially factorize in the context of F_p.

[14] One can appreciate the situation easily if one considers numbers instead of polynomials. Actually, the comparison is not out of place technically, because many properties are commonly shared by arithmetic with numbers, and arithmetic with polynomials. Suppose we are given a number N (implicitly or explicitly). If N is zero, then *every number* will divide the number N. On the other hand, if N is not 0, then only the factors of N will divide it. The number of factors of a number N is very few in comparison with the value of N.

n or not, they now had a new randomized algorithm for primality testing. Importantly, theirs was the *first* algorithm which could tackle exponentially large degree polynomials. Soon thereafater, Manindra came up with a simple method for reducing the identity testing problem of multivariate polynomials of relatively small degrees to that of univariate polynomials. (The resulting univariate polynomial may have exponentially large degree, but as said before, their algorithm can handle large degree univariate polynomials). Therefore, in addition to a new randomized polynomial time primality test, Manindra and his colleague now also had a new randomized polynomial time identity test for all the classes of general polynomials considered till then.

There was an irony in the new results: although what first spurred Manindra and his colleague was the idea of *extending* the given context, in effect what they finally exploited was the *breaking down* of the given context. For numbers, the 'breaking down' idea is embodied in a well-known result called the *Chinese remainder theorem*. The proposed new algorithms essentially make use of the Chinese remainder theorem for polynomials. This was why the publication that disseminated the new results was entitled '*Primality and Identity Testing via Chinese Remaindering'*. This was concluded in mid 1999. By then, Manindra's colleague had left IIT Kanpur on sabbatical leave. Before he left, Manindra and the colleague had discussed a few times if and how their randomized test could be turned to a deterministic polynomial time algorithm, thereby setting at rest the real big problem. Invariably, these discussions would get stuck at very hard technical issues. Manindra's colleague did not really believe that anything further could be done from what they had. Had not all others, in spite of using highly sophisticated tools, failed so far in discovering a polynomial time primality test? In comparison with those attempts, the Chinese remaindering idea was fairly elementary. In any case, they already had a nice, state-of-the-art result; Manindra's colleague left Kanpur on his sabbatical fairly contented.

4 1999–2001: Period of Groping

Manindra, however, was far from contented, he had a hunch that more could be done. Off and on, during 1999 to 2001, he would think about how their randomized primality testing could be derandomized. Quite early on, he came to focus his attention on a condition that he felt was worth investigating as a possible test for primality.

4.1 An intriguing test

In the randomized primality test that used Chinese remaindering, the polynomial $(x+1)^n - x^n - 1$ was divided by a randomly chosen polynomial. The algorithm would not give the correct answer if the choice was a wrong one. Given input n, could one choose a polynomial carefully so that the choice would never be wrong? Manindra had a hunch that the answer to this question is *yes*, and moreover, he felt that it was worthwhile to investigate if such a polynomial could simply be of the form $x^r - 1$, for a suitable choice of r. Given an n, the primality of which we would like to decide, we find a suitable r, and then check if the remainder of dividing $(x+1)^n - x^n - 1$ by $x^r - 1$ is zero in modulo n arithmetic. Stated another way, the condition that we are checking is: whether the two polynomials, $(x+1)^n$ and $x^n - 1$, are equal or not, when performing modulo operations simultaneously with respect to both n and $x^r - 1$ on the two polynomials. Expressed in symbols, the condition is:

$$(x+1)^n = x^n + 1 \pmod{x^r - 1, n} \tag{1}$$

Without any loss of generality, r above can be taken as a prime.

If r is not too large[15], then in polynomial time we can check whether n satisfies the condition above or not. The hope is that if one finds the right small prime r for the given n (and if indeed, should such an r exist in the first place), then verifying the condition (1) will be the same as verifying $(x+1)^n = x^n + 1 \pmod{n}$. Since the latter condition is satisfied only by a prime we will then have a polynomial time algorithm for primality testing.

Upto a point one can appreciate the preference for the specific form $x^r - 1$ for the polynomial for taking the modular operation. One reason is that performing the modular operation becomes computationally very simple: going modulo $x^r - 1$ means considering $x^r - 1$ as 0, which in turn means replacing x^r, wherever it may occur, by 1. Thus, whenever we encounter some term x^{r+k} where $k \geq 0$, we simply replace it by x^k. Another reason is that when r is a prime, $x^r - 1$ factorizes into (besides the factor $(x-1)$) the largest possible irreducible factors, and this has certain technical advantages.

At the same time it appears very unlikely that there would exist a small prime r such that the test (1) by itself, which after all makes use of just one polynomial modular operation, could give us all the relevant information. In spite of it apparently being such a long shot, Manindra just could not leave (1) as a possibility. During the period of 1999 to 2001, he tried several different approaches to prove the sufficiency of this test. In the process, he discovered two properties of (1) which later on would turn out to be two

[15] For some constant k, we should have $r \leq (\log n)^k$.

key ingredients of the final proof that Manindra, Neeraj and Nitin would discover in 2002.

Notational convention:
In the rest of the article, the symbol n will always be used to denote the number whose primality we wish to check, and r will be used for a prime which is small with respect to n.

4.2 Two key ingredients

We need to consider a slightly different version of the test (1) to appreciate the two key ingredients. Given an input number n, what we check computationally is whether the condition in (1) is satisfied or not. However, because n can be a composite in general, analyzing the consequences of modular n arithmetic can become quite a messy affair technically. What is simpler is to consider modulo p arithmetic for some prime p. Let p be a prime factor of the given n. Since two quantities which are equal modulo n are also equal modulo p, the condition

$$(x+1)^n = x^n + 1 \pmod{x^r - 1, n}$$

implies

$$(x+1)^n = x^n + 1 \pmod{x^r - 1, p} \tag{2}$$

It is known that for every polynomial $g(x)$, and for every power p^k of any prime p, the following is satisfied:

$$(g(x))^{p^k} = g(x^{p^k}) \pmod{p}$$

Therefore, it follows that the condition

$$(g(x))^{p^k} = g(x^{p^k}) \pmod{x^r - 1, p}$$

will be satisfied for every polynomial $g(x)$, and for every power p^k of any prime p.

We know, therefore, that the numbers $1, p, p^2, p^3, \ldots$, i.e. every power of p, will satisfy[16] the condition (2). Suppose our n is such that it satisfies (2). Therefore, we know that powers of p, and n are instances of numbers that satisfy (2). A natural question then is: What can we say *in general* about numbers that satisfy (2)? What we call the *first key ingredient* is the

[16] When we say that m satisfies (2), we mean that

$$(x+1)^m = x^m + 1 \pmod{x^r - 1, p}$$

holds.

following property of this set of numbers:

First key ingredient:
If n_1 and n_2 satisfy (2), n_1, n_2 not necessarily distinct, then their product, viz., $n_1 n_2$, will also satisfy (2).

The second key ingredient is also a property about the numbers that satisfy (2). It can be stated as:

Second key ingredient:
If n_1 and n_2 satisfy (2), and are both congruent modulo r, then n_1 and n_2 will also be congruent modulo N, where N is a number which in general will be much larger than r.[17]

In other words, the above property says that if n_1 and n_2 both satisfy (2), and r divides $n_1 - n_2$, then a much larger number N will also divide $n_1 - n_2$. This is a remarkable property; let us understand why it is so.

Let us keep in mind that r is a small number, whereas N is a very large number. Consider the range 0 to $N-1$, and suppose that we have two numbers n_1 and n_2, both belonging to this range, and both satisfying (2), which are congruent modulo r, that is, both leave the same remainder on division by r. The second key ingredient says that n_1 and n_2 must also leave the same remainder on division by N. But that is not possible if n_1 and n_2 are distinct. Because, as each is less than N, on division of such a number by N, we will get the number itself as remainder. The conclusion, therefore, is that in the range 0 to $N-1$, two distinct numbers satisfying (2) cannot be congruent modulo r, they must leave different remainders on division by r. There are only r possible distinct remainders on division by r (which are: $0, 1, \ldots r-1$). Therefore, in the huge range 0 to $N-1$, there can only be *at most* r numbers which can satisfy (2).[18]

To summarise, whereas the first key ingredient says that lots of numbers *can* satisfy the condition (2), (because the moment two numbers do, then their product will also satisfy (2)), the second key ingredient says that such numbers cannot occur densely.

[17] When we say that two numbers are congruent modulo a number, say k, it means that the two numbers both leave the same remainder on division by k. Another way of stating it is that the difference between the two numbers is a multiple of k. For example, 28 and 73 are congruent modulo 5.

[18] Much later, Neeraj and Nitin during their BTech project work independently discovered this key ingredient. Neeraj was so excited on noticing the property that he could not sleep the entire night!

4.3 A hazy landscape

It is clear that each of the two, what we have called key ingredients, on its own reveals something of interest about the intriguing test (1). However, during the period 1999 – 2001, neither Manindra, nor those of his students who had spent time on the primality question, could make use of these ingredients to make a definite advancement. And yet, Manindra could discern a hazy landscape which he kept on exploring in various ways.

For (1) to be an effective primality test, one needs to show that if n satisfies (1), then, as we had outlined earlier, n *cannot be* any number other than some power of a prime. We have also said that technically it is more viable to deal with (2), which n satisfies by implication when it satisfies (1). We have seen that all powers p^k of p will satisfy (2), where p is a prime factor of n, in terms of which (2) is expressed. Therefore, the goal Manindra had in mind was to prove that *no number other than the various powers of* p*, namely,* $1, p, p^2, p^3, \ldots$ *can possibly satisfy* (2).

Suppose n satisfies (2). Then we know from the first key ingredient that $n \times n$, i.e., n^2 will also satisfy (2), and then so will n^3, and so on. Thus, *every* power n^s of n will satisfy the condition. We also know that every power p^t of p will also satisfy (2). Let us now consider the second key ingredient. Let N in the statement of the second key ingredient be larger than both n^s and p^t. Therefore, $1, p, p^2, \ldots, p^t$ as well as $n, n^2, \ldots, n^s$ all lie in the range 0 to $N-1$. However, the second key ingredient tells us that in this range there are at most r places where these powers of p's and n's can be, and thus $1, p, \ldots p^t$ and $n, n^2, \ldots n^s$ all compete for these r positions. One may hope that due to this jostling for occupying the few available positions, some power p^i of p, and some power n^j of n may come to occupy the same position. That is, $p^i = n^j$, which immediately implies that n is some power of p,[19] and the goal is reached.

If things work out as above, then we do have a polynomial time primality test. Given an n, we first check if it is of the form a^b, where both $a, b \geq 2$.[20] If this is the case then we conclude that n is composite, and we are done. Next, we test if n satisfies the condition (1). If it does not, again we know that n is composite. If n does satisfy (1), then it is a prime.

Although this scenario does seem to contain the goal, many issues are hazy. We may not find a suitable small prime r, N may not be large enough, and most damagingly, all of $1, p, \ldots, p^t$ and $n, n^2, \ldots, n^s$ may find their places uniquely in the range 0 to N, without any p^i and n^j coinciding.

[19] The simple reasoning for this is: the number p^i has only p as its prime factor. Any number (here n^j) which happens to be equal to p^i cannot then contain any prime factor other than p.

[20] Computationally, this check can be done in polynomial time.

At this juncture, Manindra would have created in his mind a list of the issues to be tackled in order to chart a path through this confusing and hazy landscape. First of all, one needed to ensure that N, of the second key ingredient, should be large. For this to happen, first, the size of a certain group[21] needs to be large; and next, the order of $(x+1)$ in this group needs to be large. For the group to be large, the order of p modulo r should be large.[22] How would one ensure this condition? Manindra had noted an obvious point: if order of p modulo r had the largest possible value, viz., $r-1$, then the goal was attained (assuming N was large enough), because then all the available r slots in the range would be taken up by the powers of p, thereby forcing n to coincide with one of these powers. Of course, another unresolved issue was how to obtain an appropriate value of r, which would be small enough to be found out by an exhaustive search in polynomial time, for all the happy things to happen. r could not be too small either, for in that case N would not be large enough.[23] It was also clear to Manindra that the nice case that arose because the order of p modulo r took the largest possible value was at best suggestive of the possibility that large order of p might be helpful, but one could not pin one's hope on p actually taking a value even close to the largest possible one. Though number theory did suggest that there are many primes p whose order modulo a fixed r are near to the maximum value, it would be foolhardy to expect that the input n would have such a prime as one of its factors.

The reader may very well find the discussion above to be quite confusing; it was indeed an era of confusion with many issues requiring clarity and no clear direction in sight. And yet, Manindra had a hunch that it would be worthwhile to focus attention to two questions:

1. how large can one *guarantee* order of p modulo r to be?, and
2. how large can one *guarantee* N to be?

As we will see later, Manindra's hunch did lead him and his two students to the right path.

[21] It is the multiplicative group of $F_p[x]/(g(x))$, $g(x)$ being an irreducible factor of x^r-1.

[22] Order of p modulo r is the least value m such that $p^m = 1 \pmod{r}$. Order of $(x+1)$ is similarly the least value m such that $(x+1)^m$ in the multiplicative group is the identity element of the group.

[23] Quantitatively, r cannot be larger than $(\log n)^k$ for some fixed constant k, at the same time it has to be larger than $\log n$.

4.4 End of the era: some encouragement, some disappointment

Manindra had asked an undergraduate student to do some experiments. The student, Rajat Bhattacharya, shared with Manindra the enhusiasm for (1) as a basis for primality testing. He tested all numbers upto hundred million and found that composites that satisfied the condition (1), did so for very few r's even in the small range of 2 to 100. Thus (1) indeed worked as a criterion for separating all primes from composites in the range of 2 to hundred million.

Neeraj Kayal and Nitin Saxena started their BTech project work in August 2001 with Manindra as the supervisor. They further extended the experimental work of Rajat, and checked numbers till 10^{10} (i.e., ten billion). Again, it turned out that (1) as a criterion for primality testing was spectacularly successful. The three thus had a fair amount of empirical support that the criterion may work for all numbers. Neeraj and Nitin, with all their youthful zeal and energy, joined Manindra in the hunt for a *proof*. It is interesting that on their own, the two students rediscovered several facts that Manindra had found before, most notably, the second key ingredient. Though the facts they discovered and the observations they made were quite interesting, Neeraj and Nitin realised, as Manindra had done previously, that some crucial links of the proof were missing, if indeed such a proof did exist. They completed their BTech project at the end of April 2002; it was exceptionally good work for an undergraduate project work and earned them the *Best BTech Project Award* of the Department for that year. In early May, their bags packed, Neeraj and Nitin were ready to leave IIT Kanpur having graduated with their BTech degrees.

5 May – August, 2002: Discovery of the Missing Links

It so happened, however, that Neeraj and Nitin stayed on for the summer in Kanpur, for reasons which we have narrated earlier. This time around no sooner had Manindra and his two students started taking a fresh look at their problem of proving the adequacy of (1) for primality testing, than they discovered one of the missing links: how to make N (of the second key ingredient) *provably* large. They came upon this missing link by modifying their earlier view slightly: instead of verifying the condition

$$(x+1)^n = x^n + 1 \pmod{x^r - 1, n}$$

verify

$$(x+a)^n = x^n + a \pmod{x^r - 1, n}, 1 \le a \le T \tag{3}$$

That is, try out (3) for all values of a's lying in a certian range 1 to T. If the condition holds for all the $(x+a)$'s, then it is an easy deduction that the condition

$$(g(x))^n = g(x^n) \pmod{x^r - 1, n}$$

also will be true where $g(x)$ is any polynomial obtained by multiplying together various powers of each $(x+a)$. In particular therefore, this $g(x)$ can be taken to be the *generator* of the underlying cyclic multiplicative group, which makes the corresponding N (the order of $g(x)$ in the group) as large as the group itself. Suppose that order of p modulo r is d. Manindra and his students knew that a lower bound on the size of the group could be given in terms of this d; this lower bound being 2^d. Therefore, how large N could be was now seen to depend crucially on d.

Manindra saw that it was fairly easy to prove that a small (relative to n) prime r existed such that d is at least $\sqrt{r}$. Now, the remaining big question was: could one provably show that there would exist some p^i and some n^j, both less than N (thereby being in the range 0–$N-1$), such that both eavaluate to the *same* value modulo r? Again a slight shift in perspective led the way forward. So far they were considering $1, p, p^2, \ldots$ and $n, n^2, \ldots$ for a clash in the range 0 to $N-1$. One morning in July 2002, as Manindra was taking his younger daughter to the campus school, the shift in perspective suddenly occurred to him. Instead of considering powers of p's and powers of n's, consider $n^i p^j$ for various values of i and j. [24]. Let each i and j take every value in the range 0 to $\sqrt{r}$. That gives us more than r $n^i p^j$'s, and therefore, two distinct $n^i p^j$'s must evaluate to the same value modulo r.[25]

The equality modulo r is also an equality modulo N, from the second key ingredient. If $N \geq n^{2\sqrt{r}}$, [26] then the equality modulo N will not just be a conditional, modular equality, but the equality will be unconditional. In other words, we will have $n^{i_1} p^{j_1} = n^{i_2} p^{j_2}$ for some i_1, i_2, j_1, j_2, where at least one of $i_1 \neq i_2$ or $j_1 \neq j_2$ holds. Immediately then, n is some power of p, and we are done.

The only issue that remained was to ensure that N would be greater than $n^{2\sqrt{r}}$. Manindra could show fairly easily that this would be achieved if d, the order of p modulo r, was to satisfy $d \geq r^{\frac{1}{2}+\epsilon}$, for any constant (which could be a fraction) ϵ, however small. There was a proviso, however; for d to have such a value, r needs to be of the magnitude $(\log n)^k$, for a certain

[24] The second key ingredient is applicable for $n^i p^j$'s, because, from the first key ingredient we know that $n^i p^j$ will satisfy (3), as both n^i and p^j do.

[25] This kind of argument, though very simple, is very powerful– it goes by the name *pigeonhole principle*: there are more than r pigeons ($n^i p^j$'s) and only r pigeonholes (the r possible values modulo r operation can take), and the commonsensical principle says that (at least) two of the pigeons must find themselves in the same pigeonhole.

[26] Recall that p is a factor of n, so $p < n$. Therefore, $n^i p^j < n^{i+j}$, and as each $i, j \leq \sqrt{r}$, every $n^i p^j$ that we are considering is *less than* $n^{2\sqrt{r}}$.

constant k depending on ϵ. An earlier simple reasoning of Manindra's could ensure the condition that $d \geq \sqrt{r}$, i.e., $d \geq r^{\frac{1}{2}}$. But if such simple reasoning could give a bound on d which was almost adequate (but not quite!), would not number theorists have already discovered some results implying a better bound on d? With this in mind, Manindra and his two students did a search on the Internet and came across a result by Fouvry, a French mathematician, proved way back in 1985. From Fouvry's result, in a few steps, came the proof that a prime r in the appropriate range can be found corresponding to which d would be as large as $r^{\frac{2}{3}}$. And that concluded the quest for a deterministic polynomial time algorithm for primality testing that Eratosthenes had started around 300 BCE.

6 Some Speculative Remarks on Truth and Beauty

'Elegance', 'beauty' are words which are often used in describing great mathematical results. The work of Manindra, Neeraj and Nitin also has been hailed as one of great elegance and beauty. As one reads their paper, one gets the feeling that they first select a number of opaque, irregular shaped pieces, and then they assemble these pieces together and suddenly what emerges is a translucent sculpture. They use familiar mathematical objects in an unfamiliar manner, they use facts and attributes that were previously regarded as reasons for difficulty in bringing about clarity. They make use of a small prime to unlock the primality question of a large candidate number. Till then people had found primes which are *smooth* to be easier to handle, but they make essential use of the very non-smoothness of one prime to decide the primality of the input number. Indeed, the paper of Manindra, Neeraj and Nitin is a work of beauty in the same sense that many great results are considered beautiful.

However, I would like to speculate on a deeper notion of beauty. In 2002, soon after the discovery was made, Manindra was asked by a media person, 'What made you keep on looking for a deterministic polynomial time algorithm for primality testing in spite of knowing that the problem was so hard?' Manindra's reply was that because he had an approach which no one had tried before, he felt that it was worth pursuing. Later, he confided in someone that that had been only half the answer– the other half was that he *knew* that the approach would succeed. One is tempted to enquire into the nature of such *knowledge*. Surely, no one could really know any such thing in the familiar sense of the word *know*; after all, had Manindra shared with any other researcher all that he was *consciously* aware of till, say, 2001, about his approach, most certainly the researcher would not have pursued the approach with any degree of seriousness for long. What was it that Manindra *knew* about the issue which was so convincing to him, but

would not have been so convincing to someone else?

The history of science is replete with many examples, as Lakatos has pointed out, where it appears that one first *knows* the end result; the verification or proof comes later, through a process guided by a kind of conviction that no hypothesis can ever generate, but a vision of truth can. Kepler verified his laws of planetary motion through a huge amount of calculations on observed data. Curiously, there were many mistakes in his calculations, but overall, the mistakes nullified one another. No sane man would climb up a tall tower, as Galileo did, merely to observe two objects of different weights fall from the height, unless he *knew* that the result would be different from what the authority of Aristotle had pronouncd. One is tempted to hold that such *knowledge* and the resultant conviction can come only from an awareness of *truth*, not in the ordinary sense of the word, but as that which *'is what the voice within tells you.'*

However, there are also voices without, and they tell us to be prudent, to be careful, not to waste time in running after that which could very well turn out to be a mere chimera. How comes it then that one can ignore these voices and remain steadfast to the voice within? The reason perhaps is that simultaneously the eye within sees something of great beauty. Manindra found something so attractive in the the condition (2) that he just could not ignore it, he kept coming back again and again to (2) in spite of many and repeated disappointments. The voyage of discovery is often a long and painful one, but what guides the voyage is perhaps the perception of truth and what sustains the voyage is a vision of beauty. Ours is a blessed existence because some amongst us sometimes are capable of perceiving truth and beauty.

Notes and Acknowledgements: The Dutch computational number theorist in the Introduction section is Heinrich W Lenstra, Jr. The young Indian mathematician who along with Lenstra carefully verified the correctness of the proofs in the manuscript Manindra had sent is Manjul Bhargav. Carl Pomerance was the colleague of Lenstra who had contacted Sara Robinson, *The New York Times* science correspondent, giving her the news of the discovery. Manindra's colleague mentioned in Section 3 is Somenath Biswas.

By making use of an observation made by Lenstra, Manindra, Neeraj and Nitin were able to avoid depending on the density result of Fouvrey, thereby making their final proof completely elementary; the journal version of their result appeared as the paper *PRIMES is in P*, Annals of Mathematics 160(2): 781 – 793 (2004).

I am grateful to Manindra Agrawal for many discussions and to Kiran Biswas for her help in getting rid of many awkward sentences. My sincere thanks to V. Arvind, Amiya Dev and Rajnish Mehra for their comments on the draft version of this article.