Journal of Advances in Science and Technology
Vol. 14, Issue No. 1, June-2017, ISSN 2230-9659

An Analysis upon Various Algorithmic Solutions in Algebraic Number Theory

Vijaysingh Digambar Gaikwad*

Assistant Professor, Dayanand Science College, Maharashtra

Abstract – Algebraic number theory studies algebraic properties of the ring of algebraic integers in a number field. We describe various algebraic invariants of number fields, as well as their applications. These applications relate to prime ramification, the finiteness of the class number, cyclotomic extensions, and the unit theorem.

In this study we discuss the basic problems of algorithmic algebraic number theory. The emphasis is on aspects that are of interest from a purely mathematical point of view, and practical issues are largely disregarded. We describe what has been done and, more importantly, what remains to be done in the area. We hope to show that the study of algorithms not only increases our understanding of algebraic number fields but also stimulates our curiosity about them. The discussion is concentrated of three topics: the determination of Galois groups, the determination of the ring of integers of an algebraic number field, and the computation of the group of units and the class group of that ring of integers.

----------♦----------

INTRODUCTION

The main interest of algorithms in algebraic number theory is that they provide number theorists with a means of satisfying their professional curiosity. The praise of numerical experimentation in number theoretic research is as widely sung as purely numerological investigations are indulged in, and for both activities good algorithms are indispensable. What makes an algorithm *good* unfortunately defies definition—too many extra-mathematical factors affect its practical performance, such as the skill of the person responsible for its execution and the characteristics of the machine that may be used.

The present study addresses itself not to the researcher who is looking for a collection of well-tested computational methods for use on his recently acquired personal computer. Rather, the intended reader is the perhaps imaginary pure mathematician who feels that he makes the most of his talents by staying away from computing equipment. It will be argued that even from this perspective the study of algorithms, when considered as objects of research rather than as tools, offers rich rewards of a theoretical nature.

The problems in pure mathematics that arise in connection with algorithms have all the virtues of good problems. They are of such a distinctly fundamental nature that one is often surprised to discover that they have not been considered earlier, which happens even in well-trodden areas of mathematics; and even in areas that are believed to be well-understood it occurs frequently that the existing theory offers no ready solutions, fundamental though the problems may be. Solutions that have been found often need tools that at first sight seem foreign to the statement of the problem.

Algebraic number theory has in recent times been applied to the solution of algorithmic problems that, in their formulations, do not refer to algebraic number theory at all. That this occurs in the context of solving diophantine equations does not come as a surprise, since these lie at the very roots of algebraic number theory. A better example is furnished by the seemingly elementary problem of decomposing integers into prime factors. Among the ingredients that make modern primality tests work one may mention reciprocity laws in cyclotomic fields, arithmetic in cyclic fields, the construction of Hilbert class fields of imaginary quadratic fields, and class number estimates of fourth degree CM-fields. The best rigorously proved time bound for integer factorization is achieved by an algorithm that depends on quadratic fields, and the currently most promising practical approach to the same problem, the *number field sieve*, employs "random" number fields of which the discriminants are so huge that many traditional computational methods become totally inapplicable. The analysis of many algorithms related to algebraic number fields seriously challenges our theoretical understanding, and one is often forced to argue on the basis of heuristic assumptions that are formulated for the occasion. It is considered a relief when one runs into a standard conjecture such as the generalized Riemann hypothesis

or Leopoldt's conjecture on the no vanishing of the *p*-adic regulator.

In this study we will consider algorithms in algebraic number theory for their own sake rather than with a view to any of the above applications. The discussion will be concentrated on three basic algorithmic questions that one may ask about algebraic number fields, namely, how to determine the Galois group of the normal closure of the field, or, more generally, of any polynomial over any algebraic number field; how to find the ring of integers of the field; and how to determine the unit group and the ideal class group of that ring of integers. These are precisely the subjects that are discussed in *Algorithmic algebraic number theory* by M. Pohst and H. Zassenhaus (Cambridge, 1989), but our point of view is completely different. Pohst and Zassenhaus present algorithms that "yield good to excellent results for number fields of small degree and not too large discriminant", but our attitude will be decidedly and exclusively asymptotic. For the purposes of the present study one algorithm is considered better than another if, for each positive real number *N*, it is at least *N* times as fast for all but finitely many values of the input data. It is clear that with this attitude we can make no claims concerning the practical applicability of any of the results that are achieved. In fact, following Archimedes one should be able, on the basis of current physical knowledge, to find an upper estimate for all sets of numerical input data to which any algorithm will ever be applied, and an algorithm that is faster in all those finitely many instances may still be worse in our sense.

To some people the above attitude may seem absurd. To the intended reader, who is never going to apply any algorithm anyway, it comes as liberation and a relief. Once he explicitly gives up all practical claims he will realize that he can occupy himself with algorithms without having to fear the bad dreams caused by the messy details and dirty tricks that stand between an elegant algorithmic idea and its practical implementation. He will find himself in the platonic paradise of pure mathematics, where a conceptual and concise version of an algorithm is valued more highly than an ad hoc device that speeds it up by a factor of ten and where words have precise meanings that do not change with the changing world. He will never need to enter the dark factories that in his imagination are populated by applied mathematicians, where boxes full of numbers that they call matrices are carried around and where true electronic computers are fed with proliferating triple indices. And in his innermost self he will know that in the end his own work will turn out to have the widest application range, exactly because it was not done with any specific application in mind.

There is a small price to be paid for admission to this paradise. Algorithms and their running times can only be investigated mathematically if they are given exact definitions, and this can apparently be done only if one employs the terminology of *theoretical computer science*, which our intended reader unfortunately does not feel comfortable with either. It is only out of respect for his feelings that I have not called this paper *Complexity of algorithms in algebraic number theory*, which would have described its contents more accurately.

Although it is, from a rigorous mathematical point of view, desirable that I define what I mean by an algorithm and its running time, I will not do so. My main excuse is that I do not know these definitions myself. Even worse, I never saw a treatment of the appropriate theory that is precise, elegant, and convenient to work with. It would be a laudable enterprise to fill this apparent gap in the literature. In the meantime, I am happy to show by example that one can avoid paying the admission price, just as not all algebraists are experts on set theory or algebraic geometers on category theory. The intuitive understanding that one has of algorithms and running times, or of sets and categories, is amply sufficient. Exact definitions appear to be necessary only when one wishes to prove that algorithms with certain properties do *not* exist, and theoretical computer science is notoriously lacking in such negative results. The reader who wishes to provide his own definitions may wish to consult for an account of the pitfalls to be avoided. He should bear in mind that all theorems in the present study should become formal consequences of his definitions, which makes his task particularly academic.

My intended reader may have another allergy, namely, for *constructive mathematics*, in which purely existential proofs and the law of the excluded middle are not accepted. This has only a superficial relationship to algorithmic mathematics. Of course, it often happens that one can obtain a good algorithm by just transcribing an essentially constructive proof, but such algorithms do not tend to be the most interesting ones; many of them are mentioned in §2. In the design and analysis of algorithms one gladly invokes all the help that existing pure mathematics has to offer and often some not-yet-existing mathematics as well.

PRELIMINARIES

Algorithms and complexity - It is assumed that the reader has an intuitive understanding of the notion of an *algorithm* as being a recipe that given one finite sequence of nonnegative integers, called the *input* data, produces another, called the *output.* Formally, an algorithm may be defined as a *Turing machine,* but for several of our results it is better to choose as our "machine model" an idealized computer that is more realistic with respect to its *running time*, which is another intuitively clear notion that we do not define. We refer to and the literature given there for a further discussion of these points.

The *length* of a finite sequence of nonnegative integers n_1, n_2, ..., n_t is defined to be $\sum_{i=1}^{t} \log(n_i+2)$. It must informally be thought of as proportional to the number of bits needed to spell out the n_t in binary. By analyzing the *complexity* of an algorithm we mean in this study finding a reasonably sharp upper bound for the running time of the algorithm expressed as a function of the length of the input data. This should, more precisely, be called *time* complexity, to distinguish it from *space* complexity. An algorithm is said to be *polynomial-time* or *good* if its running time is $(l+2)^{O(1)}$, where l is the length of the input. Studying the complexity of a *problem* means finding an algorithm for that problem of the smallest possible complexity. In the present study we consider the complexity analysis complete when a good algorithm for a problem has been found, and we will not be interested in the value of the 0-constant. Informally, a problem has a good algorithm if an instance of the problem is almost as easily *solved* as it is *formulated.*

Sometimes we will refer to a *probabilistic* algorithm, which is an algorithm that may use a random number generator for drawing random bits. One formalization of this is a *nondeterministic* Turing machine. Unless we use the word *probabilistic*, we do *not* allow the use of random number generators, and if we wish to emphasize this we talk of *deterministic* algorithms. In the case of a probabilistic algorithm, the running time and the output are not determined by the input alone, but both have, for each fixed value of the input data, a *distribution.* The *expected* running time of a probabilistic algorithm is the expectation of the running time for a given input. Studying the complexity of a probabilistic algorithm means finding an upper bound for the expected running time as a function of the length of the input. For a few convenient rules that can be used for this purpose we refer to. A probabilistic algorithm is called *good* if its expected running time is $(l+2)^{O(1)}$, where / is the length of the input.

Parallel algorithms have not yet played any role in algorithmic number theory, and they will not be considered here.

Many results in this study assert that "there exists" an algorithm with certain properties. In all cases, such an algorithm can actually be exhibited, at least in principle.

All 0-constants are absolute and effectively computable unless indicated otherwise.

Encoding data - As stated above, the input and the output of an algorithm consist of finite sequences of nonnegative integers. However, in the mathematical practice of thinking and writing about algorithms one prefers to work with mathematical concepts rather than with sequences of nonnegative integers that encode them in some manner. Thus, one likes to say that the input of an algorithm is given by an algebraic number field rather than by the sequence of coefficients of a polynomial that defines the field; and it is both shorter and clearer to say that one computes the kernel of a certain endomorphism of a vector space than that one determines a matrix of which the columns express a basis for that kernel in terms of a given basis of the vector space. To justify such a concise mode of expression we have to agree on a way of encoding entities such as number fields, vector spaces, and maps between them by means of finite sequences of nonnegative integers. That is one of the purposes of the remainder of this section. Sometimes there is one obvious way to do the encoding, but often there are several, in which case the question arises whether there is a good algorithm that passes from one encoding to another. When there is, we will usually not distinguish between the encodings, although for practical purposes they need not be equivalent.

We shall see that the subject of encoding mathematical entities suggests several basic questions, but we will not pursue these systematically. We shall not do much more than what will be needed in later sections.

Elementary arithmetic - By Z we denote the ring of integers. Adding a sign bit we can clearly use nonnegative integers to represent *all* integers. The traditional algorithms for addition and subtraction take time $O(l)$, where l is the length of the input. The ordinary algorithms for multiplication and division with remainder, as well as the Euclidean algorithm for the computation of greatest common divisors, have running time $O(l^2)$. With the help of more sophisticated methods this can be improved to $l^{1+o(1)}$ for $l \to \infty$.

An operation that is *not* known to be doable by means of a good algorithm is decomposing a positive integer into prime numbers , but there is a good probabilistic algorithm for the related problem of deciding whether a given integer is prime. No good algorithms are known for the problem of recognizing square free numbers and the problem of finding the largest square dividing a given positive integer, even when the word "good" is given a less formal meaning.

For some algorithms a prime number *p* is part of the input. In such a case, the prime is assumed to be encoded by it rather than that, for example, *n* stands for the *nth* prime. Since we know no good deterministic algorithm for recognizing primes, it is natural to ask what the algorithm does if *p* is not prime or at least not known to be prime. Some algorithms may discover that *p* is nonprime, either because a known property of primes is contradicted in the course of the computations, or because the algorithm spends more time than it should; such algorithms may be helpful as primarily tests. Other algorithms may even give a

nontrivial factor of p, which may make them applicable as integer factoring algorithms. For both types of algorithms, one can ask what can be deduced if the algorithm does appear to terminate successfully. Does this assist us in proving that *p* is prime? What do we know about the output when we do not assume that *p* is prime? An algorithm for which this question has not been answered satisfactorily is Schoofs algorithm for counting the number of points on an elliptic curve over a finite field.

Linear algebra - Let *F* be a field, and suppose that one has agreed upon an encoding of its elements, as is the case when *F* is the field Q of rational numbers or the field F_p for some prime number *p* . Giving a finite-dimensional vector space over *F* simply means giving a nonnegative integer *n* , which is the dimension of the vector space. This number *n* is to be given in *unary*, i.e., as a sequence 1, 1, ... , 1 of *n* ones, so that the length of the encoding is at least n. This is because almost any algorithm related to a vector space of dimension *n* takes time at least *n*. The elements of such a vector space are encoded as sequences of *n* elements of F. Homomor- phisms between vector spaces are encoded as matrices. A subspace of a vector space can be encoded as a sequence of elements that spans the subspace, or as a sequence of elements that forms a basis of the subspace, or as the kernel of a homomorphism from the vector space to another one. For all fields *F* that we shall consider the traditional algorithms from linear algebra, which are based on Gaussian elimination, are polynomial-time: algorithms that pass back and forth between different representations of subspaces, algorithms that decide inclusion and equality of subspaces, that form sums and intersections of sub- spaces, algorithms that construct quotient spaces, direct sums, and tensor products, algorithms for computing determinants and characteristic polynomials of endomorphisms, and algorithms that decide whether a given homomorphism is invertible and if so construct its inverse. The proofs are straightforward, the main problem being to find upper bounds for the sizes of the numbers that occur in the computations, for example when *F* = Q.

If one applies any of these algorithms to *F* = Z//pZ without knowing that *p* is prime, then one either finds a nontrivial divisor of *p* because some division by a nonzero element fails, or the algorithm performs successfully as if *F* were a field. In the latter case it is usually easy to interpret the output of the algorithm in terms of free Z/pZ-modules, thus avoiding the assumption that *p* be prime.

Rings - We use the convention that rings have unit elements, that a subring has the same unit element, and that ring homomorphisms preserve the unit element. The *characteristic* char of a ring *A* is the nonnegative integer that generates the kernel of the unique ring homomorphism $\mathbf{Z} \to A$. The group of units of a ring *A* is denoted by A^*. All rings in this study are supposed to be *commutative.*

Almost any ring that we need to encode in this study has an additive group that is either finitely generated or a finite-dimensional vector space over Q; for exceptions. Such a ring *A* is encoded by giving its underlying abelian group together with the multiplication map $A \otimes A \to A$. It is straightforward to decide in polynomial time whether the multiplication map satisfies the ring axioms.

Ideals are encoded as subgroups or, equivalently, as kernels of ring homomorphisms. There are good algorithms for computing the sum, product, and intersection of ideals, as well as the ideal $I : J = \{x \in A\colon xJ \subset I\}$ for given I and *J*, and the quotient ring of *A* modulo a given ideal.

A *polynomial* over a ring is always supposed to be given by means of a complete list of its coefficients, including the zero coefficients; thus we do not work with sparse polynomials of a very high degree.

Most *finite rings* that have been encountered in algorithmic number theory "try to be fields" in the sense that one is actually happy to find a zero-divisor in the ring. This applies to the way they occur in §4 and also to the application of finite rings in primality testing. Nevertheless, it seems of interest to study finite rings from an algorithmic point of view for their own sake. Testing whether a given finite ring is local can be done by a good probabilistic algorithm, but finding the localizations looks very difficult. Testing whether it is reduced or a principal ideal ring also looks very difficult, but there may be a good algorithm for deciding whether it is quasi-Frobenius. I do not know whether isomorphism can be tested in polynomial time. Many difficulties are already encountered for finite rings that are F^-algebras for some prime number p. Two finite etale Fp-algebras can be tested for isomorphism in polynomial time, but there is no known good deterministic algorithm for finding the isomorphism if it exists; if they are fields, there is, but the proof depends on ring theory.

Local fields - A ***local field*** is a locally compact, nondiscrete topological field. Such a field is topologically isomorphic to the field R of real numbers, or to the field C of complex numbers, or, for some prime number p, to a finite extension of the field Q_p of *p*-adic numbers, or, for some finite field *E*, to the field *E((t))* of formal Laurent series over *E*. A local field is uncountable, which implies that we have to be satisfied with specifying its elements only to a certain precision. The discussion below is limited to the case that the field is non-archimedean, i.e., not isomorphic to R or C.

The complexity theory of local fields has not been developed as systematically as one might expect on the basis of their importance in number theory. The first thing to do is to develop algorithms for factoring polynomials in one variable to a given precision; and §4 below. Here the incomplete solution of the corresponding problem over finite fields causes a

difficulty; we are forced to admit probabilistic algorithms, or to allow the running time to be $\sqrt{p}$ times polynomial time, where *p* denotes the characteristic

RINGS OF INTEGERS

In this section we consider the following problem and its complexity.

Problem-1 : Given an algebraic number field K, determine its ring of integers $\mathcal{O}$

Constructing an order in *K,* we see that this problem is equivalent to the following one.

Problem-2. Given an order *A* in a number field *K*, determine the ring of integers $\mathcal{O}$ of K.

Much of the literature on this problem assumes that the given order is an equation order $\mathbf{Z}[\alpha]$, and it is true that equation orders offer a few advantages in the initial stages of several algorithms. It may be that in many practical circumstances one never gets beyond these initial stages , but in the worst case—which is what we are concerned with when we estimate the complexity of a problem—these advantages quickly disappear as the algorithm proceeds. For this reason we make no special assumptions about *A* except that it is an order.

Most of what we have to say about Problem 2 also applies to the following more general problem.

Problem-3. Given a commutative ring *A* of which the additive group is isomorphic to $\mathbf{Z}^n$ for some *n* , and that has a no vanishing discriminant over Z, determine the maximal order in $A \otimes_{\mathbf{Z}} \mathbf{Q}$ The main result on Problem 1, which is due to Chistov, is a negative one.

Theorem-1. *Under deterministic polynomial time reductions, Problem* 1 *is equivalent to the problem of finding the largest square factor of a given positive integer.*

The problem of finding the largest square factor of a given positive integer *m* is easily reduced to Problem 1 by considering the number field $K = \mathbf{Q}(\sqrt{m})$ For the opposite reduction, which in computer science language is a "Turing" reduction, we refer to the discussion following Theorem 2 below.

Since there is no known algorithm for finding the largest square factor of a given integer *m* that is significantly faster than factoring *m*, Theorem 1 shows that Problem 1 is currently intractable. More seriously, even if someone *gives* us $\mathcal{O}$, we are not able to recognize it in polynomial time, even if probabilistic algorithms are allowed. Deciding whether the given order *A* in Problem 2 equals $\mathcal{O}$ is currently an infeasible problem, just as deciding whether a given positive integer is squarefree is infeasible. This is not just true in theory, it is also true in practice.

One possible conclusion is that $\mathcal{O}$ is not an object that one should want to work with in algorithms. It may very well be that whenever $\mathcal{O}$ is needed one can just as well work with an order *A* in *K*, and *assume* that *A* equals $\mathcal{O}$ until evidence to the contrary is obtained. This may happen, for example, when a certain nonzero ideal of *A* is found not to be invertible; in that case one can, in polynomial time, construct an order *A'* in *K* that strictly contains *A* and proceed with A' instead of *A* .

If it indeed turns out to be wise to avoid working with $\mathcal{O}$, then it is desirable that more attention be given to general orders, both algorithmically and theoretically. This is precisely what has happened in the case of quadratic fields.

The order *A* equals $\mathcal{O}$ if and only if all of its nonzero prime ideals p are nonsingular; here we call p nonsingular if the local ring $A_{\mathfrak{p}}$ is a discrete valuation ring, which is equivalent to $\dim_{A/\mathfrak{p}} \mathfrak{p}/\mathfrak{p}^2 = 1$. One may wonder, if it is intractable to find $\mathcal{O}$, can one at least find an order in *K* containing *A* of which the singularities are bounded in some manner? One result of this sort is given below in Theorem 3; it implies that given *A* , one can find an order *B* in *K* containing *A*such that all singularities $\mathfrak{p}$ of *B* are *plane* singularities, i.e., satisfy $\dim_{B/\mathfrak{p}} \mathfrak{p}/\mathfrak{p}^2 = 2$

The geometric terminology just used should remind us of a situation in which there does exist a good method for finding the largest square factor, namely, if we are dealing with polynomials in one variable over a field. Thus, Theorem 1 suggests that, for a finite field E, finding the integral closure of the polynomial ring *E[t]* in a given finite extension of *E(t)* is a tractable problem, and results of this nature have indeed been obtained. In geometric language, this means that it is feasible to resolve the singularities of a given irreducible algebraic curve over a given finite field. The corresponding problem over fields of characteristic zero has been considered as well , and one may wonder whether the geometric techniques that have been proposed can also be used in the context of Problem 2. In any case, we can formulate Problem 2 geometrically by asking for the resolution of the singularities of a given irreducible *arithmetic curve.*

Theorem 2. *There is a good algorithm that given K and A as above, as well as an integer* $q > 1$*, determines an order B in K that contains* $A^{(p)}$ *for each prime number p that divides q exactly once.*

To prove this, one first observes that it suffices to exhibit a good algorithm that given *K*, *A* and *q* either finds *B* as in the statement of the theorem, or finds a nontrivial factorization $q = q_1 q_2$. Namely, in the latter case one can precede recursively with *q*\ and q_2 to find orders B_1, B_2, and one lets *B* be the ring generated by B_x and B_2.

To find *B* or q_x, q_2, one applies the algorithm outlined above, with a few changes. The first change is that one starts by checking that *q* is not divisible by any prime number $p \leq n$; if it is, then either one finds a nontrivial splitting of *q*, or *q* is a small prime number and one can apply the earlier algorithm. So let it now be assumed that *q* has no prime factors $p \leq n$, and that $q > 1$ The second change is that one replaces, in the above algorithm, *p* and everywhere by *q* and $\mathbf{Z}/q\mathbf{Z}$. This affects the linear algebra routines, which are only designed to work for vector spaces over fields. However, they work just as well for modules over a ring $\mathbf{Z}/q\mathbf{Z}$, *until* some division in $\mathbf{Z}/q\mathbf{Z}$ fails, in which case one obtains a nontrivial factor q_x of *q*. The third change is that $\mathfrak{r}/q\mathbf{Z}$ should now be calculated as the "radical of the trace form," i.e., as the kernel of the Z/qZ-linear map $A/qA \to \mathrm{Hom}(A/qA, \mathbf{Z}/q\mathbf{Z})$ that sends x to the map sending *y* to Tr(jc*y)*, where $\mathrm{Tr}: A/qA \to \mathbf{Z}/q\mathbf{Z}$ is the trace map. If *q* is a prime number exceeding *n* then this is the same r as above.

One can show that the modified algorithm has the desired properties. This concludes our sketch of the proof of Theorem 2.

Using Theorem 2 we can complete the proof of Theorem 1. Namely, suppose that one has an algorithm that determines the largest square divisor of any given positive integer. Calling this algorithm a few times, one can determine the largest square free number *q* for which q^2 divides the discriminant of *A* . Applying the algorithm of Theorem 2 to *q* one obtains an order *B* that contains $A^{(p)}$ for each prime *p* for which p^2 divides the discriminant of *A* , so that $B = \mathcal{O}$

We now formulate a result that also gives information about the local structure of *B* at primes *p* for which p^2 divides *q* . Let *A* be an order in a number field *K*, and let *q* be a positive integer. We call *A nonsingular* at *q* if each prime ideal of *A* containing *q* is nonsingular. We call *A tame* at *q* if for each prime ideal $\mathfrak{p}$ of *A* containing *q* there exist an unramified extension *R* of the ring Z_p of *p*-adic integers, where $p = \mathrm{char}\, A/\mathfrak{p}$, a positive integer *e* that is not divisible by *p* , and a unit $u \in R^*$, such that there is an isomorphism $\varprojlim_m A/\mathfrak{p}^m \cong R[X]/(X^e - uq)R[X]$ of Zp-algebras. As a partial justification of the terminology, we remark that for prime *q* the order *A* is tame at *q* if and only if each prime ideal $\mathfrak{p}$ of *A* containing *q* is nonsingular and tamely ramified over *q* ; this follows from a well-known structure theorem for tamely ramified extensions of Z_q. If *A* is tame at *q* and p is a prime ideal of *A* containing *q*, then $\mathfrak{p}$ is nonsingular if and only if either $p = \mathrm{char}\, A/\mathfrak{p}$ divides *q* exactly once or the number *e* above equals 1, and otherwise $\mathfrak{p}$ is a plane singularity.

Theorem 3. *There is a good algorithm that, given an order A in a number field K of degree n, finds an order B in K containing A and a sequence of pairwise coprime divisors qi,* $1 \leq i \leq t$*, of the discriminant of B, such that*

1. *B is tame at* $q = \prod_{i=1}^{t} q_i$*;*
2. *All prime numbers dividing q exceed n;*
3. *B is nonsingular at all prime numbers p that does not divide q.*

This follows from a closer analysis of the algorithm of Theorem 2. Using this theorem and the properties of tameness, one can deduce the following result, which expresses that one can approximate $\mathcal{O}$ as closely as can be expected on the basis of Theorem 1.

CONCLUSION

In summary, using the tools gained via abstract algebra, we are able to take on a different field of mathematics, extending what the reader might already be familiar with known as elementary number theory. We created an abstract algebra analogue to a familiar theorem, and shed more light on its properties.

REFERENCES

Gabor Ivanyos, Marek Karpinski, Lajos Ronyai, and Nitin Saxena (2008). Trading GRH for algebra: algorithms for factoring polynomials and related structures. CoRR, abs/0811.3165

H.Cohen (1993). A course in computational algebraic number theory, Springer-Verlag, Berlin, MR 94i:11105

H.P.F. Swinnerton (2001). Dyer. *A Brief Guide to Algebraic Number Theory*. University Press of Cambridge.

Harper, M., and Murty, R., Euclidean rings of algebraic integers, *Canadian Journal of Mathematics*, **56**(1), (2004), pp. 71-76.

Ono Takashi (1990). An Introduction to Algebraic Number Theory. Plenum Publishing Corporation.

S. Lang (1964). Algebraic numbers, Addison-Wesley Publishing Co., Inc., Reading, Mass.-Palo Alto-London, MR 28 #3974

Serge Lang (2002). Algebra, revised 3rd ed. Springer-Verlag.

Steve Chien and Alistair Sinclair (2004). Algebras with Polynomial Identities and Computing the Determinant. In FOCS, pages 352-361.

Victor Shoup (2009). A Computational Introduction to Number Theory and Algebra. Cambridge University Press, New York, Available from http://shoup.net/ntb/. 19

W. Bosma, J. Cannon, and C. Playoust (1997). The Magma algebra system. I. The user language, J. Symbolic Comput. 24, no. 3-4, 235-265, Computational algebra and number theory (London, 1993). MR 1 484-478

Corresponding Author

Vijaysingh Digambar Gaikwad*

Assistant Professor, Dayanand Science College, Maharashtra

E-Mail – **vijaysinhgaikwad0@gmail.com**