A STUDY ON COMPUTER SCIENCE AND ITS RELATION TO ABSTRACT ALGEBRA

Rishabh Choudhary[1], Dr. Ajay Kumar Gupta[2]

[1] *Research Scholar, Department of Mathematics, Bhagwant University, Ajmer, India*
[2]*Associate Professor, Department of Mathematics, Bhagwant University, Ajmer, India*

ABSTRACT

In this we are studding on computer science and its relation to abstract algebra. Abstract algebra is applied in many fields of computer science and is a basic method of modeling problems. On one side you have machine learning, data science, and statistics, while on the other side you have rich computer graphics and animations. Most mathematical fields actually have uses / techniques of abstract algebra - differential equations, analytical and differential geometry, differential geometry, abstract algebra, mechanics, and more.

Keyword: - *Computer Science, Abstract Algebra, Relation, fields, Ring etc.*

1. INTRODUCTION

I was wondering if anyone could give a concrete example where abstract algebra and number theory appear in computer science. The only application I know of is cryptography. Do they have more areas? Exactly how will these courses help advanced computer science studies? The primary reason I ask is that I want to make sense of taking them as courses. While it is easy for computer science to repeat as much mathematics as possible. I think it is equally important to understand that beyond knowing what you can actually do with mathematics. I am working on my degree in Mathematics, and want to choose classes carefully to maximize my computer science ability. If they have very few applications of abstract algebra or number theory, it seems that it might be better to take a different course. Algebra is incredibly useful in computer science [1]. I will present my answer with my opinion: I see a good part of computer science as a branch of mathematics. So my answer will be quite broad. Note that my opinion is one that not everyone shares. Recall Kelly's theorem from group theory, which states that each group is a subgroup of some symmetry group. That is, each group is a permutation group. This immediately inspired the study of algebraic combinatorics. Group verbs are used to study the symmetries or automorphisms of mathematical objects. Informally, a group action is a dynamic process on an object, dividing its members into sets that we call classes. Significant combustible results are obtained from the study of the structure and volume of these orbitals. You must have seen the dihedral group, which is the automaker group of a cycle graph. In general, it is a non-trivial practice to locate the complete self-immune group of an object (such as a graph). There are many applications that come to mind. The first is Pola Enumeration Theory. Given a mathematical object (such as a graph), we color the vertices (not necessarily an appropriate color, in the sense that two adjacent vertices can receive the same color). Then looking at some subgroup H of automorphisms (such as the twisting of a necklace), how many different colors exist? That is, two colors are equal if they are h. Polya belong to the same class under the action of computation theory, so let's study these types of problems. If your combination is weak (and even if it is not), Lohar's bijective combination text is an excellent resource [2].

Automata Theory: A set of regular languages creates a Klein semi-ring on the operation of union (addition) and conception (multiplication). The notion of derivation also comes into play. See the Brozowski derivative. We can take advantage of the fact that a linear-algebraic algorithm for converting an FSM into a regular expression (see Brozozowski algebraic method). (I include it in my notes as well.) [3]

Complexity Theory: Babai has (or should have) a recent result on the graph isomorphism problem for algebra in CS. He uses several techniques such as group verbs and representation theory to study problems such as graph isomorphism and group isomorphism. Regarding the group Isomorphism problem, it is unspecified in the general case. For finite groups, the best boundle is something like $n^{\log(n)}$ (which is easy to obtain, simply by showing that

each finite group has a generating set with most $\log_2(n)$ generators). For Abelian groups, the group isomorphism has an O(n) time solution. For solvable groups, the $n^{\log(n)}$ bound has remained relatively untouched. Babai and Kiao presented a polynomial time symmetry test for groups with abelian silo towers. To the best of my knowledge (which is limited), it is the largest class of polynomial groups with a polynomial time isomorphism test.
Many of Babai's techniques are quite heavy. If these types of problems catch your interest, then I can spend some time on representation theory. I think a solid handle on group theory (rest with silo theorem) and abstract linear algebra, with inspiration, is enough to take a book into the field. There are several books on representation theory from a combustible point of view. Sagan's text on symmetrical grouping is a text that comes to mind. Babai also teaches classes on the algorithm in finite groups, which you can see on her home page. Computational group theory is an important phrase, if you decide to pay more attention to this area [4].

Cryptology, coding theory, and computational number theory: these are clear, and are fairly well covered.
Perhaps because theoretical computer science deals with the abstract, logical and mathematical aspects of computing, and in addition to mathematical logic, abstract algebra is the mathematical discipline that comes closest to CS at an advanced level? Anyway, I don't give too much credence to the label. As far as I know, some of the best (pure) mathematics students in my class consider theoretical physics as a branch of pure mathematics. I'm sure there are computer scientists who share a similar line of thought, although I can't say much about it [5].

2. MODERN ALGEBRA

The 19th and early 20th centuries saw tremendous changes in the method of mathematics. Abstract algebra emerged in the early 20th century under the name Modern Algebra. Its study was part of the drive for greater intellectual rigor in mathematics. Initially, the assumptions in classical algebra, on which the whole of mathematics (and the major parts of natural sciences) depend, took the form of axiomatic systems. No longer satisfied with the establishment of the properties of solid objects, mathematicians began to focus on general theory. Formal definitions of some algebraic structures began to emerge in the 19th century. For example, the results of different groups of permutations were seen as examples of general theorems that concern a general notion of an abstract group. The question of the structure and classification of various mathematical objects came to the forefront [6].
All these processes took place during mathematics, but were especially pronounced in algebra. Formal definitions through primitive operations and axioms were proposed for many basic algebraic structures, such as clusters, rings, and spheres. Therefore things like group theory and ring theory take their place in pure mathematics. Algebraic investigation of common fields by Ernst Steinitz and then the construction of common rings by David Hilbert, Emil Artin, and Amy Knuther, on the work of Ernst Coomer, Leopo Kronecker, and Richard Dedekind, who considered ideals in commutative rings, and the theory of representations of groups. In the subject, George Frobenius and Issai Shur came to define abstract algebra. These developments from the last quarter of the 19th century and the first quarter of the 20th century were systematically revealed in Bartel van der Varaden's Modern Algebra, a two-volume monograph published in 1930–1931, which changed the mathematical world forever. Were algebra from the theory of equations to the theory of algebraic structures [7].

3. APPLICATIONS

Because of its prevalence, abstract algebra is used in many fields of mathematics and science. For example, algebraic topology uses algebraic objects to study topology. The Poincaré conjecture, proved in 2003, states that the basic group of manifolds, which gives information about the linkage, can be used to determine whether the manifold is a sphere. Algebraic number theory studies various number rings that normalize sets of integers. Using the tools of algebraic number theory, Andrew Wills proved the final theorem of the format.
In physics, groups are used to represent symmetry operations, and the use of group theory can simplify differential equations. In gauge theory, the requirement of local symmetry can be used to reduce the equations that describe a system. The groups that describe those analogies are the Lai group, and the study of the Lai group and the Lai algebra reveal much about the physical system; For example, in a theory the number of force carriers is equal to the amplitude of Lie algebra, and these bosons interact with force if Le algebra is nonbellion [8].

4. ABSTRACT ALGEBRA AND PROGRAMMING

For example, group theory is very important in cryptography, especially in asymmetric encryption schemes such as RSA and El Gamal, finite groups. They use finite groups based on multiplication of integers. However, there are other, less obvious types of groups that apply in cryptography, such as elliptic curves.

Another application of group theory, or, to be a more specific, finite field, is checksum. The widely used checksum mechanism is based on modular arithmetic in the polynomial ring of the CRC finite field GF (2).

A more abstract application of group theory is in functional programming. In fact, all these applications exist in any programming language, but functional programming languages, especially Haskell and Scala (z), make it into algebraic structures such as monoids, groups, rings, fields, vector spaces, and so on. Let's embrace it. . The advantage is that, of course, functions and algorithms can be specified at a very general, high level [9].

On a meta level, I would also say that an understanding of basic mathematics such as this is essential for any computer scientist (not so much for a computer programmer, but for a computer scientist - of course), because it can change your way of thinking. Shapes the way. Required for more advanced mathematics. If you want to do 3D graphics stuff or program an industry robot, you'll need linear algebra, and for linear algebra, you should know at least some abstract algebra [10].

5. RESULTS AND ANALYSIS

5.1 Using Computer Algebra Systems in Teaching Algebra

The first category includes activities to guide students to convert patterns into equations. In most cases students can achieve well-known algebraic principles or rules by doing these types of activities. Algebraic structures can be discovered by students even when they lack the proper definitions, and this is just a learning outcome of this practice. Another category of computer applications promotes a deeper understanding of a concept by providing many examples. GAP enables students to examine groups and other complex algebraic structures. In addition, GAP arouses students' interest and curiosity and can be used as a means to motivate and engage them in the classroom (Rainbolt, 2002). ISETL (for DOS) or ISETLW (for Windows) is another type of computer software that is commonly available, and is due to Java version 2010 (Kirschmeyer, 2010). The software gives students the opportunity to set logical positions on elements, form a group, and then explore algebraic properties such as order, coset, quotation, homomorphism, and isomorphism. It provides students with an active learning environment similar to GAP [11].

Kepelman and Webb (2002) reported that FGB can be used to drill and practice with concrete examples of groups with novice learners. The program runs on a library of all finite groups up to isomorphisms of order 16 or less and all non-Abelian groups of order 40 or less. Tables for groups can be saved as text files with a specific structure and users can keep notes on the attributes they search for. Each search is then stored at a different location in the file with the group information. Multiple groups can be studied at the same time for comparison (FGB, 2010). The FGB program has seven tabs for each group analyzed [11]:

1) Axiom check: This allows validation of axioms for any given group table: symmetry, existence of an identity, and inverse [11].

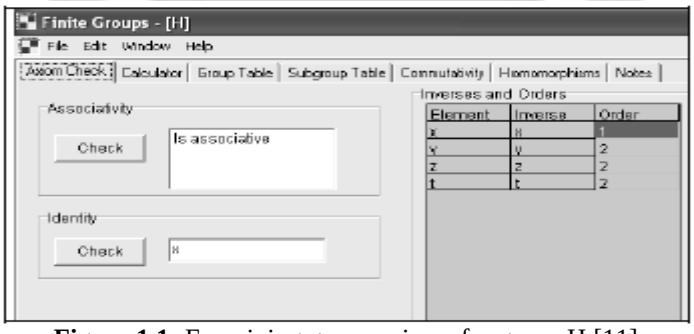


Figure 1.1: Examining group axioms for group H [11]

2) Calculator: It is used for calculating powers and inversions for elements as well as complex calculations involving multiple elements.

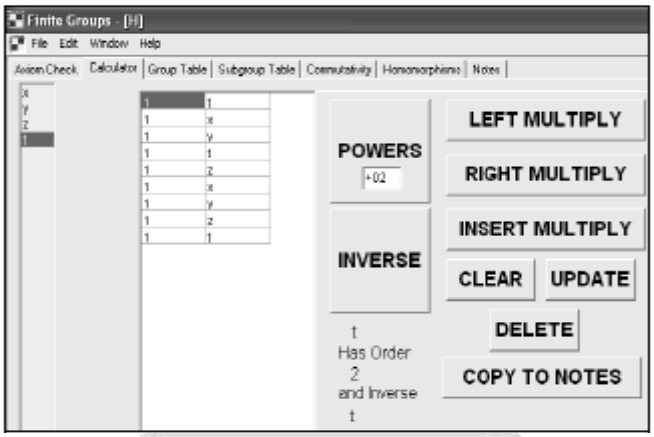


Figure 1.2: Calculating the strengths and inversions of elements in group H[11]

3) Group Table: This tab shows the Kelly ™ table for the group. It also enables elements to be reordered or renamed[11].

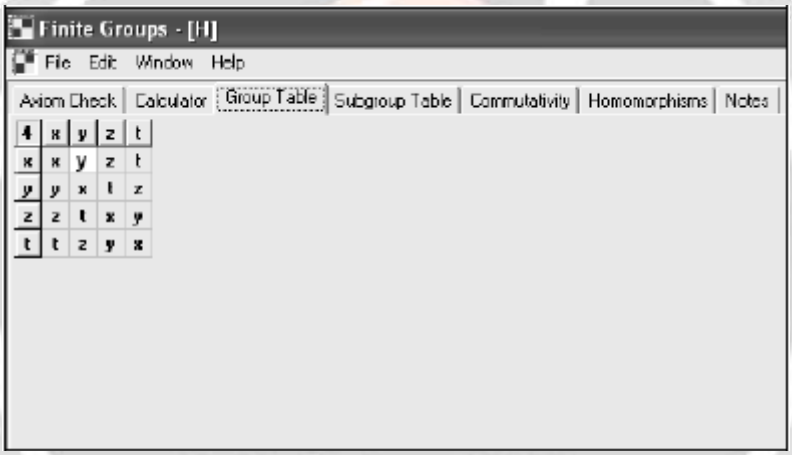


Figure 1.3: Group table for H[11]

4) Subgroup table: This tab is for creating an open group subgroup. Cosets for groups can be viewed in a special colored framework from which one can easily see whether the resulting collection of caskets forms a group. Subgroups can be conjugated by arbitrary elements and both group groups and subgroups can be exported to create their own group files. Notes can be made to record information about a subgroup and can be used for later reference.

Figure 1.4: Calculating the cosets of group H[11]

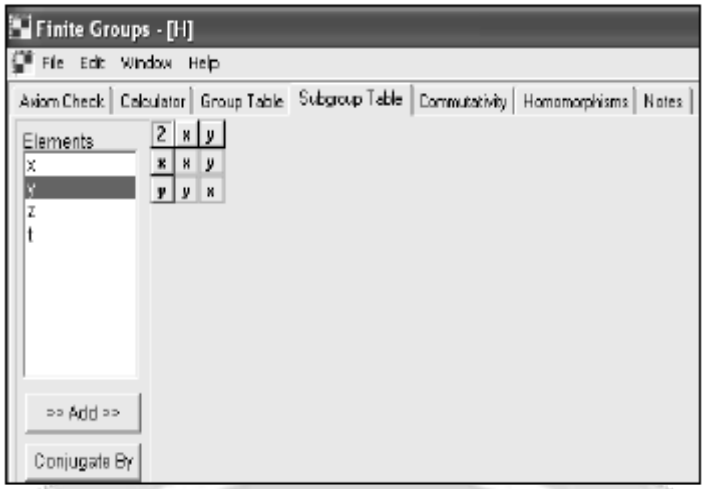


Figure 1.5: Finding a subgroup of group H[11]

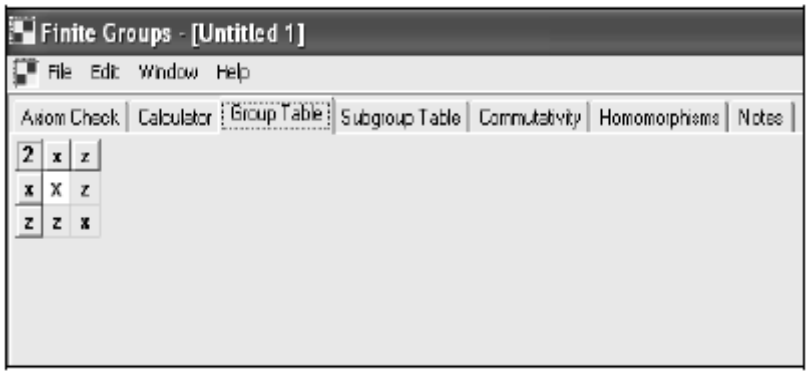


Figure 1.6: Determining a quotient group of group H[11]

5) Commutivity: Using this tab, the center of the group and the center of the elements can be easily calculated[11].

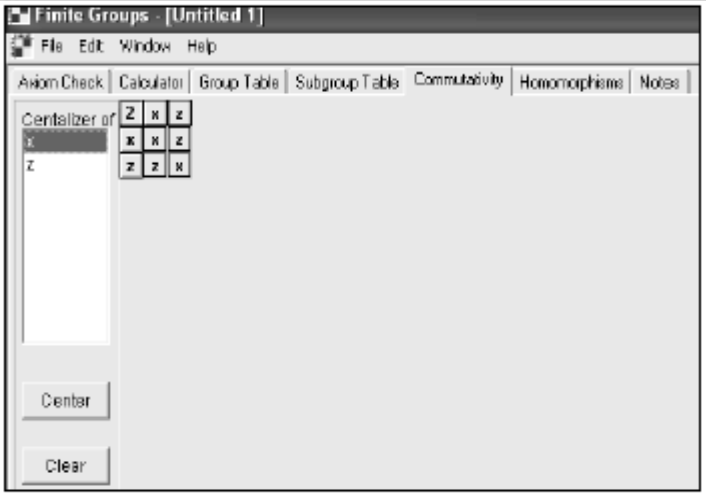


Figure 1.7: Calculating the center and centralizers of group H[11]

6) Homomorphisms: This tab allows one to construct homomorphisms between two group files.

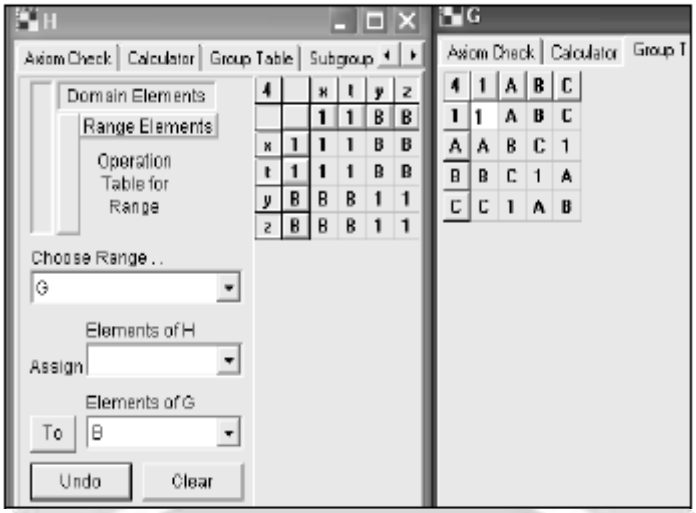


Figure 1.8: Constructing a homomorphism between groups H and G[11]

5.2 Results & Analysis

The characteristics of a group can be recorded here. The note also contains explanatory information about groups when it is appropriate which includes an explanation of marking for descriptive elements.

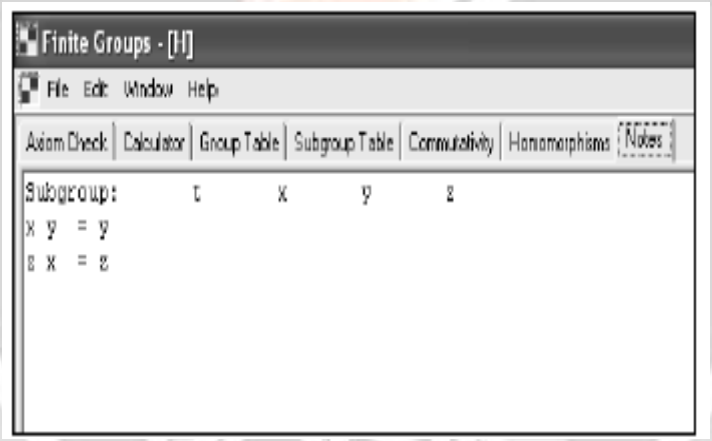
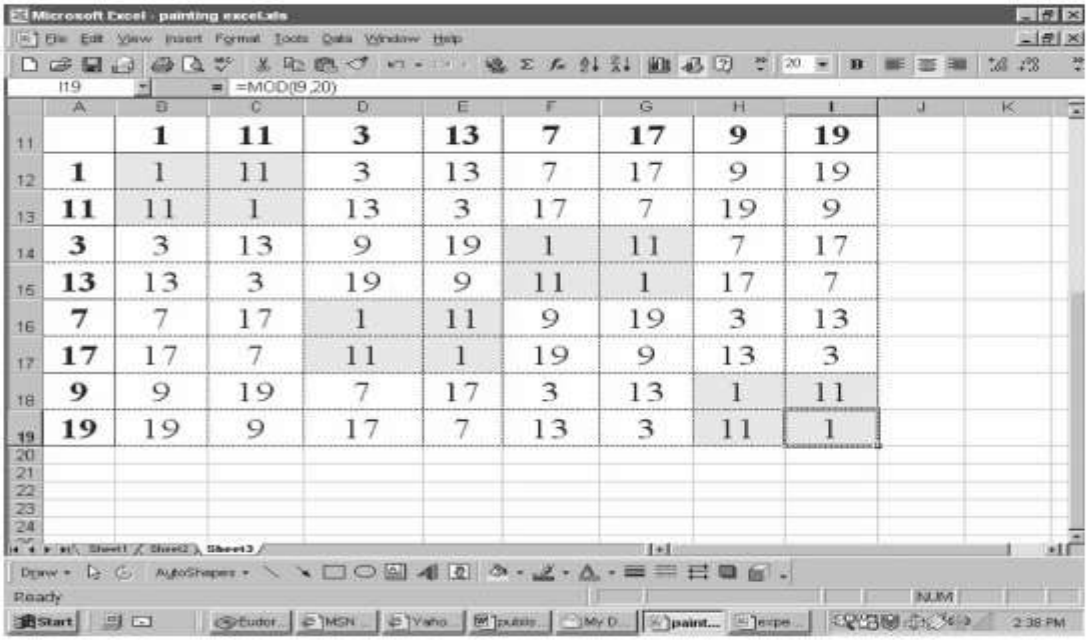


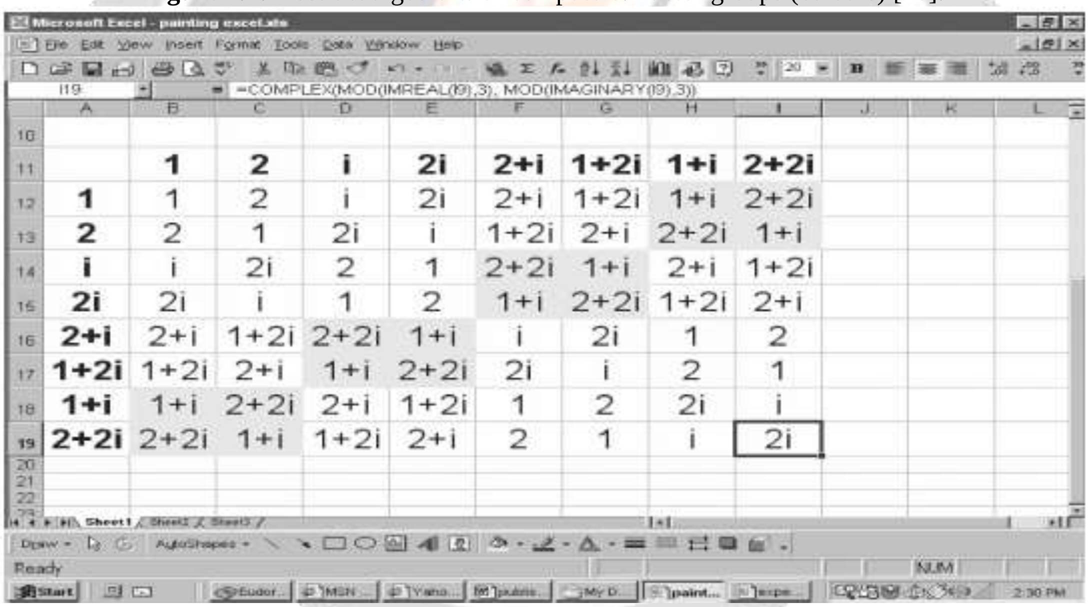
Figure 1.9: Storing exploratory and descriptive information about group H[11]

The activity here reflects the concepts of causal groups. Without help, students were able to construct the table for group U (20) and, after proper arrangement, formed a causal group. The details of the computational approach are omitted here because it is similar to the first example. Figure 1. 1 shows the final table of the factor group G / U where G (U (20) and U¼ {1, 11}, a subgroup of order 2. The set of parentheses of U in G {U, 3U, 7U , Are. 9U}, and each block in the table represents a coset; considering each block as an element, the four parentheses form a group, which students color as follows: yellow for coset, u ; Red for 3U; Blue for 7U; Green for 9U. Students then factor It was encouraged to think of the muh, which included the four colors that define the Kelly table constructed here. This example gave the student group G / U, namely (gU) (hU) forgh (U) Helped students to think of an operation for all g, g in g. This example also helped students understand the proof of the first isomorphism. Theorem in later lectures [10].



	1	11	3	13	7	17	9	19
1	1	11	3	13	7	17	9	19
11	11	1	13	3	17	7	19	9
3	3	13	9	19	1	11	7	17
13	13	3	19	9	11	1	17	7
7	7	17	1	11	9	19	3	13
17	17	7	11	1	19	9	13	3
9	9	19	7	17	3	13	1	11
19	19	9	17	7	13	3	11	1

Figure 1.1: Abstract algebra with computers: Factor groups (colours) [10].



	1	2	i	2i	2+i	1+2i	1+i	2+2i
1	1	2	i	2i	2+i	1+2i	1+i	2+2i
2	2	1	2i	i	1+2i	2+i	2+2i	1+i
i	i	2i	2	1	2+2i	1+i	2+i	1+2i
2i	2i	i	1	2	1+i	2+2i	1+2i	2+i
2+i	2+i	1+2i	2+2i	1+i	i	2i	1	2
1+2i	1+2i	2+i	1+i	2+2i	2i	i	2	1
1+i	1+i	2+2i	2+i	1+2i	1	2	2i	i
2+2i	2+2i	1+i	1+2i	2+i	2	1	i	2i

Figure 1.2: Cosets of subgroup {1, 2} in Z3 [i] [10].

6. CONCLUSIONS

The concepts of mathematics, particularly abstract algebra, are often considered difficult to understand, because the concepts are abstract. They need to be agreed upon by examples to easily conceptualize. Most abstract concepts taught in abstract algebra courses can be incorporated through mathematical software. We hope that this article has contributed to the challenge in computer science that we have mentioned in the results. We also hope that there will be a case that current achievements in teaching sequential programming can be extended to concurrent programming. Computational evidence is commonly used in the functional programming community to demonstrate algorithm accuracy. Studies indicate that the use of computer-aided instruction methods can clearly contribute to students' understanding of conceptual relationships, reasoning, and mathematical proofs. These computer programs

provide many benefits in learning abstract algebra concepts. For example, some software programs such as Matlab ™, GAP, ISETL and FGB are useful for visually defining group relationships, calculating the order of elements, and showing the elements of subsets. Students using these software packages also have the chance to investigate non-commutative group types. Students can also set logical positions on the elements to form groups and then explore algebraic properties i.e. order, coset, quotation, symmetry and symmetry. Using computer programs can create an active environment and opportunities for students to learn what they have learned.

7. REFERENCES

[1]. Allenby, R. B. J. T. (1991), Rings, Fields and Groups, Butterworth-Heinemann, ISBN 978-0-340-54440-2
[2]. Artin, Michael (1991), Algebra, Prentice Hall, ISBN 978-0-89871-510-1
[3]. Burris, Stanley N.; Sankappanavar, H. P. (1999) [1981], A Course in Universal Algebra
[4]. Gilbert, Jimmie; Gilbert, Linda (2005), Elements of Modern Algebra, Thomson Brooks/Cole, ISBN 978-0-534-40264-8
[5]. Lang, Serge (2002), Algebra, Graduate Texts in Mathematics, 211 (Revised third ed.), New York: Springer-Verlag, ISBN 978-0-387-95385-4, MR 1878556
[6]. Sethuraman, B. A. (1996), Rings, Fields, Vector Spaces, and Group Theory: An Introduction to Abstract Algebra via Geometric Constructibility, Berlin, New York: Springer-Verlag, ISBN 978-0-387-94848-5
[7]. Whitehead, C. (2002), Guide to Abstract Algebra (2nd ed.), Houndmills: Palgrave, ISBN 978-0-333-79447-0
[8]. W. Keith Nicholson (2012) Introduction to Abstract Algebra, 4th edition, John Wiley & Sons ISBN 978-1-118-13535-8 .
[9]. John R. Durbin (1992) Modern Algebra : an introduction, John Wiley & Sons
[10]. Kenneth K. Nwabueze, Computers in abstract algebra, This article was downloaded by: [Florida State University] On: 31 December 2014, At: 21:46 Publisher: Taylor & Francis Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, U
[11]. Muzaffer OKUR, Ramazan DİKİCİ, Vehbi Aytekin SANALAN, Enver TATAR, Computer Applications in Teaching Abstract Algebra, www.ijastnet.com, International Journal of Applied Science and Technology Vol. 1 No.1; March 2011