A novel approach for Problem Solving using Computational Thinking

Vartika Sharma
Department of CSE
Assistant Professor,GSSSIETW, Mysuru, Karnataka, India

Abstract*-If information and communication technologies (ICT) are to bring about a transformational change to a sustainable society, then we need to transform our thinking. Computer professionals already have a conceptual toolkit for problem solving, sometimes known as computational thinking. However, computational thinking tends to see the world in terms of a series of problems (or problem types) that have computational solutions (or solution types). Sustainability, on the other hand, demands a more systemic approach, to avoid technological solutions, and to acknowledge that technology, human behavior and environmental impacts are tightly inter-related. This paper concerns training about computational thinking in discrete mathematics teaching. Firstly, four main components of computational thinking are given, which are abstract thinking, logical thinking, modeling thinking and constructive thinking. Secondly, some content of discrete mathematics, which have close relationship with computational thinking, are described by corresponding application example. Finally, we give a mapping from knowledge unit of discrete mathematics onto corresponding of details of computational thinking.*

Keywords*-*Problem solving; computational thinking; discrete mathematics teaching; knowledge unit; constructive thinking; mapping

I. INTRODUCTION

Discrete mathematics is a branch of applied mathematics that deals with arrangements of discrete objects which are separated from each other, such as integers, real numbers, propositions, sets, relations, functions and graph [1]. It has many applications in computer science and software engineering, such as how to search useful information by searching engineer (Google, Baidu), how to describe a static structure and dynamic behavior of software system, and how to verify a software specification by logic statements, etc. In order to teach discrete mathematics successfully for teacher, and specifically for student majoring in the department of computer, we put forward an idea that introducing "computational thinking" into discrete mathematics teaching.

Getting computers to help us to solve problems is a two-step process: First, we think about the steps needed to solve a problem. Then, we use our technical skills to get the computer working on the problem. Take something as simple as using a calculator to solve a word problem in maths. First, you have to understand and interpret the problem before the calculator can help out with the arithmetic bit. Similarly, if you are going to make an animation, you need to start by planning the story and how you'll shoot it before you can use computer hardware and software to help you get the work done. In both of these examples, the thinking that is undertaken before starting work on a computer is known as computational thinking. Computational thinking describes the processes and approaches we draw on when thinking about problems or systems in such a way that a computer can help us with these. Computational thinking is not thinking about computers or like computers. Computers don't think for themselves. Computational thinking is about looking at a problem in a way that a computer can help us to solve it.

The term "computational thinking" has been coined to describe how to think like computer scientist [2], it was advocated by M.Wing, a Carnegie Mellon University professor. Computational thinking has become a fundamental skill, ranking alongside reading, writing and arithmetic, it can be found on all the subjects [3]. "Computational Thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science." In the case of systems biology, it means the ability to pull together the multiple abstractions that molecular biology has accumulated. In computer science, it can help people understand and construct computer system to solve given problem[10].

In this paper, we introduce "computational thinking" into discrete mathematics teaching. Section 1 introduces the application of discrete mathematics and computational thinking briefly. Section 2 illustrates contents of computational thinking in detail, and gives some definition by discrete mathematics. In section 3, we give contents of discrete mathematical designed by ACM/IEEE Computing Curricula 2005. In the final section, some mapping from DM (Discrete Mathematics) to CT (Computational Thinking) is given.

II. CONCEPTS OF COMPUTATIONAL THINKING

A. Logical Thinking

Logical thinking is the process in which one uses reasoning consistency to come to a conclusion. Some computer problems or computer states (situation) that involving logical thinking always call for mathematics structure, for relationships between some hypotheses and given statements, and for sequence of reasoning that make some conclusion more reasonable.

The core and basis of all logical thinking is sequential thought which arrange a serial of statements by a chain, the frontal element represents earlier conclusion, sequential thought process involves taking some statements in a chain-like progression that takes on a meaning in and of it. To think logically is to construct some approaches step by step.

It has been proved that logical training of discrete mathematics can make computer student smarter and more meticulous. A student who has logical thinking capability always rejects quick answer some computer problem, such as “it is too difficult”, or “I don’t know”. On contrary, he will apply logical thinking to delve into proposed problem and understand better the method and arrive at a solution. Logical thinking is not a magical process or a matter of genetic endowment, but a learned mental process which is taught in discrete mathematics.

B. Abstract Thinking

In order to understand main body of computer problem, abstract thinking is essential in computer science and technology. In solving an interesting problem, abstraction thinking is one very general purpose heuristic that can help to attack this problem. Informally, abstraction thinking can be thought of the mapping from a ground representation to a new but simpler representation [4]. Abstract representation is simpler because the mapping usually throws away details but preserves certain desirable properties, and translates the old problem into a new problem which can be solved by our knowledge.

Definition 1 (Formal system): A formal system ∑ is a triple <L, Ω, Δ >, where L is the specified domain language, Ω is the set of axioms about rule used and Δ is the deductive machinery of ∑.

Usually, a language is defined by the alphabet, the set of well formed terms which can construct domain language, and the set of well formed formulae. The axioms are the basic well formed formulae, that is to say, Ω □ L. deductive machinery is the set of inference rules which can induce new theorems from existing ones.

Definition 2 (Abstract): An abstract is a triple, A = < ∑1, ∑2,f>, where ∑1 and ∑2 are formal system ,and f is a function which maps from the language of ∑1 onto that of ∑2.

Definition 3 (Abstract thinking): Abstract thinking is a thinking method which solves a new problem by abstract, it always translate source problem l1□∑1 into a target problem l2 = f(l1)□∑2, then use some axioms and deductive machinery of ∑2 to solve l2.

C. Modeling Thinking

Modeling thinking, in the technical use of the term, refers to the translation of objects or phenomena from the real world into mathematical equations, and/or computer relations. It is choosing an appropriate representation or modeling the relevant aspects of a problem to make it tractable. Computer modeling is the representation of reality objects on a computer. A problem which will be solved by computer must be modeled by corresponding software model.

Computer modeling is a mathematical and computer method for solving real world problems. By virtue of modeling thinking, students can study a problem-solving process. Also, they can learn how to identify a problem, construct or select appropriate models, figure out what data needs to be collected, test the validity of a model, calculate solutions and implement the model. In order to promote student creativity and demonstrate the link between computer theoretical and real world applications, we must place great emphasis on model construction. Computer models can provide explanation facilities that support multiple perspectives, ranging from the conceptual level, to logical level and finally to physical level, all these levels can slake anyone's thirst for knowledge.

D. Constructive Thinking

Target of theory is to practice in reality. How to solve some problems (such as performing arithmetic, organizing data, graphically displaying information, playing chess with other player) is an important skill. Constructive thinking can help us solve these problems by algorithm and program, many interesting and useful programs require greater effort, and do some algorithm exercises in discrete mathematics[11].

Computers work with algorithms. An algorithm is a step- by-step unambiguous mechanical procedure requiring no insight or ingenuity to perform. It just likes cookbook recipes, devising a good recipe can be a difficult and creative task, but following a recipe should be straightforward and routine. Performing these operations has innovative thinking.

Definition 4 (Constructive thinking): Constructive thinking is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output. Informally:

Constructive thinking: = <Q, I, Ω, F >, where Q is a set of computing states, both I and Ω are subset of Q. Among which, I stands for input set of computing, and Ω stands for output set of computing. F is computing rule, namely F is a mapping function. By using F function, we can define an order which is arranged as following: x0, x1, x2,… xk, where xk = F(xk-1). Order "x0, x1, x2,… xk" represents the constructed computing steps beginning with step x0, ending with xk.

As a carrier of constructive thinking, an algorithm or program is thus a sequence of computational steps that transform the input into the output. We can also view an algorithm as a tool for solving a well-specified computational problem. The algorithm describes a specific computational procedure for achieving that input/output relationship.

III. CONTENT OF DISCRETE MATHEMATICS

Discrete mathematics has applications to all fields of computer science, it is used extensively in telecommunications and information processing. In DM, we are concerned with objects such as integers, propositions, sets, relations and functions which are all discrete. We learn concepts associated with them, properties and relationships among them. DM includes sets, functions and relations, matrix algebra, combinatory and finite probability, graph theory, finite differences and recurrence relations, logic, mathematical induction, and algorithmic thinking [7, 8]. Because of this diversity of topics, it is perhaps preferable to study all these content of discrete mathematics, but, discrete mathematics has a minimal set which is a necessary condition to grasp computational thinking.

In the late 1990s a Joint ACM/IEEE Task Force was formed to revise the undergraduate computing curricula. They report six topics as the knowledge base for discrete structures—(DS1) functions, relations and sets, (DS2) basic logic, (DS3) proof techniques, (DS4) basics of counting, (DS5) graphs and trees, and (DS6) discrete probability. They came to the conclusion that the DM material should be taught with examples and applications from computer science [1], because the applications would enhance the understanding of DM.

Mathematical Logic

Logic is a language for reasoning for some assertion. It is a collection of rules which can be used when doing logical reasoning. Human reasoning has been observed over centuries from at least the times of Greeks, and patterns appearing in reasoning have been extracted and abstracted. The foundation of the logic was laid down by a British mathematician Boole in the middle of the 19th century. Mathematical logic is interested in true or false of statements, and how the truth/falsehood of a statement can be determined from other statements. We use symbols to represent arbitrary statements so that the results can be used in many similar but different situations, so logic can promote the clarity of thought and eliminate ambiguity and mistakes.

There are various types of logic such as logic of sentences (propositional logic), logic of objects (predicate logic), uncertainties logic, fuzziness logic, modal logic, and temporal logic etc. But in DM course, we are only concern with propositional logic and predicate logic which are fundamental to other logic.

Example 1 (logical as boolean searching): In propositional logic, there are many connectives (¬, □, □) which are used extensively in searching of information of webpage on internet. Considering Google search engines, it supports boolean searching technique, which usually can help find web pages about particular subject. In Google, "+" or bank stands for logical connectives "□", "-" stands for logical connectives ¬, "OR" stands for logical connectives□. If we input sentence "computational thinking " in Google, then all the webpages about computational thinking are searched; If we input sentence "computational - thinking " in Google, then all the webpages which conclude "computational" but no "thinking" are searched, so searching result becomes little.

Set Theory

The concept of set is fundamental to computer science. For example, relationships between two objects are represented as a set of ordered pairs of objects, the concept of ordered pair is defined using sets; natural numbers, which are the basis of other numbers, are also defined using sets; the concept of function, being a special type of relation, is based on sets, and graphs and digraphs consisting of lines and points are described as an ordered pair of sets.

The relation is a special set which consist of two-tuples, it is an abstraction of relations we see in our everyday life such as those between parent and child, address and telephone number, main calling function and sub called function, etc. In set theory, we focus our attention on properties of those relations, such as reflexivity, irreflexivity, symmetry, anti symmetry and transitivity.

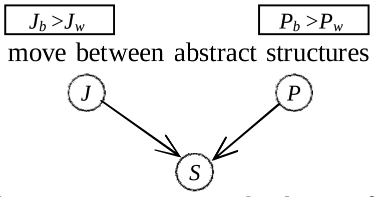
A function is something that associates each element of a set with an element of another set. It appears quite often even in no technical contexts. For example, a social security number uniquely identifies the person; the income tax rate varies depending on the income, and so on. As you might have noticed, a function is quite like a relation, but one element in function doesn’t map onto many elements.

Example 2 (relation closure application in mobile telephone): Mobile telephone network has data centers in Beijing, Shanghai, Taiyuan, Yantai and Nanjing. There are direct, one-way optical cables from Beijing to Shanghai, from Beijing to Taiyuan, from Shanghai to Yantai, and from Taiyuan to Naning. We can model this situation by relation. Let R be the relation, <a, b>$\in$R if there is an optical cables from the data center a to that in b, How can we guarantee there is some link composed of one or more optical cables from one city to another? Although R cannot be used directly to answer this, however, we can find all pairs of data center that have a link by constructing transitive closure of R.

Abstract Algebra

Abstract algebra is a typical application of abstract thinking. Mathematicians work with and delight in bizarre numbers such as i (the square root of -1) which seem likes an irrational mix of mysticism. Nonetheless, the basic operations in arithmetic (addition, subtraction, multiplication, division etc.) apply remarkably well to many disparate real-life situations, such as balancing your checkbook, keeping score in a game of cards, or measuring ingredients for a recipe. In all these situations, we use add and multiply operator to model the situation perfectly.

In abstract algebra, we will quickly move to a more general point of view. We will often move back and forth between the general abstract concepts and specific concrete examples; this move between abstract structures and concrete



instances of these structures is at the heart of the abstract algebra. Abstraction means that we often feel that we are proceeding without intuition, guided by bare logic, so proof becomes crucial. Rather, after studying a few examples carefully we will move to the general case. There are two reasons for this: first, doing otherwise would take too much time; second, the abstract point of view is considerably easier than the concrete.

Example 3 (software is a abstract algebra): Software is a algebra structure SW = <Language, Strcat, Strcomp, Strcpy,…>, where Language is a set of strings which can been processed by software, Strcat, Strcomp, and Strcpy are string operators. For example, Strcmp can compare two strings whether they are equal. In nature, any software is a machine which can process and translate string; in addition, it is an algebra structure whose operation object is string.

Graph Theory

A graph G consists of two disjoint sets V (vertices) and E (edges), and an incidence relation which associates a pair of vertices with each edge. It has many applications in various disciplines such as Biology (e.g. phylogenic trees), Computer Science (e.g. checking program deadlock, modeling of the Internet), Economic (e.g. social networks), Engineering (e.g. computer networks), and other branches of sports (modeling of tournaments).

Using graph theory, we will explore along the way some of its numerous applications, especially in computer science. These include critical path analysis, graph coloring problems, minimal spanning trees, and bin-packing techniques. There are two important teaching goal of graph theory: teaching students to write complete and concise proofs, and understanding application of graph theory in computer and software engineering.

Example 4 (CP-net application) Extracting preference information from users is generally an arduous process, and human decision analysts have developed sophisticated techniques to help elicit this information, CP-net is an important and useful graph model to represent preference [9].

Definition 5 A CP-net over variables V = {X1, X2, …, Xn } is a directed graph G over X1, X2, …,Xn whose nodes are annotated with conditional preference tables CPT(Xi) for each Xi $\in$V. Each conditional preference table CPT(Xi) associates a total order > with each instantiation u of Xi's parents Pa(Xi) = U.

$J_b \wedge P_b$	$S_r > S_w$
$J_w \wedge P_b$	$S_w > S_r$
$J_b \wedge P_w$	$S_w > S_r$
$J_w \wedge P_w$	$S_r > S_w$

Figure 1. CP-Net for “my evening dress": Jacket, Pants and Shirt

Fig. 1 illustrates a CP-net that expresses my preferences for evening dress. It consists of three variables J, P, and S, standing for the jacket, pants, and shirt, respectively. I unconditionally prefer black to white as a color for both the jacket and the pants, while my preference between the red and white shirts is conditioned on the combination of jacket and pants: if they have the same color, then I prefer a red shirt. Otherwise, if the jacket and the pants are of different colors, then I prefer a white shirt.

IV. MAPPING FROM DISCRETE MATHEMATICS TO COMPUTAIONAL THINKING

As stated above, computational thinking has some kinds of thinking and discrete mathematics studies discrete object and their relations. In order to understand the content of discrete mathematics, and grasp the idea of computational thinking in discrete mathematics, we give the relationship between computational things and discrete mathematics.

Abstract by algebra structure and graph

Many of knowledge units of discrete mathematics can map onto abstract thinking. Formal proposition of mathematical logic is an abstract which can describe some statement about reality world or computer sciences. Such as “TCP protocol is a protocol of internet”, it tells us that in internet technology, TCP is an important protocol; In addition, set, relation, function, can been seen as an abstraction of some discrete objects which are studied by computer scientist; In Unified Modeling Language of software engineering, they are many diagrams (such as class diagram, object diagram, state diagram…) which are abstracted of software components.

Logical thinking by computer logic

Logical thinking runs through all the teaching of discrete mathematics, among which, mathematical logical lay a foundation of reason. In set theory, algebra structure and graph theory, a lot of proof of theorem can been seen as logical thinking. In the logical unit, we outline the basic structure of and give examples of each proof technique. By logical thinking, we can discuss which type of proof is best for a given problem. In addition, we can relate the ideas of mathematical induction to recursion and recursively defined structures.

Modeling thinking by set theory and relation

Discrete mathematics has applications to almost every conceivable area of computer science. Modeling with discrete mathematics is an extremely important problem-solving skill, which give ability to develop some programs to solve problems of computer by constructive thinking. Modeling tools are: proposition, set, permutation, relation, graph, tree, finite state machine, operator, and algebra structure (such as group, ring, boolean algebra).

Constructive thinking by algorithm and proving

In discrete mathematics course, there are many knowledge units about theorem proof and algorithm construction. Grasping of proof nature, direct and indirect proofs, proof by contradiction, counterexamples, existence and constructive proofs can tell us that construction thinking is an important computational approach. Moreover, math induction (weak, strong, structural), well-ordering principle, standard searching and sorting algorithms, algorithm correctness arguments, recursive definitions, iterative and recursive algorithms are some concrete constructive approaches .

V.CONCLUSION

Future mathematics teaching has not only to focus on concepts and teaching techniques of computing, but also on problem solving and problem posing to reach general aims like creativity, ability of systematization, abilities of communication, argumentation, presenting mathematics results, and ability of working in a team as well as getting a vivid view and a positive belief about mathematics and its application in real world.

REFERENCES

[1] ACM/IEEE Task Force Report on Computing Curricula 2001-Computer Science, Volume,2001. http://www.acm.org/education /curricula.html

[2] J. M. Wing. Computational thinking. CACM 49(3):33-35, 2006.

[3] M. Guzdial. Paving the way for computational thinking. CACM 51(8): 25-27, 2008.

[4] J. S. Warford. “An experience teaching formal methods in discrete mathematics,” SIGCSE Bulletin, 1995, pp. 60-64.

[5] M. Shaw. Software Engineering for the 21st Century: A Basis for Rethinking the Curriculum. CMU-ISRI-05-108. School of Computer Science, Carnegie Mellon University, Pittsburgh PA. 2005.

[6] P. Dourish, Hayes. Informatics at UC Irvine. Abstracts on Human Factors in Computing Systems. CHI '08. ACM, 2008, 3651-3656.

[7] N. Crisler, P. Fisher. Discrete mathematics through Applications. W. H. Freeman and Company, 1994.

[8] B. Marion. Final Oral Report on the SIGCSE Committee on the Implementation of a Discrete Mathematics Course. In SIGCSE Technical Symposium on Computer Science Education, 2006, pp 268-9,

[9] C. Boutilier, R. Brafman. CP-nets: A tool for representing and reasoning about conditional ceteris paribus preference statements. Journal of Artificial Intelligence Research (JAIR), 2004, 21, 135–191.

[10] Wing, J. M. , “Computational thinking”, Communications of the ACM, Vol.49, 33-35, 2006.

[11] King, P. M. & Baxter-Magolda, M. B., “A developmental perspective on learning”, Journal of college student development, 37(2), 163-173, 1996.