

# The Complexity of Teaching: Computability and Complexity

**Muhammad Aasim Qureshi**

UIIT, PMAS-Arid Agriculture University Rawalpindi, Pakistan  
maasimq@hotmail.com

**Onaiza Maqbool**

Dept. of Computer Science, Quaid-e-Azam University, Islamabad, Pakistan  
onaiza@lums.edu.pk

**Abstract:** Teaching computer science is as difficult as the subject itself, indeed even more so. A teacher of computer science has the dual responsibility of making the students understand complex concepts/methods as well as equipping them with the ability to apply these methods to solve problems. An understanding of the problem and its significance is essential and should be the first step. After making sure that students have gained an appreciation of the problem, the teacher should guide them towards the solution(s). Guidance has to be "balanced" enough to leave room for independent thought, while at the same time steering the students in the right direction. The traditional way of teaching is through reading from the textbook and doing problems through rote memory of formula and facts. There is a critical need to restructure the methodology of teaching, specially teaching mathematics and science. This paper is based on our experiences during the Teaching Science and Mathematics: Discovery Based Learning course, an advanced level computer science course designed to enhance teaching skills. In this course students from diverse areas were enrolled to share and present their discovery based teaching styles. We selected Computability and Complexity from the field of theoretical computer science to illustrate the topics which students find difficult to understand and the psychological reasons for these problems. Computability and Complexity is one of the most intellectually challenging courses that students study during the computer science programme. Its abstract nature makes it an appropriate choice for discovery based learning. The main focus of the subject is on theoretical aspects of computation, with computability focusing on unsolvable problems, and complexity focusing on difficult problems. The topics are so mind-boggling that they slip from the mind and cannot be grasped in their entirety. Even when students feel they have understood a concept, what they have grasped in the first instance is often only a part of the problem, and as the intricacies are revealed, the concept becomes elusive once again. Godel's theorem, which says that there are some true statements that cannot be proved, is one such example. To teach any course, specially such a challenging one, it is essential for the instructor to know the intricacies as well as the complexity level of the topic that he/she is going to teach. There will be considerable improvement in the students' learning if the instructor has knowledge of the difficult topics, understands reasons why the topics are difficult and devises ways to overcome the difficulties beforehand. In this regard our paper identifies intellectual challenges in the selected course, while also pointing out reasons why some topics are particularly challenging. Appreciating the fact that individual students process information and approach problem solving in different ways, we employ techniques to guide the students through visualizations and examples in such a way that they can discover and reach conclusions on their own.

**Keywords:** Computability and Complexity, discovery based learning, recursion theorem, Cantor's theorem, reductions

## Introduction

Computability and Complexity is one of the most intellectually challenging courses that students study during a computer science programme. The abstract nature of the course makes it particularly difficult for students as well as instructors. It has been observed that students have different styles of learning. Some learn through examples taken from real life, some learn through visualizations whereas some go through the steps of a problem in

detail in order to understand it. It is difficult to use any of these methods in computability and complexity since the course is abstract and often no real life situations can be utilized to explain or re-enforce the problem or solution. The steps taken to solve problems are complex and often lengthy. Even when the instructor is guiding the students, it is difficult for students to move to the next step. In other words, students find it very difficult to identify intermediate steps or the right operators which will bring them closer to the solution. The reason for this is that students have to be able to relate the problem at hand to the concepts they are familiar with. It is difficult to find this relation because problems encountered in this course may not resemble problems they have come across previously. Moreover, even if there is some similarity it may not be apparent.

The first step in teaching such a course effectively is to identify the intellectual challenges in the course and the reasons why students find a topic or a step particularly difficult. Once the challenge and its reason have been identified, the next step would be to devise techniques to make it easier for the students to gain an understanding of the topic as well as be able to use this understanding as a building block to solve more complex problems. In this paper, we focus on identifying the intellectual challenges in the course as well as reasons why these challenges exist. In the last section, we demonstrate techniques to teach difficult topics which are based on the idea of guiding students in such a manner that they can arrive at conclusions on their own rather than depending on the instructor.

The outline of the paper is as follows. Section 2 presents an introduction to computability and complexity. Section 3 presents a categorization of various topics in the course according to their level of difficulty. It also describes techniques for teaching selected topics. Section 4 presents the conclusions.

### **Computability and Complexity**

Computability and Complexity is an advanced level course in the theoretical computer science category, offered at many universities having a graduate level computer science programme. At most universities, it has two pre-requisites which include Analysis of Algorithms and Automata & Complexity Theory. The main focus of the course is on theoretical aspects of computation rather than on practical problems. This course is a 'pure' computer science course that helps us understand and address philosophical questions about computations and limitations of computing. The basic question addressed is what are the fundamental capabilities and limitations of computer **Error! Reference source not found..** In computability theory, the emphasis is on categorizing problems according to whether they are solvable or not. In complexity theory, the aim is to categorize problems as easy or hard.

When students register for the course, they are expected to be familiar with the concepts of algorithms and time complexity, various algorithms for common problems e.g. sorting, searching etc., deterministic and non-deterministic finite automata and regular and context free grammars. Having studied the course, students should have gained an appreciation of the fact that

- certain problems cannot be solved
- it is possible to prove that certain problems are unsolvable

ods in computability  
tions can be utilized  
to solve problems are  
tudents, it is difficult  
d it very difficult to  
g them closer to the  
the problem at hand  
on because problems  
ne across previously.

ntify the intellectual  
or a step particularly  
e next step would be  
understanding of the  
block to solve more  
llectual challenges in  
the last section, we  
n the idea of guiding  
their own rather than

ction to computability  
topics in the course  
for teaching selected

e theoretical computer  
evel computer science  
h include Analysis of  
as of the course is on  
ems. This course is a  
address philosophical  
asic question addressed  
ater **Error! Reference**  
categorizing problems  
theory, the aim is to

amiliar with the concepts  
n problems e.g. sorting,  
mata and regular and  
should have gained an

- certain problems are complex, and the complexity arises from the inherent nature of the problem rather than from the computational model being used

### Challenges

The table below lists topics in the course and categorizes them as soft, intermediate or hard challenges. For the intermediate level and hard topics, we have tried to identify intermediate steps/challenges which represent the mental jumps that are required to understand the topic. If an instructor is able to identify these intermediate challenges and find means to help the students overcome them, the topic as a whole becomes easier. The techniques that were used to reach the conclusions include a discussion with the instructor, detailed discussions with students who have studied or are studying the course as well as our own experience as students of the course.

**Table 2: Hard, Intermediate and Soft Topics in the Computability and Complexity Course**

Level	Topic
Soft	Turing Machine Concepts, Multi-Tape Turing Machines, Equivalence of Turing Machines, Complexity Classes
Intermediate	Universal Turing Machine, Diagonalization, Cantor's Theorem, Cook Levin Theorem, Savatch's Theorem
Hard	Halting Problem, Recursion Theorem, Rices' Theorem, Godel's Theorem, Reductions

In the sections below, we identify intermediate challenges within selected topics and discuss techniques to overcome these challenges.

#### Cantor's Theorem

Cantor's first theorem states: Let  $C$  be the set of all infinite 0, 1 sequences and  $N$  be the set of Natural numbers. Then  $|C| > |N|$ .

#### Reasoning about Infinity: The First Challenge

The theorem performs a comparison between the set of real and natural numbers and proves that the set of real numbers is larger than the set of natural numbers. In this theorem, Cantor tackles the abstract notion of infinity and tries to quantify the concept and reason about it in a meaningful way. Infinity is an elusive concept that defied definition and measurement for a long time because it was viewed as something abstract and immeasurable. One of the earliest recorded views on infinity is Aristotle's view, who says that the infinite is imperfect, unfinished and unthinkable. Galileo thought it wrong to speak about comparisons between infinite quantities. The influence of the earlier philosophers was so great that even till the 19th century, scientists treated infinity as an immeasurable quantity. In 1831, Carl Frederick Gauss said, "I protest against the use of an infinite quantity as an actual entity; this is never allowed in mathematics." Thus the first challenge in teaching Cantor's theorem is confronting an abstract or vague notion and reasoning about it in a precise way. The theorem forces us to ask some fundamental questions about infinity and infinite sets. Can we compare infinity? Can we say for sure that two infinite sets are equal? Can we say for sure that two infinite sets are unequal? When students are asked these questions, it is not possible for them to come up with answers on their own. They need to be guided, and as the following paragraphs show, it

is possible to guide them with simple examples and concepts instead of involving them in complex mathematical detail. These simple concepts lead them to a proof of the Cantor's theorem, and equip them with the knowledge to reason about problems related to infinity in a logical manner.

The first step in studying Cantor's theorem is to question students about infinite sets. This generates interest in the theorem and also helps in dispelling certain wrong notions. Many students, when asked what an infinite set is answer by saying that it is a set containing everything. However, this certainly may not be the case. As an example, consider the set of natural numbers. When asked whether it is an infinite set, their reply is in the affirmative. It can then be demonstrated to them that this infinite set does not contain 1.1 (or for that matter any real number) and hence we can't say that an infinite set contains everything. Present to them more examples of infinite sets and ask questions which force them to think and reason instead of guessing. One pair of sets which can be presented is:

$$A = \{1, 2, 3, 4, 5, \dots\}$$
$$B = \{4, 5, 6, 7, 8, \dots\}$$

Ask the students which of the sets is larger. The answer to this question is mostly that set A is larger. The reason for this answer is that B seems to have some 'missing' elements. Another pair of sets of particular interest is:

$$A' = \{1, 2, 3, 4, \dots\}$$
$$B' = \{2, 4, 6, 8, \dots\}$$

Again the students seem to think that A' is larger because B' seems to have missed out 1, 3, 5, .... One of the reasons why questions about infinite sets are interesting is that the answer is often different from what one might intuitively think as correct. At this point, it may be better to leave the questions unanswered. The students should be presented with examples so that they can be gradually steered towards the correct answers themselves.

#### **Performing Comparisons without Counting: The Second Challenge**

As has been demonstrated by the examples above, a challenge that is faced when dealing with infinite sets is that we have to perform comparisons even though we can't count the elements. To help students understand how it is possible to do so, a very simple example can be given. Two four year olds are given chocolates in red and blue wrappers respectively. They don't know how to count, but can they check whether they have been given the same number of chocolates? Students arrive at the answer quite quickly. The children can compare the number of chocolates by pairing the chocolates, one red and one blue. If pairs are made without any one color being left over, we can safely say that the number of chocolates is the same. This is exactly the concept that is used in comparing infinite sets. We don't know how many elements there are, but if one-to-one correspondence can be established between them, we can say that the sets are equal. It is time now to revisit the pairs of sets in the example above. Having been introduced to the concept of one-to-one correspondence, what do the students say about A and B? What about A' and B'? Their views should be different now, and in case they are still confused,



d of involving them in  
 a proof of the Cantor's  
 lems related to infinity

about infinite sets. This  
 wrong notions. Many  
 t it is a set containing  
 mple, consider the set  
 their reply is in the  
 et does not contain 1.1  
 in infinite set contains  
 questions which force  
 1 can be presented is:

stion is mostly that set  
 ne 'missing' elements.

ms to have missed out  
 interesting is that the  
 correct. At this point, it  
 ould be presented with  
 answers themselves.

t is faced when dealing  
 ough we can't count the  
 a very simple example  
 ed and blue wrappers  
 whether they have been  
 ver quite quickly. The  
 hocolates, one red and  
 we can safely say that  
 oncept that is used in  
 e are, but if one-to-one  
 the sets are equal. It is  
 been introduced to the  
 about A and B? What  
 they are still confused.

form pairs of elements from the two sets to demonstrate the equality of both the above  
 sets. In the above cases:

$$F(x) = x+3$$

$$F(x') = x' * 2$$

Thus it is possible to find a one-to-one and onto function between the sets in a pair. If  
 such a function can be found, it means a one-to-one correspondence is present between  
 the two sets and thus the size of the two sets is equal. The size of a set is also known as  
 the cardinality of the set, and sets having the same cardinality as the set of Natural  
 numbers are said to be countable.

### The Diagonalization Argument: The Third Challenge

If two sets can be proved to be equal in this way, how can they be proved to be unequal?  
 The answer to this question is simple now, and students don't have much difficulty in  
 answering it. To prove two sets to be unequal, we have to show that one-to-one  
 correspondence does not exist between the elements of the sets. If we can find an infinite  
 set which does not have a one-to-one correspondence with the set of natural numbers, we  
 are talking about infinities coming in different sizes. This is a very interesting idea. The  
 proof that Cantor presented to show that a one-to-one correspondence cannot be  
 established between real and natural numbers involves diagonalization, which presents  
 another challenge to the students. This is an example of a proof by contradiction, where  
 we assume that every real number can be paired with some natural number. So

$$F(1)=0.011011$$

$$F(2)=1.010101$$

$$F(3)=1.111111$$

$$F(4)=1.000101$$

$$\dots\dots\dots$$

Cantor argued that there was at least one sequence or real number which was not present  
 in the above correspondence. Students should be encouraged to come up with such a real  
 number on their own. They may be guided through various questions. How can we form  
 a number that is different from the 1st number? How can we form a number that is  
 different from the 1st and 2nd number? How can we form a number that is different from  
 all the numbers given above? Cantor formed the number by taking any number as the  
 integer part, and then forming the fractional part as follows. Take the 1st digit after the  
 decimal point to be different from the 1st digit after the decimal point of the 1st number.  
 Take the 2nd digit after the decimal point to be different from the 2nd digit after the  
 decimal point of the 2nd number. Continue in this way, and you will get a number that is  
 different from all the numbers present in the list. (In this case the number is 0.100000).  
 Hence, whatever list we form we can always come up with one number that is not present  
 in the list i.e. one number that has been missed out and so the set of real numbers is larger  
 than the set of natural numbers.

The confusion that students face here is the following: Since we have not enumerated all  
 natural numbers, it is possible to pair up the number we have just created with some

natural number which has not been enumerated so far. For example we can take  $F(5) = 0.1000$ . The point to remember here is that even if that number has been paired up, we can find another number using the same argument above. In this case, the number would be  $0.100010$ . If this was paired up, we can find yet another number, and so no matter where we are in the list of natural numbers, we can find at least one missing number. This is the diagonalization argument used by Cantor to prove that the set of real numbers is larger than the set of natural numbers and that infinity comes in different sizes.

### Recursion Theorem

The recursion theorem states that we have to come up with a Turing machine that can obtain its own description and then computes with it. It is a mathematical result that plays an important role in advanced work in the theory of computability.

To understand this concept let us summarize a paradox that arises in the study of life.

- 1- living things are machines
- 2- living things can self produce
- 3- machines can not self produce


Statement 1 is true as we believe that organisms work in a mechanistic way. Statement 2 is obvious but statement 3 is confusing as there can be machines that can produce themselves as in an automated factory of cars can be manufactured without human involvement. In short the simple answer is that the third statement is false i.e. Making machines that can reproduce themselves is possible.

The recursion theorem is simple to state but it is difficult to make students understand the concept, and even more difficult to make them understand the proof. To make them understand the concept, an example of a self replicating program can be given. A typical example of such self replicating programs with which the students are expected to be familiar are computer viruses. To allow the students to come up with the solution or come close to the solution themselves, the method of teaching can be divided into the following stages/steps.

### Writing a Program that Outputs its own Code: The First Challenge

Ask students to come up with a program that outputs its own code, without using file handling. Our experience with students shows that students get stuck at

```
#include<iostream.h> void main(){
cout <<"#include<iostream.h>\nvoid main(){}\n cout << ..... }
} A
```



**Figure 1: Attempting to output the exact code program**

the point where they have to incorporate the statement that they were currently writing. The figure shows the situation where the program outputs 'A' but 'A' now also needs to be incorporated in the `cout` statement as it has also become a part of the program and thus is to be output. This problem remains there no matter how one tries to write the `cout` statement. From here students normally reach to the conclusion that no program can print

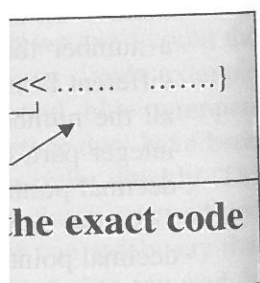
like  $F(5) =$   
ed up, we  
ber would  
no matter  
nber. This  
umbers is

ie that can  
t that plays

of life.

Statement 2  
an produce  
out human  
i.e. Making

derstand the  
make them  
n. A typical  
pected to be  
tion or come  
he following



ently writing.  
also needs to  
gram and thus  
write the cout  
gram can print

its own description/coding since a program that does this must be larger than its output. At this point, to enhance the motivation level of the students, give them examples which indicate that the problem is solvable. For example, if in a fully automated car factory, robots can make cars then why can't robots make robots just like them?

### Coming up with the Two Function Concept: The Second Challenge

We need an example to convince students that this seemingly impossible problem is not impossible but what they need to do is to think differently. Put forward a new problem from real life and ask its solution. "You have a bicycle with one pedal. The pedal moves, only, in forward direction with the pressure. What you will do? Would you be able to ride?" The obvious answer will be "no". Now ask them "What do you need to ride successfully?" Their answer will be that they need two pedals like the normal bicycles. Normally students reach this conclusion quickly because they have memory schemas in the right place to support them. But you need to re-phrase the answer to make them understand how you are forming a bridge between the recursion theorem and its proof through this example. In many cases re-phrasing a problem statement can result in better understanding of the problem as well as cues for its solution. The rephrasing would be "The pressure from one foot will move one pedal down and the second up, the pressure from the second foot will move the second pedal down and at the same time move the first pedal up as well". Now you have given a clue to students to think differently. Ask them to apply this result to the problem in hand of writing a self-replicating program. Some of them will be able to relate the two feet to two functions/parts of programs, and will suggest that the problem can be solved or at least attempted in terms of two functions, one printing the other and vice versa. At this step students are very close to the solution. They have reached the conclusion that they can come up with a program having two functions X and Y such that X will print Y and Y will print X. Now you have to tell the students how to do this.

### How to Write two Parts of Program: The Third Challenge

So here you need to say that let your program is in two parts say 'X' and 'Y' and you need to print the whole program i.e. you have to print 'X' as well as 'Y'. It is obvious that 'X' can easily print what ever is written in 'Y'. But the question arises, "can we do the same thing for printing 'X' ". The simple answer is 'No'. If we try this we will not be able to get the required output. So here we have to do something different.

### How the Second Part has to Print First Part: The Forth Challenge

To help the students form a better idea of how the functions will work, ask them the following question. There are two accountants one working in the 1st shift and the other in the 2nd shift. Accountants are used to writing each and every calculation/transaction from start till end, now can accountant2 somehow tell what operations accountant1 performed in the morning? The answer is trivial, since accountant2 can tell exactly what operations accountant1 performed through the output that accountant1 had generated.

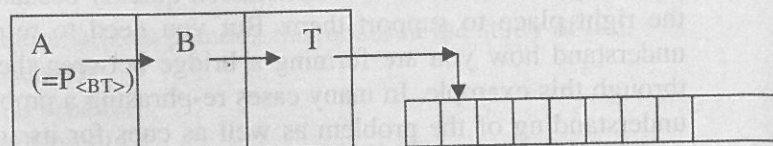
### Constructing the Turing Machine: The Fifth Challenge

Here the above example of accountants will work. The instructor has to tell the students that as accountant2 can find out and tell what operations accountant1 had performed in the morning, the same thing we have to do to print 'X' from the output of 'X'. Here a small programming example can also be given.

**Example** Write, "Recursion theorem is a beautiful theorem" and ask them to find the operations that lead towards this output. Students can easily do it as they have well-established cues as well as schemas to handle such problems. Now tell students that if a program can be written in a way so that it can output some predetermined output then why not a function?

Let's continue our discussion on the self-reflecting program where 'Y' will print 'X' not through the coding but through the output that 'X' had generated. For this thing we have to write 'X' in such a way that it can be generated from its own output (and it is not a tough thing to understand).

Coming towards the Turing machine, now after describing all this, students will, hopefully, understand how to come up with such a Turing Machine, quickly. You just have to tell them the whole program is your Turing machine and the two functions are two parts of the Turing Machine as shown in figure 2. Because of the example of the bicycle, students can easily come to the point that this machine is to be implemented in two parts, A and BT. To get a machine that outputs BT, define a sub-machine  $P_{\langle BT \rangle}$  (a submachine that prints the encoding of BT like our function X) as  $q(\langle BT \rangle) = \langle A \rangle$  (it means that you have to code A in such a way that its own output can describe its operations i.e.  $\langle A \rangle$  is a function of  $\langle BT \rangle$ ). The second part of machine takes the output of A (that is in fact encoding of BT i.e.  $\langle BT \rangle$ ) from the tape and with its help outputs the encoding of A (like accountant2 can tell what accountant1 did in the morning or example of the code given above).



**Figure 2: Self-Reflecting Turing Machine**  
here part A is equivalent to program

### Reductions

In order to prove that a problem is NP-complete, a known NP complete problem (e.g. the SAT problem) can be reduced to the problem under consideration. In this way, a solution to the problem under consideration would lead to the solution of the SAT problem. Since the SAT problem does not have a Polynomial time solution, neither can the problem under consideration. This is the basic idea of reductions.

A large number of problems have been proven to be NP-Complete. We discuss below reduction from 3CNF SAT to Clique. The 3NF problem can be stated as: Given a boolean formula of the form:  $\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6)$  where  $x_i$  is a variable that can be either true or false, can we assign the variables true/false values such that the formula evaluates to true (is satisfied)? A clique can be defined as a complete subgraph. Finding a clique in a graph is an NP-complete problem.



them to find the  
they have well-  
students that if a  
ined output then

will print 'X' not  
his thing we have  
ut (and it is not a

is, students will,

ing Machine  
to program

can easily come to  
and BT. To get a  
ine that prints the  
you have to code A  
A> is a function of  
act encoding of BT  
A (like accountant2  
iven above).

te problem (e.g. the  
this way, a solution  
SAT problem. Since  
er can the problem

. We discuss below  
as: Given a boolean  
is a variable that can  
such that the formula  
subgraph. Finding a

### Deciding the Direction of Reduction: The First Challenge

It is important to emphasize that an NP complete problem needs to be reduced to the problem under consideration and not the other way round. Simple as this may seem, the reason is sometimes difficult to understand.

Most of the students do not realize the reason or the essence of reductions. One reason for this confusion is their previous experiences with mathematical proofs. In mathematical proofs, they often have to prove a given problem to be the same as another given problem. To do this, they normally start out with the first problem and carry out steps to prove it equivalent to the second. In the case of reductions, even though a problem has to be proved to be equivalent to another set of problems, the direction of proof is the opposite i.e. we take a problem from the second set and prove it to be equivalent to the first problem. Having learnt to prove things in a certain way, students now have to contradict their past experience and unlearn their previous learning. This represents a case of set effects i.e. students become biased by their previous knowledge and prefer to use certain steps to solve a problem.

### Selecting the Problem for Reduction: The Second Challenge

A number of NP-complete problems exist. It may be easier to reduce one NP-complete problem to the considered problem and may be quite difficult to reduce another one. Quite a large number of NP-complete problems need to be considered before a suitable selection can be made. When experts in the subject choose an NP-complete problem to reduce from, very often apparently there is no relation between the two. Experts can utilize their experience to make a selection, but this is often not obvious to students who are neither familiar with too many NP-complete problems nor can find a relation between two problems when on the surface the problems look different. The NP completeness of the clique problem is often proved by a reduction from 3CNF. To explain the relation between the two, it is necessary to work on some examples of CLIQUES first. Familiarize students with the visual technique of finding CLIQUES of sub-graphs. This will clarify the concept of CLIQUES and will further help them as well as the instructor in the proof. Basically this exercise will generate some memory structures in their short term memory as well as in their long term memory.

We start our reduction from a very simple formula and gradually complicate the formula at each milestone, asking them a question again and again, "Convert the CNF formula into a graph such that when you take its clique it will give you a set of nodes so that if you assign 'true' to the corresponding variables in the formula you will get a required result".

### Performing the Reduction: The Third Challenge

Normally a graph is constructed directly from a CNF formula without describing the reasons why the graph was constructed in a certain way. We suggest the following steps to explain this reduction:

#### Step I:

The simplest formula that we decided to put forward is:  $\phi = x_1$ . This problem can be represented through a graph with only one vertex/node. Now ask them to



calculate the clique of this problem. They will answer that a clique of size 1 exists, and the set contains only one vertex,  $x_1$ . To enable them to find a relation between the formula and a clique, give them a hint: If 1 is assigned to elements in the set and then these values are put in the formula the correct answer, i.e. "true" is obtained. This example should enable them to realize that we can use the 3-SAT problem to reduce from.

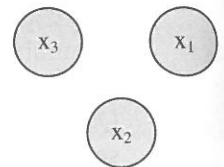
#### Step II:

After having used the simple formula  $\phi = x_1$ , move to the next level and change the formula to  $\phi = (x_1 \vee x_2)$ . Students will be able to tell that two vertices are required to represent the problem. The question arises whether we should put an edge between them or not, to complete the graph. That is an interesting question and students will surely come up with the answer that you need not put an edge between the two vertices. The reason is that the formula is true if either one of  $x_1$  or  $x_2$  is true. The rule that they will get from here is, "If there is OR in two variables then you need not add edge in the corresponding vertices"



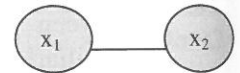
#### Step III:

Now move to the next level and change the formula to  $\phi = (x_1 \vee x_2 \vee x_3)$ . The students themselves will come to the conclusion and the answer will not be required from the instructor's end. The rule that they made in step 2 will be confirmed. You only need to ask these rules from the students and write them on the board.



#### Step IV:

Now modifying the intermediate step a little bit we change formula as:  $\phi = (x_1 \wedge x_2)$  and again put forward the same old question. Most probably students themselves will come to the conclusion that there must be an edge between the two. They will realize that in order to make formula "True" they need truth assignments to both of the variables. And it is possible only if you have a clique of size 2 and both corresponding vertices/nodes are selected in the clique set. Here you can add a comment that  $x_1$  and  $x_2$  can be two sub-graphs that will help us further in the proof. The rule that they will get from this step is "If there is an AND in two vertices/sub-graphs then edge(s) are added between/among the corresponding vertices".

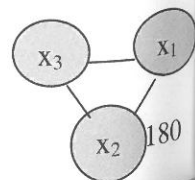


#### Step V:

Now moving towards our next intermediate step we made a change in the formula as:  $\phi = (x_1 \wedge x'_1)$  and again put forward our old question. Most probably students themselves will come to the conclusion that there must not be an edge between the two. Because if there was an edge then both will be selected in clique set and the selection of the two in clique set will give a contradiction as  $x_1$  and  $x'_1$  both cannot be true at the same time. So the rule that they themselves will make would be, "If there one of the two vertices is the complement of the other then the edge is not added in the two corresponding vertices".

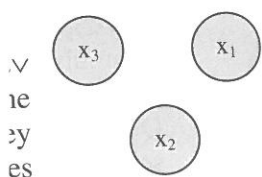


#### Step VI:



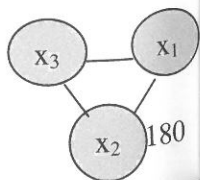
size 1 exists, and  
ation between the  
n the set and then  
is obtained. This  
problem to reduce

el and change the  
s  
e  
s  
swer that you need  
ula is true if either  
there is OR in two



as:  $\phi = (x1 \wedge x2)$  and  
nselves will come to  
1  
h  
ave a clique of size 2  
Here you can add a  
her in the proof. The  
vertices/sub-graphs

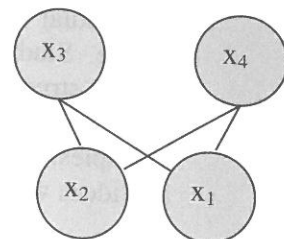
in  
on.  
hat  
n edge then both will  
ll give a contradiction  
they themselves will  
t of the other then the



Now modify your intermediate step  $\phi = (x1 \wedge x2 \wedge x3)$  and again put forward the same old question. Here students themselves will come to the conclusion that there must be an edge between the three as there is ANDing and there is no NOTing. This step confirms the rule that they established in step IV.

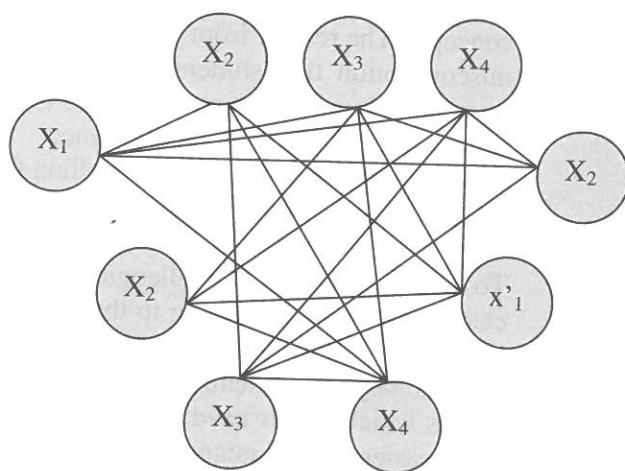
#### Step VII:

Now modify your intermediate step to the next difficulty level like  $\phi = (x1 \vee x2) \wedge (x3 \vee x4)$  and again put forward the same old question. Here the problem is to combine two different problems (of ANDing and ORing) into one. Here you have to convince them that they still can do it you just have to rephrase it like "we need phi to be true and phi can be true if we get true out of first term as well as from 2nd term. A slight hint would be enough at this stage. Just tell them to handle each term independently and then apply ANDing on two sub-graphs as done earlier. Here they will come up with a rule that will be, basically, the combination of different rules, "Handle the terms independently and then add edges between two sub-graphs such that no violation of previous rules are made".



#### Step VIII (Final Step):

Now moving towards our final step and presenting the right 3-CNF boolean formula as:  $\phi = (x1 \vee x2 \vee x3) \wedge (x2 \vee x3 \vee x4) \wedge (x2 \vee x'1 \vee x4)$  and ask students to come up with the graph themselves using the rules they have made in previous 7 steps. And optimistically they will come up with the right graph. Using the rule found in step VII they will be able to handle each term independently and come up with three independent sub-graphs and then combine them according to the rules defined by them.



By carrying out the above steps and gradually increasing the complexity level of the presented problem, students can be guided towards a proof of the NP-completeness of the clique problem. When a graph is constructed as suggested above, students fully understand the reason behind each step and hence can solve related problems as well as more complex problems with ease.

What is left behind proving that CLIQUE is NP-Complete is that if, as you have seen in steps, you could find out the CLIQUE of the graph then you can find the required results for 3-CNF. But as we know that 3-CNF is NP-Complete so there exist no such algorithm from which we can find CLIQUE so CLIQUE is also NP-Complete.

Now coming towards the second problem i.e. Vertex Cover (Finding a set of minimum most vertices that cover the whole graph). Now just like CLIQUE, you start with some

practice examples to consolidate the definition and clarify the definition in the minds of students. If you ask them to find some way to prove it to be NP-Complete then, at least they would not be lost at the very start but they would be able to find some way to attempt the problem (and our observation confirms it).

### Conclusions

Computability and Complexity is a difficult subject to teach and presents many intellectual challenges. Most of these challenges arise due to the abstract nature of the course. Studies in learning show that learning proceeds through the development of mental structures which remain long after the actual sentences or phrases studied have been forgotten. These mental structures are strengthened through visualization and examples. A new concept is easier to remember if it can be related to existing concepts and ideas with which students are familiar.

Techniques/Approaches discussed above were implemented in classes and the results (in the form of feedback) were studied in two categories

- (i) newly enrolled students
- (ii) previously failed students (studying these topics for the second time)

The results showed that the techniques were successful in making them understand such concepts. The results from second category of students were then analyzed. To nullify the misconception that students of second category understood these topics because they were studying this course a second time, the students were interviewed and asked the reasons for improved performance. The reason that they gave for their better understanding was the way of handling the concepts: Small jumps and smooth inclination rather than big jumps from one point to the second led them to grasp things easily.

To tackle a subject as challenging as computability and complexity, our first step is to categorize topics according to their level of difficulty. Having done this, the next step is to identify within each topic, steps which are particularly challenging for students and the reasons why the students find the step difficult. Through this identification, the instructor knows which topics need more time and appropriate examples can be given to make the topic easier to understand. In this paper we present challenges for three difficult topics, the reasons for difficulty and techniques to overcome these challenges. This analysis is useful for instructors as well as students, and can enhance students' learning considerably while also providing them with the means to think about and solve more complex problems independently.

### References

- Sipser, M., Introduction to the theory of computation, PWS publishing company, 1997  
G. D. Allen, The History of Infinity, Department of Mathematics, Texas A&M University, 1999  
J.R. Anderson, Cognitive psychology and its implications, 3rd edition, Freeman and Company 1990  
<http://www.heartofmath.com/IR/>  
<http://www.mathacademy.com>