Some Thoughts on Teaching Computer Science

Michael Gelfond

Computer Science Department

Texas Tech University

April 19, 2012

What, Why, and How

I teach my students that in any type of activity one should regularly stop and ask oneself the following questions:

- **What am I trying to do?**
- **Why do I want to do that?**
- **How should I do that?**

In what follows I talk about some of my answers to these questions related to teaching.

Some Teaching Goals

- awaken and increase a student's sense of wonder and awe;
- demonstrate importance of analytical thinking and scientific rigor; make them learn more about the process of discovery.
- help students to improve their taste and to develop personal scientific and programming styles;
- show that science in general and computer science in particular is an integral part of human culture.
- relate all these closely with practical needs of their trade and other aspects of their life.

Wonder

The begining of wisdom is awe **(The Bible).**

All rich educational experiences begin with an immediate emotional involvement on the part of the learner **(A. Whitehead).**

I remember how surprised and delighted I was when I first discovered that one can not see a triangle, or that a set of even numbers is equinumerous to the set of natural numbers, etc.

Many of my students are equally surprised and delighted by these facts, but some are upset and irritated.

The latter should be endured – sometimes their reaction changes with time.

Understanding the Consequences: Analysis

Are there important consequences of the fact that triangles cannot be perceived by our physical senses?

Yes, since they share this property with algorithms, programs, and other mathematical objects.

This means that to really know if an algorithm or a program satisfies certain property we need to use our mind, not physical senses.

It also means that existence of algorithmic solutions of problems can only be established by reasoning.

Rigour: naming

Problem: Design an algorithm which takes a set S and object x as an input and checks if $x \in S$.

Sadly, many people immediately start working on the design. This indicates the lack of rigour.

How to teach rigour?

I stress importance of definitions,

test students on definitions (with no partial credit),

show how changing a few words in the definition leads to a substantial change in meaning, etc.

Rigour: reasoning

Seeing the object and knowing its name is not enough to become the master of the object.

First one needs to become the master of the name, e.g. be able to extract important hidden consequences of definitions.

Need to teach correct reasoning!

Mathematical proofs and emphasis on correctness in algorithm design help.

Rigour in discussion

The wise man does not break into his fellow's speech; he is not in a rush to reply; he asks what is relevant and replies to the point; of what he has not heard he says "I have not heard"; and he acknowledges what is true **(Talmud, The wisdom of the fathers).**

Difficult to achieve but worth trying. I always repeat the question and ask a student if I understood him correctly; try to prevent the discussion from jumping from one topic to another, insist on explaining the terms used by a student, etc.

Very useful for me but I believe also useful for students.

Examples

Given: set S and object x.

Return: *yes* if $x \in S$ and *no* otherwise.

- **What is a set?**
- **What does it mean for a set to be given?**
- **Suppose x is a real number. What does it mean for a real number x to be given?**
- **What is an algorithm?**
- **Are there algorithms defined on real numbers?**
- **How can we define infinite sets by finite means?**
- **Can the above problem be solved for any S?**

Discovery, History, and Practice

All the above questions are answered in a course somewhat inappropriately named “Theory of Automata”.

There are colorful figures of Turing, Post, Kleene, Rabin, Pratt, and others, connections with mathematics, philosophy, linguistics, etc.

The course also contain some of the best examples of algorithm design.

Similar wealth of material can be found in the AI class.

Developing the Taste

I often start my classes with asking students to name some great discoveries made by their favorite computer scientists, or to describe their favorite programs or systems.

Sometimes I ask them if they like a program or an algorithm we developed in class, etc.

Unfortunately, too many of them do not know what I am talking about.

They, however, have no problem with naming their favorite songs, books, writers, or musical groups.

In my view this is a symptom of a very serious problem!

Developing the Taste

So I ask them to evaluate and pass judgment on programs, results, and theories!

They are culturally conditioned to avoid giving judgment so it is difficult but doable.

Example:
Given: a list $S = [x_1, \ldots, x_n]$ of natural numbers and a natural number x.
Return: index of an occurrence of x in S, 0 if there is no such occurrence.
Restriction: S should be stored in an array:

Comparing the Solutions

Solution 1: $S = [T[1], \ldots, T[n]]$**.**

```
i := 1;
while S[i] <> x and i < n do
      i := i+1;
if S[i] = x return(i) else return(0)
```

Solution 2: $S = [T[1], \ldots, T[n]]$**;** $T[0]$ **is unused.**

```
T[0] := x; i := n;
while S[i] <> x do
      i := i-1;
return(i)
```

Conclusion

I love teaching and try to constantly learn how to do it better.

The possibility of discussing teaching with colleagues is an indispensable asset – I wish I had more of it.

But the most important thing is to increase (or at least not to loose) my own sense of wonder and awe, and to continue to learn and think about computer science.

What are the new great results? Important questions? Deeper insights?

Any help in learning this will be appreciated.