On Learning History

Michael Gelfond

Texas Tech University

April 10, 2014

What am I doing here?

Some time ago I was asked to give the ACM banquet talk explaining importance of learning history of science.

I was very busy, but there was about a month until the banquet and it seemed like I had a lot of time to prepare. So I said “yes”.

This is an hour when I have to pay for my foolishness.

I had very little time to prepare but I’ll do my best anyway.

On Nature of Learning

Many people believe, I think incorrectly, that most of their learning comes from their personal experience.

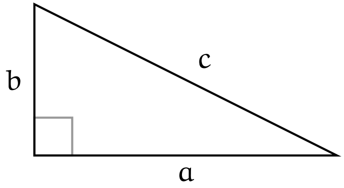
Even though such learning is very important most of our knowledge was developed by previous generations and transmitted to us by our teachers and institutions.

Today I mostly going to talk about the right ways to learn this inherited knowledge.

I'll do this by discussing a few examples of great discoveries well known to most of you.

Example: Pythagorean Theorem

Pythagorean Theorem: $c^2 = a^2 + b^2$.



This is an important theorem which has been passed from generation to generation for more than 2000 years. Eventually the theorem reached us and, I believe, is known to people in the audience.

But do you really know it?

During my 40+ years of teaching I conducted many (unscientific) experiments to answer this question and strongly believe that an answer is mostly negative.

Very few people, for instance, were able to explain to me why this theorem is important.

Even a fewer number believed that the theorem has any relevance to their lives.

This means that for most people this knowledge is dead, and having a lot of dead knowledge is worse than having no knowledge on the subject at all.

So let us see if we can learn a little more about the theorem and if history can help us to do that.

The Learning Game

Let us assume that you just became aware of this theorem and want to understand it.

You have a teacher capable of answering most of your questions but he does not voluntier any information.

What should you ask? Here are some suggestions:

- Why is it true?
- What question does it answer, and why this question was asked?
- Does it have important consequences?
- Does it come with or helps to create a new perspective, a new way of looking at things?

How do we know that Pythagorean Theorem is true?

Everyone knows how to answer this question – the theorem was proved.

But very few students realize that the concept of mathematical proof itself was at that time new and revolutionary.

After all Pythagoras lived a few hundred years earlier than, say, Euclid.

The theorem certainly gave a strong impetus to the development of the accurate notion of proof.

Why was the question asked?

No one seems to know.

It could be that people were simply trying to reduce measurement to computation.

There could be some practical or mystical reasons.

Much knowledge is lost after 2500 years.

Pythagorean Theorem: An Unexpected Consequence

Theorem: ***The diagonal,*** c ***of a square is incommensurable with its side,*** a**.**

Why is this surprising?

Greeks believed a segment to be a physical object. All physical objects consist of atoms, which can be used to measure the lengths of segments.

Let n **be the length of an atom. Then there are** m **and** k **such that** $a = m/n$ **and** $c = k/n$**.**

BUT THAT's IMPOSSIBLE!

So what went wrong?

Is the proof wrong?

Is the physical theory wrong?

If no error is found in the proof should the physical theory be changed?

After much thought Greeks realized that the problem may be in *viewing a segment as a physical object.*

They decided to separate between physical and mathematical objects. The great philosophical divide was born.

What is a mathematical object?

We can restate the question as: *How can we know that a mathematical object exists and how can we determine its properties?*

Greeks opted for determining existence *algorithmically*: a geometric figure exists if it could be constructed using a compass and a straight-edge.

Properties of these objects are to be determined by *pure reason*, without relying on our five senses.

It took almost 2000 years to show that, according to this definition, there is no square commensurate with a circle.

Pythagorean Theorem (continued)

I hope you understand that the Theorem dramatically changed our way of looking at things. And this is only one of many of its important consequences.

Do we still have the separation between physical and mathematical objects? Is it important for our science? How do we justify this separation?

And how all this is related to you as a computer scientist?

Effect on Programming Methodology

Are programs mathematical objects? If so, their properties can only be discovered by reasoning. This view lead Dijkstra, Hoar, Wirth, and others to develop the methodology of *Structured Programming*:

I only have time for two quotes on the subject:

Programs (and algorithms) should be developed together with proofs of their correctness.

In programming simplicity and clarity —in short: what mathematicians call "elegance"— are not a dispensable luxury, but a crucial matter that decides between success and failure.

The Pythagorean Theorem (conclusion)

I gave a short, mostly historical, analysis of one ancient theorem.

What I mentioned in this talk is only a very small portion of answers to the questions I wanted you to ask yourself in your effort to understand this result.

I hope, however, that this allows you to have a glimpse on the methodology of learning and the role of history in this process. If nothing else you will better understand what is meant by a great discovery.

It is unlikely that a talk can make the theorem to become alive and active part of your heritage.

For that, you have to spend time thinking about this story and the related methods.

Example: Recursion in Computer Science

Recursion: the method of defining algorithms or other objects in terms of itself.

Wikipedia: ***Recursion is one of the central ideas of computer science.***

This time the knowledge received from our ancestors is not a theorem but a method (or methodology).

To understand it we need, of course, learn its use at a number of examples. Most people here went through this process.

So now we can ask ourselves: is the Wikipedia's statement true and, if so, why?

Roots of recursion

Recursion is a special case of self-reference – another difficult logical notion the Greeks struggled with.

Because of this difficulty recursive definitions in mathematics were used very sparingly.

Godel defined a subclass of “good” recursive definitions for the proof of his famous Incompleteness Theorem.

Kleene, Godel, Peters and a few others expanded the theory to define partial recursive functions.

The idea spread and led to many discoveries. Here is an example:

Recursion in Computer Science: ALGOL

ALGOL is an early programming language designed by a group of scientists with the goal to make it possible for people to *communicate algorithms to each other*.

It appeared a few years after Fortran (which was designed for communication between a programmer and a computer).

Fortran didn't allow recursive functions since it was not at all clear how to implement them.

Dijkstra, who was one of the authors of ALGOL, argued that additional restrictions on the language make it more complex and less elegant, and, therefore, should be abandoned.

Fortunately, he prevailed – ALGOL has means to express recursion.

T. Hoare on ALGOL

From Turing Award Lecture:

I wrote the procedure, immodestly named QUICKSORT, on which my career as a computer scientist is founded.

Due credit must be paid to the genius of the designers of ALGOL60 who included recursion in their language and enabled me to describe my invention so elegantly to the world.

I have regarded it as the highest goal of programming language design to enable good ideas to be elegantly expressed.

Implementing Recursive Functions

A number of people working independently developed implementation of function calls based on the notion of *stack* which didn't make any distinction between recursive and non-recursive functions.

A little later stacks were implemented on the machine level which made for efficient execution of function calls.

Use of Bachus-Naur form (or Post grammars) to recursively define the syntax of ALGOL led to the development of efficient and correct compilers.

ALGOL (conclusion)

The development of idea of recursion led to better ability to communicate our ideas to other humans and eventually to computers.

ALGOL greatly influenced this development.

Recursion and recursive functions played a crucial role in the development of functional and logic based programming languages (Lisp, Prolog, etc.), in theoretical computer science and many other areas.

I believe development of ALGOL to be one of great advances in computer science.

The Last Slide

The last thing I'd like to say is this: if you really want to understand some idea or phenomena go to its roots, learn what great people thought about it.

Nothing can replace this journey.

This advise goes far beyond science.

Want to understand what it means to be human - read religious texts and classical literature.

Want to understand what it means to be an American – read Declaration of Independence, Constitution, and Founding Fathers.

After that test the ideas by their clarity and by their consequences in history and in your own life.

THANKS FOR LISTENING