ARE YOU INTERESTED IN THEORETICAL COMPUTER SCIENCE? (HOW NOT???) I HAVE SOME ADVICE FOR YOU

Michael Fellows
Department of Informatics, University of Bergen
michael.fellows@uib.no

Long ago, in the time of MEGA-Math (sponsored by the Los Alamos National Laboratories) Nancy Casey and I were doing some of those activities that are now better known as *Computer Science Unplugged!* activities (both easily googled) that are designed to present mathematical ideas that are foundational to computer science in concrete ways accessible to school-age children. We were visiting the first-grade classroom of legendary teacher Prudy Heimsch in Moscow, Idaho. Wondering how these concrete activities basically about theoretical computer science fit with her classroom objectives, we asked her, "Do you have any particular objectives in mind for your six-year-olds?"

Prudy answered without hesitation, *Yes I do! I have three goals for them:*

I want them to learn to communicate.

I want them to learn to think about their own thinking.

I want them to be able to formulate a project of their own devising, and carry it through to completion.

These core objectives will serve until graduate school and beyond! This being the case, here is my advice for students.

The call for these contributions of advice from the EATCS Fellows, asks for the discussion to be organized as advice for four different levels of education: School level: meaning primarily High School, University level, PhD students, Young researchers beyond the PhD. But what about the kids? The burning heart of curiousity is 6-7-8-9-10 year-olds! Attend to the fire.

Primary School. We were in love with Einstein when we were ten years old. We wanted to be theoretical physicists when we grow up. There was no chance for Relativity to be in the school curriculum. But there was a chance, in those olden days, to get a bit of what they used to call "enrichment". Someone would come to the school and tell us exciting stuff that was not in any lesson plan.

If you are a little kid, with any inkling of what theoretical computer science is about, or you just want to know, my advice to you is to demand **enrichment** to theoretical computer science, so that you can be exposed to exciting contemporary science, heart of modern civilization.

That's the advice if you are in the lower grades: **demand some exposure to theoretical computer science**. If your parents work in IT: **demand** that they visit your classroom, or that they arrange for one of their friends to come to your classroom.

Take charge of educating yourself. There is plenty of stuff on *Computer Science Unplugged!* at `http://csunplugged.org/`.

Much of my advice for High School and beyond is aimed at you as well. Explore questions. Fill notebooks with drawings, ideas, speculations, guesses, investigations. Look for the big picture, and then try to see details and how details fit together to make the big picture.

High School. Explosion of interests and engagements! Bring it on! Later you may appreciate all the ironies involved. Cultivate a passionate interest in literature. You cannot be a great scientist without it–to a first approximation. There is some weird correlation between physical and intellectual courage. Go surfing! Who is to advise people at this flowering age beyond th emselves? OK, pick an intellectually ambitious book about TCS, stick it in your backpack and GO. Mine was Goldblatt's book on Topoi, to the High Sierras of California. Just go, go, go! Develop mentors! Famous scientists (and younger ones) are easier to approach than you might imagine — just email, or write a letter on paper. There are tons of open problems — just join in. Do not waste too much time with computers. If you are interested in theoretical computer science, you are going to need a lot of mathematics, the ultimate magical mind game, with power over most everything going on, except for love.

University Student. There is almost no chance for a serious research career in theoretical computer science without solid foundations in Mathematics.

Learn math! If you are a college student, a double major in both Mathematics and Computer Science is a good option. If you must choose, choose Math. Mathematics is somehow ever-metacognitive, across all fields.

Join in! If you are an undergraduate student studying Mathematics with an interest in Computer Science, then you should join both the *American Mathematical Society* and the *Association for Computing Machinery*. I did that when I went to college after the Vietnam War, and although I didn't have much money at the time, it was the best investment I ever made. Publications will arrive to your mailbox describing exciting contemporary developments in these fields. Read about them, even if at the beginning, you don't really understand much of what is going on. Fields Medal winner Michael Freedman, who was a mentor to me, once told me about how he got into Mathematics as an undergraduate by browsing research journals and becoming obsessed with trying to figure out what they were on about.

Do that!

Educate yourself! Do not expect the university or the high school or the elementary school to educate you. At best, realistically, they won't get too much in your way. The school curriculum is inherently way too slow to keep up. You must depend on forming and making support groups of friends, finding mentors (easier than you might think), and taking things into your own hands.

Do research! It is never too early to take up a research project appropriate to your level. Mentors will happily help you. Exploring questions that nobody knows the answer to is *really fun*. Maybe surfing a really big wave is competitive in the short term, but doing science is fun and gratifying. If you manage to publish research as a high school student, which is a perfectly reasonable objective, this will open doors when you apply to the universities. If you manage to publish research as a university student, this will open doors when you apply to graduate school. Working on a real project is the best way to learn.

PhD Student in Computer Science. During a wonderful visit with the research group of Prof. Jianxin Wang at Central South University, Changsha, China, a student engaged me over dinner asking for advice and this is the result of our conversation. There are two different advices. One is for science and one for career. Often, they overlap.

SCIENCE

• Do not trust your advisor. Watch out for "mini–Me". Be prepared to sniff it out and bolt. Your interests do not align. Your advisor is (especially if young) almost always desperately interested in advancing their own career, and you should study the blues. There are exceptions, but they tend to be rare. Your true mission is to pioneer new and outlandish angles, ideas and kinds of questions, and be prepared to defend your own new directions of research.

• Put research first. Learn by doing. Try things out. Dream big.

• Work on more than one problem at a time, or in more than one research direction. Have several functional advisors.

• Don't be shy. Write to other scientists and to the authors of papers you have read.

• Choose your problems. Some problems have important applications, while others have the potential to build theory. Some people are natively problem solvers with sharp tools and others are theory builders with a big picture view, although probably all are a bit of both.

• Visit. This is part of, "Don't be shy". Visit other research teams. Learn who else is working in your areas. Get to know the other students and leaders.

• Pick good partners. Good research partners inspire each other to keep going past the 'finish line' and get the job completely done and on time.

• Make friends in other fields. Each field has its own vocabulary and solving techniques.

CAREER

There are two targets that you may be working towards: science and career. Both are important and there is no shame in favouring one over the other at various points in time—or in point of your interest or career (there may be ebb and flow).

• Enjoy writing for grants. It may seem strange to suggest enjoyment of grantwriting, but it is an opportunity to hone your writing skills, explain what you care about to people who may not be in the same area, (often they are not), to clarify in your own mind your plans and aspirations, rethink and reconnect with your collaborators. Once you have written a solid grant proposal, you may want to give away the ideas to others who are seeking funding from other sources. Good ideas deserve to be funded.

• Serve the reader. We have all encountered speakers and writers with an agenda of making themselves look intelligent and knowledgeable by using words and phrases that obfuscate the issues. Some writers give multiple citations to obscure references. Our job is to help the reader in every way possible to understand the sometimes arcane material we are offering.

• Story is central. Story is a bigger force than science. Everybody lives by stories. They are a primal force. In mathematics, we add formalism. We have equations that lead to solutions but story has its own logic. Find the story in what you are telling and presenting. This will help the listener meet you more than half-way.

• Generously credit everyone else's work. Fiercely defend your own IP.

Postdoctoral Researcher and Young Professor. The advice above for PhD students in Computer Science is appropriate for you, as well.

• When writing papers, generously credit previous and related work. Do not make the mistake of citing only FOCS, STOC and SODA papers. Your proper goal is to *serve the reader* not *impress the reader*.

• Now is your time to liberate your curiosity! Hedge your bets and work on more than one project. Have several functional advisors. Communicate with and visit other teams in other universities.

• Do stuff with young people. – Refreshes your morale (and amazes your colleagues and is very easy to do.)

– Good for your kids.

– Good for everybody, really.

– Good for science, government and industry because the effort to clearly communicate your ideas brings out new basic questions. – This is NOT about university recruitment!

• Think about moving (geographically)! Don't be overly concerned about moving to a "small" place. When I was a young professor, I joined a small, fairly remote university for family reasons. I was the only research professor, and this meant that I got ALL the research budget, which was considerable since the university got almost all the state's research budget. You may become a "big fish in a small pond" and be able to influence important research directions in your area.

Horizons.

The original call for these contributions of advice from the EATCS Fellows, asked for: *A short description of a currently favourite research topic.*

Many parameterised problems are FPT and the toolkit for approving such results is quite varied. Yet in some sense, because a parameterised problem is FPT if and only if there is a P-time kernelization algorithm, one can say that proving the best possible P-time kernelization bound is the canonical issue. Many FPT kernelization results can be proved by neatly structured and algorithmically well-behaved argumentation. This programmatic point of view is advocated in the paper *FPT is P- time Extremal Structure Theory: the Case of* MAX LEAF which gives a good example. I am interested in axiomatizing and exploring the limits and meta-theorems about *groovy FPT*: kernelization results where everything in sight is polynomial time or has a polynomial time interpretation. From such well-structured results, one can canonically derive P-time approximation algorithms, and canonically associated inductive gradients of interest to local search algorithms and greedy heuristics, thus neatly connecting several different basic algorihmic issues. There are also concrete examples showing that groovy FPT is a proper subset of FPT, so that this is a really exciting and fundamental direction in FPT research.