How to Succeed in Computer Science

Richard Bornat,
Director of Undergraduate Studies
Dept of Computer Science
Queen Mary, University of London
September 2001

Executive summary

1. ‘University’ learning is different from ‘school’ learning.

2. ‘Computer science’ learning is different from any other.

3. You are joining an intellectual community. It’s exciting.

4. University education is about change. You have to leave behind some old stuff, and you get some new stuff. People who change, succeed.

5. Programming is fun, if you let it be fun. Give it a go!

0. To begin

Lots of people come to University keen and eager to learn Computer Science.

You are ready to go.

You are capable of success.

Will you let us help you succeed?

We can help you to be a big success.

But you have to be ready to change, prepared to accept new ideas, in more ways than one.

Is Computer Science different?

Yes, it is.

The subject you have chosen seems to be stranger than most.

Why that is – and what you can do about it – come later on in the talk.

First, some general remarks about University study.

(Any book about study skills will cover these general points.)

Motivation

Everybody agrees that to succeed at university study, you must be ***properly motivated***.

Question: are you
(a) here to learn useful things which will help you get a job?
(b) just here to have a good time?

If you want to succeed, the correct answer is **(*b*)**. Don't believe me? I have evidence to prove it!!!

But you must have your 'good time' ***inside your subject***. You do better if you are studying ***for fun*** than if you are doing it for economic/family/(any non-fun) reasons.

Being clever

Employers prefer graduates. So young people come to University to get a degree, to get a job. That's OK.

Why do employers prefer graduates? Is it because of what graduates remember from their studies?

(***No!*** Information is easy to acquire: anybody can read it from a book.)

Is it because graduates are cleverer than non-graduates?

(***Yes!***)

Cleverness is like physical fitness: it improves with exercise. The purpose of a University education is to make you cleverer, ***not*** to provide you with 'useful information'.

Do-It-Yourself

School/FE is quite like ***training***: the subject is defined by an exam set by outsiders; student and teacher work together to beat the examiners.

University learning is like ***exploring***: you are given a map, and left to find out as much about the country as you want to, on your own. The lecturer is also the examiner (and so is *not* really on your side).

You are on your own – *unless* you grab all the assistance you can.

(You have to grab us: we won't grab you.)

(We want to be grabbed!)

(We won't always help you to an easy ride – see 'being clever' above.)

Being a programmer

What do people think about programmers? Do they think we are all nerds?

Yes, they do: listen to stand-up comedians; watch comedy programmes; read the newspapers.

Being a nerd is a bad thing; being a programmer is a good thing. ***We*** don't confuse the two, even if other people do.

Programming is marvellous, intricate, difficult work. It's creative work, like architecture. It's design work, like the best arts and crafts. Programmers build wonderful tools, beautiful interfaces. And they surf the wave of technological change.

They think we're nerds. They will call you a nerd, when you join us. Can you take it?

That was scary: what can I do?

Come and live inside your subject:

- enjoy it, like it, ***love*** it;
- don’t bother if people outside don’t understand what you are doing;
- stretch your mind by studying difficult things;
- don’t be afraid of seeming ‘keen’;
- and don’t believe (next section of this talk) in old-fashioned ideas about learning.

What a wrong theory can do

In 17th and 18th century England, people thought disease was carried by bad air. The rich walked about the street with nosegays – bouquets of flowers – which they sniffed to keep well.

Meanwhile, plague-rat fleas were biting them.

In early 19th century England, cholera was thought to be a product of bad dwelling conditions.

There is a monument in Soho to John Snow, who deduced that cholera was a consequence of drinking-water pollution, and removed the handle from the parish pump to prove it.

The wrong theory of disease was a deadly delusion. The right theory saved lives.

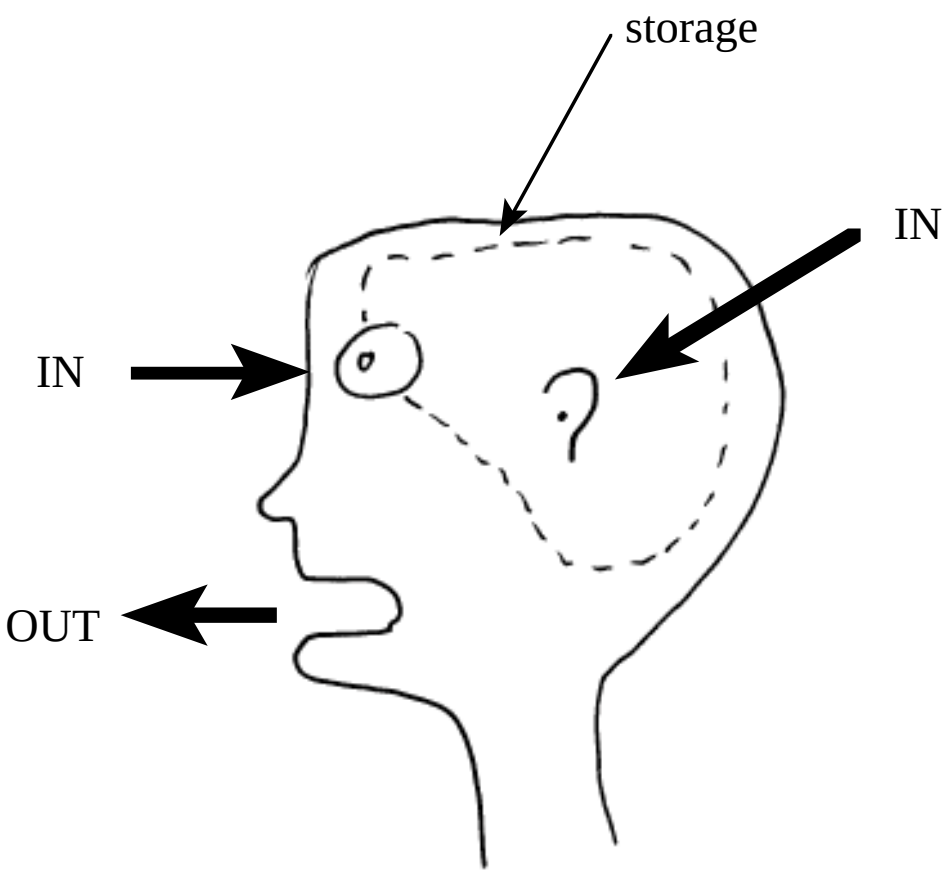
Only the ***right theory of learning*** can lead to success.

The wrong theory of learning

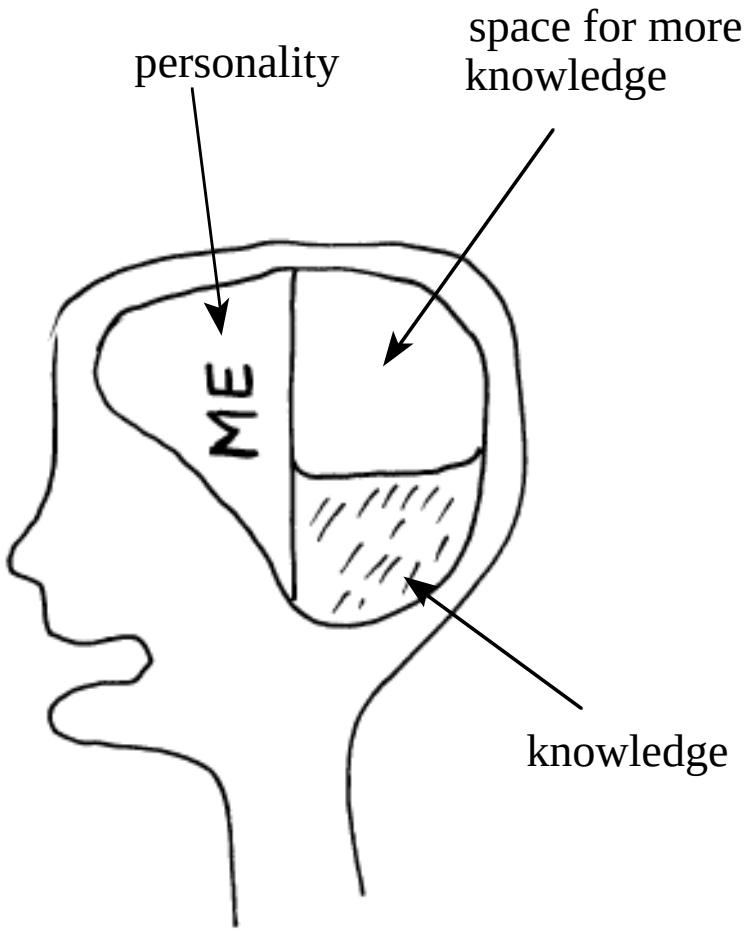
Most people, from the Government downwards, and most new students, think

"learning = knowing = remembering".

Here is a diagram of the outside of your head, according to this theory:



and the inside:



"Knowledge", according to this old-fashioned and ***wrong*** theory, comes in through the ears and eyes and is stored in the brain. It is separate from the personality.

It's not like that in there!

A better theory of learning (1)

1. Children actually *can* learn stuff without understanding it (so they can all work video recorders, to their parents' delight).

2. Listen/remember/revise learning works very well – *for children.*

3. Adults (that's you!) actually *cannot* learn stuff which they don't understand (that's why they can't work video recorders).

4. Most of your head has already changed over to adult mode.

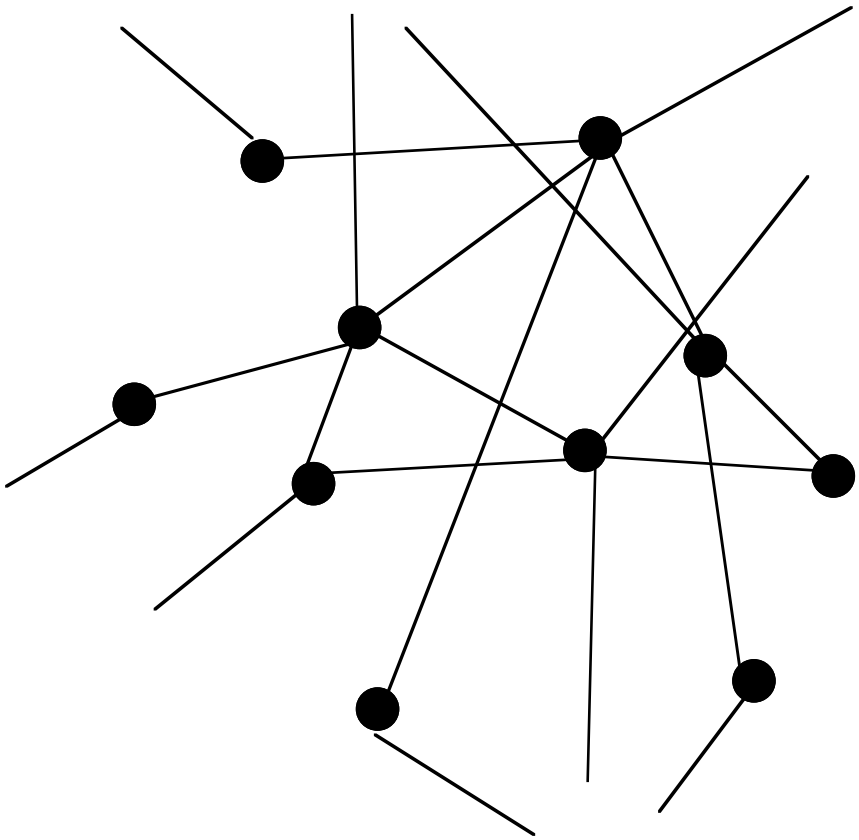
5. Learning new stuff changes you. You aren't separate from what you know!

6. Because of the nature of Computer Science, listen/remember/revise ***just doesn't work***.

7. ***Use/practise/discuss/reflect*** learning works best for adults, and is essential in Computer Science.

A better theory of learning (2)

Our best guess is that ‘knowledge’ is a complex network, a ***matrix*** of interconnections in your head. Stuff you know is defined by what it’s connected to.



The more connections, the more you can *use* a piece of knowledge. Example: you know arithmetic (from constant use) and you can use it, if you have to, to work out your taxes (etc., etc.).

Recalling something – e.g. for an exam – is just a weak kind of use. Weakly connected knowledge is easily forgotten.

'Learning' is an ***active*** process. You have to connect new knowledge into the matrix, with as many connections as possible.

If new knowledge won't fit in your matrix, you either have to reject it, or you have to rearrange the matrix. Rearranging means throwing away (or at least reconnecting) stuff which you previously believed/understood/'knew'.

Rearranging the inside of your head is hard work. Learning expands your mind, and it ***hurts***, just like hard physical exercise hurts your body.

How can teachers help?

I can't rearrange what's in *your* head! You have to do it.

I can, from my experience, *guess* what you might be thinking. So I can suggest rearrangements which you might need.

But it's best if you tell me! (In tutorials, in labs, in exercise classes, ...) Then I have a better chance of helping ***you*** with ***your*** particular rearrangements.

Nobody knows what is inside your head, unless *you* tell them.

So nobody can 'teach' you anything, except in discussion/conversation/debate/dispute/argument where you reveal what is inside your head and they reveal what is inside theirs.

Learning is what ***you*** do.

(Teaching doesn't really exist.)

Why does this matter so much?

Computer Science is pretty tough to begin with – almost everybody struggles.

That's because all Computer Science, and ***especially programming***, has a very peculiar character.

It is based on mid-20th century axiomatic logic. You have to understand a few basic axioms (facts, instructions), plus a few rules (techniques for putting things together into larger things like methods, procedures and classes), and then you can do an infinite number of things.

There is almost nothing to *know*, but a lot to *do*. It's an *intellectual skill* – a very odd combination.

Your head is not ready for this! School/FE teaching and learning didn't prepare you!

Your head needs a *lot* of rearranging before the new Computer Science stuff will fit. This is *not* just adding a few facts round the edge!

Axioms and rules have to go right in the middle of the matrix – they are so basic they can't go round the edge.

Some of their consequences have to go near the middle, some can go farther away.

It all has to mesh with what you already know. A lot of what you already know can be simplified; but lots of it *has to be thrown away*. You have to re-examine what you already know, think again what it really means.

Doing such a rearrangement *takes time*. Revision doesn't work because a week – or a month – just isn't long enough.

Once the axiom/rule stuff is in there, it doesn't get forgotten very easily. ***Use/practise/discuss/reflect*** learning means *no more revision*.

Learning Computer Science

[***Please*** believe me]

1. 29 years of experience has shown me that listen/remember/revise ***just doesn't work*** for our subject.

(Even though it *does* work in other subjects.)

[***Please*** believe me]

2. 29 years of experience has shown me that programming is the hard bit: once that's dealt with, the rest is much easier.

(One kind of knowledge supports another.)

[***Please*** believe me]

2. 29 years of experience has shown me that the ***only way*** to learn Computer Science is by sustained *use* and *practice* , backed up by *discussion* and *reflection*.

Use/practise/discuss/reflect

Programming is an intellectual skill. It's also creative. Learning to program is more like learning to paint than like learning history, or science, or any standard subject.

By using something you force yourself to know it. Programming is simple but slippery.

Practice is the only way to get it inside your head; *use* is the only way to keep it there.

Intellectual skills are like physical skills in that they have to be learned. You have to practice. You have to practice like mad!

Intellectual skills are *not* like physical skills in that they can be forgotten. You can't forget how to ride a bicycle (fact!) but you *can* forget how to calculate, or program, or analyse.

To keep an intellectual skill active, you have to *use* it. That stops it drifting out of the matrix.

Because you have an adult head, you can’t learn like children do, piling in meaningless facts. You have to fit knowledge properly into your matrix, or there will be trouble.

The way you do it is to examine and re-examine your knowledge, your beliefs, your experience. You have to *work* on the jelly in your head, pummel it into shape, constantly remake it.

Discussion exposes your knowledge to yourself, and exposes you to the knowledge of others. (I often don’t know what I know till I try to explain it to somebody else.) Discussion reinforces knowledge that works, and shows up knowledge that doesn’t.

Reflection – turning things over in your mind – works on the problems that discussion exposes, finds new problems, finds solutions. It’s mental housekeeping. (It’s great fun, but watch out when you’re in company!)

- **PRACTICE** gets you started. It's essential because you are learning a skill, how to *do* something.
- **USE** makes your skill part of you. Using a skill to do something important to you connects it to your emotions, makes the skill important, nails it down.
- **DISCUSSION** opens you up to new knowledge, tests your knowledge against objections, increases confidence.
- **REFLECTION** refines and simplifies your knowledge, makes it more understandable, ties it in with the rest of your brain.

Use and practice take time. You can't do them in the 'revision period', on the night before an exam. They need time in the lab, ***doing*** it, experimenting, succeeding, failing. Trial-and-error learning.

Discussion is part of the course (tutorials, labs), but it's also what you do with your friends over lunch, in the corridor, in the coffee break, in the bar. Reflection is private: you can do it in the bath, on the bus, in the park, in the light, in the dark – anywhere.

'Revision' – memorising lecture notes, spotting questions, learning examples – only works in subjects full of meaningless isolated facts; we have very few of those in Computer Science.

(But reflection is true revision, and the best revision technique of all.)

A plagiarist's progress (1)

(This is a story. It's not about you. Plagiarism (play-jer-ism) is cheating by copying.)

A is in a programming class of 200. The teachers don't know his/her name. *A* has a piece of coursework to do; *A* hasn't done it.

A needs the marks, but there isn't time before the deadline to do the work.

A decides to ask friend *B* for help. Just to get an idea where to start; just like looking in a book.

A finds him/herself in a copyist's paradise. It's *really easy* to copy computer files, usually without anybody knowing about it.

A changes a few things in *B*'s work – maybe the layout, maybe the names of the methods – and hands it in with *A*'s name on it.

A goes back to bed.

A plagiarist's progress (2)

Who's been hurt? *B* doesn't lose any marks, and *A* gets some for free.

It seems unlikely *A* will be spotted (it takes $200 \times 200/2 = 20$ thousand comparisons to check everybody's work against everybody else's).

A is very unlucky, isn't caught, and gets hooked. He/she spends more time in bed, and less time at uni. *A* copies more and more courseworks, and is never caught.

With three weeks before the exam, *A* starts to 'revise' – but in *A's* case, that means learning it all from scratch.

Teaching yourself to program in three weeks isn't possible. Not enough practice; no time for use or reflection; no discussion.

So *A* fails! Then he/she buys a computer and a book, and gets ready for the resit exams ...

So what, exactly?

Somebody copied; we didn't catch them. Was there a crime? Who was hurt?

A was hurt. He/she failed the summer exam; 90% chance of failing the resit exam as well – and then it's goodbye, usually for ever. The plagiarist has wasted a year of their life.

Even if *A* is lucky in the resit, 100% chance that they will struggle in the second year, and a good chance that they will fail. Then they have wasted *two* years of their life.

Or maybe they get away with it for three years. They just manage to get a degree, but their employer finds out that they can't actually do anything (except copy other people's work).

Even with a degree, plagiarists can't hold a good job. They've wasted *three* years of their life, missed their chance at education, and it's too late to get a second try.

Plagiarism deprives the plagiariser of an education. Plagiarism is "*stealing the thoughts or writings of others, and giving them out as your own*". Plagiarists never develop thoughts of their own. Learning how to think new thoughts is what education, especially university education, is all about.

(Plagiarists don't see it like that. They see free marks and an easy life.)

Punishment for plagiarists?

Plagiarists hurt themselves. Should we punish them as well?

If people can graduate without working, without learning what they should, what's a degree worth? Employers can't trust that kind of degree.

Plagiarism affects the gold standard which you hope will get you a job. Plagiarists hurt the honest student: they hurt you.

Plagiarists destroy standards, increase failure rates, depress the standing of the university.

So: we are going to catch as many as we can. We'll use technology to make those 20 thousand comparisons. We'll use our ingenuity to beat the tricks that they use. For your sake and for theirs.

And: when we catch a plagiarist, we shall be as severe as we can be, to protect the standards of the university. For everybody's sake.

How you will be tempted

The temptation to plagiarise is always there. It arises whenever there seems to be too much work to do.

It's made worse because most people don't tell each other the truth about their problems. Some they hide; some they exaggerate.

Some people pretend they aren't doing any work; some pretend they are working like mad.

At some point you will think at least one of these thoughts:

- I don't understand this stuff, but everybody else does. I'm lost!
- None of my friends understands this stuff. It's too hard!

Then you might think "If I just cheated on the next coursework, surely I could catch up later!"

Plagiarism beckons! Can you resist it?

How to resist temptation

If you are lost or confused or frightened or bored, you've got a problem. Here's how to deal with it.

0. Don't panic! Anybody can have problems. Most people do, at one time or another.

1. Your resources are your colleagues (your fellow students, your friends) the teaching assistants, your tutor, your lecturers.

2. If your friends really do understand something, let them teach it to you! It will do them good (teaching is advanced reflection). You will probably find you aren't as lost as you thought you were. Maybe you will end up teaching them something.

3. Asking for help is smart. Don't be afraid to seek it out. Use tutorials; use exercise classes; talk to us after lectures; try teaching assistants in the labs. We're here to help: use us; get help.

4. Don’t despair. This stuff is hard (that’s what university is for) but it isn’t impossible. I’ve been teaching it for decades, and people ***do*** get through, even when they think they won’t.

(But you still have to do the climbing. Your job is learning how to learn. Push on!)

(So we may help you by asking questions rather than by giving answers. It might seem heartless at first, but it’s the only way.)

I may be old, but I'm not daft

I know that I'm advising you not to be a 'student'. 'Students' - especially first-year 'students' – drink a lot, go to lectures only sometimes, avoid labs, do as little work as possible, *never* talk to their lecturers, copy if they can, and revise like mad in the last week before the exam.

That strategy works – in some subjects.

Our subject – yours and mine – is ***different***. Intellectual skills take time to learn. ***We*** have to put the hours in, especially when we're programming.

Students (and 'students') like to avoid work. I claim that my way of learning – use, practice, discuss, reflect – is ***less work in the end***.

We lend you the machinery to practice on: the ITL is open 16 hours a day, stuffed full of computers. If you stay late, or come early, you can get all the practice you need. (And reflect on the bus home.)

An invitation

I love my subject. My colleagues love it too. We want you to succeed. We invite you to join us in our little world.

Entry is easy: in your first year you just play at programming with a computer 8 hours a day or so, for 24 weeks or so.

(The rest of your time would be your own, if it wasn't for the fact that programming gets under your skin and you find yourself thinking about it all the time, even in the shower ...)

Once you've learned to program, it all seems easy. But to get started needs magic, and the magic only works if you treat learning as play.

Come on in! The subject's lovely!