Undergraduation

March 2005(Parts of this essay began as replies to students who wrote to  
me with questions.)Recently I've had several emails from computer science  
undergrads asking what to do in college. I might not  
be the best source of advice, because I was a philosophy major in  
college. But I took so many CS classes that most CS majors thought  
I was one. I was certainly a hacker, at least.HackingWhat should you do in college to become a   
good hacker? There are two  
main things you can do: become very good at programming, and learn  
a lot about specific, cool problems. These turn out to be equivalent,  
because each drives you to do the other.The way to be good at programming is to work (a) a lot (b) on hard  
problems. And the way to make yourself work on hard problems is  
to work on some very engaging project.  
Odds are this project won't be a class assignment. My friend Robert  
learned a lot by writing network software when he was an  
undergrad. One of his projects was to connect Harvard to the  
Arpanet; it had been one of the original nodes, but by 1984 the  
connection had died. [1] Not only was this  
work not for a class, but because he spent all his time on it  
and neglected his studies, he was kicked out of  
school for a year. [2] It all evened out in the end, and now he's  
a professor at MIT. But you'll probably be happier if you don't  
go to that extreme; it caused him a lot of worry at the time.Another way to be good at programming is to find other people who  
are good at it, and learn what they know. Programmers tend to sort  
themselves into tribes according to the type of work they do and  
the tools they use, and some tribes are   
smarter than others. Look  
around you and see what the smart people seem to be working on;  
there's usually a reason.Some of the smartest people around you are professors. So one way  
to find interesting work is to volunteer as a research assistant.  
Professors are especially interested in people who can solve tedious  
system-administration type problems for them, so that is a way to  
get a foot in the door. What they fear are  
flakes and resume padders. It's all too  
common for an assistant to result in a net increase in work. So  
you have to make it clear you'll mean a net decrease.Don't be put off if they say no. Rejection is almost always less  
personal than the rejectee imagines. Just move on to the next.  
(This applies to dating too.)Beware, because although most professors are smart, not all of them  
work on interesting stuff. Professors have to publish novel results  
to advance their careers, but there is more competition in more  
interesting areas of research. So what less ambitious professors  
do is turn out a series of papers whose conclusions are novel because  
no one else cares about them. You're better off avoiding these.I never worked as a research assistant, so I feel a bit dishonest  
recommending that route. I learned to program by writing stuff of  
my own, particularly by trying to reverse-engineer Winograd's  
SHRDLU. I was as obsessed with that program as a mother with a new baby.Whatever the disadvantages of working by yourself, the advantage  
is that the project is all your own. You never have to compromise  
or ask anyone's permission, and if you have a new idea you can just  
sit down and start implementing it.In your own projects you don't have to worry about novelty (as  
professors do) or profitability (as businesses do). All that matters  
is how hard the project is technically, and that has no correlation  
to the nature of the application. "Serious" applications like   
databases are often trivial and dull technically (if you ever suffer  
from insomnia, try reading the technical literature about databases)  
while "frivolous" applications like games are often very sophisticated.  
I'm sure there are game companies out there working on products  
with more intellectual content than the research at the  
bottom nine tenths of university CS departments.If I were in college now I'd probably work on  
graphics: a network game, for example, or a tool for 3D animation.  
When I was an undergrad there weren't enough cycles around to make  
graphics interesting, but it's hard to imagine anything more fun  
to work on now.MathWhen I was in college, a lot of the professors believed (or at least  
wished) that   
computer science was a branch of math. This idea was  
strongest at Harvard, where there wasn't even a CS major till the  
1980s; till then one had to major in applied math. But it was  
nearly as bad at Cornell. When I told the fearsome Professor Conway  
that I was interested in AI (a hot topic then), he told me I should  
major in math. I'm still not sure whether he thought AI required  
math, or whether he thought AI was nonsense and that majoring in  
something rigorous would cure me of such stupid ambitions.In fact, the amount of math you need as a hacker is a lot less   
than most university departments like to admit. I don't think you   
need much more than high school math plus a few concepts from the  
theory of computation. (You have to know what an n^2 algorithm is  
if you want to avoid writing them.) Unless you're planning to write  
math applications, of course. Robotics, for example, is all math.But while you don't literally need math for most kinds of hacking,  
in the sense of knowing 1001 tricks for differentiating formulas,   
math is very much worth studying for its own sake. It's a   
valuable source of metaphors for almost any kind of work.[3] I wish   
I'd studied more math in college for that reason.Like a lot of people, I was mathematically abused as a child. I   
learned to think of math as a collection of formulas that were  
neither beautiful nor had any relation to my life (despite attempts  
to translate them into "word problems"), but had to be memorized   
in order to do well on tests.One of the most valuable things you could do in college would be  
to learn what math is really about. This may not be easy, because  
a lot of good mathematicians are bad teachers. And while there are  
many popular books on math, few seem good. The best I can think  
of are W. W. Sawyer's. And of course Euclid. [4]EverythingThomas Huxley said "Try to learn something about everything and   
everything about something." Most universities aim at this  
ideal.But what's everything? To me it means, all that people  
learn in the course of working honestly on hard problems. All such   
work tends to be related, in that ideas and techniques from one   
field can often be transplanted successfully to others. Even others  
that seem quite distant. For example, I write   
essays the same way  
I write software: I sit down and blow out a lame version 1 as fast  
as I can type, then spend several weeks rewriting it.Working on hard problems is not, by itself, enough. Medieval   
alchemists were working on a hard problem, but their approach was   
so bogus that there was little  
to learn from studying it, except possibly about people's ability   
to delude themselves. Unfortunately the sort of AI I was trying   
to learn in college had the same flaw: a very hard problem, blithely  
approached with hopelessly inadequate techniques. Bold? Closer   
to fraudulent.  
The social sciences are also fairly bogus, because they're so much   
influenced by intellectual fashions. If a   
physicist met a colleague  
from 100 years ago, he could teach him some new things; if a psychologist  
met a colleague from 100 years ago, they'd just get into an  
ideological argument.  
Yes, of course, you'll learn something by taking a  
psychology class. The point is, you'll learn more by taking  
a class in another department.The worthwhile departments, in my opinion, are math, the hard  
sciences, engineering, history (especially economic and social   
history, and the history of science), architecture, and the classics.  
A survey course in art history may be worthwhile. Modern literature  
is important, but the way to learn about it is just to read. I  
don't know enough about music to say.You can skip the social sciences, philosophy, and the various  
departments created recently in response to political pressures.  
Many of these fields talk about important problems, certainly. But  
the way they talk about them is useless. For example, philosophy   
talks, among other things, about our obligations to one another;   
but you can learn more about this from a wise grandmother or E. B.  
White than from an academic philosopher.I speak here from experience. I should probably have been offended   
when people laughed at Clinton for saying "It depends on what the   
meaning of the word 'is' is." I took about five classes in college  
on what the meaning of "is" is.Another way to figure out which fields are worth studying is to   
create the dropout graph. For example, I know many people   
who switched from math to computer science because they found math   
too hard, and no one who did the opposite. People don't do hard  
things gratuitously; no one will work on a harder problem unless   
it is proportionately (or at least log(n)) more rewarding. So  
probably math is more worth studying than computer science. By  
similar comparisons you can make a graph of all the departments in  
a university. At the bottom you'll find the subjects with least   
intellectual content.If you use this method, you'll get roughly the same answer I just   
gave.Language courses are an anomaly. I think they're better considered  
as extracurricular activities, like pottery classes. They'd be far  
more useful when combined with some time living in a country where   
the language is spoken. On a whim I studied Arabic as a freshman.  
It was a lot of work, and the only lasting benefits were a weird   
ability to identify semitic roots and some insights into how people  
recognize words.Studio art and creative writing courses are wildcards. Usually   
you don't get taught much: you just work (or don't work) on whatever  
you want, and then sit around offering "crits" of one another's  
creations under the vague supervision of the teacher. But writing and  
art are both very hard problems that (some) people work honestly  
at, so they're worth doing, especially if you can find a good  
teacher.JobsOf course college students have to think about more than just  
learning. There are also two practical problems to consider: jobs,  
and graduate school.In theory a liberal education is not supposed to supply job training.  
But everyone knows this is a bit of a fib. Hackers at every college  
learn practical skills, and not by accident.What you should learn to get a job depends on the kind you want.  
If you want to work in a big company, learn how to hack   
Blub on  
Windows. If you want to work at a cool little company or research   
lab, you'll do better to learn Ruby on Linux. And if you want to   
start your own company, which I think will be more and more common,  
master the most powerful tools you can find, because you're going  
to be in a race against your competitors, and they'll be your horse.There is not a direct correlation between the skills you should   
learn in college and those you'll use in a job. You should aim   
slightly high in college.In workouts a football player may bench press 300 pounds, even  
though he may never have to exert anything like that much force in  
the course of a game. Likewise, if your professors try to make you  
learn stuff that's more advanced than you'll need in a job, it may  
not just be because they're academics, detached from the real world.  
They may be trying to make you lift weights with your brain.The programs you write in classes differ in three critical ways  
from the ones you'll write in the real world: they're small; you  
get to start from scratch; and the problem is usually artificial   
and predetermined. In the real world, programs are bigger, tend   
to involve existing code, and often require you to figure out what   
the problem is before you can solve it.You don't have to wait to leave (or even enter) college to learn   
these skills. If you want to learn how to deal with existing code,  
for example, you can contribute to open-source projects. The sort  
of employer you want to work for will be as impressed by that as   
good grades on class assignments.In existing open-source projects you don't get much practice at  
the third skill, deciding what problems to solve. But there's   
nothing to stop you starting new projects of your own. And good  
employers will be even more impressed  
with that.What sort of problem should you try to solve? One way to answer  
that is to ask what you need as a user. For example, I stumbled  
on a good algorithm for spam filtering because I wanted to stop   
getting spam. Now what I wish I had was a mail reader that somehow  
prevented my inbox from filling up. I tend to use my inbox as a  
todo list. But that's like using a screwdriver to open  
bottles; what one really wants is a bottle opener.Grad SchoolWhat about grad school? Should you go? And how do you get into a   
good one?In principle, grad school is professional training in research, and  
you shouldn't go unless you want to do research as a career. And   
yet half the people who get PhDs in CS don't go into research.  
I didn't go to grad school to become a professor. I went because   
I wanted to learn more.So if you're mainly interested in hacking and you go to grad school,  
you'll find a lot of other people who are similarly out of their   
element. And if half the people around you are out of their element in the  
same way you are, are you really out of your element?There's a fundamental problem in "computer science," and it surfaces  
in situations like this. No one is sure what "research" is supposed to be.   
A lot  
of research is hacking that had to be crammed into the form of an  
academic paper to yield one more quantum of publication.So it's kind of misleading to ask whether you'll be at home in grad  
school, because very few people are quite at home in computer  
science. The whole field is uncomfortable in its own skin. So  
the fact that you're mainly interested in hacking shouldn't deter   
you from going to grad school. Just be warned you'll have to do a lot of stuff   
you don't like.Number one will be your dissertation. Almost everyone hates their  
dissertation by the time they're done with it. The  
process inherently tends to produce an unpleasant result, like a cake made out  
of whole wheat flour and baked for twelve hours. Few dissertations   
are read with pleasure, especially by their authors.But thousands before you have suffered through writing a dissertation.  
And aside from that, grad school is close to paradise. Many people  
remember it as the happiest time of their lives. And nearly all  
the rest, including me, remember it as a period that would have   
been, if they hadn't had to write a dissertation. [5]The danger with grad school is that you don't see the scary part  
upfront. PhD programs start out as college part 2, with several  
years of classes. So by the time you face the horror of writing a   
dissertation, you're already several years in. If you quit now,  
you'll be a grad-school dropout, and you probably won't like that  
idea. When Robert got kicked out of grad school for writing the  
Internet worm of 1988, I envied him enormously for finding a way out  
without the stigma of failure. On the whole, grad school is probably better than most alternatives. You meet a   
lot of smart people, and your glum procrastination will at least   
be a powerful common bond. And of course you have a PhD at the  
end. I forgot about that. I suppose that's worth something.The greatest advantage of a PhD (besides being the union card of  
academia, of course) may be that it gives you some baseline confidence.  
For example, the Honeywell thermostats in my house have the most  
atrocious UI. My mother, who has the same model, diligently spent  
a day reading the user's manual to learn how to operate hers. She  
assumed the problem was with her. But I can think to myself "If  
someone with a PhD in computer science can't understand this  
thermostat, it must be badly   
designed."If you still want to go to grad school after this equivocal  
recommendation, I can give you solid advice about how to get in.   
A lot of my friends are CS professors now, so I have the inside  
story about admissions. It's quite different from college. At  
most colleges, admissions officers decide who gets in. For PhD  
programs, the professors do. And they try to do  
it well, because the people they admit are going to be working for  
them.Apparently only recommendations really matter at the best schools.  
Standardized tests count for nothing, and grades for little. The  
essay is mostly an opportunity to disqualify yourself by saying   
something stupid. The only thing professors  
trust is recommendations, preferably from people they know. [6]So if you want to get into a PhD program, the key is to impress  
your professors. And from my friends who are professors I know   
what impresses them: not merely trying to impress them. They're  
not impressed by students who get good grades or want to be their  
research assistants so they can get into grad school. They're  
impressed by students who get good grades and want to be their   
research assistants because they're genuinely interested in the   
topic.So the best thing you can do in college, whether you want to get  
into grad school or just be good at hacking, is figure out what you  
truly like. It's hard to trick professors into letting you into  
grad school, and impossible to trick problems into letting you solve  
them. College is where faking stops working. From this point,  
unless you want to go work for a big company, which is like reverting  
to high school, the only way forward is through doing what you   
love.Notes  
[1] No one seems to have minded, which shows how unimportant  
the Arpanet (which became the Internet) was as late as  
1984.[2] This is why, when I became an employer, I didn't care  
about GPAs. In fact, we actively sought out people   
who'd failed out of school. We once put up posters around Harvard  
saying "Did you just get kicked out for doing badly in your classes  
because you spent all your time working on some project of your   
own? Come work for us!" We managed to find a kid who had been,   
and he was a great hacker.When Harvard kicks undergrads out for a year, they have to get jobs.  
The idea is to show them how awful the real world is, so they'll   
understand how lucky they are to be in college. This plan backfired  
with the guy who came to work for us, because he had more fun than  
he'd had in school, and made more that year from stock options than  
any of his professors did in salary. So instead of crawling back  
repentant at the end of the year, he took another year off and went  
to Europe. He did eventually graduate at about 26.[3] Eric Raymond says the best metaphors for hackers are  
in set theory, combinatorics, and graph theory.Trevor Blackwell reminds you to take math classes intended for math majors.  
"'Math for engineers' classes sucked mightily. In fact any 'x for  
engineers' sucks, where x includes math, law, writing and visual  
design."[4] Other highly recommended books: What is Mathematics?, by  
Courant and Robbins; Geometry and the Imagination by Hilbert and   
Cohn-Vossen.  
And for those interested in graphic design,  
Byrne's Euclid.  
[5] If you wanted to have the perfect life, the thing to do would  
be to go to grad school, secretly write your dissertation in the  
first year or two, and then just enjoy yourself for the next three  
years, dribbling out a chapter at a time. This prospect will make  
grad students' mouths water, but I know of no one who's had the  
discipline to pull it off.[6] One professor friend says that 15-20% of the grad students they  
admit each year are "long shots." But what he means by long shots  
are people whose applications are perfect in every way, except  
that no one on the admissions committee knows the professors who  
wrote the recommendations.So if you want to get into  
grad school in the sciences, you need to go to college somewhere with  
real research professors. Otherwise you'll seem a risky bet  
to admissions committees, no matter how good you are.Which implies  
a surprising but apparently inevitable consequence:  
little liberal arts colleges are doomed.  
 Most smart  
high school kids at least consider going into the sciences, even  
if they ultimately choose not to.  
Why go to a college that limits their options?Thanks to Trevor Blackwell, Alex Lewin, Jessica Livingston,  
Robert Morris, Eric  
Raymond, and several   
anonymous CS professors   
for reading drafts of this, and to the students whose questions  
began it.