Contents

1 Overview

In this document, I plan to give some insight into my college journey as well as some general advice for making the most out of your college career.

Note: A lot of the material will be biased towards Computer Science (where applicable). Additionally, **everything from this point on is *my opinion.***

Note: This document is pretty long. For convenience, the topics are ordered from most to least important. The blue Example boxes are less important than the plaintext.

1.1 Introduction

I am a transfer student (from IVC and Saddleback College) attending UCLA, majoring in Computer Science, and graduating in the Spring of 2024. I will be a Master's student at UCLA starting the Fall of 2024 studying Computer Science.

2 Choosing a College

2.1 Ranking (Name Brand)

A college's ranking (usually) ***does not matter***. Unless you want to attend a school for the brand name (which is fair), do more research into the field(s) you are interested in!

Example

The University of Illinois Urbana-Champaign (UIUC) is ranked 35^{th} overall but 5^{th} (above Cornell, Princeton, and UCLA!) in Computer Science.

While a college's rank gives a general heuristic for the quality their education, it should not dictate where you go.

2.2 Major Choice

If you do not know what you want to study in college yet, that is ***okay***! I ***do not*** recommend applying for an "easy to get in" major with a plan to switch once you are there. Some departments do not allow for students within the school to change into a subset of majors.

Example

It is (allegedly) extremely difficult to transfer into EECS at UC Berkeley from another major. At UCLA, if you are a transfer student and get admitted for a major that is outside the School of Engineering, they do not allow you change into it.

2.3 Criteria

Here are a few things you may want to consider before deciding where to apply (or attend). The criteria for choosing a college differs from person to person. Some questions to get you started are

→ What *exactly* do you want to study?

→ What do you want to get out of college?

→ Are you planning on going to graduate school?

We take a more in-depth look into these questions here.

What *exactly* do you want to study? This may be simple to answer, but remember to consider *all* of your options. Even if you think you know what you want to study, be sure to explore the nuances of your major!

Example

Suppose you want to study Mathematics. The question then becomes "What branch of Mathematics?". You may want to pursue a Pure Mathematics degree if the answer is theory. If you want to research topics in industry, then you may want to major in Applied Mathematics. If you want to study the foundations of Mathematics, then maybe a Philosophy degree is more appropriate.

What do you want out of college? This question is pretty straightforward. "I want to go to college to make money" is a respectable answer. So is "I want to make friends"! As long as you are honest with yourself, any answer is the correct answer.

Are you planning on going to graduate school? Depending on your major and career goals, graduate school may be an option or required. Even if you aren't planning on going to graduate school, I recommend keeping it open as an option. Your opinion might change as you go through college!

Example

A graduate degree in Computer Science is usually not required for an entry level job. However, a graduate program in Computer Science typically dives more into the theory of computation as well as provides a structured environment to learn or specialize in particular branches of the discipline.

3 General Advice for Education

3.1 Courses and Course Load

Take classes that are interesting to *you*. Explore outside of your major and see what piques your interest (even if it doesn't *directly* help with your major)!

Example

One of my friends is a Cognitive Science major and is taking (hard) Computer Science electives because he was interested in the similarities of how humans and computers think and learn.

Take a managable course load! "Managable" means something different to everybody, so use your best judgement.

Example

Throughout community college and the first year and a half at UCLA, I was working an average of 72 hours a week and taking four or more major-related classes a semester/quarter. It was not the best for my sleep schedule and I wasn't able to focus as well in class. Once I quit my job, the grass got greener, the sky turned blue, and I was able to engage in the course material.

3.2 Professors

Talk to your professors! It sounds trivial but you would be surprised how empty some office hours can be. Professors are experts in their field, and are a great resource and mentors. Who knows, maybe they will offer you opportunities!

Example

I got to know my Programming Languages professor very well, and we now climb together! I even presented an advanced topic in data structures as a guest lecturer for his introductory Data Structures class.

Example

I got to know Professor Paul Eggert pretty well, who is a well-respected developer in the open source community and a distinguished professor. He has made significant contributions to the Linux kernel, particularly in GNU `coreutils`, helped develop Emacs (a text editor similar to Vim), and currently maintains the timezone database (tz) backed by ICANN. I learned so much from his (lower division) Software Construction. He is extremely knowledgeable about the Unix system and systems computing in general (which is one of my many interests).

3.3 Grades

While grades are important, it isn't as important as actually *knowing* the material. This rings true especially for major courses. I recommend prioritizing learning the material over chasing high grades. Typically, if you know the material, good grades will follow. Quoting my professor,

> "If I didn't have to give out grades, I wouldn't. Only *you* know how much you have grown throughout this course. Your grade is merely a reflection of three 2 hour *slivers* of time during the exams; it does not capture the full learning process."
>
> – Professor Alexander Sherstov

3.4 Aside: My Academic Journey

I was a very bad student up to and including high school. I rarely did homework in a timely manner, often doing it the period before it was due. I can't remember a time I did homework at home or studied for an exam. I didn't take any AP or Honors classes until junior year, and even then, it was because all of my friends were taking them. Out of high school, I was rejected from almost every UC and CSU that I applied to, and so I was (in a way) "forced" to go to IVC (and eventually Saddleback).
At community college, I met some brilliant instructors and professors, and I was very fortunate study under them. The slower pace of CC gave me an opportunity to mature and figure out what I wanted to study at university. I was able to maintain a good GPA throughout CC and got into every UC and CSU I applied to at the end of my second year. I ended up choosing to go to UCLA because it was comparatively cheaper than Berkeley.

This anecdote is here to (hopefully) alleviate some worry about getting into a university right out of high school, or not knowing what you want to study. I took a pretty nontrivial path to university, but it definitely worked out for the better.

~ The rest of this page is intentionally left blank. ~

4 The UC System

You were probably told that UC's are more research oriented and CSU's are more applied. While that certainly is true, there are some things you should keep in mind. Since I only have experience with UC's, I will focus the conversation on them.

4.1 Professors at a UC:

Depending on the department, most professors at a UC are hired for research and not for teaching. This is one of the reasons why you tend to learn topics in a more theoretical context at UC's as opposed to CSU's.

4.2 Coursework at a UC:

To build atop of the last point, coursework is typically more theoretical. So, even if you study an applied science, you may learn more theory than you expect.

Example

Most upper-division Computer Science courses at UCLA are *very* theoretical. The amount of actual coding I do for coursework is minimal. I often write out homework with pen and paper, since most of the questions assigned are not programming tasks, but problem solving/proof-based questions. The coding I did do for assignments typically did not directly relate to the course material. However, *because* of my theory courses, I now better understand the (seemingly infinite) web of Computer Science. I am able to solve new and unique problems by applying, in part, the theory I learned as well as my general software engineering skills.

One *very* important note: You should ***not*** expect to learn "everything" from your classes. In order to get the most out of the course, you should be researching topics outside the scope of the course. With how quickly the quarters pass by (unless you're at UC Berkeley), there simply isn't enough time to teach everything about a course topic in just ten weeks. Therefore, things *will* be left out.

~ The rest of this page is intentionally left blank. ~

5 Opinions

This section details my ***personal outlook*** on education.

5.1 Theoretical Coursework

I believe that the theory behind your field is the most important thing you can learn. This is because I believe that it is important to know *why* you are doing the things you are, not just *how*. With the sheer amount of information on the internet, you are objectively able to learn *how* something works, or *how* to do a job (*especially* now due to tools like ChatGPT). However, a formal course provides a structured environment to learn *why* these things work. They will most definitely ***not*** be applicable to day-to-day life, but the process of learning theory gives you an insight on how to solve new, creative problems.

Example

I took abstract math courses (Algebra, Number Theory, and Set Theory) that outline the foundations of the Mathematics we are familiar with. While they don't directly make me better at Math or Computer Science, it gave me very important insight into how I can solve new problems with the tools I am given. For example, Number Theory and Ring Theory are heavily used in the field of Cryptography. The RSA encryption algorithm is built using concepts from Ring Theory. Set Theory is heavily used in Database and Programming Language Theory, which gives me insight into the limits of computation in those disciplines.

5.2 Purpose of College

I believe the purpose of college is to immerse yourself into a field of study that ***you enjoy***. You often hear people say "I don't use anything I learned from college at my job", which may or may not be true. However, education and industry typically have disjoint philosophies. Education typically requires you to think rigorously about a question. Industry (typically) just wants things to get done. You are given (roughly) four years to explore the topic(s) of your choice, so you should take advantage of it!

∼ The rest of this page is intentionally left blank. ∼

5.3 Ancecdote: Professors

In my opinion, a truly captivating professor has the ability to change your entire outlook on education, which was what happened to me. I originally attended college just to get a job, and so I tried to rush through it by taking as many courses per quarter as I could. However, my entire outlook changed after taking ***one*** course (Theory of Computation). It was the first time I had met a professor so passionate about not only his research, but teaching as well. He is the sole reason I decided to pursue graduate school.

Example

Professor Alexander Sherstov is a brilliant professor, and has an unparalleled passion for teaching. His Theory of Computation course (traditionally one of the most challenging courses in any Computer Science curriculum) was made intuitive only because we were being taught by someone who had a true passion for teaching. His teaching philosophy has no doubt transformed generations of students.

> "I firmly believe that research and teaching are completely disjoint skills. A lot of times students, faculty, and administration believe that those who conduct important and compelling research can naturally teach subjects well, but this really is not true. A teacher must really keep the students' best interest and understanding in mind, but in many cases, professors do not necessarlily care about this since their first and foremost interest is their research. Even textbooks are often written from a researcher's perspective rather than a student's perspective; and again, the student loses."
>
> – Professor Alexander Sherstov

Towards the end of the quarter, he said something that really stuck with me, and it changed my outlook on college as a whole.

> "We started with nothing but first principles, and worked our way up to Turing machines! It has been an absolute joy to walk this journey with you all these past ten weeks. Now, you know the theory behind what we call *computation*, and no one can take that away from you. You should all be proud of yourselves; this course is the most difficult course in the Computer Science curriculum here at UCLA. No matter how much you *think* you have learned, the reality of it is that you have learned so much more."
>
> – Professor Alexander Sherstov

∼ The rest of this page is intentionally left blank. ∼

The personal anecdote he gave in his last lecture was also very wholesome.

> "About twenty years ago I was in your shoes, and when I learned about undecidability for the first time, and it shook me to my core. For a few days, food didn't taste the same, I couldn't sleep well, and life really sucked. I thought *How can this be? I have a theorem that tells me that the halting problem is undecidable?* Surely, if someone gives me some computer program, I'm going to roll up my sleeves, dig in, and find a solution. It may take some time, but if I put my mind to it, I'm going to find out if the Turing machine halts on that program.
>
> It really shook me because there's this cognitive dissonance between what you *feel* should be true and what you are *told* is true. What makes things worse is that there's someone telling you that something is *impossible*. When someone tells you that you cannot do something, all of a sudden, for some weird reason, you *really* want to do it, even if you weren't interested before.
>
> I wanted to resolve this seeming contradiction, and eventually, I found a way. Here's the thing: the theorem is right, but I'm *also* right in my own way. The theorem says we cannot decide the halting problem on *all* inputs, and that's fine. I never claimed that I would live forever to solve infinitely many inputs.
>
> Humans can only solve finitely may inputs by definition; you get to pick which inputs you solve, but it's always a finite number. *That* is the distinction. The theorem doesn't say that humans cannot solve *instances* of undecidable problems, it just tells us that humans cannot solve *all* of the infinitely many instances. It's just a biological limitation, not a cognitive one.
>
> You know, sooner or later, innocence has to give way to experience. What we want is to solve an infinite language, right? We want to rake in an entire constellation. But if we thought about it really hard, we would realize that getting a hold of even a *single* star, solving a language on a *single* input, would be a lifetime achievement[1]. That would make us happier people; it certainly made me happier.
>
> As you all graduate and move on with your careers, I know the future of computing is in good hands. Thank you so much; it has been a pleasure having you in this class. I hope you can see the beauty of computation as I do, and I hope you learned something about computation."
>
> – Professor Alexander Sherstov

[1] He has solve *multiple* problems that were previously thought to be unsolvable!

6 Statistics

For those interested, these were my high school and community college statistics.

6.1 High School

Grade Point Averages

Type	Weighted	Non-weighted
Academic 9-12	4.1538	3.7179
Academic 10-12	4.3793	3.7931

Freshman Year

Course	Grade	Course	Grade
English 1A	A−	English 1B	A
Latin 1A	A−	Latin 1B	B+
Math IIA	A	Math IIA	A
Biology A	B	Biology B	B+
Cultl Gbl Age A	B	Cultl Gbl Age B	B+

Sophomore Year

Course	Grade	Course	Grade
English 2A	A	English 2B	A
Latin 2A	A	Latin 2B	A−
Math IIIA	A+	Math IIIB	A+
Chemistry A	A	Chemistry B	A
World History A	A	World History B	A

Junior Year

Course	Grade	Course	Grade
H American Lit A	A−	H American Lit B	A
H Latin 3A	A−	H Latin 3B	A−
H Precalculus A	A−	H Precalculus B	A
AP Statistics A	A	AP Statistics B	A
AP Physics 1A	C	AP Physics 1B	A−
AP US History A	B	AP US History B	A−

Senior Year

Course	Grade	Course	Grade
AP Eng Lit A	B	AP Eng Lit B	CR
AP Latin A	A−	AP Latin B	CR
AP Calc BC A	B−	AP Calc BC B	CR
AP Com Sci A	B+	AP Com Sci B	CR
AP Macro Econ	A−	AP Macro Econ	CR
Beg Ceramics A	A+	Beg Ceramics A	CR

6.2 AP Exam Scores

Exam	Score
AP Physics 1	2
AP Statistics	4
AP Calculus BC	4
AP U.S. History	3
AP Lit. and Comp.	4
AP Macroeconomics	5
AP Computer Science A	4
AP Environmental Science	3
AP U.S. Gov. and Politics	3

6.3 Community College

Grade Point Average

Type	GPA
Cumulative	4.00
Department	4.00

First Year

Course	Grade
C Programming	A
H Psychology 1	A
College Writing 2	A
Analytical Geometry/Calculus III	A

Course	Grade
H Communications	A
Java Programming	A
Physics I (Kinematics)	A
Intro. to Linear Algebra	A
Intro. to Computer Systems	A

Summer Session

Course	Grade
Intro. to Computer Science I	A
H Principles of Microeconomics	A
Physics II (Electricity & Magnetism)	A

Second Year

Course	Grade
Assembly Language I	A
H Film & US Culture	A
Discrete Mathematics I	A
Intro. to Computer Science II	A
Elementary Differential Equations	A

Course	Grade
Physics III (General)	A
Assembly Language II	A
Discrete Mathematics II	A
H Academic, Career, Life	A
Intro. to Computer Science III	A
Data Structures and Algorithms	A