Cracking the CS Job Search & Interview

Max Bernstein & Yuki Zaninovich

Who are we?

Max

- Compiler freak
- Bikes till his legs get gangrene
- Gets to go on Facebook at work
- Rumor has it he met Mr Linux himself

Yuki

- Some kind of CS nerd
- JumboCode Supreme Overlord
- Listens to ear-bleeding EDM
- “Alexa, how do I get a job?”

What we want to instill

- Confidence in your job hunt
- Comfort with networking
- Fiendish abilities with hash tables and trees

Week by week overview

1. How to meet people & apply
2. How to interview
3. How to interview (more)
4. ~ viewer's choice ~

Ask questions any time!

Where do I begin?

Ask yourself:

- What kind of work excites me?
- What kinds of industries and causes excite me?
- What kind of environment excites me?

Where do I look?

- Jobs (the artist formerly known as COMP199)
- Angel.co
- LinkedIn
- Twitter
- In-person events:
  - Career fairs
  - Hackathons
  - Meet-ups
  - Campus tech talks

Day 1: How to meet people

How to interact with employers at career fairs

- Research companies you're interested in beforehand
- Introduce yourself, keep it short (elevator pitch)
- Ask interesting and probing questions
- Express why you're excited and passionate to work *specifically for them*
- Don't try and force paper on people
- Get business card and follow-up via email

How to get the most out of hackathons

- Go in knowing *exactly* what you want out of it
  - Winning a general/API award
  - Getting a headstart on a personal project
  - Focusing on non-hacking things - workshops, networking, snagging swag
  - Making friends and building things with interesting people
- Don’t be afraid to ask for help!
- Don’t just make some bullshit project with Ethereum

How would you email someone?

A poll for the room (chatterfall).

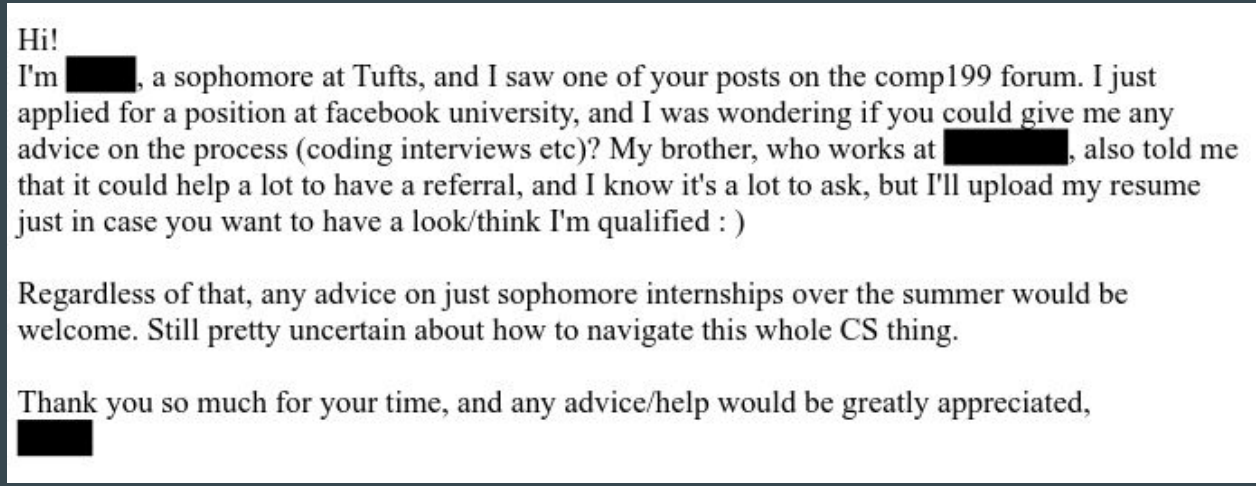
How (not) to email

Pros:

- To the point

Cons:

- Very general; no context or interesting question
- Asks for referral immediately
- Sends resume immediately



Hi!
I'm [redacted], a sophomore at Tufts, and I saw one of your posts on the comp199 forum. I just applied for a position at facebook university, and I was wondering if you could give me any advice on the process (coding interviews etc)? My brother, who works at [redacted], also told me that it could help a lot to have a referral, and I know it's a lot to ask, but I'll upload my resume just in case you want to have a look/think I'm qualified :)

Regardless of that, any advice on just sophomore internships over the summer would be welcome. Still pretty uncertain about how to navigate this whole CS thing.

Thank you so much for your time, and any advice/help would be greatly appreciated,
[redacted]

Thanks for reaching out and welcome to computer science. What follows is largely going to be unsolicited advice on communication and professional relationship building.

It helps to be more specific with your requests and also what you have/have not done already. For example:

> if you could give me any advice on the process

There's a *lot* of process. Have you ever done any technical interviews before? Have you done behavioral interviews before? Where else are you interviewing? What kind of prep have you already done? What, specifically, are you concerned about?

> My brother [...] told me that it could help a lot to have a referral

It's relatively uncommon for somebody to refer someone they don't have an existing relationship with. I often refer former students of mine, former coworkers, etc, but rarely do I refer someone I have not worked closely with before.

Even if I did do that with some frequency, it wouldn't mean much; there's a tick box for how well I know the person I am referring, and recruiters and hiring managers don't give very much weight to the "I don't know this person" box.

In that vein, attaching your resume in the first email leaves a bad taste in the mouth.

> advice on just sophomore internships ... would be welcome

Again, I really don't know what boat you're in. Would this be your first internship? What courses have you taken or are you taking right now? Do you have any inclinations toward some particular computing subfield, like algorithms or networking or ...?

Further information can be found in this recently-released (on COMP199 and elsewhere) slide deck created by some Tufts alumni from '17 and '18. I would recommend reading it.

https://bernsteinbear.com/excollege/assets/networking_101.pdf

I'm happy to be of service, but the service should be relatively specific.

Cheers,
Max

How to email

Pros:

- Gives me context on what they need help with
- Read my note on how not to email
- Asks a question they think I might be able to help with
- Doesn't treat me like a referral factory

Cons:

- A little long (but honestly, that's okay)

Hey Max,

I saw your post in the Comp199 Piazza (about bad reach out emails) and it inspired me to reach out with (a hopefully good) email of my own.

For introductions, I'm [redacted]. I studied [redacted] at Tufts, and graduated a few years behind you. I work generally in [redacted] stuff, though I do more general software engineering too. I think we've actually been in the same room a few times and definitely have some mutual friends, but I don't remember if we've actually met per se.

Anyway, on to the actual point of this email. I currently work as a solutions engineer at [redacted], working on i[redacted] (more on that later). But I'm also currently interviewing at a few other places, and have a final round interview for [redacted] scheduled in a few weeks.

So the question (at long last) is: If I get an offer from [redacted], should I take it?

A few pros and cons of [redacted] right now. [redacted] gives me a lot of work flexibility, basically allowing me to take time off whenever I want as long as I'm hitting my deadlines. The work is fairly interesting and not especially icky moral-wise, which is nice. I'm pretty good at it, and people tell me I do a good job (which is always nice to hear), and the people I interact with on a day-to-day tend to be pretty chill and friendly. The pay is probably lower than what I could theoretically be making, but tech pays an absurd amount no matter what so honestly that's not really a big concern.

But the con, the major con that makes me apply to other places, is that I'm incredibly bored. It's not an exaggeration to say I probably work an average of 25 hours a week (if that), and around 10 of those are me working on random work-relevant side projects that may or may not actually matter to the company in question. Furthermore, a lot of what I do is pretty easy python scripting; a lot of "pull from this API, push to that one" rather than production-level software engineering.

The result of this is I don't feel like I end up actually learning very much, either things that would help me in my career or things that interest me personally. This is compounded by the fact that [redacted] is somewhat stingy when it comes to professional development money (though some of that is COVID preventing any travel to cons). What I should probably do is just keep my current job and do side projects to learn with my absurd amount of free time, but I tend to do a pretty bad job of self-directed learning, and get distracted from that stuff pretty easily.

So, is it worth leaving my pretty comfy gig for something that will probably be much harder (in a good way), but risk worse culture/work life balance/etc? I doubt you have a firm yes or no answer here, but I was wondering if you had any thoughts on the subject.

If you got to the bottom of this long ramble, thanks for your time :).

Sincerely,
[redacted]

What does a resume get me?

- First glance snapshot of the applicant (you)
- Opportunity to showcase accomplishments in brief
- Good place to start conversations later in the process, like
  - Previous or current jobs or responsibilities
  - Previous or current projects
  - Previous or current hackathons
- A place to demonstrate attention to detail
- But most importantly, an interview

Resume Example 1

resume

Anthony Monaco

Gifford House, Medford, MA 02155 | 617-627-2000 | anthony.monaco@tufts.edu

Education

Harvard University - Cambridge, MA — Class of 1987
PhD in Neuro Science
Cumulative GPA: 3.71; Dean's List

Princeton University - Princeton, NJ — Class of 1981
Bachelor of Science in Computer Science
Cumulative GPA: 3.71; Dean's List

Monaco High School – Monte Carlo, Monaco — August 1973 - June 1977
International Baccalaureate Diploma, SAT (New): 1320

Related Experience

JumboCode
Member - Medford, MA — September 2017 - Present

- Developed the front-end of website using HTML and CSS
- Participates in weekly meetings

Personal Projects

Shortest Route Finder

- Created an application that allows the user to pick certain destinations and deduces the shortest route that visits all the chosen destinations

Skills

- Programming Languages: Java, C++, HTML, CSS, JavaScript
- Software: Microsoft Office, Adobe InDesign
- Spoken & Written Languages: Korean (Native language), Mandarin (Intermediate)

Work Experience

YMCA Summer School 2017
Counselor – Medford, MA — July - August 2017

- Assisted teachers with MMM (Mindful Mixed Media), CID (Culture, Identity & Diversity) and SS (Science & Society) classes, including helping students with weaker English abilities
- Facilitated daily reflection meetings of 8 students; mentored students throughout the program

Leadership Activities

HEAR (Harmony for East Asian Relationships)
Co-Founder – Tokyo, Japan — August 2015 - November 2016

- Cofounded the organization aiming to improve the relationships between Japan and Korea, and Japan and China, on the belief that people's emotional responses to past events generate the tension
- Held workshops in high schools throughout Japan to empower Japanese high school students to contemplate why this issue exits, and debate what actions they can take

Robotics Club
Co-Founder - Seoul, South Korea — August 2015 - June 2017

Resume Example 2

resume

Joe Schmoe

(123) 456-7890 • joe@schmoejoseph.com • github.com/jschmoe • schmoejoseph.com

Education

Tufts University Medford, MA · B.S. in Computer Science (Class of 2018)

Relevant Coursework Teaching Computer Science · Functional Programming · Programming Languages · Networks · Compilers · Computer Graphics · Computation Theory · Operating Systems · Concurrent Programming · Algorithms · Machine Structure

Relevant Achievements Delta Epsilon Phi (German Honor Society) · Dean's List Fall 2014, Spring 2015, Spring 2016, Fall 2016

Experience

XYZ BigCo Some City, WA (Summer 2017)
Software Engineering Intern

- Created automatic to-string function generation for types in the Reason compiler.
- Redesigned release flow for the whole Reason open-source project and led multiple releases.
- Led effort to split Reason into more easily consumable sub-packages.
- Rewrote Reason code formatter (`refmt`) command-line tool in applicative functor style.

ABC, Inc Some City, MD (Summer 2016)
Software Engineering Intern

- Decreased UX friction by improving mention feature with Mozilla's "frecency" algorithm.
- Raised issues and fixed bugs on the ABC, Inc core application.
- Contributed to jekyll/jekyll-admin (open-source content management system).

Tufts University Medford, MA (Spring 2015—Present)
Teaching Fellow, Lab Leader, Teaching Assistant

- Developed Data Structures final project specification and reference implementation.
- Implemented alternative Data Structures independent final project.
- Mentored 10 students as they designed and created their own inventive projects.
- Created new Machine Structure lab designed to help students transition from C++ to C.

Tufts Cheese Club Medford, MA (Spring 2015—Present)
Adviser, President, Executive Board Member

- Ran Mozzarella Interest Group, which hosted talks about milk, enzymes, etc.
- Organized 270 Tufts students for a 24-hour eatathon (Tufts PolySnack) and demo series.
- Built Tufts PolySnck's QR code invitation and check-in software with JS and Parse.

Lifttt Some City, HI (Summer 2015)
Software Engineering Intern

- Implemented an efficient priority and cost-based SQL query scheduler (18k queries/week).
- Wrote an amortized constant-time regular expression to string map.
- Created a timed and threaded caching and cache invalidation Python class wrapper.

Projects

- **"Intro to Caseiculture"** *(schmoejoseph.com/cheesemaking)*
  A series of blog posts that illustrate how to make cheese from the ground up
- **Koi, Hare** · C *(github.com/jschmoe/koi, github.com/harcVM/hare)*
  Several iterations on and revisions of a small RISC virtual machine; featured on Hacker News
- **Distributed Lisp interpreter** · Erlang *(github.com/jschmoe/distlisp)*
  A cluster of nodes that automatically distribute the computation of a Lisp interpreter
- **Distributed ray tracer** · C++ *(github.com/MaiaRT/tracer)*
  A hierarchical network of CPU workers that ray trace parts of a given image, including a small thread-safe networking library

Skills

- **Programming Languages** C · C++ · OCaml · Haskell · SML · Ruby · Python · JavaScript
- **Software** NGINX · Apache · MySQL · PostgreSQL · Git · Cron · Ruby on Rails · Jekyll
- **Other** Digital and film photography · road biking · cooking · German · Spanish

Solid resume guidelines

- Formatting: 1 page, .pdf
- Order matters
- Be consistent in tense
- Prioritize technical skills/accomplishments -> leadership -> other activities
  - and IMPACT
- Hyperlink wherever you can (paper & PDF)
- Don’t have too much or too little explanation
- Don’t have buzzwords unless you can back them up

What does a cover letter get me...?

- A place to talk about the company-specific opportunities that excite you, or
- A way to stand out in a way that would be strange on your resume
- Can also simply be a place to put additional relevant information

- Some bizarre / outlandish example

How do I manage my applications?

- Spreadsheet. Everything.
- Track who you talked to and when
- Track what status you are in the pipeline
- Get used to being rejected :'(

Please wait... We will begin shortly...

Day 2: How to behavioral interview

A note about side projects

Types of Interviews

Behavioral interview

Overview:

- Getting to know you more as a person and team member than as a coder
- Mostly asking what you would do in hypothetical scenarios or how you actually dealt with adversity in the past

Tips:

- Prepare responses to common questions
- Don’t fake anything

Technical interview

Overview:

- Short intro -> technical questions -> Q&A
- Data structures & algorithms
  - Sometimes really design-heavy
- Some problems are in disguise
- More specific questions if you are interviewing for a targeted role
- Different from normal coding because you must verbally explain thought process throughout
- Different from normal coding also because sometimes they are stupid problems

Tips:

- Just interview, over and over and over
- Learn time complexities of data structures
- And what they can be used for
- Everything is hash tables

Interview Modalities

Phone Interview

- Kind of awkward because you're talking to someone you've never met and trying to get a job
- Focus on being understandable when you're speaking out loud
- Use headphones
- Helpful to have a notepad near you to organize your thoughts
- If it's just Q&A (no coding), get ready for some abstract questions
    - "how would you implement malloc?"

On-site Interview

- Dress code: *Not* business casual/formal (usually)
  - “Wear whatever is comfortable!” — every tech company ever
  - Wear a little bit dressier than they recommend
- Four or five back to back interviews (45 mins — 1 hour each)
  - Mix of behavioral and technical, but mostly technical
  - Sometimes allow coding on laptop, but mostly whiteboarding
  - Take your bathroom and drink breaks
- Lunch is normally somewhere in the middle and not of import for your interview process
  - The purpose is for you to eat and ask questions with no pressure
  - But don't be an ass

Cracking the Behavioral Interview

Types of questions

- Self-introduction
  - “Tell me about yourself”
  - “What got you interested in back-end development?”
  - “What projects do you hope to work on with us?”
- Hypothetical scenarios
  - “How many tennis balls can you fit inside a 747?”
  - “Given one heavy marble and eight light marbles of identical appearance, can you figure out which is the heavy one by using a scale three times at most?”
  - “Design a refrigerator to be used underwater”
- Recalling a previous experience
  - “Tell me about a time you had a disagreement with a teammate”
  - “Tell me about a time you missed a deadline”
  - “Tell me about a time you had to sacrifice quality for speed”

Self-introduction

- Figure out your identity within the scope of tech/CS
- Have a compelling narrative about your CS journey
- Research the company in question and mention specifics

Hypothetical scenarios

- Ask clarifications before anything to make sure you understand the problem
  - Sometimes the problem is designed solely to test this ability
- Verbalize your thought process
- Write or draw to provide yourself visuals or examples
- Don’t be afraid to ask for hints, but only after you’ve made attempts

Previous experience: STAR method

- **Situation**
  - Provide some context
- **Task**
  - What did you set out to accomplish?
  - What were your options?
- **Action**
  - What did you do?
  - Why? What were the tradeoffs?
- **Result**
  - What were the consequences of your actions? Provide tangible data if possible
  - What did you learn?

How to practice

- Peer interviewing
- Cracking the Coding Interview/PM Interview books
- Your first couple interviews of the season

Behavioral Grilling!

Day 2: How to technical interview

Data Structures

Arrays

- Most rudimentary collection data structure
- Association between key-value pairs, where keys must be integers

- O(1) indexed element access
- O(log n) search **if sorted**, O(n) otherwise
- O(n) insertion/deletion

- O(n) space

Pros:

- Good cache locality
- Easy to make, minimal additional information other than your data

Cons:

- Any unused indices between 0 - [max index] is wasted space
- Layout does not give additional information about the contents of the data

Linked lists

- No indexed element access, instead they can be iterated through by following pointers between them.
- Can augment pointing structure
  - e.g. doubly-linked, circular, etc.

- O(n) search
- O(1) insertion/deletion

- O(n) space

Pro:

- Simple insertion and deletion

Con:

- Pointers is small space overhead
- Can’t perform binary search

Hash sets & tables

- Map any arbitrary *key* to some associated *value*
- Values are transformed into a key, which is used to index and find given value
- Generally ok to take their functionality for granted during coding problem, but be prepared to explain its inner workings
  - Probing, hash function, load factor, etc.

- O(1) insertion/deletion
- O(1) search *if searching by key*

- O(n) space

Pros:

- Most languages provide an easy-to-use built-in hash set/table
- Easy to use

Cons:

- Generally tricky to *implement* right
  - Delete is hard
- Many assumptions need to be met for time complexity to be true
- Higher overhead per-element (not good for small collections)

Please wait... We will begin shortly...

Day 3: How to technical interview

Tips

Ask questions

- What is the reason for this code existing -- in what context will it live?
  - e.g. "this is for a distributed system" vs "this is for a microcontroller"
- What size will the input be?
- What constraints are there?
  - e.g. "this must run in linear time"
- What assumptions can you make?
  - e.g. "this will only ever take ints"

Talk out loud

- Try and connect question parts to the concrete tools you can use to answer it
  - e.g. "To look up users, I can use a hash table"
- Connect your assumption to code simplifications
  - e.g. "Since this only takes positive integers less than 2 billion, I can use a primitive integer type"
  - e.g. "Since this only takes up to 1000 elements and microcontrollers have limited stack space, we can do a linear search"

Test your code

- Find test cases based on the problem description
  - e.g. sorting should sort [a, b] if a < b and [b, a] if a > b
- Find test cases based on your solution
  - Any place where you have a conditional, loop, etc, especially if it looks a little fragile, try and break it

Write the "dumb" solution first

- Don't go straight for the hyper-optimized graph algorithm you vaguely remember from Algorithms
- Do something that you know you can implement without too many bugs, even if it is slow
  - This mimics how things tend to work in real-world engineering teams, too
- Iteratively improve your solution once you have it working with test cases

Practice problems (small)

1. Write a function to check if a string is a palindrome
2. Given a function repeatedly called with different bytes, turn them into a list of floats
3. You are given an int array of size n with unique values ranging from 0 to (n - 1). Suppose one element is replaced with a -1. Find the value of the original element. Explore ways to do this in-place, mathematically, with an extra array, and an extra hash table.
4. Write a program that checks for brace imbalances in a string
5. Traverse a "filesystem" recursively and return a list of "files"
6. Delete the last logical element in an array (using an index)

Practice problems (large)

1. Write a spreadsheet engine
2. Write a task scheduler
3. Write malloc
4. Write a configuration file parser

Other

1. What is your favorite programming language? Language feature?
2. Tell us about an interesting piece of code
3. Tell us about an interesting bug you fixed

Technical Grilling!