BEYOND CRACKING *the* CODING INTERVIEW

Pass Tough Coding Interviews,
Get Noticed, and Negotiate Successfully

With replays & data from over 100k interviews on
interviewing.io

GAYLE LAAKMANN MCDOWELL
MIKE MROCZKA | ALINE LERNER | NIL MAMANO

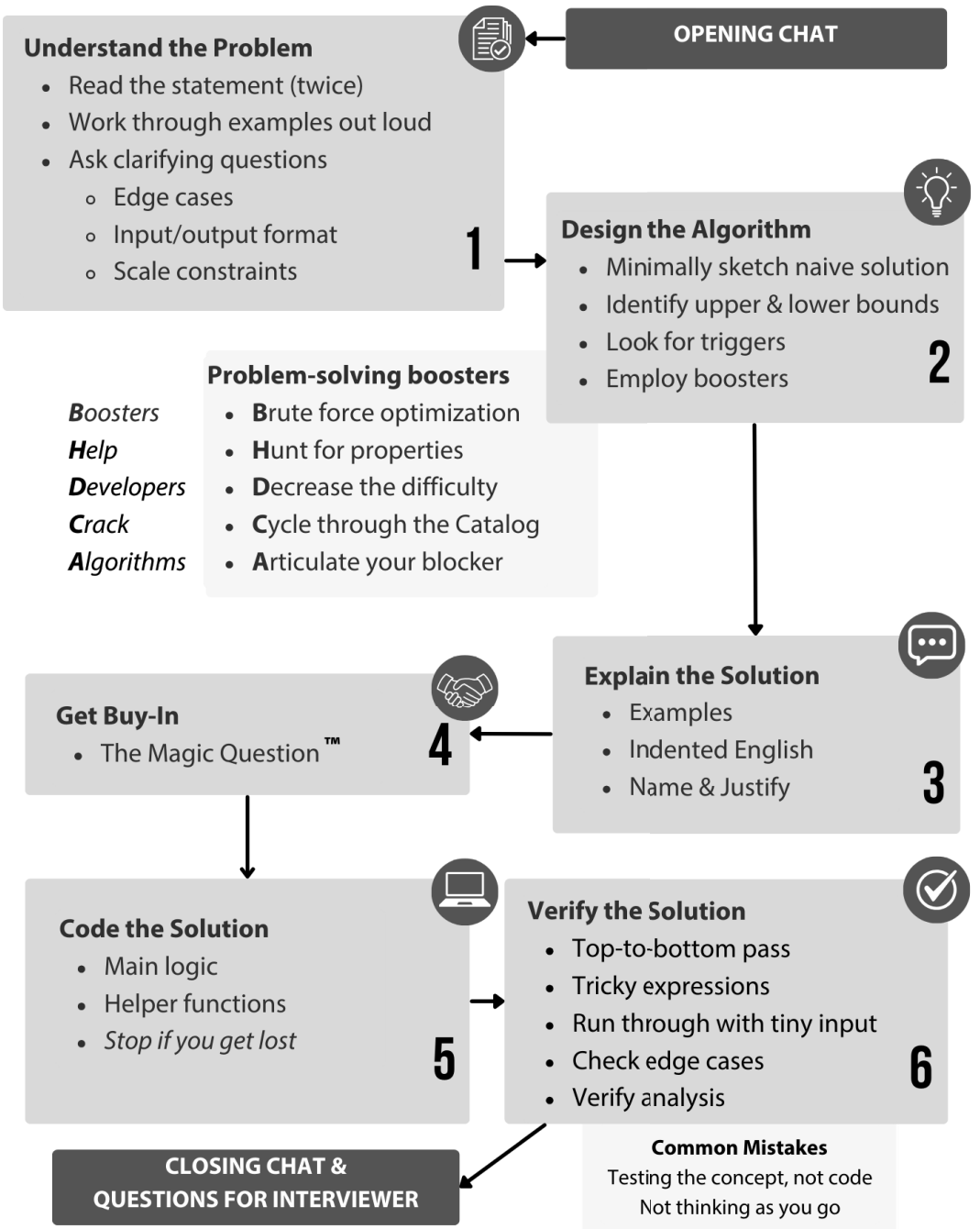
BEYOND
CRACKING
the
CODING INTERVIEW

PASS TOUGH CODING INTERVIEWS,
GET NOTICED, AND NEGOTIATE SUCCESFULLY

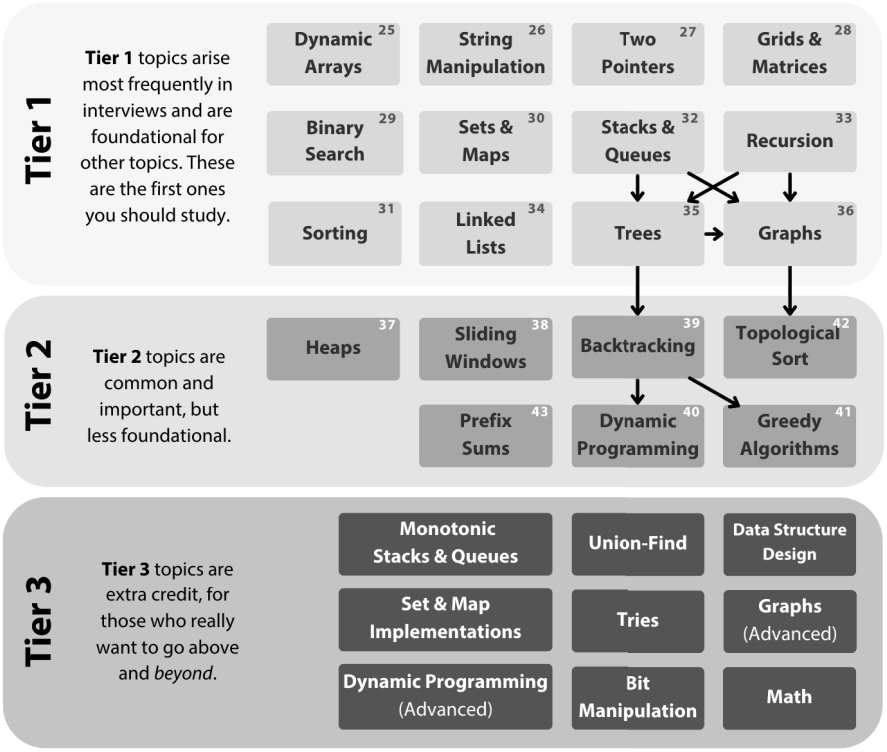
SNEAK PEEK

amazon.com/dp/195570600X

INTERVIEW CHECKLIST



STUDY PLAN



BOOSTERS

If boundary & trigger thinking don't point you to the right approach, start with the brute force

Brute Force Optimization
- Preprocessing Pattern
- Data Structure Pattern
- Skip Unnecessary Work

1

If you need a new approach...

Hunt for Properties
- DIY
- Case Analysis
- Reverse Engineer the Output Pattern
- Sketch a Diagram
- Reframe the Problem

2

If you can't find any approach...

Decrease the Difficulty
- Tackle an Easier Version
- Break Down the Problem

3

Solution might be in your blindspot

Cycle Through the Catalog
- *Think: Could ___ be useful?*

4

If you are still stuck...

Articulate Your Blocker
- Don't Say "Hint"
- Show Your Work

5

Cracking the Coding Interview
189 Programming Questions and Solutions

Cracking the PM Career
The Skills, Frameworks, and Practices to Become a Great Product Manager

Cracking the PM Interview
How to Land a Product Manager Job in Technology

Cracking the Tech Career
Insider Advice on Landing a Job at Google,
Microsoft, Apple, or Any Top Tech Company

BEYOND CRACKING *the* CODING INTERVIEW

GAYLE L. MCDOWELL
MIKE MROCZKA
ALINE LERNER
NIL MAMANO

CareerCup, LLC
Palo Alto, CA

BEYOND CRACKING THE CODING INTERVIEW

Published by CareerCup, LLC, Palo Alto, CA. Compiled Jun 3, 2025.

For more information, or to enquire about bulk or university copies, contact support@careercup.com.

Please report bugs or issues at beyondctci.com.

978-1955706001 (ISBN 13)

To my favorite coders, Davis and Tobin—
Gayle

To my dog, my wife, and our readers (and not necessarily in that order)—
Mike

To my two wonderful kids (or if I have more, then whichever two are the most wonderful)—
Aline

Als meus pares—
Nil

WHAT'S INSIDE

__INIT__()

PART I

Watch for these speech-bubble icons, which mean there is online content.

- **Interview Replay:** Recordings of actual mock interviews.
- **Snippet:** Material that you can copy/paste online, including email templates and code recipes.
- **Resource:** Bonus chapters, worksheets, and other material.
- **Problems, Solutions, and AI Interviewer:** You can try every problem in the book with the AI Interviewer. Solutions are online in four languages (Python, Java, JavaScript, and C++).

__INIT__()

README

This is a big book, and yes, it needs an instruction manual. We ask (beg?) you to read this. We'll keep it short and to the point. We also know lots of books have online materials, and they're often junk. We promise this isn't the case with ours.

> Before we do that, we'd like to address the relationship between this book and interviewing.io. You'll see a lot of references to it. This book is not *from* interviewing.io, but we do partner with them for access to lots of data, interview replays, and an AI Interviewer. Because of this relationship, we know that sometimes mentioning interviewing.io might sound promotional. We've tried to avoid that as much as we could. We hope—trust—that you'll forgive this in exchange for access to lots of data, resources, and tools (and the discount code).

The book is roughly split into two segments: the first segment (Parts I–V) is the soft squishy stuff (backed up by a lot of qualitative and quantitative data). The second segment (Parts VI and onwards) is the technical content, which has its own README (pg 168)**.** Please read it before diving into those parts.

ONLINE MATERIALS AND DISCUSSION

You can access all of our online materials and bonus chapters here:

bctci.co

Talk with the authors, get help if you're stuck, and geek out with us on Discord.

bctci.co/discord

ICONS

Whenever you see an icon wrapped in a speech bubble, it means that there's online content that goes with it, like so:

Problems, Solutions, and AI Interviewer: Each problem in the book has a Python solution (without any fancy Python-specific tricks). We also provide Java, JavaScript, and C++ solutions online. We encourage you to try each problem with AI Interviewer and to only look at the solution if you get stuck.

Interview Replay: These are recordings of actual mock interviews from interviewing.io's collection, conducted by engineers from FAANG and other top companies. We use them to showcase real-life examples of successes and mistakes.

Snippet: These are snippets that you can copy and paste. Sometimes they're code. Sometimes they're text that you can steal verbatim (e.g., emails to recruiters).

Resource or document: These are bonus chapters, downloadable problem sets, or worksheets (e.g., an equity calculator, a calculator to help you figure out the order in which to approach companies).

BUGS, QUESTIONS, AND CONTACT

Website: beyondctci.com

Bugs: bctci.co/bugs

Errata: bctci.co/errata

Email: beyond@gayle.com

Get $50 off anonymous mock interviews on interviewing.io

While this book gives you the tools and strategies to tackle tough technical questions, it is essential to put these tools into practice. With the purchase of this book, you get $50 off on interviewing.io mock interviews. There, you can practice anonymously with FAANG interviewers.

Even if you don't use interviewing.io, find a way to practice with another person; grinding problems by yourself quickly reaches the point of diminishing returns.

bctci.co/discount-X3A4

HELLO WORLD. HELLO READER.

The only thing worse than technical interviews is not having technical interviews.

Oscar Wilde(ish)

Always stay gracious, best revenge is your paper.

Beyoncé Knowles-Carter

Cracking the Coding Interview (*CtCI*) has been the industry leader in teaching engineers how to get into top tech companies for over a decade. It's referenced in almost every major interview prep source, including books, podcasts, blogs, and online courses, as the source of truth on how to get into the major tech companies in our industry. And—although this was never the intention—it's even been used as a classroom textbook and a question bank for companies to draw on.

So, why write *another* interview prep book? And why now?

While CtCI remains an excellent foundation for interview preparation, the landscape of technical interviews has evolved dramatically since its publication. Online interview resources have changed how candidates practice, and the questions asked in interviews have grown both in difficulty and variety. And, as more candidates are preparing for interviews than ever before, the bar for passing has risen accordingly.

But it's not just the questions that have changed. The process of landing a job at top tech companies has grown even more competitive. The technical interview format has sparked increasing debate, and AI is reshaping how people apply for jobs, how companies screen candidates, and even how interviews are conducted.

Amidst these shifts, we felt it was time to take a step back and rethink what a modern interview prep book should look like—one that complements existing resources and patches the gaps between what candidates are doing and what they need to be doing. We realized that a new edition wouldn't cut it. It would have to be a brand new book, and this book would have to:

- **Focus on understanding, not memorization.** We aim to teach you the computer science fundamentals *and* the interview skills needed to excel, regardless of your educational background. By building a deep understanding of these concepts, you'll be prepared to tackle problems you've never seen before—something memorization alone can't achieve.

- **Cover important non-technical topics.** CtCI only briefly touched on what some call the "squishy" stuff—resumes, negotiation, and managing your job search. This book devotes ~150 pages to these topics because they are now critical to success. We'll guide you through technical recruiting with practical advice, down to exactly what to say in the situations you're likely to encounter.
- **Ground our advice in data.** This book draws on a decade of insights from real-world interviews, including a corpus of over 100,000 interviews conducted on interviewing.io by senior engineers from FAANG+ companies. You'll get a data-driven perspective on what technical interviews look like at top companies today.

Now, to address the angry elephant in the room: technical interviews are flawed, right? Absolutely. We don't just acknowledge that; we're going to dive into a candid discussion of everything wrong with technical interviews, and what you can do about it.

Despite what some have suggested, CtCI didn't invent this format—nor is BCtCI going to end it. As far as we can tell, technical interviewing isn't going anywhere. But we hope to make the process a little less daunting and a lot more transparent—by leveling the playing field and giving you access to what was previously insider knowledge. We hope that you're able to put away your hatred of the format, read this book with an open mind, put in the work, and get your revenge... by nailing your interviews, fearlessly negotiating, and landing the high-paying, challenging, awesome job that you deserve.

Before we get on with the real stuff, we have one request for you. Do not *read* this book—at least not in the traditional sense of left-to-right, top-to-bottom, page-to-page. Rather, we implore you to *do* this book. Use it. Interact with it. This is so important that we have a literal README on how to use it.

We hope you enjoy ~~reading~~ doing this book, inasmuch as one can enjoy a book about interviews and job searches. We really enjoyed ~~writing~~ building it.

CRASH & LEARN: OUR FAILED INTERVIEWS

Everyone loves a success story, but failure is often the best teacher. We will share here our most humbling interview experiences—times we bombed, blanked, or were simply unprepared. These stories aren't just about the mistakes we made; they're about the lessons we learned and how those failures shaped us.

Our hope is that by sharing these moments, you'll see that even "experts" have stumbled, arguably in career-altering ways. More importantly, you'll learn that just as these interview failures don't reflect on *our* skills as engineers, nor do yours.

GAYLE

Entering my fourth year of a five-year undergrad/master's program, I was fortunate to have three Microsoft internships behind me. Determined to try something new for my final internship, I sent countless cold emails and resumes into the void of online applications. Somehow, Google picked mine. I was thrilled.

From the bits I'd heard about technical interviewing, questions could be anything from implementing an ArrayList to the heavy-ball brainteaser[1]. Imagine my surprise when, instead, I got a math question: "What is 2^20?" My answer was the always-brilliant "Um, I don't know? Can I use a calculator?"

All I could think about was what a stupid question this was and how I definitely didn't know the answer. And why should I? Who cares? *Just look it up if you need to know it.* (Sound familiar?)

What I didn't realize then was that—probably—my interviewer wasn't expecting me to just *know* this. Most likely, she wanted me to start from what I did know, perhaps that 2^{10} is about 1000. From there, if I knew my exponent rules, I could solve it: $2^{20} = 2^{10} * 2^{10}$ is about 1000 * 1000… so approximately 1 million.

In *her* mind, it was a problem-solving question—albeit one that relied on a lot of math[2]. But in *my* mind, I didn't know the answer, and I was supposed to, and <***PANIC***>. All I needed to do was focus on what I *could solve* rather than what I *knew*.

Still a stupid question though[3].

1 Given a balance (i.e., a scale that only tells you which side is heavier) and eight balls—all the same weight other than one which is slightly heavier—find the heavy ball in as few measurements as possible.

2 In general, asking math-heavy questions is a no-no for interviews. However, I'll let her off the hook here, a little. I was a computer science major with a math minor. If I didn't know exponent rules, something had gone very wrong.

3 Why is this a bad question? Put aside the "it's not relevant" part. The relevant factor is: is it predictive? Approached the right way, this *could be* a problem solving question. However, it didn't feel like that to me—and perception matters. Additionally, even if I had approached it as a problem-solving question, there is very little "meat" to the question; what distinguishes between good and bad other than understanding how to break up the exponents?

MIKE

In my first year of college as a CS student, I was the only student to get an opportunity to interview at Google for their internship position[4]—and I was ecstatic about it. The problem? I hadn't taken my data structures and algorithms class yet. My GPA took a colossal hit that semester as I spent way too much time cutting class to watch YouTube lectures on sorting algorithms and NP-hard problems in an attempt to prepare for my interview.

The interview day came, and I had two back-to-back meetings with different Googlers. I described the optimal solution for my first interview, which I nervously coded in C++ with a hashmap—but then the second interviewer stumped me with a binary tree question[5]. After much fumbling, the interviewer walked me through a viable way to solve the problem, which I coded, but I knew I had bombed my chance at a Google internship.

A few years later, I passed Google's full-time interviews twice—declining the first offer for a remote role at Salesforce and accepting the second for a permanent position at Google.

ALINE

After graduating from MIT, I spent three years cooking professionally. While my culinary detour could fill a book, I wasn't good enough at it to make it a career. Out of money, I returned to coding. Coming off three years of flipping pans, chopping food, and drinking nightly, I found myself back in a well-lit office, standing at a whiteboard, asked to reverse a linked list.

Not only had I forgotten what these interviews were like (I was expecting to talk about my past projects), but I completely blanked on what a linked list was, much less how to reverse one. And this particular interview was with Sasha Aickin, Redfin's CTO, back when they were like five people. It's one of the things I still kick myself for, every now and again.

To his credit, Sasha patiently explained how linked lists work, and, through a series of hints, got me to change pointer directions. I clumsily erased and redrew arrowheads, but I don't think I wrote much code. It was too little, too late.

I never forgot how shitty failing that interview felt. I also never forgot how gracious Sasha was, in the face of my failure, and how he patiently walked me through the problem by asking leading questions. Both of those experiences helped me come up with the idea for interviewing.io: a place where people could fail privately, without the stigma, and learn from those failures, by pairing with kind, empathetic people who care.

NIL

Nil insists that he's never failed an interview. We would like to clarify that it's because he's only done *one*—and passed. True to his name, Nil has exactly zero failure stories to share. Classic edge case.

4 For the curious, I sent over 100+ cold email messages to former alumni until someone was willing to refer me. Even back then, I was using the techniques we show you in this book.

5 I was asked to compute the maximum sum of any path in the tree, which did not necessarily start at the root but could include it. For a variant of that problem, see Problem 35.1: Aligned Chain (pg 436).

UGLY TRUTHS & HIDDEN REALITIES

PART II

Watch for these speech-bubble icons, which mean there is online content.

- **Interview Replay:** Recordings of actual mock interviews.
- **Snippet:** Material that you can copy/paste online, including email templates and code recipes.
- **Resource:** Bonus chapters, worksheets, and other material.
- **Problems, Solutions, and AI Interviewer:** You can try every problem in the book with the AI Interviewer. Solutions are online in four languages (Python, Java, JavaScript, and C++).

UGLY TRUTHS & HIDDEN REALITIES

CHAPTER 0

WHY JOB SEARCHES SUCK

Job searches suck—*especially* for engineers, who are, by and large, rational, well-intentioned people who expect the world to function according to some set of predictable rules. Why do job searches suck so much?

- **Job searches are not deterministic, and neither are interview outcomes.** In job searches, effort doesn't always correlate with results. For technical interviews specifically, there's little predictability in how the same person will perform from interview to interview (pg 26).
- **No feedback loop.** When you apply online and don't get a response, you can't tell if you weren't a fit or if no one even saw your application. Your insecurities can convince you that not only did a human look, but they quickly sized you up, saw right through you, and lasered in on every single flaw to conclude (correctly, in your mind) that you're unfit for the job. When you interview, whether you pass or get rejected, you often don't know why, which makes it difficult to know how to prepare the next time.
- **The content of your resume is often eclipsed by the brands in it. If you don't have brand-name companies or schools, it's much harder to get noticed.** Recruiters are notoriously bad at making value judgments based on resumes[1]. Despite these shortcomings, resumes are still the gold standard, and that means candidates from non-traditional backgrounds enter the game with a significant disadvantage.
- **To get in the door, you very likely have to know someone.** Surprisingly, this is even true for candidates who look good on paper; recruiters often ignore online applications because the signal-to-noise ratio is so poor. But it's especially true if you don't look good on paper.
- **Technical interviews are notoriously flawed and not representative of the actual engineering work you do every day.** This one is especially rough, and it bears out in our data. Senior engineers often do *worse* than juniors in their first few interviews because junior engineers have more recently completed an algorithms class or have done extensive interview prep. Senior engineers have been in the trenches, often focusing on building applications; there are very few engineering roles where you're doing the types of academic problems that you get in interviews day in and day out.

These are just a few of the challenges, but the strategies in this book will help you navigate them and achieve success—however you define it.

Given all these flaws, you might ask: How did we get here, where our technical interviews feel so divorced from the work and so unpredictable in their outcomes? For that, let's take a brief look at the history of technical interviewing.

1 See https://www.reddit.com/r/recruitinghell/comments/qhg5jo/this_resume_got_me_an_interview/

CHAPTER 1

A BRIEF HISTORY OF TECHNICAL INTERVIEWS

A definitive work on the history of technical interviewing was surprisingly hard to find, but we were able to piece together a narrative by scouring books like *How Would You Move Mount Fuji*, *Programming Interviews Exposed*, and the bounty of the internets. The story goes something like this.

Technical interviewing has its roots as far back as the 1950s, at Shockley Semiconductor Laboratories in Mountain View, California. William Shockley's[1] interviewing methodology came out of the need to keep up with the innovative, rapidly moving, Cold War-fueled tech sector, something that traditional hiring approaches taken from established, skills-based assembly line industries simply couldn't handle.

And so, Shockley relied on questions that could gauge analytical ability, intellect, and potential quickly. One canonical question[2] in this category has to do with coins:

> You have eight identical-looking coins, except one is lighter than the rest. Figure out which one it is with just two weighings on a pan balance.

The techniques that Shockley developed were adopted by Microsoft during the 1990s, as the success of the desktop computer, and later, the first dot-com boom spurred an explosion in tech hiring. Like Shockley, Microsoft also needed to quickly and scalably assess high volumes of candidates for potential. As software engineering became increasingly complex, it was no longer possible to have a few centralized expert programmers manage the design and then delegate away the minutiae. Even rank-and-file developers needed to be able to produce under a variety of rapidly evolving conditions, where just mastery of specific skills wasn't enough.

The puzzle format, in particular, was easy to standardize because individual hiring managers didn't have to come up with their own interview questions, and a company could quickly build up its own interchangeable question repository. Over time, most companies did away with puzzle questions[3] for engineers, and moved to algorithmic questions: these questions seemed more relevant but still assessed problem-solving skills.

At many top companies, such as Google, this need for interchangeable parts ultimately carried over to the interview process as well—rather than having individual teams run their own processes and pipelines,

1 We are acutely aware that this is the same William Shockley who became the poster boy for eugenics. He was a pretty awful person.

2 A first attempt—if you're an engineer—is to do something akin to binary search: split the coins into two sets of four coins each. Then, take the lighter set, and divide it into two sets of two coins each. Then, split in half again. But that will be *three* weighings, not two. To reduce a weighing, consider that the balance will also tell us if the sets are equal. We can divide the coins into *three* sets.

3 https://www.nytimes.com/2013/06/20/business/in-head-hunting-big-data-may-not-be-such-a-big-deal.html

companies standardized it. This way, in addition to questions, you could effectively plug and play the interviewers themselves—any interviewer within your org could be quickly trained up and assigned to speak with any candidate, independent of the prospective team.

At the same time, companies didn't always create incentives for engineers to work hard at being good interviewers, and as you'll see later in this book, we believe that much of the flak that algorithmic interviews get is due to the interviewers conducting them (and, often, lack of training or proper incentives).

So where does this leave us? Technical interviews are, at best, a *proxy* for the day-to-day tasks that a software engineer actually does, and not all interviewers are good. But, regardless, do technical interviews work? Well, that's complicated and depends a lot on your definition of "work." For whom, the candidate or the company? For what type of company? Compared to what?

We would argue that interviewing as a whole is flawed, and it's really a matter of picking your poison. However, even the most ardent defenders[4] of these sorts of technical interviews agree that false negatives—great engineers who get rejected—are common. FAANGs and other companies who adopt these processes tolerate a high false negative rate, under the rationale that it's better to reject a good candidate than to hire a bad one. The process is optimized to reduce false positives.

For you, the candidate, that kind of sucks. But it is what it is, and that's what this book is here for: to help you avoid being one of those false negatives.

4 Let's call out the elephant in the room. Some might assume that, as authors of a coding interview book, we must adamantly believe in the value of coding interviews. Not so. Not only has our intimate look at coding interviews exposed many flaws, but the entire existence of coding interview prep means that coding interviews are, at least, a little bit broken.

CHAPTER 2

WHAT'S BROKEN ABOUT CODING INTERVIEWS

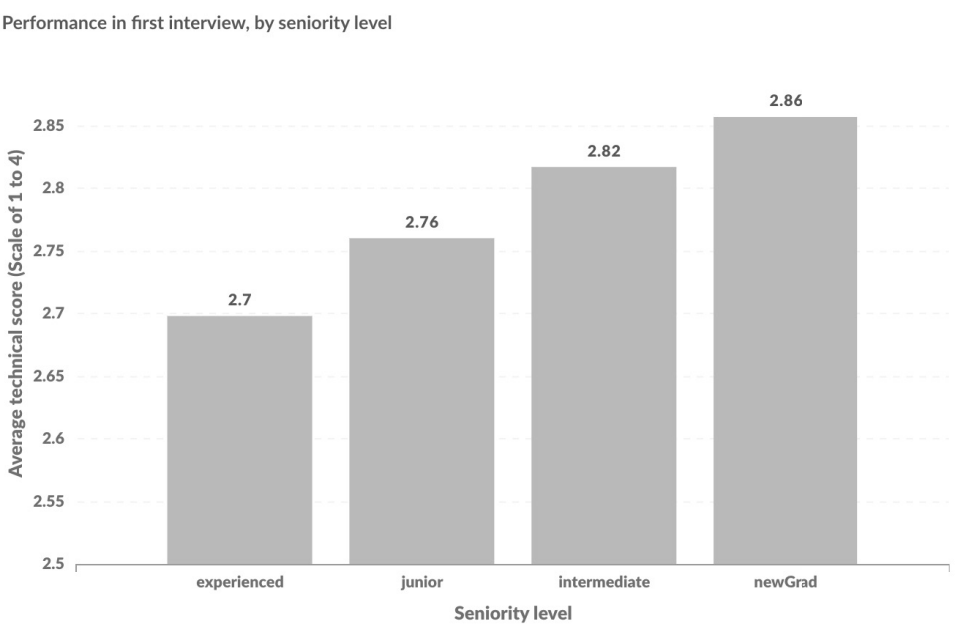
This chapter dives into the systemic flaws of technical interviews, from the prevalence of bad questions and bad interviewers to the randomness of interview outcomes and the growing interview-industrial complex. But it's not all doom and gloom. Once you understand the challenges and accept that the system is flawed, you'll be able to operate within it and win (and do so with confidence and integrity).

IT'S NOT THE WORK YOU DO EVERY DAY

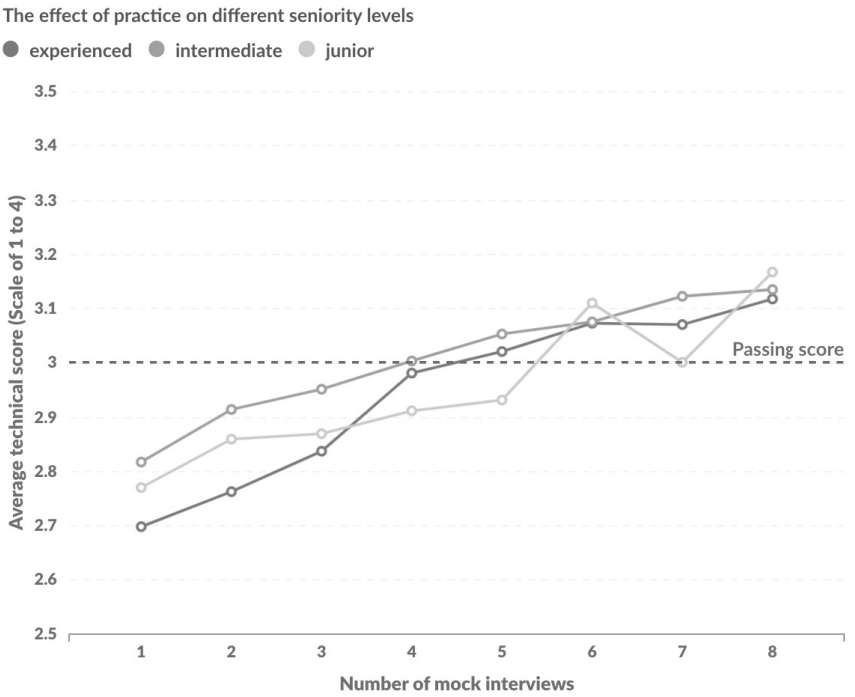
One of the most persistent critiques of technical interviews is that they feel disconnected from the work you do every day. If interviews were like the work you did every day, we'd expect that senior engineers would outperform juniors in interviews. As it turns out, that's not the case: frustratingly, the more experienced you are, the worse you perform.

We actually have data for this. If you look at performance in their first mock interview on interviewing.io, junior engineers significantly outperform senior ones. In the upcoming graph, you can see the average score that candidates got in their first mock interview on interviewing.io, broken out by seniority. Not only do junior engineers significantly outperform experienced engineers,[1] but experienced engineers perform the worst out of all the groups.

1 At this point, you're probably thinking that the bar is different for more junior engineers. At some companies, it is. At some, it is not. Our interviewers know the experience level of their candidates and adjust their bar accordingly when giving feedback. With that in mind, new grads likely do the best in interviews because they're fresh off a data structures and algorithms course.



This effect gets less pronounced as people practice more; once everyone has done a bunch of mock interviews, they all roughly converge, as you can see in the next graph. But, out of the gate, recency with the material gives you a significant advantage.



You might also notice in this graph that it takes about five mock interviews, on average, to start passing these interviews (pg 40).

BAD QUESTIONS (AND MEMORIZATION OVER UNDERSTANDING)

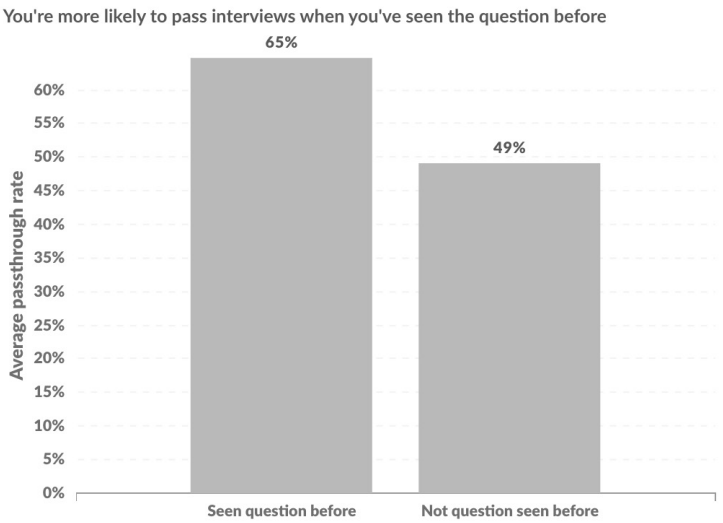
There is so much vitriol targeted toward technical interview questions that rehashing it in detail probably isn't worth the paper this book is printed on. If you've ever read any thread on Hacker News about interviewing, you know the main points:[2]

- **Questions are too academic and obscure.** You've probably never had to invert a binary tree at work.
- **Bad questions.** Many questions require a serendipitous "Ah-ha!" moment to solve, which even the best engineers may not be able to guarantee. Some questions aren't well-formulated or require too much domain-specific knowledge.
- **Too much memorization.** With how interviewing works today, it's rational to memorize a bunch of common questions, which turns the exercise into a test of memorization rather than understanding or ability.

We do not disagree with any of these points, and yes, these flaws are real. We've already talked about how we got here and why technical interviews are the way they are. It's easier for huge companies to scale if they can reduce interviewer training time and not have to come up with original questions/use LeetCode questions verbatim. Sadly, smaller companies often "cargo cult" large company practices, not realizing that they hire good candidates despite their processes and not because of them.

It's also true that memorizing helps a lot. On interviewing.io, after every interview, both interviewers and interviewees fill out a feedback form. One of the things we ask interviewees is whether they've seen this question before. We don't share whether they have or not with interviewers, so there's no reason to lie.

Here is a graph of the pass rate for algorithmic interviews as a function of whether candidates have seen the question before.



As you can see, familiarity with a question before gives you a serious edge in interviews: you are 33% more likely to pass. We expect that this disparity would be even higher if we had rephrased our feedback form to say something like, "Have you *practiced* this question before?"

2 We've also surveyed our users, as part of some research we did to figure out what makes someone a good interviewer: https://interviewing.io/blog/best-technical-interviews-common

The fact that memorizing questions gives you an edge is ironic, given that the whole purpose of modern technical interviewing is to evaluate one's ability to think like an engineer, rather than come to the table with a bunch of specific skills. It's also one of the things that makes it harder to stomach practicing for interviews. It's a tough pill to swallow to know that, ultimately, you're competing with memorizers.

In this book, we'll arm you with the kind of foundational understanding that will make memorization less important (but will be just as effective at improving your performance).

BAD INTERVIEWERS WHO DON'T WANT TO BE THERE

> Good interviewers can get good signal from a bad question. Bad interviewers cannot even get good signal from a good question. For a good interviewer, the question is just a tool to guide the discussion into interesting areas. For bad interviewers, the question is an absolute to which they must hear the exact answer they have in mind.
>
> Jos Visser, Member of Technical Staff at OpenAI, and formerly of Google, Facebook, and Amazon

Yes, bad questions are bad. However, bad interviewers are, in our minds, the biggest problem with technical interviews. A terrible question, in the hands of a skilled, engaged interviewer, can yield meaningful signal. A great question asked by an unskilled, disconnected interviewer will always be bad.

We talked about how large companies adopted modern technical interviewing in part because of the "interchangeable parts" approach to provisioning interviewers. However, human beings are not gears or sprockets—each comes with their own unique hangups and proclivities. It's naive to think that you can swap one interviewer for another and achieve the same result.

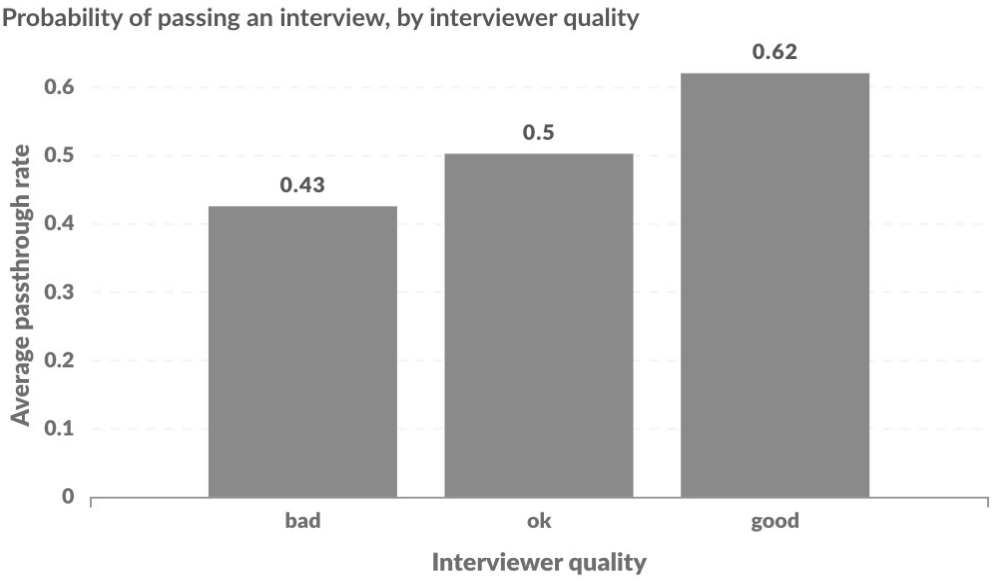
> In our experience hiring professional mock interviewers, we saw very quickly that whether someone likes to conduct interviews is bimodal: either they love it or they hate it, with not much in between.

The people who like interviewing tend to enjoy teaching. They tend to have higher-than-average empathy, they remember a time when they were on the other side of the table, and they want to make that experience less painful for their candidates. They also tend to approach interviews with a certain curiosity. They are curious about novel ways to solve the problem, about new rabbit holes their candidates will inevitably go down, and about the candidates themselves.

The people who hate interviewing treat it as a disruption—a necessary evil between shipping features. They do the bare minimum, and it shows. Over the years, we've listened to a lot of interviews. You can immediately identify when an interviewer is checked out. You'll hear them typing. You'll hear them go silent for a while. They'll often need to ask the candidate to repeat themselves. You certainly won't hear them collaborating with their candidate or gently guiding them away from a perilous rabbit hole. Most of us have been on the receiving end of an interviewer's callous indifference and know what it feels like.

Bad interviewers are common across companies, even top-tier ones,[3] and there is an added complication: companies don't usually track who their best interviewers are,[4] nor do they reward them. Sadly, it's often the opposite—bad interviewers get rewarded because they focus on writing code instead of conducting interviews. In other words, curmudgeons who alienate their recruiting team (and get scheduled less often as a result) get rewarded. Engineers who end up being less present in their interviews because their mind is elsewhere, still churning through the code they were writing when they got interrupted, get rewarded. In the rare instances where we've seen companies really care about interviewer quality, it's because an eng leader there has taken it upon themselves, as a labor of love.

Why does it matter if an interviewer is bad, outside of it being a poor experience for candidates? In this graph, you can see the average interview pass rates on interviewing.io, broken out by interviewer quality.[5]



Interviewer quality matters because the purpose of a technical interview is not to see if you can get the perfect answer. You're not just solving a coding problem online by yourself. As such:

- Bad interviewers miss out on critical two-way interactions with the candidate. You're meant to discuss approaches with your interviewer and ask questions, all in the service of determining if you're the kind of person who can pick up new skills in a rapidly changing landscape and work well with other smart people. When your interviewer is not holding up their end, it's much harder for you to hold up yours, and it turns into a contest of who's memorized the most questions.
- Bad interviewers are incentivized to be unnecessarily harsh. Better to say no to a good candidate than risk hiring a bad one, right?

3 Some people assume that top-tier companies invest more into interviewer training than others. In our experience, that isn't true. We won't throw specific companies under the bus in this book, but if you read interviewing.io's guide to FAANG interview processes (https://interviewing.io/guides/hiring-process), you'll see how much or how little some of them invest in interviewer training.

4 https://interviewing.io/blog/best-technical-interviews-common

5 We determine quality as a function of how highly rated interviewers are by their candidates with respect to question quality, hint quality, and how excited the candidate would be to work with that person. It's a "candidate experience" score, in other words. Candidates submit these ratings before they know how they did. This graph is also not about interviewer leniency—for what it's worth, the best-calibrated interviewers tend to be the best-rated, and those who are too lenient tend to be among the worst.

- Bad interviewers will judge too much on superficial grounds, which is especially unfair to candidates who come from a non-traditional background or lack some little bit of institutional knowledge.

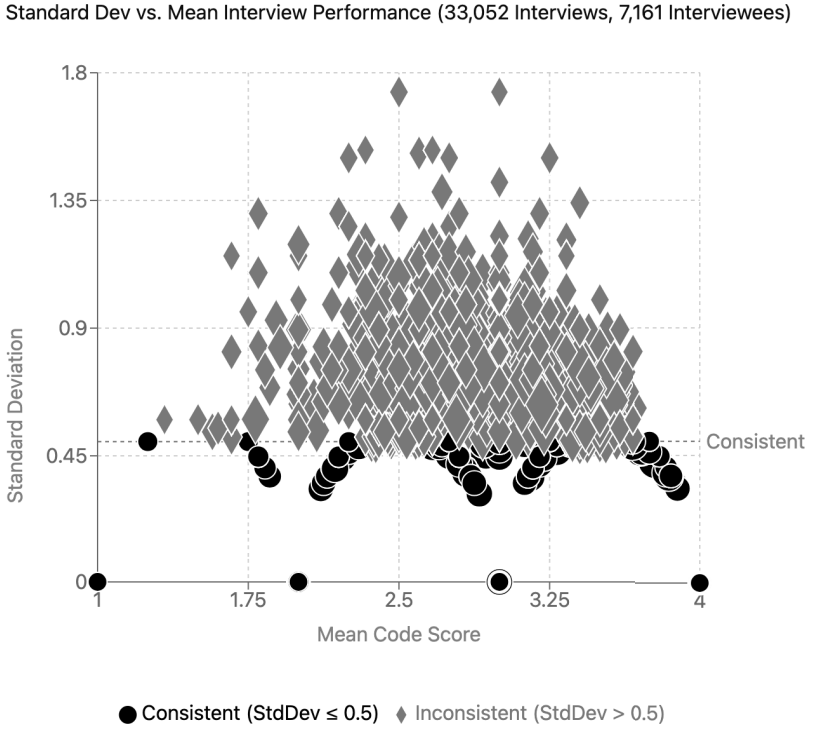
For all their perceived objectivity (and certainly they're more objective than ones where you talk about your experience), coding interviews are a complex interaction between two humans. When one of the parties isn't truly present, the candidate pipeline suffers, and you end up with fewer candidates to choose from and, ultimately, worse hires.

NON-DETERMINISTIC OUTCOMES

Anyone who's done multiple technical interviews has probably felt in their gut that outcomes are somewhat arbitrary. So much depends on serendipity and, well, clicking! *Did you click with your interviewer? Did something click in your head at the right time when trying to solve the problem?*

If you've felt like this, you're not alone, and we have the data to prove it. On interviewing.io, after every interview, you get a technical score from your interviewer, on a scale of 1 to 4. The same candidate can do multiple interviews, each of which is with a different interviewer and/or different company, and this opens the door for some pretty interesting and somewhat controlled comparative analysis.

With that in mind, we looked at how the same person performed from interview to interview.



We analyzed interviewing.io's data to understand how individuals performed across multiple interviews (for this analysis, we included people who did between 3 and 20 interviews). Each circle or diamond represents people with that specific average score and standard deviation across their interviews.

The y-axis is the standard deviation of performance; a higher standard deviation reflects more volatility. Some surprising takeaways from this analysis:

- Only 25% of people are consistent from interview to interview (standard dev of <= 0.5). Everyone else is all over the place.
- 64% of the people who scored at least one 4 have also scored at least one 1 or one 2.

So, what did the most highly volatile performers have in common? The answer appears to be, well, nothing. About half were working at top companies. About 60% had attended top schools. And years of experience didn't have much to do with it either—a plurality of interviewees had between 2 and 6 years of experience, with the rest all over the board (varying between 1 and 20 years).

When we corrected for interviewer strictness, the effect didn't go away either.

Why is this bad? This inconsistency means randomness significantly impacts your career. In a way, interviews serve the same function as standardized tests—giving an organization a way to make a value judgment about someone's ability relatively quickly and without a lot of priors, in a way that's consistent and repeatable. For all their flaws and biases, standardized testing providers have made a lot of effort to make sure that their results are repeatable because their results often determine the social mobility and livelihoods of millions of students every year. Even though they have a similar impact on outcomes for millions of engineers, tech companies have not done the same for their interviews.

BUCKETED SCORING

Imagine that we wanted to evaluate whether students in a given classroom were tall or short. But rather than measuring people's heights, we first bucketed them into "very short," "short," "tall," or "very tall." Inevitably, there will be many cases where two students are nearly the same height, but a few millimeters make the difference between "short" and "tall."

Any bucketed scoring will lose precision and create arbitrariness, but it's even worse when we have few buckets and split the middle zone—which is common. That is, if, rather than providing a bucket for "average height" (where most people might fall), we split these people into "short" and "tall" depending on whether they cross some threshold. Now, we don't just have some arbitrary scores; we have a lot of them.

This is effectively what's happening in many technical interviews. In fact, it can be a triple-whammy.

- **Bucketed Scoring:** Interviewers don't score you as a 2.6; almost always, interviewers are forced into bucketed scores, such as 1, 2, 3, or 4 (or its qualitative equivalent: strong no-hire, no-hire, hire, strong hire).
- **Risk-Averse Scoring:** Interviewers are encouraged to be risk-averse: "better to reject a good one than hire a bad one," they say. That 2.6 interview is more likely to be reported as a 2 than a 3.
- **Splitting the Middle:** In many cases, interviewers are bucketed into just *four* buckets— with no place for "average" or "maybe." The average score is often around a 2.5 to 2.7, so a four-point scale splits the middle candidates. This means that *a lot* of people will get rounded up or down into "hire" or "no-hire."

A candidate who gets {3.2, 3.3, 2.7, 3.4} on a four-point scale may be hired, but the candidate who gets {3/hire, 3/hire, 2/no-hire, 3/hire} (a possible risk-averse rounding equivalent) may not be. We've effectively lost the information that the "no-hire" was actually really close to a "hire."

No matter how you bucket, bucketed scoring effectively forces interviewers to translate very typical scores into something more extreme and loses fidelity in the process. No wonder we have so much variability in interviews!

INTERVIEW PREP BEGETS INTERVIEW PREP

Technical interviewing has given rise to a booming preparation industry. This is somewhat ironic, as it defeats the purpose of this form of interviewing; the goal was to understand the candidate's aptitude, independent of what they currently know.

The reality is that—as we've shown—interview prep *works*. We might not like it, but people do better with preparation. That's why interview prep is a multi-billion dollar industry, including everything from books and courses to asynchronous coding challenges and mock interviews. This also means that you are being compared to candidates who are prepping for interviews (and, in many cases, simply memorizing a ton of questions), which means that the expectations for *you* have gone up too. What do you do about this? You, of course, prepare for interviews too.

It's an unfortunate cycle; interview prep begets interview prep.[6] But for the record, memorizing problems without *also* working on understanding goes against our preparation philosophy, and it's honestly not that effective.

All that said, it's time to change our lens and talk about how to work the system. Technical interviews are here to stay, and if you want a job at a top-tier tech company, you have to jump through this hoop.

6 Would it be better if we did the impossible and magically got rid of all interview prep? Probably not. Before today's industry, there were still some books and resources—but it relied more on word of mouth and "inside" sources (friends telling you what to expect). This favored people with connections. The bar might have been lower, but the playing field was more unfair.

CHAPTER 3

WHAT RECRUITERS WON'T TELL YOU

Show me the incentives, and I'll show you the outcome.

Charlie Munger

It is difficult to get a man to understand something when his salary depends on his not understanding it.

Upton Sinclair

Even though recruiters try to position themselves as your advocate, remember this: recruiters are not your friend, and they don't work for you.

I (Aline) used to be a recruiter. I ran my own third-party agency, and I also worked in-house before starting interviewing.io. That means I've had to struggle with the tangled incentive structure[1] that comes with being a recruiter. There's always tension; recruiters are, by and large, good human beings who genuinely want to help their candidates, but they also have an employer they're beholden to, as well as a comp/bonus structure that rewards certain behaviors, some of which run counter to candidates' best interests.

There's some distinction between in-house recruiters and third-party recruiters (recruiters who work for an agency that does placement, rather than a specific company that's hiring engineers).

THIRD-PARTY RECRUITERS

Working with third-party recruiters (also known as agency recruiters) is, at best, mixed. The most important thing to realize about them is that you are not the customer. The employer is. As such, even when they're paid a commission based on your salary, their incentives are, at best, *sometimes* aligned with yours.

Ultimately, a recruiter's incentive is to *get the deal done*, not to get you the best possible deal. Here's why.

A recruiter, depending on market conditions, gets anywhere from 8% - 25% of a candidate's base salary when they make a placement. However, that cut is going to the recruiting agency as a whole rather than to the individual recruiter; you will almost always end up working with large agencies rather than a single person.

Let's say that you get an offer. You talk to your third-party recruiter and tell them that you would like more money. The recruiter may go to the hiring manager and try to advocate for you, but they're not going to

1 https://blog.alinelerner.com/if-youre-an-engineer-who-wants-to-start-a-recruiting-business-read-this-first/

push very hard because the incremental difference in their cut is going to be pretty small[2], and to them the thing that matters is making the hire. After all, they're evaluated on the number of hires they make, first and foremost, independent of compensation. Third-party recruiters are incentivized to get the deal done, not to risk the deal by negotiating hard for you.

This means:

- If they have multiple clients potentially matching the same company, and they know you're less likely to take their offer, they'll fight for the other candidates more than you.
- They may expose information to the company that reveals whether you'll demand a high salary. Remember: they're paid by the company, not you. If the company likes them, that's good for them.
- They may encourage you to take a lower salary. Sure, they get a slightly higher commission if you negotiate a higher salary, but they get *no* commission until the deal is done.
- They may encourage an employer to lowball you, if they think you'll take it.[3]

So, when you work with third-party recruiters:

- Do not tell them *anything* about your job search.
- Do not share your compensation history/expectations.
- Always deal directly with the companies they introduce you to, once you establish a point of contact there.
- Assume that anything you tell your recruiter is going to get back to every company you're working with.

IN-HOUSE RECRUITERS

Some in-house recruiters get a bonus for hires, but this bonus is rarely tied to your compensation. In fact, in some cases, they may get a bigger bonus if they're able to negotiate you down.

At big companies, in particular, in-house recruiters follow a playbook, and are evaluated accordingly. They're trained to make offers within specific bands, and they're trained to mobilize such that they don't lose candidates to other big companies; if you wave a Facebook counteroffer in front of Google, they will act. If you tell them you're interviewing at a startup, they will not, because they know that startups don't pay as much.

Because of this playbook[4]—and because they are working for the employer—their incentives do not align with yours. They're incentivized, first and foremost, to follow the rules their head of department sets for them. This is true for how they evaluate candidates, who they let through, and how they read resumes. And it's definitely true for how they negotiate.

Generally speaking, recruiters want to help, and many are rooting for their candidates. But they're also operating inside a box, and that box isn't set up to put your interests first.

---

2 Understanding that, let's do the math anyway. Let's say your offer has a base salary of $150k. Say that your recruiter goes to bat for you and tries to get you up to $165k. Before, the agency would have gotten paid $15k. Now the agency gets paid $16.5k. That incremental $1.5k isn't worth risking a deal over (even a few thousand dollars would not justify jeopardizing the deal). On top of that, the individual recruiter is only going to maybe get a few hundred dollars total from that increase.

3 Why would a recruiter recommend that an employer not pay you more? It seems counterintuitive, but remember that the employer is their customer, not you. A savvy business person will often take a short-term hit in the service of building an enduring relationship with their customer. Telling an employer that they don't need to pay a candidate more (even though paying the candidate more would get the recruiter marginally more money) builds trust and makes it more likely that that employer will keep coming back to them for future searches. Employers often work with several agencies at once and cycle between agencies, so anything agencies can do to retain employers is a win.

4 For more insight, watch this video on how recruiting leaders think about looking at resumes, debriefing interviews, and extending offers. https://www.youtube.com/watch?v=dHSufqvgUqY

A NOTE ABOUT IN-HOUSE RECRUITER TENURES

Many recruiters at FAANG companies are contractors. The longer a contractor works, the more their role may resemble that of a full-time employee. To avoid legal risks, companies often limit contract durations to around 6 months. While this practice is most common in California due to its strict labor laws, similar limitations are applied in other states, though the typical contract length may vary.

Why do recruiter tenures matter to you?

- Never assume that the recruiter you're talking to now will still be there in a few months, when you're ready to interview.
- Build good rapport with your recruiter. Odds are good that, even if you don't pursue their current company, you might be interested in their future company.
- Don't lie about other offers—it's not super likely your recruiter is checking whether you have the offers you say you do, but chances are, because recruiters have short tenures and work at lots of companies, your recruiter knows someone there. We've seen it happen.

We'll talk a *lot* more about how to work well with recruiters in future chapters, and now you'll understand why we give the advice that we do.

RECRUITERS WILL TRY TO CALL/TEXT. USE EMAIL INSTEAD.

Recruiters are notorious for calling and texting instead of using email. Why?

The generous interpretation is that recruiters generally want to move things along quickly. If they can get you to respond quickly, through a synchronous medium like phone or text, then they can likely move you through the process faster as well.

There is one other, less generous interpretation—phone and text give recruiters the upper hand. Even the worst recruiters, if they have any work experience at all, quickly become seasoned negotiators. Even inexperienced recruiters have an edge because they are following a script and playbook. They've been told what to say in a variety of situations, and they interact with candidates multiple times a day, over years.

You, on the other hand, do one job search every few years.

Email lets you level the playing field. You can craft every word, and figure out what to say and how much to reveal. Over email, you can play it cool, and you can take the time to get advice from friends or experts.

When a recruiter calls you on the phone, they get your impromptu responses. They can see if you're excited about the money, and possibly get you to answer questions that you might not have otherwise answered (e.g., your salary expectations). The phone will keep you off balance.

While texting allows you to think longer about your response, its casual nature lulls you into a false sense of security. Moreover, the fact that texts interrupt you from something else puts you at a disadvantage—when you get interrupted, your instinct is to quickly respond to make the interruption go away. But knee-jerk responses are rarely the right ones, and you end up giving away information you shouldn't have.

We encourage you to stick to email as much as possible. You will have to get on the phone eventually, of course, likely when they are ready to extend an offer, but you never have to text.

CHAPTER 4

WHAT INTERVIEWERS WON'T TELL YOU

Material for this chapter at bctci.co/interviewer-secrets

Much of interviewing advice takes an idealistic stance: interviewers are fundamentally well-intentioned rational actors who want to run a fair process, and any flaws in the process itself are circumstantial or occur because of very rare bad actors.

That's not strictly true. As we previously discussed (pg 24), most employers don't reward strong interviewing skills, so interviewers have little motivation to improve.

Here are some additional interviewing tropes, which may or may not be true. We'll discuss them all and address them with data. Can you guess which ones are true?

- It's not about whether you get the right answer. Rather it's about demonstrating your thought process.
- Interviewers decide early on if you've passed the interview.
- If you're a great communicator and build rapport with your interviewer, you can pass the interview, even if your technical skills are wobbly.

IS IT REALLY ABOUT DEMONSTRATING YOUR THOUGHT PROCESS, OR DO YOU NEED TO GET TO AN OPTIMAL SOLUTION?

On interviewing.io, candidates who clearly and correctly explained their approach but failed to reach an acceptable working solution received a "thumbs up" only 32% of the time, compared to the platform's overall average of 51%. This means that failing to produce a working solution reduces your chances of passing the interview by 37%.[1]

How Interview Replays Work and How to Use Them

Interview replays come from mock interviews on interviewing.io. Replays are shared with the permission of both participants. Each replay includes the interviewer's feedback. You can watch just the relevant snippet, but we include the entire interview.

We strongly recommend pausing your reading and watching these replays. Hearing real people interview lets you learn from their mistakes—so that you aren't doomed to repeat them.

Where we know it (and where enough time has elapsed since they did mock interviews), we'll share the candidate's outcome.

1 To be clear, if you've gotten to the optimal solution and have *mostly* working code (maybe some small syntax errors or an off by one error), we believe most interviewers would still give a Hire rating. But if it's not working, it has to be very close.

NOT GETTING A WORKING SOLUTION, DESPITE DESCRIBING IT SUCCESSFULLY INTERVIEW REPLAY

View Online:	bctci.co/interviewers-replay-1 @ 45:50 - end
The Question:	Given an array of integers, are there elements a, b, c such that a + b + c = 0? Find all unique triplets which give the sum of zero.
What You'll See:	Although the candidate explained the solution well, their code had bugs. The interviewer offered some suggestions on how to improve coding speed and discussed the importance of writing compilable code.
Who:	*Interviewer*: Staff Software Engineer at Lyft *Candidate*: 4 years of experience

So, why do many interviewers insist on saying that it's about your thought process rather than the end result? That's complicated, but there are a few reasons:

1. It *feels good* to say. Interviewers *want* to believe that they're grading on something deeper and more qualitative than getting the right answers to really hard questions.
2. It is true, in a sense; interviewers prioritize your *solving ability*, not the answer itself. If you just spit out the right answer because you already know it, this won't be impressive to a good interviewer and could even lead you to getting eliminated for "cheating."
3. It *used to* be even more true. As standards have increased, there has been a bit less forgiveness for a candidate who made good progress but didn't get to the best solution.

The reality is that interviewers *are* looking for strong problem solvers, and—for a good interviewer—it *is* about the process, not the solution. However, it's often hard to convince them that you're a strong problem solver if you can't reach the optimal solution.

Either way, this trope stresses the importance of practice—the more you practice, the faster you'll get, and the more likely you get to an acceptable solution.

FIRST IMPRESSIONS AREN'T PREDICTIVE. EARLY PERFORMANCE IS.

You might have heard that the interviewers decide in the first few minutes whether you've passed the interview; we've certainly heard that plenty. But even supposing that's true, is it true for technical interviews? Given that technical interviews are focused on problem solving, it would be troubling if interviewers did in fact make such snap judgments *before the problem solving had even happened.*

At interviewing.io, we looked into this claim by analyzing sentiment—interviewers can make a comment during the interview (we call these "annotations") with an associated sentiment (positive, neutral, or negative), which the candidate will see afterwards.

Those who believe that the first impression is make-or-break might be reassured by this: the interviewer's first annotation is aligned with the outcome just 56% of the time—barely better than a coin flip. The first annotation occurs at an average of 13 minutes into the interview, which means the chit-chat is done and we've kicked off the problem solving portion. We can't speak for what happens in non-eng interviews, but at least for eng interviews: no, interviewers do *not* decide in the first few minutes. Whew!

However, your interviewer's impression of you by the 18-minute mark *is* predictive.

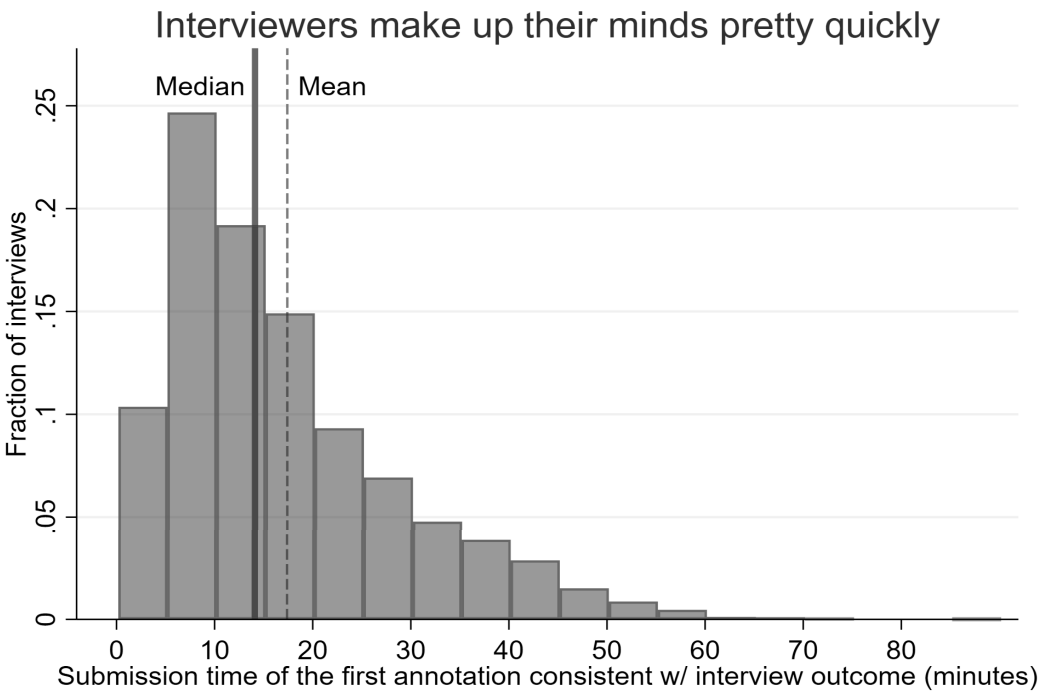


Figure 1. This graph is based on 7,883 interviews with annotations from the interviewing.io corpus. You can see when the first annotation that was predictive of the interview outcome happened. Most happen in the first 18 minutes.

Moreover, the average sentiment gets worse over time for both successful and unsuccessful candidates, with the sharpest decline occurring in the first 15 minutes.

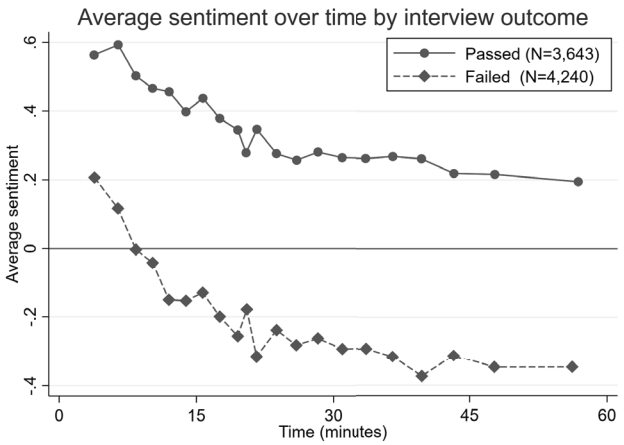


Figure 2. This graph is based on 7,883 interviews with annotations from the interviewing.io corpus.

You could interpret these findings cynically—all candidates do badly, and interviewers get stuck in preconceived (or at least early) notions. But as authors who have—collectively—spoken with thousands of candidates and interviewers, we see this in a more positive light.

Good interviewers[2] adjust their guidance based on your performance: when you do well, they give you less assistance; when you struggle, they give you more. It's like a live version of computer adaptive testing. This means that while you might do well in the first 10 minutes (mostly chit-chat and explaining the problem), *good* interviewers will adjust so that you struggle just enough. Perhaps seeing that struggle is why sentiment generally gets more negative over time; a good interviewer is purposefully putting you into situations that challenge you, thus creating more room for negative sentiment[3]. But that's part of the interview process.

This means, for you, don't panic when things get more challenging. *Embrace the struggle*. Struggling doesn't mean that you're doing poorly; it can just mean that your interviewer is doing their job. They've given a problem of appropriate difficulty and adjusted their guidance to suit your abilities.

We also believe, optimistically, that it's a good thing early-formed opinions are predictive. At the 18 minute mark, we're about 10 minutes into the problem solving portion, by which point our interviewer has seen some of our skills. If the first 10 minutes of problem solving were not predictive of the next 30 minutes, that would be a further indictment of the interview process. After all, it's not like we're magically becoming better or worse problem solvers, so our performance at the beginning *should* typically be predictive of the rest.

It's good that we see consistency. It means that our interviewers, if they're good, are actually looking at our problem solving process, not quantitative metrics like "number of minutes to solution" or "did you get it perfect right away", nor dinging us because we have some tiny bug.

This means, again, there's no reason to panic at little mistakes; these aren't making interviewers flip flop. Not all interviewers are good interviewers (unfortunately), but still—interviewers are generally looking at your performance as a *process.*

2 Fortunately, this data set is limited to interviewers who were rated well by their candidates. Our interpretations of the data do not hold for bad interviewers. As we discussed earlier (pg 24), there are many bad interviewers who aren't engaged and check out after the first few minutes.

3 However, this doesn't mean that you are *truly* doing worse. A good interviewer will also be looking at signal (pg 180), which means that they are taking into account the complexity of the problem.

COMMUNICATION SKILLS ARE NICE, BUT TECHNICAL SKILLS ARE NICER

Communication in technical interviews boils down to two aspects: rapport-building at the start (about yourself, your projects, etc) and clear communication throughout the interview.

RAPPORT-BUILDING AT THE START

Look, this isn't the kind of thing you usually say in an interview prep book, but the chit-chat in the first few minutes does not matter… *much*. Data shows positive communication in the first five minutes increases pass rates by 6 percentage points for underperforming candidates, compared to a 25-point boost from strong technical performance. Be enthusiastic and personable, but don't stress this unless small talk is a challenge for you. The technical stuff really does matter more.

COMMUNICATION THROUGHOUT THE INTERVIEW

Although many candidates believe in coding as soon as possible, the data shows the opposite: successful candidates actually code *slightly* later (about two minutes later[4] in a one-hour interview). This difference is small, but extra surprising given that many unsuccessful candidates will be quite delayed in coding because they lack a working approach. So why would coding later be linked to better outcomes? Communication, essentially. Candidates who explain their approach do better in interviews, but communication takes time.

Thinking out loud is essential in technical interviews, but beyond a baseline, communication may not significantly impact outcomes—at least for roles below Staff. On interviewing.io, candidates are rated on technical skills, problem-solving ability, and communication skills (all on a 1 - 4 scale), alongside a pass/fail score. Our analysis found that once a candidate scores 2 / 4 in communication, additional improvements yield diminishing returns. In contrast, boosting technical or problem-solving scores by 1 or 2 points can increase the odds of passing by 1.5X to 2.4X, emphasizing the greater importance of these skills.

As roles become more senior though, communication and behavioral skills play an increasingly critical role in interview outcomes. At the Staff level and beyond, our anecdotal experience—supported by feedback from interviewers—shows that the evaluation criteria shift significantly to emphasize these skills.

All that said, we can't fully tease apart technical ability and communication skills; when a candidate can't communicate their solution, interviewers are also less likely to be impressed by their approach—and they are less able to give you hints to help you along.

Where does this leave you?

- Don't worry about practicing specifically for the first 5 minutes of the interview. You can prepare for it a little, but don't stress about it.
- Get used to thinking out loud.
- Spend the bulk of your time getting better at solving problems.

Fortunately, practicing thinking out loud doesn't come at the cost of practicing solving problems. You can quite literally do them at the same time.

4 Candidates with successful interviews first run code 27% of the way through the interview, whereas candidates with unsuccessful interviews first run code 23.9% of the way into the interview. This difference is small, but nonetheless statistically significant. https://interviewing.io/blog/we-analyzed-thousands-of-technical-interviews-on-everything-from-language-to-code-style-here-s-what-we-found#user-content-fnref-3

CHAPTER 5

MINDSET AND THE NUMBERS GAME

No matter how good an engineer you are, if you get too much in your head during interviews, you'll fail. We've seen countless candidates self-sabotage because they jumped into the interview prep material before they accepted some fundamental truths about the journey. This chapter is about how to approach these interviews, and we encourage you to take it seriously. It's not woo-woo nonsense. It's our way of making sure you can apply your full potential for the rest of the book. With that in mind, here are the things you need to internalize and truly believe before you begin your studies.

TECHNICAL INTERVIEWING IS A NUMBERS GAME

Technical interviews are a numbers game, but many engineers underestimate just how much. It takes doing many interviews, and even more importantly, access to a peer group going through the same thing, to really internalize it. Here, we'll try to short-circuit all of that and convince you that failing an interview doesn't reflect on your engineering skills or potential; it's often the byproduct of a broken system.

Remember the graph (pg 26) where we showed that most people's technical interview performance is all over the place? Here it is again (Figure 1). Only 25% of people performed consistently, and about two thirds of people who got at least one 4 also got at least one 1 or 2.

Due to this inconsistency, even great engineers routinely fail interviews. This is a particularly big issue in technical phone screens, where interviewers must decide if someone gets to onsite based on *just one data point*.

Many candidates go into the process assuming it's repeatable, like a standardized test—a reasonable but flawed assumption. The truth is, unless you've been through the wringer, and unless you have people around you who have *also* been through the wringer, you're unlikely to realize just how unpredictable and variable the process can be.

For me (Aline), this became clear when I was a student at MIT. Attending a top-tier computer science program offered a number of advantages, but one of the most important (and least obvious) was access to a peer group that was going through the same things.

Having this group around me meant that we could all practice with each other, share our successes and failures, and have multiple shots on goal at top companies. Everyone was interviewing everywhere, and we quickly learned that bombing a Microsoft interview did not mean that you weren't meant to be an engineer. It just meant that you needed to work some more problems, do some more mock interviews, and try again at Google.

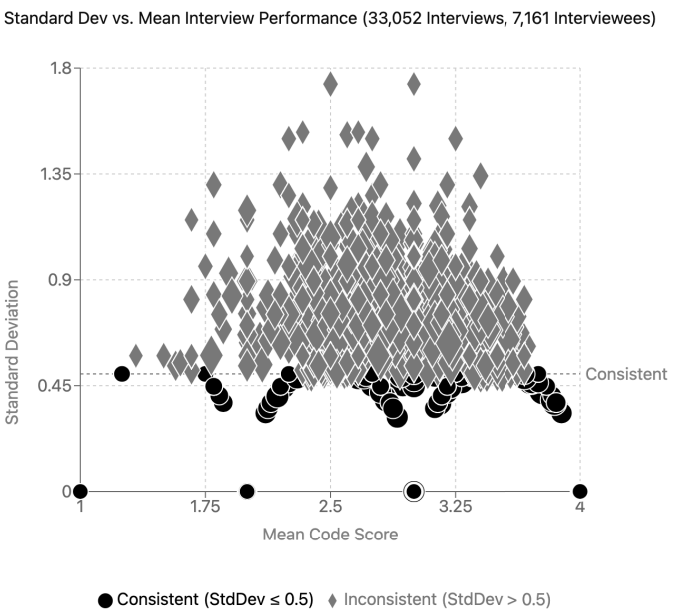


Figure 1. Standard deviation vs. mean interview performance (33,052 interviews; 7,161 interviewees). We analyzed interviewing.io's data to understand how individuals performed across multiple interviews (for this analysis, we included people who did between 3 and 20 interviews). Each circle or diamond represents people who had that specific average score and standard deviation across their interviews.

> One of the most critical things to internalize is that if you fail an interview, that's all it is. It is *not* a well-reasoned indictment of your potential as an engineer. Keep practicing and try again.

Unfortunately, dusting yourself off and trying again is harder for some groups than for others.

PERSEVERANCE AFTER FAILURE IS HARDER FOR ENGINEERS FROM NON-TRADITIONAL BACKGROUNDS

Many years ago, we noticed that on interviewing.io, women were performing significantly worse in technical interviews than men. This disparity still exists today. But before you jump to conclusions, the reason is *not* that women are actually technically weaker. Let us explain.

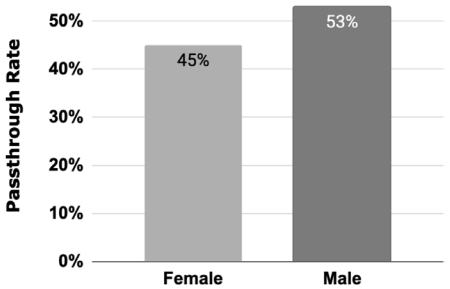


Figure 2. Gender differences in interview passthrough rate.

When we first noticed this disparity, we wondered if it was due to bias against women. So, we ran an experiment[1] using a real-time voice modulator that could change the pitch of the user's voice (making women sound like men, and vice versa).

Contrary to what we expected, masking gender had *no effect* on interview performance.

Perhaps the women were less senior? Doing different work? Nope. Neither of those factors seemed to differ meaningfully between groups.

What *was* different then? Women left interviewing.io roughly seven times as often as men after they do badly in an interview. And the numbers for two bad interviews aren't much better.[2]

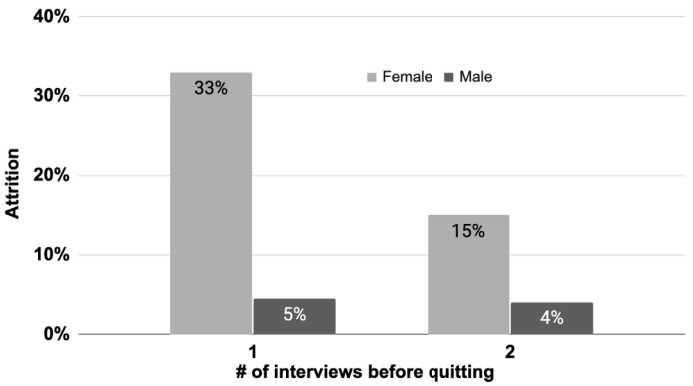


Figure 3. Attrition after poor interview performance.

When you correct for attrition, the difference between men and women goes away entirely.[3]

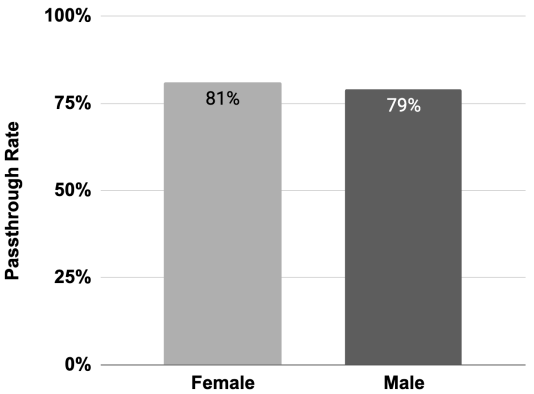


Figure 4. Average passthrough rate by gender when correcting for people who drop out after two failures.

1 We split our users into three groups: unmodulated (the control group), modulated *without* pitch change (another control; we needed it because modulated voices sound somewhat processed, and the last thing we wanted was for interviewers to guess the gender of their interviewees because anyone who sounded processed and male must actually be a woman and vice versa), and modulated *with* pitch change. This last one was the treatment group. https://interviewing.io/blog/voice-modulation-gender-technical-interviews

2 The differences between men and women are extremely statistically significant with $P < 0.00001$.

3 We have not studied this effect in race or in socioeconomic status, but we'd expect that you'd see a disparity in perseverance between groups, with engineers from non-traditional backgrounds being more likely to quit.

We are calling this out because we want it to change your behavior: everyone fails these interviews, regardless of background. If you're a woman, and very likely if you're from a non-traditional background, know that failure isn't a career end. Accept the system's flaws and learn to navigate them.

This means learning as much as you can, practicing as much as you can, then learning some more, and then practicing some more.

End-to-end, the practice journey may take anywhere from two to six months, depending on where you start and your previous exposure to algorithms and data structures. But, no matter who you are, you will need to practice, and you will fail some interviews. This is an axiom of our book.

It might take two failures before things turn around. It might take fifteen. But we implore you to exert yourself and keep going. If you approach technical interviewing from a place of curiosity, we *promise* that you'll get through it and find success at the end… which brings us to our final point.

PRACTICE IS ~~REALLY IMPORTANT~~ CRITICAL

It's all well and good to talk about who is doing well in mock interviews, but at the end of the day, your goal is to do well in *real* interviews. To understand what is driving strong performance in real interviews, we surveyed interviewing.io users about how well they did in real interviews at Google, Meta, and Amazon and compared their performance at these companies against their past total interview count (both mock and real), gender, whether they had a computer science degree, and how they learned to code.

The factor that stood out most was how many technical interviews an engineer had done in the past. Across all users, people who had completed five or more interviews had higher rates of passing a phone interview than those who had done fewer. The number of interviews mattered more than people's starting technical proficiency, and it also mattered more than gender and pedigree (those factors didn't actually matter).[4]

PROBABILITY OF PASSING A TECHNICAL PHONE SCREEN	1 - 4 INTERVIEWS BEFOREHAND	5+ INTERVIEWS BEFOREHAND
Amazon	65%	81%
Meta	40%	71%
Google	51%	80%

Figure 5. The probability of passing phone screens at Amazon, Google, and Meta, as a function of how many technical interviews candidates had done previously, based on data from interviewing.io users. Practice helps, and the "tipping point" is five interviews.

But what, specifically, is it about practice that is so impactful? According to survey results, it's about direct feedback (when candidates were fortunate enough to get it; most often it was in mock interviews, rarely in real ones).

Engineers typically can't gauge how they did in interviews,[5] so they probably can't gauge *why* they passed or failed—although, unfortunately, they often *think* they can. As a result, they are bound to over-index on the wrong things and neglect the ones that truly matter. We'll talk more about the limited utility of using real interviews for practice in "What about using companies for practice?" on page 79. Mock interviews offer direct, honest feedback from other people so that you know how you're perceived and what to improve.

4 https://interviewing.io/blog/how-know-ready-interview-faang

5 https://interviewing.io/blog/people-cant-gauge-their-own-interview-performance-and-that-makes-them-harder-to-hire

Finally, practicing can help resolve some of your (valid[6]) frustrations about technical interviewing. Working for a FAANG+ means having to do algorithmic interviews, and if you're resentful of the format, you will not learn as quickly or effectively. Making peace with the format—whether that's seeing its benefits or just putting aside your feelings—will make your preparation much more effective (and probably much less unpleasant). As one engineer we know put it:

> [Practicing] unambiguously changed the game for me. I went from hating interviews and ranting about whiteboarding to "If you go to med school you have to take the MCAT."

THE TWO FUNDAMENTAL METRICS

The two most important metrics to getting a job boil down to:

1. How many interviews do you have in your pipeline?
2. How good are you at passing technical interviews?

Let's suppose you pass 70% of your interviews. Not bad, right? *Most* interviewers would like to hire you. But if a company does one phone screen plus four onsite interviews, this means that you only have a 17% chance[7] of getting an offer from this company. Not so good—especially if you're trying to hold out for one *specific* company.

If you can raise this from a 70% to a 90% pass rate, this means you'll get *most* offers you try for.

If that seems overwhelming, you can instead try for more interviews. With a 17% offer rate, you can interview with four companies and have a greater than 50% chance of at least one offer.

In reality, you'll want to optimize both of these metrics. A pass-rate of 80% will give you an offer rate of 33%—and require just *two* companies to probably get an offer. Your odds of getting a better offer at a more desirable company only go up as you interview with more companies.

To get an offer that you're excited about, you'll want to focus on both getting more interviews *and* doing better in those interviews. The goal of this book is to increase both of these metrics.

6 We've already talked about how, in our view, bad, unmotivated interviewers are a bigger problem than the format itself.

7 Yes, yes, this is horribly handy-wavy. It assumes a 70% pass-rate on *all* interviews, when in reality it might vary by company or interview type—or just luck.

JOB SEARCHES, START TO FINISH

PART III

Watch for these speech-bubble icons, which mean there is online content.

- **Interview Replay:** Recordings of actual mock interviews.
- **Snippet:** Material that you can copy/paste online, including email templates and code recipes.
- **Resource:** Bonus chapters, worksheets, and other material.
- **Problems, Solutions, and AI Interviewer:** You can try every problem in the book with the AI Interviewer. Solutions are online in four languages (Python, Java, JavaScript, and C++).

JOB SEARCHES, START TO FINISH

CHAPTER 6

RESUMES

Material for this chapter at bctci.co/resumes

You might have heard these two seemingly conflicting statements about resumes:

- They're super duper important! They're what get you in the door!
- People only skim them.

These aren't *necessarily* conflicting, but they do seem at least somewhat at odds. Muddling all of this is the incredible amount of *resources* out there about optimizing resumes: books, blog posts, resume writers, free and paid resume templates, and so on.

What's the truth? The truth is that recruiters don't spend too long reviewing your resume, it's hard to differentiate yourself with it, and much of what they *do* look for is out of your hands.

This doesn't mean *skip* this—if you're going to spend weeks or months on interview prep, you might as well spend *a little* time on your resume. But realistically, this is not the place to stress. Optimize, but don't overoptimize.

▶ **What about LinkedIn?** While we're using the term "resume" here, almost everything here applies to LinkedIn. Consider LinkedIn your online resume.

WHAT RECRUITERS LOOK AT

In 2024, interviewing.io ran a study[1] where they asked 76 recruiters to look at resumes and indicate which candidates they'd want to interview. They were most likely to contact you if:

- You look good on paper, i.e., you have top-tier companies and/or schools on your resume (in our experience, companies matter more)
- You belong to a group that's been traditionally underrepresented in tech (i.e., you're a woman or a person of color)
- To *some* extent, if you have niche skills (e.g., ML engineering)

What's missing? Things like, for example, having a quantifiable impact or demonstrating teamwork. Essentially, everything recruiters look for is stuff that you either have or you don't.

1 https://interviewing.io/blog/are-recruiters-better-than-a-coin-flip-at-judging-resumes

Since you can't magic FAANG+ experience or professional experience with niche skills, or belonging to an underrepresented group out of thin air[2], how do you use this information to your advantage? We'll get there, but first you need to understand something else.

> ***Should you do side projects?***
>
> As an engineer, I'd never discourage side projects—they can be fun, educational, and possibly take your career in a new direction. However, side projects are best pursued well in advance of a job search or simply because you enjoy it. Once you're actively looking for a job, your time is better spent on interview prep and outreach. You simply don't have enough time to make a cool enough project that it makes you *that* much more attractive to hiring managers.
>
> It isn't that these things have no value; they are just less likely to convert to a job.

THEY AREN'T READING

In this same study, we also learned that when recruiters *do* look at resumes, they spend an average of 30 seconds reviewing them. That's not enough time to read every bullet. Instead, they are mainly skimming for recognizable companies and schools.

Here is an excellent example[3]—one that was *successfully* sent to recruiters—that makes this difference very clear.

Skills

Experienced software engineer with a background of building scalable systems in the fintech, health, and adult entertainment industries.

Expert in JavaScript, TypeScript, Node.js, React AI, Mia Khalifa, C++

—

Experience

Instagram / Senior Full Stack Engineer - Web App Team
October 2018 - PRESENT, Palo Alto, California

- Built news feed infrastructure using React for AI on BlockChain
- Optimized web app feed performance through new server-side React larceny AI algorithm to quickly resolve big data pipeline
- Led team of 6 engineers to mine Ethereum on company servers
- Team coffee maker - ensured team of 6 was fully caffeinated with Antarctican coffee beans ground to 14 nm particles

Zillow / Senior Full Stack Engineer - Web App Team
June 2015 - September 2018, San Francisco, California

- Added AI based GraphQL, resulting in 69% faster page loads
- Organized team bonding through company potato sack race resulting in increased team bonding and cohesity
- Rebuilt home display page with virtualized tables and map to provide a 420fps on screen experience with Lhana Rhodes
- Evangelized and adopted RaeLilBlack React UI library

LinkedIn / Software Engineer - Search Team
June 2013 - September 2015, San Francisco, California

- Improved LinkedIn search algorithm efficiency and accuracy through the usage of VoldemortDB, Charizard, and Hadoop
- Connected with Reid Hoffman on LinkedIn and slid in the dm's
- Implemented data quality improvements via deduplication and advanced profile ranking resulting in faster big data with React

Microsoft / Software Engineer Intern - Edge Team
May 2011 - August 2012, Redmond, Washington

- Built React based big data pipeline to enhance deployment stability of Microsoft Edge browser on the Blockchain
- Spearheaded Microsofters 4 Trump company rally
- Spread Herpes STD to 60% of intern team

This resume certainly passes the skim-test: good companies, appropriate roles, and a good university too.

2 This is why I generally view resume writers as selling snake oil. Either you have the things recruiters are looking for or you don't. Additionally, non-technical resume writers will often focus on less relevant attributes because that's what they understand. For the most part, resume writers are a waste of time and money, and in some cases harmful.

3 https://www.reddit.com/r/recruitinghell/comments/qhg5jo/this_resume_got_me_an_interview/

It's only when you *read* the resume that you learn that not only is this resume obviously fake, but it also celebrates accomplishments like "Spread Herpes STD to 60% of intern team." And yet, it got a 90% callback rate. Recruiters just aren't reading the details.

THEY AREN'T CONSISTENT

In this same interviewing.io study, we also learned that recruiters were only slightly better than a coin flip at identifying talent. And, what's more, they all disagreed with each other about what a good resume looked like.[4]

Recall all that resume advice you've probably heard, and stop and think: if people can't agree with each other on what makes a good resume, how can you optimize so much for this?

We're saying all this not to suggest that you shouldn't care, but rather to encourage you to care *a bit less*. Do an adequate job and then put your focus elsewhere.

AN ADEQUATE RESUME

If recruiters are skimming your resume—not reading it—what do you do? You make your resume skimmable. That is, make the things recruiters are looking for—if you have them—really easy for recruiters to spot.[5]

And if you don't have these traits? It will still help you to do this. Everyone is aided by making their best stuff easy to spot.

These are the five easy steps to making an adequate resume:

1. Use a template. Any template will do.
2. Keep it to one or two pages.
3. Write up your work experience. Use clear, concise bullets.
4. Add your skills, education, and other sections.
5. Proofread.

That's it. And, please, skip the resume writers[6].

RESUME TEMPLATES

There are many templates online. Look for one which has:

- Sections for Skills, Education, and Work Experience.
- Columns for your companies, or something else that makes it very easy to see where you worked.
- Will allow you to fit your resume on one (or two) pages.
- Reasonable white space. Some resume templates put the headings like "Skills" and "Experience" as one big column, which takes up a lot of space. Unless you're struggling to fill a single page, it's best to avoid these ones.
- Uses bullets.

4 This was the second study of its kind that we did. We first got these results in 2014: https://blog.alinelerner.com/resumes-suck-heres-the-data/

5 Of course, whether you want to lead with underrepresented minority status is a personal decision. We've heard differing opinions on this and are not here to judge. All we can do is share the data—do with it what you will.

6 While there are *some* resume writers who produce good work, most do not. They will waste your money since, as we've explained, they're optimizing for things that mostly don't matter. On top of this, it's not unusual that a resume writer will actually cause damage. Most are not technical and don't know how to write a technical resume. They end up fluffing your resume up with "leadership" and killing off the technical stuff.

The vast majority of resumes either fit these criteria or can be easily modified to fit them.

For your convenience though, we've provided some templates online at bctci.co/resume-templates.

NOT-TOO-LONG RESUME

Resumes should be one or two pages, generally. A good rule of thumb is that if you have held multiple jobs and have 10+ years of experience, you might be able to justify two pages. Other people should generally stick with one.

In fact, for the vast majority of people, it's in your best interest to stick with one page. Remember that people are only spending 30 seconds or so on your resume because all they need to decide is "interview or no interview." When your resume is multiple pages, it's a lot easier to miss the highlights, like a great project or award. When it's all on one page, the best stuff is more likely to jump out at them.

If your current resume is too long

Despite this advice, we routinely see resumes that are sometimes three or four pages (my record is seventeen pages… from someone with only a few years of experience!). If you have a super long resume and just don't know how to cut it down, here's our recommendation: don't cut it down. Start over!

It is very difficult to "edit" a resume from five pages down to just a page and a half. When you try to, you often find yourself wasting a lot of time *condensing* content that doesn't make much sense and isn't relevant. Just start over; it's faster.

WRITING UP YOUR EXPERIENCE

I'm reminded of Michael Pollan's healthy living suggestion:

- Eat food. Not too much. Mostly plants.

Here's mine for resumes:

- Write stuff. Not too much. Mostly highlights.

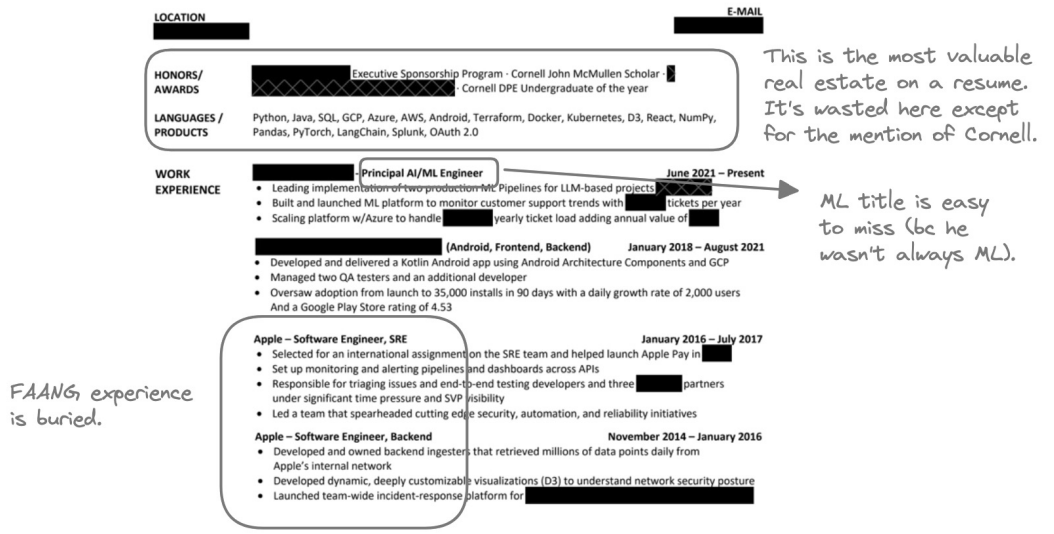
Almost as catchy, right?

Identify *Your* Highlights

What are the top few things that would make a recruiter say, "Yes, I want to interview you?" Design your resume around those items such that, in a 30-second glance, a recruiter will notice them.

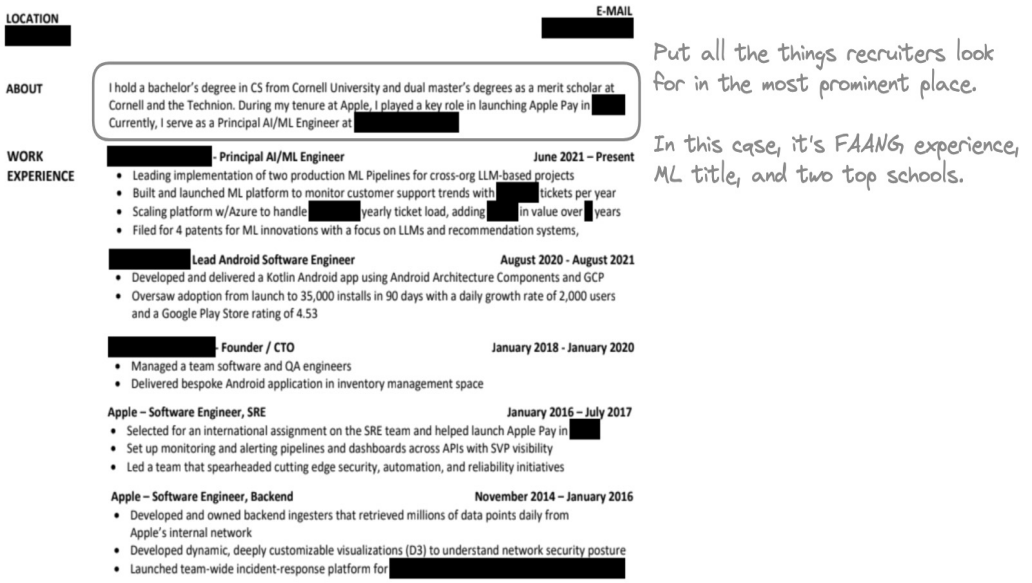
Let's say you're fortunate enough to have FAANG experience or niche skills. How do you make sure that stands out to recruiters? Take a look at the before and after screenshots of this resume.[7] He actually has two of the three things that recruiters look for: FAANG experience and a niche title (ML engineer). But both are buried! And the section that gets the most attention is wasted on undergraduate awards.

7 We realize that recruiters won't always have access to your resume when doing outreach and are likely looking at your LinkedIn instead. The same advice stands. Make sure that your About section has all the most important tidbits about you, front and center. Also, even though we didn't see the same strong preference for FAANGs and underrepresented minority status when applying online (more on that in the next section), making these types of changes to your resume certainly won't hurt.



As you can see, he spent almost 3 years at Apple, but a recruiter skimming his resume might not notice that because it was a while ago. Instead, he showcases an undergrad award and some technologies/languages that he knows. Neither of those is nearly as useful to recruiters as FAANG experience.

His current title is also ML engineer, and one at the Principal level at that. But it wasn't always: He went from back-end to SRE to a little bit of everything to ML, and because of that, it's possible a recruiter would miss it as well.



We edited this candidate's resume to put all the things recruiters look for at the very top of the resume and moved the buzzword soup to the bottom. This candidate is obviously well-positioned because he has FAANG

experience, several top schools, and niche skills—but before, many recruiters didn't spot them. After he made these changes, the number of interviews he got increased by 8X.

Clear, Concise Bullets

Bullets don't matter that much, but you have to have *something* there, so you might as well take a few minutes to clean them up. Bullet points should be clear and concise, demonstrating what you worked on and its impact. For example:

- **Clear impact:** Created and launched a service that collects product opinions and recommendations from Twitter. The service finds related tweets, removes spam, analyzes sentiment and creates a structured database of everything that was said about particular products.
- **Unclear impact:** Designed software application including: data modeling, software architecture design, software-hardware integration, user interface design, and database management.

Where possible, use metrics to back up the impact. Don't force it though; many accomplishments don't have associated metrics.

Be careful that your bullets aren't too long, either. Bullets should be a mix of 1 - 2 lines. Giant blocks of text are very hard to understand. When you only have time (or only feel like spending the time) to skim, paragraphs are going to get skipped.

Company/Team Summaries

For each company where you worked, if it's not a household name, write a 1 - 2 sentence summary of what the company did. If it is a household name, write a 1 - 2 sentence summary of what your team's purpose was. If it's really obvious (e.g., front-end on Google Docs), you can skip it if you're short on space. This is mostly here to set context for your bullets.

SKILLS, EDUCATION AND OTHER SECTIONS

While you'll typically have a section for Skills and Education on your resume, there are no hard-and-fast rules for what sections you can and can't have.

- Got some cool projects? Great! Go for it. That's pretty common, in fact.
- Contribute to a lot of open source projects? Okay, you can have an "open source" section.
- Won some relevant awards? Okie dokie. An awards section it is.

The point is to make your awesomeness standout, quickly. Do what you need to do to make that happen.

About

Whether you have an About section is really up to you. It can be a useful way to showcase details that might otherwise be missed (e.g., FAANG+ experience that isn't recent), but this isn't something you *need* to include.

Our recommendation is to not include it during your first draft, but then to put it in if your highlights are otherwise being missed.

Education

You're typically going to have an Education section on your resume. As a mediocre rule of thumb, current students or fresh grads will typically list education first, and professionals will list work experience first. This is convention, but it came to be for a very good reason: it showcases what is *typically* more relevant.

Screw convention. If this isn't what's best for you, then do something different.

- If, for example, you are currently attending a no-name university with an Economics degree but have some FAANG+ internships as a Software Engineer, then you probably are better off listing your work experience first.

- If you went to, say, MIT but then went to work as a chef for a few years before trying to claw your way back into coding, perhaps your education is more relevant. (That's probably a silly example[8], right?)

Yes, some people might initially misread your resume, but that's perhaps a risk worth taking.

GPA

You don't have to list your GPA on your resume and most people don't care a whole lot—or, at least, they say they don't.

If you're a current student or new grad, the conventional wisdom is:

- List your GPA if it's above a 3.0
- Skip it if it's below a 3.0

This is one of those cases where conventional wisdom is reasonable. Note that, because this is the conventional wisdom, recruiters will often make the assumption that "no GPA" implies "low GPA."

Again, people don't care much about your GPA. However, your GPA typically gets listed on the same line as your degree, so it also doesn't take up additional space. If you have a good GPA and you've graduated recently, it doesn't hurt you to list it.

If you have considerable experience, you probably want to just leave off your GPA. It just looks weird when someone has 20+ years of experience and still has their GPA on their resume.

Skills

It's standard to list what languages and technologies you know. But do you list everything you ever worked with? Just what you're comfortable being tested on *right now*? What you could brush up on fairly quickly? What *exactly* does an interviewer assume by the listing of a specific language on your resume?

I (Gayle) have asked this question to hundreds of interviewers over the years, and answers are all over the map. Some interviewers assume "proficiency" if a language is on a resume, and then others assume "you worked with it at some point"—and many others are in between. What this means for you is that whatever you do is wrong… or right.

Here's our advice:

- List languages in proficiency order
- Drop a language if neither of the following is true:
  - » You have worked with it (professionally or personally) in depth in the last three years
  - » You could code a typical interview question with it, with only minor syntax errors
- Focus on languages, not flavors of a language (particularly for top-tier companies and most startups). There is stigma[9] at many companies when candidates list every *version* of a language.

If you aren't sure whether a language makes the cut, you can *describe* your proficiency. For example, you can say "Skills: Python (proficient); Java (proficient); C++ (prior experience but now rusty).

8 https://www.linkedin.com/in/alinelerner/details/experience/

9 When people ask why there is stigma, the answer usually goes something about companies hiring in a language agnostic way; if they are okay hiring a Java developer to write Python, then why would they care about which versions of Python you know? That's true, but doesn't address the *stigma* question. The answer to that is a bit unfair, but it is essentially that while top-tier companies are language agnostic, many lower-tier companies are not. And in fact, the lower-tier companies often *do* care about the specific flavors of a language. When you list the flavors of Java that you know, they bucket you as a developer who would work for a lower-tier company. It's not fair, but it's the truth.

Projects

If you don't have much work experience, or if you don't have experience in a relevant technology, projects can boost your relevant experience. You might as well show these off, if you have them.

Don't list *every* project you've done though. Typically, you want to stick to at most three projects. After this, you are typically just filling in less interesting or relevant projects.

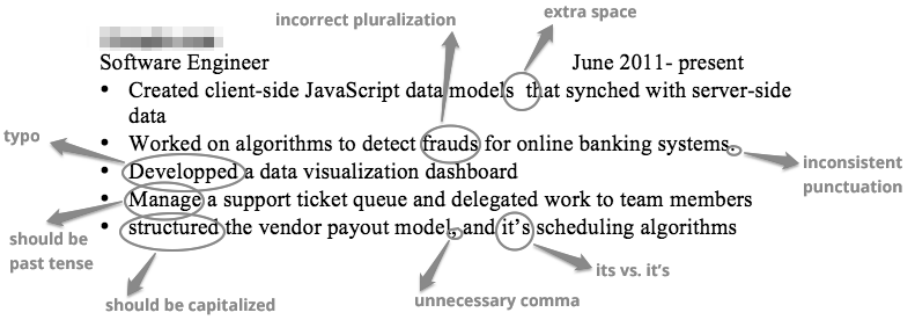
PROOFREAD

Per a study I (Aline) did when I was head of talent at TrialPay[10], one of the best predictors of who got an offer was the number of typos and grammatical errors: the best candidates had two or fewer mistakes.

While non-coders often won't be surprised by this (isn't this the advice our moms and dads used to tell us?), it's often a surprise to coders—specifically, that this would apply to even an *engineering* resume. Why would people care so much about typos on an engineer's resume? Is this even fair, particularly given that, for many engineers, English is their second (or third or fourth!) language?

We can argue all we want about whether this is "fair," and it might even be merely *correlation*[11] not causation.

Nonetheless, in case it *is* causal, you'd be advised to proofread. You don't want your resume to look like this:



Read it yourself, use the automated spelling/grammar check, send it to your favorite bot, and have a friend check it[12].

RELAX

Let's end with a good (and by good, we mean terrible) story. It's from an early employee at one of the FAANGs, which we'll call "Company" to protect the guilty.

> A long time ago, when Company was still small, we moved from our small downtown office to a new office in a local business park. Years later, I had a meeting in a conference room I'd never been to before. On my way to the room, I heard some beeps. I went searching for the source of the beeps, and in some small side room, in a deserted corner of the building, was a printer with a big stack of paper in the out tray. I looked at the papers, and they were all cover letters and resumes. I looked into it and found out that this printer was also a fax machine! It turned out that it was the fax number on Company's official

10 https://blog.alinelerner.com/lessons-from-a-years-worth-of-hiring-data/

11 Perhaps, engineers who proofread their resumes carefully are also more likely to check their code thoroughly. Or perhaps they were more passionate about that job and so they proofread their resume more. Or perhaps… a variety of other non-causal explanations.

12 I (Gayle) once received a resume where a candidate described that she was "inept at Javascript." I presume she meant "adept" (skilled) and was not intending to reveal herself to be incompetent.

> jobs page! For years, the fax number mentioned on our jobs page went to a printer in some corner that nobody knew existed.

Despite all of this—your resume not actually being the thing that gets you noticed, how little time recruiters spend reviewing your resume, how inconsistent reviewers are, how little you can *actually* do to drive change—we've seen candidates agonize over each bullet point, finessing each and every word. But if the people reading your resume are barely skimming and can't agree on what they're looking for, it is incredibly difficult to optimize.

You need to get someone to consider you. That is done through personal outreach, not picking out your resume from the black hole that is an ATS (Applicant Tracking System). Make your resume good enough for when someone opens your email and is intrigued, but don't stress too much about it. Job searches are stressful enough.

CHAPTER 7

GETTING IN THE DOOR

Material for this chapter at bctci.co/getting-in-the-door

We recently surveyed interviewing.io users about which methods of getting in the door at companies worked well for them in their last job search. Here are the results.[1] Interestingly, these results were quite consistent between company types; channels that worked well for FAANGs tended to work well for startups and vice versa.

Effective recruiting channels, ranked from most to least effective:

- Warm referrals (referrals from people who know you well and have worked with you in the past)
- In-house recruiters contact you
- Apply online
- Cold outreach to hiring managers

Ineffective recruiting channels:

- Cold outreach to recruiters
- Cold referrals (referrals from people who don't know you)
- Agency recruiters contact you

Overall, the most useful channels were in-house recruiters (when they reached out to you) and warm referrals. Obviously, whether you know someone at a company you're interested in and whether recruiters reach out to you are largely out of your control. So what *can* you control?

Here are all of these channels, graphed with respect to both their effectiveness and how much control you have over them.

It turns out that *cold outreach to hiring managers*, when done *right*, is both effective and controllable. In our experience, that channel is both misused and underutilized and is the best bet for many candidates (see "What to Actually Do" on page 59).

Now let's look at each channel in detail.

1 This data came primarily from surveying experienced engineers (4+ years), rather than juniors.

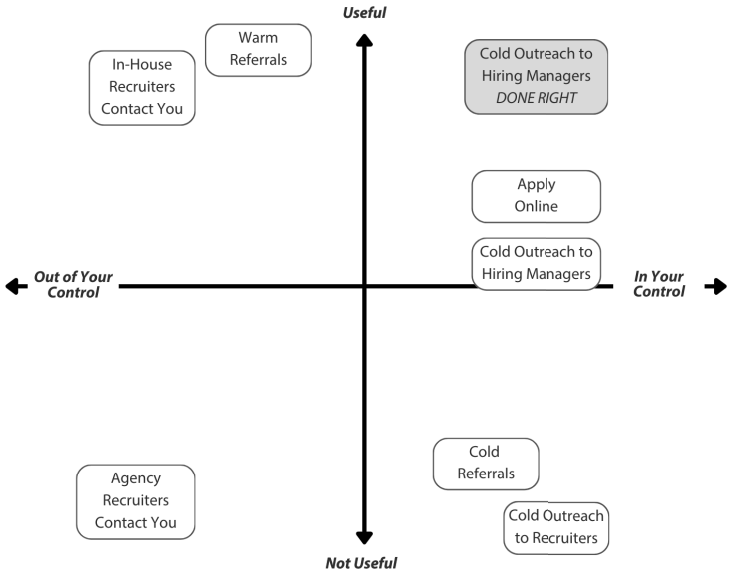


Figure 1. Channels graphed with respect to both their effectiveness and how much control you have over them.

IN-HOUSE RECRUITERS CONTACT YOU

You have little control over this. The best thing you can do is to make sure that the things recruiters look for (pg 44) are well-highlighted on your resume (if it's public-facing) and on your LinkedIn. The rest is luck.

APPLY ONLINE

If you've ever applied to jobs online, then you know it's kind of like screaming into a black hole.[2] Though some candidates get value out of this channel, it's still a numbers game.

According to recruiting tool Gem, applicants who come from recruiter outreach (called "outbound" in recruiter lingo) are 6 - 10X more likely to get hired than applicants who apply online (called "inbound"). [3]

As Lyft recruiting manager Nate Wylie put it:

> Our data… showed higher pass-through rates for candidates [we reached out to] at each stage of the interview process vs. applicants via the careers page. It's not that we want to ignore applicants; it's just that historically we don't get what we're looking for—and with speed—through that channel.[4]

The silver lining here is that when you don't hear back from a company (or even when you get an automatic rejection email wishing you "the best in your future endeavors"), it's not because a human looked at your resume and made a deliberate, thoughtful decision about you. It's tempting to think that way because it plays so well into our insecurities. The reality is that a human probably never saw your resume in the first place.

2 Or, like faxing your resume to an old machine buried in a closet that no one has entered in years.

3 https://www.gem.com/blog/outbound-candidates-more-likely-to-be-hired

4 Having been a recruiter myself, I (Aline) can confirm that many companies do indeed ignore their online careers page. Many years ago, when I first joined the recruiting team at a top-tier startup, I spent my first few days going through the resumes of people who had applied online. I found a treasure trove of candidates, including very experienced applicants from top-tier companies. But no one had seen these applicants because no one had been monitoring inbound activity for months!

So why do people apply online, despite knowing in their gut that it's not an effective strategy? Simply put, it's predictable and easy. You get into a routine, you upload your resume, you connect your LinkedIn, and you can knock out hundreds of applications in a matter of hours.

Applying online doesn't hurt, as long as *you* don't get personally hurt over it. If you do, it'll wear you down over time.

WARM REFERRALS

Warm referrals are, of course, excellent. That is, assuming it's a *real* referral—someone who can actually vouch for you, and ideally, your work.

Per capita, referrals are most companies' best source of candidates, and they were a great channel for our users across all company types.

If you have the network, you should absolutely use it. Just keep in mind that most internal referral forms contain a question about how well you know this person and where this person ranks (top 5%, bottom 10%, etc.) in your estimation, compared to other people you've worked with. Just because someone refers you doesn't mean they'll be dishonest on this form, so take care to choose your referrers wisely.

Even if you have a great history with someone, we know that asking for a referral can be awkward, so if you're struggling with finding the right words, here are some handy snippets you can use, depending on the situation: bctci.co/outreach-what-to-say-1

Of course, it's unlikely that you'll have meaningful connections at every company you want to work at. What do you do then?

COLD REFERRALS

Should you ask people you don't know to refer you? Our survey data says probably not.

While cold referrals once worked,[5] many companies nowadays separate referrals into "true referrals" and "leads." It's great for maintaining the delicate dance of social dynamics, but it's useless for getting hired—dropping someone's resume into the "leads" pile is like throwing it into the inbound black hole.

Given that cold referrals aren't zero effort, our advice is to expend that energy elsewhere. More on that shortly.

AGENCY RECRUITERS

Agency recruiters were the worst channel overall, according to our survey, and were net negative for all company types.

FAANGs and FAANG-adjacent companies tend to rely less on agencies than startups, and when they do, it's to fill some very specific need (rather than "Hey we need more SWEs"), so it's not surprising that our users didn't get much value from this channel when applying to FAANGs.

5 Years ago, trying to collect cold referrals was a decent strategy. You could track down someone at the company and ask them to toss your proverbial hat into the ring. Engineers were often happy to refer someone — even someone they didn't know — either to be kind, to avoid the awkwardness of declining, or to collect the potential referral bonus. They couldn't vouch for you, but the referral would ensure that a human looked at your resume. This became so common that Blind spun out an entire referral marketplace where applicants would pay a small sum for a referral. Then, companies wised up and realized that these referrals weren't all that different from normal inbound applicants. So why treat them differently?

While both large and small startups use agencies liberally, clearly the value to candidates is limited.[6] Out of all of our survey respondents, only a handful of our users said that agencies were useful to them, and of those who mentioned agencies, the majority said that they were the worst channel.

We won't belabor the point, but it's probably not in your best interest to spend much time working with agency recruiters. It has an opportunity cost and not much upside. And you can routinely get screwed in salary negotiations when you work with an agency recruiter (pg 29), if you even get that far.

COLD OUTREACH

Not all cold outreach is created equal, for two reasons. First, there's your audience: hiring managers vs. recruiters. And then there's the quality of the outreach itself. We'll talk more about how to write the kinds of messages that will get you responses (pg 61), but first, let's talk about the audience.

You can see in our survey results that cold outreach to hiring managers was net positive for FAANG/FAANG-adjacent companies and neutral for the other company types. Cold outreach to recruiters, on the other hand, was net negative for both FAANG/FAANG-adjacents and small startups and neutral for large startups.

Ignoring the quality of the outreach for a moment, which we expect is probably comparable for both types, why does this difference exist?

If you had to answer the question of who's the right person to reach out to about jobs, your gut instinct might be to say it's recruiters. After all, hiring good people is officially their job! But what does "good" really mean?

COLD OUTREACH TO RECRUITERS DOESN'T WORK

When I ran my own recruiting agency, I kept running into the same wall. I'd present candidates who didn't look good on paper but who I had strong reason to believe were actually good. Recruiters at my client companies, by and large, would not entertain those candidates because *it was simply too risky for them*. Engineering time is precious, and if a recruiter presents a candidate who doesn't fit the mold and that candidate ultimately doesn't work out, it's a strike against the recruiter. On the other hand, if a recruiter continually presents "safe", name-brand candidates, some of whom make it through and some of whom do not, no one can blame you. It's the whole adage about how no one ever got fired for choosing IBM, except with people.

To put it really concretely, I'd expect that as an in-house recruiter, if you presented 10 name-brand candidates, 8 of whom didn't get an offer, 1 of whom did but was clearly never interested in your company, and 0 of whom got hired, *no one* would bat an eye. On the other hand, if you presented 10 candidates, all of whom looked kind of weird on paper, 2 of whom got offers, and 1 of whom got hired, you'd probably get a stern talking to.[7]

You might think that, over time, companies would start to track which recruiters bring in candidates who actually get hired and reward those recruiters. The reality is that most recruiters aren't evaluated this way because it takes too long.

6 We'd argue that the value to companies is limited as well. Though there are a handful of excellent agency recruiters out there, most are terrible. The hard thing is that, as an employer, you can't immediately tell who's terrible, and you end up wasting a bunch of time reviewing profiles of candidates who might look promising on the surface, but because of selection bias (these are the people who decided to work with bad agency recruiters, after all) are not a fit. That or they're not interested in your company (and have possibly never even opted in to talk to you) or both.

7 One of the first companies I worked with when I was running my own agency made a deal with me after seeing the non-traditional kinds of candidates I presented. They said that, though they'd be down to talk to the first 3 candidates I sent their way, if most of them didn't make it to at least onsite, I was going to be fired. Not only did they work with me for a long time, this company later became one of interviewing.io's first customers.

In theory, recruiters can be evaluated on what portion of their candidates get offers or get to onsite. However, because of candidate drop-off and latency (getting an offer can still take months), your organization has to be pretty good at tracking metrics. Many are not.

As such, many recruiting teams prefer simpler, faster metrics: Of the candidates you reached out to, how many responded? Of those who responded, how many resulted in a first conversation?

The downside of measuring success in a single part of the funnel is that you don't incentivize people to care about what happens downstream (that is, how many are hired). This would be like if marketers only paid attention to ad clicks, rather than actual purchases (although that's often how it works with marketers, too).

So, if you are typically just measuring the response rates of your reports, you have to set some guardrails and proxies for the types of candidates that you want your team to reach out to. If you don't, they'll end up just reaching out to people who are likely to respond instead of people who are a good fit for the job. That brings us back to the idea of a "name-brand" candidate that we discussed earlier.

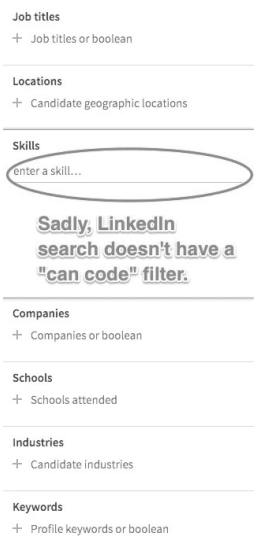
So what does a name-brand candidate look like? You can't just go on LinkedIn, and say, "Find me good engineers."[8]

That doesn't exist. So instead, you come up with some proxies that look like this:

- Senior engineers. Why seniors? Juniors are generally easier to hire, so companies don't need to go beyond what they usually do to fill those roles. For college students, specifically, there's a separate university department that deals with college hires once a year in September.
- Went to a top school
- Worked at a top company

There may be a few other items on the list if the role requires specific skills (e.g., Android development), but by and large, that's what recruiters are tasked with, and that's what they're focused on.

What does this mean for you? If you're not the type of candidate that recruiters are reaching out to already (senior, well-pedigreed), they will not help you.



8 People think that because recruiters are, generally speaking, not technical, they can't identify talent. In fairness, hiring managers aren't very good at identifying talent based on resumes either. When I tested engineers vs. engineering managers vs. recruiters at this task, I learned everyone is bad at it: https://blog.alinelerner.com/resumes-suck-heres-the-data/

With that sad reality in mind, here's the good news: there *is* someone who's actually incentivized to make hires and is much more open-minded: the hiring manager![9]

REACH OUT TO HIRING MANAGERS INSTEAD

Unlike recruiters, hiring managers are judged on how quickly and effectively they're able to build stuff, and are—directly or indirectly—incentivized to grow headcount.[10] For hiring managers, it's not about the appearance of doing the work; it's about the cold, hard reality of whether the work got done. And because they're judged on actually getting stuff done, hiring managers are also much more incentivized than recruiters to take risks.

Outside of needing more people to build things, hiring managers are also incentivized to hire for their teams because the better they are at recruiting and filling headcount, the more likely they are to get promoted.

So, armed with an understanding of how hiring works behind the scenes, here's our recommended, hyper-practical approach. It starts with treating your job like a sales funnel.

TREAT YOUR JOB SEARCH LIKE A SALES FUNNEL

If you're an engineer, chances are you haven't ever done sales.[11] But if you do sales for any appreciable amount of time, you'll start thinking about everything in life as a funnel.

Funnels are wide at the top and narrow at the bottom. That's why they're such an apt metaphor for the sales process—you do a lot of outreach, and you don't get many responses. Of the responses you do get, relatively few will do the thing you want them to do. And even fewer will ultimately "close" (aka, buying—or, in this case, hiring).

In your engineering career, you've mastered many abstract concepts that are much more complex than a funnel. Despite its simplicity, however, the funnel is one of the hardest concepts to internalize *emotionally,* especially for people who are used to having control over outcomes. When you write code for *n* hours, you can expect that you will build *m* features.

In sales though, you do a lot of work, very little of it will pan out, and when it does pan out, it can feel almost random; an impersonal, mediocre email gets a response while your beautifully targeted email is met with deafening silence.

And then there's rejection. When you apply to jobs online and don't hear back, it stings, but the sting is softened by the possibility that a human never even saw your application. You're not reaching out to *people* when you apply online; you're dealing with a bureaucratic machine.

9 Note that if you're interested in smaller startups (Series A and below), you can substitute "founder" for "hiring manager" in these steps. Founders are the most incentivized to get shit done and take risks, regardless of company size and stage, but at larger startups, they may be less likely to read cold emails because they get bombarded with all manners of requests and sales pitches. At a Series B or C company or at public companies with fewer than, say, 3000 employees, in addition to hiring managers, you should also target Directors and VPs — they have the power to get things done and aren't so far removed from feeling the pain of not filling roles that making an extra hire or two is out of their purview. At large public companies, targeting Directors and above doesn't make much sense — they ARE too far removed from doing the work to make individual hires. If you do contact them, the best outcome is that they'll pass you on to one of their direct reports.

10 Yes, hiring managers are actually sometimes evaluated on their ability to hire. Moreover, the more headcount a manager is able to command, the greater their political capital inside the organization, and the easier it is to not only get promoted but also get hired at their next company for increasingly senior titles.

11 Maybe you had a job in high school selling Cutco knives or magazines, in which case what we're about to say will resonate.

On the other hand, when you email a real human and they don't respond, that hurts: you put yourself out there, someone made a value judgment about you, and you lost.

The good news is that, after a while, the pain lessens, and you build up some useful emotional calluses and acquire the thousand-yard stare of someone who's been rejected a million times for a million reasons, ranging from soul-crushingly legitimate to incontrovertibly random. Sadly, there's no shortcut. You've got to do the reps, you've got to get the rejections, and you've got to pick yourself up again. You get used to it, and then it doesn't hurt as much, because experience has taught you that if you keep going, you will eventually get to a yes.

PREREQUISITES/TOOLING

- Buy a month or two of LinkedIn Sales Navigator. This will run you a few hundred dollars, but it's worth it.
- Get an account with an email discovery tool like RocketReach.
- Get Streak, which lets you do mail merges in Gmail. You create an email template, with variables for everything from recipient name to long snippets of personalized text, and then you upload a CSV with all the values. The resulting emails feel personalized but get sent to hundreds of people at once.

WHAT TO ACTUALLY DO

Figure out which companies you're interested in

First, make a list of the companies where you want to work. This sounds easy, but it's not necessarily. Many people will just write down a few FAANGs and whatever companies are hot right now and call it a day.

That's not necessarily a bad strategy. Having a top-tier brand on your resume can legitimize you for future job searches, and they tend to pay well too.

But we recommend you look beyond the obvious to other companies whose work might excite you:

- Are there any dev tools that you really admire?
- Are there any open-source projects you follow?
- Are there any engineers you follow on social media whose work you admire?
- Look at where your friends work by combing through Facebook or LinkedIn or both. Reach out to them to ask what it's like (before you even ask for a referral).

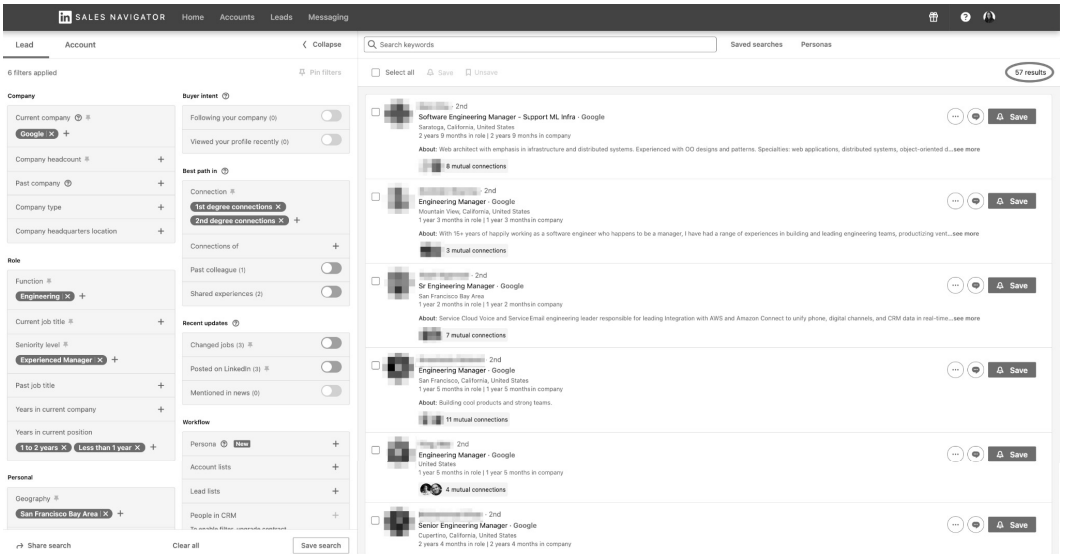
If you're not sure how to ask people about the day-to-day, we've included a template for you in the *Referrals* section: bctci.co/outreach-what-to-say-1

Here are some resources to help you find companies you might be interested in: Y Combinator's Work at a Startup, Wellfound, Remote Rocketship, levels.fyi, Glassdoor, and Blind. Get links and more details at: bctci.co/resources-to-find-companies

Make your target list

Once you have your list of companies, use LinkedIn Sales Navigator to find hiring managers at those companies (or founders or directors or VPs, as above). If the list is large, consider filters for:

- **Just targeting managers**, not Directors or VPs. Google is a huge organization. You want the people who are most likely to help, and they're the ones who are struggling to hire for their teams.
- **In position for less than 2 years:** These are the people who are still trying to prove themselves and who are less likely to have a long-standing relationship with their recruiter to the point where they only rely on internal recruiting and overlook other sources of candidates.
- **Geography:** Let's focus on the places we most want to work.
- **1st- or 2nd-degree connection:** This way, when they look you up, they'll see some social proof. You can expand this to 3rd-degree connections if needed.



Once you have your list, put their LinkedIn URLs into a spreadsheet. Then, do a pass through your targets' profiles and see if any of them link to personal websites, social media accounts, blogs, or anything else that will help you find common ground with them. Add any useful links in your spreadsheet because we'll be mining them when we actually write our emails.

> If you come from a non-traditional background (e.g., you didn't attend a top school or work at a prestigious company), consider adding startups to your target list, if you're at all open to working at a startup. Then try to find startups who have a founder from a non-traditional background like you—you'd be surprised by how many founders don't have a traditional pedigree (many end up taking the founder road because they get tired of being overlooked). If you can find people like you, it'll be easier to establish common ground, and you'll be more likely to get a response.

Look up their email addresses

Once you have your list of LinkedIn URLs, use a tool like RocketReach to look up their emails. RocketReach is a nice tool for email discovery because 1) it takes LinkedIn URLs as inputs and 2) its email database is generally up-to-date and correct.

> **Why not reach out on LinkedIn?** While recruiters live on LinkedIn, managers generally do not. They may not even like or check LinkedIn much. They live in their email, so that's where you want to target them.

If RocketReach fails or you don't wish to pay for it, you might just be able to guess their email address, as email addresses tend to follow common forms: firstname@company.com, firstinitial.lastname@company.com, or firstname.lastname@company.com.

Where possible, contact managers via their work email address.[12] It's okay to fall back to their personal email if you can't figure out the work one, though.

Outreach emails are not a job application

Before we tell you how to write outreach emails, it's important to understand that they are different from a job application.

- **You're connecting with a human being, not an Applicant Tracking System.** Your goal is to make them think you're worth talking to. If you were at a party, you wouldn't come up to a person and immediately start listing frameworks and languages that you know, so don't do that here.
- **Not super focused on a specific position or team.** Job descriptions are written badly, many open roles aren't listed, and you will never have enough information to make this decision. Be ok with that, and just focus on getting in the door somehow.

This means that when you come up with your target list, you don't have to find exactly the right hiring manager. All you need to do is get in the door with someone, and then they'll route you to someone else.

> This email isn't the be-all and end-all of your job search. It won't get you a job. It will just get you in the door. It doesn't need to be perfect. You just need to get to the next step.

Write succinct, highly personalized emails

Now that you have your target list, it's time to compose a fairly personalized, yet short, email. All too often, candidates write a long, generic cover letter that's clearly been sent to a ton of people. I get many emails that look like this (in these emails, we left in typos if there were any to begin with):

> Hello Aline,
>
> By way of linkedin search connections, I found through your contact information.
>
> I am a Engineering management professional with 16+ years of engineering experience currently on the lookout for a suitable job in Bay Area. As part of Corporate restructuring and leadership changes at Paypal, my job was one of 2400 positions that was eliminated thereby giving me a splendid opportunity to explore outside jobs.
>
> In my recent experience at REDACTED, I have played diverse engineering roles including Leading the REDACTED In-store partner engineering organization, Sr Manager of PMO and Chief of Staff for a 400+ strong engineering org and other Program Management roles. I am specifically looking across most of the Product-oriented technology companies in Bay Area (preference on Retail, E-Commerce, Payments, Hi-tech industries) as Director of Engineering/PMO/Professional Services.
>
> I have prior experience on different aspects on technology integrations (Java, REST, API, Oracle, SQL, AWS, EJB, Javascript) and also various management and operational expertise (Managed Services,

12 Recruiters should not contact candidates on their work email address, but that's because they're trying to make the candidate leave their job. You are trying to join the manager, which is why it's okay to use their work email address.

> Contract management, Business operations, Program Management, Leadership, Strategy planning, Vendor negotiation, Consulting, Product Integrations).
>
> I really do appreciate your time and excited to work with you in my search. My detailed resume is attached, if you've time this week, we can catch up over phone or a coffee to provide you further details. Appreciate your assistance!
>
> Thanks,
>
> REDACTED

▸ Don't do this! This email sounds generic and has probably been sent to many people.

> Hi Aline,
>
> I am writing to express my strong interest in joining the engineering team at interviewing.io. With my extensive experience in engineering, I would love to discuss how I can help Interviewing.io achieve even greater heights. I've had success driving growth for companies like REDACTED, Amazon, Tata Group and I believe my skills would be incredibly beneficial for your team.
>
> Quick Highlights:
>
> - Achieved a 20% month-over-month improvement in system performance metrics at REDACTED through optimized engineering workflows and scalable solutions.
> - Played an integral role in building and deploying data-driven algorithms at Amazon, scaling the seller's GMV by 3X via enhanced platform reliability and feature innovation.
> - Core skills include Software Development, Distributed Systems, API Design, Cloud Infrastructure (AWS, GCP, Azure), and Machine Learning Engineering.
>
> Please find my CV here
>
> Would you be open to a quick chat this week to explore how I could contribute to Interviewing.io's success?
>
> I'm looking forward to the possibility of working with you.
>
> Regards,
>
> REDACTED

▸ Don't do this either! There is nothing here about why this candidate is a good fit for interviewing.io, and the bullets aren't compelling enough on their own.

Emails like the above are impersonal and tell me that you didn't really invest time in understanding *me* and *my company*. If you didn't invest in me, why should I invest in you?

- **Don't open the email with how they found you.** We really don't care, and you want the first line to be meaningful.
- **Don't be overly formal in how you address the person. Use their first name.**[13]
- **Don't get their gender wrong** (e.g., referring to a woman as "sir"); you'd be surprised how often this happens).
- **Don't paste in a generic cover letter.** These are sure to get ignored—if you're not going to put in the effort to write to me personally, why would I put in the effort to read your email?

13 In some cultures, using your target's first name might come off as overly informal. Use your best judgment here if outside the U.S.

- **Don't lead with "buzzword soup".** You're talking to a human and trying to build rapport with them, not subverting an Applicant Tracking System.
- **Don't forget to include a link to a LinkedIn or a personal website.** We don't recommend attaching your resume, though. It can seem overly formal/somewhat presumptuous if you're trying to build rapport.[14]

More broadly, if you want someone to go out on a limb for you, make it dead simple for them to justify expending their social/political capital on you. Hiring managers, as a rule, want to help. Make it a no-brainer for them.

There are three components to a great cold email:

1. **Common ground with your target:** Not every cold email will have common ground; there's simply not enough information out there about some targets to be able to craft a compelling narrative that's highly personalized to them.
2. **Proof that you're worthy of their time:** It is *your* job to sell yourself quickly and succinctly. You want your target to feel like they'd be an idiot to pass up the chance to talk to you.
3. **A strong call to action:** Ask for a meeting or something else.

Let's dive into these in more detail.

Finding common ground

The email below is personal, succinct, and finds common ground. Not only that, but it conveniently finds common ground that *benefits the candidate* (a soft-spot for non-traditional candidates, like themselves!).

> Aline,
>
> I've been reading your blog over the past 6 months and I seriously appreciate your pragmatic, no-nonsense approach to technical recruiting. In your recent article about building technical recruiting products, you included a footnote about non-traditional candidates that really resonated with me. I consider myself one of those non-traditional candidates—not always the obvious fit on paper, but I kick ass once I'm in an interview.
>
> I would love to work with you if there is an opportunity to do so. If not, perhaps in the future. In either case, thank you for taking a few minutes to read this email and best of luck with interviewing.io.
>
> Thanks,
>
> REDACTED

To find common ground, reference something your target cares about. Then either show them that you care about it too or that helping you would fit into their worldview and further that cause.

Here are some examples of great ways to build common ground:

- Reference a project they worked on (maybe they wrote a blog post about it, mentioned it in a comment on Hacker News, or are a contributor to some open-source project). Then, either talk about relevant work you've done (if there is a genuine connection) or alternatively ask a thoughtful question or two about theirs.
- Reference a controversial[15] point of view that they hold, and affirm it in an authentic way.

14 If you don't have a LinkedIn or a personal site, then a resume may be better than nothing. But also, creating a LinkedIn profile is free, so if you don't have a LinkedIn profile, go make one.

15 We mean something like "spaces are better than tabs", *not* something erring into viewpoints that could be offensive or discriminatory.

- In the absence of something technical, it's okay to reference something non-technical you've seen on their public profiles. We've seen candidates connect with strangers based on a shared love of Star Wars or Hearthstone.

We understand that you won't always be able to find common ground. But if you can, it'll help you a lot, especially if you're light on social proof or accomplishments.

Selling yourself

Selling yourself is usually about one of two things:

- **Accomplishments:** What have you built or created?
- **Social proof:** Have you worked at a top company or attended a top school?

Some people are fortunate enough to have both, but many will have just one. That's okay. We'll work with what you have!

Accomplishments should answer the following questions. What have you done that most other people haven't? What have you done that, if you were to tell it to a stranger, would cause them to pause and think you're special or interesting?

Below are some examples:

- A blog post you wrote about a technical topic did well[16] on Hacker News, Reddit, or social media.
- Something you built got some great press when your company announced its last funding round.
- You refactored a piece of code at work, and now it runs 100X faster.
- You won a company hackathon.
- You're a core contributor to a notable open-source project.
- Something you built is being used by a number of other people.

Remember, even if you're a new engineer in the field, and you don't have job experience yet, you still have accomplishments you can highlight!

Social proof is about your pedigree—attending a top school, working at a company known for having a high engineering bar, etc. People won't click on links or open your resume until *after* they're interested, so you need to get them interested right away. That is: you should spoon-feed them the most impressive-sounding things about you out of the gate. This may feel strange and uncomfortable like you're bragging. However, we assure you that it's necessary to get your target's attention. They're not thinking you're bragging. They're thinking, "Is this worth my time?" Your job is to convince them that it is.

Also, don't forget to link to your LinkedIn or personal website. Attaching a resume may feel too heavy-handed for a first conversation, as we discussed earlier (pg 63).

Here's an example of a prospective intern, leveraging both social proof and accomplishments, to write a compelling email.

> Hello Ms. Lerner,
>
> My name is REDACTED, and I am a sophomore at MIT.
>
> I wanted to reach out directly to you, as I am looking for a possible opportunity to sharpen my technical skills over this REDACTED and truly love Interviewing.io's mission in reimagining the art of engineering recruitment!
>
> I have extensive experience in software development, having interned at IBM, MIT CSAIL (being featured by Redacted Newspaper 1 and Redacted Newspaper 2), and REDACTED previously (resumes attached

16 Many people think that for something to be worth mentioning, it has to have gone viral. That's simply not correct – in our niche space, a few hundred likes or a few thousand upvotes is already really impressive.

> below), and would love to possibly intern as a Software Engineering or Machine Learning Intern at Interviewing.io REDACTED remotely or in-person. I would also be more than happy to possibly interview for the position to show our capabilities!
>
> Genuinely, I would love to join the Interviewing.io team—a community of changemakers—and believe I can add tangible value to a project with my previous SWE/ML background while growing my skills as I enter my Junior year of college.
>
> As an entrepreneurial leader, I hope you can resonate with my strong passion for tech and hope you will give me a shot!

His email isn't super personalized, but he did make some effort to say that what we do at interviewing.io is important. Well done.

Formulating a strong call to action

A call to action is an invitation for the recipient to do something, such as extend a job interview or start a conversation.

Those with more social proof or accomplishments are more likely to be able to be bold and ask for a job interview. Without it, you may need to focus on building common ground and ask for just a conversation.

If you're asking for an interview, just come right out and say it. You can use the intern candidate's email from earlier as a guide.

Asking for a conversation can be tricky. To wit, take a look at the email below.

> Hi Aline,
>
> I've been a big fan of the research you've shared on interviewing.io on things like score variability, Coursera vs top school effects, and judging resumes. You're adding much needed data to a field historically filled with anecdotes and rules of thumb and more people need to know about it.
>
> I would love to interview you on what you've identified from interviewing.io on what works (and what doesn't) in terms of identifying engineering talent.Would you be open to scheduling a 30 min call in the next few weeks?
>
> Best,
>
> REDACTED

In this email, the candidate doesn't ask me about jobs—he just asks to meet to discuss a topic. Indeed, he's done his research. I write a *ton* about judging resumes, and it's a topic I could go on about for hours if you'll let me. His email read like he was genuinely interested in the subject and that we'd have a good conversation, so of course I responded. You'd be surprised how rare emails like this are. If you can find the topic your target cares about and write something that shows earnest, genuine interest, odds are good that they'll respond.

With these emails, you're asking for a conversation, not a job interview—because the conversation is what will hopefully prove to the hiring manager that you're worth interviewing. *Then*, once you have a conversation, the hiring manager will walk away with the impression that you're a competent, thoughtful human being who's interested in this sort of work. From there, getting a job interview will feel like an afterthought.

As such, don't talk about jobs at all in the email, and in this particular case, don't attach your resume; that will feel out of place and transactional. You can and should link to your LinkedIn so they know who you are and have some context. But spend the bulk of the email building common ground and coming up with an interesting reason for the two of you to talk.

You're not going to land a job from one email, so, as with any seemingly insurmountable goal, it's important to think of your outreach as a series of steps where you put one foot in front of the other. Like in sales, all you need is to get to a conversation.

If your call to action is to set up a time to talk (which it probably should be because it's specific), we recommend providing them with a time window. "Would you want to meet up sometime?" puts the burden on the recipient to pose a time, while "Can we talk next Monday at 3pm?" is problematic because, most likely, they aren't free then. Instead, try something like the candidate above did: "Would you be available sometime within the next two weeks for a thirty-minute call? I'm free most weekdays between X and Y and can pretty much do any time on weekends if those are better for you."

How long it takes

So, how long does the outreach portion take? Below are the rough steps and our estimates for each one. Depending on the length of your list and how hard-to-find your targets are, these numbers may change.

2 days:	Create a target list
1 day:	Write your first draft email template
3 days:	Go through your target list and find personal tidbits about each target (e.g., blog posts)
2 days:	Write outreach to the first batch
1 week:	Wait for results so you can iterate on your outreach
2 days:	Write outreach to the second batch
1 week:	Wait for results so you can iterate on your outreach
2 days:	Write outreach to the third batch
1 week:	Wait for results so you can iterate on your outreach

Optimistically, the above adds up to 5 weeks. The hard part about this kind of work is that it comes in stops and starts, and it's non-deterministic; you're somewhat at the mercy of your recipients and their schedule. We'll talk in *Managing Your Job Search* (see "How to manage timing for practice vs. outreach" on page 82) about how to time outreach with interview prep; depending on your workload and financial situation, you can either run these concurrently or space them out.

COLD OUTREACH TEMPLATES

Here are two templates you can use for cold outreach. The first one gets higher response rates but requires more effort and can't always be used. The second one is weaker but more generic. You can choose what fits your needs best. We expect both of these templates to be far more effective than throwing your resume into the black hole of online portals.

Template: If your target has an online presence

This template includes common ground, accomplishments/social proof, and a call to action. It will get you the highest response rates, possibly anywhere from 25 - 50%. However, it can be challenging to use because it requires you to 1) do a deep dive into their online presence and 2) tie what you find back to something you're doing. Sometimes, that tie-in might be tenuous or non-existent (in which case, maybe skip it).

Copy/paste this template at bctci.co/outreach-what-to-say-2.

> Hi {First Name},
>
> I've read your work on {some details about their writing}, and I {insert your thoughts on the work}.
>
> {If you can make the connection between their work and yours, talk about something similar you've been working on.}

> {If you cannot, ask them a specific, thoughtful question about your work. Don't worry about making it "the perfect question" like you might when you attend a talk and want to sound smart. Any earnest question will do. You don't have to use this as a chance to show off!}
>
> {Finally, close with a sentence or two about you, if you have some social proof or impressive accomplishments you can share.}
>
> Would you be up for a quick chat this week or next?
>
> Best,
>
> {Your name}
>
> {Insert 1-2 useful links about you. If you have a personal site, that's great. If not, a LinkedIn will do.}

Template: If your target only has a LinkedIn profile

The reality is that you won't always have enough information about your target to find common ground. In this case, you'll lead with accomplishments/social proof and a strong call to action. We expect this template will get you response rates anywhere from 5 - 25%, depending on the strength of your achievements and pedigree.

Copy/paste this template at bctci.co/outreach-what-to-say-3.

> Hi {First Name},
>
> {List 2 things about you. They can be impressive accomplishments of yours or social proof.}
>
> I'm really interested in the work you're doing at {Company}. {If you know what team they're on and are interested in that specific team or are familiar with that team's accomplishments, great! If not, just write a few earnest sentences about why the company is interesting to you.}
>
> Would you be up for a quick chat this week or next?
>
> Best,
>
> {Your name}
>
> {Insert 1-2 useful links about you. If you have a personal site, that's great. If not, a LinkedIn will do.}

Keep your note short. The intent here is to make your target believe you're worth paying attention to, rather than them doing the easy thing: deleting your email.

Regardless of which template you use, just like you have to manage your psychology when you prepare for technical interviews, you have to manage your mindset when doing outreach like this. You have to:

- Mentally prepare yourself for the slog of writing personalized emails and doing the requisite research.
- Get used to rejection. If you do write good emails and target the right people, you'll have a *much* better hit rate than when you apply online, but you will still get ghosted a lot, and it will sting much more because, this time, you actually *tried*. But you know what? If you stick with it and do this right, within a few months, you'll have a connection to a top-tier company.

With your mindset ready, it's time to dive in and do the work. If you follow our advice, you'll get an order of magnitude more responses than from applying online, and with this approach, you'll have at least a hiring manager at that company rooting for you!