Who Am I

• Matt Papakipos

• Lived and coded in Silicon Valley since high school.
  • My parents worked in tech too.

• My career: 20+ years in tech
  • Hands on coding jobs from ages 16-28
  • Then many engineering management jobs

• Resume
  • Early — Worked at a bunch of startups that failed, but I learned a lot
  • NVIDIA — Shipped the first programmable GPU. Hired 50 graphics engineers.
  • PeakStream — Founded this startup, sold to Google after 2 years. Hired 30 compiler engineers.
  • Google — Ran half the Chrome team, including all of Chrome OS. Hired 100 OS engineers in one year.
  • Facebook — Early Mobile Apps & Local Mobile Ads. Hired 200 iOS/Androidp engineers
  • Pied Piper Robotics — My new startup. Building home robots
Why Develop Software?

• Because it is an amazing job!

• It’s useful: We make products people use and love

• It’s powerful: We can change the world.

• It’s very creative, at many levels.

• It pays very well.
Where Can You Work?

- Main choice is startups vs. larger companies
  - Expected value of earnings (E[$]) is different.
  - Startups = High E[$], but high volatility.
  - Large companies = Medium E[$], less volatile.
- The correct choice depends on how well you can handle volatility and failure
  - I love it, but some people hate it.
  - Handling volatility well means learning to accept & learn from failure quickly.
Interview Process

- Recruiter conversations
- Phone screen with an engineer
- Onsite interview
- Onsite followup interviews
- Get an offer
- Closing conversations
Studying

• Before any interviews or phone screens with engineers, you should consider studying.

• What to study:

  • Get used to writing code on a whiteboard. Do it ahead of time.

  • You should know lists, trees, heaps, tries, hash tables well. Go do coding problems off the web, till you can do these comfortably.

  • Big-O notation. You should be able to analyze big-O behavior for anything you code on the board.

• Get a good night’s sleep the night before the interview.
Know Their Products

• Learn about the company’s products before you talk to the recruiters

• Use the products yourself if at all possible

• Have a list of things you love and hate about their products

• Show them that you want to come help them
Recruiter Conversations

• Don’t be a jerk to the recruiter
  • Lots of people are jerks to recruiters, but it’s not doing you any favors.
  • Recruiters are often pretty awesome people, and they seriously want you to get this job too. If you befriend them, they will help you and advocate for you.

• Talk about job location: Is this job in SF, or NYC, or LA?

• Talk about roles — but be careful, don’t get crazy
  • Asking for CTO job now is not a wise move. Be realistic.

• Talk about which group you would work in at the company
  • Depending on company, this can be super-important (e.g. Apple, Google)
  • You have to steer this conversation if you care which group you end up in (at most companies)
Which Group at Company?

• Depending on company, you may need to negotiate which group you’d work in before the interview
  • Yeah, it’s crazy, but this is traditionally how it works (Apple, NVIDIA, usually at Google)

• At a few companies like Facebook, you choose your group after you start work at the company
  • IMO it makes a lot more sense in 95% of cases

• So you have to talk to the recruiter to figure out whether group is chosen after starting job or not
  • If not, make sure you are talking to the right group before interviews or phone screens happen
  • Otherwise you will likely end up in wrong group, or — worse — get rejected and then it’s likely game-over at that company

• Frame the conversation with recruiter like this:
  • “I want to make sure I’m interviewing with the group in which I’ll be the best fit”.
  • If you’ll be happy in any group at the company, don’t be pushy. If you would only accept an offer from one specific group then definitely make sure they know that now
Phone Screen

- What to expect
  - 1-2 coding problems
  - Why do you want to work at company X?

- There may be more discussion about which group you are applying for
  - Be ready to steer that discussion, or not, depending on how strong your preferences are
  - But also be sensitive to whether the interviewer seems prepared for that conversation. They may not be, so react to how they react
Phone Screen

• What to expect
  • 1-2 coding problems
  • Why do you want to work at E-Corp?
• There may be more discussion about which group you are applying for
  • Be ready to steer that discussion, or not, depending on how strong your preferences are
  • But also be sensitive to whether the interviewer seems prepared for that conversation. They may not be, so react to how they react
Decision: Screening

• Now the company decides whether you’re getting an onsite interview or not

• So how will they decide?
  
  • How was your code? Did it work? Did you explain it well? Did you explain Big-O correctly?
  
  • Did you communicate excitement about the company?
  
  • Do they think you’d be fun to work with? “Culture fit”
Onsite Interview

• Typical format
  • Four 45-minute interviews
  • Each interviewer asks 2 coding problems
  • Most will give you time at end to ask them questions. Ask good questions. This is part of the interview!
  • At least one interviewer will ask questions to determine whether you are a jerk, and whether you work well in teams. Come armed with examples of how you’ve worked well in teams before.

• Pro-tip: Figure out whether you interview better in morning vs. afternoon and ask for interviews in that time slot

• Remember: The recruiter wants to hire you. Get them to help you do the best you can do.
Coding Interviews

• Expect problems like “reverse a linked list”
  • Basic CS algorithms, about 15 minutes to code it
  • Prove that it works.
  • Talk about testing approach.
  • Characterize Big-O for the algorithm.
    • If there’s a better algorithm, or you can prove that this is the best, show the interviewer
    • Do all of these without prompting! It’s never enough to just write code.
  • Explain what you’re doing as you code it up.
    • Partial credit is available if you show your work
Coding Interviews

• Expect problems like “reverse a linked list”
  • Basic CS algorithms, about 15 minutes to code it
  • Prove that it works.
  • Talk about testing approach.
  • Characterize Big-O for the algorithm.
    • If there’s a better algorithm, or you can prove that this is the best, show the interviewer
  • Do all of these without prompting! It’s never enough to just write code.

• Explain what you’re doing as you code it up.
  • Partial credit is available if you show your work
Proving that Code Works

• It’s not enough to just write some code
  • Most code written on whiteboards, even by top candidates has at least one error in it when first coded up
• You have to prove that it works. Two approaches:
  • Explain how you would test it to be sure it works. And walk through some test cases by hand on whiteboard to assure them that your code works
  • Prove it outright. I’m a math guy, so I like this. Induction proofs are possible in some cases. (Loops and recursion are really the same thing as induction in many cases.)
• Even top coders usually find and fix at least one bug during an interview, uncovered through self-testing
  • This is what acing an interview looks like: Finding and fixing your own bug without prompting because you tested it before declaring victory.
  • Never declare yourself done with the coding problem till you prove it or self-test it
Other Onsite Tips

• Definitely don’t wear a suit unless you’re interviewing at a company where that’s the norm for engineers (Wall St maybe?) Certainly not for Google or Facebook.

• You should leave each conversation confident you made a good social impression

• Don’t panic if you flub one interview. Lots of people get hired despite bombing 1-2 coding problems.
  • Seriously, self-sabotage is a major issue! Don’t panic.
  • Keep calm and ace the rest of the interviews.
Don’t Be Reluctant

• Interviewees who come across as reluctant to work at the company…

• … never get hired. Why would they want you?

• If you can’t communicate some enthusiasm about the company, you’re not going to get the job

• You don’t have to flatter 100% of the time

• But you do have to show that you get their vision and want to be part of making it happen
Architecture Interviews

• For undergrads, most tech companies will just do the kind of coding interview I described already.

• But more experienced candidates may get an “architecture” interview as well.

• They will ask you a much larger problem.
  
  • Example: “How would you build Twitter?”

• You should draw a block diagram on the board showing a full system software architecture. The main goal is to get everything on there: client, server, … the works. Come up with a workable design.

• You should be honest about which parts of that you know well and which you may know nothing about. (Never done server-side before? Say so!) But then proceed to design it all anyway, because they want to see how you do on something harder.
So…

How would you build Twitter?
How To Build Twitter

Client

Server
How To Build Twitter

iOS app  server APIs  PHP scripts

Client  Server
How To Build Twitter

Client

iOS app

server APIs

PHP scripts

Tweet storage

Client storage

Server
How To Build Twitter

Client

- iOS app
- Client storage
- Notifications

Server

- server APIs
- PHP scripts
- Tweet storage
- Apple Push API
How To Build Twitter

Client

- iOS app
- Client storage

Server

- Tweet storage
- PHP scripts
- Apple Push API
- server APIs
- Notifications

How is feed served quickly?
How does it scale to 1B users?
Architecture Interview

• Keep in mind that there isn’t one right answer

• Ask good clarifying questions

• Draw a convincing big-picture diagram

• Discuss the larger space of possible designs

• Explain your choices as you make them
  • You are demonstrating that you have a good design process

• Adapt based on discussion with interviewer
Architecture Interviews

• This is all about understanding a problem statement and breaking it down into something you could build.

• It’s also about showing that you understand the larger “space” of solutions to this problem

  • And can explain why you selected the one that you did.

  • Show them that your system design choices were not arbitrary

• Again, do this **proactively**, as part of your answer.
Onsite Followups

• That might be it for on site interviews, but in many cases they’ll ask you back for followup interviews

• Onsite followup can be anywhere from 1-4 more interviews

• This is a good sign. They haven’t rejected you. They’re trying to talk themselves into a “yes”

• Ask the recruiter why they are doing followups, so that you can focus on right things
Decision: Offers

• Now the company decides whether you are getting a job offer or not

• So how will they decide?
  
  • Will you integrate well into the culture? Did you make a good social impression? Did you dispel any fears you might be a jerk?

  • Did you show that you can code? And that you know Big-O? And that you know how to prove code works, and test it to be sure

  • Did you show that you’re excited about working at the company? Did you show them that you know their products are excited about how they will be evolving in the future?
Get an Offer

• Congratulations!

• The recruiter will call you
  • The recruiter is now trying to figure out how to get you to accept the offer

• Be excited, but not so excited that you forget that you need to review offer details before deciding whether to accept. Often the recruiter is trying to decide how much $$ to offer you, so don’t be too eager right now

• Here’s how I frame it to recruiters at this stage: “I’m excited, but I need to see the details on role, group, compensation”
Closing Conversations

• Now they try to get you to say “yes”
  • And there may be some negotiation about role, group, compensation.
  • Don’t get crazy about role. Your role is probably “software engineer”.
  • For most companies, you need to now do final negotiation about which group you will work in. Make sure to have this conversation now, or make sure company explains to you how group assignments will be figured out, and that you’re ok with it.
  • On compensation, if this is your first job after undergrad you probably don’t have much negotiating room. If it’s a startup, you should get someone experienced with startups to help you evaluate the stock part of the offer and probability that the startup succeeds.
  • If you know you need vacation in next six months, have that conversation now. Don’t surprise them with this request one month after you start; get an agreement now that it will be ok. Ask this last, right before you sign the offer letter. Once you ask this, any ability to negotiate compensation is gone.
Getting Rejected

• Getting rejected is no fun
  • But we’ve all been rejected now and then, so bounce back and move on

• Ask for feedback as to why, but don’t be surprised if you don’t get it
  • Most companies don’t provide much feedback due to fears about getting sued by you if they say the wrong thing

• When I think back to interviews that didn’t result in me getting a job offer
  • Most of the time it’s because I don’t like the products of the company, and it was hard to conceal that during the interview
  • Or there’s something about the company’s culture that turned me off, so I appeared reluctant
  • Companies rejected me because of those misfits, but really I should thank them
What If Interview Was Easy?

- What does it mean if the interview was easy, or didn’t ask many coding questions?

  - It means the company **sucks at interviewing**
  - So their engineers are probably not very good. Think about it.

- **Run away!** Life is too short to work on a “B team”
  - I have rejected working at many startups this way. I have never regretted it.

- Get yourself a better job offer, and then give them the feedback:
  - “The interview was too easy, so I have concerns about the quality of your team”
  - Now they’ll want you more. Sad, but true. Human nature. Run away.
What If Interview Was Easy?

• What does it mean if the interview was easy, or didn’t ask many coding questions?
  
  • It means the company **sucks at interviewing**
  
  • So their engineers are probably not very good. Think about it.
  
  • **Run away!** Life is too short to work on a “B team”
  
  • I have rejected working at many startups this way. I have never regretted it.

• Get yourself a better job offer, and then give them the feedback:
  
  • “The interview was too easy, so I have concerns about the quality of your team”
  
  • Now they’ll want you more. Sad, but true. Human nature.
Interview the Company

- Keep in mind that you should be interviewing the company as well

- Ask the interviewers hard questions
  - Do they like their jobs?
  - What do they think about company’s future?

- If there’s something that makes you reluctant to work at the company, talk about it directly

- If there’s some product they should build but haven’t yet, tell them about it. Show them you have good ideas.
  - But be careful they don’t think this is only thing you would work on, since that could be a negative.
Intern Jobs

• Intern-specific issues

• Freshmen internships
  • There are some specific Freshman programs like FB-U at Facebook

• Housing & social aspect is amazing
  • At large companies like Facebook & Google

• Big companies rather than startups
  • This is my general recommendation in most cases, unless you have done a couple big-company internships already

• Most startups suck at running intern programs
How Many Companies?

- How many companies should I interview with?
  - Recommend >1
  - But you don’t need a lot
  - Just a few — only ones that you are serious about
“Culture Fit” & Diversity

• This is a contentious topic

• Companies want to make sure you’ll integrate well into their “engineering culture”
  
  • This mostly means that you’re not a jerk. Seriously, great engineers who are jerks usually don’t get hired if it’s detected during interview or ref-check process

  • You do need to convince the interviewers that you’ll be an interesting fun person to work with.

• “Culture fit” can also be an excuse for prejudice and profiling, and result directly in lack of diversity in a team

  • Hiring only people like yourself is lazy

  • Evidence says that diverse teams produce better products, and we live in a world where those products are used by diverse people. You want that same diversity in the team that built it. Beware any company that does not embrace diversity, or uses “culture fit” as an excuse to grow a mono-culture.

  • If everyone who interviews you is a white guy, something is very wrong. Investigate and reconsider whether you should work there.
Startups vs Big Companies

• So what’s my personal recommendation?

• Generally do startups, not big companies
  
  • But consider starting your career with 1-2 big companies, to learn the craft
  
  • Startups typically make more $$ in the long term, though they are highly volatile in the short term
  
  • If the idea of your wages being volatile scares you, then don’t do startups
  
  • I think startups are a lot more fun. And I like the underdog / competitive aspect of building something new

• Consider starting your own company right now
  
  • What are you waiting for? Then you don’t have to go get a job. I’m serious
  
  • If you want more structure, join a Startup Incubator. They are legion, so do be careful to join a good one
Got Questions?

• Track me down. I love this stuff.
  • Interviewing, Hiring, Startups, Career advice, …

• I would not be where I am today without great advice I received from many people

• So my other big advice is: Find a mentor experienced in the field you want to work in, and meet with them frequently

• I’m matt@papakipos.com