Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

Announcements:

In a week or so, I will update this document again~!

I RELEASED MY FIRST YT VIDEO! HIGHLY RECOMMEND WATCHING THIS VIDEO TO GET A QUICKSTART ON THE ENTIRE PROCESS OF INTERVIEWING + RESOURCES + MY #1 TIP! If you like it enough pls subscribe and help me~ be a great thing to give back to me haha. But I do recommend it as I do interweave parts of it in this guide.

https://www.youtube.com/watch?v=m71kZqOj5VU&feature=youtu.be

https://linktr.ee/justinwlin
Link to my socials too! And also an easy way to reference my YT video + written guide in the future.

Who Am I / Why Am I Writing This:

My name is Justin Lin, I've gotten a lot of help from people online when I was getting an internship and now a full time job offer and I want to provide guidance to other people b/c I find it rewarding to see other people succeed.
I've interned at:

1. JP Morgan Chase my sophomore year summer as an iOS Developer intern
2. Amazon SDE Intern my junior year of summer
3. Gotten a full time return offer at Amazon
4. Gotten offers at Blizzard, IBM, and a couple other smaller companies.

I am proficient at:

1. Web Development
    a. NodeJS
    b. React
    c. Vue
    d. SQL/NoSQL Databases
2. Native iOS Development
3. Multitudes of other technologies have dabbled in or be proficient in.

https://www.linkedin.com/in/justinlinw/ <- if you want to add me on Linkedin send me a msg too as to where you found me :) Always cool to see.

This guide is **aimed mainly towards students** but will serve as a strong guide in general to the interview process.

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

I also recommend checking out another guide by a fellow friend named Edbert Chan, who is a L6 equivalent at Uber, or basically a senior engineer who has a 50 page guide on the tech interview process too: Edbert Chan Guide.

So you must be wondering? How did I get into JPMC + Amazon and seem so successful? BTW I want to note that I got into Amazon with a **2.9 GPA**, (lots of reason, including being hospitalized before my finals my junior year), meaning you don't need to be the smartest person in the world or that I am anything special. Instead, I just got lucky, and was able to take that opportunity b/c I had been preparing whenever I could on the side. I did a lot of self-studying, found a lot of great friends, burnt through dozens of resources,and through a lot of trial and error I figured out something that I think is more mechanical and replicable to a degree, and I want to share that.

Before beginning, I do recommend watching: (How to get into the Big 4)
It really encouraged me. The speaker was held back at school, academic probation, etc. and yet he got into Amazon, then he ended up working at Microsoft, and now Google in his professional career.

Again also quick start YT video I made on the entire process: Here
Notes:
If you want to read about my experience at getting into JPMC / Amazon I wrote a small thing here: Click here

I also recommend taking Edbert Chan's guide with a grain of salt due to a bit of caustic language lol, but I think it is pretty accurate on all levels even if more focused on higher level engineers.

BTW, if you see YELLOW HIGHLIGHTS, these are things that I consider the most important

Why should you Apply to FAANG?

So I think first of all, a lot of people wonder should they even try to apply to FAANG, or FAANG level equivalents such as unicorn startups or companies such as Paypal/Intuit/Salesforce/Datadog, and the answer is yes!!!

First if someone doesn't already know of this resource, you can check out the salary of these companies at: https://levels.fyi

But the gist is just think about it, on levels.fyi, and you will see that a FAANG employee will earn around 150K starting new grad/junior dev total compensation. This can scale up to companies like FB to 220K. FAANG companies' interviews are also insanely standardized. They aren't asking you to be an amazing engineer, they are asking you to have good fundamentals and problem solving skills, and like learning the SAT you can study these things.

Getting into a large company, they are also willing to train you. Working at JP Morgan, I had mentors that were spanish majors, math majors, or just non-CS majors originally. Working at these large companies, they will invest time and resources b/c they have the ability to, they don't care what you know, b/c they will teach you and you will learn. All they ask is that you **learn quickly** and that you have something to contribute even if it is just your enthusiasm to learn and work, and **that you care about what you work on**. THERE IS EXPECTATIONS, but that is the same for any company. And also for those wondering about are the hours insane, no. I basically never felt pressured to work beyond the normal 8 hours a day, I probably only worked 5 to 6 hours tbh, but it is pressure to get your work done and to learn as fast as you can.

However, let's say you try to apply to a smaller company, not saying it is bad, but as a new grad/junior I think there are better options b/c at these smaller companies they usually require:

- More knowledge
- Have less resources to train you
- Might have worst work-life balance (not always true, depends on the company)
- Get paid less

And it isn't the small companies' fault, it's just the reality of the situation that they don't have as much resources. So your goal, again, should be to get into FANG, or a similarly level company. Obviously working at a small company still does give you a lot of knowledge, I personally went from a startup to JPMC to Amazon, so it is still a very valid path.

P.S Everyone always says: I send out 100 resumes and get no responses. This is common, and also it just b/c they don't have the resources to process these resumes. But if you limit to just FAANG level companies, they probably will have a drastically better response rate b/c they actually have the allocated resources.

Great Link to How to Get a Job in Tech!
https://www.gautamtata.com/series/how-to-get-a-job-in-tech/

What language to use?

Python. it is faster, easier to understand, and close to pseudocode. It is not hard to learn and most explanations are done in Java or Python online.

Python though is much less syntax heavy and allows for heavier abstraction and fits an interview where the point is analysis, explanation, and speed. Also the speed at which you can write python is so important bc leaves time to think, review, check over your work when things are more abstracted away. Ultimately, do what you want, but I just always recommend python.

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

I also use:
http://pythontutor.com/live.html#mode=edit
To help me debug and understand problems and makes such a big difference in the learning curve too.

Resume

Formatting Resumes

First things first, if your resume is two columns, if it has graphics, if it has a picture, if you did it yourself in word and styled it, it probably looks bad. Your bullet points are probably bad if it is one sentence, or if you have more than 3 or 4 bullet points per experience you are probably too wordy. If you are greater than 1 page, cut it down (your resume is the summary of the most relevant things right now, NOT EVERYTHING, find this is a common problem with Masters and PHD students).

Oh also, no jargon like: worked on blah blah driver in order to do cross analysis on xyz, blah blah. Like, realize a lot of people reading this might not have your knowledge and that HR people also do read a lot of these resumes before they get passed up, so you need to make it digestible to read. And if you are just beginning, you probably have a lot of whitespaces, meaning you need to add projects or experience, or if ur bullet points are like less than 10 words each. These are the most common issues.

Great article btw, part 1 covering resumes, that also talks about ATS systems, which parse resumes and why graphics / formatting like above is bad.
https://www.gautamtata.com/series/how-to-get-a-job-in-tech/

What do I recommend in term of formatting:
I recommend you use Latex, which you can use sites like Overleaf to work with latex online, or just use the standard template in word / pages.

Some great Latex resume generators to get you started online, which I then usually download the latex and upload to Overleaf to edit so that I can add bullet points since I don't like some of the base format completely, but they are 90% of the way there.
Resume Formatters Generator:

1. https://www.rezi.io/
2. http://latexresu.me/

REALLY COOL LITTLE PUBLIC RESUME PARSER TO SEE IF YOUR RESUME CAN BE PARSED, though do take with a grain of salt since you never know what system a company may use, and I've seen wonky resumes or resumes that are completely project

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

based (which by this little website) would mean that none of it gets parsed, get pass into FAANG. BUT this should serve as a good criteria / guideline! :
https://itsjafer.com/#/parser

Article talking about the dos and don'ts:
Interesting note is that some ATS parsers, if I am understanding correctly like GreenHouse ATS, actually preserves structure and just gets rid of formatting. So where Lever projects don't seem to show projects for ex., it would show in other ATS systems.

It also seems like as the gautamata website points out that it is also looking for keywords / density of keywords, so it probably is closer to something like ATS -> Screens -> Match criterias -> gets filtered through manual reviews -> etc etc….

Again these are just GUIDELINES! No need to follow these so strictly, as long as it is a traditional format and readable, usually is ok. I did completely fine not over optimizing for these things.
https://www.jobscan.co/blog/lever-ats/

https://www.jobscan.co/blog/greenhouse-ats-what-job-seekers-need-to-know/

https://www.linkedin.com/pulse/beating-machine-how-get-your-resume-hands-human-recruiter-kunal-kerai/

Great Reddit Thread about these floods of resumes and how a recruiter processes it.

https://www.reddit.com/r/cscareerquestions/comments/inrex1/ive_reviewed_thousands_of_applications_for/

https://www.reddit.com/r/cscareerquestions/comments/irjodn/ive_reviewed_thousands_of_applications_for/?utm_source=share&utm_medium=web2x&context=3

Bullet Points of Resumes

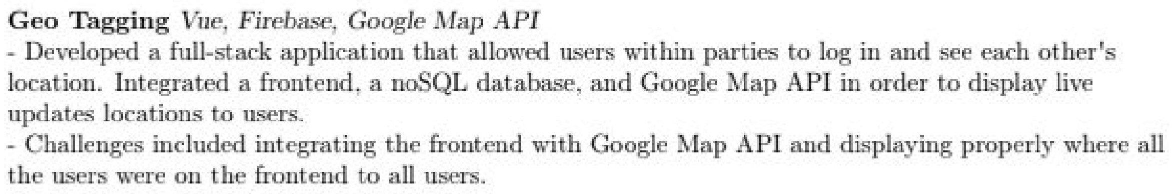
Also, you should make sure that your bullet points are strong / well written, here is a great article covering that:
https://2by22.blog/overhaul-resume-highly-effective-tips/?fbclid=IwAR27MQ5oQ7R4v0RiZj3ldHva5dWqMcpLCkAilW6tVKJ0MBkx8_aJL9tWSQE

I also personally recommend some people to follow the format of what you developed + challenges as a very simplified bullet point if you cannot come up with one that follows the

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

above structure. And the nice thing is you can make even simple projects look very completed and flushed out.

Ex. This is a resume snippet that I helped one of my mentees develop and we followed a what she developed + challenge format.



Geo Tagging *Vue, Firebase, Google Map API*
- Developed a full-stack application that allowed users within parties to log in and see each other's location. Integrated a frontend, a noSQL database, and Google Map API in order to display live updates locations to users.
- Challenges included integrating the frontend with Google Map API and displaying properly where all the users were on the frontend to all users.

Some other tips, in terms of bullet-points:
Do not write!!! I worked at a company with 1 million dollar asset. I organized….

Write it neatly and concisely on **WHAT YOU DID, not what the company was or boring things like, I can use microsoft word ._.**

People can tell you are desperate at that point.
Example:
I worked on a react project to design a dashboard.

Can be improved by writing (following a bit in inspiration to 22 by 2)...
What did you do, why did you do it, if there are metrics you can qualify or quantitate it, or give it substance that you know what you are talking about.

Example reworked:
I developed a UI dashboard to help parse incoming data streams from our backend server, making it easier for clients to get a high level intuitive overview of what is going on. I was able to quickly pick up on React within only two weeks worth of time, create mock-ups, and deliver UI and functionality.

Here is my own resume:
https://drive.google.com/file/d/12KYw0IbxWK0mYh4z-XP-dT5w5617pfbQ/view

You will notice in my own resume, I like to add an additional bullet point at the end of every experience stating:
Technology: React, Redux, etc etc…

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

Don’t have enough things to talk about on your resume?

What happens if you don’t have enough things to put on your resume? Do projects!!!! I have talked to freshmen and sophomores with nothing on their resume, and all I do is tell them to take a Udemy course and to just follow along with the projects and then bam, instantly 3 to 5 projects to instantly put on your resume.

Some recommendations:

1. https://www.udemy.com/course/complete-python-developer-zero-to-mastery/
   a. Scripting with the Python section under this course is a great place to just instantly gain 4 projects in 3 hours that you can immediately put on your project.
2. https://www.udemy.com/course/build-web-apps-with-vuejs-firebase/
   a. Also another great course with 3 full stack web apps, all written in Vue, and very easy to learn it quickly.
3. I ALSO HIGHLY RECOMMEND ZERO TO MASTERY ACADEMY :D
   a. I am a mentor in it! But it is one of the best places I think where the academy monthly pricing gives a lot of access to the Python Developer course, React Developer course, a Mastering the Coding interview course, and a Big Tech Coding interview course so it definitely is more cost effective I feel in terms of resources and very highly quality!
4. Scroll down to my Udemy Section to find out how to get Udemy courses for cheaper + only buy Udemy courses between it’s “fake sales” of 9.99 to 12.99, never pay more.

Nots:
Side addition (someone did note you can have nice looking resumes two columns, colors, etc). While true, I find most people are unable to accomplish this and use fancy resume templates they find online with meaningless bars and charts, too much wasted space, and so on.

Also the above might make it harder for some ATS systems at large companies to read. So while yes, you can have a great resume, just a standard template is also great, lower effort, and still gets the same practical benefit.

What do companies look for on the resume during an interview?

Generally, from my experience, some sort of full-stack understanding or project that shows the knowledge that you say you have chosen to specialize into.

Some personal things I have encountered during a behavioral interview:
Ex. Questions:

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

- What did you do with this project?
- Why did you do it?
- How did you do it?

Ex. of detailed questions I got ask on my projects:

- Do you know what an API is?
- What is an HTTPS request
- How do you connect a frontend to a backend
- Do you know SQL versus noSQL
- What is the difference between a frontend and backend
- What is Test driven development?
- What is CI/CD?
- How did you store the passwords?
- On this project what were difficulties you encountered?
- How did you do state management?
- Did this ever have any applicable use or what was the end goal? [Usually I say to learn and to use it as a template b/c my dad runs an ecommerce business so learning payment integration and such is helpful for me]
- Etc etc etc.

To people who are experienced btw, I know the above questions are basic questions, but to a lot of people if they've never worked in any development before a lot of people don't know the answer to those questions.

So what do you do…
You get an udemy course :) follow along and do a simple project on the side, so this way you know how to talk about it, and then you just keep studying on the side for interviews.

Timeline for Summer Internship/New Grad

Speaking from a student perspective, if you are going for an internship, your timeline to apply is between **September 1st to September 20th. (FREAKING RECORD THIS)** AND DON'T WAIT TO APPLY B/C U NEED TO GET UR OAs ASP, and get interviewed afterwards, and acceptances are usually rolling.

For new grads, it's about the same, also mainly in September.

While there are positions that are always released earlier, sometimes they are just sitting there such as FB they leave their position out all year, but it doesn't mean that there are any open spots internally to apply to.

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

What this means is you should wait until mid August / September 1st to September 15th depending on the company to apply.

You should be applying when the job openings open in late August to mid September, maybe late September.

From there, you will get an OA (online assessment), in 2 weeks to a month, to weed out anyone who isn't prepared for interviews.

If you get past that, it is usually for interns 2 phone interviews before you get an offer.

If you are going full-time it is usually 3 to 5 rounds.

Some companies, depending on your school, ex. Microsoft / JP Morgan, tends to try to do as many in person interviews as they can, so if you schedule early they will send you an invite for an in-person interview at your school.

How to Study:

The optimal time to study is 3 to 4 months before September if possible, but if not, it is fine, just start studying NOW!!!

I HIGHLY RECOMMEND GROKKING THE CODING INTERVIEW [me / Edbert guide, who is also senior engineer at Uber, recommends this]. If you spend money on anything this is the resource that is worth it. So my guide below will be predicated on that. (You can view the Resources section on more resources that I recommend)

Why should you study Grokking the Coding interview? B/c it breaks down interviews into 15 distinct patterns, and by "CHUNKING" the information, or dividing information like that, it gives you a systematic way to approach a problem. It isn't always going to work, but 90% to 95% of the time, you will find that these patterns will work. (if you don't like it, you do have a 2 week return period I believe - but check their refund policy).

You need:
1 month to finish Grokking
1 month to go through Algoexpert / mix of Leetcode (the objective here is to start covering grounds that Grokking might miss, and actually applying what you learnt by CODING IT OUT), b/c i already know that people won't be coding out everything in grokking.
1 month of mock interviews

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

Short on time:
If you are short on time for Grokking I recommend the following sections in order: Sliding Window, Two Pointer, Fast and slow pointer, Tree breadth first search, tree depth first search, Subsets, merge intervals, Modified Binary Search, Cyclic Sort, Two heaps, Top K, Dynamic programming, LinkedList, Topological,

The minimum chapter to get to would be merge intervals that should be ok for 80% ur interviews, then up to Top K would be the next milestone, and finishing up with DP and topological and linkedlist are probably the least important.

Arguably, Linkedlist could be moved up to after modified binary search, but I'll leave that to someone's own discretion.

Probably **Sliding Window to Subsets in my short on time section** would be the most important chapters to study to just get to the level of **Online Assessments!!** If you just started studying in September and just trying to pass the online assessment, and using the time before you hear back for a scheduled interview to keep studying the rest.

REALIZE THAT MOCK INTERVIEWS IS VERY IMPORTANT. Being able to type out a diagram, being able to click a space bar and show a "pointer moving", these are very important minute things that you won't be used to if you don't practice mock interviews for telephone questions.

I find that if you get good at mock interviewing online, you also get quite good at mock interviewing on a whiteboard since a telephone interview needs much more organization than a whiteboard.

MOST IMPORTANT THING THOUGH…. Just start. And make it sustainable. You will go crazy, it's a lot of work to study this much, but you need to do it. So get a friend, just start 30 minutes a day if you have to, but you need to do it. It will get depressing and shit, but you got it! Watch the youtube video on the Who Am I section, lol, b/c that guy's video really kept me going.

NOTE:
Sometimes I get bored of Grokking, so I do Algoexpert on the side. You can definitely learn in parallel!!! I wouldn't do anything beyond easy-medium on Algoexpert tho, since Hard+ gets ridiculously time consuming. But Algoexpert video explanation might make it much easier for some people to understand and feel better! :)

Tips on studying / How to Study:

1) You need spaced repetition

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

    a) You will forget things. **You lose 50% of your knowledge after the first day of reviewing, so every 4 days you must do a review on what you have studied.**
    b) YOU NEED THIS, SPACED REPETITION IS HOW HUMANS LEARN

2) Study with a friend!!! You need to suffer with someone else… what can I say lol, suffering alone sucks, suffering with a friend is more bearable.

3) **Organize how you approach your problems**
    a) **Coding Template**
    b) **THIS IS SUPER IMPORTANT HAVING A STRUCTURE ON HOW YOU APPROACH A PROBLEM**
    c) https://www.youtube.com/watch?v=m71kZqOj5VU&feature=youtu.be
        i) **AGAIN WATCH MY VIDEO ON THIS, IT WILL EXPLAIN HOW TO USE THIS CODING TEMPLATE**
    d) This is a template that I've made of my experience
        i) I try to analyze time/space, figure out what tools I have
        ii) Then I LITERALLY GO DOWN THE LIST OF QUESTIONS:
            (1) IS MY INPUT THIS
            (2) IS MY OUTPUT THIS
            (3) DOES THIS MAKE SENSE ETC
        iii) YOU NEED THIS B/C QUESTIONS AND NARROWING DOWN POSSIBILITIES ARE SUPER IMPORTANT

How to Get Better at Doing OAs?:

WIP But some notes below:
1. Do more Algoexpert
2. If you have Leetcode Premium they have a Company Explore section / Mock section where you can take mock OAs that have shown up before
3. Do the leetcode competition every saturday night, and try to solve at least 2 / 5 problems, so that you can raise your sharpness ability to think on the fly.

What to do when you get an interview?:

FIRST, REALIZE THAT THE PHONE INTERVIEWS / INTERVIEW ROUND IS ABOUT CONSISTENCY AND EXPLANATIONS!!!!

The reason why I recommend everything the way I do is BASED on the idea of trying to increase the consistency and structure of interviewing + having great explanations + NOT JUST SOLVING THE PROBLEM. You solving the problem, but the interviewer not following along is horrible.

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

Anyways, moving on…
If you get an interview, you should be ready, so accept it as soon as possible.

If not ready, you can technically push it back by 4 to 6 weeks for more studying, but this is why I recommend to just start studying as early as possible b/c you are giving more people chances to take spots. But I've seen people get accepted as late as mid november, so don't stress too much if you need to, just the bar will rise higher since there is restricted spots.

Also, let's be honest, if you push it back by 4 to 6 weeks, you will still be an anxious mess b/c you probably understudied, and even if you have been studying, those 4 to 6 weeks might not make much of a difference **unless you are just grinding those company guides on leetcode / doing a lot of mock interviews.**

Before or leading up to the interview, you should start doing mock interviews with friends, and take turns doing mock interviews. You both can throw problems at each other that you guys are familiar with, and even if the other person already knows it, it is still important to do it b/c it is more about EXPLAINING YOUR THOUGHTS and being under pressure.

What do I do When I am not passing interviews:

This could be multiple reasons. Maybe you are studying a lot or you are grinding those 1000 leetcode questions, but the problem is probably you are not consistent and/or you are not explaining your thought process well.

At the end of the day, you need a systematic way to approach problem for consistency (and also for your own sanity), but also be able to explain what is going on in your head.

Things to fix:

1) Fix your behavioral interviews.
    a) Are you sighing at any point?
    b) Are you smiling?
    c) Do you have a compelling story / way to talk?
2) Do you have a strong template to explain your thought process when you go through the interview?
    a) **https://docs.google.com/document/d/1WojFLF2HEn09sKTFoOwJv81MYxnF2pSUi4ufL7kWYYg/edit?usp=sharing**
3) If you aren't passing OAs,
    a) you can practice situations such as Leetcode Contest every Saturday night, where you get 3 problems to solve in 1 and a half hours. You usually should be

able to solve the first one and second one with some struggle, and the third one is impossible unless you are a pro leetcoder, lol.

i) But do this with friends, or get a group to do the above

b) If you aren’t passing OAs you can also just do more mock online assessment questions on Leetcode or Algoexpert also has a coding assessment section to practice on.

Tips on How to Get Cheap Udemy Courses

1) Udemy.com is a great place to get courses
2) They are on “sale” all the time between 9.99 to 12.99. They have dynamic pricing models so if you are a new user “suddenly a sale” or that sometimes they will have “surprise sale” from 199 to 9.99 lol. If you ever see it not on sale just go on incognito and you can see this “sudden sale appear” b/c they are using cookies to track.
3) If you have a VPN you can set yourself to turkey on a new account on Incognito -> will convert the cost to 24.99 turkish dollars for a **new account**. After you make your first purchase will lock the location. If you open up VPN on your phone using the Udemy app, a new account too, or if that account location has been locked, will also automatically show in USD pricing instead of turkish pricing, 3.99.
4) You can GIFT your main account the course too lol instead of purchasing it to your turkish account. So make a new turkish account, gift to main account.
5) Also Udemy has the most generous easiest return policy of 30 days. You just copy and paste the link for your course and they will refund you. I tend to keep the courses as references though for the future and b/c i appreciate the instructors, lol. But if you do end up hating a course, that 30 day money back guarantee is really great!

Another tip is what you can do if no VPN just go INCOGNITO -> MAKE A NEW ACCOUNT -> YOU ARE A NEW USER “WOW” -> MAGIC SALES DISCOUNT -> Buy a course gifting it to your main account!

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

Resources:

HIGHLIGHTED RESOURCES ARE #1 RECOMMENDATIONS!!!

Side note: Resources are definitely worth the money and will be paid back a lot if you end up getting a job anyways :) Don't be stingy, you are a software engineer, and making probably the top salaries in the US even if you don't get into FAANG.

Github Student Pack
https://education.github.com/pack
This is a great Github Student pack that has a lot of free stuff! Such as free courses to Frontend Master which teaches u Vue/React/Etc!!! :) :D

This is an academy my one of the Udemy instructors that I love alot. If you are thinking about doing multiple courses I recommend this person:
https://bit.ly/2QdRWfM

Great article covering the entire process:
https://www.gautamtata.com/series/how-to-get-a-job-in-tech/

Great way to do mock interviews online:
https://www.pramp.com/dashboard#/schedule

Facebook Groups:
Subtle Asian Tech + Subtle Asian Networking

Under the mentor tab there are a lot of people you can message to ask them questions, and people are kind enough to usually give you advice on how to prepare, what they do etc. You don't need a referral, what you need is information. Also being a part of a community is very important! And being able to ask questions is quite good too :)

Grokking the Coding Interview on educative.io
Note: Grokking on educative.io has now turned into a subscription based service versus paying 1 time. B/c of that I recommend you to ask friends to split the cost, or to balance what you think is worth the cost or not. They do have a refund policy, you can look into so look into that if you end up not finding it is worth it.
One of the best systematic ways to approach coding interviews. It really prepares you in patterns on how to approach problems. Some of the ways are not optimal, but it does give you a

tool to fall back on. And you can use it in collaboration with youtube videos or other sources in order to maybe see other ways to do it.

Grokking the Dynamic Programming Interview
Probably one of the best resources in terms of understanding Dynamic Programming. It definitely takes a while for it to click, and it took me a bit to understand what is Knapsack and what is the pattern it was trying to explain, but once it does, things begin to click. But tbh, dynamic programming questions at this difficulty is quite rare, and almost never comes up.

Algoexpert.io
While it isn't structured as much as Grokking the coding interview, it has an excellent question bank, explanation, and very helpful crash course to data structures / algorithms for those who are not well acquainted with O(n) analysis. However, I would suggest someone to do more analysis study on the side if you can't do it. But definitely a great resource to use in conjunction with Grokking the Coding Interview.

Leetcode / Leetcode Premium…
This is not my favorite resource. But once you are ready, you can go through the company guide for premium and those are quite helpful. The problem with leetcode is there are no solid explanations even through forum discussions, and just doing a lot of leetcode doesn't equate to consistency. But it is good as a question bank.

50 page Guide to Getting Gud:
This is an amazing guide written by someone named Edbert from a FB group, he is an amazing person and wrote a lot of details on studying here
https://docs.google.com/document/d/1eKirumpmwDWTtKCJKn2HuoQ2NavEfR41whmTyaQcio4/edit?usp=sharing

Linkedin Recruiter Networking Sheet (Sheet with 2K Technical Recruiters from different companies):
https://docs.google.com/spreadsheets/d/1Lhs-F-KEzyw0XmJd0P23-KiMh1RJ6Zj2SN5EDuOQ3XM/edit?usp=drivesdk

Great Article on getting a job in tech:
https://www.gautamtata.com/series/how-to-get-a-job-in-tech/

Udemy course on Data Structures / Algorithms
If you have never done Data Structures / Algorithms before, I recommend that you take Andrei Neagoie (Javascript) or a Python Data Structures / Algorithms course.

I personally think that Python is the best way to go, but I feel that any course you have, if you feel comfortable with programming it is very easy to switch between languages like Python, so even if a course is in Javascript you can translate the code. But if you do want a specific course

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

with Python, can search on udemy for such a course.

Just get any course 4.5 stars or better.

Looking at those 150K to 300K salaries as Motivation! LOL
https://www.levels.fyi/

Q/A

What is my opinion of CTCI / EPI, and why do I recommend doing Grokking / Algoexpert / Leetcode.

Ok, the first thing is that Grokking gives a systematic breakdown. Compared to all these other resources, it is the only one that I know of that breaks things into distinctive patterns which results in consistency and ability to understand approximately where things begin to fall intuitively.

The next thing is that CTCI is very limited in scope of it's languages, is older, and while still a great book, Algoexpert has video explanations and support for many different languages and built-in IDE. Basically, they both accomplish the same objective, but Algoexpert is just a better version of CTCI in every aspect.

The final thing is that EPI, or elements of programming interviews, is extremely dense. While it definitely is a great book, I wouldn't recommend this to beginners b/c it isn't friendly and that Grokking + Algoexpert will basically cover things to a similar degree. I would say that EPI is more if you need a challenge beyond Grokking / Algoexpert, or already have your fundamentals established, and in that aspect I did the one month and three month questions that the EPI book recommended.

Thus:
Grokking for systematic approaches and breakdown
Algoexpert for question banks and great explanations of problems that are very common and popular
Leetcode Premium for Company Guides / Questions pertaining to those companies

What about System Design? [take with a grain of salt since I do not need to do system design for an intern / new grad]
https://www.educative.io/track/scalability-system-design-for-developers

I recommend the above track along with Algoexpert System Design course.

Guide being rewritten soon to be cleaner and not be just my one hour of rambling on a google document.

If you are a student, you can search up, Educative.io student pack and it is included if you have a Github student pack for everything that is a non-interview course. So I would recommend to read the Web architecture 101 section of the system design track for free, and then go to Algoexpert and take the System design course there, and take the System design course on Educative.io