Guide to the Application Process

30DaysCoding guide to ace the Tech Interview

MARCH
2021

This document provides a complete guide to understanding the application process in tech fields – specifically for Computer Science jobs/internships and all associated materials. It contains a collection of information compiled by people who have successfully completed interviews at FAANG companies as well as a useful information, tips, graphics, etc. to help you put your best self out there.

As with all 30DaysCoding resources, this guide is provided to you for free with the mission of making the world's resources more accessible to all.

Enjoy!

Table of Contents

The document is divided as follows. We recommend reading the entire document, but feel free to jump around through the sections that you feel that you need to work on:

Understanding the Process

Understanding the application process pipeline is a huge part of having successful interviews - after all, it is impossible to do well in an interview if you do not even make it to that step.

When a company has a specific hiring need, they will make a public post describing the position and what they are looking for, and then they will post this on their company website careers section and/or on hiring websites such as indeed, LinkedIn, Monster, Glassdoor, etc. The company's HR department will work with whatever department has the hiring need to go through a process to find the right candidate to hire.

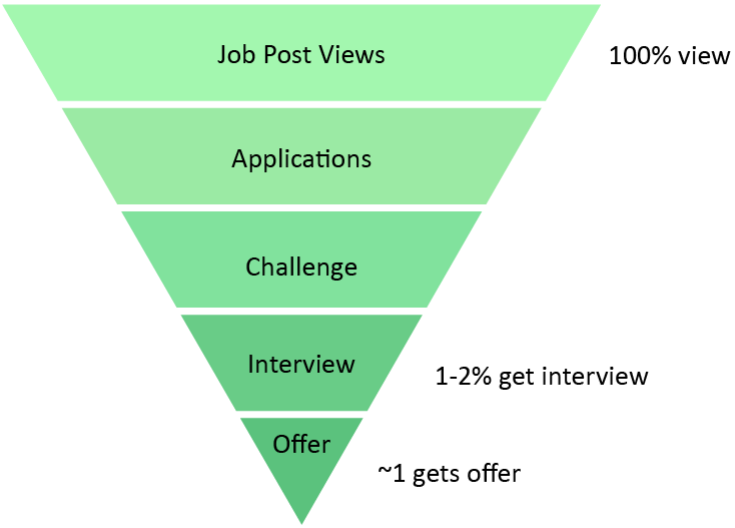
As you search these websites, you (and other potential job seekers) will filter through hundreds of these posts to find the ones you like based on location, pay, qualifications, duties, etc. Not everyone who views a post applies to the position, but this doesn't necessarily mean that one should apply to every listing that they see. Submitting hundreds of applications is very time consuming and can be very demoralizing – especially since it is likely that less effort will be put into each individual application and it is less likely to be hired.

> **Important Concept:**
> It is better to **put more effort into individual applications and hand tailor each application** to the specific job posting rather than using the same application for hundreds of postings.

Understandably, people without a lot of relevant work experience like high schoolers and those in the first few years of college may not have all that much to modify about their resume, but there are still things that can be done to maximize your chances of getting the job.

After receiving lots of applications or reaching a certain deadline, the company that created the posting will review all of the applications that have been submitted to them. If they are a large company or if they expect lots of applications, they may take lots of intermediary steps before directly interviewing the candidates.

For instance, most companies use computer programs to filter out weaker applications and leave only the ones with strong resumes containing relevant keywords and features that they are looking for. They can further filter applications by giving each applicant some challenges to complete to ensure that they are left with only the best candidates for the position. Then with only a few candidates left, they will interview the top few and send an offer to the one they decide on.



HR Statistics https://zety.com/blog/hr-statistics

The diagram above shows an example of the step-by-step pipeline of how an application goes to an offer for most CS applicants. Very few applicants end up at the interview stage and even fewer get an offer.

Do some research on LeetCode, Jumpstart, Reddit, or Glassdoor about how a specific company's interview process works. For instance, some interview pipelines might go something like: application, ATS, coding challenge 1, coding challenge 2, coding challenge 3, technical interview, offer.

However other processes might not even have a coding challenge and may go straight into an interview (this sometimes happens if your resume particularly stood out, the company just does not do challenges, the company uses tools/skills they don't expect you to already know, or you had some sort of referral).

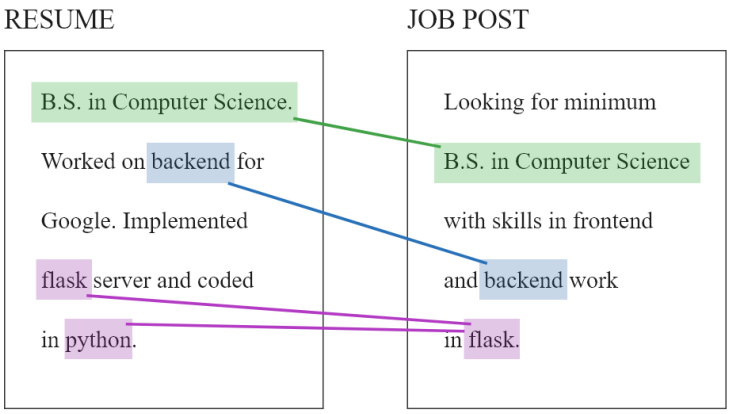
Job Materials and the ATS

This section discusses the materials that are part of the application and specifically how the company analyses these.

ATS:
One of the most common mistakes computer science applicants make with their resumes is underestimating the ATS (Applicant Tracking System) and the importance of tailoring a resume for a specific job.

Today, many companies use an ATS system to parse every applicants' resumes and lots of systems work by filtering the number of applicants that make it past every step. An ATS is essentially a program that

reads through every resume very quickly and gets rid of any resume it deems as unqualified by comparing keywords in the application and job posting. This step sometimes comes with some assistance from HR staff to double check the work, but for bigger companies it might just all be automatic.



Resume

Since at this stage it is read by a robot and not a human, you must pay attention to the following:

1. Make sure your resume contains **keywords** that are featured in the job posting as well as the skills that they are looking for. Read through the job post and try to add those keywords to skills or work history.
    a. Don't stretch the truth: experience in JavaScript != experience in React.js
    b. If the job post is looking for a specific subset of a skill you have, a rewording of something you know, or even just a skill that you have but didn't add to your resume for whatever reason, be sure to include it
2. Make sure your resume is in an **easy-to-read file forma**t like .pdf, .txt, .docx. When in doubt stick with .pdf as this is pretty standard.
3. **Keep it simple.** Avoid using fancy borders, weird bullets, word art, etc. These really don't serve much of a purpose to a human reader anyway and could really confuse the ATS. There's nothing wrong with having a simple Times New Roman resume like so
4. **Avoid pictures and graphics**. ATS systems usually can't really read graphics or pictures so if your resume contains screenshots, images, or anything that isn't text based, the ATS will not be able to read it and whatever is in it will not help you. Type out the entire resume.

Here is an easy to use template in word or latex. Many applicants will use templates similar to this one as it is proven to work and there isn't a ton of benefit in deviating from this.

30DaysCoding is working on implementing a free resume checker, but for now, the following are some limited free resources to see how your resume stands up to an ATS scan with a job description you are going for:
https://resumeworded.com/
https://www.jobscan.co/

Cover Letter

Some companies still require a cover letter as a part of their applications, and **even if it is not required then it might still be good to have one**. They show that you are passionate about the job and have more than just programming skills.

> **Important Concept:**
> The cover letter is a chance for you to give a recruiter **some extra information about why you might be a good fit for a position that might not be as easy to convey in a bulleted resume**.

The following document contains a great guide to resumes as well as cover letter samples and walkthroughs towards the last few pages:
https://ocs.fas.harvard.edu/files/ocs/files/hes-resume-cover-letter-guide.pdf

Since the cover letter is supposed to be one page long, you will find that they are actually not that difficult to write and it is once again important to tailor it specifically to the job you are applying for. Some guides and examples of cover letters are here:

1. Cornell Engineering
2. CICS UMass tips
3. CMU Cover Letter

Resume Review

We're currently developing a free local ATS software to parse and show all the skills listed inside a resume so that you can exactly know how your resume parses when you apply for a position. There are multiple options available online but most of them are paid and the free services do not tell you exactly how you rank. Check back on the website as we will be rolling out this feature soon.

Build your Profile

Making a solid profile before applying is a must. You need to present yourself as a strong candidate to recruiters and a candidate with a good profile will likely be more interesting than one without.

The first and most important rule is to **be wary of what you post publicly on social media**. Always be aware that potential (or even current) employers may be looking at what you are doing so make sure that there isn't anything out there (even on old accounts) that they wouldn't like to see.

If you have a unique name, try Googling it and seeing what comes up. Good things to see here are things like mentions in school newspapers, social media profiles, honor roll mentions, competitions, projects, personal websites, etc.

LinkedIn

Sometimes companies like to have recruiters do some research into your background after you apply to see if they should send you an interview. Alternatively, people who are just hiring for a position will do searches on platforms like LinkedIn to try to find people they think might be interested in applying for a new opening. Either way, it is critical to have a strong profile on social media and the internet.

By far the most popular professional platform is LinkedIn so the odds are that if a recruiter actually does research about you, **they will check out your LinkedIn profile**.

Fill out all of the information for the profile:

- Profile Picture: Select a nice and clear professional looking photo of just you - shoulders and up
- Background/Banner: Select a picture of your school if you want - leaving this blank isn't bad
- Headline: This section is surprisingly important as it is the first thing people see when they search up your profile. It should contain a short summary of what you do professionally or what you are looking for. If you have industry experience, hobbies, or projects you should write what you have experience in and after that you can add some vertical bars for multiple entries. Some sample headlines look like so:

    Firstname Lastname
    Honors Computer Science and Mathematics Student at University of Massachusetts Amherst

    Firstname Lastname
    Data Scientist | Open Source Contributor | Lecturer | Entrepreneur

Firstname Lastname
PhD Computer Science student actively seeking summer internship

Firstname Lastname
Founder & CEO of 30DaysCoding Software

- About: Fill in the about section with a relevant and professional summary about your education, work experience, and other interesting details.
- Experience: Enter all relevant work experience to the field you are interested in and fill out all the corresponding information. For instance if you had a position as a lifeguard in High School, it's probably safe to get rid of that unless you have nothing else. These entries can mostly be copied from your resume/CV
- Education: Be sure to fill in your majors, minors, degree type, awards, and graduation dates
- Skills: Add all the important skills you have - languages, software, soft skills, frameworks, etc. Feature the top 3 that are most relevant to the positions you typically apply for.
- Accomplishments: This is a great place to enter project work or courses taken for a student without a lot of industry experience.

Regularly log into LinkedIn and build your network. Connect with people you know and have worked with and make sure to have them endorse your skills. It looks very impressive to recruiters to see a candidate with a strong profile and lots of connections and endorsements.

Personal Website

While it is optional, it is often helpful to have a personal website to hold additional information. With companies looking at things like company fit and backgrounds of their candidates, you may find it worth your time to make a small personal site - especially in something like web development or graphic design.

With **graphic design, UX/UI, and anything art related, your personal website will usually be a portfolio of all your best works** . With other fields, a website can serve as an extension of the resume where you can add better descriptions and examples of more projects you worked on. You should try **creating your own website from scratch** as it proves you are capable of good design and have web development skills. A template is fine if you don't have a lot of web development experience, but be sure to really make it your own by adding where you can and removing unnecessary stuff.

A personal site allows you to go more in depth with your bio and add some more pictures of yourself as well as some of your interests and hobbies. Companies like Google especially love passionate and talented employees (they have a word for this: googly). Here is the chance to show off all of your good

web development skills as well as linking your projects and writing more about the explanations and skills.

GitHub Pages allows you to host a simple website (without a server or anything) from a public GitHub repository. Just name the repository yourUsername.github.io, make it public, name the main file index.html, make a few pushes to the repository and you should see your webpage at yourUsername.github.io. Take this for an example - download the template here.

Projects

Personal projects are very famous in computer science - everyone has several that they are always working on or worked on in the past. For people without a lot of industry experience, personal projects are a great way to bridge that gap to add lots of good keywords and experience to a resume. They also give a good example for a more technically inclined recruiter or engineering manager to **look at your code or for you to discuss during an interview**.

Group projects provide an excellent way of building useful team building skills, learning things from peers, and creating much more impressive work in less time. Hackathons and open source projects are both excellent ways to gain experience and come out with a great entry to your resume.

We highly recommend **participating in Hackathons** and joining random teams or just starting a project with some friends and working on it regularly. Pick something you are interested in or passionate about so you can be sure that you will have the motivation to see it through until completion. Sometimes projects go unfinished and this is fine so long as there is some working result available to view somewhere and that there is good writeup.

The biggest question is - How to get started?
A lot of students face this problem of coming up with ideas and starting with something new and this is where hackathons and GitHub can play a huge role. Start learning a **framework** for developing mobile/web apps if you're interested in software engineering or machine learning or AI if that interests you more.

Here are some hackathon guides to choose and ace your next hackathon:

1. Find a hackathon
2. MLH hackathons
3. Data Science Hackathons
4. Hackathon Guide
5. 25 cool ideas

Apart from hackathons, your college has coding, computer science groups which you join or even start which promote project based learning.

When listing the project on your resume, be sure to write down all of the tools, technologies, and frameworks used in that project. On a personal website or portfolio website you can provide a link to the project repository and make sure that there is a detailed readme, commented code, and a detailed demo or pictures to impress visitors. GitHub is an excellent platform for all of this work and is even a great skill to have since lots of workplaces use a git model for collaborative software development.

An example for a food delivery app you made:

> *"Name: A mobile app which tracks and delivers food right to your doorstep within a couple of hours. iOS, Android, UI/UX"*

Open Source learning has become the absolute beast in today's world and it's really important that you know about it. Open source simply means "original source code which is made freely available and may be redistributed and modified". Open source projects not only help other developers to code something on top of it but also inspires them to code something for our community.

Here are some guides to get you started with open source projects:

1. Google Open Source
2. First Timers only
3. Code Triage
4. GitHub

GitHub is the best resource online to work on open source projects. Open source projects are publicly displayed and can be used as reference for other developers (credit them for their work). It's the biggest 'marketplace' for projects where you can literally find anything in terms of computer science. Just mention the word GitHub after your google search and you'll most probably end up finding a project under that topic.

Checkout what exactly GitHub is here. A lot of recruiters and software engineers who interview you stress on having a good GitHub profile. It reflects that you do work outside of class and can self learn skills, which is great. Have a look at this profile on GitHub and see how to fork, star, or make a new repository.

The best way to use GitHub is to follow people doing a lot of open source projects, fork repositories you like, and study on different smaller projects posted by people. So if you're learning Machine Learning, then the best way would be to look at smaller projects for regression or classification with smaller datasets on GitHub.

30 days Coding Skill section:

We're currently developing the **skill** section of 30dayscoding.com and hope to finish soon. You'll have free access to the 'self learning series' of the top computer science skills in 2020. It will include a "beginner to intermediate" guide on:

1. Mobile App development - Android, iOS, Flutter etc.
2. Web App development - Basic HTML, CSS, Js, JQuery, Node Js, ExpressReact, Angular etc.
3. Machine Learning - Regression, Classification, Neural Nets, Deep learning, Markov models etc.
4. Data Science - Probability, Distributions, Algorithms, Models etc.

If you need a list of free resources (right now), email us at ***30dayscoding@gmail.com***

Applying for Internships, Jobs

- Apply to companies actively hiring during Covid-19: https://www.levels.fyi/still-hiring/.
- Jumpstart is hiring interns who have lost their internship here https://jumpstart.me/.

We've discussed all ways which you can use to apply for internships, jobs, and co-ops during and after your college.

Company websites

Companies have direct links to the job openings they offer and it's usually under the 'careers' section. So just like Facebook(https://www.facebook.com/careers/) most companies have a designated page for their jobs. Applying directly from their website sends your resume and cover letter through the ATS which then sends the parsed profile to the recruiters.

A lot of companies also use 3rd party services like workday which allow you to apply using your resume and cover letter. The only thing to keep in mind is that **they don't parse your resume correctly sometimes and it's better to check** before proceeding for the next steps.

Make sure to fill in all information properly - projects, work experience, sponsorship questions. (USA only) If you're an international student on an F-1 visa, you need to say that you're legal to work here but will require sponsorship in the future.

Should you **apply to multiple positions at the same company**? It's fine, people do that all the time but usually the process boils down to the same one for bigger companies. If you want to apply for UI/UX and Software Engineer both, then yes, go for it!

Career Fairs

All universities have career fairs once or twice a year which can be very useful to make good connections at companies. Companies send software engineers and/or recruiters to the career fair so it's a good chance to meet them and have a good first impression.

Dress professionally and carry copies of your resume when going to a career fair. Have a conversation, rather than just handing them your resume and saying thank you. They want to know something about you and it's a good chance to make yourself stand out. You can start off this conversation with a brief 1 minute elevator pitch which is usually a much shortened version of the content you have in your cover letter. Start with an introduction, highlight some relevant skills, and then finish with why you are interested/the best for the position.

If you impress the recruiters, they may invite you to interview without having to go through all of the traditional steps.

Along with that, companies also do on campus interviews which allow you to fast forward your process or even get the offer after just one interview in some cases. The interviews are a mix of coding and design questions, some of which are:

1. Design - game of war, tic-tac-toe, minesweeper
2. Array - fizz buzz, subset sum, two-sum, find an element
3. Linked list - remove a node, create a linked list, find the middle element
4. Trees - add elements, balance

Most of the interviews that come from career fairs are white board based so practice before to get yourself comfortable with that style of coding. Some general tips for whiteboard style coding is to write the question, comment your code, use proper spacing, and write neatly. **It's also important to speak out your thoughts** so that the interviewer can tune with you and help you if you're stuck.

Since you don't have access to a compiler, white board interviews will usually be simpler questions, but you should mentally run through the code with some test cases to make sure it works. Additionally, many interviews will allow you to choose a language you are more comfortable in, or write in (near) pseudocode for these styles of interview.

Referrals

Most of the software companies offer you to apply with a referral of their employee. This trick usually works but lately everyone has been sending out cold emails to ask for referrals. However, it definitely makes you a better candidate than others and can get you the first round interview quicker.

If you know someone from any software company, then just send them a message asking for a referral. They can either send you a specific link for referrals or just forward your resume to the hiring manager - both of which will help get you noticed over other candidates. **Developers working at companies often get incentives if someone joins the company from their referral, so there's nothing to feel shy about in this.**

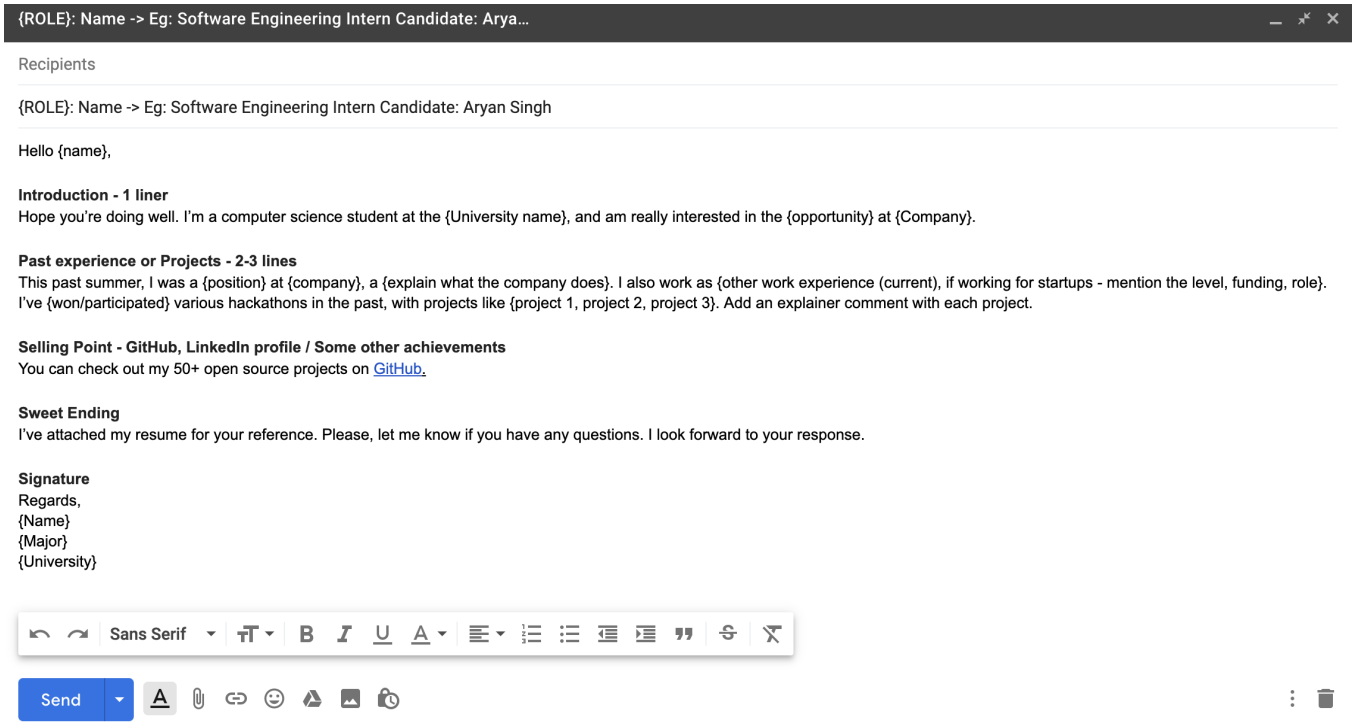
You can also ask for referrals from LinkedIn or emails using the "cold email" approach, which is discussed below. Pick out the companies you're really passionate about and then devote your time to email them properly, rather than emailing literally everyone on your contact list. People are also more likely to give you a referral if they went to the same school as you. You'll be surprised how well this works out and helps you with your job hunt.

Cold Email

Cold email means an email without specifically knowing the person who's receiving it. This might sound absurd, but works a **lot of time**. So it's like sending emails or messages on LinkedIn to your connections at Google. Will they reply? Maybe not, but it's likely that they see the email and forward it to the hiring manager (if they like your profile).

One thing which makes a good cold email/message is a selling point. Imagine yourself as the recruiter or the software engineer at a company and getting hundreds if not thousands of emails from different students. How would you differentiate a normal candidate from a special one?

Here is a template for a perfect cold email:



Sending connection requests on LinkedIn is also a part of networking which is essential for your process. Send a personalized message to someone you don't know so that they can read before accepting the request. It's a pleasant way to introduce yourself and you can do it in the following way:

> *"Hey person, hope you're doing well. It was great talking to you the other day about xyz. I would love to further connect with you on LinkedIn"*

> *"Hey person, hope you're doing well. I loved your 30dayscoding.com website and would love to talk to you regarding some contributions. Thank you"*

Online Platforms

There are a lot of online websites which are famous for jobs, some of which are:

1. LinkedIn (https://www.linkedin.com/jobs/)
2. Monster (https://www.monster.com/)
3. Indeed (https://www.indeed.com/)
4. Jumpstart (https://jumpstart.me/forum/all)
5. Tech internships (https://www.techinternship.io/)
6. CS interns (https://csinterns.com/)
7. AngelList (https://angel.co/jobs)

LinkedIn offers some other advantages than the other sites. You can comment on posts by recruiters, hiring managers, and software developers who are looking to hire for their companies. You can also message recruiters, managers, or other service providers about the opportunity they've listed somewhere.

Recruiters and engineering managers share a lot of posts about hiring for their team and it's good to look out for those. Like the post, put in a small comment, and then message them personally asking about the opportunity. A sample message would look like this:

> *"Hey recruiter, hope you're doing well. I'm really interested in the xyz opportunity you just posted about and would love to talk more about it. I've attached my resume and look forward to your response"*

AngelList is the most popular website for applying to **startups**. If you feel like a growing company might benefit you the most right now, angelList is the way to go. There's risk involved with all kinds of situations but keeping a long term goal will help you rise well.

Coding Interviews

If you make it past the ATS part of the process, some companies may send you a coding test before an interview to try to filter down candidates even further.

The way these typically work is one of three ways: a coding test, live coding interview, or a take home project.

Coding Test

In a coding test, a company will send you a link to a site where you can choose a time to sit down and take the test using their software. The site will monitor your browser usage so be sure to only use the approved resources (usually limited to the Java JDK or C manuals). Once the test begins, a timer will start and you will have between 30 minutes to an hour (sometimes more) to complete and submit every question. Types of questions can be:

- Simple **logic puzzles** like the ones commonly on standardized tests - find the next number in a sequence, find the main idea of a paragraph, etc.
- **Long coding questions** where they will ask you to fully implement a specific function that does something like reverse a string, count duplicates, etc.
- **Short coding questions** where they ask you to find and fix a one line bug in the code

- **Code analysis** where you will need to analyze the runtime of a particular algorithm

You can **choose which language** you want to code in and will usually have an environment where you can **compile/run your code against some test cases**. Thus you will know if your solution to coding questions is correct if it passes all of the test cases. Be sure to watch for common edge cases like empty arrays, negative numbers, etc. You will not probably not need to worry about things like incorrect inputs or other oddities.

Here, it is important to understand how you will be graded in this section. Each company will have **specific cutoffs for what they will accept in terms of correctness and runtime** so be wary of these. More prestigious companies will require near perfect correctness on every problem and will require decently fast runtimes for each of the questions (they will usually print how long it ran for in milliseconds). Smaller companies may let you get away with getting a few of the questions wrong or partially correct.

The coding challenges can be tricky but they usually cover beginner data structures (arrays, linked lists, graphs, etc.), so if you know what a Trie is, no one is asking about those for the first round.

On the 30 Day Challenge part of the website (https://30dayscoding.com/), we've listed 100-110 handpicked questions which cover most of the concepts for coding interviews. **Understanding** these questions is much more important than doing thousands of questions without covering all the topics or memorizing solutions.

You have to put in **time** to succeed and become good at coding. Practicing and studying the concepts are the only two ways your coding will improve. Read all the articles, watch videos, and practice the questions we have for you!

Good strategies for managing these challenges are very similar to general test-taking strategies:

- Skip questions that you get stuck on and return to them later
- If you don't immediately see the answer, try to think up a brute force solution and then see if you can make it faster
- Try converting the data structure to something that might work better for the problem at hand - like a priority queue or hashmap (depends on the problem).

Live Coding Challenge

Ex: Webex, Google doc, Video interview

If your coding challenge is live or in person, you will usually only be given a small amount of problems to do in the span of 20-45 minutes. Most likely they will give you one large problem with some follow up

questions to improve the algorithm. If the coding challenge is done over the computer, you will likely be talking to the interviewer over the phone or some teleconferencing software while you enter code into some shared document for them to see.

As a reminder, **be sure to have all required software installed beforehand** and try to enter the call a **little early** if possible to allow for troubleshooting technical problems.

In a live coding challenge, be sure you are **always explaining what you are thinking** and what you are doing. The interviewer might help you work through some things if they see that you're stuck and will need to understand what you're thinking. It is also a good chance to mentally walk yourself through what you need to do and may impress upon the interviewer that you really know what you're doing. Most companies love to see calmness and confidence in their applicants so talking and explaining is a great way to show that or help yourself with that.

Another important thing to think about is discussing the runtime of your code while you're doing the live coding challenge. Interviewers may ask you to think about the asymptotic complexity of the code or how much memory it will take up. Day 20 covers everything you need to know about Time complexity: Big-O. Read the articles, watch the videos and practice with every new question you solve. If you want extra practice, then email us at ***30dayscoding@gmail.com***.

Lots of questions will have multiple solutions that vary in terms of both the runtime and memory. Companies that are harder to get into (FAANG) will only accept the best solutions so you will want to think about whether or not something can be done faster.

For instance the LeetCode question Two-Sum (https://leetcode.com/problems/two-sum/) has several possible solutions. If we treat it as an array and try to brute force the solution, your code will need to check every combination of two numbers to see if it sums to the target. This is O(n^2). Alternatively, if you use a HashMap, you can just go through the elements in the array and for each entry just check if the HashMap contains the target - value. This is O(n).

Common strategies for reducing runtime are to store previous work (dynamic programming), binary search, sort, use a special data structure, or try to reduce the number of times you need to loop through an array.

> **Important Concept:**
> In the months when you are applying to lots of internships/jobs, be sure to complete as many practice questions on sites like leetcode and hackerrank as you can**. Practice is by far the best strategy to do well in the challenge.**

If you want extra practice apart from the 100-110 questions mentioned, you can visit online websites like leetcode, hackerrank, topcoder, project euler or email us at ***30dayscoding@gmail.com.*** It is very common for people to spend entire days just solving and resolving problems on these websites in preparation for a big interview.

Take Home Projects

The last type of coding challenge is the much longer take home project which is often just a part of the application itself, but it requires the applicant to read through the project guidelines and develop a much larger solution to something. The task here is usually to write up some sort of documentation, create a full stack application based on their specifications, or other longer projects.

Here is your chance to impress the company with more specific skills. Carefully read through the project requirements and make sure that you fulfill every detail. Be sure to document all of your work and use popular frameworks if you know them. At the interview stage you will likely talk about what you did for this project so it will be important to do your best and comment/document everything you do - especially if the interview is weeks later.

Regardless of what style of challenge is given, a great strategy to being prepared for these interviews is to do some research about the company. For more well known companies, previous applicants often post their experience in the interview on glassdoor, leetcode, jumpstart, etc. This can be useful to figure out how many stages of the interview process you can expect and what will be required at each step. Sometimes these reviews even contain common questions that the companies ask which can give you a general idea about the difficulty and what is expected of you.

An example of this is Braintree (a division of Paypal which acquired Venmo) which sends the interviewees an API style take home coding challenge. It consists of you making some API calls to apply, then receive a coding challenge, then solve it to write some code. They extend this to multiple rounds when they call you for onsite.

If you want more information about a specific company, email us at ***30daycoding@gmail.com*** with the company name and we'll get back to you!

Behavioral Interview

After passing the coding challenge (or if you skip it altogether), you will move on to the actual interview. It is very possible that this will be the first time you will interact with a human in the entire application process. The coding challenge is usually graded by bots so the interviewer is possibly only reading your application and coding work during the interview itself.

Important Concept:
Practice your interviewing skills in the mirror or with a friend. Have them ask you common interview questions and give genuine responses. These responses can be practiced and constructed however you like. **The objective is to make sure you are giving clear and powerful responses.**

The behavioral interview may be in person or may be over the phone or video chat, but either format is equally important to impress the recruiters. At this point in the process there are very few candidates remaining so you will need to show how you stand out from the others.

In the interview it is often helpful to remember to use the S.T.A.R method for explaining past experience. Another good strategy is to turn your experience into more of a story since people usually enjoy listening to a good story. Give relevant background and a sense of flow to whatever you want to say and you will find yourself having a far more engaging conversation with your interviewer.

Also be sure to study up about the company you are interviewing with. Lots of interviews start with the interviewer asking what you know about the company so this is your chance to give a good summary and tell them about the things you are interested in. They don't necessarily need you to explain the company to them, but just show that you know what it is.

Some companies even have specific company wide core values or principles so it can sometimes help to have a general idea remembered or written down (if it's a phone interview) to reference those. The core values are often what interviewers are looking for in their applicants - things like ability to learn, motivation, adaptability, etc. Pick stories, projects, or past experience that demonstrate the important things on these lists.

It is becoming increasingly more popular for companies to focus less on coding ability and more on learning ability and motivation. Large companies are hiring people that they think are smart and capable of learning new things quickly and easily over people who have more experience, but cannot demonstrate ability to learn.

Company fit is also becoming more important as recruiters also think about whether or not a candidate will fit into the company culture and whether or not a particular candidate is someone that they want to work with. Amazon has its "Leadership Principles", Google has its "10 Things" - both of which are very important for interviews and you should be ready with all the answers beforehand.

These two resources contain a list of the more common behavioral interview questions that companies may ask:

- The Muse interview guide

- Inc most common behavior questions

Other common talking points during interviews will be to discuss parts of your resume - particularly past work experience. The interviewer may ask clarifying questions about some points on your resume like giving a summary about what you did at a past job, explaining how much you know of a particular skill, or to explain a project that you listed.

Checkout this **Behavior Interview Preparation guide,** make a copy of it and put in 3 projects for each heading. Practice before every interview and say it out loud a few times to be comfortable with it.

One way Interview

In the modern world, we also have to deal with a totally virtual interview experience. The One Way Interview is a relatively new creation where the applicant seems to have no interaction with a human interviewer. While it is rare in CS fields at the moment, thousands of companies use this method so it will likely become more popular over the years (https://www.hirevue.com/customers**).**

Using such technology a hiring manager can send out a series of interview questions to every candidate who will then record and upload their responses so that the HR team can analyze them and select their favorites or the ones that best fit into the company. They save time by not having to meet and interact with candidates as all they really have to do is watch and rate their videos.

From the applicant's end, you have to install software, create an account, or follow a link to the interviewing website. From here you will start the session and a series of questions will come up one at a time. The applicant starts the recording for each question, talks for as long as they need (or if there is a timer - however long they are allowed), and then clicks stop and submits. At the end of all the questions, you just submit and it is sent off to the HR department.

Most of the strategies to finding success with these interviews are similar to in person interviews, but there are of course some changes. Given the format, you will need to pay special attention to appearance, surroundings, and your behavior during the interview.

Lots of these systems have an AI to help the HR teams screen out candidates by their answers to interview questions. The AI processes the videos and analyzes features like how much a candidate smiles, whether they make eye contact, look friendly, give answers that contain relevant keywords to the job posting, and more. The HR team can then choose to focus more on the people who scored higher than the others in these fields and will then specifically watch their videos to narrow it down. This means that once again you must tailor your performance to pass both the robot and the human viewer. Luckily, the strategies can be combined to make some helpful tips:

- Be conscious of your surroundings:
    - Inform roommates or people nearby that you will be conducting an interview and politely ask them to be quiet
    - Silence devices or anything that might make noise like an air conditioner
    - Put your phone away unless you are using it to record
    - Temporarily move kids or pets to another room so they do not bother you
- Use proper video framing - center yourself in the camera and give a little bit of room above your head
- Make sure there is a good light source so that you are clearly visible, but not too bright
- Make use of the time before a question to think about how you want to answer it before you start the recording (if possible)
- Maintain good eye contact with the camera, smile, and make good hand gestures when appropriate
- Use plenty of keywords in your answers and refer to the position you're applying for
- Be wary of how you begin and end your answers.
    - Avoid awkward endings of fumbling for the stop button
    - Avoid using phrases like "umm", "so yeah", "yeah that's about it", etc.
    - Try rephrasing the question
- Do the practice round if they offer it - it can help to make you more comfortable for the real questions
- Practice well before the interview and try to record yourself and play it back to see how you do. Share it with a friend and see if they have any advice.
- For the real thing be sure to dress formally, brush/comb your hair, etc.
- Use a notepad if you are given prep time before the questions so you can write down what you want to talk about
- If using a mobile device to record, avoid shaking the device too much. Preferably use a tripod or rest the device against something (while maintaining proper framing and lighting).

These two videos offer very good advice about One Way Interviews and are very well made:

- https://www.youtube.com/watch?v=J2VnJOw5Cd0
- https://www.youtube.com/watch?v=jn0dc1cOctA

With good performance in the recording and a solid resume and talking points, you should easily make it in front of a real recruiter who will evaluate if you are a good fit for the company and you may either immediately get an offer or will be invited to do another round of in person interviews.

Follow Up

After the interview, the company likely has a few more candidates to go through before they reach a point where they need to choose the candidate they will extend an offer to.

Following up after an interview is very important and shows a lot of intent from your side and can ensure that you are the candidate they will end up choosing. There are a couple of different ways but the underlying principles are the same, where you thank them and show interest in the opportunity.

Email, LinkedIn

Sending a follow up email 12-24 hours after the interview can be helpful in getting the interviewer to like you and may help to convince them to hire you. It is generally seen as a sign of politeness and will bring your name and interview fresh into their memory.

It can be a simple message in accordance to your position. Some key things to remember are - mention the people you talked to (if you talked to different teams), mention the things you liked about the talk, mention something you would love to work on (if you don't know that right now). For instance, you could say: "testing is one thing which is really important and I'm going to read more and start integrating it in my projects from now".

Another good strategy is to use this email as a chance to add something you might have forgotten to talk about during the interview or one or two things you feel were not emphasized enough - almost like a second cover letter (but not as long - you don't want to go overboard).

Here's the general procedure for such an email:

1. Greet them and thank them for their time in interviewing you. Be sure to personalize each email and refer to the interviewer by name. Send a separate email for each interviewer.
2. Talk about some specifics from the interview
3. Add some information that might reinforce that you are the best candidate and are super interested in the position
4. Close it out with some contact information so they can follow up if they have any questions

Here's a template for one:

> *"Hey person, hope you're doing well. It was great talking to you and/or your team today/yesterday and I got a lot of learning points from it. It was great to talk to you about ___ and I believe I would be a strong fit for the position because of my experience in ____. I have attached some links to my GitHub repositories where I have some projects that show ___ . Please*

feel free to reach out to me if you need any more information. Looking forward to hearing back from you, thank you!"

Getting an Offer

If everything has gone well, you might receive some offers from companies that are interested in hiring you for a job/internship! Congratulations!

Usually, this step comes in 2 parts with an informal acknowledgement first from the company or hiring manager congratulating you and saying they are deciding to move forward with you, and then a later follow up where they would send you documents to read and an actual contract to sign.

Evaluating an Offer

If you receive multiple offers or simply need to know if it is a good choice to accept, it is critical to evaluate the differences between offers. By far the best resources for this process are:

- Glassdoor: https://www.glassdoor.com/index.htm
- Levels.fyi: https://www.levels.fyi/

Both of these sites contain self-reported reviews, salaries, and other information about jobs and job offers so that you as an applicant can see if you are getting a reasonable offer for your skill set. Financial points to consider about an offer are:

- **Location** – Cost of living in California is much higher than in some other states so the salaries there will be larger
- **Education/Degrees** – Typically positions requiring a masters, and candidates that possess a master's degree should pay/be paid more
- **Experience** – If you have a lot of experience in a particular field, you have more value to the employer and the salary should reflect this higher value as it means you will be able to better perform your duties
- **Skills** – Jobs that require more advanced skills, and candidates that have more advanced skills should see a higher salary as these qualifications are usually rarer and more demanding
- **Company** – Companies in lucrative industries and large companies are usually expected to have higher salaries as they are typically more competitive and more demanding. While this is not always the case, you can certainly expect a company like Google, Facebook, etc. to have higher pay than some other companies.

Do your research on what candidates with your skills or other employees of the company should be making and compare it to the offer you have received. As a general number, recent university graduates with a bachelor's degree in computer science should expect to be making somewhere between $80k-120k yearly depending on location. For those getting an internship while pursuing their bachelor's degree, salaries are often compared in the hourly wage and should be somewhere in the range of $20-

50 per hour depending on the above factors. For those who already have a bachelor's degree and are pursuing an advanced degree, internships should pay more and should be approximately equal on a month-by-month basis to what you would be making in a month in a full-time job. For example, someone with the qualifications to be getting a $120k full time position should be getting approximately $10k/month or $62.5/hr

Negotiating

If still you feel you should be earning more, do not hesitate to try negotiating for a better offer. This is the point in the interview process where you have the most power to persuade the company to increase the offer sometimes as much as 10% higher than what they initially offered. Reach out to the source that sent you the offer and express your gratitude for the offer and then describe exactly why you are looking for a higher offer – whether it be your experience, skills, or other factors from above. At worst they will say that the current offer is the highest they are willing to go.

Conclusion

Thank you so much for reading this guide and we hope this helps you land your next internship or job. Next things which we've decided to bring out are:

1. Ace the Coding challenge - usually the first round interview
2. Beginner to Intermediate guide for Mobile development, Web development, Machine learning, and Data Science

Best of luck with the interview. You can do it!

Regards,
30dayscoding.com
Aryan, Yefim

Useful Links

Coding Interview:

- Afternerd
- Freecodecamp
- Interview cake
- John Washam, GitHub
- Towards data science

Behavior interview:

- Celarity
- Simple Programmer
- Huffpost
- 3 key tips
- Vault blog

Resume resources:

- Leetcode Patterns
- Resume checklist
- The S.T.A.R method
- Cover Letter tips
- Glassdoor
- Balance Careers
- The Muse
- Glassdoor

Sample Computer Science Resumes

- UMass CICS