Pathrise PRESS

< HOW TO LAND A SOFTWARE ENGINEERING JOB/>

Intro

What is this Guide?

This guide is free resource to help students best approach their job search. We share many of the best practices we've learned and developed from working with the hundreds of students that we've helped land their dream jobs.

Contents

What is Pathrise?

Pathrise invests in university students or young professionals by providing personalized advice and training on how to get the best possible internship or job. The program is completely free upfront. Pathrise fellows pay zero tuition until they land an offer they are happy with and get hired. Instead of focusing on a technical education, Pathrise is entirely about optimizing your job search. This involves services like resume review, prospecting, referrals, interview preparation, and negotiation advice. We track every data point so we can hold ourselves accountable to actually produce significant and measurable value for our students.

Part 1: Finding opportunities

How to Find Opportunities

If you have a 5% chance of getting an offer from any opportunity, that might seem like it sucks, but that means that 50 such opportunities will give you a 90%+ chance of getting an offer assuming these outcomes are independent. Part of your success will be dependant on the number of opportunities you have available, and if you don't get enough, it will be difficult to succeed.

Referrals

Referred candidates are nearly 15x more likely to get a position and nearly 80% of recruiters noted referrals as their best way of sourcing hires.

Tactics for finding people to refer you to opportunites include:

- Ask fellow students from the student organizations you happen to be in if they can submit a referral for you to the companies that they worked at last summer. Even if there isn't a formal referral system, you can have your friends send an email to their recruiter from last summer introducing you, and that will usually work.
- Parse your LinkedIn network and even consider connecting with employees at the company you're targeting, even if you didn't know them beforehand. If you connect and ask politely, many engineers in the industry are looking to give back, willing to answer your questions, and would consider you for a referral even if you reach out cold.
- Utilize an alumni database provided by career services to find alumni at the companies you are interested in. Alumni are generally great warm leads and very open to helping students from their alma mater.

It's definitely possible that you won't be able to get a referral everywhere, and so there will have to be some positions where you apply on your own. There are also a few ways to optimize that.

Career Fairs

Career fairs can be a really great opportunity to network and connect with employers at your university, but don't just stand in random lines. Instead, do some research before and review the list ofemployers that will be attending. Ideally you have a "why you are a good fit" for the high priority companies which you add to the end of your elevator pitch.

Go to the career fair prepared:

1. Wear clothes that follow the career center dress code (If there is no dress code, wear clothes that make you feel confident)
2. Come with multiple copies of your resume (40 is a good number)
3. If you want to share an app you made, have it ready on your phone or ipad

Have your elevator pitch ready:
We talk about how to construct an elevator pitch later in this guide in the interviewing section. You usually want to do an abbreviated version of your go-to interview elevator pitch for career fairs (so 30 seconds to 1 minute rather than 1 to 2 minutes).

At the career fair, try to make sure you talk to as many companies as possible, and follow up over email afterwards as quickly as possible. Ask for the email of the person you chatted with and follow up - this is a very effective way of getting into an interview process.

Online

The best places to find jobs are your ABC's. These are job **A**ggregators, job **B**oards, and job **C**ollections.

Aggregators are usually a combination of a job board and an automatic scraper that collects jobs from elsewhere. They're the most convenient places to look for jobs where you can expect to find 5-10 viable opportunities in just a few minutes.

Boards are places where companies pay or sign up to list specific jobs. They're great places to look for positions that you might not find elsewhere, but you'll find jobs at a slower rate than when using Aggregators.

Collections are extremely effective tools that essentially are manually curated lists of available jobs. Though sometimes they aren't always properly updated, for the most part their a great way to "browse" through 100s of individual company career pages at once.

Here are a few of the best examples of each of the categories:

https://www.google.com/ | *Aggregator*
Surprisingly good search tool, try to limit to only more recent jobs like posted in the past week

https://www.glassdoor.com/index.htm | *Board*
Also try to look under more unconventional locations if you can, different listings than Indeed

https://www.linkedin.com/jobs/ | *Board*
There are plenty of good jobs listed here

https://www.wayup.com/s/ | *Board*
University-specific jobs so that's great, but use the search function first

https://www.joinhandshake.com/ | *Board*
University-specific jobs and exclusive to each university. If your university is using handshake, this is one of the highest response services there is.

http://www.intern.supply/ | *Collection*
For interns only but really high quality. Isn't always the most up to date.

https://github.com/j-delaney/easy-application | *Collection*
Mostly only full-time positions, but pretty comprehensive list

Here are some other sample sites that are more for specific purposes or more general, but worth mentioning:

https://www.indeed.com/ | *Aggregator*, https://www.simplyhired.com/ | *Aggregator*, https://www.ziprecruiter.com/ | *Aggregator*, https://news.ycombinator.com/jobs | *Board*, https://angel.co/jobs | Board, http://jobs.gamasutra.com/ | *Board*, https://www.monster.com/ | *Aggregator*, https://jobs.github.com | *Board*, https://stackoverflow.com/jobs | *Board*, https://www.producthunt.com/jobs | *Board*, https://whoishiring.io | *Aggregator*

Ways to Look

Another tool that's not used often enough are X-ray searches. An X-ray search is something that you can type into a Google search bar to find new opportunities. This way you don't have to depend on your ABCs in order to find positions for yourself. Try typing in the following search string into Google:

site:lever.co "software engineer" "new grad"

You've basically created your own personal list of jobs. It should look something like below:

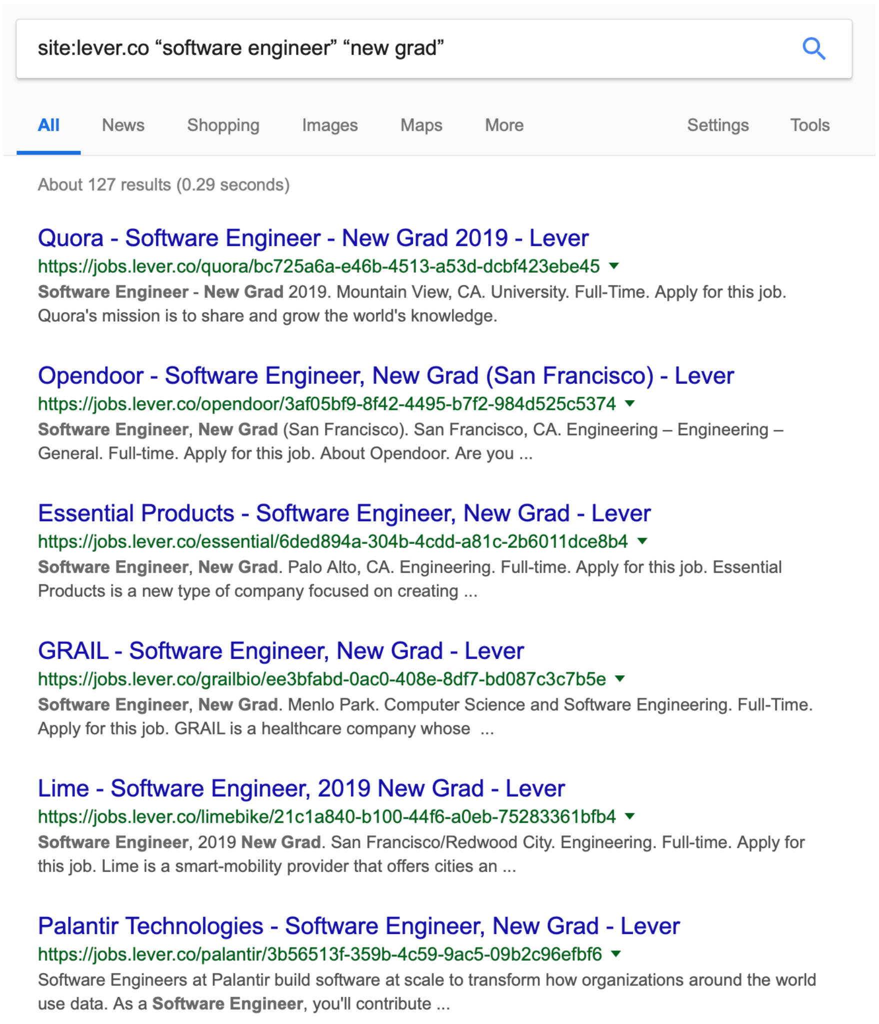
You can switch out terms as you see fit. The general anatomy of an X-ray search is this:

site:[insert the site you want to search] [keywords you want to search]

So other example x-ray searches you might want to try are:

site:lever.co "associate software engineer"
site:workable.com (software engineer AND "san francisco")
site:greenhouse.io ("software engineering internship" OR "software engineering intern")

These searches are very customizable and are good way of creating basically your own job aggregator or search engine, and finding positions that are more hidden away from to other candidates.



About 127 results (0.29 seconds)

Quora - Software Engineer - New Grad 2019 - Lever
https://jobs.lever.co/quora/bc725a6a-e46b-4513-a53d-dcbf423ebe45
Software Engineer - **New Grad** 2019. Mountain View, CA. University. Full-Time. Apply for this job. Quora's mission is to share and grow the world's knowledge.

Opendoor - Software Engineer, New Grad (San Francisco) - Lever
https://jobs.lever.co/opendoor/3af05bf9-8f42-4495-b7f2-984d525c5374
Software Engineer, **New Grad** (San Francisco). San Francisco, CA. Engineering – Engineering – General. Full-time. Apply for this job. About Opendoor. Are you ...

Essential Products - Software Engineer, New Grad - Lever
https://jobs.lever.co/essential/6ded894a-304b-4cdd-a81c-2b6011dce8b4
Software Engineer, **New Grad**. Palo Alto, CA. Engineering. Full-time. Apply for this job. Essential Products is a new type of company focused on creating ...

GRAIL - Software Engineer, New Grad - Lever
https://jobs.lever.co/grailbio/ee3bfabd-0ac0-408e-8df7-bd087c3c7b5e
Software Engineer, **New Grad**. Menlo Park. Computer Science and Software Engineering. Full-Time. Apply for this job. GRAIL is a healthcare company whose ...

Lime - Software Engineer, 2019 New Grad - Lever
https://jobs.lever.co/limebike/21c1a840-b100-44f6-a0eb-75283361bfb4
Software Engineer, 2019 **New Grad**. San Francisco/Redwood City. Engineering. Full-time. Apply for this job. Lime is a smart-mobility provider that offers cities an ...

Palantir Technologies - Software Engineer, New Grad - Lever
https://jobs.lever.co/palantir/3b56513f-359b-4c59-9ac5-09b2c96efbf6
Software Engineers at Palantir build software at scale to transform how organizations around the world use data. As a **Software Engineer**, you'll contribute ...

Part 2: Optimize your Application

Resume

How do Recruiters Read Resumes?

Recruiters will spend no longer than 30 seconds on a single resume, and will routinely spend even less than 10 seconds reviewing most of them. This means your resume has to be easily skimmable and impressive at a first glance, and this is a guide on how to optimize for those qualities.

General Qualities of a Good Resume

Here are some of the general qualities of a good resume. Most of them are pretty self-explanatory, so we'll just leave them here with a short descriptor:

- **Pick a narrative** - Will help you decide what to keep.
- **Use keywords** - Use them in context if possible
- **Make every word count** - Don't waste space on unhelpful words; avoid passive voice
- **Lose the objective** - Oftentimes this doesn't add much information to a resume, and just wastes space
- **Put the most impressive things firs**t - Maintain reverse chronological order for the most part
- **No typos** - Correctness is important for professionalism
- **Showcase projects** - Pick 2 to 4 only, 5 is pushing it but possible. Side projects outside of class are especially interesting.
- **Add contact information** - Name, number, email, website (optional)
- **Avoid too many colors and shades** - Most of the time, just do black with different font weights, even if you are using color, 1 is more than enough
- **Make sure text is selectable** - ATS's won't be able to process it otherwise
- **Margins** - Generally 0.5" - 0.7"
- **GPA** - GPA under 3.5 is doubtful to list, GPA under 3.0 is definitely don't list
- **Spacing & Distinction** - Make sure your section headers, your item titles, and your item subtitles are visually distinct using different font styles. Also keep spacing as consistent as possible.

The 2 Biggest Mistakes on Most Resumes

Almost every resume we review makes the same two mistakes or has the same two points where they need to be optimized. We go into detail on what these mistakes are and how to fix them below.

Mistake 1 - Grunt vs. Impact

Grunt is our internal reference for resume points that are focused only on what you worked on and what you were assigned to do. You're essentially describing the grunt work that you did, and it's usually never a good description of how you spent your time in any past experience or project. Grunt statements usually look something like 'Developed X for Y' or 'Worked on X using Y'. Grunt statements are sometimes a necessary evil, but for the most part should be avoided. Instead…

Impact statements are statements that focus on what you accomplished and what your results were. They normally follow a structure that's more like 'Accomplished X by implementing Y which led to Z' or 'Developed X to accomplish Y and Z happened'. They might sometimes be a little bit longer, but being able to show some reading your resume why what you did actually mattered is definitely worth it.

Mistake 2 - Lack of Quantification

Students often struggle to find numbers because they feel like their projects need to be launched and successful to find numbers, but this couldn't be further from the truth. There are so many statistics that you might be able to list about your experience and projects and how they went. It's just a matter of knowing where to look.

In order to find numbers where there may seem to be none, you should ask yourself questions like:

What was the scale of this problem?

- How large was my dataset or many rows of data did I analyze?
- How many devices did I serve?
- How many scenarios/permutations/tests did I consider/handle?
- How many different methodologies did I implement?
- How much more time did I invest than is normal?
- How many people did I manage or how many teams did I act as a liaison for?

What did I achieve as a result?

- How many users did I launch to or will I launch to?
- How many users/groups used it?
- How much money did I produce in value?
- How many many hours did I save the company?
- What percentage did I improve our old process?
- What percentage of our old process did I replace?

Generally, you will find that coming up with these questions is more of a matter of understanding the general gist of where to look rather than some magic sauce. Hopefully the above examples give you a better idea of how to get started looking to quantify your resume points.

Below, we've included some examples of mistakes of both grunt instead of impact and lack of quantification, and how students applied the concepts above to fix them:

Before	After
Created a larger social media presence by blogging on the company website and writing ads that promoted housing options.	Spearheaded social media and advertising efforts, creating leads that contributed to selling 30+ properties worth $20M+ in total.
Generated KPI's (number of blockers, average resolution time…) on the integrated data through a Java web application using Spring/Hibernate.	Developed a Java web-application using Spring/Hibernate that reduced JIRA task resolution time by over 25%, meaning that overall company efficiency increased by 25%.
Originated concept for social network app for campus organizations.	Built a social media platform for campus organizations, successfully launching to ~15 clubs and aiming to have 200+ clubs by Spring 2018.
Aided undergraduate students with understanding of code, homework, project, and topics (including data structures, algorithms, and Java Swing).	Aided undergraduate students with understanding of code, homework, project, and topics (including data structures, algorithms, and Java Swing).

See an example of a good resume on the following page (8).

SAMMY SPARTAN

San Jose, California | linkedin.com/in/sammy-spartan | sjsparty@gmail.com | 555-555-555

EDUCATION

San Jose State, San Jose 2018
Bachelor of Science Computer Science: GPA 3.70

TECHNICAL SKILLS

Mobile and Web: Swift, Ruby, Javascript, C++, HTML, CSS
Data: Python, R, SQL, VBA, Stats, Advanced Analytics & ML
Computer Science Coursework: Data Structures & Algorithms, Computer Security

PROJECTS

Split (Bill-Splitting) App: iOS, Swift, Google Cloud Vision ML API, Firebase, Alamofire
Built app using Optical Character Recognition (OCR) to scan a receipt, parse text to identify meal items and prices, then allows user to assign items to guests and send Venmo requests – enabling one person to put the bill on one card.
Seed Microlending Web App: Ruby on Rails, jQuery, Postgres, Venmo API, Devise Authentication
Platform that enables users to act as a 'lender' or 'borrower' within a network of their trusted friends to enable easy facilitation of informal microloans via Venmo.
NBA Sleep-Based ML Prediction Model: R, Machine Learning, Neural Net, Logistic Regression, KNN, SVM
Inspired by an article related to impact of sleep on athlete performance, built a program to analyze historical NBA schedule data, calculate delta, and flag games with large mismatch between rested and rest-deprived teams. Utilizing NeuralNet and Logistic Regression models, backtested with strictly rest/travel-based variables – achieved 76.4% prediction accuracy.

EXPERIENCE

Appstrax, Mobile Developer, Cape Town, SA | 2017-present

- Led development of iOS "Uber for Laundry" app, including: modeling database architecture, scaffolding user flow, integrating API endpoints with front-end, authentication, integrating Firebase storage, and QA
- Implemented scrum methodology and produced client-facing demos after each sprint, inviting feedback with each production iteration

SpaceX, Summer Intern, Los Angeles, CA | Summer 2016

- Created model to dynamically generate optimal rollout strategy for new broadband satellite internet markets based on aggregated geographic data on speeds, prices, depth of competition, and regulatory landscape; findings presented to SpaceX CEO
- Designed granular cost forecasting methodology for 2nd largest opex category; utilized Excel/VBA to build models which resulted in 43% lower variance of forecast vs. actuals

Delta Air Lines Innovation Lab, PM Intern, Atlanta, GA | Summer 2015

- Conducted talent analysis and needs assessment around innovation skill sets necessary to facilitate a more entrepreneurial Delta culture; findings presented to c-level executives
- Devised strategy based on aforementioned research which resulted in opening a 7200 sqft innovation lab, and creating a recruiting manifesto to strengthen talent pipeline

ACTIVITIES

- Spent past 10 months doing remote software development and traveling to 6 continents, 20+ countries
- Worked with 50+ companies for MHacks, one of the largest hackathons in the US, raised over $300K in sponsorships
- Surfing, rock climbing, skiing, hiking, scuba diving, biking and travelling

Reverse Recruiting

One of the best ways to add on to your application and make it more effective is to send an email along with it. This will get recruiters to look at your resume a second time, and increase your response rate significantly. This is what we like to call "reverse recruiting" because instead of waiting for the recruiter to reach out to you, you reach out to them first. According to our data, sending an email along with your applications will increase the likelihood of a response by an average of 3x, which is a huge benefit, especially since sending an email can take less than 5 minutes per application.

For every application that you do, search for someone on the engineering or hiring team, and send them an email. There are many ways that you can find someone's contact information. On one hand, you can see if their contact information is available via an email-finding tool like the ones below:

https://connect.clearbit.com/
https://www.circleback.com/contactcloud/
https://hunter.io

Alternatively, you can find somebody on LinkedIn and try to guess their emails. The most common patterns are:

first@company
firstlast@company
flast@company
first.last@company
f.last@company
first.l@company

After guessing and before sending, you can use verifiers or automatic email guessers like the following sites to verify if your email is correct or not, so that it won't bounce when you send it:

Hunter (https://hunter.io/email-verifier)
LeadFinder (https://www.leadfinder.pro/)
Email Checker (https://email-checker.net/)

If the company is small enough (under size 100), you can even reach out to the CEO or CTO, and sometimes they'll reply, and sometimes they'll even interview you themselves. That's the sort of attention that you can receive if you ask for it in the right way.

But what is the right way of asking for it? A good cold email has the following 5 features:

1. **Concise** - Short and to the point
2. **Compelling** -Mentions something about you and your interest in the company that is convincing for the person reading it to respond. Does it excite them?
3. **Personalized** - Doesn't seem overly templated and seems like you wrote something unique for the particular company
4. **Friendly** - Comes across as friendly and casual. Usually you can be more relaxed in the technology industry than in some other fields like finance.
5. **Correct** - Avoid typos, misspellings, and grammatical errors.

Below is an example of a high quality outreach email that illustrates all of the above principles from Sammy who is passionate about joining Discord as a software engineer.

Hi Gabby,

I'm Sammy, a senior at San Jose State University majoring in Computer Science and Business graduating at the end of 2018, and I'd love to learn about how I could contribute at Discord.

I've been using Discord for more than 3-4 years; from day one I was advocating for all my friends to quit Skype. Surprise! It happened. Seeing how much Discord has improved and how many amazing features have been added over the years has made me super proud. Since then I've always been interested in developing awesome things at Discord myself.

I know you're super busy, but it would be awesome to be able to hop on a call with you to chat about what software engineering roles you and the Discord team would be potentially hiring for in the near future. Would you be free for a 15-minute call on Monday either at 1:00 pm or 3:00 pm (PST)?

Just in case, I've attached my resume to the e-mail. I appreciate your time and consideration!

Regards,
Sammy

There are many ways that you can adapt the cold emailing strategy above to do even more interesting tactics.

For example, if you want to switch up the platform, you can use the following features above to write a strong Twitter message to a software engineer or engineering manager you admire from a target company and see if they'll be interested in you as a candidate or in an initial chat.

Another example, if you want to switch up the purpose, you can use the following tactics above to email someone before you apply to a position, so that you can set up a conversation with them to learn more from their career and receive their mentorship. Afterwards you can follow up with politely asking them for a referral if they're comfortable with it, and get your foot in the door at a company that way.

With creativity and the building blocks above, the variety of tactics that you can apply with reverse recruiting are endless!

Part 3: Nailing your Interviews

Technical Interviews

Interviewers are going to ask you questions about fundamentals like algorithms and data structures. They want to test the tools in your toolbox. With stronger basic skills, they'll know that you'll be prepared to learn any new framework or technology.

Along these lines, there are a few keys to understanding how you want to be prepare for technical interviews:

1. The first thing to do is to master a single language. Most top companies want to know if you're good at coding but don't care as much if it's in one language or another. While you don't want to choose a language that makes your life harder like C, and generally want to tend to pick languages with some functions pre-built for convenience like Python and Java, at the end of the day, it's up to you.
2. Next, you want to make sure you are good at debugging. You will need to do it plenty of times throughout an interview.
3. Finally, you need to get comfortable with thinking out loud and vocalizing your thoughts. It makes a huge different if your interviewer can actually follow along and an interview without a solution but with thinking out loud is much better than an interview with a solution in complete silence.

Remember that almost all of your questions will require understanding of one of the following:

- Hash tables
- Linked lists
- Breadth-first search
- depth-first search
- Quicksort
- merge sort
- Binary search
- 2D arrays
- Dynamic arrays
- Binary search trees
- Dynamic programming
- Big-O analysis

So study them hard and commit each of these fundamentals to memory.

How to be most effective when prepping for technical interviews alone

Almost every resume we review makes the same two mistakes or has the same two points where they need to be optimized. We go into detail on what these mistakes are and how to fix them below.

Use Pencil/Pen and Paper (Landscape orientation)
This simulates the whiteboard environment the best. If you happen to have a whiteboard, use that instead! If you are rusty on syntax for your chosen language or are using some new semantics that you roughly recall but don't know the exact details, we recommend the following:

- While actually coding the question, just attempt to write the best pseudocode possible for the portions of code where you forgot the exact semantics. For example, if I forgot how to initialize an integer array in java with values I would just write:

```
int pseudoCodeJavaArray = [1,2,3,4,5]
```

- Once you're ready to execute the code in a programming environment, it's fine to look up the right semantics so your code can compile/run. We recommend you have another document where you record all the semantics you forgot and what's the best/correct way to write them. So continuing with the example above you would note down:

```
# Initialize an int array
int[] myIntArray = {1,2,3};
```

Stay completely focused for 90 minutes
Block of 90 minute times on your calendar where you do nothing else but attempt the question (~20-45 minutes) and then review the answer afterward (~30-45 minutes). Remember, we're trying to simulate the interview environment/pressure as much as possible. You wouldn't revert to Facebook if you're having a brain fart in a middle of a real interview right?

Consider this a time limit for completing the question as well, and even try to finish most questions within 45 minutes at most. This will help you simulate the actual speed required in a real technical interview. You can use the latter 45 minutes for review like mentioned above.

Speak out loud for the parts that you would be speaking in a real interview. Some guidelines are:

Problem solving stage

- Talk through the solutions your thinking of. Discuss tradeoffs between solutions if you have many approaches.
- Always state the runtime and space complexity of each solution you come up with
- If you're struggling to come up with a more optimal solution, talk out loud about your thought process.

Coding stage

- It's not necessary to talk about every line of code your writing. Interviewers know it's hard to talk and code. A good balance is to let the interviewer know when you're doing a logical chunk of work. For example: "I'm going to create two for loops that iterate over the array and sum the unique pairs." However, do whatever you feel most comfortable with here.
- It's also totally fine not to talk at all as long as you're making progress on the code.
- If you are struggling for more than 2 minutes to come up with the code to solve particular state you're trying to make, talk it through out loud. A common question interviewers will ask is:
  - "What are you struggling with?"
  - "Can you tell me what your thinking about?"

Verification stage

- Tell the interviewer about which edge cases you're going to test
- When you catch a bug, explain to the interviewer what the bug is and how you going to fix/what you fixed.

You can even get novel with this! One candidate told us he talks to a basketball and uses that as a prop for the interviewer.

When do I start coding?

- You should spend at most 10-15 minutes problem-solving. Then, with the most optimal solution you have thought of, attempt to code that one. If you have two solutions that are almost equally optimal, code the solution you feel more uncomfortable with.
- By all means, if you already are confident you know how to code the solution (i.e. the naive solution), there's no need to waste time practicing what you already know. Spend more time problem-solving!

What happens after I'm done coding?

- Once you come up with a solution that you mentally verified is correct, don't just look up the answer!
- Type the code into an environment where you can compile/run it. Make sure it works in all the reasonable edge cases.
- When you catch a bug, explain to the interviewer what the bug is and how you going to fix/what you fixed.

Behavioral Interviews

The first interviews in many processes will often be behavioral and we'll detail how to nail those below.

Answer Structures and Guidelines

When in a high stress situation it can be difficult to effectively organize your thoughts. That is why having a couple of answer frameworks memorized and ready to go is critically important. There are dozens of these frameworks out there for behavioral questions, but we have found that the following have been the most useful especially in software engineering interviews.

Elevator Pitch

The elevator pitch is a classic pitch format named after a hypothetical situation where you find yourself riding the elevator with the CEO or an investor and need to quickly pitch them an idea. Your personal elevator pitch should sell the most attractive parts of you as an employee. This pitch should be ready to go at a moments notice and briefly summarize the most impressive points about your education, experience, projects, and end with a summary of yourself as a candidate as well as a preview of why you want to work at company x. The structure looks something like:

Education: Introduce yourself, your major, and your class or year of graduation, which is really important for the recruiter to understand so they know what type of internship or new grad position you are looking for.

Experience: Talk about the past work that you've done in previous internships or even student organizations and activities. Show that you are not just somebody who learns coding in the classroom and that you have done outside activities.

Projects (Optional): If you would like or if you don't have much experience, supplement your elevator pitch be mentioning 1 or 2 of your most interesting projects.

Conclusion: Make sure to end strong. Don't just trail off when describing yourself. End with a brief reason why you are interested in the company you are pitching or end with a summary of your strongest skills and high level background.

Generally, you will want elevator pitches to take 1 to 3 minutes in an interview, and 30 seconds to 1 minute in a career fair.

Below is an example of a high quality elevator pitch that illustrates all of the above principles.

Google Recruiter: Could you tell me a little bit more about yourself?

Student: Yea! Of course - happy to share. I'm a graduating senior at Pathrise University majoring in Computer Science.

During my sophomore year, I worked as a software engineering intern at a local startup called Startup X. It was a great experience, mainly because I was given a lot more responsibility than is normal for an intern - I worked alongside the founders and pushed code to production nearly every day.

Last summer, I worked as an intern at Company X on the payments team. While there, I helped build features around the payments system that helped Company X process tens of thousands of transactions, mainly improving the speed and reducing error rate of the overall process.

I like to work on a few projects outside of school as well. For example, the last mobile app that I made helps users recognize if an object is a banana or not.

Anyways, I'm really excited about who you all are doing at Google. I've admired Google products for a while now, but I think the overall impact that Google engineers can make on how we catalog and make information available for the world is incredibly inspiring. I'm looking forward to learning more about the new grad role you have available - really excited overall!

STAR

The STAR method is a pretty common way of answering general interview questions. You don't necessarily have to follow it exactly, but it will help you give easily understandable and compelling answers:

Situation: Describe the premise of the situation or problem. What needed to be accomplished and more importantly why was it important?

Target/Task: What goals were you working toward or what is the definition of your project?

Action: Describe the body of your work and the actions that you took to solve the problem or achieve the goals at hand.

Result: What was the impact that you made and how did the team or users react? Why was this experience either a great learning experience or a resounding success?

Generally, you will want your answers to be around 2 to 3 minutes, but this varies greatly, and it is better to give a full 5 to maybe even 10 minute answer as long as it is focused rather than an incomplete answer.

Keep Your Answers Succinct

Generally, you'll want to keep your answers focused on the specific story or topic at hand. If you have an interesting tangent, you can indicate that by saying something like "I'd be happy to add more detail to this later" and continue on with your main answer for now.

Be Specific

Describe concrete steps that you took to solve problems and what tools you utilized. Talk about the programming languages you used, the types of questions that you asked, or the kind of solution you used.

"I" not "we"

The interviewer cares only marginally what your team accomplished. They are not hiring your whole team or your manager, the are hiring you. Talk about your specific tasks, actions, thoughts, or questions. The more impact that you can show that you specifically made the better.

Avoid Dangerous Words

Anything overly pessimistic, critical, or self-degrading should be avoided at all cost. Examples of dangerous words include:

Overly Pessimistic - Fail, Disappoint
Overly Critical - Stupid, Idiot
Overly Self-degrading - Lazy, Clueless

Before the Interview

Research the company. 15 minutes of research can have a serious impact on your ability to relate yourself and your experience to the company. The "About" and "Product(s)" pages will often have all the information you need. Look for the company's mission statement, their history, and read the descriptions about their core products. If you're looking to go above and beyond you can also dig into their blog and get some good bits of information from there as well.

In these interviews you will be evaluated based off of how well you fit the cultural values. These can typically be found on a company's "About" page on their website. Shaping your answers to fit their values is critical. Also of incredible importance is effectively communicating why you are interested in the mission of the company and you certainly can't do that if you don't know what their mission is. You can typically find information about the company mission on their "About" page as well.

The End of the Interview

Ask thoughtful questions to show how interested you are in the company and that you've actively been thinking about the position, their products, their mission, etc. You can avoid sounding generic by simply rephrasing an overused question.

The most important question to ask is what are the next steps. Clarifying that you would like to move forward in the process and that you are planning for the next round is of the utmost importance.

After the Interview

Be sure to follow up with your interviewer with a kind, grateful message, however don't overdo it here and risk coming off as not genuine. This followup is also a chance for you to correct any mistakes that you may have made in your interview. Mentioning how you continued to think about questions or problems that they asked and suggesting corrections to your mistakes can be the difference between a "yes" and a "no".

Conclusion

These Are Just the Basics

There's so much more to talk about when it comes to best preparing for your job search. We've covered some of the basics to get you started above but to give you an idea of what else there is that the Pathrise program covers, including but not limited to:

- Portfolio websites
- Project development
- Career path case studies
- Advanced resume optimization
- Informational interviews
- Cold email retroactive templating
- Boolean searching on LinkedIn
- Preparing for high frequency behavioral interview questions
- Building a behavioral matrix
- Advanced technical interview questions
- Company specific interviewing frameworks
- Mapping out interview processes
- Understanding level/geo compensation codes
- Negotiation frameworks
- Modeling job search pipeline probabilities

The job search is already an incredibly stressful process and our intention with this guide is certainly not to overwhelm you. Our goal here is to help you understand that the job search is something that should be approached systematically and optimized. The more effort and thought you put into it, the more you get out of it. So don't just apply to a few places and hope to hear back. Take your destiny into your own hands and starting being proactive with some of the tactics that we've shared with you above.

Good luck and best wishes from Pathrise!