Engineer Seeking FIRE

(https://engineerseekingfire.com/)

A SOFTWARE ENGINEER'S PATH TO FINANCIAL INDEPENDENCE AND EARLY RETIREMENT (FIRE)

HOME (HTTPS://ENGINEERSEEKINGFIRE.COM)

START HERE (HTTPS://ENGINEERSEEKINGFIRE.COM/YOUR-PLAN-TO-ACHIEVE-FINANCIAL-INDEPENDENCE/)

FINANCIAL INDEPENDENCE (HTTPS://ENGINEERSEEKINGFIRE.COM/CATEGORY/FINANCIAL-INDEPENDENCE/)

TECH CAREERS (HTTPS://ENGINEERSEEKINGFIRE.COM/CATEGORY/CAREER-DEVELOPMENT/)

EXPENSE REPORTS (HTTPS://ENGINEERSEEKINGFIRE.COM/CATEGORY/EXPENSE-REPORTS/)

ABOUT ME (HTTPS://ENGINEERSEEKINGFIRE.COM/ABOUT-ME/)

(https://www.twitter.com/EngSeekingFIRE) (https://www.facebook.com/EngineerSeekingFIRE)
(https://www.pinterest.com/EngineerSeekingFIRE) (https://www.instagram.com/EngineerSeekingFIRE)
(https://engineerseekingfire.com/feed/) (mailto:engineerseekingfire@gmail.com)

Engineer Seeking FIRE

I am an engineer, in my late 30s. I have 15+ years of professional experience in consulting, retail and technology. I've worked at Microsoft, Amazon and Google as a Software Engineer, a Product Manager and a Technical Program Manager.

In my personal life, I am married with 2 children.

How to Prepare for Software Engineering Interviews

Never miss a story!

/

Engineer Seeking Fire (https://engineerseekingfire.com/author/engineerseekingfire/) -
April 4, 2020 -
Tech Careers (https://engineerseekingfire.com/category/career-development/) -
9 Comments (https://engineerseekingfire.com/how-to-prepare-for-software-engineering-interviews/#comments)

Disclosure: This post might contain affiliate links. If you click through and make a purchase, I'll earn a commission, at no additional cost to you. Read my full disclosure here (https://engineerseekingfire.com/privacy-policy/).

The goal of this post is to explain how to prepare for Software Engineering interviews. I will give an overview of the interview structure, dive deep into the 3 types of interview questions and provide resources for mock interviews.

This is part of a mini-series of posts related to career development. The other related posts are:

1. **Interview Preparation Guides**
   - How to Prepare for **Product Manager (PM)** Interviews (https://engineerseekingfire.com/how-to-prepare-for-product-manager-interviews/)
   - How to Prepare for **Technical Program Manager (TPM)** Interviews (https://engineerseekingfire.com/how-to-prepare-for-technical-program-manager-interviews/)
2. **Role-specific information**
   - What are the differences between Product Managers, Program Managers, Engineering Managers and Marketing Managers? (https://engineerseekingfire.com/differences-between-product-program-engineering-marketing-managers/)
   - Inside the Culture of the Top Tech Companies (https://engineerseekingfire.com/inside-the-culture-of-the-top-tech-companies/)
   - Salaries for Software Engineers are On FIRE! (https://engineerseekingfire.com/salaries-for-software-engineers-are-on-fire/)

Overview

A typical interview for high-tech companies consists of 2 rounds:

1. Phone interview (45 minutes): The interviewer either calls the interviewee via phone or they use a Communications app (e.g. Skype, Zoom, etc)
2. Onsite interview (4-5 interviews, 45 minutes each): The interviewee visits the company's offices and talks to the interviewers face to face

There are 3 types of questions:

1. **Coding questions**, e.g. merge 2 sorted linked lists into a sorted linked list
2. **System designs questions**, e.g. how would you design Twitter/Uber/AirBnb/Facebook/Skype/etc
3. **Behavioral questions**, e.g. tell me about a time when you had to resolve a conflict

For more information about the types of questions, you can take a look at the **official interview preparation guides for Software Engineers** from:

- Facebook (https://www.facebook.com/careers/swe-prep-onsite) (download the interview prep guide (https://scontent-sea1-1.xx.fbcdn.net/v/t39.2365-6/75448664_529293754518370_5563533277542744064_n.pdf?_nc_cat=108&_nc_sid=ad8a9d&_nc_ohc=mh9pla0KOSAAX_5-pHc&_nc_ht=scontent-sea1-1.xx&oh=bf14ba63240ce3e8815725e41b5d6746&oe=5EAFC3A8) at Step 2)

Sign up to get an email when a new post is published.

First Name
Your first name

Email address:
Your email address

SIGN UP

Follow Me

(https:// www.in stagra m.com/ Engine erSeeki ngFIRE)

(https:// www.fa cebook .com/E ngineer Seekin gFIRE)

(https:// www.tw itter.co m/Eng ineerSeekin gFIRE)

(https:// www.pi nterest. com/En gineer Seekin gFIRE)

(https:// www.e ngineer seekin gfire.co m/feed)

/

    - Official blog post (https://www.facebook.com/careers/life/preparing-for-your-software-engineering-interview-at-facebook/) in the Facebook Careers website
- Google (https://www.slideshare.net/interviewcoach/google-interview-prep-guide-software-engineer)
- How to Interview At Amazon (https://www.linkedin.com/pulse/how-interview-amazon-leadership-david-anderson/) by David Anderson (Bar Raiser)

The types of questions that are asked in each interview depend mostly on the experience of the interviewee, as is shown in the matrix below.

	CODING QUESTIONS	SYSTEM DESIGN QUESTIONS	BEHAVIORAL QUESTIONS
Entry-level Google/Facebook: L3 Amazon: L4 Microsoft: 59,60	Yes (most important question)	No	No/Limited
Intermediate Google/Facebook: L4 Amazon: L5 Microsoft: 61, 62	Yes	Yes (diffentiating between Intermediate and Senior level)	No/Limited
Senior Google/Facebook: L5 Amazon: L6 Microsoft: 63, 64	Yes	Yes (diffentiating between Intermediate and Senior level)	Yes (diffentiating between Senior and Principal level)
Principal Google/Facebook: L6 Amazon: L7 Microsoft: 65-67	Yes	Yes	Yes (diffentiating between Senior and Principal level)

As is shown in the matrix above:

1. Entry-level candidates are only asked coding questions
2. The differentiating factor between whether a candidate will get an offer for Intermediate level or Senior level are the System Design questions
3. The differentiating factor between whether a candidate will get an offer for Senior level or Principal level are the Behavioral questions, e.g. to understand previous leadership experiences

The following sections go into further detail about how to prepare for each question type.

If you find that my advice has benefited you in any way, be aware that I love a nice warm cup of non-fat latte with chocolate syrup!

Buy me a co

(https://www.buymeacoffee.com/

Categories

Decrease expenses (https://engineerseekingfire.com/category/financial-independence/decrease-expenses/)

Expense reports (https://engineerseekingfire.com/category/expense-reports/)

Increase income (https://engineerseekingfire.com/category/financial-independence/increase-income/)

Investments (https://engineerseekingfire.com/category/financial-independence/investments/)

Tech Careers (https://engineerseekingfire.com/category/career-development/)

Archives

August 2020 (https://engineerseekingfire.com/2020/08/) (1)

June 2020 (https://engineerseekingfire.com/2020/06/) (3)

May 2020 (https://engineerseekingfire.com/2020/05/) (1)

April 2020 (https://engineerseekingfire.com/2020/04/) (4)

March 2020 (https://engineerseekingfire.com/2020/03/) (3)

February 2020 (https://engineerseekingfire.com/2020/02/) (3)

/

January 2020 (https://engineerseekingfire.com/2020/01/) (6)

Coding Questions

Step 1: Understand the fundamental concepts

In the beginning, you need to understand/refresh the fundamental concepts, e.g. linked lists, string handling, tree parsing, dynamic programming, etc. The following **books** are great resources:

- Cracking the Coding Interview (https://amzn.to/3eZ8dQ3): This is a great introductory book
- Elements of Programming Interviews: This book goes into much more depth. Depending on your preferred programming language, you can buy the version that uses C++ (https://amzn.to/2A6WNuD), Java (https://amzn.to/2Y5AAVL)or Python (https://amzn.to/2Y2cSK2)
- The Algorithm Design Manual (https://amzn.to/2z7hTZh): This book is more advanced and can be used as a very useful resource while working on Leetcode problems. Many Leetcode problems are based on the algorithms that are shown in the book

If you prefer to watch **YouTube videos**, then you can use the following:

- Back to Back SWE (https://www.youtube.com/channel/UCmJz2DV1a3yfgrR7GqRtUUA/playlists) channel: Lots of videos on fundamental concepts. Also, it includes very easy-to-understand explanations on dynamic programming, which is always a tricky subject
- Dynamic Programming for Beginners (https://www.youtube.com/channel/UClnwNEngsXoIp_tgJ2jZWfw): Great introduction to Dynamic Programming, which is one of the most difficult types of questions that are asked during the interviews

Finally, if you prefer to **read the structured material online** in a class format, then take a look at the following:

- 14 Patterns to Ace Any Coding Interview (https://hackernoon.com/14-patterns-to-ace-any-coding-interview-question-c5bb3357f6ed) (by Hackernoon): Great way to start your preparation by mastering these 14 basic patterns
- Problem Solving with Algorithms and Data Structures using Python (https://runestone.academy/runestone/books/published/pythonds/index.html) by Runestone Academy: Includes free open source book, as well as class videos
- Educative.io (https://engineerseekingfire.com/recommends/educative-io/): Multiple paid courses targeted specifically to interview preparation. All of them are great classes, especially if you are starting with your interview preparation.

- **Coding**
    - Grokking the Coding Interview: Patterns for Coding Questions (https://engineerseekingfire.com/recommends/grokking-the-coding-interview-patterns-for-coding-questions/)
    - Grokking Dynamic Programming Patterns for Coding Interviews (https://engineerseekingfire.com/recommends/grokking-dynamic-programming-patterns-for-coding-interviews/)
    - Hacking the Coding Interview (https://engineerseekingfire.com/recommends/hacking-the-coding-interview/)
- **Data Structures**
    - Mastering Data Structures (https://engineerseekingfire.com/recommends/mastering-data-structures/)
- Geeks for Geeks (https://www.geeksforgeeks.org/hard/?ref=leftbar): Online portal that goes through several Computer Science concepts and also includes a step-by-step preparation guide (https://www.geeksforgeeks.org/a-complete-step-by-step-guide-for-placement-preparation-by-geeksforgeeks/?ref=leftbar), as well as company-specific preparation guides (https://www.geeksforgeeks.org/company-preparation/?ref=leftbar)

Step 2: Practice programming questions using LeetCode

LeetCode (https://leetcode.com)is the current standard for coding interview preparation. Most candidates will solve hundreds of questions (e.g. 300-500) as preparation for the top high-tech companies. Also, interviewers typically select their questions from the LeetCode question database. There are other similar platforms (e.g. HackerRank (https://www.hackerrank.com/)), but my suggestion is to use LeetCode as your main website for coding practice.

Here are some answers regarding the most common questions about how to structure your preparation.

Which programming language should I use?

- If you are an experienced programmer, you probably have a preferred programming language that you are most comfortable with. Use that one.
- If the job description refers to a specific language, then use that one.
- If you feel comfortable with multiple languages or want to learn a new one, then most interviewees use Python, Java or C++. Fewer use C# or Javascript.
    - Python: Requires the least lines of code, which is important, when you have time constraints. Also, it's the only option for ML jobs. However, it might lead to complicated code (especially for interviewers, who are not familiar with Python) and it does not support binary trees natively.
    - Java: Supports OOP and simplifies garbage collection
    - C++: Lots of advanced features such a pointers, memory management, etc which make the language very powerful, but also requires more lines of code and is more prone for errors. If you decide to use it, you should also learn STL.
- You can also read some additional thoughts regarding this question from the Tech Interview Handbook (https://yangshun.github.io/tech-interview-handbook/picking-a-language)

How should I approach the problems?

- In the beginning, try to spend at least 45 minutes in each problem before looking at the solution. As you progress, this number should be decreasing quite a bit, because in most interviews you'll need to answer 2 Medium questions in 45 minutes.
- If you look at a solution and understand it, then always return to the problem and write the code yourself (without looking at the solution).
- Keep notes for each problem that you solve, so that you can refer to them later

- Make sure that you understand the pattern that is used to solve each problem, instead of memorizing a solution. This way you'll be able to recognize patterns between groups of problems (e.g. "this is a new problem, but the algorithm looks similar to the other 15 related problems that I've solved".
- You'll know that you are progressing, when a) you understand the pattern required to solve each problem just by reading it and b) you see the time required to solve each problem get lower (e.g. "I solved problem A in 50 minutes and problem B in 30, even though both of them use the same approach")

How should I prioritize which problems to solve?

1. Start with the Curated List of Top 75 Questions (https://www.teamblind.com/post/New-Year-Gift---Curated-List-of-Top-100-LeetCode-Questions-to-Save-Your-Time-OaM1orEU) (also available as a LeetCode list (https://leetcode.com/list/xoqag3yj/), but it's missing the problem "Cherry Picking #741 (https://www.teamblind.com/tag/741)"). This list was created by somebody who finished 500 LeetCode problems and kept track of the most useful ones.
2. Continue with LeetCode's Top Interview Questions (https://leetcode.com/problemset/top-interview-questions/). This list has been created by LeetCode and consists of very popular questions. It also has a partial overlap with the previous list. Make sure that you finish with all the Medium problems on that list.
3. Take a look at the curated list of 170+ Leetcode questions (https://seanprashad.com/leetcode-patterns/) grouped by their common patterns by Sean Prasad (answers shown here (https://github.com/SeanPrashad/leetcode-patterns))
4. The Tech Interview Handbook has a 5-week list of Leetcode problems (https://yangshun.github.io/tech-interview-handbook/best-practice-questions) to practice on a weekly basis
5. Buy LeetCode Premium (currently $35/month). Look at the top problems for the companies that you are targeting. At a minimum, practice with the lists that include the top 50 questions for Google and Facebook.
6. Remember that most Easy problems are too easy for an interview, whereas the Hard problems might not be solvable within the interview slot (45 minutes typically). So, it's very important to focus most of your time on Medium problems. A good ratio of Easy/Medium/Hard is 20/70/10.
7. If you identify an area, which seems more difficult than others, you should take some time to dive deeper by reading books/tutorials, viewing videos, etc in order to understand the concepts better.
8. Practice with the mock interview sets. They have a time limit and prevent access to the forum, so they are more geared to a real interview. You can also compare your rating against everybody else, who did the same set of problems.

What if I need more detailed/structured solutions for some problems, because the idea is not clear, even after reading the solution?

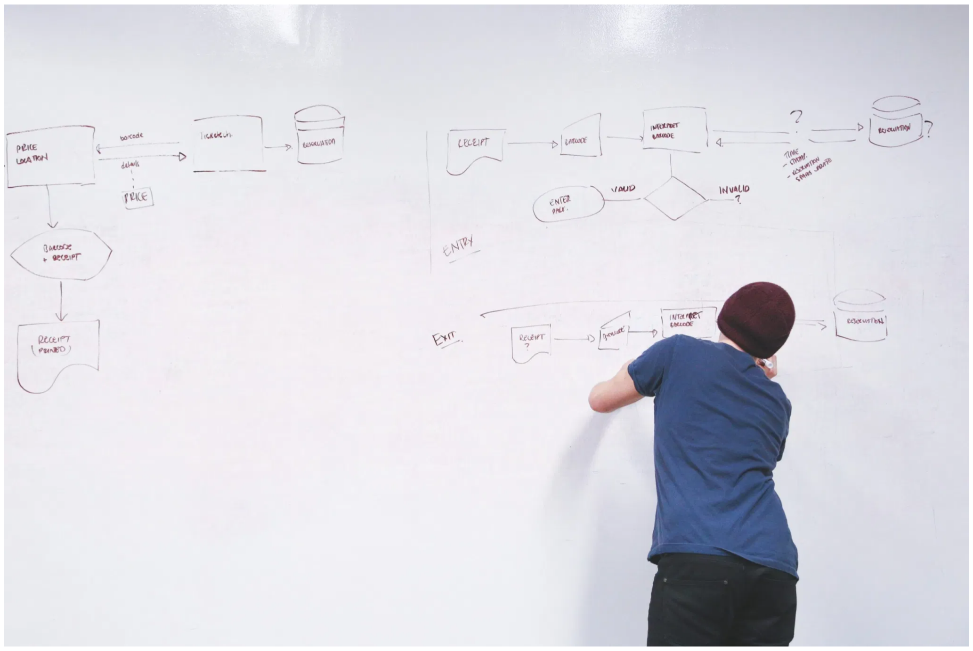
Observe how others approach the problems and learn from their detailed thought process, instead of directly looking at the final solution:

1. Nick White's LeetCode Solutions playlist (https://www.youtube.com/playlist?list=PLU_sdQYzUj2keVENTP0a5rdykRSgg9Wp-) on YouTube: Nick has recorded himself solving ~190 LeetCode problems
2. Kevin Naughton Jr's LeetCode Coding Interview Questions playlist (https://www.youtube.com/playlist?list=PLi9RQVmJD2fYXgBAUfaN5MXgYPUTbzqeb) in YouTube: Kevin has recorded himself solving ~120 LeetCode problems
3. User Grandyang from Cnblogs has solved > 900 Leetcode problems and has posted his solutions with comments in Mandarin (https://www.cnblogs.com/grandyang/p/4606334.html)(English translation via Google Translate (https://translate.google.com/translate?sl=auto&tl=en&u=https%3A%2F%2Fwww.cnblogs.com%2Fgrandyang%2Fp%2F4606334.html))

4. Hacking the Coding Interview (https://engineerseekingfire.com/recommends/hacking-the-coding-interview/) by Educative.io: This is a paid course that covers ~80 LeetCode problems in detail

How will I know when I am ready to interview?

- The most important metric is how fast you can solve each category of problems.
- Remember that the duration of most interviews is 45 minutes, during which you'll have to solve either 2 Medium problems (most common case) or 1 Hard.
- Here are some same target times for each category:
  - Easy: 5-10 minutes
  - Medium: 15-20 minutes
  - Hard: 30-40 minutes
- In order to reach those levels, you'll need to solve around 200-500 LeetCode problems, which corresponds to 3-5 months of preparation time. These numbers are just a rule of thumb, as the actual time depends on your current coding experience and knowledge.



System Design Questions

Step 1: Understand the structure of the expected answer

Your first goal should be to understand what the system design interview is about, i.e. what types of questions are asked, how to structure your answer and how deep you should go in your answer.

- Read the section "How to approach a system design interview question" from the System Design Primer (https://github.com/donnemartin/system-design-primer/blob/master/README.md#how-to-approach-a-system-design-interview-question)
- Read the course materials from Hired In Tech (https://www.hiredintech.com/classrooms/system-design/lesson/52) to get another view about the sample structure
- Start with this example from Practice Coding Interview (http://www.practicecodinginterview.com/blog/2018/8/18/the-system-design-interview) to a see a full answer
- Educative.io (https://engineerseekingfire.com/recommends/educative-io/) classes
  - Grokking the System Design Interview (https://engineerseekingfire.com/recommends/grokking-the-system-design-interview/) class (section "System Design Problems")
    - I highly recommend buying the full class, especially if you are starting with system design. It is a great class to understand the basic structure of a system design problem

    - Grokking the Object Design Interview (https://engineerseekingfire.com/recommends/grokking-the-object-oriented-design-interview/) class
        - Great class to learn about Object-Oriented design
- Read the System Design Cheatsheet (https://gist.github.com/vasanthk/485d1c25737e8e72759f) by vasanthk

Step 2: Dive deeper into the building blocks required to answer these questions

After going through a few of the above examples, you have probably heard lots of new terms and technologies that you are not so familiar with. In order to be able to answer these types of questions you will need to dive deeper into a lot of different technologies.

My suggestion is to start with the following videos that provide an **overview** about how to design scaleable systems.

- Building Dynamic Websites (https://www.youtube.com/watch?v=-W9F__D3oY4&list=PLmhRNZyYVpDmLpaVQm3mK5PY5KB_4hLjE&index=10) – Harvard CS75 (Summer 2012) by David Malan. The linked video is from lecture 9, which summarizes the whole class, but if you want to dive deeper into specific areas, you can also view the rest of the videos in the class (https://www.youtube.com/playlist?list=PLmhRNZyYVpDmLpaVQm3mK5PY5KB_4hLjE)
- Distributed Systems in One Lesson (https://www.youtube.com/watch?v=Y6Ev8GIlbxc) by Tim Berglund
- Four Distributed Systems Architectural Patterns (https://www.youtube.com/watch?v=tpspO9K28PM) by Tim Berglund
- Building Financial Systems on Eventually Consistent DBs (https://www.youtube.com/watch?v=KH0l8QqhzYk) by Rahul Pilani

The following websites have sample list of **additional design concepts** that you should be familiar with:

- System Design Primer (https://github.com/donnemartin/system-design-primer/blob/master/README.md#index-of-system-design-topics) (section "Index of system design topics")
- Gaurev Sen's YouTube Tutorials (https://www.youtube.com/playlist?list=PLMCXHnjXnTnvo6alSjVkgxV-VH6EPyvoX)
- Grokking the System Design Interview (https://engineerseekingfire.com/recommends/grokking-the-system-design-interview/) (section "Glossary of system design basics"). This is a good list of topics that you need to know
- Mario Gerard (https://www.mariogerard.com/interview-questions-for-a-technical-program-manager-tpm/) (section "Here Is My List of Fundamental Technical Workings A TPM Needs To Know" – scroll towards the bottom)

Finally, if have enough time and really want to dive deep into system design, then you can read about more **advanced concepts** at the following resources:

- The High Scalability blog (https://highscalability.squarespace.com/blog/category/example) has a series of posts titled Real Life Architectures that explain how some popular websites were architected
- The book "Designing Data-Intensive Applications (https://amzn.to/2BF6xg7)" by Martin Klepmann is the best book regarding system design
- The "Amazon's Builder Library (https://aws.amazon.com/builders-library/)" explains how Amazon has built their own architecture
- If you **really really** want to dive deeper into how existing systems are built, you can also read the published papers on GFS, Dynamo, Haystack, Cassandra, Bigtable, Raft, Paxos, Chubby,

Zookeeper, Spanner, Haystack, Kafka, Azure Storage (https://sigops.org/s/conferences/sosp/2011/current/2011-Cascais/printable/11-calder.pd), TAO by Facebook (https://www.usenix.org/system/files/conference/atc13/atc13-bronson.pdf), Memcache (https://www.usenix.org/system/files/conference/nsdi13/nsdi13-final170_update.pdf)

The list of resources is endless, so make sure that you don't get lost trying to understand every little detail about every technology available. Also, feel free to go to the next step, where you practice with additional questions, and use this section as a reference.

Step 3: See how others answer design questions

Now that you understand the structure and the building blocks, your next goal should be to get a good understanding of how other people answer these types of questions. Videos are a great way to do this. YouTube has a lot of great channels answering how to design Twitter, Pastebin, Uber, etc. Here are some great channels to start with.

- Tushar Roy – System design (https://www.youtube.com/playlist?list=PLrmLmBdmIlps7GJJWW9I7N0P0rB0C3eY2)
- Success in Tech (https://www.youtube.com/playlist?list=PLA8lYuzFlBqAy6dkZHj5VxUAaqr4vwrka)
- System Design Interview channel (https://www.youtube.com/channel/UC9vLsnF6QPYuH51njmIooCQ/videos)
- YouTube playlist with multiple system design videos (https://www.youtube.com/playlist?list=PL73KFetZlkJSZ9vTDSJ1swZhe6CIYkqTL)

Step 4: Practice, practice, practice

The most important part of your interview preparation is to practice real design questions. The following websites have questions and answers:

- System Design Primer (sections "System design interview questions with solutions (https://github.com/donnemartin/system-design-primer/blob/master/README.md#system-design-interview-questions-with-solutions)" and "Additional system design interview questions (https://github.com/donnemartin/system-design-primer/blob/master/README.md#additional-system-design-interview-questions)")
- System Design Preparation by shashank88 (section "Common Design Questions (https://github.com/shashank88/system_design/blob/master/README.md#designques)")
- Grokking the System Design Interview (https://engineerseekingfire.com/recommends/grokking-the-system-design-interview/) class (section "System Design Problems")
- System Design Interview (section "Hot Questions and Reference (https://github.com/checkcheckzz/system-design-interview#qs)") by checkcheckzz
- Crack the system design interview (https://tianpan.co/notes/2016-02-13-crack-the-system-design-interview) at tianpan.co

Behavioral Questions

Some examples of behavioral questions are:

- Tell me about a time where you influenced or pushed back on a tech design.
- Tell me about a time that you made a trade-off call that involved engineering
- Tell me about a project that you're most proud about
- Tell me about a time that you failed
- Tell me about a time when you had to resolve a conflict
- Tell me about a time when you led a team

These types of questions are answered using the STAR framework:

- **S**ituation (2-3 sentences)
  - Explain what was happening, e.g. the project that you were working on, the teams involved and what was at stake
- **T**ask (1-2 sentences)
  - What was your role
- **A**ctions (the bulk of your answer)
  - Talk about 3 problems that you were facing
  - Explain what actions you took, in order to solve these problems
  - IMPORTANT: Use "I", not "we", since you want to talk about your tasks, not the team's tasks
- **R**esult (2-3 sentences)
  - What happened in the end
  - e.g. you got an award/promotion, the project launched on time and the company earned $XXX

Please note the following:

- Prepare 3-5 answers using the STAR framework before the interview and write them down (in a text file, on paper, etc)
- Your answers should not be longer than 2 minutes
- Remember to use "I" instead of "we" in your answer, because the interviewer is interested on the actions that you took and not on what the team did

You can find lots of resources about these questions in the internet (also check YouTube), so I won't go deeper into these questions. Here are same good links to start with:

- Grokking the Behavioral Interview (https://engineerseekingfire.com/recommends/grokking-the-behavioral-interview/) by Educative.io (free)
- 7 Proven Job Interview Questions—and What to Look for in the Answers (https://hire.google.com/articles/7-proven-job-interview-questions/) from the Hire Team (by Google)
- Common behavioral questions from the Tech Interview Guide (https://yangshun.github.io/tech-interview-handbook/behavioral-questions)
- How to Ace Your Behavioral Interview (https://simpleprogrammer.com/ace-behavioral-interview/)
- 45 Sample Behavior Questions to Use During Non-Technical Interview With Developers (https://devskiller.com/45-behavioral-questions-to-use-during-non-technical-interview-with-developers/)
- Behavioral Interviews for Software Engineers (https://www.youtube.com/watch?v=zlJ1qRCPHUw)
- Amazon Interview: Behavioral Questions & Leadership Principles for Software Development (SDE) Roles (https://www.youtube.com/watch?v=RMA7tI-LTWY)

Mock interviews

After going through all the materials above and getting ready to start the interview process, it is always helpful to get some feedback in a low-stress environment. Mock interviews help you test your knowledge and get feedback from another person.

Here is how you can find interviewers for mock interviews:

- Pramp.com (https://pramp.com) (free): This website connects you with other peers, who are also studying for interviews, so that you can all learn together
- PractiseCodingInterview.com (https://www.practicecodinginterview.com/) (paid), GainLo (http://www.gainlo.co/)(paid) and interviewing.io (https://interviewing.io) (paid): These websites will match you with experienced interviewers, who are currently working in the top high-tech companies
- Use your network: You can talk to other developers that you know (either from your own company, via LinkedIn, etc) and ask them to do a mock interview for you

Related

(https://engineerseekingfire.com/how-to-prepare-for-technical-program-manager-interviews/)
How to Prepare for Technical Program Manager (TPM) Interviews (https://engineerseekingfire.com/how-to-prepare-for-technical-program-manager-interviews/)
April 13, 2020
In "Tech Careers"

(https://engineerseekingfire.com/how-to-prepare-for-product-manager-interviews/)
How to Prepare for Product Manager (PM) Interviews (https://engineerseekingfire.com/how-to-prepare-for-product-manager-interviews/)
April 16, 2020
In "Tech Careers"

(https://engineerseekingfire.com/differences-between-product-program-engineering-marketing-managers/)
What are the differences between Product Managers, Program Managers, Engineering Managers and Marketing Managers? (https://engineerseekingfire.com/differences-between-product-program-engineering-marketing-managers/)
March 24, 2020
In "Tech Careers"

TAGS: APPLE (HTTPS://ENGINEERSEEKINGFIRE.COM/TAG/APPLE/), FACEBOOK (HTTPS://ENGINEERSEEKINGFIRE.COM/TAG/FACEBOOK/), GOOGLE (HTTPS://ENGINEERSEEKINGFIRE.COM/TAG/GOOGLE/), INTERVIEW PREPARATION (HTTPS://ENGINEERSEEKINGFIRE.COM/TAG/INTERVIEW-PREPARATION/), MICROSOFT (HTTPS://ENGINEERSEEKINGFIRE.COM/TAG/MICROSOFT/), SOFTWARE ENGINEERING INTERVIEWS (HTTPS://ENGINEERSEEKINGFIRE.COM/TAG/SOFTWARE-ENGINEERING-INTERVIEWS/)

⟵ Previous Post
Inside the Culture of the Top Tech Companies

Next Post ⟶
How to Prepare for Technical Program Manager (TPM) Interviews

[(https://engineerseekingfire.com/inside-the-culture-of-the-top-tech-companies/)](https://engineerseekingfire.com/inside-the-culture-of-the-top-tech-companies/)

[(https://engineerseekingfire.com/how-to-prepare-for-technical-program-manager-interviews/)](https://engineerseekingfire.com/how-to-prepare-for-technical-program-manager-interviews/)

THIS POST HAS 9 COMMENTS

Pixel_28

13 AUG 2020 REPLY

Thank you so much for this post. I found it very helpful and I have made a note of some points mentioned here.
I would like to know what was your typical daily schedule like? How did you distribute your time to cover this?

Loading...

PRANAV TRIPATHI

23 JUN 2020 REPLY

Amazingly compiled list. Thanks a lot man! 🙂

Loading...

Kirill (http://kiri11.ru/)

8 JUN 2020 REPLY

Correction: Python does support heaps. There is a standard package heapq.
There are no built-in binary trees in Python, though. But you can always say “imagine we have such library…”

Loading...

Engineer Seeking Fire (https://engineerseekingfire.com)

9 JUN 2020 REPLY

Thanks, Kirill! I fixed the text above.

Loading...

Nghi

18 MAY 2020 REPLY

Thank you Fireman!!! I was so lucky to stumble on this today since I plan studying from the basics to get into big tech company despite being a junior in CS. Your guide is a perfect road map for my plan and I am going to follow your advice to achieve my goal!

Loading...

Engineer Seeking Fire (https://engineerseekingfire.com)

9 JUN 2020 REPLY

Happy to help, Nghi! Good luck!

Loading...

True

12 MAY 2020 REPLY

Thank you fireman. I am currently aiming for a SDE 2 position at Amazon and this post is very helpful to me. Just a question, in the system design round of a typical SDE 2 interview, how often do you run into a interviewer who goes past the things taught in grokking or system design primer and drills you on very domain/technology specific questions ?

Loading...

Engineer Seeking FIRE

15 APR 2020 REPLY

Well, a typical interview is 45 minutes. In the beginning you will spend a few minutes for introductions and at the end you will spend a few minutes to ask questions to the interviewer. As a result you have 30-40 minutes for the actual interview. During that time you will have to answer the interviewer's question. If that question is a Leetcode Hard problem, then that's the only time that you get. Practice, practice, practice…

Loading...

Nah

15 APR 2020 REPLY

You need to solve a hard in 30-40 minutes, are you crazy. No one can do that unless you are John von Neumann. Some of the hard problems are based on PhD work that people spend 3-4 years to solve.

Loading...

Leave a Reply