

# The Developer's Guide To **TECHNICAL** Interviews. **WS.**

99.9% candidates end up getting rejected in technical interviews. And chances are, you will get rejected too.

The only way to avoid this is to do, what 99% are not doing.

**Understand:** Your skills, education, and experiences play a minor role in your selection.

And

Success in interviews depends upon your level of preparation. Your ability to understand the interviewer's perceptions. And delivering beyond his expectations.

In today's guide, you're going to learn all of this and much more.

Let's do this.

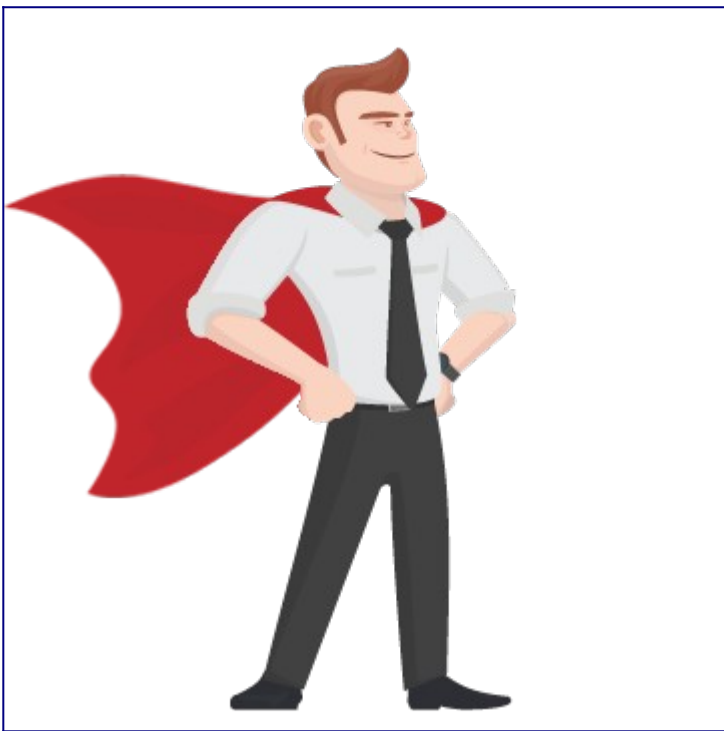


## **Contents**



## **CHAPTER 1**

### **How to fail an interview**



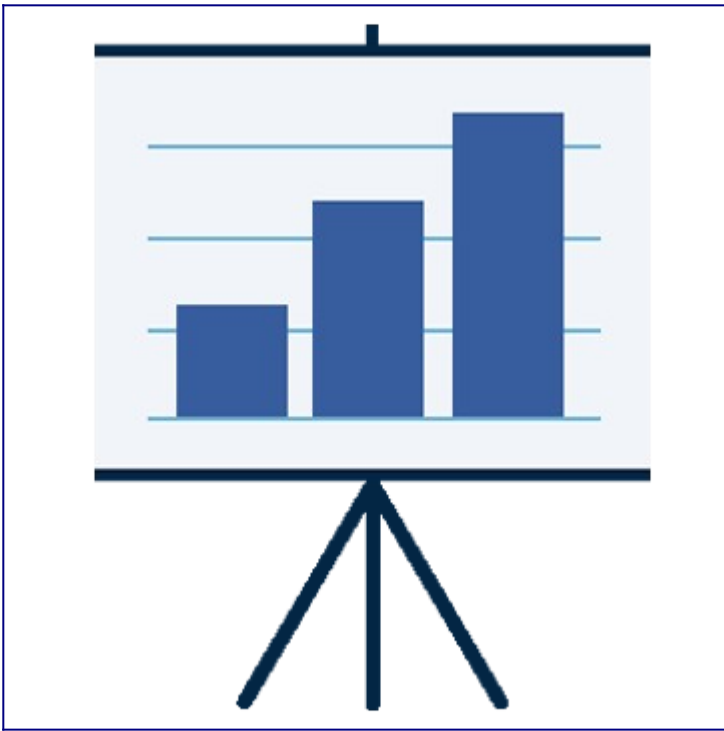
## **CHAPTER 2**

### **Interview success factors**



## **CHAPTER 3**

### **Pattern Behind Technical Interviews**



## **CHAPTER 4**

### **Interview rounds you need to go through**



## **CHAPTER 5**

### **How to prepare for technical interviews**



## **CHAPTER 6**

### **How to crack programming interviews**



## **CHAPTER 7**

### **What To Do If You Fail An Interview**



## CHAPTER 8

### Interview Questions & Resources

# What Is Technical Interview

---

Programming interviews are full of infinite complexities. In fact, it is much like going to war, where we fight for the survival of our career and ego.

Like no war can be won without a game plan, no job interview can either survive without it.

My whole professional life I have missed such a game plan. I was rejected numerous times, feeling dead on the ground, I was broke and scared.

It took me a long time to get it, but now I get it. And now I want to tell you about the importance of such a game plan.

Following guide has helped me to [survive](#) the most terrible and brutal interviews in my career. And I am sure it will help you too, not just survive but thrive in your next interview.

## You Must Watch This Video, If You Want to Beat The Competition

---



- Special Thanks to Tom Bilyeu for this video





There are two kinds of failures.

The first one comes from never preparing and understanding the interview process.



This kind of failure is worst. You can never learn from it.

And

The second one comes from being overconfident

It is also wrong because it blinds you from looking at your shortcomings.

Below are some of the most important reasons for people failing in technical interviews.

## 5 Reasons: You keep Failing In Interviews

---

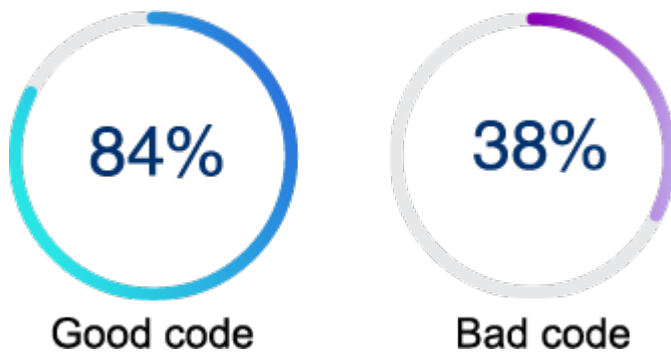
**Go unprepared:** The easiest way to fail any programming interview is to go without preparing any technical questions

**Be overconfident:** This is astonishingly common, especially with smart people. Being a jerk is the easiest way to fail an interview.

**No knowledge about the job:** Most people go for the interview without having any idea about the requirements of the job, even worse without any knowledge about the company.

**Bad code quality:** The code you will write during the interview will say a lot about your personality and overall skill level.

So writing bad code will land you an immediate rejection.



One research shows that candidates who write good code in interviews have 84% more chances of getting hired, whereas bad coders have only 38% chance.

**No Followup:** Not following up after the interview is another easiest way to get a rejection

### Interview Facts

On average, each corporate job attracts 250 applications. 4 to 6 candidates get an interview invitation. And only one gets the job.

*(Source: Glassdoor)*



### BOTTOM LINE:

Failing in interviews is easy. But to ace technical interviews most crucial things are.

Your attitude. Your level of preparation. Your knowledge about the job. And finally how well you follow up afterward.

---

How can I be successful in a job interview?

How can I improve my performance in technical interviews?

How can I beat the competition?

Then wait no more.



This chapter will teach you about the only success factors that matter the most.

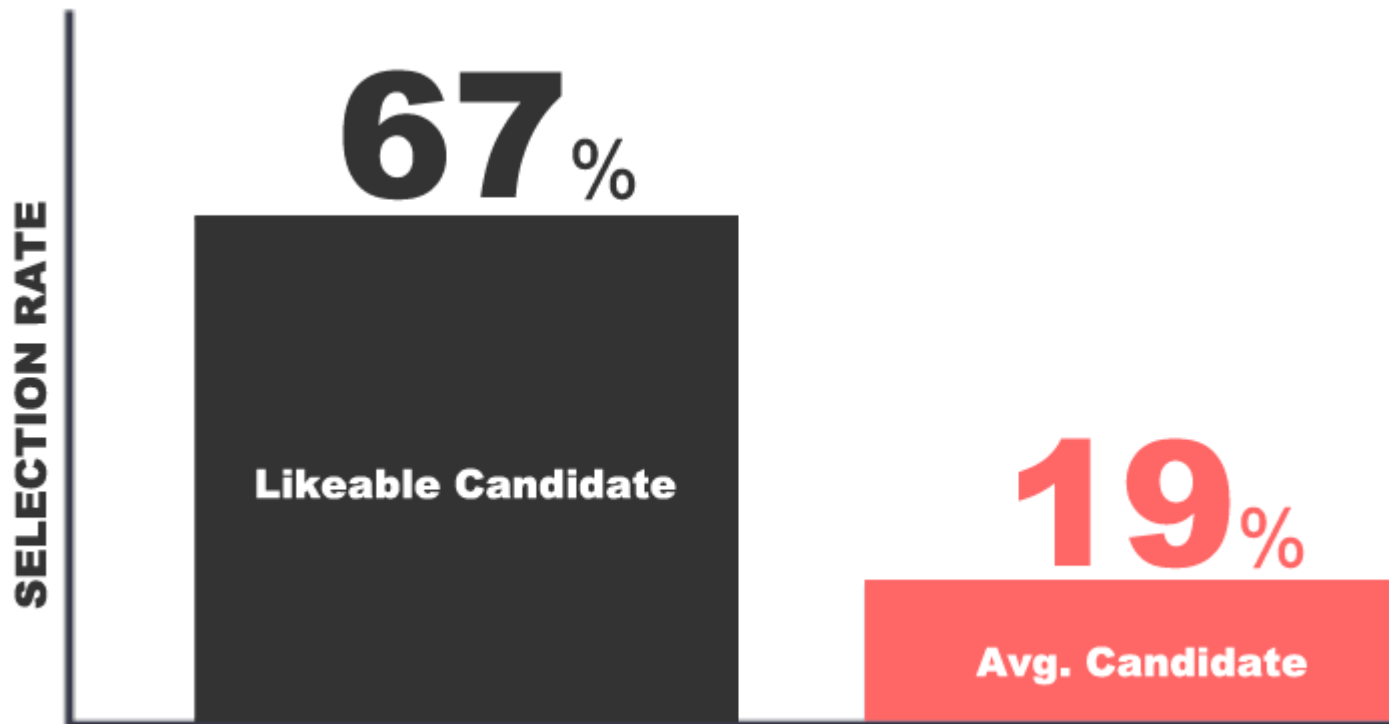
## Success Factors in Interviews

---

**Internal State of mind:** You don't get in life what you want, you get what you fight for. And when you decide my career, my future, my dreams and my financial independence is at stake and its worth fighting for it.

You will naturally [approach the technical interviews](#) with a different state of mind. And this will be the single most important factor for your success.

**Candidate Likability:** Likability is the cornerstone of success. Through likability alone you can standout and win.



No matter how talented you are. If interviewers do not like you, then there is no chance of getting hired.

**Ability to code:** Can you code? And how well you can code is another important factor for your success in the interview.

**Personality:** Candidate's overall personality, mannerism and ability to communicate and ask questions is also an important success factor.

## **Bonus Video: How To Rise & Shine In Interviews**

---



**BOTTOM LINE:**

Intelligence is overrated. What you need to succeed at interviews is a fighter's mindset, A killer personality and the ability to code.

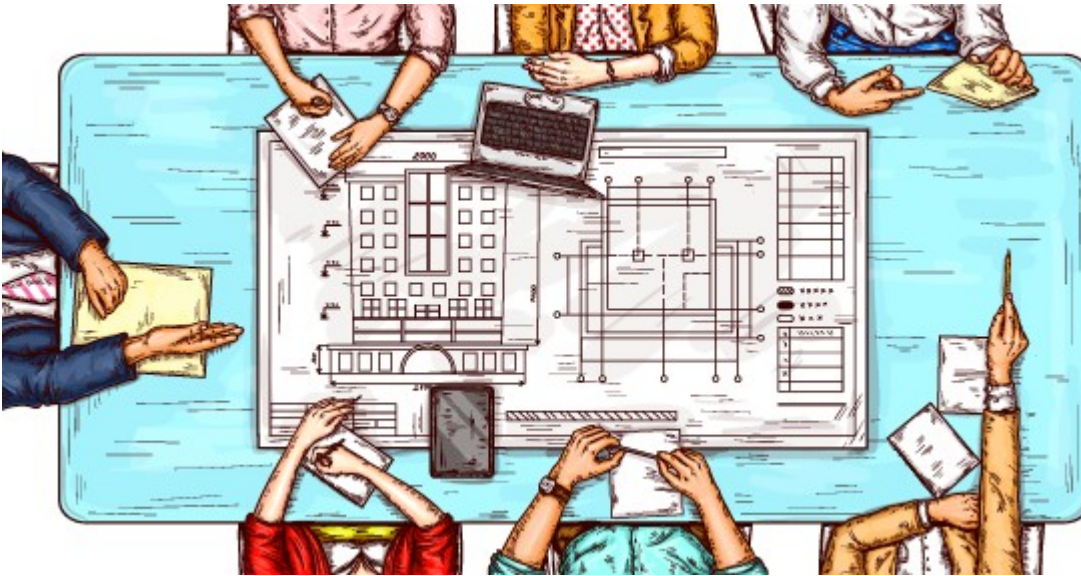
## **CHAPTER 3:**

# **Technical Interview Structure**

---

No two technical interviews are equal.

In fact: Every company regardless of the size or location has its style, merit and pattern of conducting technical interviews.



But in general technical interviews can be categorized into following patterns.

## Types Of Technical Interviews

---

### 1) Technical assignment before interview

In this type of interview, companies send candidates with some questions, exercises or a case studies to solve.

Candidates are expected to implement and test a fully working solution and send it to the company before the interview.

Then on the interview day, technical persons will ask very deep and detailed questions about the implementation, design considerations, performance and efficiency of the solution.

### 2) White board test

It is the most common and widely used method during the technical interviews.



You will be required to solve questions or design and implement an algorithm on the whiteboard.

Google is very notorious for these kinds of questions during their 6-7 on-site technical interviews.

To crack this part, you need some serious skills (Don't worry I will teach you exactly how you can get yourself ready to crack this type of interview at any big company)

### **3) Asking technical questions**

This is usually the most easy approach and is widely used in telephonic interviews.

They ask questions like what is faster hash map or treeMap, how to sort an array, or how to find the largest or smallest value in an array

### **4) Code refactoring/Pair programming**

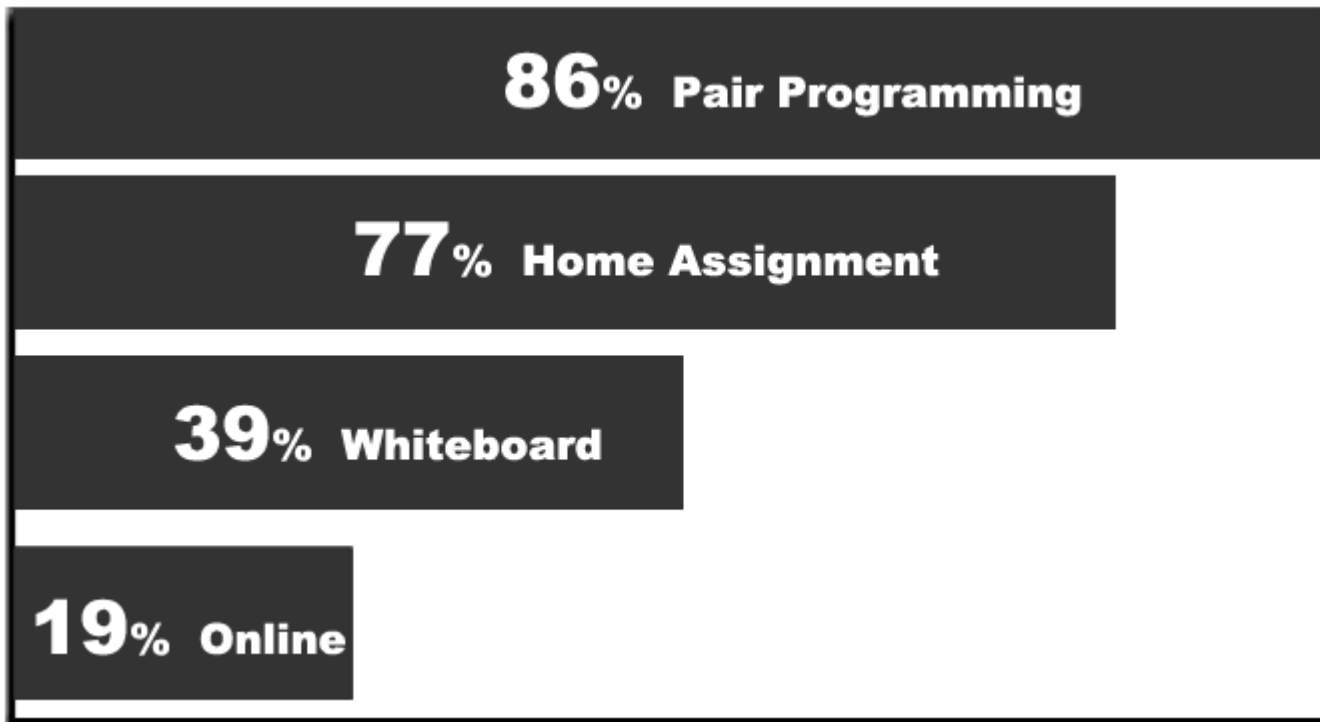
You will be presented with a semi-implemented solution and asked to complete the implementation or refactor the code.

Like whiteboard test, this part is also tough and usually one of the interviewer's sits with the candidate to discuss the solution.

Following graph shows the percentage of each evaluation method

\*Data provided by <http://stateoftechhiring.com/>

## EVALUATION METHODS



## BOTTOM LINE

Some companies (Google, Microsoft, Amazon, Facebook) use almost all of the above methods.

While many other companies use a combination of different approaches, It all depends on what kind of job you are going for and which company it is.

But you must be prepared to tackle any of these. And that is exactly, what I will teach you in this guide. So stay focused, and keep reading.

---

The process of going through various rounds of evaluation in your job hunt can be complicated.

In fact, It cannot only be difficult but:

It can be messy.

It can be lengthy.

It can be cumbersome.

You need to have patience, as you move through layers of evaluation rounds.

Sometimes the process varies over months.

But how many rounds of interviews you need to go?

It varies from company to company.

Have a look at the following graph about average interview rounds.

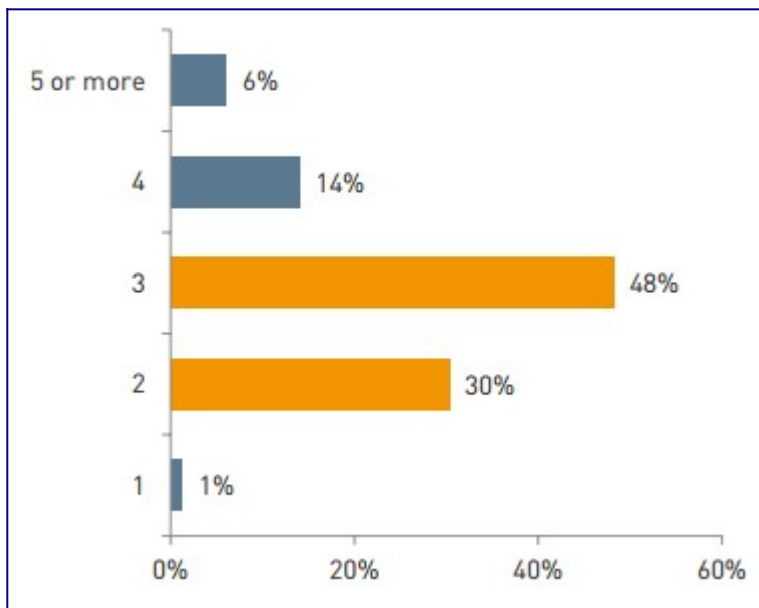


Image source: [MRINetwork](https://www.mrinetwork.com/)

**5 or More** = Average no of interviews at fortune 500 companies like [Google, Facebook, Amazon, Microsoft], With Distribution as two phone interviews and seven rounds of onsite technical interviews.

**4** = 14% candidates get job offers after four interviews.

**3** = Most job offers come after three interviews. With Distribution as one phone interview and two rounds of onsite technical interviews.

**2** = 30% candidates get job offers only after two interviews.

**1** = 1% candidates get job offers only after one interview. Which is very rare.

**BOTTOM LINE:**

You will go through multiple rounds of evaluations divided into [telephone](#) and onsite interviews.

## How To Prepare For Technical Interviews

---



# How to prepare for interview?

---

How you prepare for a technical interview will determine how well you perform during the actual meeting.

But

Before you can prepare efficiently, you need to understand that your level of preparation should depend on two factors

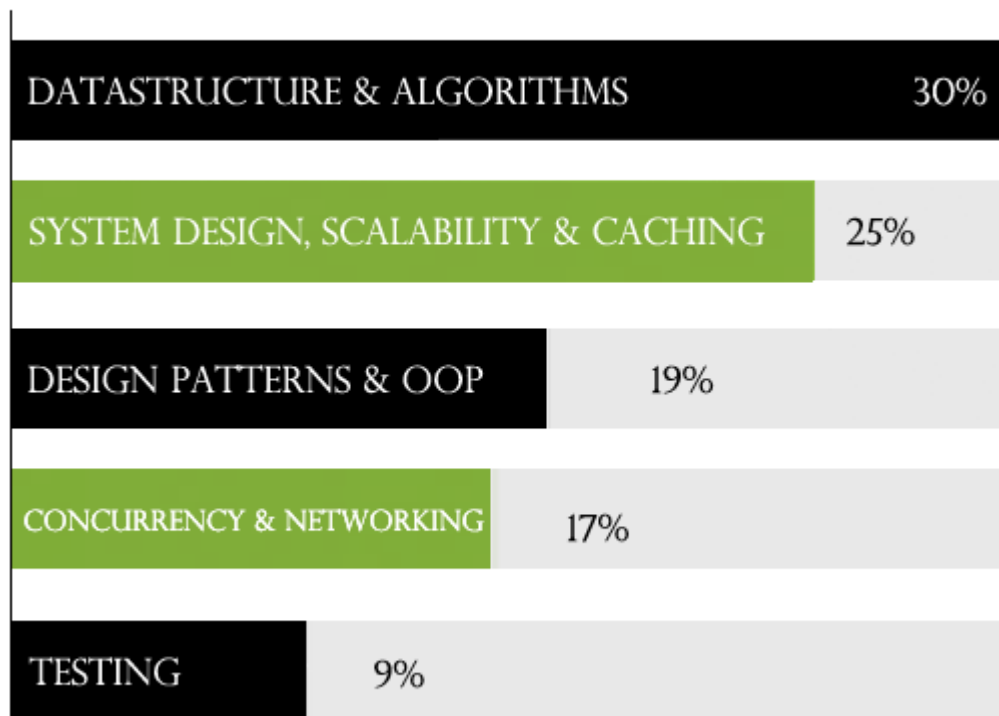
- Which job role you are preparing for
- Which company you are preparing for

For example, Technical interviews at Google and a one-person startup will not be the same.

So your preparation should not be the same as well.

In short, you need to know what is criteria and expectations for the position and company, you are going to interview with, and how much you need to prepare.

Following graph gives you an overview of the technical questions distribution in programming interviews.



In this guide, I have curated a list of interview questions, resources, and tutorials. They can be used to prepare for an interview from Google to startups.

It is divided into five sub parts

1. Do your homework
2. Emotional preparation
3. Mental preparation
4. Technical preparation
5. Understanding interviewer's perspective

Now we will go through them one by one

## 1. Do Your Homework

---



## **2. How To Prepare Mentally For The Interview**

---



## **3. How To Prepare Emotionally For The Interview**

---



## **4. How To Prepare Technically For The Interview**

---





# Arrays:

One of the most important part of technical interviews is Arrays. It is 100% guaranteed that you will be asked questions about arrays.



## Array Interview Questions:

- [100+ Array Interview Questions](#)

## Array Concepts and Types:

- [Introduction To Arrays \(video\)](#)
- [Multi dimensional Arrays \(video\)](#)
- [Dynamic Arrays \(video\)](#)
- [Jagged Arrays \(video\)](#)
- [Resizing Arrays \(video\)](#)

# LinkedList:

- [Linked list Interview Questions](#)

## Description, Types & Comparisons

- [Singly Linked Lists \(video\)](#)

## Linked List vs Arrays:

- [Core Linked Lists Vs Arrays \(video\)](#)
- [In The Real World Linked Lists Vs Arrays \(video\)](#)

Why you should avoid linked lists



## Doubly-linked List:

- [Description \(video\)](#)

## Stack:

[Stacks \(video\)](#)

[Using Stacks Last-In First-Out \(video\)](#)

## Queue:

- [Using Queues First-In First-Out\(video\)](#)
- [Queue \(video\)](#)
- [Circular buffer/FIFO](#)
- [Priority Queues \(video\)](#)

## Hash table:

Hashing with chaining



- [Table Doubling Karp-Rabin \(video\)](#)
- [Open Addressing, Cryptographic Hashing \(video\)](#)
- [PyCon 2010: The Mighty Dictionary \(video\)](#)
- [\(Advanced\) Randomization: Universal & Perfect Hashing \(video\)](#)
- [\(Advanced\) Perfect hashing \(video\)](#)

## Hash Map:

- [HashMap Interview Questions](#)

## ArrayLists:

- [Array List Interview Questions](#)

# Interfaces:

- [Interfaces Interview Questions](#)

# Abstract Classes:

- [Abstraction Interview Questions](#)

# Binary Search Trees:

Introduction to algorithms [MIT]



- [Algorithms Interview Questions](#)
- [Binary Search Tree Review \(video\)](#)
- [BST implementation - memory allocation in stack and heap](#)
- [Find min and max element in a binary search tree](#)
- [Find height of a binary tree](#)
- [Binary tree traversal - breadth-first and depth-first strategies](#)
- [Binary tree traversal: Preorder, Inorder, Postorder](#)
- [Check if a binary tree is binary search tree or not](#)
- [Delete a node from Binary Search Tree](#)
- [In order Successor in a binary search tree](#)

# Heap / Priority Queue

- [Heap](#)
- [Binary Trees \(video\)](#)
- [Basic Operations \(video\)](#)
- [Heap Sort - jumps to start \(video\)](#)
- [Building a heap \(video\)](#)
- [MIT: Heaps and Heap Sort \(video\)](#)

Binary heaps [Linear time]



# Sorting

- [Sorting Algorithm Stability](#)
- [Stability In Sorting Algorithms](#)
- [Merge Sort For Linked List](#)
- [Sedgewick - Mergesort \(5 videos\)](#)
- [1. Mergesort](#)
- [2. Bottom up Mergesort](#)
- [3. Sorting Complexity](#)
- [4. Comparators](#)
- [5. Stability](#)
- [Sedgewick - Quicksort \(4 videos\)](#)
- [1. Quicksort](#)
- [2. Selection](#)
- [3. Duplicate Keys](#)
- [4. System Sorts](#)

## UC Berkeley Lecture:

- [CS 61B Lecture 29: Sorting I \(video\)](#)
- [CS 61B Lecture 30: Sorting II \(video\)](#)
- [CS 61B Lecture 32: Sorting III \(video\)](#)
- [CS 61B Lecture 33: Sorting V \(video\)](#)
- [Bubble Sort \(video\)](#)
- [Insertion Sort \(video\)](#)
- [Merge Sort \(video\)](#)
- [Quicksort \(video\)](#)
- [Selection Sort \(video\)](#)

- **Merge sort code:**

- [Using output array \(C\)](#)
- [Using output array \(Python\)](#)
- [In-place \(C++\)](#)

- **Quick sort code:**

- [Implementation \(C\)](#)
- [Sedgewick - Radix Sorts \(6 videos\)](#)
- [1. Strings in Java](#)
- [2. Key Indexed Counting](#)
- [3. Least Significant Digit First String Radix Sort](#)
- [4. Most Significant Digit First String Radix Sort](#)
- [5. 3 Way Radix Quicksort](#)
- [6. Suffix Arrays](#)
- [Radix Sort \(video\)](#)
- [Randomization: Matrix Multiply, Quicksort, Freivalds' algorithm \(video\)](#)
- [Sorting in Linear Time \(video\)](#)

Visual Representation of Sorting Algorithms



## Graphs:

Graphs can be used to represent many problems in computer science, so this section is long, like trees and sorting were.

- [6.006 Single-Source Shortest Paths Problem \(video\)](#)
- [6.006 Dijkstra \(video\)](#)
- [6.006 Bellman-Ford \(video\)](#)

- [Aduni: Graph Algorithms I - Topological Sorting, Minimum Spanning Trees, Prim's Algorithm - Lecture 6 \(video\)](#)
- [CS 61B 2014: Weighted graphs \(video\)](#)
- [Greedy Algorithms: Minimum Spanning Tree \(video\)](#)
- [Strongly Connected Components Kosaraju's Algorithm Graph Algorithm \(video\)](#)
- [Algorithms on Graphs \(video\)](#)

## **Object Oriented Programming**

### **[OOP]:**

- [Inheritance Interview Questions](#)
- [Interfaces Interview Questions](#)
- [Abstraction Interview Questions](#)
- [Bob Martin SOLID Principles of Object Oriented and Agile Design \(video\)](#)
- [SOLID Principles \(video\)](#)
- S - [Single Responsibility Principle](#) | [Single responsibility to each Object](#)
- [more flavor](#)
- O - [Open/Closed Principal](#) | [On production level Objects are ready for extension but not for modification](#)
- [more flavor](#)
- L - [Liskov Substitution Principal](#) | [Base Class and Derived class follow 'IS A' principal](#)
- [more flavor](#)
- I - [Interface segregation principle](#) | clients should not be forced to implement interfaces they don't use
- [more flavor](#)
- [Interface Segregation Principle in 5 minutes \(video\)](#)
- D - [Dependency Inversion principle](#) | Reduce the dependency In composition of objects.
- [Why Is The Dependency Inversion Principle And Why Is It Important](#)
- [more flavor](#)
- <https://www.guru99.com/java-platform.html>

## Design Patterns:

- [Design Patterns Interview Questions](#)
- [Series of videos \(27 videos\)](#)
- [Handy reference: 101 Design Patterns & Tips for Developers](#)
- [Design patterns for humans](#)

## Caches:

The magic LRU cach



- [Implementing LRU \(video\)](#)
- [LeetCode - 146 LRU Cache \(C++\) \(video\)](#)
- [MIT 6.004 L15: The Memory Hierarchy \(video\)](#)
- [MIT 6.004 L16: Cache Issues \(video\)](#)

## Multithreading, Processes & Synchronization

- [Concurrency & Multithreading Interview Questions](#)
- [Operating Systems and System Programming \(video\)](#)
- [What Is The Difference Between A Process And A Thread?](#)
- [Paging, segmentation and virtual memory \(video\)](#)
- [Interrupts \(video\)](#)
- [Scheduling \(video\)](#)

## Software Testing:

- [Agile Software Testing with James Bach \(video\)](#)
- [Steve Freeman - Test-Driven Development](#)
- [TDD is dead. Long live testing.](#)
- [Test-Driven Web Development with Python](#)
- [Tao Of Testing](#)
- [How to write tests](#)

# String Manipulations & Search:

- [Sedgewick - Substring Search \(videos\)](#)
- [1. Introduction to Substring Search](#)
- [2. Brute-Force Substring Search](#)
- [3. Knuth-Morris Pratt](#)
- [4. Boyer-Moore](#)
- [5. Rabin-Karp](#)
- [Search pattern in text \(video\)](#)

# Unicode:

- [The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets](#)
- [What Every Programmer Absolutely, Positively Needs To Know About Encodings And Character Sets To Work With Text](#)

# Networking:

- [UDP and TCP: Comparison of Transport Protocols](#)
- [TCP/IP and the OSI Model Explained!](#)
- [Packet Transmission across the Internet. Networking & TCP/IP tutorial.](#)
- [HTTP](#)
- [SSL and HTTPS](#)
- [SSL/TLS](#)
- [HTTP 2.0](#)
- [Java - Sockets - Introduction \(video\)](#)
- [Socket Programming \(video\)](#)

# System Design, Scalability & Data Handling:

- [The System Design Primer](#)
- [How Do I Prepare To Answer Design Questions In A Technical Interview?](#)



- [8 Things You Need to Know Before a System Design Interview](#)
- [Algorithm design](#)
- [Database Normalization - 1NF, 2NF, 3NF and 4NF \(video\)](#)
- [System Design Interview -](#)
- [How to ace a systems design interview](#)
- [Numbers Everyone Should Know](#)
- [How long does it take to make a context switch?](#)
- [A plain English introduction to CAP Theorem](#)
- [Scalable Web Architecture and Distributed Systems](#)
- [Pragmatic Programming Techniques](#)
- [extra: Google Pregel Graph Processing](#)
- [Introduction to Architecting Systems for Scale](#)
- [The Importance of Algorithms](#)
- [How to Remove Duplicates in Large Datasets](#)
- [To Compress Or Not To Compress, That Was Uber's Question](#)
- [Asyncio Tarantool Queue, Get In The Queue](#)
- [When Should Approximate Query Processing Be Used](#)
- [Machine Learning Driven Programming: A New Programming For A New World](#)
- [A Patreon Architecture Short](#)
- [Design Of A Modern Cache](#)
- [Live Video Streaming At Facebook Scale](#)
- [A Beginner's Guide To Scaling To 11 Million+ Users On Amazon's AWS](#)
- [How Does The Use Of Docker Effect Latency?](#)
- [Does AMP Counter An Existential Threat To Google?](#)
- [Serverless \(very long, just need the gist\)](#)
- [The System Design Primer](#)
- [Design a CDN network](#)
- [Design a random unique ID generation system](#)
- [Design an online multiplayer card game](#)
- [Design a key-value databaseDesign a picture sharing system](#)
- [Design a recommendation system](#)
- [Design a URL-shortener system](#)
- [Design a cache system](#)

# Questions To Ask The Interviewer:

- How large is your team?
- Could you give an example of typical working day?
- What do you think are the most important qualities for someone to excel in this role?
- Where do you think the company is headed in the next 5 years?
- What are the biggest opportunities facing the company/department right now?
- What is the typical career path for someone in this role?
- How do I compare with the other candidates you've interviewed for this role?
- What are the next steps in the interview process?
- How many people work in this office/department?
- What are the prospects for growth and advancement?
- What's the most important thing I should accomplish in the first ninety days?
- What are the company's plans for growth and development?
- What type of background do you feel would be best suited for success in this position?
- How long is the average tenure of an employee?
- Where would the company like to be in five years?
- Who would I be reporting to? Are those three people on the same team or on different teams? What's the pecking order?
- How has this position evolved?
- Beyond the hard skills required to successfully perform this job, what soft skills would serve the company and position best?
- Can you give me an example of how I would collaborate with my manager?
- Can you tell me what steps need to be completed before your company can generate an offer?

- How would you score the company on living up to its core values? What's the one thing you're working to improve?
- How do you help your team grow professionally?
- Is this a new position? If not, why did the person before me leave this role?
- Will I have an opportunity to meet those who would be part of my staff (or my manager) during the interview process?
- What's your staff turnover rate, and what are you doing to reduce it?
- What is the single largest problem facing your staff and would I be in a position to help you solve this problem?

## **Things You Need To Prepare Before Interview:**

There are certain things you need to prepare before you go for the interview.

### **- Go Through Your CV**

Remember to go through your CV before the interview. Sometime candidate's send targeted CV's for each job. And they forget about the details during the interview.

Most probably interviewer's will have a copy of your CV. They will use it to ask you specific questions regarding the skills, tools, and projects you have described in it. It is also highly recommended that you keep extra copies of your CV with you.

### **- Prepare References**

You need to have at least 2-3 references before the interview. Talk with the people you plan to use as your references. Tell them for which job you are planning to use them.

Most important have important details (Name, Work place, role, Linked in profile, Mobile no. and email id ) of references with you.

## **- Brush Up Your Online Profiles**

According to a recent research. around 80 % of recruiters check the social profiles of the candidates.

In another study done by career builder they found out that recruiters will disqualify candidates if they find evidence of the following on their social media profiles:

- Provocative or Inappropriate Content - 46%
- Alcohol and Drugs - 43%
- Bigoted Content (Race, Religion, Gender, etc.) - 33%
- Bad-mouthing Previous Company - 31%
- Poor Communication Skills - 29%(*Careerbuilder*)

So bottom line is remember to update your profiles[Linked in,Facebook, Github,Twitter] before the interview.

## **- Practice Introducing Yourself**

How you introduce yourself to the interviewers is the single most important predictor of your success. So you need to practice it before the interview.



## **- Practice Making A Great First Impression**



Following guide will teach you how to make a [great first impression](#).

## **- Physical Preparation Before Interview**

Physical preparation is as important as technical preparation. You need to plan and prepare the following things before the interview.

- **Dressing:** Decide in advance what kind of clothing you will wear on the interview day. Try to choose professional dressing. And most important thing it should be clean and pressed.

- **Hygiene:** Your nail and hairs should be clipped. Remember to take a shower, before you leave home for interview. And use your favorite perfume.

## 5. How To Understand Interviewer's Perspective

---



The greatest power you could have in interviews would come neither from limitless resources nor it would arrive by even knowing all the answers.

It would come from explicit knowledge about the expectations of the interviewer.

Given that knowledge, you could anticipate the needs, requirements and thinking processes of the interviewer.

Try to think of yourself as the one, who is going to conduct the interview.

What would you expect?

What kind of answers will you be looking?

What possible questions you will ask?

How will you make sure that you are hiring the right candidate?

Thinking in this way will open up new horizons. You will enable yourself to realize, how the interviewers think, and what they expect.

It will make you cater your responses and answers accordingly.

## **Bonus Video: The Winning Mentality for Interviews**

---



### **BOTTOM LINE:**

Thinking from the interviewer's perspective will give you ideas about what they are expecting.

It will allow you to prepare and act according to their expectations.

# Cracking The Programming Interviews:

I once had an interviewer.

Who would push his glasses down on his nose.

While I was explaining my solution, and he would look at me above the glasses.

Like my solution was not worth focusing. Like I was an idiot.

He hated me.

And then he Rejected me.

Rejection is hard to deal.

But

If you learn the right tricks.

The tricks to solve and answer all the interview questions.

Then you don't have to deal with rejection.



This is the section where i will show you how to crack the technical interviews.

This part is divided into following sub sections.

- How to answer technical questions

- How to solve algorithms on white board

- How to refactor code in interviews

- How to solve programming assignment and defend it in interview

- What to do after the interview

We will go through these points step by step.

## A Disclaimer

---

Before we dig deep into details you need to understand and accept that your performance in various interviews will not be the same.

It depends on various factors, like nature of the questions, interactions with the interviewers and level of your emotional and mental preparation etc.

But more interviews you have better you will do. Every experience in itself will teach you a valuable lesson and eventually you will become very good.

## 1. How To Answer Technical Questions

---

This is one of the most fundamental skill you need to crack the technical interviews. 200 other candidates competing for the same job will answer the same questions.



So the question is how you can differentiate your answers from the others?

The answer lies in the onion method

## 1.1- The Onion Method

---



The best way to succeed in interview is to answer the questions.

The best way to answer the questions is by using a structured approach.

The best way to use a structured approach is by knowing what when and how to reply.

The best way to know what, when and how to reply is by using the onion method.

And The onion method is

"During technical interviews, do not answer a question randomly. Instead reveal your knowledge layer by layer.

Just like you are peeling layers of an onion. Where each layer will reveal something new and important.

Starting from the most basic layers and moving towards deeper ones, until it reaches to the core of the concept."

This is the most powerful way to answer technical questions.

Because often times we are so stuck in our own mind that we forget to realize that interviewer's are judging us based on our answers.

They want to know how well we understand a concept, how deep we know about it and how we can apply this knowledge to fix the problems they are facing.

And by reveling our knowledge in no particular order with timidity and complexity, we create more doubts then clarifications.

Where as using the onion method will make your answers look structured and professional making you sound like an expert on the topic.

### **BOTTOM LINE:**

By using a structured and disciplined approach you will sound like an expert. Better then the other's more appealing to the interviewers, and increasing your chances of success by 10 folds.

## **2. How To Solve Algorithms On Whiteboard**

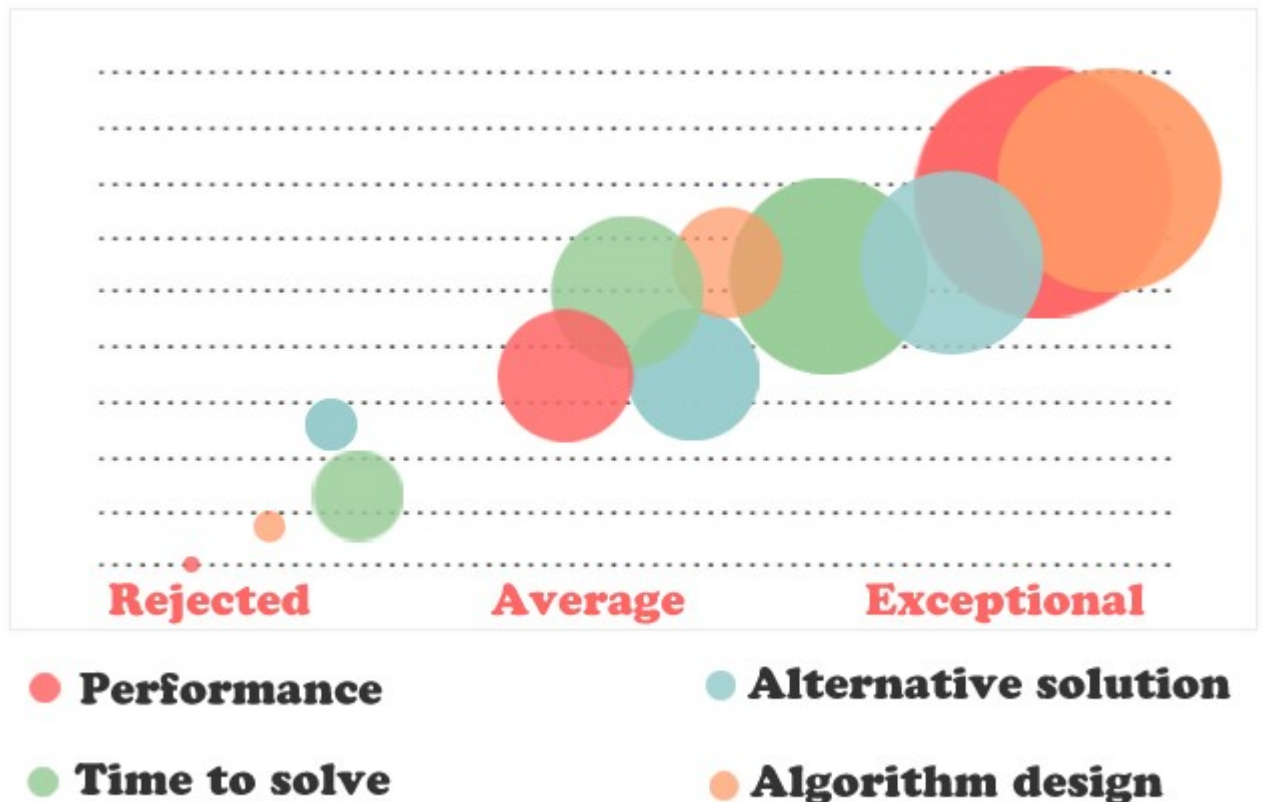
---



This is one of the most common and often difficult phase of technical interviews.

Most of the candidates are unable to get past this successfully.

Following graph shows what four elements interviewer's are interested in and how candidates perform them.



Now here is how you should approach it.

## 2.1- Understand And Repeat The Question

---

You will be asked to solve a problem or design an algorithm on the white board. The first thing you need to do is to pay close attention to the question.

Ask the interviewers if something is not clear. Make sure you understood the core of question.

Once you are comfortable with your understanding next thing you have to do is to repeat the question loudly and clearly.

This will ensure to the interviewer that you have understood their question.

## 2.2- Explain Your Approach/s To Solve The Questions

---

The No. 1 mistake 95% of candidates make is that they immediately jump to solve the solution on the white board.

This is the wrong approach. Remember that you are trying to win over the interviewer with your solution.

The correct way to do is by explaining them in what possible ways the problem can be solved. Why and by which approach you will solve the problem

There are two benefits for doing this

**FIRST:** If there is something wrong with your approach then interviewer's will provide you feedback about it or clarify the questions again.

**SECOND:** Interviewer's will get a great first impression, and they just want to see the solution in action.

## **2.3- Design A Pseudo Code**

---

Until now you have the solution in hand, but still you should resist the urge to directly solve the problem. Instead you must design the pseudo code for your solution.

Understand that most of the problems will be algorithmic in nature, and the best way to solve them is by designing an overall outline using pseudo code.

Again by doing this you will get immediate feedback for your solution and if there is some gaps, you can adjust them even before you jump into the full implementation.

## **2.4- Write The Full Implementation**

---

You have already explained the solution, Great.

You have shown your pseudo code to the interviewer, Fantastic. And finally now is the time to go with the full implementation on the white board.

Much of the anxiety is already taken away from you, since in your mind you have the solution and a skeleton to follow.

Now your 100% focus should be on the optimal implementation of your pseudo code.

As you will be solving the problem interviewer's will ask you a lot of questions.

These will be about your design considerations, selection of data structures, and most importantly the performance of your algorithm/solution.

They will also discuss alternatives or better ways to solve the problem.

The best way to respond to interviewers questions is by LOA(Listen- Observe- Adjust) loop which is explained below

## 2.5- Listen- Observe- Adjust loop

---



**LISTEN:** The most important skill you need to have is your ability to listen and listen well.

Throughout the interview, the interviewers will be talking, discussing or giving feedback to you regarding your solution.

In order to increase your chances by 1000% all you have to do is to listen to them very carefully.

**OBSERVE:** You should not only listen but observe. You need to train yourself to read the signals they are giving to you.

Words like "this is it", "are you sure", "But", "consider a situation like",

"It can be like this or this", "What if " , "OK", all of these means there is something missing in your solution

Or they want you to consider the other situations as well, or they are keen to find out whether you fully understood the implications of your solution or not.

Similarly If they are saying hmm this solution is good but what if its size is huge etc.

This means that they are keen in listening about the performance of the solution.

**ADJUST:** Once you listened carefully and have observed what the interviewers want to see or listen, next step is to adjust your answer or solution according to the feedback.

The best way to do this is by repeating what they are saying and then explaining how would you fix/solve this.

And once they seems to agree then you can actually proceed to adjust your solution.

## **2.6- Use Cosmetic Beauty To Standout From 100 Other Candidates**

---

Remember that you will not be the only one interviewing for the job. Interviewer's will usually see the answers to the same questions over and over again.

Then the most interesting question will be how you can differentiate your solution from other candidates?

**Are there any factors which will make interviewer's to remember your solution better than the others ?**

Simple answer is Yes, and it lies in the cosmetic beauty of your code. You might be wondering

**What do you mean by cosmetic beauty of code?**

It means how well your code is groomed, there are certain factors that can make your code standout.

Like naming of variables, classes and objects, Selection of most optimized data structure, How well you refactor your code.

Are you utilizing any language specific features. Usage of design patterns, Unit testing and simplicity, maintainability and scalability of the code are among the most important factors.

### **BOTTOM LINE:**

You need to listen, repeat, design an algorithm and then implement the solution on the white board.

After you have solved the problem then listen to the feedback, observe the signals interviewers are giving you and then adjust your answer accordingly.

## **3. How To Refactor Code in Interviews**

---

Code refactoring is another popular interview tactic. You will be presented with semi implemented code and will be asked to refactor it.

Here is how to deal with this part.

### **3.1- Scan the code to Identify the problem**

The first thing you need to do is to identify the pattern in the code. Identification means that you need to quickly scan the program and find out following things

- What is missing in the code e.g if there are multiple classes with same methods. This would mean that you need to introduce inheritance
- Are there any unit test cases + if yes are they broken or correct
- Are there multiple calls to same methods.Can you use recursion to fix this?
- Is there some wrong algorithm implementation
- How the classes, variables and methods are named
- Are there any variations from the standard coding practices
- Are there any circular links between classes

## **3.2- How to fix the missing/wrong code**

Once you have figured out what's wrong then the next step would be to think about the solution. Not one but all possibilities, performance and other implications as well.

Create a design in your head and then before you refactor the code, explain it to the interviewer what you have found and how you are going to fix it.

## **3.3- How to refactor the code**

Once you know what's wrong and have a plan to fix it, Then proceed with your fix. But it is very important that you get the interviewer engaged with you while you are doing this and you can achieve it in the following way.

**FIRST:** Read out aloud your internal thinking process for example say

"Ok, these three classes can be refactored by using the inheritance and now i will create one parent class with these methods named like this and make other three classes to inherit these methods."



**SECOND:** Remember to use the coding best practices, make your code looks beautiful. Always test your refactored code with unit tests and explain the performance of your solution etc.

**THIRD:** Always be ready to explain/discuss the other alternative approaches the code could have been refactored and how those approaches compares with your chosen approach. What are the advantages and disadvantages of your own approach and how it could be made even more better.

### **BOTTOM LINE:**

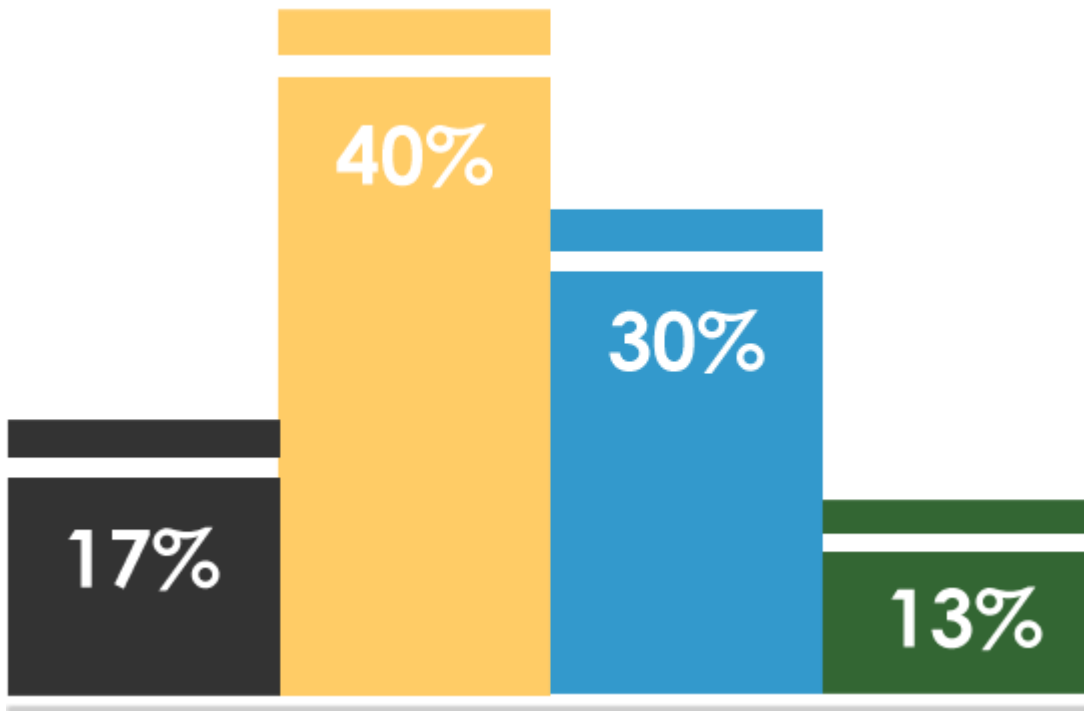
Most important thing is to identify what is wrong/missing in the code. Once you know it then think about the solution, discuss it with the interviewer and finally fix the problem.

## **4. How To Solve Programming Assignments**

---

There are four elements in solving technical assignments and then defending it in interviews

But before we look into these. Let's have a look at how does companies evaluate the technical solution.



40 % Depth & Efficiency of solution

27 % Propose an alternative solution

17 % Coding style, formatting, readability

13 % Time to solve the assignment

Here are the four elements

## 4.1- Keep It Simple

---

Sometime candidates refer to larger issues, a great design, or a very complicated solution.

They go for the big picture when simple but concrete solution would have much more appeal.

What they do not realize is that even the most complex problem can be broken down and implemented as simple and elegant solutions.

Which has a much larger appeal for the interviewer. If you make no appeal to him, he merely sees you as desperate or, at best, a waste of time.

The key is to, not only solving the problem, but how well you have solved it. What choices you have made in designing your solution.

Which common pitfalls you have avoided. How well it is tested. How you have thought about the performance, maintainability and readability of your design.

When you are designing any problem keep in mind the following things

- Use standard naming conventions
- If possible stick to clean code principles
- Make sure there are no cyclic dependencies among the classes
- When solving algorithms, always focus much more on simplicity and the performance of your solutions
- At all cost avoid any complex or fancy solution
- Always test your solution thoroughly

## **4.2- Know When To Stop**

---

In designing your solution never go beyond the goal you had aimed for, and by going too far.

You make more mistake and you end up revealing your weak points.

Solving complex problems has its own merits. Those who succeed at this are the ones, who knows what to achieve and when to stop. Going too far can set you off balance.

The lesson is simple: Keep it nice and clean. Aim for certain depth and know when enough is enough

## **4.3- Know About The Pitfalls**

---

In interviews defending your solution is more important than its implementation. Never make it too complex, that you neglect to understand it yourself.

The greatest skill is the ability to solve a solution, be aware of its weaknesses and strengths and be able to implement other alternative solutions.

Never assume that your design and solution is perfect. The interviewer will ask you a lot of questions about your solutions, the design choices you have made.

They will particularly look for what is missing, and will test your knowledge about the solution you have made.

Be ready to know what you have made in deep details, Be ready to explain why you have chosen to implement certain thing in this way.

## **4.4- Know How To Defend**

---

In the heat of defense, arrogance and overconfidence can push you past your limits and often you end up offending the interviewer.

Never try to go too far with your contradiction.

The problem in trying to prove a point through argument is that in the end you can never be certain how it affects the interviewer.

They may appear to agree with you politely, but inside they may resent you.

Or

perhaps something you said inadvertently even offended them. Words have that insidious ability to be interpreted according to the other person's mood and insecurities.

Even the best argument has no solid foundation.

You need to understand that the interviewer has his own views and preferences about how things should be solved and designed.

If you can resonate with the interviewer you can go farther than other candidates. Know about the preferences

and tastes of the interviewer and try to talk from that point.

It will make him feel confident that you are much more aligned with his style of working



### **BOTTOM LINE:**

Message is simple, keep your solution simple and short. Know about all the pitfalls and alternative approaches. And defend it well during the interview.

## **5. What To Do Right After The Interview**

---

I was feeling dumb, idiotic and retarded. For a few moments I wished that I could escape or hide somewhere.

They gave me many hints and ideas but I was unable to resolve the issues.

But then I did something right after I went home on that day and guess what next day I received an offer letter.

**What was it ?**

It was no magic, It happens many times that candidates are unable to solve or explain an issue during the interviews and then they feel that's it, Game over. But it's not true.

What I did on that day was that the moment I reached home I searched on Google about that problem.

And then I made a small nice implementation of the algorithm and sent the program to the interviewer.

Explaining what I was not able to explain in interview and then asked them to have a look at the solution.

I also wrote about the explanations and ideas about some of the topics we discussed during the interviews.

And also thanked them for such a deep and good discussions about the various topics.

It clicked and to my amazement they actually liked it and decided to offer me the job.

Since then i have used this trick after every technical interview. And it has always worked wonders for me and you can too apply this and see the results by yourself.



## **CHAPTER 7:**

### **What To Do If You Fail An Interview:**

---

Failure is not good.

Failure is the worst thing possible in this world. You feel sick and ashamed.

There is absolutely nothing good about failure.

And there's nothing you can do to go to the past and change the failure.

But there is one thing you can and you should definitely do to learn from the failure.

And it will make all the difference in your future success.

It is called [The Interview sex](#).



# **Programming Interview Resources**



In your quest to get ready for your next job interview:

It's easy to get lost or waste a lot of time, energy and resources in the wrong direction.

In order to maximize your chances of success You have to get yourself ready in a different way.

Your chances will only increase by preparing in a smart and efficient way That differentiates you from the crowd.

In order to achieve all of this you need a lot of help.

And



In this chapter i have curated a list of "THE BEST" resources.

■

Consider this list as a manual, Whenever you feel lost or unsure of something consult a relevant resource from this list.

Believe it or not you will increase your probability of success by 1000%.

That's "THE ONLY" way to beat the competition and get ahead in your career.

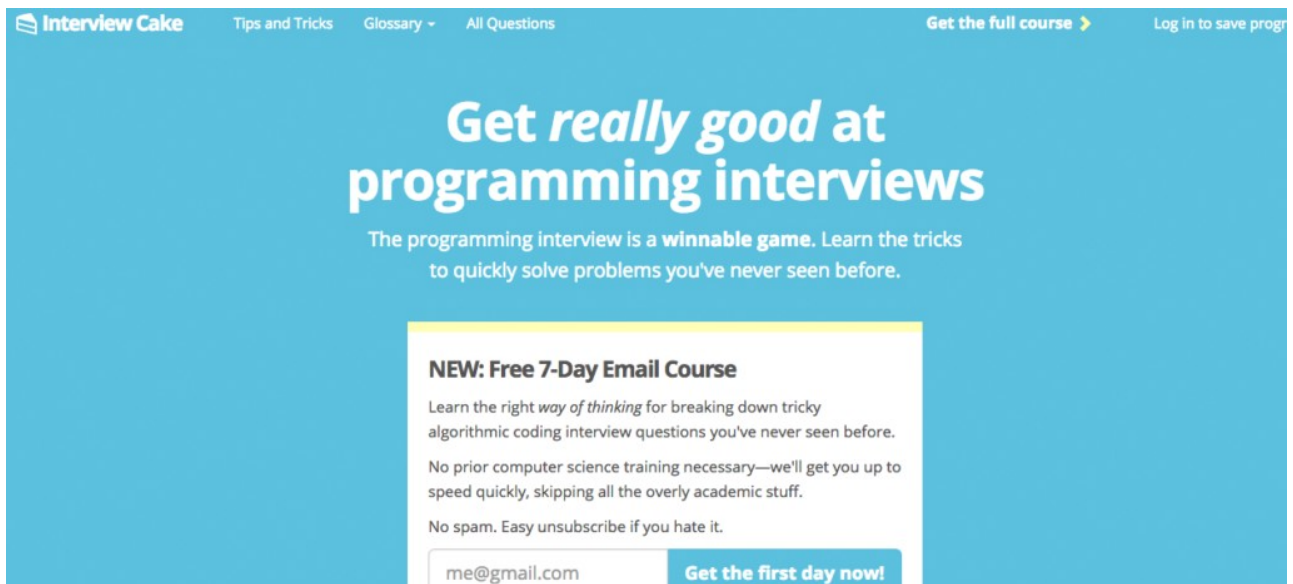
## Interview Preparation Tools:

---

Here is list of best technical interview resources

### 1. Interview Cake

Interview cake is the No. 1 resource for candidates, preparing for Technical interviews at companies like Google, Facebook, Microsoft and Amazon.



Interview Cake   Tips and Tricks   Glossary   All Questions   Get the full course >   Log in to save progress

# Get really good at programming interviews

The programming interview is a **winnable game**. Learn the tricks to quickly solve problems you've never seen before.

**NEW: Free 7-Day Email Course**

Learn the right way of *thinking* for breaking down tricky algorithmic coding interview questions you've never seen before.

No prior computer science training necessary—we'll get you up to speed quickly, skipping all the overly academic stuff.

No spam. Easy unsubscribe if you hate it.

me@gmail.com   **Get the first day now!**

## About The Founder:

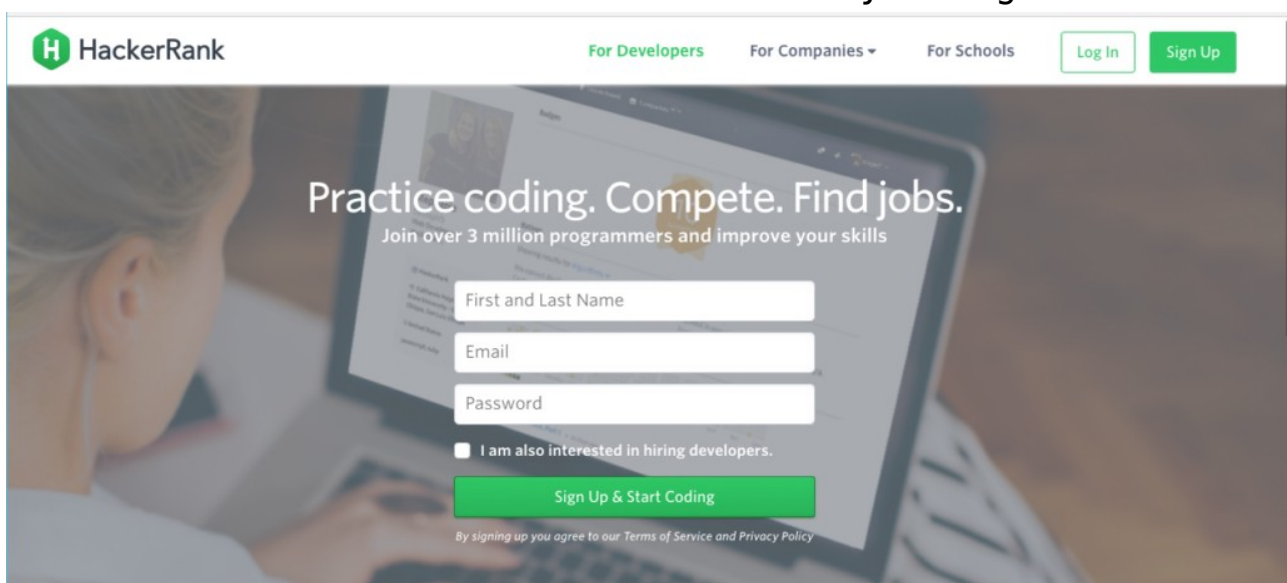
Interview Cake is founded by Parker Phinney. Parker have worked as an engineer at Google and a handful of startups.

**Link:** <https://www.interviewcake.com>

---

## 2. Hacker Rank

Hacker Rank is a place where developers can practice programming problems and compete with other candidates. Based on the rank of the candidate they can get hired.



HackerRank   For Developers   For Companies   For Schools   Log In   Sign Up

# Practice coding. Compete. Find jobs.

Join over 3 million programmers and improve your skills

First and Last Name

Email

Password

☐ I am also interested in hiring developers.

**Sign Up & Start Coding**

By signing up you agree to our [Terms of Service](#) and [Privacy Policy](#)

## About The Founder:

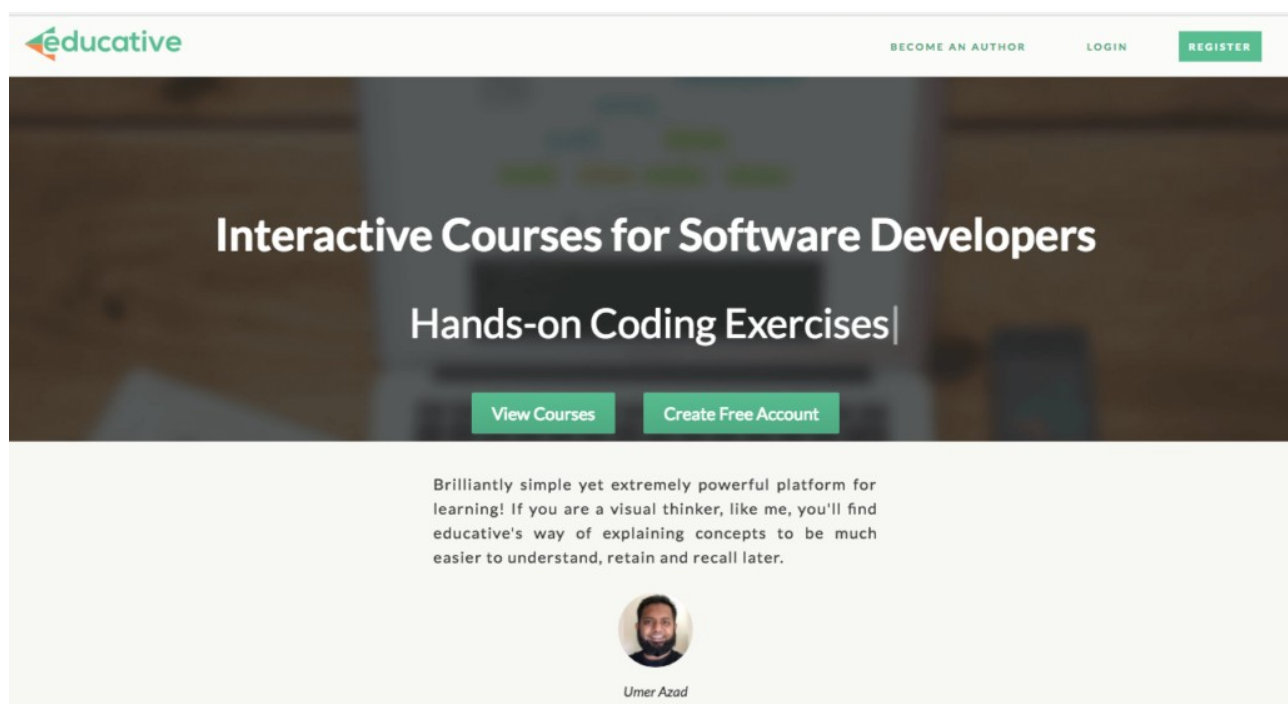
Vivek Ravisankar and Hari Karunanidhi founded Hacker Rank in 2008. When they were tired of spending too much time in interviewing candidates.

**Link:** <https://www.hackerrank.com/>

---

## 3. Educative.io

Educative.io offers a huge library of Interactive Courses for Software Developers. You can find an online course for almost any topic in programming languages and Frameworks



## About The Founder:

Educative is founded by two brothers Fahim Ul Haq and Naeem Ul Haq both of them have extensive experience in working as software developers at fortune 500 companies.

**Link:** <https://www.educative.io/>

---

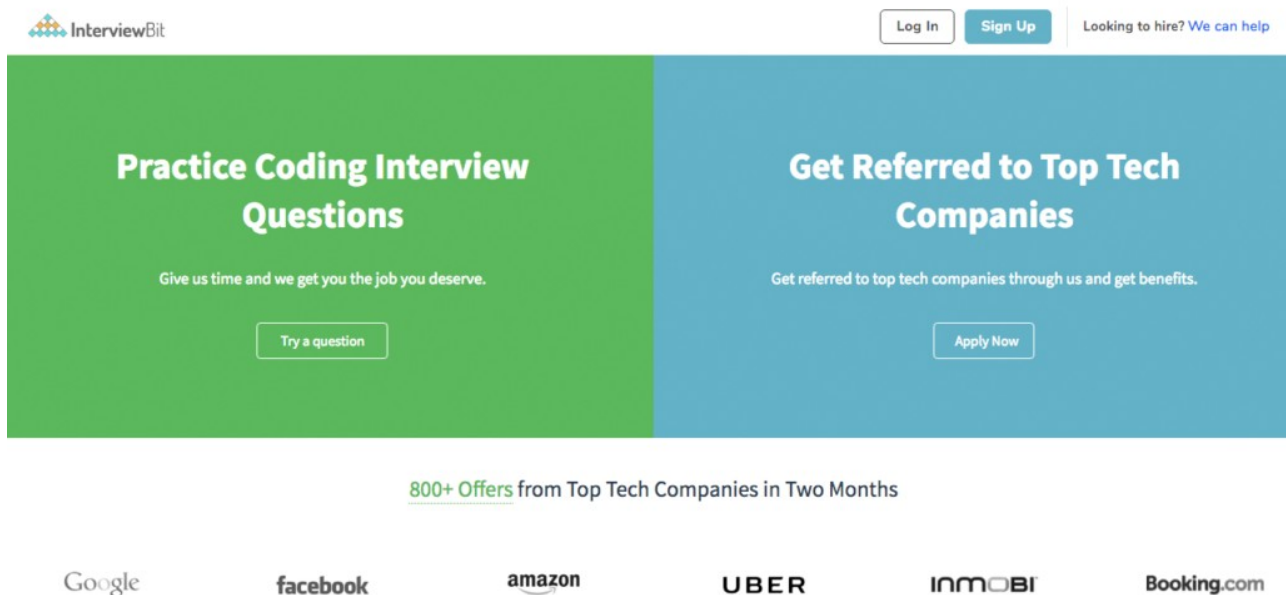
## 4. Interview Bit

Interview bit is a site which gamifies the experience of practicing for technical interviews.

Best thing about interview bit, is its vast collection of sample problems and company specific Interview questions e.g Google interview questions, Facebook interview questions.

Interview bit also offers a service where candidates can get referred to top companies through their vast network.

Its worth checking Interview bit.



### About The Founder:

Interview bit is founded by Anshuman Singh who has vast experience of interviewing candidates at Facebook. He also leads the course design at Interview bit.

Abhimanyu Saxena is co founder of Interview bit and has worked at companies like fab.com. He oversees the technical strategy of the company.

**Link:** <https://www.interviewbit.com/>

---

## 5. Interview Kick Start

InterviewKickStart offers a comprehensive job interview training both online and onsite. It had comprehensive curriculum to get you ready for the real interviews.

The best thing about interview kick start is that It offers training with mock interviews on white-boarding, pair-programming & timed-tests.

It gives immediate feedback to candidates and help them improve their ability to solve complex problems

**Interview  
Kickstart**  
No shortcuts to places worth going to.

[About Us](#)  
[FAQs](#)  
[operations@interviewkickstart.com](mailto:operations@interviewkickstart.com)

### Nail your next coding interview

Filling fast, for:  
**May 2018 (Onsite / Remote)**  
[March & April cohorts: Full, w/waitlist]  
[All prior cohorts: Oversubscribed]

First name, Last name	<b>Get an invite</b>
Phone Number	
Email	

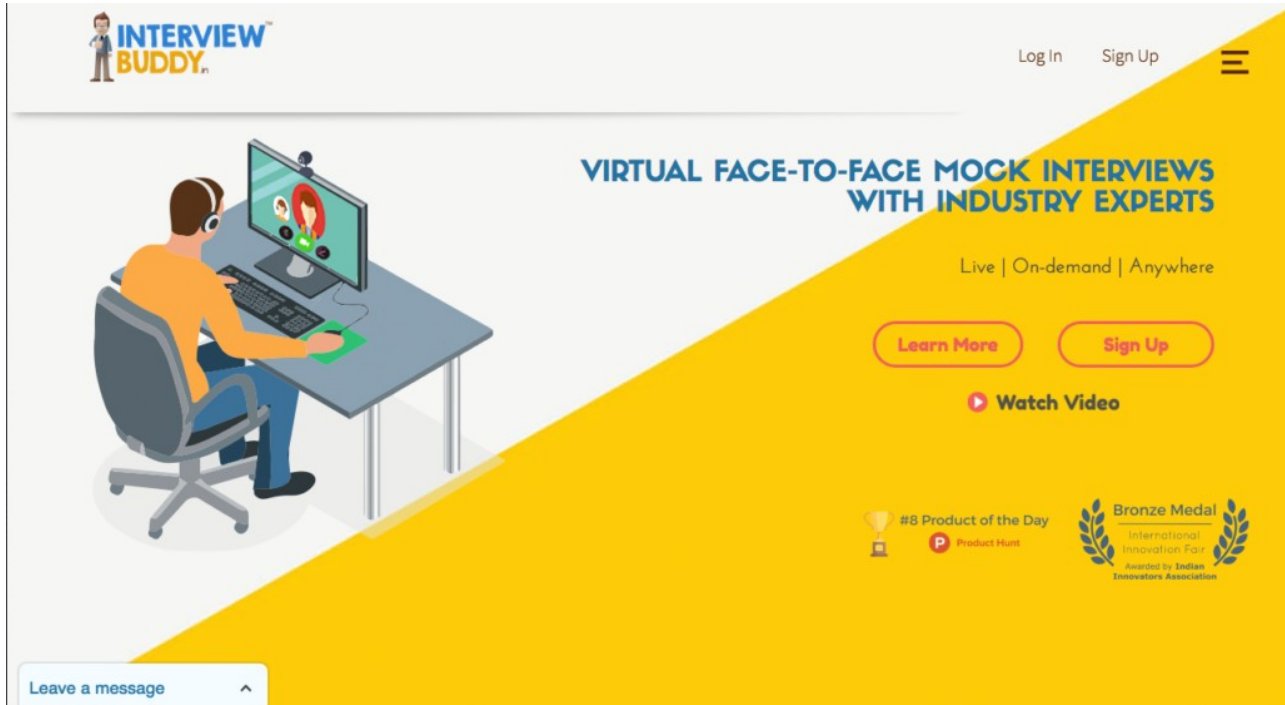
### About the Founder:

Soham Mehta has worked at Box as Director Of Engineering. The experiences at Box led him to start interview kick start so that he can teach other people what he has learned in doing so many technical interviews

**Link:** <http://www.interviewkickstart.com/>

## 6. Interview Buddy

Interview buddy offers virtual Face-To-Face mock interviews with industry experts to help you train & prepare for job interviews.

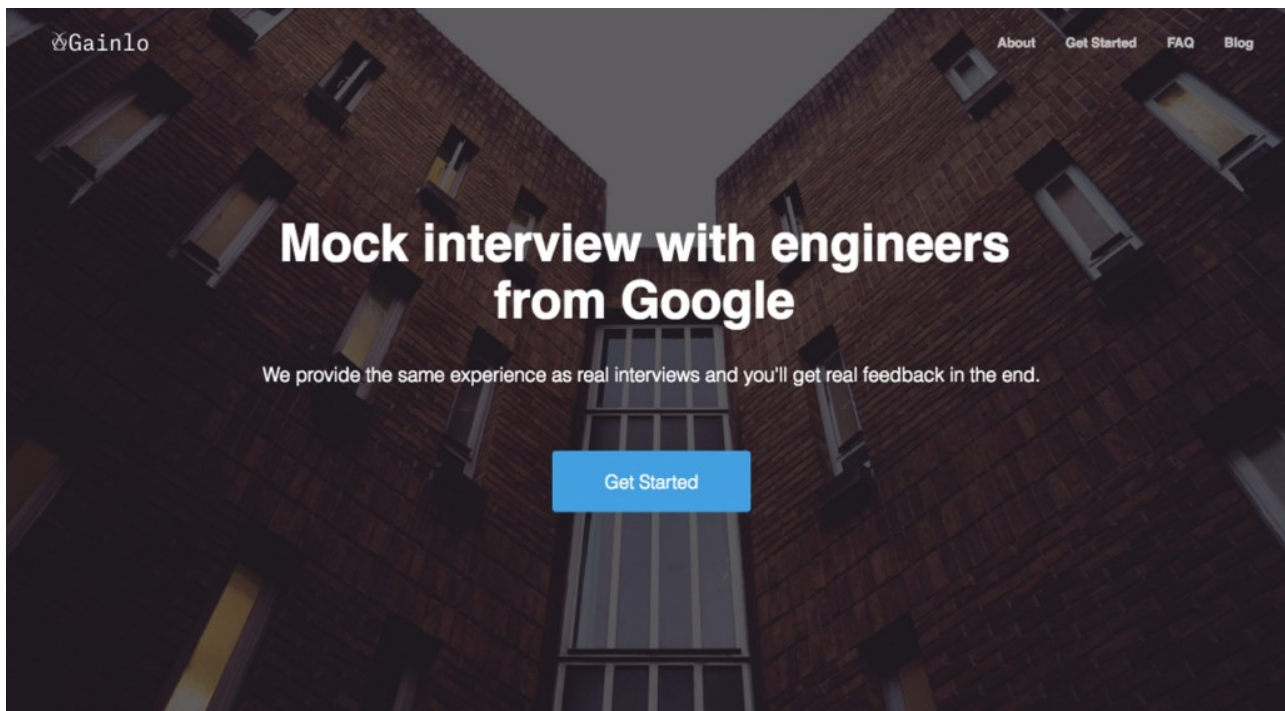


**Link:** <https://interviewbuddy.in/>

---

## 7. Gainlo

Gainlo offers mock interview with engineers from Google, Facebook and Microsoft. It strives to offer a real interview experience to candidates and offers a near real feedback.



**Link:** <http://www.gainlo.co/>

## 8. Devskiller

<https://devskiller.com/> [[Read My Review Here](#)]

---

# Java Interview Questions:

---

## 1. Journal Dev

Journal Dev is my personal favorite website when it comes to Java. It has a wide range of interview questions. There are also a lot of high quality and in depth articles on different areas in Java.

Each Article covers all the elements of the topic from basics to advanced knowledge. And The best thing is there are always code examples.

I am using Journal Dev from almost 4 years.

**Link:** <https://www.journaldev.com/jav>



---

## 2. Java Revisited

Java revisited is another high quality Java blog. It has a huge list of Java interview Questions and general programming tutorials.

I have always consulted this website before any major interview. I recommend anyone to consult this website for Java tutorial.

**Link:** <http://javarevisited.blogspot.dk/> , <http://www.java67.com/>

---

## 3. Baeldung

If you are looking for anything related to Java Spring framework. Then look no further then Baeldung. This is in fact the best blog in the world to learn about spring and spring security.

**Link:** <http://www.baeldung.com/>

---

## 4. Java2Novice

Java2Novice is another high quality Java blog. It contains java interview programs. The implementation of programs are easy to understand.

**Link:** <http://www.java2novice.com/>

---

## 5. Geeks For Geeks

This is another very famous java blog. It has almost a cult following. More and more people are using it everyday to study about algorithms and Data structures.

**Link:** <https://www.geeksforgeeks.org/>

---



# **.Net (C#) Interview Questions**

---

- [52 .Net Interview Questions And Answers](#)
- [C# Programming Interview Questions](#)
- [C# Advanced Interview Questions](#)

## **PHP Interview Questions**

---

- [PHP Interview Questions](#)
- [Top 100 Interview Questions](#)
- [Advanced php Interview Questions](#)

## **Angular JS Interview Questions**

---

- [AngularJS Interview Questions](#)

## **Spring Interview Questions**

---

- [Spring Interview Questions](#)

## **DevOps Interview Questions**

---

- [DevOps Interview Questions](#)

## **Others**

---

- [Advanced Interview Questions](#)
- [Algorithm Interview Questions](#)
- [Java algorithms interview questions](#)
- <http://www.codespaghetti.com/interview-questions/>

# Non-technical Interview Questions

---

- [Top 15 Behavioral Interview Questions](#)
- [Behavioral Interview Questions And Answers](#)
- [45 Behavioral Interview Questions](#)
- [Behavioral interview Question](#)

## Technical interview Guides

---

- [How To Avoid Interview Rejection](#)
- [How To Make Great First Impression](#)
- [Top Technical Interview Questions](#)
- [How to Answer Most Important Interview Questions](#)
- [Technical Interview Questions And Answers](#)
- [How To Ace Technical Interviews](#)
- [How To Prepare And Ace Technical Interviews](#)

## Company specific Guides and Questions

---

### Google interview

- [Google's Hardest Interview Questions](#)
- [Google Interview University](#)
- [Google Interview Questions](#)

### Facebook interview

- [Facebook Interview Questions](#)
- [Facebook Interview Problems](#)
- [10 Things To Know Before Facebook Interview](#)

## Programming Interview Books

---

- [Cracking The Coding Interviews](#)
- [Coding Interview Questions](#)
- [Programming Interviews Exposed](#)

## Interview Psychology & Hacks

---

- [How To Avoid Interview Rejection](#)
- [How To Have Interview Sex And get the Job](#)
- [5 Questions To Ask In An Interview \(Based On Psychology\)](#)
- [Mastering The Psychology of Interviews](#)
- [7 Psychology Tricks To Nail Your Next Interview](#)
- [Psychological Tricks For Interviews](#)
- [A Guide To Interviewing](#)
- [How To Prepare Mentally For Interview](#)

## Interview Success Stories

---

- <https://www.quora.com/What-are-some-of-the-funniest-Google-onsite-interview-stories>
- <https://medium.freecodecamp.org/software-engineering-interviews-744380f4f2af>
- <https://medium.com/swlh/this-is-what-its-like-to-fail-your-interviews-at-google-83c772e6e654>
- <https://www.glassdoor.com/blog/interview-horror-stories/>

## Cool Bonus:

New Bonus PDF

**Download a free PDF version of this guide. It contains all the links, tips and resources explained here... Plus its print friendly.**



Unlock your Interview potential

9346 people have already downloaded it

## **Summary:**

# HOW TO ACE PROGRAMMING INTERVIEWS

## HOW TO FAIL INTERVIEWS

The easiest way to succeed interviews is to know how to fail an interview so you can avoid those mistakes



- Go unprepared
- Be over-confident
- No knowledge about the job
- Bad code quality

## SUCCESS FACTORS

- These are most important success factors
- Internal State of mind
- Candidate Likability
- Ability to code
- Personality



## INTERVIEW INTELLIGENCE



Remember that your every word will have an impact on the interviewer. Learn to focus deeply on the interviewers, reading their behavior and reactions and

["How to Ace technical interviews"](#)

[Click to Tweet](#)

## About The Author:

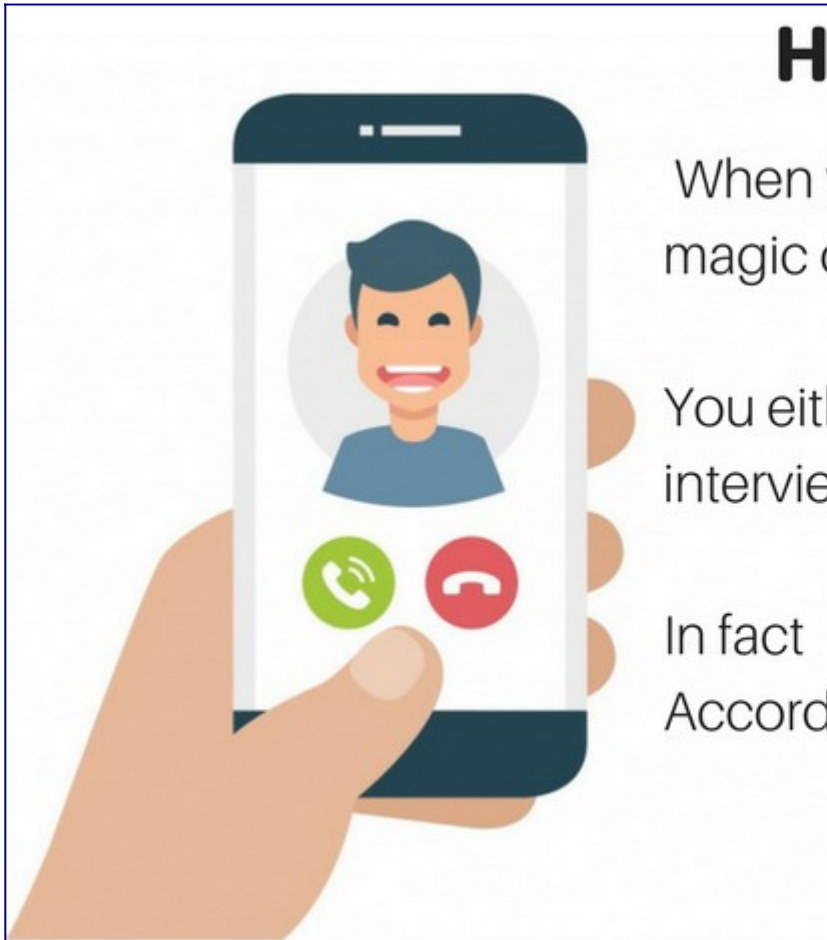




If you have any questions or need some help with your upcoming interview, feel free to [contact me](#). I will make sure to help you in the best possible way.

All the materials and services on this site are 100% free. Just to help someone who might be stuck as I once was.

## Recommended For You



### How to ace phone

When you are doing a phone interview, there is no magic or luck involved.

You either fail, or proceed to the next round of interviews.

In fact  
According to a new research

**Continue**



## Questions for 100% guaranteed programming interview



Developers that succeed with tech interviews do these things very well:

First, they find out what kind of questions are asked in interviews.

Second, they put 100% of their effort into preparing those questions.



## References:

- <http://blog.triplebyte.com/how-to-pass-a-programming-interview>
- <https://www.themuse.com/advice/5-smart-moves-to-make-in-a-technical-interview-that-have-nothing-to-do-with-coding>
- [https://en.wikipedia.org/wiki/Miyamoto\\_Musashi](https://en.wikipedia.org/wiki/Miyamoto_Musashi)
- <https://www.thebalance.com/best-impression-at-an-interview-2060572>
- <https://www.youtube.com/watch?v=aClxtDcdpsQ>
- <http://www.tsearch.com/news-advice/2011/12/how-to-succeed-in-a-technical-interview/>

Can I ask you a small favour?

<http://www.codespaghetti.com/interview-success/>