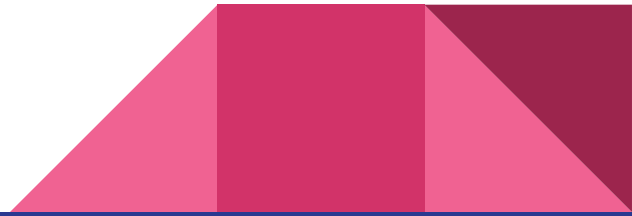
How to crack FAANG

By Rishabh Garg

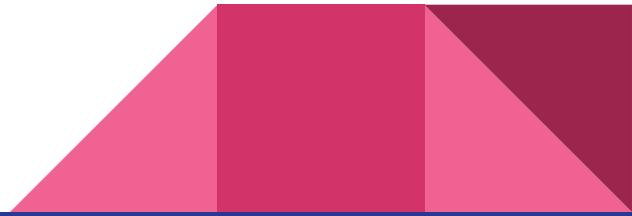
About me

- Currently working at Meta in Ads Tech
- Previously employed with Amazon
- Co-founded my Startup - Insterviews (2018-20)
- Have worked both on 0-1 startups and MNCs
- Off work, love playing cricket and chess
- Loves mentoring and teaching



Agenda

- What are FAANG companies?
- Learning fundamentals
- Preparing for Interviews
- FAANG worthy resume
- Getting on FAANG-radar
- Preparing for interviews
- Additional tips

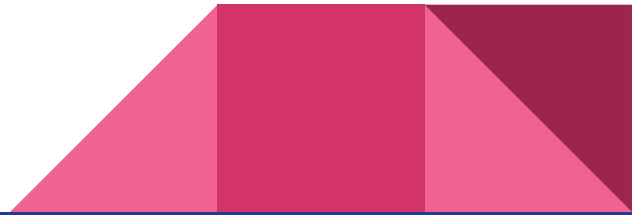


What & Why FAANG Companies?

- Facebook, Apple, Amazon, Netflix, Google
- Why?
  - Best software engineering processes
  - Smart people
  - Competitive Compensation

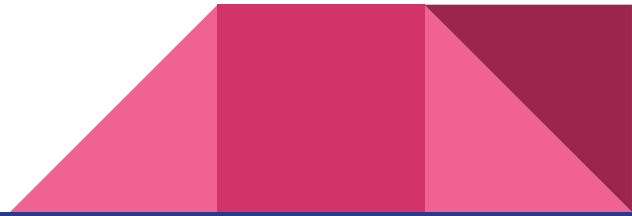
Learning Fundamentals

- Pick up *any* language
  - Java, Python, CPP, etc
- Learn the basics
  - Basic syntax
  - Data types, functions, loops
  - Cracking the coding interview, GeeksforGeeks, Coursera, MIT-OCW, Youtube
- Learn thee basic data structures
  - Sorting
  - Arrays, Strings
  - Calculating time complexity



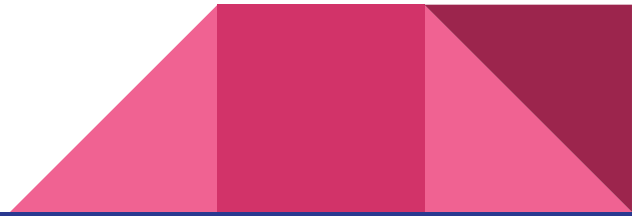
Preparing for Interviews

- Build your resume
- General structure of FAANG interviews
- Coding Interview Practice
- Behavioural Interview Practice
- Mock Interviews



Crafting a FAANG-worthy Resume

- One pager resume
- Show impact on the projects with metrics
  a. Number of users
  b. Revenue
  c. Scaled to x users etc
- Tailor resume to each role with key keywords (relevant to the jd)
- Supplement with portfolio site, GitHub, LinkedIn



General structure of FAANG interviews:

- Initial phone screen (30-45 mins)
  - Assess communication skills
  - High-level questions about your experience
  - May include a simple coding question
- Technical phone interview(s) (45-60 mins)
  - More in-depth coding questions
  - Focused on data structures, algorithms, object oriented design
- On-site interview (4-5 rounds)
  - Coding questions increase in difficulty
  - System design and object oriented design questions
  - Behavioral and situational questions
  - Includes lunch and campus tour
- Follow-up interview
  - For senior roles
  - High-level questions from CXO and VP levels
- The process typically lasts 4-8 weeks and
- Consists of both technical and behavioral interviews
- The on-site round is critical and will have multiple intense coding, system design, and behavioral questions.

Coding Interview Prep

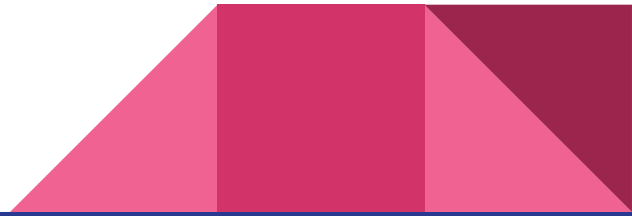
- Start with easier data structures like arrays, strings, linked lists
- Progress to more complex topics like trees, graphs, recursion
- Focus on patterns like sliding windows, fast & slow pointers, etc
- Use resources like LeetCode, HackerRank, CodeSignal
- Sort questions by frequency and difficulty
- Begin with easier difficulty questions, then work up to medium and hard
- Practice out loud explaining your thought process
- Review solutions only after attempting questions fully
- Focus on improving speed and efficiency over time
- It's important to build a strong foundation with simpler data structures and algorithms before advancing to more complex coding interview questions. Resources like LeetCode allow you to progressively practice by difficulty level. Explaining your thinking out loud and only reviewing solutions afterwards simulates the interview environment.

Behavioral Interview Prep

- Communication style
  - Speak clearly and concisely
  - Demonstrate active listening skills
  - Ask clarifying questions
- Culture fit
  - Research company values and principles
  - Align your answers and stories to their priorities
  - Show passion for their mission
- Storytelling
  - Use STAR method (Situation, Task, Action, Result)
  - Focus on impact and lessons learned
  - Be succinct yet compelling

Resources:

- Big Interview - Lessons on interview skills and practice questions
- Interviewing.io - Anonymously practice behavioral interviews
- Pramp - Peer video interviews with feedback on soft skills



Mock Interviews Are Key

- Practice explaining your thinking and coding approaches out loud
- Simulate the pressure and problem-solving of real interviews
- Identify areas for improvement ahead of time
- Gain feedback on communication style and interview skills
- Build confidence through experience

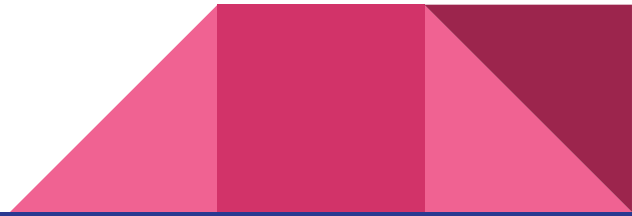
Recommended platforms:

- Pramp - Peer mock interviews with software engineers
- LeetCode - Mock interviews focused on coding problems
- Interviewing.io - Anonymous mock interviews with FAANG engineers
- Triplebyte - Formal mock interviews with trained interviewers

Mock interviews are critical preparation because they replicate the high-pressure FAANG interview format. Taking advantage of practice interviews helps identify weaknesses and improve performance in real scenarios.

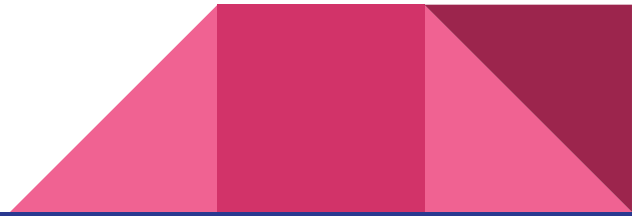
Getting on FAANG's Radar / Apply for jobs

- Leverage your network for referrals
  - Alumni at companies
  - Friends, colleagues, mentors
- Attend career fairs and company info sessions
- Reach out directly with a compelling note
  a. On LinkedIn
  b. Company websites
- Have an outstanding resume and online profile
- Updated LinkedIn Profile



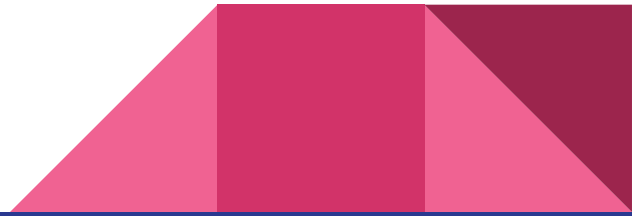
Navigating FAANG interviews

- Communicate your thought process clearly and explain your logic
- Ask clarifying questions if you don't fully understand the problem
- Write clean, organized code with good variable names and comments
- Test your code thoroughly with different inputs
- Think through edge cases and test your code thoroughly:
  - Zero, negative, extremely large inputs
  - Empty inputs/missing data
  - Repeated elements
  - Inputs that cause errors
- Analyze the efficiency and complexity of your algorithms
- Come up with multiple solutions if possible and compare tradeoffs
- After finishing coding, walk through example inputs and outputs
- Be confident, friendly, and enthusiastic
- Admit if you don't know something rather than try to fake it
- Thank the interviewers sincerely for their time and the opportunity
- Ask smart questions at the end demonstrating your interest



System Design Interviews

1. Clarify requirements and constraints
    a. Ask questions to understand scale, traffic, priorities, etc.
2. Sketch High-Level Design
3. Drill Down on Components
4. Discuss Tradeoffs
5. Estimate capacities needed based on scale and traffic
6. Discuss alternate architectures and when you would use them
7. Highlight areas that could become bottlenecks and how to avoid them
8. Explain how the system can scale up over time as demand increases
9. Focus more on principles and tradeoffs rather than getting stuck on details
10. Communicate Thought Process



Working at FAANG ??

- Fast-paced environment - Things move quickly and you'll need to adapt to keep up. Requirements can change rapidly.
- Smart coworkers - You'll be surrounded by extremely talented engineers, so be ready to step up your game. There's a high bar for quality.
- Cutting-edge tech - You'll have access to the latest tools, infrastructure, and platforms to build innovative products at massive scale.
- Perks and benefits - Free meals, transportation, onsite services, generous pay and stock options are standard. Lots of employer-sponsored social events.
- Flexible work styles - Remote work options have increased post-COVID. But overall still a mix of in-office and WFH.
- Constant learning - Continual opportunities for training, mentoring, conferences etc to develop your skills. Always new things to master.
- High impact projects - Your work reaches millions of users which is highly rewarding. Products have global visibility.
- Not perfect - Despite hype, still just companies at the end of the day. Office politics and slow-moving bureaucracy exist too.

Important tips that are often overlooked

Mock interviews - Practice whiteboarding and coding out loud. Identify gaps before real interviews.

Application timing - Applying early in the recruiting cycle gives you the best chance.

Soft skills - Don't neglect communication, cultural fit and behavioral prep.

System design prep - Study core principles like scaling, caching, databases, load balancing.

Metrics and data - Quantify accomplishments on your resume with numbers and metrics.

Research thoroughly - Know the company's products, roadmap, team structure, and open roles.

Review basics - Refresh fundamental CS concepts like bit manipulation, recursion, Big-O notation.

Test edge cases - Walk through examples verbally. Check 0, null, negative, extreme inputs.

Clarify ambiguities - Ask follow up questions. Don't make assumptions in problems.

Portfolio/GitHub - Showship side projects, contributions to open source, coding blogs.

Q/A