Hi Pranam,

Warm Greetings from Google India !

I wanted to share with you some materials that will help you prepare for our technical interviews. The information may seem lengthy, but trust me, it's super helpful.

To get up to speed with respect to how and what to prepare, please find below certain materials/links which will be

extremely useful to you. Do have a thorough look at them. Will follow up next week as well!

Request you to start preparing meanwhile so that we are in a comfortable position by the time the interviews come up. Here are certain key points you should keep in mind while preparing for the Technical Phone screen interview.

As discussed Please go through this and prepare well and let me know once you're ready.

Estimated Preparation time : 2-3 weeks.

Interview: Technical Phone Screen

- Duration: 45mins
- Google Hangouts call (video is optional)
- Coding is done on Google Docs

Please ensure you have a working internet connection and current web browser :)

There will be about 1-3 different questions (dependent on complexity - **e.g. 1 Hard OR 2 Medium problems**) related to coding, data structures and algorithms. You will be expected to come up with optimized and production-ready code, free of bugs. You should not write in pseudo-code.

I would **highly recommend** watching this video first to get a good grasp of the tips/ideas/areas of focus for the interviews here at Google. Do then read the next part to concretise these interview tips.

Important areas of assessment

- **Coding** - Good code quality (No IDE. Fully functional code. Take care of syntax, good language constructs, clean and concise code) & coding fast (you can code in Codechef timed challenges to practise coding under time pressure)
- **DS & Algo** - Refer to below topics on what we are assessing for
- **Good communication** - Talk out loud, explain your thought process to the interviewer
- **Attitude** - Being independent, lead/drive the discussion forward while being receptive to hints from interviewers and integrating it in your solution
- **Speed of solving the question** (Ideally 20mins per question - refer to below notes on time allocation)

Strategy for Success (Framework)

- Step 1: Clarify the problem
- Step 2: Define your approach

- Step 3: Propose a solution
    - Propose a solution before coding.
    - Feel free to say that your first solution will be refined later
    - Run through at least one or two examples to check for correctness
    - Use reasonable variable names or clean up the code after the first pass
    - Ask if the interviewer has any questions before refinement
- Step 4: Propose an alternative solution
- Step 5: Implementation

Breakdown of Framework

- Tip 1: Think out Loud & Communicate your Thought Process
    - Google interviewers are assessing you based on your problem solving skills, and want to know your full thought process behind why and how you derived the final result
    - **Talk through your entire thought process & explain about your approach, how you derive a certain algorithm, explain the trade offs, and discuss the complexity of the solutions you are proposing and explain your code (Talk and Code).**
    - Most important would be to pick the right data structure and algorithm for a specific problem! Talk about how they're implemented and why you'd choose one implementation or data structure instead of another.
    - **Do not** mumble to yourself or keep silent when thinking.
- Tip 2: Ask Clarifying Questions (Very Important!)
    - Never jump straight and code up the solution. Always take a pause, look at the interview question and ask some clarifying questions
    - **These clarifying questions should enable you to cover for edge/boundary cases better and to define the scope of the question**
    - Eg:"Describe a good algorithm for sorting a million numbers"
    - Good Clarifying Questions:
        - What is the range of the numbers? How are they distributed? Are the numbers integers or floating points? How much memory is available? Are there duplicates?
- Tip 3: Discuss algorithmic complexities & Identify all Edge Cases Independently
    - For Algorithms, you will need to know big-o notation very well.
    - **Always state time and space complexities upfront. Think of how you can reduce the complexity further to reach an optimised solution!**
    - Distinguish between average case/worst case runtime
    - Consider amortized time complexities!
    - **The goal is to reach the most optimised solution** at the end of the interview, and to have a complete working solution.
- Tip 4: Test your Code
    - **Check for boundary conditions!**
    - Stress tests to ensure that the code runs within time constraint
    - Create tests that have 100% code coverage
    - Rectify any bugs in your code before the interviewer points it out
- Tip 5: Ensure Good Code Quality on Google Docs
    - You are expected to code in Google Docs. Since it does not have any IDE, you are expected to type out your code from scratch
    - Type as close to fully functional code as possible. This code should be **maintainable and readable by a large database of engineers**
    - **Code in the latest version of your preferred language and use appropriate language constructs**. Take care of variable names and syntax.
    - **Do not** use Pseudo code or shortcuts, it is not good enough.
    - Always cover for Edge/Boundary cases
- Tip 6: Positive & Independent Attitude & Being Open to Feedback
    - If you get stuck, stay calm, asking questions can help to reduce the scope of the problem.
    - **Aim to solve as much independently as possible, ideally with as few hints as possible.**
    - Always take the initiative in the interview, and treat it like a technical discussion.
    - Listen attentively to the interviewer, and integrate the hints/suggestions by the interviewer to your solution.

Interview Preparation Plan (created by Google Software Engineers)

- **1. Revise all concepts on data structures & algorithms- you can also use this gitHub link on CS fundamentals that can serve as a checklist while preparing. This BigO cheat sheet <http://bigocheatsheet.com/> could help you as well!**
- 2. **Structured Revision plan** on the topics that to cover (Eg. hashtable, hashmaps, trees, arrays, strings, graphs, dynamic programming and more)
    - Practise per category
    - Practice up to a level that you reach competency - Solve the question in 20/40 minutes (for medium and hard problems respectively) and come up with the optimal solutions
- 3. **Practice Problem Identification** by picking random problems and practice identifying "Which category does this problem belong to? Backtracking/Dynamic programming? Solution/Algorithmic design?".
- 4. **Practice coding without an IDE**, be familiar with the differences in how you should write code in Google Docs. Do practise coding in Google Docs within a set time frame and getting comfortable talking while coding.
- In summary: Practise a wide variety of questions, and simulate actual interview conditions! You can also run **mock interviews** here at pramp.com

Frequently asked topics (in no particular order)

- Binary search
- BFS/DFS/Flood fill
- Tree traversals
- Hash tables
- Linked list, stacks, queues, two pointers/sliding window
- Binary heaps
- Dynamic programming
- Union find
- Ad hoc/string manipulations
- Other good to know topics: Trie, segment trees/fenwick trees, bitmask
- Google Interview Style Guides (C++, Python, Java; Javascript)
- You'll be expected to know and apply: **lists, maps, stacks, priority queues, binary trees, graphs, bags, and sets**.
- For algorithms you'll want to know **greedy algorithms, divide and conquer, dynamic programming, recursion, and brute force search**.
- You'll definitely want to be conversant with bigO notation, time-space complexity, and real world performance of all of this.

What we are assessing for Data Structures & Algorithms

- Can you implement the most optimized data structure and algorithm for the question?
- Can you explain the tradeoffs between the data structure/solution?
- Can you explain why you choose a data structure for implementation
- Can you explain and analyze the time and space complexity correctly
- Can you translate the algorithm to code well?

Useful Resources

Videos/Blogs:	Coding Practice:	Free Refresher Courses:
1) Example Coding/Engineering Interview	HackerRank Topcoder LeetCode Interviewcake	Coursera - Algorithms, Part 1 Coursera - Algorithms, Part 2 Udacity - Intro to Algorithms

2) How to: Prepare for a Google Engineering Interview 3) Interview tips from Google Software Engineers 4) Steve Yegge's Blog (read me!) 5) Check out this YouTube playlist!	Kattis Geeksforgeeks Pramp - Mock Interview	MIT Open courseware - Introduction to Algorithms

https://medium.com/swlh/my-preparation-journey-for-google-interviews-f41e2dc3cdf9
https://www.geeksforgeeks.org/google-interview-preparation-for-software-engineer-a-complete-guide/
https://www.youtube.com/watch?v=6ZZX9iIgFoo&t=114s
https://www.youtube.com/watch?v=fNpQrYwxsvo
https://www.youtube.com/watch?v=HYK5lpkKBPo
lLearn more about our candidate privacy policy.l

All of us at Google also recognize that the world is going through a challenging time. If there is any way that we can better accommodate you through the process, please let us know.