<https://medium.com/@siddhism/how-i-prepared-for-google-0-leetcode-questions-to-200-questions-e37690ebce85>

# How I prepared for Google — Solving 200 leetcode questions.

[siddhesh suthar](https://medium.com/@siddhism?source=post_page-----e37690ebce85--------------------------------)

#### [siddhesh suthar](https://medium.com/@siddhism?source=post_page-----e37690ebce85--------------------------------)

#### [Mar 29, 2020·6 min read](https://medium.com/@siddhism/how-i-prepared-for-google-0-leetcode-questions-to-200-questions-e37690ebce85?source=post_page-----e37690ebce85--------------------------------)

The journey has been somewhat unbelievable. If someone would ask me 1 year back why don’t you try for Google, I would have said not possible. I am not that good at DS/Algo. I had solved 0 questions on leetcode a year back. I solved 200 Questions until my final interview and got selected. Sharing my experience and some tips here.

I graduated from Engineering College Bikaner in 2015. I got an opportunity to give an interview at Google India in July. And I started brushing up my skills for the first time seriously. I had basic knowledge before this, which was acquired from solving simple questions and reading up [code monk](https://www.hackerearth.com/practice/codemonk/) articles on HackerEarth.

Most of my preparation and practice was done on three platforms -

1. Geeksforgeeks (mostly when I need an explanation on basic questions for any topic)
2. Leetcode (for practice)
3. HackerEarth (For reading up basics for any data structure)

Especially I like the articles on HackerEarth about Heap, Union Find & Dynamic Programming. For any topic, I would start reading the basics of a particular data structure/ Algorithm and read about some of the common problems on that topic. After that, I used to move to leetcode to solve problems.

I had 1 phone interview & 5 on-site interviews (4 Algo & 1 Behavioural). the on-site rounds were split in two days

**Resources**

1. Leetcode, geeksforgeeks, HackerEarth.

2. Cracking the coding interview <https://www.amazon.com/Cracking-Coding-Interview-Programming-Questions/dp/0984782850>

# Focus

It’s all about focus. First of all, you need to set up space where you can work independently and concentrate on spans of longer than one hour. Focus in the distracted world was the key for me and it may be the most likely missing factor for most of us.

*Follow*[*deep work*](https://www.amazon.in/dp/0349413681?linkCode=osi&th=1&psc=1)*. Few hours of deep focused work > sitting unfocused for hours.*

## What works for you

* Find out what time works best for you. maybe early morning or late at night. before office or after office.
* You need to give at least 1–2 hours per day (or 10–15 hours per week)
* If you are going to college/office, find out what time you have the mental energy to focus. It can be different for different people. I used to study from 1–4 AM (yeah after midnight). because distractions are zero at that time. There is no right or wrong time. find out **what works for you**. The timing which works for you is the best.

*Get rid of all distractions, social media, put your phone on silent if you can for 1 hour. Trust me if you are not giving at least 30 minutes of focused time per problem as a beginner, you are not trying hard enough.*

# How many Questions you should solve? — Whatever it takes

The focus has to be on quality more than quantity. The answer to this question varies according to your experience and expertise on each topic.

If you are starting from scratch. Target 100 leetcode problems as a number and divide it well across different topics and difficulty levels.

If you have already practiced some topics (like DP, graph) you can reduce the number for that topic.

Overall I solved 200 Questions on leetcode.

There are all kinds of extremes to this answer, there are people who have solved more than 500 questions and weren’t lucky to clear the interview. Some people solved 300 questions and cleared interview in 22 companies. You need to find out what works for you. when you feel confident enough on one topic, you can move on to next.

Key pointers specific to leetcode-

1. Pick a topic and solve problems with that. Do not move to a different topic before finishing it.
2. Solve Easy, Medium & Hard problems in 3:6:1 ratio (3:5:2 is also recommended).
3. Solve Easy problems by sorting them on two factors

i) Acceptance rate (no. of people solved/ attempted) this indicates the question is easy.

ii) Frequency (no. of times question appeared in interviews). This data is crowdsourced, sorting by frequency is a premium feature.

# Problem Solving Strategy

1. Make sure you understand the question correctly. Validate your assumptions by thinking of some different input and output.

*Try to think in verbose mode. Practice the approach on paper or whiteboard. Think of it as if you are explaining the approach to the interviewer.*

1. Move on to write code. make sure you understand the time complexity of the solution before writing it. and if it can be made better.
2. Try to look at discussion even if you scored well. There might be better approaches/learnings in each of the questions you solve.

# Track your progress

Once done with theory. Start maintaining a record of questions you do from all platforms on your own. I prefer google sheets, you can use any tool you like. This is highly recommended. As it creates a positive feedback loop and creates the much required motivation-action cycle.

Record these things

1. Question link
2. Topic, Difficulty level
3. Start date, start time, finished coding, submitted with 100% success (We might take some time in getting to all tests passed stage sometimes because of missing corner cases. That’s why we need two finish times).

Here’s [list of questions](https://docs.google.com/spreadsheets/d/1KkCeOIBwUFfKrHGGZe_6EJRCIqaM6MJBo0uSIMSD9bs/edit?usp=sharing) that I solved.

# Consistency

In **running**, **cadence** is often defined as the total number of steps you take per minute.

while solving problems you need to maintain a cadence. However most of us have a day job, you’ll not solve problems at the same rate. But you should still set some goals. My suggestion is avg at 2 problems/day or 14 problems/week. If you are not able to cover on weekdays, try to up the game on weekends.

***Of course, it’ll need some sacrifices and some social distancing and giving up on some plans.***

# Unblock yourself

If you are not able to solve a problem, it doesn’t mean you are inadequate. Remember this, make a poster and stick it on a wall.

It only means that there is some new learning for you. You need to learn a new way to solve a problem or a new concept. that’s it.

Sometimes you’ll have to fight with your ego and learn that you can’t solve all problems. Take help, ask for help, discuss it with friends or peers at work.

*Do not spend more than 24 hours on a problem. Last resort is to look at hints, leetcode discussions for the problem or solution (in that order).*

If rather than this, you keep on trying to solve the problem for more than 1 day it’ll hamper your confidence and consistency.

# Leetcode Premium

Leetcode premium is something which helps you in three ways

1. It lets you filter questions by company tag.
2. You can sort questions by frequency. Frequency here represents no. of times the question has been asked in actual interviews. This factor is really important if you want to keep up with the trend of the latest questions.
3. You can get access to the Top 50 Questions for each company. This is particularly good when you want to practice just before interviews. These questions can help you brush up on all concepts in a short period.

Costs ~12000 for a year. but it’s worth the money.

# During the interview

There are few things to take care of during the actual interview

1. Always understand the problem first. Ask counter questions like can the array contain negative numbers? is the graph acyclic? (just make sure things are as expected, do not assume anything). This is one of the important factors, don’t jump into solving the problem.
2. Explain your thought process for the solution. While you are thinking about the solution, you need to be verbose. you can always take some time to think and speak. But don’t be completely silent.
3. There might be whiteboard involved or not. chances are 80%. Prefer practicing your solutions on paper or whiteboard always when you are solving problems.
4. Once you and the interviewer agree that this is the most optimal approach to solve the problem. and you prove it by dry running on some example (and some examples which will have corner cases). Only after that, you’ll be asked to write code. So make sure you learn to explain your thought process and prove that solution is optimal (with what time complexity) before jumping into coding. Make it a habit in your general practice as well.
5. Finally, you write the code as expected. If time allows you to discuss the test cases and corner cases with the interviewer.

Here’s an [example coding interview](https://www.youtube.com/watch?v=XKu_SEDAykw)

Bottom line is, always try to find out what works for you. There is no optimal strategy or secret to success. You pave your own path to success and it can be unique. The only thing you need to do right now is to start and start with a plan. Hope this helps, Happy to help if you have any further questions.