Algorithm Interview Foundations

The algorithm interview is typically a 60 minute chat with an engineer in which you solve a technical problem. The expectation is to have a working solution before the end of the interview, with time left over to ask the interviewer questions about the company. The beginning of the interview will include introductions and problem explanation as well. You should aim to solve the problem and all of its parts within 30 minutes. There is a general framework for how to solve the problem which involves:

- Ask clarifying questions
- Come up with an approach, without writing any code yet
- Validate the approach with the recruiter
- Implement the solution with code
- Demonstrate correctness by running through test cases

You want to leave the following impressions with your interviewer by the end of the interview.

- You are a strong problem solver
- You are a great communicator
- You are thorough and write great code
- You move fast

Doing all the above sends great signal and will earn you a strong yes.

Let's break down this interview.

The bar

Here are some example problems that mirror the difficulty level of what you can expect in an actual algorithm interview. This is not an exhaustive list, only meant to give you a rough idea. You should feel confident handling these kinds of questions by the time you do actual screens and onsites. Your goal is to have a working solution that compiles and passes test cases within 30 minutes or less. All these questions can be found by searching their name on LeetCode.com – in general this is a useful resource for practice questions.

- Merge Intervals
- Meeting Rooms
- Number of Islands
- Spiral Matrix
- Coin Change

- Parse Lisp Expression
- Implement LRU Cache
- Text Justification
- Wildcard Matching
- Minimum Window Substring

You can also get a preview of the questions a company asks by doing a google search for "company interview questions glassdoor" or "company interview questions github". Sometimes you may find actual interview questions that will be asked on the screen or onsite!

Framework

The first step is to learn the problem solving framework. One of the best resources for this is Cracking the Code Interview (CTCI). We recommend you read chapters I - VII. These chapters thoroughly break down the framework for solving the algorithm interview. Internalize these chapters and apply this methodology when solving any algorithm question.

Foundations

With the framework in mind, the next step is to do a lot of practice problems. We recommend you do questions from either CTCI or Elements of Programming Interviews (EPI). We think these are the two best resources for algorithm questions. It is highly unlikely you will encounter a real interview question that isn't covered by these two books. We have a slight preference for EPI since it covers more questions than CTCI. At a minimum, you should do all problems covering the following sections:

- Strings
- Arrays
- Linked Lists
- Stacks / Queues
- Heaps
- Trees
- Hash Tables
- Searching
- Sorting
- Recursion

Once you have strong foundations in the above, reach further and do problems covering:

- Dynamic Programming
- Greedy Algorithms
- Graphs
- Bitwise Operators

These are really hard questions and you may never be asked them in your process. You will be extremely well prepared if you put in the time to solve and understand these kinds of problems.

When doing these problems, set a timer for 20 minutes and try to solve each question on your own. Look at the solution if you finish or time runs out – spend as much time as you need to understand the answer. There are a lot of tricks and concepts to learn, take the time to internalize them. Once you know the trick, future problems will be easier. Your goal isn't to see and memorize every problem / solution. Instead, your focus should be to increase your bag of tricks and sharpen your problem solving abilities. Follow the pattern of coming up with an approach, implementing it, and running your code against test cases.

By going through lots of problems in different areas you will be ready to take on anything.

Homework

- Read chapters I-VII of CTCI
- Get your foundations by reading the following chapters in either CTCI or EPI

  - Strings
  - Arrays
  - Linked Lists
  - Stacks / Queues
  - Heaps
  - Trees
  - Hash Tables
  - Searching
  - Sorting
  - Recursion
  - Dynamic Programming
  - Greedy Algorithms
  - Graphs
  - Bitwise Operators

- Solve sample questions that match the interviewing bar difficulty by searching the following on LeetCode.

  - Merge Intervals

- Meeting Rooms
- Number of Islands
- Spiral Matrix
- Coin Change
- Parse Lisp Expression
- Implement LRU Cache
- Text Justification
- Wildcard Matching
- Minimum Window Substring