leetcode 刷题清单

一路走好
微软 研发工程师

刷LeetCode已然成为北美算法面试的正统，但刷LC不是一件轻松的事情，动辄1000+的题目，全部刷完，半年到一年就过去了。一道道全刷完肯定不是最优策略。那到底应该怎么刷，才能进入大公司呢?

咱们就来一起看看LeetCode美版上面大神们的刷题面试经验吧!

这篇文章里面提到的经验，来自于美版的discuss区。我在这里只是搬运过来，加上自己的点评，从而使得大家刷题更高效。

我按照从票数从高到低的顺序去整理这些回答。之前都是各处搜索刷LeetCode的经验，其实真的是踏破铁鞋无觅处，得来全不费工夫，因为LC的讨论区就是满满的精华！这里选取了目前点赞超过100的所有分享。

原文：leetcode.com/discuss/ca...

Title：From 0 to clearing Uber/Apple/Amazon/LinkedIn/Google

点评：细致而又有效的学习算法和面试经历，强推。

Update: My Google on-site results were positive :)

This is a post detailing how I went from bombing almost all interviews to clearing Uber, Apple, Amazon, Linkedin, Google and a bunch of other companies

A little bit of history

I was working at Microsoft where I converted my internship into a full time
then I had just relied on the knowledge that was fresh in my mind from my

Around 2017 - I gave the following interviews:
Uber - didn't get past the phone screen
Salesforce - failed onsite
LinkedIn - failed onsite
Google - failed phone screen

I was dejected, I had done a lot of my preparation from another site called GeeksForGeeks, and I found three major issues there -

1. Too much information, too many questions
2. No online judge to practice, although I hear this has changed now-a-days
3. Horrible UI. It pains my eyes to even look at the site (this is a purely personal taste)

I heard of Leetcode way back in 2013, when the community was much smaller than it currently is. A colleague of mine suggested and said that Leetcode has very good quality questions. So I figured I should give it a try.

Leetcode preparation phase

The reason I was hooked to Leetcode was the no-bullshit question format, with a clean and intuitive UI. I decided to do two questions a day consistently, gauge my progress over time and then schedule my interviews. When I first started out, I found even the ‘Easy’ questions hard to pass in the first attempt. I was easily spending 45 minutes on trivial questions. As time progressed, I started seeing improvements. In about two weeks, I was solving ‘Easy’ questions in 20-25 minutes and most of them were passing in my first attempt.

I decided to proceed to Medium questions and the same story repeated. Found them extremely hard, but as time progressed, I could solve a good number of those questions as well. I have to admit that I can do very few hard questions within an hour even after solving 250+ questions. And sometimes it takes hours to understand the solution as well.

Eventually I tried taking out Leetcode premium. I took it for a month first, and loved it, so I took a yearly subscription. The best part about premium subscription is that you get questions sorted company wise and frequency wise. I would prepare by targeting the company specific questions for the big players (Amazon, Apple, LinkedIn, Facebook, Microsoft, Uber, Google etc.) and then by frequency. It worked out great for me, and especially useful when you are running under a time crunch.

only if you’ re happy, upgrade yourself for a year.

Current Status
Eventually after about 150 questions, my interviews were as follows:

Confirmed offers from:

- Amazon (L5 / SDE2)
- Apple (ICT3 / SDE2)
- Uber (L4 / SDE2)
- LinkedIn (Staff Software Engineer)
- Salesforce (Senior MTS)
- Google (L4 / SWE3)

Narrowly lost out on Facebook - I got a positive in problem solving, and a borderline positive in System Design.

I did face a lot of questions which were word-to-word lifted from Leetcode, and I found a bunch of new questions as well. But solving questions here helped a lot in shaping my thinking process.

My advice to people in the same position as I was in

- Don't be demotivated at your current level. I started from a place where I was bombing every interview and now I've done much better than what I was expecting. Start and be consistent.
- Try getting comfortable in every topic. Don't leave anything behind.
- Don't spend more than an hour on any question. If you can't figure out the solution, mark it and revisit later.
- Use the discuss forums. They are literally the best part about Leetcode and a key differentiator why no other platform can match up to Leetcode in terms of interview prep.
- Upsolve when you can't solve a problem - i.e look at the solution, understand and then do it again on your own.
- Keep taking notes about what a problem is teaching you. Keep revisiting them. Spaced repetition helps in committing things to memory.
- Don't be in a hurry. Enjoy the ride :)

Cheers! (also thanks to Leetcode and the amazing guys who keep posting epic stuff in the forums! I owe a lot to all of you)

Tips and Resources for System Design

I'm updating the post with the resources I used for system design interviews as some of the fellow leetcoders wanted to know.
First of all, depending on your experience, the expectation in system design round varies. If you are interviewing for SDE1/entry level SDE role, most companies don't bother asking system design. As the level progresses, you will be evaluated on things like the breadth and depth you possess, the variety of approaches you are able to take and trade-offs you are able to make etc.

First, I want to share my system design template which I posted in another post - System Design Template

- On to the preparation part - clearly, experience is the best teacher in this area. But for beginners, I would suggest going through System Design Primer. This has been the single best resource for me to lay down my basics. If you have time, I would highly recommend going through every link listed under references in the repo. Highly useful!
- Second, I purchased the course Grokking the System Design Interview. I think it's $50 and a one time purchase. The theory part of this is not great, but the questions cover a lot of variety and take a very structured approach. This helped me a lot! Honestly the most return on investment for system design. If you are done with these two steps, you should be comfortable enough to crack most system design interviews (including FAANG/FLAG).
- Third, as an added bonus, and a much recommended step, please go through some of the seminal research papers in the field of distributed systems. These discuss the most fundamental ideas and trade-offs in designing large scale systems. It is not necessary to do this to pass the system design round, but it will clearly spike your curiosity and lay down a much stronger foundation:

  - Amazon - Dynamo paper
  - Google - Map-reduce paper
  - Google - GFS paper
  - Facebook - TAO paper

Everything from here on is for people targeting senior engineering pos

area. The best way to gain this is by deepening your theoretical understanding and practical understanding.

- For deepening theoretical understanding, I found the following two resources immensely helpful:

1. Distributed Systems for Fun and Profit - free book, 100 pages or so, but excellent content
2. Designing Data Intensive Applications by Martin Kleppman - this is a more in-depth book, and you can get it on Amazon/libgen

- For deepening practical understanding, going through companies' engineering blogs is the best resource.

- Netflix tech blog, Uber, Twitter, Airbnb engineering blogs are high quality and are constantly updated.
- For a consolidated list of company-wise engineering blogs, refer to this awesome github repository
- There are a lot of great Youtube videos (append to youtube.com).
  - Jeff Dean's talk at Stanford: `/watch?v=modXC5IWTJI`
  - Building Billion user Load Balancer at Facebook: `/watch?v=bxhYNfFeVF4`
  - Netflix Guide to Microservices: `/watch?v=CZ3wIuvmHeM`
  - Amazon DynamoDB deep dive: `/watch?v=HaEPXoXVf2k`

Hope this helps!

原文：leetcode.com/discuss/ca...

Title: How to effectively use LeetCode to prepare for interviews!!

点评：按照题目从Easy，medium，到hard的分类，进行了策略分析，更是[illegible]地给出了每个分类下面的经典题目。

be useful for the other person looking for some direction, and getting a clear idea of how to progress on this journey.

[EASY]

- If you are just starting off, starting from easy problems is the best way to get in touch with your coding skills again, and/or build up new ones.
- As mentioned by other users, it’ s better to have a breadth-first approach to easy problems. Pick 5-6 questions from each topic and try to master the basics of the topic.I did a depth first of all the easy ones but looking back I feel it’ s better to move to the next topic once you feel confident in one.
- The purpose of the easy questions is to ingrain the basic concepts of a particular data structure or algorithm and if you look closely you will find a lot of questions asking for solutions testing your ability/understanding of language syntax, data structure core basics,etc, basically the “dirty” work, we so often don’ t give due attention. Leverage the same and get your hand dirty.
- You will also find a healthy mix of the use of two-three data structures or algorithms in some problems. Look for such problems, for they are a great way to test your understanding of how more than one DS work together to build a beautiful solution.
- That being said, I would recommend moving from the “easy” phase as soon as possible, for its the mediums that will be asked mostly in interviews.
- Must-do Easy questions [Download the list here] :
  - 13. Roman to Integer
  - 20. Valid Parentheses
  - 21. Merge Two Sorted Lists
  - 141. Linked List Cycle
  - 155. Min Stack
  - 169. Majority Element
  - 202. Happy Number
  - 204. Count Primes
  - 242. Valid Anagram
  - 371. Sum of Two Integers
  - 88. Merge Sorted Array
  - 108. Convert Sorted Array to Binary Search Tree
  - 189. Rotate Array
  - 205. Isomorphic Strings
  - 226. Invert Binary Tree
  - 448. Find All Numbers Disappeared in an Array
  - 572. Subtree of Another Tree
  - 557. Reverse Words in a String III

- 617. Merge Two Binary Trees
- 665. Non-decreasing Array
- 669. Trim a Binary Search Tree
- 674. Longest Continuous Increasing Subsequence
- 703. Kth Largest Element in a Stream
- 705. Design HashSet
- 852. Peak Index in a Mountain Array
- 1160. Find Words That Can Be Formed by Characters

[MEDIUM]

- Once you feel confident enough with the basics, you can move onto the medium ones. Medium problems are pivotal in interview preparation since any decent interview will have at least one instance of medium problems. Moreover some of the advanced data structures and algos like graph, backtracking, dynamic programming are not well covered in the easy section, so the medium problems are a good stop to learn them.
- Again, breadth-first approach works wonder in this case as well. Pick like 10 problems from each topic, try to solve them to the best of your capabilities and practise of easy problems, and if you find it difficult to solve at first, take help from the discussion forum.
- What's important to note is the purpose of you doing N number of problems, should solely be to learn new techniques, be able to apply the learnt technique in similar problems later, and learn the basics of advanced data structure that you might have missed in the easy section.
- It's normal to not be able to solve medium level problems at first. Don't feel bad about taking help from the discussion forum, but try to learn the techniques as much as you can.
- ***Do not cram the solution***. There is no point hoping you will remember the solution in an actual interview, because in a pressure situation it's not advisable to rely just on your memory. Rather place your bets on your acquired skills.
- While you are solving questions of a particular topic, make sure to give the theory a good read before or during the process of solving questions. It's important to know the basics of implementations of various data structures and algorithms like graph traversals, backtracking, memoization, trees, binary search etc. There are various resources available for the same, geeksforgeeks, Cracking the Coding Interview, Tushar Roy's lectures, Back To Back SWE, to name a few.
- Once you feel fairly confident after say, 60-70 mediums, it's always a good idea to start giving contests. Contests not only simulate the actual interview scenario

- Keep looking at better solutions posted by other users, compare your solution with them in terms of time and space complexities, know when to make the tradeoff between the two and ask for help whenever you feel stuck. Be open to learning and the rest will fall in place.
- A comprehensive list of must-do medium problems [Download the list here]:
  - 5. Longest Palindromic Substring
  - 11. Container With Most Water
  - 17. Letter Combinations of a Phone Number
  - 22. Generate Parentheses
  - 33. Search in Rotated Sorted Array
  - 34. Find First and Last Position of Element in Sorted Array
  - 46. Permutations
  - 24. Swap Nodes in Pairs
  - 48. Rotate Image
  - 49. Group Anagrams
  - 56. Merge Intervals
  - 75. Sort Colors
  - 94. Binary Tree Inorder Traversal
  - 102. Binary Tree Level Order Traversal
  - 106. Construct Binary Tree from Inorder and Postorder Traversal
  - 117. Populating Next Right Pointers in Each Node II
  - 142. Linked List Cycle II
  - 150. Evaluate Reverse Polish Notation
  - 151. Reverse Words in a String
  - 166. Fraction to Recurring Decimal
  - 179. Largest Number
  - 200. Number of Islands
  - 207. Course Schedule
  - 208. Implement Trie (Prefix Tree)
  - 215. Kth Largest Element in an Array
  - 236. Lowest Common Ancestor of a Binary Tree
  - 300. Longest Increasing Subsequence
  - 328. Odd Even Linked List
  - 338. Counting Bits
  - 347. Top K Frequent Elements
  - 355. Design Twitter
  - 417. Pacific Atlantic Water Flow
  - 421. Maximum XOR of Two Numbers in an Array
  - 424. Longest Repeating Character Replacement

- 525. Contiguous Array
- 658. Find K Closest Elements
- 763. Partition Labels
- 767. Reorganize String
- 1171. Remove Zero Sum Consecutive Nodes from Linked List

[Other points to note]

- Leetcode is a wonderful community to learn, discuss and find support in others going through the same process. Leverage it.
- Don't shy away from being a part of discussions, ask for help, or ask questions from others just because you feel it sounds stupid.
- Constantly read articles on Leetcode, not only related to solutions but interview experiences, problems faced during the process etc. You can benefit so much from learning about stories of others here and also derive a sense of belonging to the whole process.
- Make notes of all what you have learnt from a question, a trick or techinque for these will be handy when you will need to revise.
- It's okay to feel lost, confused or difficult to solve questions at times. Some questions are harder than others, some are trickier and some have no solid pattern to them. Don't get intimidated by the process. With time and practise everything gets better. Have patience and don't lose hope.

In the end, I want to mention that I am still not 'perfect' in solving problems, but leetcode have taught me a lot and I have definately come a long way in this process, and with a little guidance, determination and patience anyone can.

Any updates to the list are welcomed, so are any questions and do share your experiences of your leetcode journey, for everyone can benefit from each other's experiences.
Good Luck and Happy Leetcoding!

原文： leetcode.com/discuss/ca...

Title：My System Design Template

(1) FEATURE EXPECTATIONS [5 min]

(1) Use cases
(2) Scenarios that will not be covered
(3) Who will use
(4) How many will use
(5) Usage patterns

(2) ESTIMATIONS [5 min]

(1) Throughput (QPS for read and write queries)
(2) Latency expected from the system (for read and write queries)
(3) Read/Write ratio
(4) Traffic estimates
- Write (QPS, Volume of data)
- Read (QPS, Volume of data)

(5) Storage estimates
(6) Memory estimates
- If we are using a cache, what is the kind of data we want to store i
- How much RAM and how many machines do we need for us to achieve this
- Amount of data you want to store in disk/ssd

(3) DESIGN GOALS [5 min]

(4) HIGH LEVEL DESIGN [5-10 min]

---

(1) APIs for Read/Write scenarios for crucial components
(2) Database schema
(3) Basic algorithm
(4) High level design for Read/Write scenario

(5) DEEP DIVE [15-20 min]

---

(1) Scaling the algorithm
(2) Scaling individual components:
-> Availability, Consistency and Scale story for each component
-> Consistency and availability patterns
(3) Think about the following components, how they would fit in and how it wou
a) DNS
b) CDN [Push vs Pull]
c) Load Balancers [Active-Passive, Active-Active, Layer 4, Layer 7]
d) Reverse Proxy
e) Application layer scaling [Microservices, Service Discovery]
f) DB [RDBMS, NoSQL]
> RDBMS
>> Master-slave, Master-master, Federation, Sharding, Deno
> NoSQL
>> Key-Value, Wide-Column, Graph, Document
Fast-lookups:

>>> RAM [Bounded size] => Redis, Memcached
>>> AP [Unbounded size] => Cassandr
>>> CP [Unbounded size] => HBase, M

```
                > Eviction policies:
                        >> Cache aside
                        >> Write through
                        >> Write behind
                        >> Refresh ahead
        h) Asynchronism
                > Message queues
                > Task queues
                > Back pressure
        i) Communication
                > TCP
                > UDP
                > REST
                > RPC
```

(6) JUSTIFY [5 min]

```
(1) Throughput of each layer
        (2) Latency caused between each layer
        (3) Overall latency justification
```

原文： leetcode.com/discuss/ca...

Title：I've failed 10 interviews and 4 onsites, what should I do?

点评：这篇文章高赞的只要原因之一是，评论区满满当当的正能量加有效经验... 享，一定要去看下面的评论。

fail by the slightest margins, I'm always so close to getting the correct solution or answer but something always happens to trip me up. Just super stressed. What should I be doing? Has anyone experienced this?

UPDATE: This blew up :O Thank you so much everyone for all the positive comments/ suggestions!
UPDATE2: 11 Failed Interviews :)
UPDATE3: 12 Failed, oh well
UPDATE4: I got the offer of my dreams!!!!!!!!!!!!!!!!!!!

原文： leetcode.com/discuss/ca...

Title：Must do questions for a beginner!!

点评：新手必备的题目合集，咱们嘛也别问，问就刷完他们！而且题目可以保存到你的LeetCode的题目list!

Start with the questions which are tagged easy. Keep a goal of solving 5 questions daily, if you are not able to solve 5 in a day, reduce it to 2-3 questions. In the same way, you can increase the count to 8-10 questions daily. Make sure you remember your approach and try to explore more approaches available for that question.

A must do list according to me:
(these are some of the easy questions - once your solution gets accepted, leetcode shows you suggestion for next similar questions, you can solve them too rather than following this list.)

👉 Save as list to practice these questions

1. #1 Two Sum
2. #268 Missing Number
3. #155 Min Stack
4. #7 Reverse Integer
5. #20 Valid Paranthesis

8. #189 Rotate array
9. #112 Path sum

(list of medium questions - once you get comfortable with the platform and the questions, you can jump to these questions.)

👉 Save as list to practice these questions

1. #15 3 Sum
2. #19 Remove Nth node from end of a Linked List
3. #98 Validate BST
4. #54 Spiral Matrix
5. #55 Jump Game
6. #215 Kth largest element in an array
7. #56 Merge Intervals
8. #23 Merge K sorted list
9. #32 Longest Valid Parantheses
10. #403 Frog Jump
11. #239 Sliding Window Maximum

(As you are now a pro already, you can jump to the hard questions and do whichever questions you find interesting)

Happy Coding!!

原文：leetcode.com/discuss/ca...

Title: 11 companies, 55 Interviews, 9 offers including Google, Amazon (and Square?)

点评：真乃神人！

I’ ve been actively interviewing for past 5 months. Starting with a quick summary, he are the companies that I interviewed for:

Reject : Agoda, Uber

Sitting through so many interviews and preparing for such a long time, I’ ve learned a lot of things along the way. Things that could be useful to other people too. I aim to share all of that and add more value to that public corpus.

I’ ve been a mentor on Interviewbit Academy in parallel to my interviews and have been telling the same stuff to my mentees too. So there’ s a small validation that these methods work generally.

I won’ t be preaching any life lessons. Instead, I’ ll be sharing with you a list of actionable items that you can readily incorporate in your process. I’ ve implemented these myself throughout the process and that’ s why I’ ll try to provide concrete examples wherever possible.

I had around *3.5 years of experience* when I started and interviewed for *SSE/SDE2/L4 positions*, mostly for the Android platform as it is my expertise area. But I strongly feel that this post is equally useful to the ones applying for a general SDE. The only thing is that you won’ t find resources for backend system design (which I think you don’ t need much after grokking).

To put it at a common place and share it across multiple platforms, I've compiled everything into a medium post here:

medium.com/@yashgirdhar...

I want to share as much as possible and that’ s why it’ s gonna be a long post. So take a coffee, sit back and go through it slowly.

Leetcode has played a major role for me in cracking all of these interviews, specially the discuss forums. That's why I want to give back to the community here.

Regarding specific questions, I've added almost all my interview experiences on leetcode discuss in the interview experiences section.
I've added them anonymously but you should be able to filter all of them by :
Current Position: Senior SDE-Android
Location : Bangalore

Please feel free to reach out to me on linkedin for any help/questions/doubts.

原文： leetcode.com/discuss/ca...

Title: How I Train Myself For Google Internship Interview

点评：这是我自己在LeetCode发的帖子，希望大家能给个支持。这些资料都是我准备过程中帮助特别大的资料。

I am a heavy LeetCode user and I just got my official Google intern offer yesterday!

1. The material I used most could be found here: docs.google.com/spreads...

   In this sheet, the poster grouped the LeetCode problems into different categories a
   found that practicing in this fashion was/is a life saver.
2. I also like this Youtuber's channel:

This man has the best explanations for a lot of LC problems.

3. If you are Chinese reader, please also check out this Youtuber:

   youtube.com/channel/UCE...

   His explanations and ways of solving problems are also golden.
4. Beyond LC, I use educative courses and you can find my experience here:

   dev.to/szl0072/i-got-an...
   or
   qr.ae/TWCOLK

   If you are Chinese users, please check out my experience in Chinese here:

   刷完LeetCode是什么水平，能拿到什么水平的 offer？
5. As suggested from @azeez, I did use codingbat.com/java to practice Java basics. This is also a gold mine to dig into.
6. I used codegym.cc/ to help me understand the ins and outs for Java as well when it was still free. It is a wonderful site but you need to subscribe now, though.
7. On average, I did 3-4 questions from LeetCode daily. As said by so many fellow friends here, we have to practice constantly. Instead of solving a LOT of questions in a day, we should leverage our time wisely so we do not burn out ourselves. This is asked a lot in the comment section, so I update here.

Thanks to LC and I will keep learning!

原文： leetcode.com/discuss/ca...

Title：How i improved my problem solving

点评：刷题技能如何提升之道！

I am writing this post to share my experience while preparing for interviews. Recently i did my onsite at Amazon but couldnt make it.
In the beginning when i started to solve the questions i couldnt even solve the easiest problem in leetcode.I used to google about this problem and the common solution i could find was people asking us try and try for atleast 30mins or more. Initially i tried to follow this approach , even then couldnt solve it or i didnt even the get the basic idea of solving the problem.

Later, i started looking at the solutions and used to solve every problem on paper practically and used to go through the discuss section for every problem and used to read comments. After solving 30-50 problems using this approach i realized that i can see the question and get the basic idea of how to solve the problem. At this stage i was so happy as atleast i got the idea and i continued this and then even if i could solve thre problem i used to go through discuss section . This is how i improved myself.

P.S : This approach worked for me and every individual is diffrent. So, just try it for yourself and see if this works. I can say that this process improved my effeciency by atleast 60%.Hope this helps!!.

原文： leetcode.com/discuss/ca...

Title：Google Interview Tips + FAQs Answered + Resources

点评：Google家的面试，题目活，比较难准备，看看贴主怎么说！详尽的心路历程和题目推荐！

Hi everyone ! I am asquare14, I am an Electronics and Communication Engineering student. I will be interning at Google next summer and I wanted to share a few things that I have learnt during my preparation. In this blog post, I have tried to clear some of the common questions every student has regarding interview preparation. I have also included resources that helped me during my preparation !

Why should you do Leetcode?

Sites like CodeChef, Codeforces, SPOJ are really good but they are not interview/job focused, they lean more towards competitive programming. If your aim is to go compete at ACM ICPC, by all means practice at those sites.

For interview preparation, the top sites are Leetcode, GeeksForGeeks and Interview Bit. Leetcode beats the other two sites handsdown. While the theory in GeeksForGeeks is good (it is often coded inefficienty) and their practice platform has weak testcases. InterviewBit has a good collection of questions but their list is limited, also they do not have a vibrant community.

Leetcode has everything that a good site should, amazing list of questions, good editorials, really good testcases as well as a **community**(best thing). The discuss section has amazing discussions, sometimes the solutions here are better than those in the editorial.

I am intimidated by the list of questions, where should I start ?

First and foremost, there is no reason to be intimidated, we all have to start somewhere :) If you are a complete beginner, start with Top 100 Liked Questions and Top Interview Questions. Sort them by difficulty level and do the easy ones, followed by medium and then hard.

How should I solve each question ?

Everyone has a different way of approaching problems, I initially looked up the solution to every problem but when I gave the Weekly Contests(another amazing feature! more about it later) I realized my technique wasn't working. I then thought about a question and worked out a basic algorithm and then coded. If it took more than a certain amount of time, I saw the solution.

Here is a curated list of "how to leetcode" from various people, do give it a read ! It is helpful :

- Leetcode Grinding Guide.Reddit
- Patterns on Leetcode Problems
- Want to crack Leetcode problems easily ?
- r/cscareerquestions - I have to literally lookup up every leetcode solution. Is it normal?
- how-to-leetcode-effectively

How do I measure my performance, I am solving questions but am I im

question,but with practice I started solving two to three. It motivated me to work harder every week and manage time better.

How "MANY" questions should I solve ?

It is not about the number of questions rather it is about "how" well you understand the concepts and are able to approach new problems. Inspite of that, doing the Top Interview Questions should be a must.

Other tips

- Read as many interview experiences of people as you can from Leetcode Discuss.
- Ask doubts in Leetcode Discuss for each question.
- If your interview is nearing, take Leetcode Premium and do company wise questions (very helpful).

What are the important topics I should study ?

1. **Big O Notation**

- Theory
- Practice problems from Cracking the Coding Interview

1. **Arrays and Maths**

- Practice a lot of Questions on Arrays and Maths. Some important topics are mentioned below.
- Circular Arrays - Typical ways to solve on Leetocode
- Boyer Moore Voting Algorithm Leetcode
- Two Sum Problem - Leetcode
- Three Sum Problem - Leetcode
- Four Sum Problem - Leetcode
- Buy and Stock problem - Leetcode
- Buy and Stock problem II - Leetcode
- Buy and Stock problem III - Leetcode
- Buy and Stock problem IV - Leetcode
- Buy and Stock With Cooldown - Leetcode
- Questions relating to Palindromes.
- Longest Palindromic Subsequence
- Finding square root of a number in logn time - Leetcode

1. **Binary Search**

- Binary Search from Topcoder(MUST)

1. **Bitwise manipulation**

- A summary: how to use bit manipulation to solve problems easily and efficiently - LeetCode Discuss
- Good website to visualize bitwise operations
- geeksforgeeks.org/bits-...

1. **Trees**

- Pre-order(BOTH recursive and iterative)
- Post-order(BOTH recursive and iterative
- In-order(BOTH recursive and iterative
- N-ary Tree Pre-order Traversal
- N-ary Tree Pre-order Traversal
- N-ary Tree Level Order Traversal
- Maximum Depth of N-ary Tree
- Serialization and deserialization of trees - Leetcode
- Binary Search Tree
- Lowest Common Ancestor - Leetcode
- Morris In-order traversal by Tushar Roy (Video)
- Threaded Binary Tree

1. **Recursion and Backtracking**

- Recursion and Backtracking Tutorial
- Blog by csgator(BEST)
- Interview Bit Theory
- Turnpike problem
- Word break Problem Leetcode
- Word break Problem 2 Leetcode
- Letter combinations of a phone-number Leetcode

1. **Graphs**

- Representing graphs
- DFS, BFS Explanation by csgator(BEST)

- Dijikstra
- Dijkstra on sparse graphs - Competitive Programming Algorithms
- Number of Islands
- Friend Circles
- Decode String

1. **Geometry**

- Geometric Algorithms - GeeksforGeeks

1. **Hashing**

- map vs unordered_map in C++ - GeeksforGeeks
- Design HashMap
- Design Hashset
- Sliding Window algorithm template to solve all the Leetcode substring search problem. - LeetCode Discuss (This is important !)
- String Hashing

1. **Linked List**

- Insertion
- Deletion of Node
- Reverse Linked List (iterative and recursive)
- Circular Linked List
- Doubly Linked List
- Floyd's Cycle Detection Algorithm
- Linked List Cycle - Leetcode
- LRU Cache - C++ Implementation - Bhrigu Srivastava
- Copy list with random-pointer (BEAUTIFUL QUESTION!)

1. **Dynamic Programming**

- TopCoder Article (VERY IMPORTANT!!! MUST)
- Top 20 DP questions(Geeks for geeks) Important!
- Tushar Roy DP playlist
- Do Questions from Interview Bit (Good List) :P
- Top 50 DP questions
- Difference between DP and Divide and Conquer

- DSU CP-Algorithm

1. **Sorting**

- Be clear with the basic algorithm and time complexity of all sorting algorithms.
- Additionally read up count sort, bucket sort and radix sort.

1. **Greedy**

- Basics of Greedy

1. **System Design**

- Leetcode System Design Posts
- System Design Videos (Sufficient for prep)
- System Design Resources
- How to succeed in a System Design Interview

For now these are the topics that I feel are important for interview, obviously this is not a exhaustive list :)

Google Interview Tips

- Practice coding on google docs.
- Write variable and function names that are descriptive.
- Be honest. Do not bluff.
- Be vocal, explain your approach while coding.
- Give a lot of mock interviews to friends in real environment.
- Do a lot of leetcode.
- Be confident !

Hope this blog post helped you ! All the best ! Feel free to post any questions :)

原文：leetcode.com/discuss/ca...

Title: Anyone having massive anxiety?

I don't know why, but for the last several months I was having massive anxiety in preparation of upcoming interviews. I'm a veteran engineers (almost 20 years in the industry), and had lost my job this year that was out of my control, and the idea of being asked coding questions drove me crazy. I wish your years of experience, work references, and the fact that you have a college degree actually meant something.

Although I've learned to accept it. I even had to learn relaxation techniques to remain calm. I'll even admit this anxiety has caused me some physical health issue.

Its strange because I've always been a highly productive and outstanding engineering in any role that I've taken (large company or startup).

In any case, I'm curious to hear if you've had this problem. How did you deal with it? How do you see these interviews?

原文： leetcode.com/discuss/ca...

Title: My notes for the night before interview.

点评：刷题刷了一大堆，屠龙技到手。但面试前一晚应该做点什么？

Sharing the notes that I put together myself for the past 2 years after 10+ interviews, 50+ leetcode contests.
They are not intented to solve a specific problem, but rather some principles that I learned from my failures.
Hope it can be somewhat helpful.

I split it into 3 parts: **Coding, System Design, Behavioral.**

Coding

- Make sure to run your solution with couple examples so that interviewer knows what you want to do!

Source of corner cases:
1. Integer : overflow? negative? 0? Duplicated number?
2. LinkedList : circles. Create dummy node is helpful
3. String: ascii char? Only numeric value?
4. When calculate mid-point, always use x + (y-x) / 2. Never use (x+y) / 2
5. Thread safety?

Think and talk
- Brutal force to get started
- Don't use edge cases to affect your algorithm too much. Try to find the intrinsic characteristic of the general cases, then use the edge cases to test your algorithm
- Don't commit to an idea too much, sometimes a problem can be solved in a different way from what seems to be the way at first.
- What to do when you got stuck:
  a. Brutal force
  b. Don't commit to one algorithm too deep
  c. Think the problem from the flip side
  d. Reduce the problem. 2D->1D
  e. Sorting helps
- Tools in the box:
  Tricks: Sweep line, 2 pointer(fast & slow, opposite), DP, sliding window, greedy
  Data structure : Stack, Segment Tree, Dequeue, PQ

Pseudo
Check if the solution works for the edge cases in step 2.

Write
Good variable name
Write according to the pseudo.
Leave enough space between lines when coding,

Test
Walk thru your code, immediately after implementation.
Edge case big and small, randomized test case, thread safe?

System Design

- Structure your design into 5 parts
  - What are the features, and how much request and storage do we need.
  - What are the APIs
  - Hight level architecture
  - Business logic + storage choices and schema
  - Scaling + Stability.
- System design requires a **top-down** mindset. **Don't** worry about the detail and edge case!
- Don't panic. System design is not like coding, you can't design for everything. There will definitely be cases that you didn't handle! Don't let them bother you ahead of time!
- Think first. Speak slowly. Don't give out half-hearted answer pre maturely.
- Move fast in feature and api. Leave time for business logic and storage. Talk about **details** in business logic. So that they believe it will work.
- When challenged, don't argue. If they have a point, say "that's a good point" and move on! Don't insist on one solution, you want to **show** pros and cons

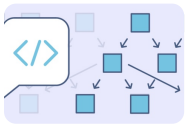
Behavioral

- There are just less than 10 common questions. Definitely prepare ahead of time.
- Bring a watch yourself, you need to control the time.
- Don't be a jerk.

Title: List of questions sorted by common patterns.

点评：这篇文章提到的分类，我在知乎也发了，具体请看：

穷码农：码农找工作之：秒杀算法面试必须掌握的14种模式
zhuanlan.zhihu.com



穷码农：别再埋头刷LeetCode之：北美算法面试的题目分类，按类型和规律刷题...
zhuanlan.zhihu.com

答主把LeetCode上面的题目也按照上面这些分类归类了，笔芯！！！

I found a list of questions sorted by pattern on Hackernoon. This is the link for the article hackernoon.com/14-patte.... I created a LeetCode list for each of the patterns. I found it pretty useful, feel free to make a copy of the list and use it.

- 2 Heaps
- Arrays
- Backtracking
- Dynamic Programming
- Fast & Slow pointers
- Graph Traversal
- In-place traversal of LL
- K-way merge
- Merge Intervals
- Modified Binary Search
- Sliding window
- Top K elements
- Topological Sorting
- Tree BFS
- Tree DFS
- Two Pointers

以上这些模式以及题目的介绍，可以参考这一篇：

Good Luck & Happy Coding!!

原文： leetcode.com/discuss/ca...

Title: What I’ ve learned from tech interviews (my personal tips)

点评1：非常宝贵的刷题面试经验，了解和掌握了这些，准备过程和面试才能事半功倍！！！

点评2：这篇的评论区有作者对上面的点进行解释的回复，大家可以去关注。

This fall, I’ m interviewing for new grad SDE positions with several companies, and had several phone screens and onsite interviews for now. Following is what I gained from my experience, and they’ re generally helpful, so I’ d like to share with you.

1. After the interviewer gave the question, don’ t rush to the solution directly.
   Rushing to the solution is the mistake I made most frequently, even if I knew I should not. Maybe at the beginning of the interview I’ m too nervous. So I put this one as the first rule. The interviewer is expecting you to ask questions, to clarify the problem. In this part, they are testing if you have the ability to clarify the problem, and also your communication intention and skill. Remember, you and your interviewer are collaborating now. It’ s not your or their own game. By asking questions, you need to try to make yourself have the same understanding of the question (the aim of your function, input/output data type, edge cases needed to consider, etc.).
2. In high level, explain your algorithm first, never jump to code without the interviewer’ s agreement.
   In complex questions, think from the brute force solution, and think how to optimal it. IN some cases, the brute force solution is not acceptable by the interviewer, so there is no need to write it out. And also don’ t wait for the best solution if you don’ t have yet. If you find some workaround in the middle, try to write it out first. It’ s always better to have something than have nothing, but remember to get the interviewer’ s agreement before writing code.
3. During your implementation, if the interviewer asks you some questions, stop and
   about it. I know you want to finish your code, but in most cases, the interviewer is
   giving you a hint by asking you questions, maybe because they find a bu
   help you. Never ignore anything from the interviewer. The key of passing

4. Never “argue” about different workaround. The best practice is to write out what’s in the interviewer’s mind. If you have your own workaround, sometimes it’s OK, but you need to convince the interviewer. Remember, be nice. Treat the interviewers as your colleagues.
5. Asking hints directly is not good. You can come up with different solutions and ask for hints implicitly.
6. Never, never, never give up. Even if you don’t know how to solve the problem after maybe 30 minutes, and you have no hope to finish it in the interview, don’t give up. One interviewer told me, in this case, he won’t give hire, but he can also give something as lean hire, which can be balanced with hires in other rounds. So cheer up and leave a good impression.
7. Be a nice person, do good things, help others in your everyday life. And finger crossed, you will be treated nicely in the interviews, you will crack the problems and get HIRE!

Hope my experience is helpful. I’m preparing for dream company’s interviews in January. Hope I can make it! Hope all of us can make it!
Best wishes! Happy holidays!

原文： leetcode.com/discuss/ca...

Title: Offers from Google/Facebook/Apple/Uber/Snap/etc.. after numerous failures

点评：从无数次失败中总结出来的拿无数大公司的经验！

I failed exactly three times each on interviews for Google and Facebook. These were my two top companies I wanted to join since college, and I have finally got offers from both.

College
Before I was in college, or even after I started college, I didn't know what Computer Science was. I chose Electrical and Computer Engineering as a major because it sounded cool and I liked to play with computer. I was very lost when I took first prog

I saw other kids getting internship from big companies while I never even received a chance to get an interview with any of those big companies for internship. I did Masters in CS just to learn more and I loved the learning, and it was time to find my first full-time job.

New-Grad Job Hunting

I started to apply to as many companies I could find, and was lucky to get some interviews. First technical phone screen ever was with Google, and without much preparation, I failed. Facebook came to campus and I was able to interview on campus, and passing a screen interview, I was invited to onsite. There, I failed miserably along with other tech companies in Silicon Valley. More interviews I did, I felt more confident, and I started getting some offers from some smaller companies. When I was thinking to accept offer at a small company in Atlanta, I got an onsite interview chance at Bloomberg. They didn't contact me when I gave them my resume at career fair, but I decided to give second shot by applying directly at the company's website. I passed the phone screen, and I was so nervous since it was my last onsite interview, and it was with one of my desired companies to work for. I struggled, and I was not very confident after onsite, and a week of waiting felt like forever. I still recall the excitement when I got the phone call for the offer..

Second attempt at Google & Facebook

After a year at Bloomberg where I was enjoying, recruiters from Facebook and Google reach out to me about same time. I wasn't too sure about moving during that time, but I was just excited for the contact and just tried to give a shot. I passed to get onsite for both company, but I failed to get an offer from neither.

Programming contest and Discovering Leetcode

I started to gain interest in programming contests and solving algorithm questions. I started to prepare for interviews, but I realized participating in online contest was fun and helped me focus under time pressure. My friend told me about Leetcode and I wished I knew such thing existed earlier! I used it to solve and practice, and it helped me improve a lot.

Third attempt at Google and Facebook, and joining Apple

After being two years at Bloomberg, I was ready to move on. I contacted Google and Facebook recruiters that I talked year before, and also couple more recruiters who reached out to me. I felt much more confident after solving hundreds of problems on Leetcode, and was able to get offer from Apple. I aced algorithm interviews on Facebook, but bombed system design.. And messing up with one of the interviews at Google again. I was satisfied, however, to finally move to Silicon Valley.

leetcode problems for fun and started participating leetcode contest whenever I could on Saturday night. The contest time was not the best, but I enjoyed whenever I could participate, and this really improved and prepared me well for the upcoming interviews!

Final attempt!
After couple years at Apple, I decided, again, to give shot at Google and Facebook. I was not 100% sure after Facebook onsite but felt somewhat confident, but with Google interviews I was very sure I would get an offer. Doing lots of leetcode practice got me prepared to solve algorithm question very quickly and efficiently, and the interview seemed easier than doing the contest. I had 13 onsites with my favorites in three weeks, (Atlanta, Seattle, LA, Chicago, SF, NY), and while it was crazy with all the travelling, it was one of the best time in my life. I actually liked doing interviews after solving lots of leetcode questions!
I received offers from about half of them, including Facebook and Google! Google had been a slight more favor, as it was number 1 on my list, and now I made it here!

Do not give up!
I remember when I first interviewed with all big companies, and I felt pretty dumb and thought I would never make to any of those silicon valley companies.. but I made it! Enjoy writing code, and I honestly think Leetcode is a great platform to practice, and also everyone should definitely try out weekly contest. I still participate in contest for fun whenever I get chance (though now it is much harder here with eastern time zone, doing at 9:30 on Saturday night..)

最后的总结：

以上的这些经验主要是针对美国程序员找工作的，英文自然要求比较高，所以我就不翻译成中文了，大家看看原汁原味的经验，根据自己的实际情况，制定好学习计划，一步步实现自己的上岸梦想吧！

这些帖子的精华之处还在于帖子下面大家无私的评论，里面也包含了很多精华。原本是想连评论一起整理出来，但发现如果那样的话，这篇文章都能成书了。于是只能忍痛不包括评论。特别希望大家能去看评论，从中汲取经验教训。

谢谢大家!

编辑于 2020-01-28

刷题 Online Judge 算法竞赛

文章被以下专栏收录

数据结构和算法-内部培训资料中的部分 关注专栏

推荐阅读

我们在国产龙芯电脑上移植了hustoj

张浩斌

还没有评论

写下你的评论...

▲ 赞同 24 ▼