Tech Interview Hand...

Algorithms study cheatsheets

- Introduction
- Array
- String
- Hash table
- Recursion
- Sorting and searching
- Matrix
- Linked list
- Queue
- Stack
- Tree
- Graph
- Heap
- Trie
- Interval
- Dynamic programming
- Binary
- Math
- Geometry

Tech Interview Hand...

Algorithms study cheatsheets

- Introduction
- Array
- String
- Hash table
- Recursion
- Sorting and searching
- Matrix
- Linked list
- Queue
- Stack
- Tree
- Graph
- Heap
- Trie
- Interval
- Dynamic programming
- Binary
- Math
- Geometry

Dynamic programming cheatsheet for coding interviews

*Last updated on **4/8/2022***

Introduction

Dynamic Programming (DP) is usually used to solve optimization problems. The only way to get better at DP is to practice. It takes some amount of practice to be able to recognize that a problem can be solved by DP.

Learning resources

- Demystifying Dynamic Programming
- Dynamic Programming – 7 Steps to Solve any DP Interview Problem
- Less Repetition, More Dynamic Programming, basecs
- Dynamic Programming, James Aspnes, Yale University

Techniques

Sometimes you do not need to store the whole DP table in memory, the last two values or the last two rows of the matrix will suffice.

Recommended questions

- 0/1 Knapsack or Partition Equal Subset Sum
- Climbing Stairs
- Coin Change
- Longest Increasing Subsequence
- Longest Common Subsequence
- Word Break Problem
- Combination Sum
- House Robber and House Robber II
- Decode Ways
- Unique Paths
- Jump Game

Sorting and searching cheatsheet for coding interviews

*Last updated on **4/8/2022***

Introduction

Sorting is the act of rearranging elements in a sequence in order, either in numerical or lexicographical order, and either ascending or descending.

A number of basic algorithms run in $O(n^2)$ and should not be used in interviews. In algorithm interviews, you're unlikely to need to implement any of the sorting algorithms from scratch. Instead you would need to sort the input using your language's default sorting function so that you can use binary searches on them.

Time complexity

Algorithm	Time	Space
Bubble sort	$O(n^2)$	O(1)
Insertion sort	$O(n^2)$	O(1)
Selection sort	$O(n^2)$	O(1)
Quicksort	O(nlog(n))	O(log(n))
Mergesort	O(nlog(n))	O(n)
Heapsort	O(nlog(n))	O(1)
Counting sort	O(n + k)	O(k)
Radix sort	O(nk)	O(n + k)

Algorithm	Big-O
Binary search	O(log(n))

Things to look out for during interviews

Make sure you know the time and space complexity of the language's default sorting algorithm! The time complexity is almost definitely O(nlog(n))). Bonus points if you can name the sort. In Python, it's Timsort.

Corner cases

- Empty sequence
- Sequence with one element
- Sequence with two elements
- Sequence containing duplicate elements.

Techniques

Sorted inputs

When a given sequence is in a sorted order (be it ascending or descending), using binary search should be one of the first things that come to your mind.

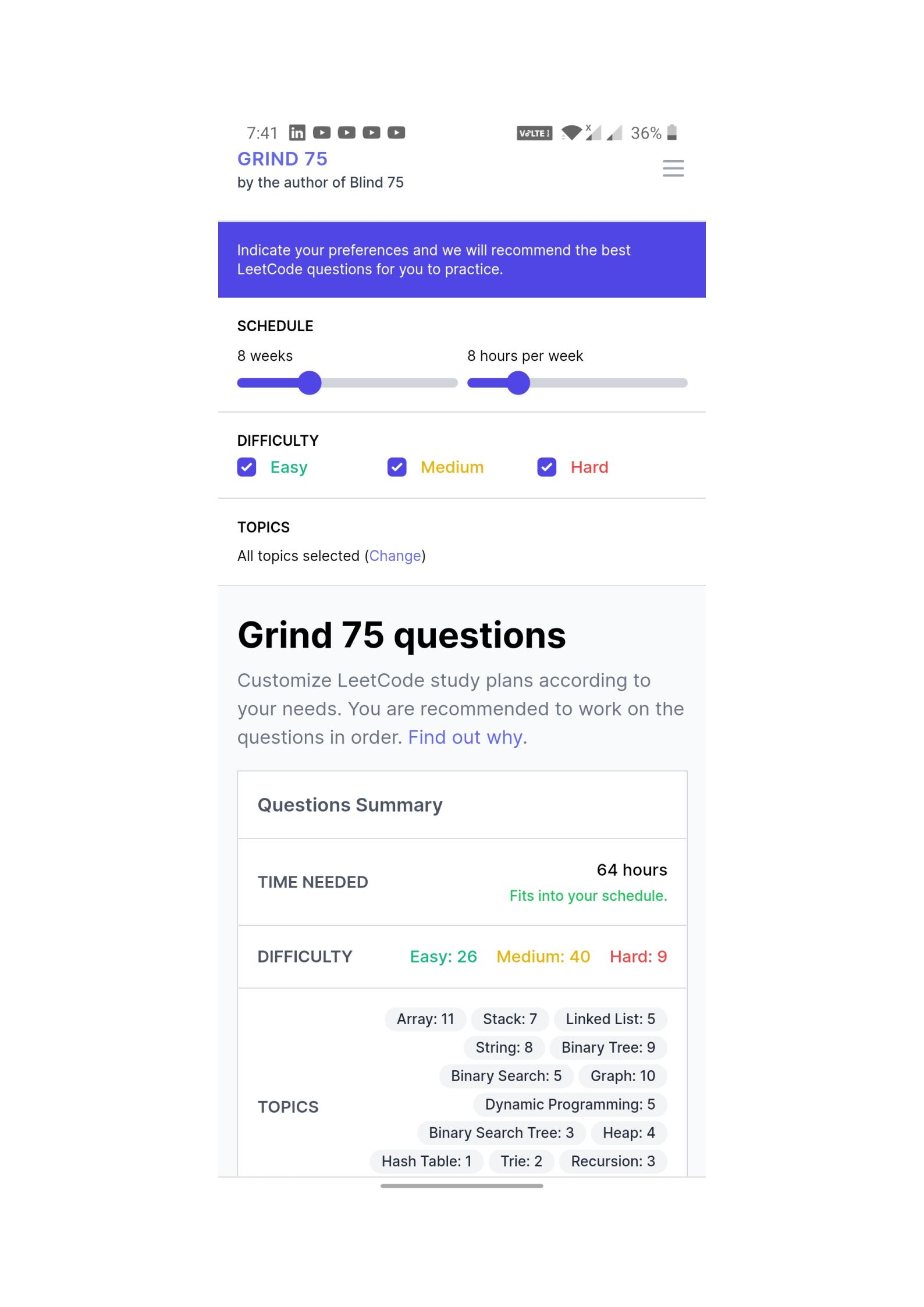
Recommended questions

- Binary Search
- Kth Smallest Element in a Sorted Matrix
- Search in Rotated Sorted Array
- Search a 2D Matrix
- Kth Largest Element in an Array
- Find Minimum in Rotated Sorted Array
- Median of Two Sorted Arrays

How will you be evaluated during a coding interview?

I have collated evaluation criteria across top tech companies and generalized them into a coding interview evaluation rubric you can use. Specific terminology or weightages may differ across companies, top tech companies always include the following criteria in their evaluation:

1. **Communication** - Asking clarifying questions, communication of approach and tradeoffs clearly such that the interviewer has no trouble following.
2. **Problem solving** - Understanding the problem and approaching it systemically, logically and accurately, discussing multiple potential approaches and tradeoffs. Ability to accurately determine time and space complexity and optimize them.
3. **Technical competency** - Translating discussed solutions to working code with no significant struggle. Clean, correct implementation with strong knowledge of language constructs.
4. **Testing** - Ability to test code against normal and corner cases, self-correcting issues in code.


7:41
36%
GRIND 75
by the author of Blind 75
Indicate your preferences and we will recommend the best LeetCode questions for you to practice.
SCHEDULE
8 weeks
8 hours per week
DIFFICULTY
Easy
Medium
Hard
TOPICS
All topics selected (Change)
Grind 75 questions
Customize LeetCode study plans according to your needs. You are recommended to work on the questions in order. Find out why.
Questions Summary
TIME NEEDED
64 hours
Fits into your schedule.
DIFFICULTY
Easy: 26
Medium: 40
Hard: 9
TOPICS
Array: 11
Stack: 7
Linked List: 5
String: 8
Binary Tree: 9
Binary Search: 5
Graph: 10
Dynamic Programming: 5
Binary Search Tree: 3
Heap: 4
Hash Table: 1
Trie: 2
Recursion: 3

Week 1

1	Two Sum Easy · 15 mins · Array	○
2	Valid Parentheses Easy · 20 mins · Stack	○
3	Merge Two Sorted Lists Easy · 20 mins · Linked List	○
4	Best Time to Buy and Sell Stock Easy · 20 mins · Array	○
5	Valid Palindrome Easy · 15 mins · String	○
6	Invert Binary Tree Easy · 15 mins · Binary Tree	○
7	Valid Anagram Easy · 15 mins · String	○
8	Binary Search Easy · 15 mins · Binary Search	○
9	Flood Fill Easy · 20 mins · Graph	○
10	Maximum Subarray Easy · 20 mins · Dynamic Programming	○

Week 2

1	01 Matrix Medium · 30 mins · Graph	○
2	K Closest Points to Origin Medium · 30 mins · Heap	○
3	Balanced Binary Tree Easy · 15 mins · Binary Tree	○
4	Linked List Cycle Easy · 20 mins · Linked List	○
5	Implement Queue using Stacks Easy · 20 mins · Stack	○
6	First Bad Version Easy · 20 mins · Binary Search	○
7	Ransom Note Easy · 15 mins · Hash Table	○
8	Longest Substring Without Repeating Characters Medium · 30 mins · String	○
9	3Sum Medium · 30 mins · Array	○
10	Binary Tree Level Order Traversal Medium · 20 mins · Binary Tree	○

Week 3

1	Clone Graph Medium · 25 mins · Graph	○
2	Evaluate Reverse Polish Notation Medium · 30 mins · Stack	○
3	Course Schedule Medium · 30 mins · Graph	○
4	Implement Trie (Prefix Tree) Medium · 35 mins · Trie	○
5	Coin Change Medium · 25 mins · Dynamic Programming	○
6	Product of Array Except Self Medium · 30 mins · Array	○
7	Climbing Stairs Easy · 20 mins · Dynamic Programming	○
8	Longest Palindrome Easy · 20 mins · String	○
9	Min Stack Easy · 20 mins · Stack	○

Week 4

1	Reverse Linked List Easy · 20 mins · Linked List	○
2	Validate Binary Search Tree Medium · 20 mins · Binary Search Tree	○
3	Number of Islands Medium · 25 mins · Graph	○
4	Rotting Oranges Medium · 30 mins · Graph	○
5	Search in Rotated Sorted Array Medium · 30 mins · Binary Search	○
6	Combination Sum Medium · 30 mins · Array	○
7	Permutations Medium · 30 mins · Recursion	○
8	Merge Intervals Medium · 30 mins · Array	○
9	Lowest Common Ancestor of a Binary Tree Medium · 25 mins · Binary Tree	○

About Grind 75

Hi there! I'm the author of Blind 75, a curated list of LeetCode questions that you can practice to save time in your job hunt. For some reason, this list of questions blew up in popularity and people have been asking for a Blind 75 2.0, so here it is!

What are the problems with Blind 75?

Blind 75 was created 5 years during my last job hunt. Although it is no longer that new, most of the questions are still relevant! However, apparently 75 questions is too few for some folks and they have been asking for more questions to practice beyond the initial Blind 75 list. Also, the Blind 75 list is static, non-personalized and questions do not have a priority. It does not take into account how much time you are willing to spend on practicing, which topics you want to focus on, and tell you which are the more important questions in that 75. I have been thinking about how to solve this for a few months now.

Introducing Grind 75

Grind 75 is Blind 75 reimagined for the future. I looked at all the top LeetCode questions (by popularity, frequency, topics covered, like rating) and picked the top 169 questions which I felt were of value. Within those questions, I ranked them again and assigned a priority to them according to the value it provides to the user. With a priority for each question, I selected the top 75 to be the new Blind 75, which I call Grind 75.

However, it doesn't end at 75 questions! As mentioned, I've selected 169 questions and you can go past the top 75 questions if you like and have the time. You probably don't need to do beyond the 169 questions.

What's new in Grind 75?

As mentioned earlier, Grind 75 is a dynamic list of recommended LeetCode questions which you can personalize according to your schedule, time constraints, and preferences. You can:

- **No premium questions** in the initial list of 75 questions
- Select how many hours you want to spend on practicing
- Select the difficulty of questions you want to practice
- Select specific topics you want to practice/ignore
- View the questions categorized by topics/difficulty/weeks

Check out the Frequently Asked Questions to know more about the thinking behind this list.

Credits

- AlgoMonster - Their top patterns to conquer the Technical Coding Interview page was helpful in prioritizing questions
- Sean Prashad's LeetCode Patterns - Sean's list is great and I tried to maintain feature parity as much as possible.
- Elements of Programming Interviews - I wish I knew about this book earlier. This