



April 13, 2022 ▪ Interviews

## Striver's SDE Sheet – Top Coding Interview Problems



Striver's SDE Sheet

(Striver's SDE Sheet – Sheet for the sole purpose of quick revision and preparation in less time focusing on top coding interview problems)

Made with [love](#) by takeUforward!

## What is Striver SDE Sheet?

SDE Sheet contains very handily crafted and picked top coding interview questions from different topics of Data Structures & Algorithms. These questions are one of the most asked coding interview questions in coding interviews of companies like Amazon, Microsoft, Media.net, Flipkart, etc, and cover almost all of the concepts related to Data Structure & Algorithms.

## Why trust the Striver SDE sheet?

This sheet is prepared by Raj Vikramaditya A.K.A Striver, Candidate Master, 6\*, who has bagged offers from **Google** Warsaw, **Facebook** London, **Media.net**(Directi). He has also interned at **Amazon** India. He is also one of the top educators at Unacademy and was at GeeksforGeeks as well. Not only this, hundreds of students cleared interviews of top companies with the help of this sheet. What are you waiting for?

problems, because these problems are solely interview-based.

Share on Whatsapp

**Note:** If you find the sheet useful, you can also contribute an article or solution for any problem to be published on takeuforward.org! [Click here for more details.](#)

Day 1: Arrays

×

Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Set Matrix Zeroes</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Pascal's Triangle</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Next Permutation</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Kadane's Algorithm</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Sort an array of 0's 1's 2's</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Stock buy and Sell</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 2: Arrays Part-II

×

Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Rotate Matrix</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Merge Overlapping Subintervals</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Merge two sorted Arrays without extra space</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Find the duplicate in an array of N+1 integers.</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Repeat and Missing Number</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Inversion of Array.(Pre-req: Merge Sort)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 3: Arrays Part-III

×

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Search in a 2d Matrix</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Pow(X,n)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Majority Element (&gt;N/2 times)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Majority Element (&gt;N/3 times)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Grid Unique Paths</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Reverse Pairs (Leetcode)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 4: Arrays Part-IV

×

Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">2-Sum-Problem</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">4-sum-Problem</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Longest Consecutive Sequence</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Largest Subarray with 0 sum</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Count number of subarrays with given Xor K</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Longest Substring without repeat</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 5: Linked List

×

Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Reverse a LinkedList</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Find the middle of LinkedList</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Merge two sorted Linked List (use method used in mergeSort)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Remove N-th node from back of LinkedList</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

[given.\(O\(1\) solution\)](#)

Day 6: Linked List Part-II



Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Find intersection point of Y LinkedList</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Detect a cycle in Linked List</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Reverse a LinkedList in groups of size k.</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Check if a LinkedList is palindrome or not.</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Find the starting point of the Loop of LinkedList</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Flattening of a LinkedList</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 7: Linked List and Arrays



Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Rotate a LinkedList</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Clone a Linked List with random and next pointer</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">3 sum</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Trapping rainwater</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Remove Duplicate from Sorted array.</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Max consecutive ones</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 8: Greedy Algorithm



Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
---------	--------------------	-------------------	--------------------



<a href="#">Minimum number of platforms required for a railway</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Job sequencing Problem</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Fractional Knapsack Problem</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Greedy algorithm to find minimum number of coins</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Activity Selection (it is the same as N meeting in one room)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 9: Recursion

×

Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of recursion.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Subset Sums</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Subset-II</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Combination sum-1</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Combination sum-2</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Palindrome Partitioning</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">K-th permutation Sequence</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 10: Recursion and Backtracking

×

Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of recursion.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Print all permutations of a string/array</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">N queens Problem</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Sudoku Solver</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">M coloring Problem</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Rat in a Maze</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Word Break (print all ways)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">The N-th root of an integer</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Matrix Median	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Find the element that appears once in a sorted array, and the rest element appears twice (Binary search)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Search element in a sorted and rotated array/ find pivot where it is rotated</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Median of 2 sorted arrays</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">K-th element of two sorted arrays</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Allocate Minimum Number of Pages</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Aggressive Cows</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 12: Heaps

×

Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
Max heap, Min Heap Implementation (Only for interviews)	<a href="#">Link 1</a>	NA	NA
<a href="#">Kth Largest Element</a>	<a href="#">Link 1</a>	NA	<a href="#">Link 2</a>
Maximum Sum Combination	<a href="#">Link 1</a>	NA	<a href="#">Link 2</a>
Find Median from Data Stream	<a href="#">Link 1</a>	NA	<a href="#">Link 2</a>
Merge K sorted arrays	<a href="#">Link 1</a>	NA	<a href="#">Link 2</a>
K most frequent elements	<a href="#">Link 1</a>	NA	<a href="#">Link 2</a>

Day 13: Stack and Queue

×

Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Implement Stack Using Arrays</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Implement Queue Using Arrays</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>



<a href="#">single queue)</a>			
<a href="#">Implement Queue using Stack (O(1) amortized method)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Check for balanced parentheses</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Next Greater Element</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Sort a Stack	<a href="#">Link 1</a>	YT	Link 2

## Day 14: Stack and Queue Part-II



Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
Next Smaller Element	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
<a href="#">LRU cache (IMPORTANT)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
LFU Cache	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Largest rectangle in a histogram</a>	<a href="#">Link 1</a>	<a href="#">YT1/YT2</a>	<a href="#">Link 2</a>
<a href="#">Sliding Window maximum</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Implement Min Stack</a>	Link 1	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Rotten Orange (Using BFS)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Stock Span Problem	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Find the maximum of minimums of every window size	<a href="#">Link 1</a>	YT	Link 2
The Celebrity Problem	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>

## Day 15: String



Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Reverse Words in a String</a>	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Longest Palindrome in a string	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Roman Number to Integer and vice versa	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Implement ATOI/STRSTR	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Longest Common Prefix	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>



## Day 16: String Part-II



Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
Z-Function	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
KMP algo / LPS(pi) array	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Minimum characters needed to be inserted in the beginning to make it palindromic	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Check for Anagrams	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Count and Say	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Compare version numbers	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>

## Day 17: Binary Tree



Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of Binary Trees.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Inorder Traversal</a>	<a href="#">Link 1</a>	<a href="#">YT1</a> / <a href="#">YT2</a>	<a href="#">Link 2</a>
<a href="#">Preorder Traversal</a>	<a href="#">Link 1</a>	<a href="#">YT1</a> / <a href="#">YT2</a>	<a href="#">Link 2</a>
<a href="#">Postorder Traversal</a>	<a href="#">Link 1</a>	<a href="#">YT1</a> / <a href="#">YT2</a>	<a href="#">Link 2</a>
<a href="#">Morris Inorder Traversal</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Morris Preorder Traversal</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">LeftView Of Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Bottom View of Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Top View of Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Preorder inorder postorder in a single traversal</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Vertical order traversal</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Root to node path in a Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Max width of a Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

## Day 18: Binary Tree part-II





I will recommend you to do [this](#) playlist at first, so that you learn A-Z of Binary Trees.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Level order Traversal / Level order traversal in spiral form</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Height of a Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Diameter of Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Check if the Binary tree is height-balanced or not</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">LCA in Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Check if two trees are identical or not</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Zig Zag Traversal of Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Boundary Traversal of Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 19: Binary Tree part-III

×

Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of Binary Trees.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Maximum path sum</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Construct Binary Tree from inorder and preorder</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Construct Binary Tree from Inorder and Postorder	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Symmetric Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Flatten Binary Tree to LinkedList</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Check if Binary Tree is the mirror of itself or not	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
<a href="#">Check for Children Sum Property</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	Link 2

Day 20: Binary Search Tree

×

Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of Binary Trees.



	Link 1	Solution	Link 2
Populate Next Right pointers of Tree	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Search given Key in BST	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Construct BST from given keys	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Construct BST from preorder traversal	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Check is a BT is BST or not	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Find LCA of two nodes in BST	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Find the inorder predecessor/successor of a given Key in BST.	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 21: Binary Search Tree Part-II

×

Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of Binary Trees.

Problem	Practice Link 1	Video Solution	Practice Link 2
Floor in a BST	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Ceil in a BST	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Find K-th smallest element in BST</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Find K-th largest element in BST</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Find a pair with a given sum in BST	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
BST iterator	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Size of the largest BST in a Binary Tree	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Serialize and deserialize Binary Tree</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 22: Binary Trees[Miscellaneous]

×

Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of Binary Trees.

Problem	Practice	Video	Practice
---------	----------	-------	----------



Find median in a stream of running integers.	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
K-th largest element in a stream.	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Distinct numbers in Window.	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
K-th largest element in an unsorted array.	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Flood-fill Algorithm	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>

Day 23: Graph

×

Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of Graphs.

Problem	Practice Link 1	Video Solution	Practice Link 2
Clone a graph (Not that easy as it looks)	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
<a href="#">DFS</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">BFS</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Detect A cycle in Undirected Graph using BFS</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Detect A cycle in Undirected Graph using DFS</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Detect A cycle in a Directed Graph using DFS</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Detect A cycle in a Directed Graph using BFS	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Topological Sort BFS</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Topological Sort DFS</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Number of islands(Do in Grid and Graph Both)	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Bipartite Check using BFS	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Bipartite Check using DFS</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 24: Graph Part-II

×

Find both C++/Java codes of all problem in the articles in the first column.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Strongly Connected Component(using KosaRaju's algo)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Dijkstra's Algorithm</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Bellman-Ford Algo</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Floyd Warshall Algorithm	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
<a href="#">MST using Prim's Algo</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">MST using Kruskal's Algo</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 25: Dynamic Programming

×

Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of DP.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Max Product Subarray</a>	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Longest Increasing Subsequence	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Longest Common Subsequence</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">0-1 Knapsack</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Edit Distance</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Maximum sum increasing subsequence	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Matrix Chain Multiplication	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 26: Dynamic Programming Part-II

×

Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of DP.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Minimum sum path in the matrix, (count paths and similar type do, also backtrack to find the Minimum path)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

<a href="#">Rod Cutting</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Egg Dropping	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Word Break	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>
Palindrome Partitioning (MCM Variation)	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Maximum profit in Job scheduling	<a href="#">Link 1</a>	YT	<a href="#">Link 2</a>

Day 27: Trie

×

Find both C++/Java codes of all problem in the articles in the first column.

I will recommend you to do [this](#) playlist at first, so that you learn A-Z of Tries.

Problem	Practice Link 1	Video Solution	Practice Link 2
<a href="#">Implement Trie (Prefix Tree)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Implement Trie – 2 (Prefix Tree)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Longest String with All Prefixes	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Number of Distinct Substrings in a String</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Power Set (this is very important)</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
<a href="#">Maximum XOR of two numbers in an array</a>	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>
Maximum XOR With an Element From Array	<a href="#">Link 1</a>	<a href="#">YT</a>	<a href="#">Link 2</a>

Day 28: Operating System Revision (Refer [Sheet](#) for OS Questions)

×

1. Revise OS notes that you would have made during your sem
2. If not made notes, spend 2 or 3 days and make notes from Knowledge Gate.

Day 29: DBMS Revision (Refer [Sheet](#) for DBMS Questions)

×

1. Revise DBMS notes that you would have made during your sem
2. If not made notes, spend 2 or 3 days and make notes from Knowledge Gate.

Day 30: Computer Networks Revision (Refer [Sheet](#) for CN Questions)

×

1. Revise CN notes that you would have made during your sem

Make a note of how will your represent your projects, and prepare all questions related to tech which you have used in your projects. Prepare a note which you can say for 3-10 minutes when he asks you that say something about the project.

Hurrah!! You are ready for your placement after a month of hard work without a cheat day.

— ~Striver

Share the sheet with your friends, created with love for takeUforward fam!

Share on Whatsapp

« Previous Post  
**For loop in Java**

Next Post »  
**Sliding Window Technique**

Load Comments