

## **Data Structures:**

Bit manipulation, linked list, trees, stacks and queues, strings, arrays, heaps, sorting, graphs

## **Algorithms :**

Binary search, divide and conquer, dynamic programming, computational geometry, maths, miscellaneous

## **References:**

Go through all the geeks for geeks links provided in this pdf and practise questions from leet code.

<https://leetcode.com/problemset/all/>

## **SORTING**

<https://www.geeksforgeeks.org/merge-sort/>

<http://geeksforgeeks.org/iterative-merge-sort/>

<https://www.geeksforgeeks.org/quick-sort/>

<https://www.geeksforgeeks.org/counting-sort/>

<https://www.geeksforgeeks.org/insertion-sort/>

<https://www.geeksforgeeks.org/heap-sort/>

<https://www.geeksforgeeks.org/bubble-sort/>

(Sorting Cont'd later )

## **BINARY SEARCH**

[Binary Search Example](#)

[Search an element in a sorted and pivoted array](#)

[Check for Majority element in a Sorted array](#)

[Floor and Ceiling in a sorted array](#)

[Find the Minimum length Unsorted Subarray, sorting which makes the array sorted](#)

[Count the number of occurrences of x in a sorted array](#)

[Find a Fixed Point \( where  \$a\[i\]\$  equals  \$i\$  \) in a given array](#)

[Find the maximum element in an array which is first increasing and then decreasing](#)

[Merge two sorted arrays of size N and M](#)

[Median of two sorted arrays](#)

[Longest Increasing Subsequence in  \$O\(N \log N\)\$](#)

[Find the minimum element/\(or search for an element\) in a sorted and rotated array](#)

[Given an array of infinite size containing 0/1 only and in sorted order, find position of first 1](#)

---

## **DIVIDE AND CONQUER**

[Introduction](#)

[Calculate  \$\text{pow}\(X, N\)\$  in  \$O\(\log N\)\$](#)

[Find the N-th Fibonacci Number in  \$O\(\log N\)\$](#)

[Closest Pair of Points -  \$O\(N \log N\)\$](#)

[Closest Pair of Points -  \$O\(n \log n\)\$  Implementation](#)

[Maximum Subarray Sum in  \$O\(N \log N\)\$](#)

---

## **BIT MANIPULATIONS**

[Check if a given number is a power of 2](#)

[Reverse bits of a number](#)

[Count set bits in an integer](#)

[Count number of set bits to be flipped to convert A to B](#)

[Rotate bits of an integer](#)

[Compute the absolute value \(abs\) without branching](#)

[Turn off the rightmost set bit](#)

[Add two numbers without using arithmetic operators](#)

[Position of the right most set bit](#)

[Swap every consecutive odd and even positioned bit in a number](#)

[Find the position of the only set bit](#)

[Perform nibble wise swap in a byte of data](#)

## **SORTING**

[Stability](#)

[Lower bound for comparison based sorting algorithms](#)

[External Sorting](#)

---

## **DYNAMIC PROGRAMMING**

[Overlapping Subproblems Property](#)

[Optimal Substructure Property](#)

[0-1 Knapsack Problem](#)

[Min Cost Path](#)

[Minimum number of jumps to reach end](#)

[Maximum size square sub matrix with all 1s](#)

[Matrix Chain Multiplication](#)

[Coin Change](#)

[Longest Common Substring](#)

[Longest Increasing Subsequence](#)

[Maximum Sum Increasing Subsequence](#)

[Box Stacking Problem](#)

[Rod Cutting](#)

[Minimum insertions to form a palindrome](#)

[Longest Palindromic Substring](#)

[Longest Palindromic Subsequence](#)

[Palindrome Partitioning](#)

[Dice Throw](#)

[Maximum sum rectangle in a 2D matrix](#)

[Largest Independent Set Problem](#)

[Egg Dropping Puzzle](#)

[Optimal BST](#)

[Find if a string is interleaved of two other strings](#)

[Optimum Strategy to maximise coins to collect from either ends](#)

---

## **LINKED LIST**

[Nth node from the end of a Linked list](#)

[Reverse a Linked List](#)

[Recursive function to print reverse of a Linked List](#)

[Check if a singly linked list is a Palindrome](#)

[Delete Linked list](#)

[Detect loop in a Linked List](#)

[Detect and remove loop in a Linked List](#)

[Middle of a Linked list](#)

[Find the intersection point of two Linked Lists](#)

[Intersection of two Sorted Linked Lists](#)

[Union and Intersection of two Linked Lists](#)

[Delete Alternate Nodes of a Linked List](#)

[Rotate a Linked List by K nodes](#)

[Reverse a Linked List in groups of given size K](#)

[Reverse alternate K nodes in a Singly Linked List](#)

[Merge two sorted linked lists](#)

[Sort a linked list of 0s, 1s and 2s](#)

[Segregate even and odd nodes in a Linked List](#)

[Move vowels to end of Linked List maintaining the order](#)

[Alternating split of a given Singly Linked List](#)

[Find a triplet from three linked lists with sum equal to a given number](#)

[Add two numbers represented by linked lists](#)

[Product of two numbers given in Linked List](#)

[Pairwise swap elements of a given linked list](#)

[Swap Kth node from beginning with Kth node from end in a Linked List](#)

[Delete N nodes after M nodes of a linked list](#)

[Swap odd and even nodes in a Linked List](#)

[Merge a linked list into another linked list at alternate positions](#)

[Merge Sort for Linked Lists](#)

[Flattening a Linked List](#)

[Copy a linked list with next and arbit pointer](#)

[Memory Efficient DLL using XOR](#)

[Memory Efficient DLL](#)

---

## **TREES**

[Size of a Tree](#)

[Height of a Tree](#)

[Diameter of a Binary Tree](#)

[Maximum width of a binary tree](#)

[Check if Two Trees are Identical](#)

[Tree Traversal](#)

[Populate Inorder Successor for all nodes](#)

[Level Order Traversal](#)

[\*\*Connect nodes at same level\*\*](#)

[Level Order Traversal in Spiral Form](#)

[Reverse Level Order Traversal](#)

[Vertical Sum in a given Binary Tree](#)

[Difference between sums of odd level and even level nodes of a Binary Tree](#)

[\*\*Check if all leaves are in same level or not\*\*](#)

[Delete a tree](#)

[Zig-Zag Traversal of Tree](#)

[\*\*Boundary Traversal of binary tree\*\*](#)

[Count leaf nodes in a binary tree](#)

[Lowest Common Ancestor in a BINARY Tree](#)

[Print nodes at distance K from root](#)

[Print Left View of a Binary Tree](#)

[In a binary tree, if parent is 0, then left child is 0 and right child is 1. if parent is 1, then left child is 0 and right child is 1](#)

[kth node value which is present at Nth level](#)

[Convert the given Binary tree to its Double tree](#)

[Find the node with minimum value in a BST](#)

[Add all greater values to every node in a given BST](#)

[Inorder Successor in BST](#)

[Lowest Common Ancestor in a BST](#)

[Check if two trees are Isomorphic](#)

[Check if a given binary tree is SumTree or not](#)

[Check if given Binary Tree is BST or not](#)

[Check if Binary Tree is Balanced or not](#)

[Check if given BT is Complete Binary Tree or not](#)

[Check if given Binary Tree can be Folded or not](#)

[Convert Tree to its Mirror Tree](#)

[Convert a given tree to its Sum Tree](#)

[Check for children sum property](#)

[Convert a BT to a tree that holds children sum property](#)

[Convert a BST to a Binary Tree such that sum of all greater keys is added to every key](#)

[Find k-th smallest element in BST](#)

[Find pair of numbers in a BST adding upto K](#)

[Two nodes of a BST are swapped, correct the BST](#)

[Print BST keys in the given range](#)

[Remove BST keys outside the given range](#)

[Remove all nodes in a BST which lie on a path having sum less than k](#)

[Print all root-to-leaf Paths](#)

[Root to leaf path sum equal to a given number](#)

[Find the maximum sum leaf to root path in a Binary Tree](#)

[Print Ancestors of a given node](#)

[Print ancestors of a given binary tree node without recursion](#)

[BST to DLL](#)

[Sorted DLL to Balanced BST](#)

[Sorted Array to Balanced BST](#)

[Sorted Linked List to Balanced BST](#)

[Merge Two Balanced Binary Search Trees](#)

[Serialisation - storing a BT in a file](#)

[Construct Tree from given Inorder and Preorder traversals](#)

[Construct Special Binary Tree from given Inorder traversal](#)

[Construct a special tree from given preorder traversal](#)

[Construct Full Binary Tree from given preorder and postorder traversals](#)

[Construct Tree from Ancestor Matrix](#)

[Construct Ancestor Matrix from Tree](#)

[Find the largest BST subtree in a given Binary Tree if entire subtree has to be taken](#)

[Find the largest BST subtree in a given Binary Tree if part of subtree can also be taken](#)

[Find the maximum weight node in a tree if each node is the sum of the weights all the node](#)

[Morris Inorder Traversal - Threaded binary Trees](#)

[Ternary Search Tree](#)

[TRIE](#)

## **STACKS & QUEUES**

[Implement two stacks in one array](#)

[Implement Stack using Queues](#)

[Implement Queue using Two Stacks](#)

[Implement stack with push\(\), pop\(\), getMin\(\) \[ each in  \$O\(1\)\$  time \]](#)

[Design a stack with operations on middle element](#)

[Check for balanced parentheses in an expression](#)

[Expression Evaluation with operator priority and multiple braces](#)

[Implement LRU Cache](#)

[The Stock Span Problem](#)

[Print the First Greater Element on the right side for each element](#)

[Largest Rectangular Area in a Histogram](#)

[Implement three stacks in one array](#)

[Find maximum element in every window of size K in an array](#)

## **STRINGS**

[Print reverse of a string using recursion](#)

[Print all permutations of a string](#)

[Given a string find its first non-repeating character](#)

[Reverse words in a given string](#)

[Print all the duplicates in the input string](#)

[Move all even-index positioned chars to end of string maintaining even-odd order](#)

[Find Lexicographic rank of a string](#)

[Run Length Encoding](#)

[Implement atoi function](#)

[Print the first unique character in a string](#)

[Write strcmp function and returns -1 if  \$s1 < s2\$ , 0 if  \$s1 = s2\$ , else returns 1](#)

[Remove from string  \$s1\$ , all the characters that are present in string  \$s2\$ .](#)

[Check whether two strings are anagram of each other](#)

[Length of the longest substring without repeating characters](#)

[Find the smallest window in a string containing all characters of another string](#)

[Recursively remove all adjacent duplicates](#)

[Evaluate a regular expression  \$a^\*b^?c\$  with  \$aaaabcc\$](#)

[String Matching - KMP Algorithm](#)

---

## **ARRAYS**

[Find Union and Intersection of two sorted arrays](#)

[Find the Number Occurring Odd Number of Times](#)

[Find missing number from array of N-1 numbers in the range 1 to N](#)

[Find the two non-repeating elements in an array of repeating elements](#)

[Find the two numbers with odd occurrences in an unsorted array](#)

[Find the next smallest palindrome](#)

[find the next higher permutation of the given number as an array of digits. If such a number doesn't exist, return -1.](#)

[Find pair of numbers with given sum X](#)

[Find a,b,c such that  \$a^2+b^2=c^2\$](#)

[Find a triplet that sum to a given value](#)

[Find four elements that sum to a given value](#)

[Find two repeating elements in a given array](#)

[Find the 3 elements such that  \$a\[i\] < a\[j\] < a\[k\]\$  and  \$i < j < k\$](#)

[Find the least positive number missing in an unsorted array.](#)

[Find the row with maximum number of 1s in a 2D row-wise sorted matrix](#)

[Find Maximum difference between two elements such that the larger element appears after the smaller element in array](#)

[Find two numbers such that their difference is minimum](#)

[Find two elements whose sum is closest to zero](#)

[Find the first subarray which has a zero sum in an array](#)

[Find duplicates in  \$O\(n\)\$  time](#)

[Find points in an array where left-sum==right-sum](#)

[Search a number in a row wise and column wise sorted 2D matrix](#)

[Print matrix spirally](#)

[Measure amount of water in j'th glass of i'th row of glasses arranged like a pyramid](#)

[Construct Product Array without division operator: each element = product of elements in arr\[\] except arr\[i\]](#)

[Shuffle a given array](#)

[Sort elements by frequency](#)

[Segregate Even and Odd numbers](#)

[Segregate 0s and 1s](#)

[Sort an array of 0s, 1s and 2s](#)

[Move all zeroes to end of array](#)

[Rearrange positive and negative numbers alternatively](#)

[Given an array \[a1b2c3d4\] convert to \[abcd1234\]](#)

[Maximum and minimum of an array using minimum number of comparisons](#)



[Given binary 2D Matrix, for all cells as 1, set corresponding row and column as 1](#)

[Turn an image by 90 degrees](#)

[Inplace M x N size matrix transpose](#)

[Intersection of n sets](#)

[Print Matrix Diagonally](#)

[Rotate an array by d elements](#)

[Largest Sum Contiguous Subarray](#)

[Maximum Product Subarray](#)

[Maximum Length Bitonic Subarray](#)

[Find continuous subarray with given sum](#)

[Largest subarray with equal number of 0s and 1s](#)

[Maximum subsequence sum such that no two elements are adjacent](#)

[Find the majority element \(with frequency  \$> N/2\$ \)](#)

[Find the maximum repeating number](#)

[Count the number of Inversions in an array](#)

[Find kth smallest element](#)

[Stock Buy Sell to Maximize Profit](#)

[Print the elements greater than all the elements to its right](#)

[calculate the area of water collected by rain holded by bar graph/histogram](#)

---

## **Graphs**

[Graph representations](#)

[Depth First Traversal for a Graph](#)

[Breadth First Traversal for a Graph](#)

[Detect Cycle in a Directed Graph](#)

[Find if there is a path between two vertices in a directed graph](#)

[Find number of connected components in an undirected graph](#)

[Bellman Ford Algorithm](#)

[Floyd Warshall Algorithm](#)

[Kruskal's MST](#)

[Dijkstra's Shortest Path Algorithm](#)

[Union Find](#)

[Union Find by rank](#)

[Topological Sorting for DAG](#)

[Detect cycle in an undirected graph](#)

[Strongly Connected Components](#)

[Shortest Path in Directed Acyclic Graph](#)

[Maximum Bipartite Matching](#)

[Check if Bipartite Graph](#)

[Stable Marriage Problem](#)

[Longest Path in a Directed Acyclic Graph](#)

[Find maximum number of edge disjoint paths between two vertices](#)

[Graph Coloring](#)

[Travelling Salesman Problem](#)

---

### **COMPUTATIONAL GEOMETRY**

[Check whether a given point lies inside a triangle or not](#)

[check if two given line segments intersect](#)

[check if a given point lies inside or outside a polygon](#)

[Convex Hull | Set 1 \(Jarvis's Algorithm or Wrapping](#)

[Given n line segments, find if any two segments intersect](#)

### **MATHS**

[Binomial Coefficient  \$nCr\$](#)

[Pascal's Triangle in nth row](#)

[Select a random number from stream, with  \$O\(1\)\$  space](#)

### **MISCELLANEOUS**

[Little and Big Endian](#)

[Memory Leak](#)

[Greedy Algorithms | Set 1 \(Activity Selection Problem\)](#)

[Print all subsets](#)

[Make a fair coin from a biased coin](#)

[Find the first circular tour that visits all petrol pumps](#)

---

### **HEAP**

[Sort a nearly sorted \(or K sorted\) array](#)

[Find the k most frequent words from a file / running stream of numbers](#)

[Sort numbers stored on different machines](#)

[Huffman Coding](#)

[Program to print last 10 lines of a file](#)

[Merge k sorted arrays](#)

[Find a median in running stream of numbers.](#)

---

---

## **C++**

OOPS concepts like:

Inheritance, Encapsulation, Abstraction, Polymorphism

virtual function, friend function

function overloading, overriding

constuctor, templates

exception handling

storage classes,type qualifiers, modifiers,

macros,inline

memory allocation

reference, pointers

---

**\*STRICTLY NOT FOR REPRODUCTION**