Topic 1: Arrays

1. Sorting
    a. O(n^2) algorithms
        i. Bubble Sort
        ii. Selection Sort
        iii. Insertion Sort
    b. O(n*logn) algorithms
        i. Merge Sort
        ii. Quick Sort
    c. O(n) algorithms
        i. Counting Sort
        ii. Bucket Sort
    d. Variations of Merge Sort
        i. Inversion Count
    e. Variations of Partitioning in Quicksort
        i. Rearranging the numbers of array to get alternate positive-negative pattern
        ii. Rearranging the numbers of array to get alternate odd-even pattern
        iii. Pushing all zeros to the end
        iv. Dutch National Flag Problem
        v. Finding Median in O(n) time
    f. Using sort() function of STL
        i. Writing a custom compare function to sort strings based on the length
        ii. Writing a custom compare function to sort using multiple keys. Example, strings of the same length should be sorted lexicographically
        iii. Achieving stable sorting using STL's sort() function
2. Searching
    a. Binary Search
        i. To find the element in the sorted array
        ii. To find the lower and upper bounds of an element in the sorted array
    b. Variations of Binary Search
        i. Finding a peak element in 1D and 2D arrays.
        ii. Given two sorted arrays, find the median of the merged array
        iii. Aggressive Cows Problem
        iv. Painter's Partition Problem

3. Rotations
    a. Array rotation
        i. Rotate the array using reversal algorithm
        ii. Print the rotated array without actually rotating the array
    b. Search in sorted and rotated array
        i. Find the pivot element in the sorted & rotated array
        ii. Binary search over sorted & rotated array
4. Sliding Window Technique
    a. Fixed-sized window problems
        i. Maximum sum subarray of size ‘k’
        ii. Maximum element in every subarray of size ‘k’
        iii. First negative element in every subarray of size ‘k’
        iv. Count occurrences of an anagram of a pattern in a string
    b. Variable-sized window problems
        i. Largest subarray with sum ‘k’
        ii. Largest substring with ‘k’ distinct characters
        iii. Largest substring with no repeating characters
        iv. Minimum window substring
5. Difference & Prefix Sum Arrays (CP Topic)
    a. Basics
        i. What is a difference array & prefix sum array
        ii. Analogy with calculus
        iii. How to use difference array & prefix sum array
        iv. Extending this concept to multiple dimensions
    b. Difference array problems
        i. CCC 2009 - Wireless
        ii. CCC 2011 - The cake is a dessert
        iii. 295A - Greg and Array
        iv. 816B - Karen and Coffee
        v. 276C - Little Girl & Maximum Sum
        vi. 1343D - Constant Palindrome Sum
    c. Prefix sum problems
        i. Leetcode - Trapping Rain Water
        ii. SPOJ SUBSEQ - Counting Subsequences
        iii. 877B - Nikita and String
        iv. 835C - Star sky
        v. 846D - Monitor

Topic 2: Linked Lists

1. Basic Operations
    a. Insertion of a node in SLL
        i. In the beginning
        ii. Somewhere in between
        iii. At the end
    b. Deletion of a node in SLL
        i. The head node
        ii. Node somewhere in between
        iii. The last node
    c. Traversing the SLL
        i. Get the length of SLL
        ii. Search an element in SLL
        iii. Left rotate the SLL by ‘k’ nodes
        iv. Reverse the SLL
        v. Reverse the SLL in blocks of size ‘k’
        vi. Check if the given SLL is a palindrome
        vii. Bring all even nodes before odd nodes
        viii. Push all zeros to the end
2. Loops in SLL
        i. Detect Loop
        ii. Count number of nodes in a loop
        iii. Remove loop
3. Sorting SLL
    a. Insertion Sort
    b. Merge Sort
        i. Merge two SLLs alternatively
4. Interesting Problems
        i. Find the intersection node when two SLLs intersect
        ii. Add two numbers represented in the form of SLLs
        iii. Given a DLL with one extra random pointer, clone this list
5. Circularly linked lists
        i. Insertion in CLL
        ii. Deletion in CLL
        iii. Traversing CLL
        iv. Insertion Sort in CLL

6. Doubly linked lists
    i. Insertion in DLL
    ii. Deletion in DLL
    iii. Traversing DLL
    iv. Merge Sort DLL
    v. Memory Efficient DLL

Topic 3: Stacks

1. Implementing Stacks
    a. Fixed size stack
    b. Dynamic stack using Table-Doubling
    c. In-built stack in STL
2. Design Problems in Stacks
    a. Implement two stacks in a single array
    b. Implement a stack that also gives minimum element in O(1) time
    c. Implement a stack that returns & deletes the middle element in O(1) time
3. Applications of Stacks
    a. Check if parentheses are balanced
    b. Interconversion & Evaluation of Infix, Prefix and Postfix notations
4. Stack problems (CP Topic)
    a. SPOJ ANARC09A - Seinfeld
    b. SPOJ STPAR - Street Parade
    c. Stock Span Problem
    d. Maximum area under histogram

Topic 4: Queues

1. Implementing Queues
    a. Fixed size queue
    b. Dynamic stack using Table-Doubling
    c. In-built queue in STL
2. Design Problems in Queues
    a. Implement a stack using queues.
    b. Implement a queue using stacks.

3. Queue Problems
    a. Given an array of non-negative integers, find the largest multiple of 3 that can be formed from array elements.
    b. Given an integer N, find the least possible integer made up of only digits 9 and 0 such that it is divisible by N.

Topic 5: Trees

1. Basics
    a. Structure of a binary tree
    b. Types of BT
        i. Full/Strict Binary Tree
        ii. Complete Binary Tree
        iii. Perfect Binary Tree
        iv. Skewed Binary Tree
    c. Recursive Codes
        i. Sum of all nodes in BT
        ii. Height of BT
        iii. Find the maximum element in BT
        iv. Check if a node exists in BT or not
        v. Check if tree is full/strict
        vi. Check if tree is complete
        vii. Check if tree is perfect
        viii. Check if tree is skewed
        ix. Compare if two trees are identical (and similar)
        x. Check if a tree is foldable (and symmetric)
        xi. Check if two trees are isomorphic
        xii. Print boundary nodes of BT
        xiii. Print left, right, top & bottom views of BT
        xiv. Given a number, create the factor tree
2. Traversals
    a. Breadth First Traversal
        i. Level Order Traversal
    b. Depth First Traversal
        i. Preorder Traversal
        ii. Inorder Traversal
        iii. Postorder Traversal

    c. Level Order Traversal Variations
        i. Insertion of Node
        ii. Deletion of Node
        iii. Find the maximum width of BT
        iv. Print corner nodes at each level
        v. Find the sum of leaf nodes at the minimum level
        vi. Print BT vertically.
        vii. Reverse level order traversal
        viii. Spiral order traversal
3. BT Construction
    a. How many different BTs (and BSTs) are possible with n nodes?
    b. If given two traversals of BT, can we construct a BT uniquely?
4. Least Common Ancestor (LCA)
    a. Find LCA of two nodes in BT
    b. Find distance between two nodes in BT
    c. Find the k'th ancestor of a node
    d. Check if two nodes are cousins
5. Interesting Problems
    a. Traversal based
        i. Given Inorder & Preorder traversals of a binary tree, print postorder traversal.
        ii. Given a preorder traversal of a full BT as a string of characters 'l' (leaf) & 'n' (node), find the depth of the BT.
        iii. Given a BT, find all duplicate subtrees.
    b. Root to leaf path
        i. Print all root to leaf paths
        ii. Check if there is a root to leaf path which adds up to a given sum
        iii. Find the length of the diameter (longest path between two nodes) of BT
        iv. Find maximum possible sum from one leaf to another
        v. Find the length of the path having maximum bends.
    c. LCA based
        i. Given two nodes, count the number of turns needed to reach from one node to another.
    d. N-ary tree based
        i. Given a very large n-ary tree, the root node has some info to pass to all its children. Each node can only pass the information to one child at a time. Find the minimum iterations required to pass info to all nodes of BT

6. Binary Search Trees
    a. Basic operations on BST
        i. Searching a key in BST
        ii. Inserting a key in BST
        iii. Deleting a key from BST
    b. LCA of BST
        i. Find the LCA of two nodes in BST
        ii. Find the distance between two nodes in BST
    c. Interval Trees
        i. Implement a data structure which efficiently performs following operations:
            - Add an interval
            - Remove an interval
            - Given an interval x, find if x overlaps with any of the existing intervals.
    d. BST Problems
        i. Given a BT, check if it is BST or not
        ii. Given a BT, return the size of the largest subtree which is a BST.
        iii. Given a BST, find the k'th smallest element in BST
        iv. Given a BST and a value k, find the node with minimum absolute difference with the value k.
        v. Given two values k1 and k2, print all keys in range k1 and k2.
7. Heaps
    a. Implementation
        i. Array based
        ii. Priority Queues from STL
    b. Applications
        i. K'th smallest element
        ii. Heap Sort
        iii. Sorting an almost sorted array
    c. Heap Problems
        i. Given an array, find k numbers with most occurrences i.e. top k numbers with maximum frequency
        ii. Given k sorted arrays of different sizes, merge them into one sorted array
        iii. Find median of the running streams of integers

Topic 6: Graphs

1. Basics
    a. Graphs = Nodes + Edges
    b. Types of Graphs
        i. Directed vs Undirected
        ii. Unweighted vs Weighted
        iii. Cyclic vs Acyclic
    c. Representation of Graphs
        i. Adjacency Matrix
        ii. Adjacency List
2. Breadth First Search (BFS)
    a. Generic BFS algorithm
        i. Order in which nodes are being traversed in BFS
        ii. Concept of level and parent array
    b. Code
        i. Adjacency List
        ii. Adjacency Matrix
        iii. Grid
    c. Variations
        i. Shortest path from source to other vertices in an unweighted graph using parent array
        ii. Number of shortest paths from source to other vertices
        iii. Finding the least number of moves where states can be represented as nodes and transitions can be represented as edges. Example,
            - SPOJ PPATH - Prime Path
            - SPOJ NAKANJ - Minimum Knight Moves
    d. BFS Problems (CP Topic)
        i. SPOJ WATER - Water among cubes
        ii. 769C - Cycle in maze
        iii. 796D - Police Stations
        iv. 821D - Okabe and City
        v. 59E - Shortest Path
        vi. 653E - Bear and Forgotten Tree 2
3. Depth First Search (DFS)
    a. Generic DFS algorithm
        i. Order in which nodes are being traversed in DFS

4. Connectivity

Topic: Divide and Conquer

1. Binary Exponentiation
    a. Implementation
        i. On Numbers
        ii. On matrices
    b. Applications
        i. Computing (x^n) mod m which will be later used in computing modular multiplicative inverse
        ii. Computing fibonacci numbers effectively
        iii.
2.