Pre-Placements Checklist

# Data Structures:

1. Array
   1. Kadane's Algorithm

<https://www.geeksforgeeks.org/largest-sum-contiguous-subarray/>

* 1. N/2, N/3 greatest Number

<https://leetcode.com/problems/majority-element/>

<https://leetcode.com/problems/majority-element-ii/>

<https://www.geeksforgeeks.org/given-an-array-of-of-size-n-finds-all-the-elements-that-appear-more-than-nk-times/>

* 1. Merge overlapping intervals

<https://leetcode.com/problems/merge-intervals/>

* 1. Rotate matrix

<https://leetcode.com/problems/rotate-image/>

* 1. Buy / Sell stocks - I, II, III: <https://leetcode.com/problems/best-time-to-buy-and-sell-stock/>

1. String
   1. Pattern matching algorithms (KMP + Rabin Karp)

<https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/>

<https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/>

* 1. Using StringBuilder class -> Add, Multiply Strings

<https://www.geeksforgeeks.org/stringbuilder-class-in-java-with-examples/>

<https://www.geeksforgeeks.org/stringbuilder-append-method-in-java-with-examples/>

* 1. String compression algorithm

<https://leetcode.com/problems/string-compression/>

1. LinkedList
   1. Implementation of LinkedList

<https://www.geeksforgeeks.org/implementing-a-linked-list-in-java-using-class/>

<https://leetcode.com/problems/design-linked-list/>

* 1. Detect cycle in a LinkedList - Floyd Algo

<https://leetcode.com/problems/linked-list-cycle/>

* 1. Reverse a linked list + reverse in groups

<https://leetcode.com/problems/reverse-linked-list/>

<https://leetcode.com/problems/reverse-nodes-in-k-group/>

1. Stack
   1. Implementation of Stack

<https://www.geeksforgeeks.org/stack-data-structure-introduction-program/>

<https://www.geeksforgeeks.org/stack-class-in-java/>

* 1. Balance parenthesis

<https://leetcode.com/problems/valid-parentheses/>

* 1. Trapping rain water

<https://leetcode.com/problems/trapping-rain-water/>

* 1. Implement min stack

<https://leetcode.com/problems/min-stack/>

1. Queue
   1. Implementation of Queue + Deque

<https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/>

<https://www.geeksforgeeks.org/queue-interface-java/>

<https://www.geeksforgeeks.org/implementation-deque-using-circular-array/>

<https://www.geeksforgeeks.org/deque-interface-java-example/>

* 1. Sliding window maximum

<https://leetcode.com/problems/sliding-window-maximum/>

* 1. Implement BFS

<https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>

* 1. Implement Level order in Binary tree

<https://leetcode.com/problems/binary-tree-level-order-traversal/>

1. PriorityQueue or Heap
   1. Implementation of Heap Data structure

<https://www.geeksforgeeks.org/heap-data-structure/>

* 1. Connect n ropes with min cost: <https://www.geeksforgeeks.org/connect-n-ropes-minimum-cost/>
  2. Median of running stream: <https://www.geeksforgeeks.org/median-of-stream-of-running-integers-using-stl/>
  3. LRU and LFU cache

<https://leetcode.com/problems/lru-cache/>

<https://leetcode.com/problems/lfu-cache/>

1. Set & Map
   1. Internal working of HashMap

<https://www.geeksforgeeks.org/internal-working-of-hashmap-java/>

* 1. 4-sum

<https://leetcode.com/problems/4sum/>

* 1. Longest substring without repeat: <https://www.interviewbit.com/problems/longest-substring-without-repeat/>

1. Binary Tree
   1. Implementation: insert, delete, traverse: <https://youtu.be/QhIM-G7FAow>
   2. Print top view, left view, right view, bottom view, level order, zig-zag traversal of Binary tree

<https://www.geeksforgeeks.org/print-nodes-top-view-binary-tree/>

<https://www.geeksforgeeks.org/print-left-view-binary-tree/>

<https://leetcode.com/problems/binary-tree-right-side-view/>

<https://www.geeksforgeeks.org/bottom-view-binary-tree/>

<https://www.geeksforgeeks.org/level-order-tree-traversal/>

<https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/>

* 1. Invert a binary tree: <https://leetcode.com/problems/invert-binary-tree/>
  2. Lowest common ancestor

<https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/>

1. Binary Search Tree
   1. Implementation

<https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/>

* 1. Check if a tree is BST or not

<https://www.geeksforgeeks.org/a-program-to-check-if-a-binary-tree-is-bst-or-not/>

* 1. AVL tree and rotation

<https://www.geeksforgeeks.org/avl-tree-set-1-insertion/>

<https://www.geeksforgeeks.org/avl-tree-set-2-deletion/>

1. Graph
   1. Implementation, BFS, and DFS traversals

<https://www.geeksforgeeks.org/graph-and-its-representations/>

<https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>

<https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/>

* 1. Topological sorting

<https://www.geeksforgeeks.org/topological-sorting/>

* 1. Bellman ford Algorithm

<https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/>

* 1. Dijkstra's Algorithm

<https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

* 1. Prim's Algorithm

<https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/>

* 1. Kruskal's Algorithm

<https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/>

* 1. Unique Islands Problem: <https://www.geeksforgeeks.org/find-the-number-of-distinct-islands-in-a-2d-matrix/>

1. Trie
   1. Implementation

<https://www.geeksforgeeks.org/trie-insert-and-search/>

1. Segment Trees : More important in CP
   1. Implementation

<https://www.hackerearth.com/practice/data-structures/advanced-data-structures/segment-trees/tutorial/>

# Algorithms:

1. Two pointers Algorithm
   1. 3-Sum

<https://leetcode.com/problems/3sum/>

* 1. Container with most water

<https://leetcode.com/problems/container-with-most-water/>

* 1. Sort the array containing only 0, 1 and 2

<https://www.geeksforgeeks.org/sort-an-array-of-0s-1s-and-2s/>

1. Math
   1. Fast Power: <https://www.youtube.com/watch?v=dyrRM8dTEus>
   2. Euclid GCD: <https://www.geeksforgeeks.org/euclidean-algorithms-basic-and-extended/>
   3. Sieve of Eratosthenes:

<https://www.geeksforgeeks.org/sieve-of-eratosthenes/>

1. Recursion + Backtracking
   1. Sudoku solver

<https://leetcode.com/problems/sudoku-solver/>

* 1. N-Queens Problem

<https://leetcode.com/problems/n-queens/>

* 1. Permutation and Combinations (Bruteforce)

<https://www.geeksforgeeks.org/permutation-and-combination/>

1. Bits Manipulation + Mathematics
   1. Find one non-repeating number, find two

<https://www.geeksforgeeks.org/non-repeating-element/>

<https://www.geeksforgeeks.org/find-two-non-repeating-elements-in-an-array-of-repeating-elements/>

* 1. Count 1 bits in a number

<https://leetcode.com/problems/number-of-1-bits/>

1. Divide & Conquer
   1. Merge Sort

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

* 1. Median of two sorted arrays

<https://leetcode.com/problems/median-of-two-sorted-arrays/>

1. Binary Searching
   1. Find upper and lower bounds using Binary search

<https://www.geeksforgeeks.org/find-first-and-last-positions-of-an-element-in-a-sorted-array/>

* 1. Allocate books: <https://www.interviewbit.com/problems/allocate-books/>

1. Greedy Programming
   1. Candy distribution: <https://www.interviewbit.com/problems/distribute-candy/>
   2. Gas station: <https://www.interviewbit.com/problems/gas-station/>
   3. Fractional Knapsack

<https://www.geeksforgeeks.org/fractional-knapsack-problem/>

1. Dynamic Programming
   1. 0/1 Knapsack: <https://www.youtube.com/watch?v=y6kpGJBI7t0>
   2. Longest increasing subsequence

<https://leetcode.com/problems/longest-increasing-subsequence/>

* 1. Matrix chain multiplication

<https://www.geeksforgeeks.org/matrix-chain-multiplication-dp-8/>

* 1. Coin change problem

<https://leetcode.com/problems/coin-change/>

# Operating System:

1. Basics of Threads
2. Process scheduling algorithms
3. Critical section Problem
4. Deadlock
5. Memory management
   1. Paging
   2. Segmentation
6. Page replacement algorithms
7. Disk scheduling algorithms

# DBMS:

1. Types of Keys: Candidate, Super, Foreign keys
2. Normal Forms
3. Joins
4. SQL queries
5. ACID properties
6. Indexing: B trees, B+ trees concepts

# 

# System design:

1. Low-level design
   1. Class, ER diagrams
   2. OOPS concepts
   3. Design Elevator system, Parking Lot, MakeMyTrip System
2. High-level design
   1. Scaling
   2. Distributed systems
   3. Microservice and Monolithic architecture
   4. Load balancing
   5. Message queue
   6. Design Whatsapp, Tinder, and Uber system