

Data Structure - LLDs - (1 Week)

List of data structures

- ❑ Lists
 - ❑ [Design Linked List](#)
 - ❑ [Design Skiplist](#)
- ❑ Stacks
 - ❑ [Implement Stack using Queues](#)
 - ❑ [Design a Stack With Increment Operation](#)
 - ❑ [LRU Cache](#)
 - ❑ [Min Stack](#)
 - ❑ [Max Stack](#)
 - ❑ [Dinner Plate Stacks](#)
 - ❑ [Implement Queue using Stacks](#)
- ❑ Queue
 - ❑ [Design Circular Queue](#)
- ❑ Hashtable
 - ❑ [Design HashMap](#)
 - ❑ [Design HashSet](#)
- ❑ BST
 - ❑ [Binary Search Tree Iterator](#)
 - ❑ [Serialize and Deserialize BST](#)
- ❑ Red Black Tree
 - ❑ [Find Median from Data Stream](#)
 - ❑ [Count of Range Sum](#)
- ❑ Heaps
 - ❑ [Design Twitter](#)
 - ❑ [Kth Largest Element in a Stream](#)
- ❑ Fibonacci Heaps
 - ❑ [Fibonacci Heaps](#)
- ❑ Disjoint Sets
 - ❑ [Review of two popular approaches, Disjoint Sets and DFS](#)
- ❑ Tries (PrefixTree, suffixTree)
 - ❑ [Implement Trie \(Prefix Tree\)](#)
 - ❑ [Add and Search Word - Data structure design](#)
- ❑ Interval Trees/Segment Tree
 - ❑ [Lazy Dynamic Segment Tree - A general template](#)
 - ❑ [A Recursive approach to Segment Trees, Range Sum Queries & Lazy Propagation](#)
- ❑ Other Tree Data Structures(Graphs)
 - ❑ [Serialize and Deserialize N-ary Tree](#)
 - ❑ [Encode N-ary Tree to Binary Tree](#)

Algorithms - Analysis Time and Space - (3 Weeks)

- ❑ **Sorting**
 - ❑ Selection Sort - [Merge Sorted Array](#)
 - ❑ Bubble Sort - [Sort Colors](#)
 - ❑ Insertion Sort - [Insertion Sort List](#)
 - ❑ Merge Sort - [Sort an Array](#)

- 2 Days

- ❑ Quick Sort
 - ❑ [Kth Largest Element in an Array](#)
 - ❑ [K Closest Points to Origin](#)
- ❑ Counting Sort - [Relative Sort Array](#)
- ❑ Tree sort - [Convert Sorted List to Binary Search Tree](#)
- ❑ Bucket Sort - [Top K Frequent Elements](#)
- ❑ Radix Sort - [Maximum Gap](#)
- ❑ Topological sort - Covered in Graphs

- ❑ **Divide-and-Conquer** - 2 Days
 - ❑ The maximum-subarray problem - [Maximum Subarray](#)
 - ❑ Strassen's algorithm for matrix multiplication - [Divide and Conquer | Set 5 \(Strassen's Matrix Multiplication\)](#)
 - ❑ The substitution method for solving recurrences
 - ❑ The recursion-tree method for solving recurrences
 - ❑ The master method for solving recurrences

- ❑ **Dynamic Programming** - 2 Days
 - ❑ Rod cutting - [Integer Break](#)
 - ❑ [Dynamic Programming for the confused : Rod cutting problem](#)
 - ❑ Matrix-chain multiplication - [Burst Balloons](#)
 - ❑ Elements of dynamic programming
 - ❑ Longest common subsequence - [Longest Common Subsequence](#)
 - ❑ Optimal binary search trees
 - ❑ [Unique Binary Search Trees](#)
 - ❑ [Unique Binary Search Trees II](#)

- ❑ **Greedy Algorithms** - 2 Days
 - ❑ An activity-selection problem - [Minimum Number of Arrows to Burst Balloons](#)
 - ❑ Elements of the greedy strategy
 - ❑ Huffman codes - [Construct Huffman Tree, Google | Onsite | Software Engineer | Huffman Coding Algorithm, Minimum Cost Tree From Leaf Values](#)
 - ❑ Matroids and greedy methods - [Matroid intersection in simple words](#)
 - ❑ A task-scheduling problem as a matroid - [Task Scheduler](#)

- ❑ **Graph Algorithms** - 6 Days

[Leetcode Pattern 1 | DFS + BFS == 25% of the problems](#)

 - ❑ [N-ary Tree Preorder Traversal](#)
 - ❑ [N-ary Tree Postorder Traversal](#)
 - ❑ [N-ary Tree Level Order Traversal](#)
 - ❑ BFS
 - ❑ [Binary Tree Level Order Traversal](#)
 - ❑ [Binary Tree Level Order Traversal II](#)
 - ❑ [Web Crawler Multithreaded](#)
 - ❑ [Web Crawler](#)
 - ❑ [Cut Off Trees for Golf Event](#)
 - ❑ [Course Schedule](#)
 - ❑ DFS

- ❑ [Binary Tree Postorder Traversal](#)
 - ❑ [Binary Tree Preorder Traversal](#)
 - ❑ [Binary Tree Inorder Traversal](#)
 - ❑ [Is Graph Bipartite?](#)
 - ❑ [Remove Invalid Parentheses](#)
 - ❑ [Construct Binary Tree from Preorder and Inorder Traversal](#)
- ❑ Topological Sort - [Topological Sort](#)
- ❑ Strongly Connected Components - SCC - [Course Schedule](#), [Facebook | Minimum number of people to spread a message](#), [Airbnb | Cover all vertices with the least number of vertices](#), [Critical Connections in a Network](#)
- ❑ Minimum spanning Tree - Prim's Algorithm
 - ❑ [Cheapest Flights Within K Stops](#)
 - ❑ [Minimum Height Trees](#)
 - ❑ [Number of Operations to Make Network Connected](#)
 - ❑ [Connecting Cities With Minimum Cost](#)
- ❑ Shortest Path Algos -
 - ❑ Bellman-Ford - [Network Delay Time](#), <https://leetcode.com/problems/get-watched-videos-by-your-friends/>
 - ❑ Dijkstra's algorithm
 - ❑ [Reachable Nodes In Subdivided Graph](#)
 - ❑ [Shortest Path Visiting All Nodes](#)
 - ❑ Floyd-Warshall
 - ❑ [Find the City With the Smallest Number of Neighbors at a Threshold Distance](#)
 - ❑ [Evaluate Division](#)
 - ❑ Johnson's algorithm
 - ❑ [All-pairs shortest paths - Johnson's algorithm for sparse graphs - GeeksforGeeks](#)
 - ❑ [Johnson's algorithm](#)
 - ❑ The Ford-Fulkerson method
 - ❑ [Google | Onsite | Network flow for the matrix with given row and column sums](#)
 - ❑ [Ford-Fulkerson Algorithm for Maximum Flow Problem](#)
- ❑ **Number-Theoretic Algorithms** - 2 Days
 - ❑ The Chinese remainder theorem - [Check If It Is a Good Array](#)
 - ❑ Greatest common divisor
 - ❑ [Greatest Common Divisor of Strings](#)
 - ❑ [X of a Kind in a Deck of Cards](#)
 - ❑ [Google | OA Summer Intern 2020 | Greatest Common Divisor](#)
 - ❑ Powers of an element
 - ❑ [Pow\(x, n\)](#)
 - ❑ [Sort Integers by The Power Value](#)
 - ❑ The RSA public-key cryptosystem
 - ❑ [Keys and Rooms](#)
 - ❑ [Shortest Path to Get All Keys](#)
 - ❑ Integer factorization
 - ❑ [Largest Component Size by Common Factor](#)
 - ❑ [Minimum Factorization](#)

- ❑ [2 Keys Keyboard](#)
 - ❑ [Bulb Switcher](#)
- ❑ **String Matching** - 2 Day
 - ❑ The Rabin-Karp algorithm
 - ❑ [Implement strStr\(\)](#)
 - ❑ [Binary String With Substrings Representing 1 To N](#)
 - ❑ [Shortest Palindrome](#)
 - ❑ [Find All Anagrams in a String](#)
 - ❑ String matching with finite automata
 - ❑ The Knuth-Morris-Pratt algorithm
 - ❑ [Shortest Palindrome](#)
 - ❑ [Rotate String](#)
 - ❑ [KMP Algorithm for Pattern Searching](#)
- ❑ **Approximation Algorithms** - 3 Days
 - ❑ The vertex-cover problem
 - ❑ [Binary Tree Cameras](#)
 - ❑ [Vertex Cover Problem-2](#)
 - ❑ [Vertex Cover Problem](#)
 - ❑ The traveling-salesman problem [Find the Shortest Superstring](#)
 - ❑ The set-covering problem
 - ❑ [Video Stitching](#)
 - ❑ [Set Intersection Size At Least Two](#)
 - ❑ [Non-overlapping Intervals](#)
 - ❑ Randomization and linear programming
 - ❑ The subset-sum problem
 - ❑ [Partition Equal Subset Sum](#)
 - ❑ [Partition to K Equal Sum Subsets](#)
- ❑ **Randomized Algorithms** - 1 Day
 - ❑ Quick Sort
 - ❑ Min Cut [Palindrome Partitioning II](#)

Concepts Problems and Maths - (1 Week)

- ❑ Matrix Operations
- ❑ Linear Programming
- ❑ Polynomials - DFT, FFT
- ❑ Computational Geometry
 - ❑ Line-segment properties
 - ❑ Determining whether any pair of segments intersects
 - ❑ Finding the convex hull - [Erect the Fence](#), [The Skyline Problem](#)
 - ❑ Finding the closest pair of points - [K Closest Points to Origin](#)
- ❑ GCD and LCM
 - ❑ [X of a Kind in a Deck of Cards](#)
 - ❑ [Greatest Common Divisor of Strings](#)
 - ❑ [Nth Magical Number](#)
 - ❑ [Ugly Number III](#)

- ❑ Prime Factorization and Divisors
 - ❑ [Largest Component Size by Common Factor](#)
 - ❑ [2 Keys Keyboard](#)
- ❑ Fibonacci Numbers
 - ❑ [Length of Longest Fibonacci Subsequence](#)
 - ❑ [Split Array into Fibonacci Sequence](#)
 - ❑ [Find the Minimum Number of Fibonacci Numbers Whose Sum Is K](#)
- ❑ Catalan Numbers - [Unique Binary Search Trees](#)
- ❑ Modular Arithmetic
- ❑ Euler Totient Function
- ❑ nCr Computations
- ❑ Set Theory
- ❑ Factorial
 - ❑ [Last Substring in Lexicographical Order](#)
 - ❑ [Snakes and Ladders](#)
 - ❑ [Factor Combinations](#)
 - ❑ [Path With Maximum Minimum Value](#)
 - ❑ [Number of Closed Islands](#)
- ❑ Prime numbers and Primality Tests
 - ❑ [Prime Arrangements](#)
 - ❑ [K-th Smallest Prime Fraction](#)
- ❑ Sieve Algorithms
 - ❑ [Count Primes](#)
- ❑ Divisibility and Large Numbers
- ❑ Series
- ❑ Number Digit
- ❑ Triangles
 - ❑ [Triangle](#)
 - ❑ [Valid Triangle Number](#)

Networks - (1 Week)

[Leetcode](#)

- ❑ Network Topology, OSI Architecture
- ❑ TCP/IP models
- ❑ TCP and UDP
- ❑ Firewall, DNS, Domains, workgroups
- ❑ Protocols i.e ICMP

OS - (1 week)

[Operating System Tutorial](#)
[Shared Memory Systems](#)

- ❑ Cache
- ❑ Multithreading
 - ❑ Producers-consumers problem
 - ❑ Dining philosophers problem
 - ❑ Cigarette smokers problem
 - ❑ Readers-writers problem

- ☐ [Web Crawler Multithreaded](#)
- ☐ Scheduling algorithms
- ☐ Deadlock
- ☐ Virtual Memory
- ☐ Mutex and semaphore
- ☐ Kernels
- ☐ Paging

Software Design Principles - (2 weeks)

[System Design Primer](#)

[Start learning about Theory of Distributed Systems?](#)

[Challenges with distributed systems](#)

[Microservices Design Guide 🧑🏫🏠 - Platform Engineer](#)

[Cloud design patterns - Azure Architecture Center](#)

[Design patterns for microservices | Azure Blog and Updates](#)

TO READ:

Domain Driven Design (DDD) | Bounded Context (BC) | Polyglot Persistence (PP) |
 Command and Query Responsibility Segregation (CQRS) | Command Query Separation
 (CQS) | Event-Sourcing (ES) | CAP Theorem | Eventual Consistency | Twelve-Factor App |
 SOLID Principles |

Just some things to focus on.

- ☐ Load balancer
- ☐ API gateway
- ☐ Microservices - Scale Cube Concept, MVC - READ
- ☐ Database Sharding
- ☐ SQL vs NoSQL - Cassandra, Postgres, Hadoop, Data lake, other algorithms related to data lake, CAP Theorem

Leadership Principles - LPs - (1 Week)

TO BE UPDATED

Resume and Miscellaneous

#ADD WHATEVER YOU HAVE PUT IN RESUME

- ☐ Algos you have mentioned
- ☐ Project work and related references to read
- ☐ Achievements and information about it

REFERENCES

Introduction to Algorithms - Cormen

Leetcode
