

DSA - Class Assignments & Expected Learning

The course focuses on hands-on learning through 65 practical assignments and 100+ problems solved during live sessions, rather than project-based learning.

Class 1: Programming Fundamentals I

Class Assignment

Easy

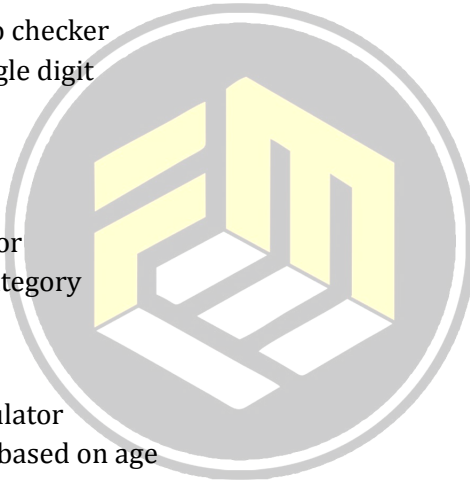
- Check divisibility by 5
- Positive/Negative/Zero checker
- Check if number is single digit

Medium

- Triangle Validity Check
- Electricity Bill Calculator
- BMI Calculator with category

Hard

- Traffic light signal simulator
- Ticket price calculator based on age



Expected Learning Outcome

- Declare and use variables of different data types confidently
- Translate a real world condition into nested if else logic
- Predict the output of conditional code by hand

Class 2: Programming Fundamentals II

Class Assignment

Easy

- Print all even numbers till N
- Sum of even/odd numbers separately
- Fibonacci series (iterative)

Medium

- Count Digits and Sum of Digits
- Armstrong Number
- GCD using loop

Hard

- Print all prime numbers till N
- Power without built-in function

Expected Learning Outcome

- Choose between for/while/do while appropriately
- Extract and manipulate digits of a number using loops
- Trace loop execution manually to debug logic errors

Class 3: Pattern Printing

Class Assignment

Easy

- Number triangle
- Inverted triangle
- Square pattern

Medium

- Diamond with numbers
- Alphabet triangle

Hard

- Hollow diamond



Expected Learning Outcome

- Break down any visual pattern into row and column logic
- Independently derive the loop bounds for a new pattern
- Apply symmetry based logic (diamond, hollow shapes) without hesitation

Class 4: Functions & Number Theory

Class Assignment

Easy

- isEven() function
- sumOfDigits() function
- factorial() function

Medium

- Function to find HCF of 3 numbers
- Function to check Perfect Number
- Sum of Divisors of a Number

Hard

- Function to generate first N Fibonacci numbers
- Sieve of Eratosthenes (basic version)

Expected Learning Outcome

- Write clean, reusable functions with correct parameter/return design
- Explain why modular code is preferred over monolithic code
- Solve classic number theory interview questions (GCD, LCM, primes)

Class 5: 2D Arrays / Matrix Basics

Class Assignment

Easy

- Print matrix elements
- Sum of all matrix elements
- Find max element in matrix

Medium

- Row with maximum sum
- Check if matrix is symmetric
- Spiral traversal of a matrix

Hard

- Search in a row-wise and column-wise sorted matrix
- Boundary traversal of a matrix

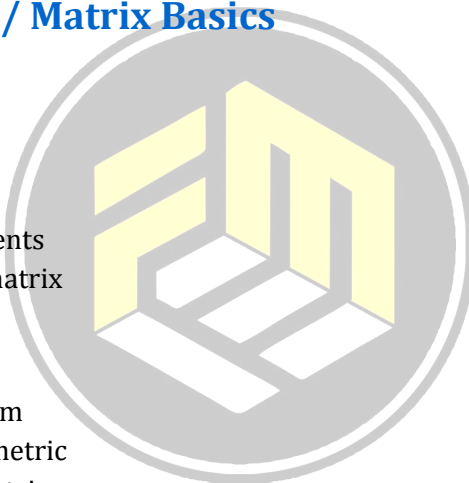
Expected Learning Outcome

- Traverse a 2D array confidently in any direction
- Solve row/column/diagonal aggregation problems
- Rotate and search a matrix using a straightforward approach

Class 6: Time & Space Complexity

Class Assignment

Easy



- Identify Big-O for 5 given code snippets
- Identify space complexity for 3 given code snippets

Medium

- Compare two given solutions to the same simple problem and justify which is faster

Hard

- Write one solution in two ways (brute force and optimized) and explain the complexity difference

Expected Learning Outcome

- Look at a piece of code and state its time complexity confidently
- Explain the difference between time and space complexity in simple terms
- Use complexity as the deciding factor when comparing two solutions, going forward in every module

Class 7: Arrays I — Foundations

Class Assignment

Easy

- Sum of array elements
- Reverse an array (in-place)
- Check if array is sorted

Medium

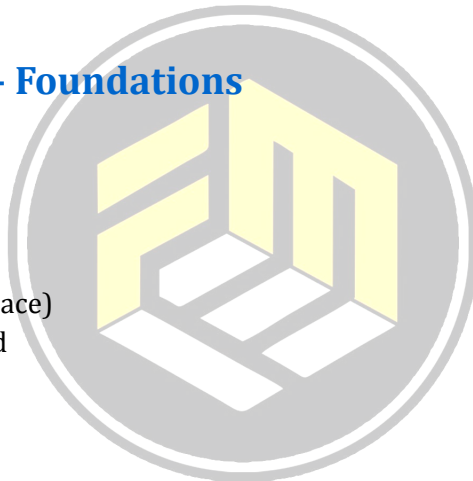
- Right rotate array by K
- Count negative numbers in an array
- Find the missing number in an array

Hard

- Best Time to Buy and Sell Stock (single transaction)
- Find the duplicate number in an array

Expected Learning Outcome

- Manipulate arrays in place efficiently
- Apply Kadane's Algorithm to maximum subarray problems
- Be ready to learn two pointer and sliding window techniques



Class 8: Arrays II — Two Pointer

Class Assignment

Easy

- Check if array has a pair with given sum
- Move all negatives to one side
- Intersection of two sorted arrays

Medium

- Three Sum Closest
- Merge Two Sorted Arrays (in-place basic)
- Merge intervals (introductory)

Hard

- Trapping Rain Water using two pointer
- Minimum size subarray with given sum (bridge to sliding window)

Expected Learning Outcome

- Recognize sorted array problems solvable with two pointers
- Convert $O(n^2)$ brute force solutions into $O(n)$ two pointer solutions
- Solve pair sum and triplet sum family problems

Class 9: Sliding Window — Fixed Size

Class Assignment

Easy

- Sum of first K elements sliding to end
- Maximum sum of K consecutive elements

Medium

- Find all negative numbers in windows of size K
- Count distinct elements in every window of size K
- Longest substring of length K with no repeating characters

Hard

- Count of subarrays of size K with sum divisible by K
- K-length substrings with exactly M vowels

Expected Learning Outcome

- Identify problems that require a fixed size sliding window
- Convert brute force $O(n*k)$ into $O(n)$ using the slide technique
- Prepare for the variable window extension

Class 10: Sliding Window — Variable Size

Class Assignment

Easy

- Fruits into Baskets (variable window)
- Longest substring with at most 2 distinct characters

Medium

- Longest subarray with sum K (non-negative array)
- Count subarrays with exactly K distinct elements

Hard

- Minimum Window Substring (introductory)
- Subarray Product Less Than K

Expected Learning Outcome

- Design expand/shrink conditions for variable size windows
- Solve substring family interview problems confidently
- Be ready for Prefix Sum as a separate array preprocessing technique

Class 11: Prefix Sum

Class Assignment

Easy

- Range sum queries using prefix sum
- Check if a given index is an equilibrium index

Medium

- Find the length of the longest subarray with sum 0
- Difference array for multiple range-update queries

Hard

- Product of Array Except Self (prefix/suffix product variant)

Expected Learning Outcome

- Build and use a prefix sum array to answer range queries in $O(1)$
- Distinguish prefix sum friendly problems from sliding window friendly ones
- Be ready to combine prefix sum with hashing in later classes



Class 12: Practice Class — Arrays Module

Class Assignment

Easy

- Rotate array by K
- Find pivot index

Medium

- Majority Element (Moore's Voting Algorithm)
- Container With Most Water (revisit)
- Minimum Size Subarray Sum

Hard

- Longest Consecutive Sequence
- Sliding Window Maximum (brute-force walkthrough)

Expected Learning Outcome

- Independently choose the correct array technique for a new problem
- Solve a full mixed problem set within a time limit
- Move confidently into the Strings module

Class 13: Strings I — Basics

Class Assignment

Easy

- Count vowels and consonants
- Toggle case of each character
- Remove all duplicate characters from a string

Medium

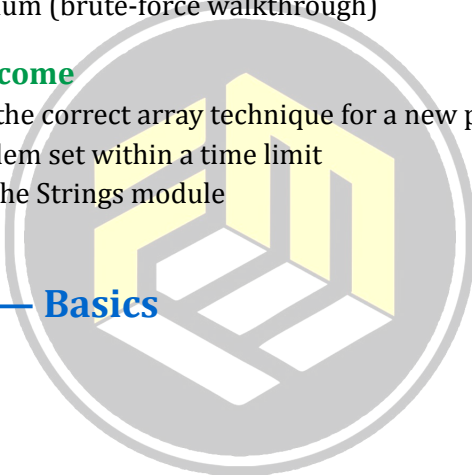
- Check if a string is a rotation of another string
- Reverse words in a sentence
- Longest common prefix among an array of strings

Hard

- Valid palindrome ignoring non-alphanumeric characters
- Roman numeral to integer conversion

Expected Learning Outcome

- Manipulate strings using arrays and two pointers



- Solve palindrome and anagram problems confidently
- Be ready for substring/window based string problems

Class 14: Strings II — Pattern Techniques

Class Assignment

Easy

- Check if a string is a subsequence of another
- Count occurrences of a substring

Medium

- Find all anagrams of a pattern in a string
- String compression (in-place)
- Count and Say Problem

Hard

- Smallest window containing all characters of another string
- Group Shifted Strings

Expected Learning Outcome

- Apply sliding window and two pointer techniques fluently on strings
- Solve substring search and palindrome problems
- Distinguish between substring and subsequence problems

Class 15: Hashing I — HashMap/HashSet Basics

Class Assignment

Easy

- Find the most frequent element in an array
- Union of two arrays using HashSet

Medium

- Find all pairs with a given difference using HashMap
- Check if a subarray with sum 0 exists using HashMap
- Check if Array Contains Duplicate within K Distance

Hard

- Group Anagrams using HashMap
- Isomorphic Strings

Expected Learning Outcome

- Explain why hashing gives average $O(1)$ lookup
- Replace brute force nested loops with HashMap based $O(n)$ solutions
- Solve classic two sum family and frequency based problems

Class 16: Hashing II — Applications

Class Assignment

Easy

- Check if two strings are k-anagrams
- Frequency sort of an array

Medium

- Find All Anagrams in a String (HashMap-optimized revisit)
- Count number of nice subarrays (odd numbers)

Hard

- Maximum size subarray sum equals K (with negatives)
- Subarrays with K different integers

Expected Learning Outcome

- Combine prefix sum and hashing to solve subarray sum problems in $O(n)$
- Confidently solve the "Subarray Sum Equals K" interview classic and its variants
- Transition into Binary Search with strong foundational tools already in hand

Class 17: Binary Search I — Foundations

Class Assignment

Easy

- Binary search on a sorted array (basic)
- Square root of a number using binary search

Medium

- Count of elements less than or equal to a given value
- Find the missing number in an arithmetic progression

Hard

- Find first and last position of element (complete edge-case handling)
- Single element in a sorted array

Expected Learning Outcome

- Implement binary search correctly, handling edge cases (empty array, single element)
- Solve first/last occurrence and floor/ceil problems
- Reduce $O(n)$ search problems to $O(\log n)$

Class 18: Binary Search II — Advanced

Class Assignment

Easy

- Search in a rotated array with duplicates (boolean version)
- Square root with precision using binary search
- Count rotations in a sorted rotated array

Medium

- Capacity to ship packages within D days
- Search a 2D matrix II (staircase search)

Hard

- Painter's Partition Problem
- Book allocation with edge cases

Expected Learning Outcome

- Apply binary search to rotated and unusual sorted structures
- Recognize and construct "binary search on answer" solutions
- Confidently handle binary search variants asked in product based interviews

Class 19: Practice Class — Hashing + Binary Search

Class Assignment

Easy

- Missing number using both hashing and binary search
- Search insert position (timed)

Medium

- 4Sum (mixed technique)
- Count pairs with given sum (timed)
- Two Sum II — Input Array is Sorted

Hard

- Find K closest elements (timed)
- Longest subarray with sum divisible by K (timed)

Expected Learning Outcome

- Independently decide between hashing and binary search for a new problem
- Solve a mixed problem set under time constraints
- Move confidently into Recursion and Backtracking

Class 20: Recursion I — Foundations

Class Assignment

Easy

- Print N to 1 using recursion
- Reverse a string using recursion
- Check if array is sorted using recursion

Medium

- Recursively find the maximum element in an array
- Tower of Hanoi

Hard

- Print all subsequences of a string (introductory)
- Recursive binary search implementation

Expected Learning Outcome

- Identify base cases and recursive cases correctly
- Trace a recursion tree manually
- Be ready to extend recursion into backtracking

Class 21: Recursion II — Backtracking Intro

Class Assignment

Easy

- Print all subsets of size K
- Generate all possible passwords of length N from given characters

Medium

- Permutations of a string with duplicate handling
- Generate Parentheses
- Print all subsequences of a string with sum K

Hard

- Combination Sum II (handling duplicates)
- Palindrome partitioning (introductory)

Expected Learning Outcome

- Apply the choose explore un choose backtracking pattern
- Generate subsets, permutations, and combinations correctly
- Be ready for classic constraint satisfaction backtracking problems

Class 22: Backtracking — Advanced

Class Assignment

Easy

- Count total ways in Rat in a Maze
- Print all paths in a grid using backtracking

Medium

- N-Queens — print one valid solution
- Subset sum existence using backtracking with pruning
- Generate Parentheses (revisit with full pruning)

Hard

- Path with maximum gold (grid backtracking)
- Word Break using backtracking (brute force)

Expected Learning Outcome

- Solve classic backtracking interview problems (N Queens, Rat in a Maze)
- Design basic pruning conditions to reduce unnecessary recursive calls
- Transition confidently into recursion based Sorting algorithms (Merge Sort)

Class 23: Sorting I — Basic Sorts

Class Assignment

Easy

- Implement bubble sort and count number of swaps
- Check if array is sorted using a single pass

Medium

- Sort array of 0s

- 1s
- 2s without extra space
- Find minimum number of swaps to sort an array

Hard

- Sort a nearly sorted array with elements at most K away
- Recognize and fix a sorting bug in given code

Expected Learning Outcome

- Implement and differentiate between bubble, selection, and insertion sort
- Explain stability and why it matters
- Be ready for divide and conquer sorting algorithms

Class 24: Sorting II — Merge Sort

Class Assignment

Easy

- Merge two sorted arrays given extra space
- Find union of two sorted arrays using merge logic

Medium

- Count Inversions (complete)
- Sort Colors using merge-sort-style divide and conquer

Hard

- Count Reverse Pairs (introductory)

Expected Learning Outcome

- Implement Merge Sort from scratch, including the merge step
- Apply Merge Sort to count inversions
- Explain why Merge Sort guarantees $O(n \log n)$ in all cases

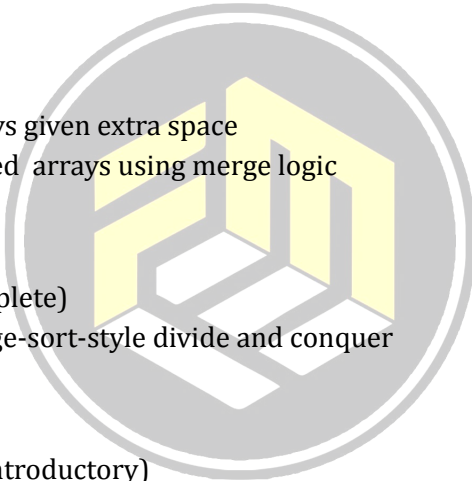
Class 25: Sorting III — Quick Sort & Others

Class Assignment

Easy

- Sort an array using counting sort given a known range
- Implement partition function standalone

Medium



- Kth largest element (complete Quickselect)
- Sort array by parity

Hard

- Kth smallest element in an unsorted array (Quickselect complete with worst-case discussion)

Expected Learning Outcome

- Implement Quick Sort and explain pivot selection
- Use Quickselect to solve Kth largest/smallest problems in average $O(n)$
- Choose the right sorting algorithm based on problem constraints

Class 26: Practice Class — Recursion + Sorting

Class Assignment

Easy

- Print all permutations of a small array (timed)
- Merge two sorted arrays without extra space (timed)

Medium

- Meeting Rooms II (sort + greedy preview)
- Sort a stack using recursion
- N-Queens (timed revisit)
- count solutions only)

Hard

- Largest Number formed from Array (custom comparator sort)
- Combination Sum (timed revisit)

Expected Learning Outcome

- Fluently combine recursion, backtracking, and sorting techniques
- Solve a mixed problem set under interview time pressure
- Move confidently into the Linked List module

Class 27: Linked List I — Singly Linked List

Class Assignment

Easy

- Insert a node at a given position
- Delete the head node

- Search an element in a linked list

Medium

- Find the middle of a linked list (brute force two-pass)
- Remove duplicates from a sorted linked list

Hard

- Rearrange a linked list (odd positions then even positions)
- Add two numbers represented as linked lists

Expected Learning Outcome

- Implement singly linked list operations from scratch
- Handle edge cases like empty list, single node, head/tail deletion
- Be ready for doubly linked lists and reversal techniques

Class 28: Linked List II — Doubly Linked List & Reversal

Class Assignment

Easy

- Traverse a doubly linked list in reverse
- Count nodes in a doubly linked list

Medium

- Reverse a linked list between positions M and N
- Rotate a linked list by K places

Hard

- Reverse Nodes in K-Group
- Swap nodes in pairs

Expected Learning Outcome

- Implement and manipulate doubly linked lists
- Reverse a linked list fully and in groups
- Be fully prepared for fast and slow pointer techniques

Class 29: Linked List III — Fast & Slow Pointer

Class Assignment

Easy



- Check if a linked list has a cycle (boolean only)
- Find the middle node and return its value

Medium

- Remove Nth Node from End (single pass complete)
- Find the intersection point of two linked lists

Hard

- Palindrome Linked List (complete $O(1)$ space)
- Reorder List using middle + reverse + merge

Expected Learning Outcome

- Implement Floyd's cycle detection algorithm confidently
- Solve middle node and Nth from end problems in a single pass
- Be ready for advanced linked list merge and manipulation problems

Class 30: Linked List IV — Merge & Advanced Problems

Class Assignment

Easy

- Merge two sorted linked lists (iterative complete)
- Check if a linked list is sorted

Medium

- Partition List around a value X
- Swap Nodes in Pairs (complete)
- Sort a Linked List (Merge Sort applied to Linked List)

Hard

- Copy List with Random Pointer (conceptual approach discussion)
- Flatten a Multilevel Doubly Linked List (conceptual)

Expected Learning Outcome

- Merge and sort linked lists using divide and conquer thinking
- Solve advanced linked list interview problems involving multiple pointers
- Be fully prepared for the Linked List

Class 31: Practice Class — Linked List Module

Class Assignment

Easy

- Delete all occurrences of a given value
- Middle of Linked List (timed single pass)

Medium

- Rotate a Linked List (timed revisit)
- Remove Nth Node From End (timed revisit)

Hard

- Reverse Nodes in K-Group (timed revisit)
- Add Two Numbers (timed revisit)

Expected Learning Outcome

- Solve any standard linked list interview problem within a time limit
- Debug pointer-related bugs quickly and independently
- Transition confidently into Stack, Queue, and Deque

Class 32: Stack I — Foundations

Class Assignment

Easy

- Implement stack using linked list
- Check if a string of brackets is valid (single bracket type)

Medium

- Design a stack that supports getMin() in O(1)
- Reverse a stack using recursion
- Implement stack using two queues

Hard

- Delete middle element of a stack
- Remove K Digits using Stack

Expected Learning Outcome

- Implement a stack from scratch using both array and linked list
- Solve balanced-parentheses and expression-conversion problems

- Be ready for monotonic stack applications

Class 33: Stack II — Monotonic Stack

Class Assignment

Easy

- Next greater element in a circular array
- Previous smaller element for every element

Medium

- Daily Temperatures (complete)
- Asteroid Collision

Hard

- Largest Rectangle in Histogram
- Trapping Rain Water using monotonic stack

Expected Learning Outcome

- Recognize when a problem requires a monotonic stack
- Solve next/previous greater/smaller element problems in $O(n)$
- Be ready for Queue and Deque, including deque-based sliding window maximum

Class 34: Queue — Foundations

Class Assignment

Easy

- Implement a queue using linked list
- Reverse the first K elements of a queue

Medium

- Design Circular Deque (introductory)
- Number of Recent Calls (design problem)

Hard

- Rotten Oranges (introductory — bridges to BFS)
- Generate binary numbers from 1 to N using a queue

Expected Learning Outcome

- Implement queue and circular queue from scratch

- Convert between stack-based and queue-based designs
- Be ready for Deque and the Sliding Window Maximum problem

Class 35: Deque + Sliding Window Maximum

Class Assignment

Easy

- Reverse a queue using a deque
- Check palindrome using a deque

Medium

- Design a Max/Min Queue
- Interleave the first half of a queue with the second half

Hard

- Sliding Window Maximum (timed complete)
- Shortest Subarray with Sum at Least K (deque-based advanced)

Expected Learning Outcome

- Implement a deque from scratch and apply it correctly
- Solve Sliding Window Maximum optimally using a deque
- Be ready for BFS traversal in Trees and Graphs

Class 36: Practice Class — Stack + Queue Module

Class Assignment

Easy

- Evaluate Reverse Polish Notation
- Baseball Game (stack simulation)

Medium

- Decode String using Stack
- Simplify Path (Unix-style path stack problem)
- Design Circular Queue (timed revisit)

Hard

- Trapping Rain Water (timed)

- both stack and two-pointer approaches)
- Asteroid Collision (timed revisit)

Expected Learning Outcome

- Fluently solve stack, monotonic stack, queue, and deque problems under time pressure
- Explain the LIFO-to-DFS and FIFO-to-BFS conceptual bridges
- Move confidently into the Trees module

Class 37: Binary Trees I — Traversals

Class Assignment

Easy

- Find the maximum element in a binary tree
- Sum of all nodes in a binary tree
- Mirror of a binary tree

Medium

- Zigzag (spiral) Level Order Traversal
- Print all root-to-leaf paths

Hard

- Boundary Traversal of a Binary Tree
- Construct Binary Tree from Preorder and Inorder Traversal

Expected Learning Outcome

- Implement all four standard traversals (preorder, inorder, postorder, level order)
- Convert between recursive and iterative traversal implementations
- Be ready to compute structural tree properties

Class 38: Binary Trees II — Properties

Class Assignment

Easy

- Count nodes at a given level
- Find the minimum depth of a binary tree
- Check if Two Trees are Identical

Medium

- Path Sum II (all root-to-leaf paths with given sum)
- Diameter of a Binary Tree (optimized single-pass)

Hard

- Maximum Path Sum in a Binary Tree (any node to any node)
- Convert a Binary Tree to its Sum Tree

Expected Learning Outcome

- Compute height, diameter, and balance of a tree efficiently
- Solve path-sum family problems using recursion
- Be ready for view-based and ancestor-based tree problems

Class 39: Binary Trees III — Views

Class Assignment

Easy

- Find the leftmost value in the last row of a tree
- Print nodes at K distance from root

Medium

- Diagonal Traversal of a Binary Tree

Hard

- Maximum Width of a Binary Tree

Expected Learning Outcome

- Solve all standard "view" problems (top, bottom, left, right, vertical)
- Recognize how BFS-with-position-tracking solves most view problems
- Be ready for ancestor and path-based tree problems

Class 40: Binary Trees IV — Ancestors & Paths

Class Assignment

Easy

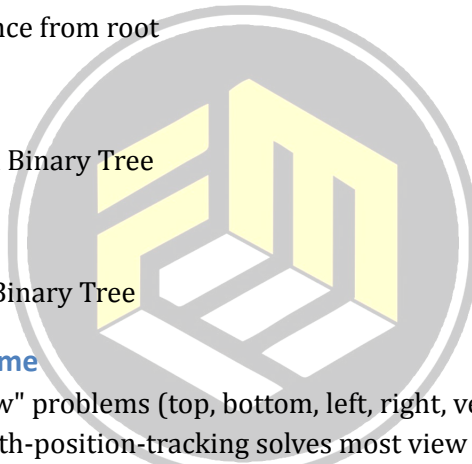
- Check if a given node exists in a binary tree
- Print nodes at K distance from root (revisit)

Medium

- All Nodes Distance K in Binary Tree

Hard

- Minimum time to burn a binary tree from a given node (conceptual)



Expected Learning Outcome

- Find LCA and node-to-node distances confidently
- Combine BFS/DFS with auxiliary maps to solve advanced tree problems
- Be fully prepared to specialize into Binary Search Trees

Class 41: Binary Search Tree I — Foundations

Class Assignment

Easy

- Find the height of a BST
- Count nodes in a BST

Medium

- Find the closest value to a target in a BST
- Construct BST from Preorder Traversal
- Convert Sorted Array to Balanced BST

Hard

- Two Sum in a BST (check pair with given sum)
- Convert a BST to a sorted doubly linked list

Expected Learning Outcome

- Implement search, insert, and delete operations in a BST
- Validate whether a given tree satisfies the BST property
- Be ready for advanced BST applications

Class 42: Binary Search Tree II — Applications

Class Assignment

Easy

- Check if a value exists in a BST (iterative)
- Find the second largest element in a BST

Medium

- Find all elements in a BST within a given range
- Predecessor and Successor of a given key

Hard

- Balance a Binary Search Tree

- Two Sum IV using BST

Expected Learning Outcome

- Solve Kth-order-statistic problems in a BST efficiently
- Apply the BST property to optimize LCA and range-query problems
- Be fully prepared for Heap/Priority Queue

Class 43: Practice Class — Trees + BST Module

Class Assignment

Easy

- Sum Root to Leaf Numbers
- Same Tree (timed revisit)

Medium

- Binary Tree Zigzag Level Order Traversal (timed revisit)
- Convert Sorted Array to BST (timed revisit)
- Right View of a Binary Tree (timed revisit)

Hard

- Delete Node in a BST (timed revisit)
- Maximum Path Sum (timed complete)

Expected Learning Outcome

- Solve any standard Tree or BST interview problem within a time limit
- Distinguish when BST-specific optimizations apply versus general tree recursion
- Move into Heap and Priority Queue

Class 44: Heap / Priority Queue

Class Assignment

Easy

- Check if an array represents a valid max-heap
- Find the K closest points to origin (heap approach)
- Last Stone Weight

Medium

- Kth Largest Element in a Stream (design problem)
- Sort a Nearly Sorted (K-sorted) Array using Heap

Hard

- Find Median from Data Stream (complete two-heap)
- Task Scheduler (heap-based complete)

Expected Learning Outcome

- Implement heap operations (insert, extract-min/max, heapify)
- Solve top-K and Kth-order-statistic problems using heaps
- Be ready to combine heap concepts with Greedy Algorithms

Class 45: Greedy Algorithms

Class Assignment

Easy

- Assign Cookies
- Lemonade Change

Medium

- Non-overlapping Intervals
- Jump Game (greedy approach)

Hard

- Job Sequencing with Deadlines (complete with profit maximization)
- Minimum Number of Arrows to Burst Balloons

Expected Learning Outcome

- Identify when a greedy approach is valid using simple exchange-argument intuition
- Solve interval-scheduling and platform/room-allocation problems
- Distinguish Fractional Knapsack (greedy) from 0/1 Knapsack (DP, covered later)

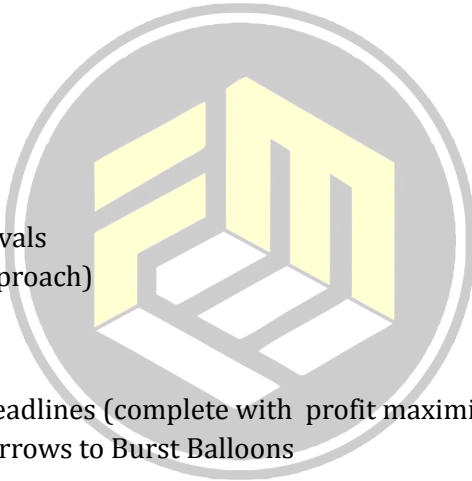
Class 46: Bit Manipulation

Class Assignment

Easy

- Check if a number is even or odd using bitwise operators
- Multiply/divide a number by 2 using bit shift

Medium



- Subsets using Bitmasking
- Find the Missing Number using XOR

Hard

- Single Number II (element appearing thrice others once)
- Maximum XOR of Two Numbers in an Array

Expected Learning Outcome

- Perform bit-level manipulation confidently (set, clear, toggle, check)
- Apply XOR tricks to solve "find the unique element" family problems
- Be ready for the mixed practice class before Graphs begin

Class 47: Practice Class — Heap + Greedy + Bit Manipulation

Class Assignment

Easy

- Last Stone Weight (timed revisit)
- Number of 1 Bits (timed revisit)

Medium

- K Closest Points to Origin (timed revisit)
- Subsets using Bitmasking (timed revisit)
- Non-overlapping Intervals (timed revisit)

Hard

- Maximum XOR of Two Numbers (timed revisit)
- Job Sequencing with Deadlines (timed revisit)

Expected Learning Outcome

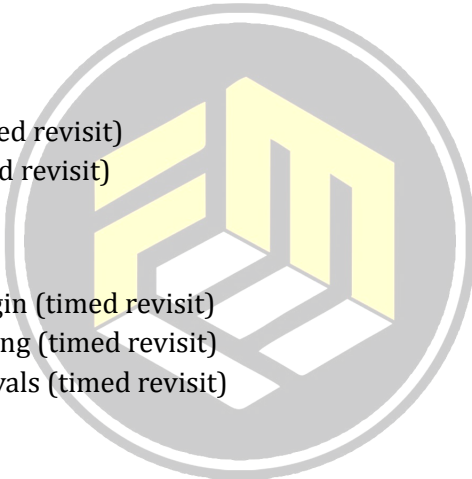
- Solve heap, greedy, and bit manipulation problems fluently under time pressure
- Recognize combined-technique problems (greedy + heap)
- Move into the Graphs module with strong supporting tools already mastered

Class 48: Graphs I — Representation

Class Assignment

Easy

- Represent a given graph using both adjacency list and matrix
- Count total edges in an undirected graph



Medium

- Find all neighbors of a given vertex
- Construct a graph from a list of prerequisite pairs (course schedule style)

Hard

- Represent a weighted graph and compute total edge weight
- Clone a Graph (adjacency list based deep copy)

Expected Learning Outcome

- Represent any graph using adjacency list or adjacency matrix
- Extract structural information (degree, neighbors, edges) from a representation
- Be fully ready for BFS traversal

Class 49: Graphs II — BFS

Class Assignment

Easy

- Print BFS traversal from a given source
- Check if a path exists between two nodes using BFS

Medium

- Bipartite Graph Check using BFS
- Shortest Path in Binary Matrix
- 01 Matrix (multi-source BFS)

Hard

- Find distance of nearest cell having 1 in a binary matrix
- As Far from Land as Possible (multi-source BFS)

Expected Learning Outcome

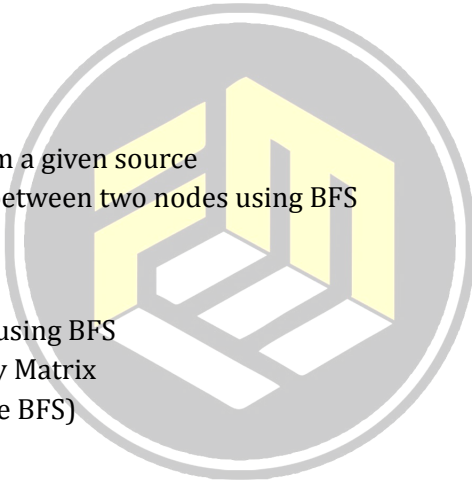
- Implement BFS traversal for both connected and disconnected graphs
- Solve shortest-path problems in unweighted graphs using BFS
- Apply multi-source BFS to grid-based problems

Class 50: Graphs III — DFS

Class Assignment

Easy

- Count number of provinces (friend circles)



- Print all connected components

Medium

- Number of Closed Islands
- Keys and Rooms
- Is Graph Bipartite (DFS approach)

Hard

- Surrounded Regions
- Number of Distinct Islands

Expected Learning Outcome

- Implement DFS traversal recursively
- Detect connected components and cycles in undirected graphs
- Solve grid-based DFS problems (islands, flood fill, area) confidently

Class 51: Graphs IV — Topological Sort

Class Assignment

Easy

- Find all sources (in-degree 0 nodes) in a DAG
- Check if a directed graph has a cycle (boolean only)

Medium

- Course Schedule (complete)
- Course Schedule II (complete)

Hard

- Find the order of compilation given dependencies
- Minimum time to complete all tasks given dependencies

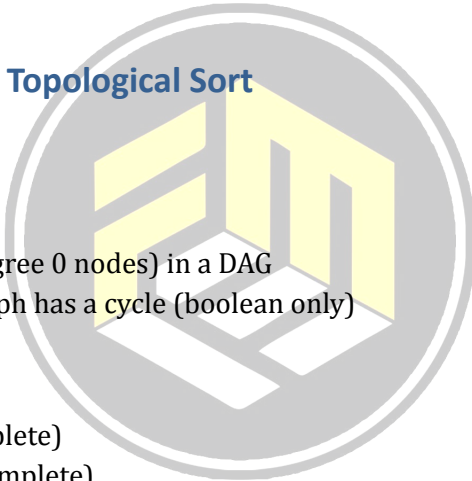
Expected Learning Outcome

- Implement both Kahn's and DFS-based topological sort
- Detect cycles in directed graphs
- Solve course-scheduling and dependency-ordering problems

Class 52: Graphs V — Union Find

Class Assignment

Easy



- Union-Find on a small given graph (manual trace)
- Check if two nodes are connected using DSU

Medium

- Accounts Merge using DSU
- Number of Operations to Make Network Connected

Hard

- Most Stones Removed with Same Row or Column
- Redundant Connection using DSU

Expected Learning Outcome

- Implement Union Find with path compression for near $O(1)$ operations
- Choose between DSU and DFS/BFS for connectivity-style problems
- Be ready for weighted shortest-path algorithms

Class 53: Graphs VI — Dijkstra's Shortest Path

Class Assignment

Easy

- Trace Dijkstra's algorithm by hand on a small given graph
- Find shortest distance from source to all nodes (small graph)

Medium

- Network Delay Time (complete)
- Cheapest Flights Within K Stops (introductory)

Hard

- Path with Minimum Effort (complete)

Expected Learning Outcome

- Implement Dijkstra's algorithm for basic weighted shortest-path problems
- Explain why Dijkstra's algorithm fails with negative edge weights
- Be fully prepared for the Graphs practice class

Class 54: Practice Class — Graphs Module

Class Assignment

Easy

- Find the Town Judge
- Find Center of Star Graph

Medium

- All Paths from Source to Target (timed revisit)
- Time Needed to Inform All Employees (BFS/DFS on tree-graph)

Hard

- Cheapest Flights Within K Stops (timed revisit)
- Evaluate Division (graph-based)

Expected Learning Outcome

- Solve any standard Graph interview problem within a time limit
- Choose the correct traversal, connectivity, or shortest-path technique confidently
- Move into Dynamic Programming with strong recursive and iterative foundations

Class 55: Dynamic Programming I — 1D DP

Class Assignment

Easy

- Nth Tribonacci Number
- Count ways to climb stairs with 1
- 2
- or 3 steps

Medium

- House Robber II (circular arrangement)
- Perfect Squares (1D DP)

Hard

- Decode Ways
- Delete and Earn

Expected Learning Outcome

- Convert a brute-force recursive solution into memoized and tabulated DP
- Identify overlapping subproblems in a new problem
- Be ready for Knapsack-style 2D DP problems



Class 56: Dynamic Programming II — 0/1 Knapsack

Class Assignment

Easy

- Check if a subset with sum K exists (small array)
- manual trace)
- 0/1 Knapsack with given weights and values (basic)

Medium

- Minimum Subset Sum Difference (complete)
- Count of Subsets with Given Sum

Hard

- Target Sum (Knapsack variant using +/- assignment)
- Ones and Zeroes (0/1 Knapsack with two constraints)

Expected Learning Outcome

- Formulate and solve 0/1 Knapsack and its DP table
- Recognize Subset Sum and Partition problems as Knapsack variants
- Be ready for Unbounded Knapsack and its variants

Class 57: Dynamic Programming III — Unbounded & Variants

Class Assignment

Easy

- Coin change with unlimited coins of given denominations (existence check)
- Minimum coins to make change (small values)

Medium

- Coin Change II (complete)
- Combination Sum IV (unbounded Knapsack framing)

Hard

- Coin Change (complete
- with edge cases for impossibility)
- Perfect Squares (Unbounded Knapsack formulation
- revisit)

Expected Learning Outcome

- Distinguish Unbounded Knapsack from 0/1 Knapsack in DP formulation

- Solve Coin Change (both minimum-coins and count-ways variants)
- Be ready for the Longest Increasing Subsequence family

Class 58: Dynamic Programming IV — LIS Family

Class Assignment

Easy

- Check if an array has an increasing subsequence of length 3
- Longest increasing subsequence length (small input manual trace)

Medium

- Maximum Sum Increasing Subsequence
- Maximum Length of Pair Chain

Hard

- Minimum Number of Removals to Make an Array Sorted (LIS-based)
- Longest Chain of Pairs

Expected Learning Outcome

- Solve LIS in both $O(n^2)$ and optimized $O(n \log n)$ approaches
- Apply LIS thinking to bitonic and chain-style variant problems
- Be ready for Grid-based DP

Class 59: Dynamic Programming V — Grid DP

Class Assignment

Easy

- Count paths in a grid without obstacles (small grid manual trace)
- Minimum path sum in a 3x3 grid (manual trace)

Medium

- Unique Paths II (complete with obstacles)
- Minimum Falling Path Sum

Hard

- Triangle (complete space-optimized)

- Maximal Square (grid DP variant)

Expected Learning Outcome

- Define DP states for 2D grid problems correctly
- Handle obstacles and constraints within grid DP
- Be ready for DP on Strings

Class 60: Dynamic Programming VI — DP on Strings

Class Assignment

Easy

- Check if two strings have a common subsequence of length K
- Count common substrings of length K

Medium

- Longest Palindromic Substring (DP approach
- contrast with earlier expand-around-center)
- Print the Longest Common Subsequence

Hard

- Edit Distance (complete with all three operations)
- Delete Operation for Two Strings

Expected Learning Outcome

- Solve LCS, LCS variants, and Edit Distance confidently
- Apply 2D DP thinking to any two-string comparison problem
- Be fully prepared for the comprehensive DP practice class

Class 61: Practice Class — DP Module

Class Assignment

Easy

- Climbing Stairs (timed revisit)
- Min Cost Climbing Stairs (timed revisit)

Medium

- Target Sum (timed revisit)
- Maximal Square (timed revisit)
- Unique Paths II (timed revisit)

Hard

- Partition Equal Subset Sum (timed revisit)
- Edit Distance (timed revisit)

Expected Learning Outcome

- Identify the correct DP category for a new, unseen problem
- Solve a mixed DP problem set within a time limit
- Move into the final module: Trie, full-course revision, and mock interviews

Class 62: Trie

Class Assignment

Easy

- Design a basic Trie from scratch (manual trace)
- Count total words in a Trie

Medium

- Design Add and Search Words Data Structure
- Top K frequent words using Trie + Heap

Hard

- Word Search II (Trie + Backtracking revisit)
- Maximum XOR of Two Numbers using a Binary Trie (bridges back to Bit Manipulation)

Expected Learning Outcome

- Implement a Trie from scratch and use it for prefix-based problems
- Combine Trie with Backtracking and Bit Manipulation for advanced problems
- Walk into the full-course revision class with all 61 core topic classes behind them

Class 63: Full Course Revision (Buffer Class)

Class Assignment

Easy

- Self-created summary sheet — one line per module naming its core technique and time complexity

Medium

- Re-attempt 2 previously assigned problems the student got wrong or skipped

Hard

- Solve 1 problem of the student's own choosing from their weakest module
- then explain the approach in writing

Expected Learning Outcome

- Recall and map every major DSA pattern covered in the course
- Identify their own weakest areas and have a concrete plan to address them
- Walk into the mock interview sessions fully warmed up

Class 64: Mock Interview Session I

Class Assignment

Easy

- Solve 2 previously unseen easy problems from Arrays/Strings under 15-minute timer each

Medium

- Solve 2 previously unseen medium problems from Trees/Graphs under 20-minute timer each

Hard

- Solve 1 previously unseen hard problem from DP under 30-minute timer

Expected Learning Outcome

- Perform under simulated interview time pressure
- Communicate their approach clearly before and during coding
- Identify their weakest pattern areas for final revision

Class 65: Mock Interview Session II + Final Interview Preparation

Class Assignment

Easy

- Self-assessment checklist — rate confidence across all 12 major topic areas covered in the course

Medium

- Solve 2 mixed-topic problems (one each from Arrays/Strings and Trees/Graphs) as a final practice set
- timed at 20 minutes each

Hard

- Complete one full simulated interview loop independently (2 problems
- 45 minutes total
- self-timed) and write a short self-review of the approach used

Expected Learning Outcome

- Approach any unseen DSA problem systematically under interview conditions
- Confidently discuss time and space complexity trade-offs live
- Walk into real interviews at service-based, product-based, and Tier-1/Tier-2 companies
- fully prepared

