COMPETATIVE PROGRAMMING

Core modules

- Pointers & Dynamic Memory Allocation
- Asymptotic Analysis
- Searching Algorithms
- Sorting Algorithms
- Stacks
- Recursion and Backtracking
- Queries
- Linked Lists
- Hashing & Chaining
- Hierarchical Data Structures
- Binary Search Tree
- Trie Data Structures
- Heaps & Priority Queues
- Graphs
- Shortest Path Algorithms
- Dynamic Programming
- Math
- Segment Tree & Binary Indexed Tree
- RMQ Sparse Table, AVL
- Querying in Tree
- Persistent Data Structures
- Strings: KMP, Z, Robin Karp, Trie, Suffix Array

Description of Core Modules

Pointers & Dynamic Memory Allocation Revisiting Theory of Pointers, Concept of Dynamic memory allocation, Structures and Unions, Analogy of Structures and Classes, An intro to Object oriented paradigm using C++ (Abstraction, Encapsulation, Polymorphism and Inheritance), Diamond Problem.

Asymptotic Analysis Exploring Asymptotic analysis, Understanding and solving Recurrence relations and working with C++'s standard template library. Concept of Dynamic arrays/vectors.

Searching Algorithms Gearing up with Searching Algorithms for Data Structures: Linear Search, Binary Search, Ternary Search. Binary search over sorted range, patterns, and domain of answers. Classical problems like 'Painter partition', 'square root evaluation', 'magnet problem' etc…

Sorting Algorithms Gearing up with Sorting Algorithms for Data Structures. O(n^2) sorting techniques: Bubble, Selection, Insertion sort. Variations of bubble sort and classical interview problems. Introduction to Divide and conquer paradigm. Learning O(n Log n) sorting : Merge sort, Quick sort. Concept of stability in sorting. Inversion counts. O(n)-based sorting algorithms: counting sort, radix sort, bucket sort. Stability and asymptotic analysis of these algorithms. Problems based of bucketizations, square root decompositions etc of a collection.

Stacks Introduction to Stack Data Structure and LIFO principle and implementation of its subroutines in C++. Understanding stack-overflows. Stock-span problem, finding next greater member of a collection, Solving problems of Histograms and Binary matrices using Stacks. Using in-built stack of C++'s standard template library.

Recursion and Backtracking Understanding the principles of Recursion and Backtracking. Solving problems involving stack data structure using recursion. Implementing classical Backtracking problems like N-Queens, Gray-Codes, Sudoku etc in C++.

Queues An introduction to Queues Data Structure and FIFO principle and implementation of its subroutines in C++. Space-based optimisation using Circular queues. Understanding Level order traversal of a recurrence tree structure using queues. Implementing Doubly ended queues, sliding window technique for problem solving. Usage of inbuilt queue of C++'s standard template library.

Linked Lists Understanding the need/motivation for dynamic collection. Implementing Linked Lists and its subroutines. Understanding malloc, calloc. Pros and cons of implementing stacks and queues using Linked Lists v/s using arrays. Implementing doubly linked lists, circular linked lists. Classical problems on Linked lists e.g. rearrangement of nodes, palindrome evaluation, intersection of lists, cycle-detection etc…

Hashing & Chaining Concept of Hashing, chaining. Understanding how ordered and unordered maps work in C++. Properties of a good hash function. Classical problems on hashing e.g. Anagrams search, pair/triplet/quadruplet finding.

Hierarchical Data Structures and Trees Motivation for hierarchical data structures. An intro to Trees. Generic N-ary tree. Implementation of Binary trees in C++ along with its subroutines. Classical problem solving on mirror-image, isomorphous and height balanced Ness of a binary tree. Tree traversals with and without recursion. More classical/interview specific question solving on trees using C++: Diameter of a tree, height of tree, constructing a unique tree from its known traversal sequences. Concept of Catalan numbers and applying it to solve problems on Trees.

Binary Search Tree Concept of Binary Search Trees, implementation in C++, Lowest common ancestors, Classical/interview specific question solving on BSTs e.g. checking for a BST, creating a balanced BST

Trie Data Structures An introduction to Trie Data Structures and implementation of its subroutines in C++. Leveraging Trie to solve String problems e.g. pattern prefix matching, palindrome pairs etc

Heaps & Priority Queues Understanding Heaps and Priority queues, implementing their subroutines in C++. Exploring STL's priority queue and its usage on structs and complex user defined data types. Classical problems on heaps e.g. Rope cutting problem, k points closest to origin, merging k Lists etc. Learning Heap sort.

Graphs Theory of Graphs. Weighted and Unweighted graphs. Graph traversals: Depth First Search (Using both recursion and stacks), Breadth First Search (Using queues), Topological Sorting, Concept of connected components. Classical problem solving in C++ like Number of islands, Detection of cycle.

Shortest Path Algorithms Shortest path Algorithms - Dijkstra's, Floyd Warshall's. Implementation of these algorithms in C++ and solving standard grid problems using these.

Dynamic Programming Concept of State, Overlapping Subproblem, 1D DP, 2D DP, Classical Problems on DP like longest increasing subsequence, Knapsack, Coin Change, Bitmasking DP, DP on Trees

Math Math - Modular arithmetic, exponentiation and inverse, extended euclidean, Chinese remainder, Sieve, ETF, Linear recurrence solving, nCr mod m, Lucas theorem, Primality tests, Gaussian elimination, FFT Strings - KMP, Z algorithm, Rabin Karp, Trie, Suffix array & LCP DP - 1D, 2D, bit-masking, iterative convergence, examples Dijkstra, Kruskal, Prim, Union Find, Floyd Warshall, Bellman Ford Advanced Data Structures - Segment Tree, Binary indexed tree

RMQ Sparse Table, AVL RMQ sparse table, AVL, HLD

Querying in Tree Querying on tree - LCA, HLD

Persistent Data Structures Persistent Data Structures - Fat node, Path copying