BCSC 0020:DESIGN & ANALYSIS OF ALGORITHMS

Objective: The objective of this course is that students will construct and application of various data structures and concepts including Trees, Recursion & Dynamic programming

Credits:03 **L-T-P : 3-0-0-0**

Module	Content	Teaching Hours
I	**Time and Space Complexity:** Introduction to Program complexity, Growth of functions (Big Oh notation), Performance measurements, Time complexity for loops, Time complexity for recursion, Time and Space complexity for various algorithms, Recurrence relations. **Binary Tree:** Implementation, Traversals on binary tree recursive & non-recursive (Preorder, Post order, in order), Delete a node, Height of BT, Diameter of BT, Path in BT, Level order traversal, Left view, Right view, Top view, Bottom view, Lowest common ancestor (LCA), Advance BT problems. **Binary Tree Search Tree:** Implementation, Traversals on BST recursive & non-recursive (Preorder, post order, In order), Complete Binary tree, Delete a node, Height of BST, Path in BST, Level order traversal, Left view, Right view, Top view, Bottom view, Lowest common ancestor (LCA), Advance BST problems. **AVL & RB Tree:** Introduction, Basic operations and properties. **Heap and Priority Queues:** Introduction to heap, Min heap, Max heap, Heap sort, Priority queue Implementation, Operations on PQ, find min/max in the continuous stream of data, Find median from data stream, Sliding window median, Merge k sorted lists, Skyline problem, Advance heap problems.	
II	**Hashing:** Introduction and Implementation of Set & Map, Symbol table, Hashing functions, Collision-Resolution techniques, Hash set, Linked-hash Set, Tree set, HashMap, Linked HashMap, Tree Map, LCS, Word pattern, Group anagram, Advance Set & Map problems. **Graph:** Introduction and Implementation, Adjacency Matrix and Adjacency List, DFS, BFS, Components, Indegree, Outdegree, Bipartite, DSU, Topological Sorting, Dijkstra's algorithm, Floyd-Warshall Algorithm, Bellman-Ford Algorithm, Kosaraju Algorithm Johnson Algorithm, Boruvka's Algorithm, Cheriton Trajan's Algorithm, Graph Colouring Algorithm: Welsh Powell Algorithm, Wigderson Algorithm, Cut edge, Cut node. **Greedy Algorithms:** Activity Selection Problem, Fractional Knapsack problem and variants, Prim's Algorithm, Kruskal's Algorithm, Bin Packing problem, Weighted Job scheduling, Interval scheduling.	

III	**Dynamic Programming:** The general method, 0/1 knapsack Subset Sum problem, Change making problem, Inclusion Exclusion Principle with DP, Matrix-chain Multiplication, longest common Subsequence Problem, Word break problem, Word wrap problem, Dice Throw Problem, Assembly Line Scheduling, DP vs Greedy, DP on String, DP Range Questions (MIN and MAX), DP(Kidane's), Longest Increasing Subsequence's (LIS) **Tries:** Implementation of Tries, Prefix Array, Overlapping and Non- Overlapping	

Online Platform: Hacker Blocks, Leetcode, Codeforces, At Coder and Spoj

Reference Books/ Textbooks / Cases:
Introduction to Algorithms 4th edition Thomas H Cormen
Programming Pearls Author: Jon Bentley. **Edition:** 2nd Edition.

Intended Outcomes:

1. **Improved Problem-Solving Skills:**
    a. Develop skills to devise efficient algorithms and strategies to solve problems within given constraints.

2. **Algorithmic Proficiency:**
    a. Gain a deep understanding of various algorithms and data structures commonly used in competitive programming.
    b. Master techniques such as dynamic programming, graph algorithms, sorting algorithms, etc.

3. **Competitive Coding Practice:**
    a. Regularly participate in online coding contests and platforms (e.g., Codeforces, Leetcode, At coder, and SPOJ).
    b. Achieve higher rankings and ratings on these platforms, showcasing improvement over time.

4. **Problem Optimization Techniques:**
    a. Learn optimization techniques to improve the time and space complexity of solutions.
    b. Understand when and how to apply these optimizations to solve problems more efficiently.

5. **Team Collaboration and Competitive Success:**