# INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal - 500 043, Hyderabad, Telangana

## COURSE CONTENT

| ADVANCED ALGORITHMS LABORATORY | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **II Semester: CSE** | | | | | | | | |
| **Course Code** | **Category** | **Hours / Week** | | | **Credits** | **Maximum Marks** | | |
| **BCSD23** | **Core** | **L** | **T** | **P** | **C** | **CIA** | **SEE** | **Total** |
| | | 0 | 0 | 4 | 2 | 40 | 60 | 100 |
| **Contact Classes: Nil** | **Tutorial Classes: Nil** | **Practical Classes: 42** | | | | **Total Classes:42** | | |
| **Prerequisites: Advanced data structures** | | | | | | | | |

### I. COURSE OVERVIEW:
Advanced Algorithms Laboratory typically focuses on ensuring that students acquire practical skills in designing, analyzing, and implementing advanced algorithms. Students should demonstrate a comprehensive understanding of various advanced algorithms, including but not limited to dynamic programming, greedy algorithms, graph algorithms, and divide and conquer**.**

### II. COURSE OBJECTIVES:
The students will try to learn:
I. The various designing techniques and methods for algorithms.
II. The performance analysis of Algorithms using asymptotic and empirical approaches

### III. COURSE OUTCOMES:
After successful completion of the course, students will be able to:
CO1 Make use of the greedy method to design algorithms for complex problems and analyze their performance.
CO2 Apply hash, Red Block and binary search tree data structures to various problems.
CO3 Make use of the divide and conquer method to design algorithms for complex problems and analyze their performance.
CO4 Apply dynamic programming for complex problems and analyze their performance.
CO5 Analyze the flow concepts of Min-Max algorithms.
CO6 Apply substitution, recurrence-tree methods to solve recurrences.

### IV. COURSE CONTENT:
**Week 1:**
Design and implement an algorithm to solve the maximum-subarray problem with improved time complexity.

**Week 2:**
Develop an efficient implementation of Strassen's algorithm for matrix multiplication. Investigate techniques to optimize the algorithm's performance in terms of both time and space complexity.

**Week 3:**
Develop an efficient implementation of the Substitution Method for Solving Recurrences. Investigate techniques to optimize the algorithm's performance in terms of both time and space complexity

**Week 4:**
Develop an efficient implementation of the recurrence-tree method for Solving Recurrences. Investigate techniques to optimize the algorithm's performance in terms of both time and space complexity

**Week 5:**
Design and implement Red-Black Trees that optimize dynamic operations such as insertions and deletions. Investigate scenarios where the standard Red-Black Tree operations might be suboptimal and propose enhancements to minimize the time complexity of these dynamic operations.

**Week 6:**
Design and implement binary search Trees that optimize dynamic operations such as insertions and deletions. Investigate scenarios where the standard binary search Tree operations might be suboptimal and propose enhancements to minimize the time complexity of these dynamic operations.

**Week 7:**
Given a File of N employee records with a set K of Keys(4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table(HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Design and develop a Program in C that uses the Hash function H: K * L as H(K)=K mod m (remainder method) and implement a hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.

**Week 8:**
Develop an efficient implementation of the matrix chain multiplication using dynamic programming. Investigate techniques to optimize the algorithm's performance in terms of both time and space complexity

**Week 9:**
Develop an efficient implementation of the Huffman codes using **a** greedy strategy. Investigate techniques to optimize the algorithm's performance in terms of both time and space complexity

**Week 10:**
Implement the Bellman-Ford Algorithm for solving the single-source shortest path algorithm. Write codes to input the graph as a linked list. Here node of the graph should also contain a field to store cost. Check the output for negative cost edges/negative cycles.

**Week 11:**
Implement of Ford-Fulkerson algorithm for solving the maximum flow problem in a network. Write code to input the network as a directed graph as a linked list. Compute the minimum capacity of the cut in the network. Compare the maximum flow with the capacity of the minimum cut.

**Week 12:**
Implement of Edmond- Karp maximum-flow algorithm for solving the maximum flow problem in a network.

**Week 13:**
Develop an efficient implementation of the Floyd-Warshall algorithm using dynamic programming. Investigate techniques to optimize the algorithm's performance in terms of both time and space complexity

**Week 14:**
Develop an efficient implementation of the greedy selection criteria for the Knapsack Problem. Investigate techniques to optimize the algorithm's performance in terms of both time and space complexity

## V. TEXT BOOKS:


1. Cormen, T.H., Leiserson,C.E., Rivest, R.L. Stein,C. “Introduction to Algorithms” Prentice-Hall of India Learning Pvt. Ltd, 4th edition, 2022.


## VII.REFERENCE BOOKS:


1. Ellis Horowitz, Satraj Sahni and Rajasekharan “Fundamentals of Computer Algorithms”, Universities Press, 2008.
2. Aho, Hopcroft, Ullman "The Design and Analysis of Computer Algorithms", Pearson Education, 2nd edition, 2018.
3. Kleinberg and Tardos "Algorithm Design", Pearson Education, 2nd edition, 2016.

## VIII. WEB REFERENCES:


1. https://www.scribd.com/document/445971276/Advanced-Algorithms-Lab-Manual
2. https://people.iitism.ac.in/~download/lab%20manuals/mathandcomp/Advanced%20Data%20Structures%20And%20Algorithms.pdf