

Subject Code : 2CS2010208	Subject Title: DESIGN AND ANALYSIS OF ALGORITHMS (DAA)
Pre-requisite :	Programming Language: C, Data Structure

Course Objective:

The objectives of the course are to:

- To understand alternate methods of writing algorithms under various categories, such as Divide-and-Conquer, Dynamic Programming, Greedy Methods, Backtracking, Branch & Bound, etc.
- To be able to analyze algorithms by working out complexity of algorithms
- To understand the basics of P, NP, and NP Complete problems

Teaching Scheme (Hours per week)				Evaluation Scheme (Marks)				
Lecture	Tutorial	Practical	Credit	Theory		Practical		Total
				University Assessment	Continuous Assessment	University Assessment	Continuous Assessment	
3	-	3	6	60	40	30	20	150

Subject Contents			
Sr. No	Topic	Total Hours	Weight (%)
1	Basic Concepts of Analysis and Design of Algorithms Introduction; Overview of basic steps to solve a problem using computer; Examples; An overview of Top-down design and Recursion; Correct use of Loops in programs, Factors affecting efficiency of algorithms, Estimating and specifying execution times, Order notation: Big-oh, Omega, Theta Design using Recursion: Introduction, Execution trace through examples	12	25
2	Algorithms Using Divide-and-Conquer Strategy Introduction, Examples, Limitations of Divide-and-Conquer strategy; Greedy Methods Introduction; Examples: Knapsack problem, Job sequencing with deadlines, Minimum spanning trees, Prim's algorithm, Kruskal's algorithm, Dijkstra's shortest path algorithm	10	20
3	Dynamic Programming Introduction; Examples: Rod cutting problem, Multistage graphs, Traveling salesman problem, Matrix multiplication, Longest common sub-sequence, Maximum flow problem	8	20
4	Backtracking Combinatorial search; Search and traversal: BFS, DFS; Backtracking strategy and examples: 8-Queen problem; Backtracking Framework : M-Coloring Problem, Hamiltonian Circuits Branch-and-Bound algorithms , Examples: Shortest path, 16-Puzzle and 8-Puzzle, 0/1 Knapsack problem, Traveling salesman problem.	10	20
5	Efficiency of Algorithms; Complexity Calculation and Categorization Polynomial-time and Non-polynomial-time algorithms; Worst and average case behavior, Probabilistic average case analysis; Time analysis of algorithms, Examples: Matrix multiplication; Efficiency of recursion Examples of Complexity Calculation: Sorting examples: Radix sort, Simple Insertion sort, Quick sort, Heap sort, Merge sort; Summary of complexity and characteristics of sorting algorithms; Complexity Categorization of Problems: Introduction, P, NP, NPC, NPH, etc	8	15

Course Outcome:

At the end of this course, the student would be able

- To decide on an appropriate category of algorithms for solving a given problem.
- To carry out complexity of algorithms and decide an efficient algorithm for the task on hand.
- To have an idea about categorization of problems into P, NP, NPC, NPH.

List of References:

1. Parag H Dave, Himanshu B Dave, "Design and Analysis of Algorithms", Pearson (2014)
2. Thomas H. Cormen, Charles E. Leiserson, Ronald L Rivest, Clifford Stein, "Introduction to Algorithms", PHI, 2nd Edition
3. Anany Levitin, "Introduction to Design and Analysis of Algorithms", Pearson (2014)
4. S. Baase, "Computer Algorithms: Introduction to Design and Analysis", Pearson (2002)
5. Kleinberg, "Algorithm Design", Pearson (2013)

List of Experiments:

Note: The experiment list provided beneath is for reference only and has to perform in C language. The course teacher may Change/formulate it as per his/her methodology and requirement.

1. Find square root of a number. Can we use Divide & Conquer approach for this problem?
2. Determine smallest divisor of an integer.
3. For a given value of n, generate prime numbers $\leq n$ (more than one algorithms are possible)
4. Find X^n . Iterative and recursive algorithms are possible.
5. Find Factorial of n. Iterative and recursive algorithms are possible.
6. Generate Fibonacci series up to n terms Iterative and recursive algorithms are possible.
7. Implement Binary Search of an ordered array. Iterative and Recursive algorithms are possible.
8. Sort a given sequence of numbers using (a) Bubble Sort, and (b) Merge Sort
9. Knapsack problem using Greedy algorithm.
10. Solution of Rod-cutting problem using Dynamic Programming algorithm.
11. Matrix multiplication using Dynamic Programming algorithm.
12. Breadth First Search (BFS) in a binary tree.
13. Depth First Search (DFS) in a binary tree.
14. 8-Puzzle and 16-Puzzle.