CODE	COURSE NAME	CATEGORY	L	T	P	CREDIT
20MCA203	DESIGN & ANALYSIS OF ALGORITHMS	CORE	3	1	0	4

Preamble: The syllabus is prepared with a view to provide a strong foundation to students in design and analysis of computer algorithms and to introduce them the advanced topics such as Network Flows, Approximation algorithms and Randomised algorithms.

Prerequisite: Knowledge in Data Structures

Course Outcomes: After completion of the course the student will be able to

CO No.	Course Outcome (CO)	Bloom's Category Level
CO 1	Discuss the basic concepts in computer algorithms and their analysis & design using Divide and Conquer.	Level 2: Understand
CO 2	Explain the concepts of Greedy Strategy and Dynamic Programming to use it in solving real world problems.	Level 3: Apply
CO 3	Explain the Branch & Bound technique, Backtracking technique and Lower bounds.	Level 2: Understand
CO 4	Describe the fundamental concepts of Computational Complexity and Network Flows.	Level 2: Understand
CO 5	Discuss the concepts of Approximation and Randomised Algorithms.	Level 2: Understand

Mapping of Course Outcomes with Program Outcomes

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
CO 1	3	3	1	2			2					
CO 2	3	3	1	2			2					
CO 3	3	3	1	2			2					
CO 4	3	3	1	2			2					
CO 5	3	3	1	2			2					

3/2/1: High/Medium/Low

Assessment Pattern

Bloom's Category Levels	Continuous Assessment Tests		End Semester Examination
	1	2	
Level 1: Remember	20	20	20
Level 2: Understand	20	30	30
Level 3: Apply	10		10
Level 4: Analyse			
Level 5: Evaluate			
Level 6: Create			

Mark distribution

Total Marks	Continuous Internal Evaluation (CIE)	End Semester Examination (ESE)	ESE Duration
100	40	60	3 hours

Continuous Internal Evaluation Pattern:

Attendance : 8 marks
Continuous Assessment Test (2 numbers) : 20 marks
Assignment/Quiz/Course project : 12 marks

End Semester Examination Pattern: There will be *two* parts; **Part A** and **Part B**. Part A contain 10 questions with 2 questions from each module, having 3 marks for each question. Students should answer *all* questions. Part B contains 2 questions from each module of which student should answer *any one*. Each question can have a maximum 2 subdivisions and carry 6 marks.

Sample Course Level Assessment Questions

Course Outcome 1 (CO 1):

1. Define "Time Complexity" of an algorithm?
2. What is the need for analysing an algorithm?
3. Define Big Oh Notation.
4. Define the terms Best Case, Worst Case and Average case complexities.
5. Explain the Merge Sort algorithm with an example.

Course Outcome 2 (CO 2):

1. Explain the Greedy Control abstraction.
2. Write the Prim's algorithm and illustrate with an example.
3. State and illustrate the Principle of Optimal Substructure.
4. Explain a solution to the Travelling Salesman problem using Dynamic Programming.

Course Outcome 3 (CO 3):

1. Explain the N-Queen's problem and its solution using Backtracking.
2. Explain the 8-puzzle problem and illustrate how it can be solved using Branch and Bound.
3. Bring out the notion of Decision Trees.
4. What is the lower bound of the time complexity of Comparison based sorting algorithms?

Course Outcome 4 (CO 4):

1. Define class P and NP.
2. What is Polynomial Time Reduction?
3. Show that the Clique problem is NP-Complete.
4. Define the Terms - Flow Network and Network Flow.
5. Explain the Ford-Fulkerson Algorithm.

Course Outcome 5 (CO 5):

1. What is an Approximation algorithm?
2. Describe the 2-approximation algorithm for Vertex Cover problem.
3. What is a Randomised algorithm?
4. Explain the Schwartz-Zippel Lemma. How this Lemma can be used to test the identity of two polynomials.

Model Question Paper

Course Code: 20MCA203

Course Name: Design and Analysis of Algorithms

Max. Marks :60 Duration: 3 Hrs

Part A

Answer all questions. Each question carries 3 marks (10 * 3 = 30 Marks)

1. Define Big Oh notation.
2. Write the control abstraction for a typical Divide and Conquer algorithm.
3. Explain a Greedy strategy which can give the optimal solution for the Knapsack problem.
4. Write a dynamic programming algorithm to compute the factorial of a number.
5. How does Backtracking differ from Branch and Bound?
6. Using a decision tree, show that any search algorithm which searches a given key within an array of n elements must perform at least O(ln n) comparisons in the worst case.
7. What do you mean by the term Polynomial time reduction?
8. Define the term Network Flow and illustrate with an example.

9. What do you mean by approximation ratio of an Approximation algorithm?

10. What is meant by a Randomised Algorithm?

Part B

Answer all questions. Each question carries 6 marks. (5 * 6 = 30 Marks)

11	Write the Linear Search Algorithm and analyse the best, worst and average case complexities of the algorithm.	6
	OR	
12	Explain the Merge Sort algorithm and give its worst-case analysis.	6
13	Write Kruskal's algorithm to compute the minimum cost spanning tree.	6
	OR	
14	Explain the dynamic programming algorithm for the Travelling Salesman problem.	6
15	Write the Backtracking algorithm for N-Queen Problem.	6
	OR	
16	Explain the 8-puzzle problem and its solution using branch and bound technique.	6
17	Show that the Clique problem is NP-Complete.	6
	OR	
18	Describe the Ford Fulkerson’s procedure to compute the Max-Flow within a given Flow Network.	6
19	Explain the 2-approximation algorithm for Vertex Cover and justify its approximation ratio.	6
	OR	
20	Describe Randomised Quick sort.	6

Syllabus

Module 1: (8 Hours)
Review of Algorithm Analysis: Time and Space Complexity, Asymptotic Notations, Recurrence Equations, Solving Recurrence Equations- Substitution method and Iteration method. **Divide and Conquer**: Control Abstraction, Merge Sort, Quick Sort, Matrix Multiplication.
Module 2: (9 Hours)
Greedy Strategy: Control Abstraction, Knapsack Problem, Minimal Spanning Tree Algorithms- Prim's and Kruskal's Algorithm, Job Scheduling with deadlines **Dynamic Programming**: Control Abstraction, Principle of Optimal Substructure, All Pairs shortest path problem, Travelling Salesman Problem, Bellman-Ford Algorithm
Module 3:(7 Hours)
Backtracking: Control Abstraction, N-Queens problem, Sum of Subsets Problem **Branch and Bound**: Control Abstraction, 8- Puzzle problem **Lower Bounds:** The Decision Tree method, Lower Bounds for Comparison based Sort and Searching *(Analysis not required)*
Module 4: (11 Hours)
Complexity Theory: Class P and NP, Polynomial time reductions, Class NP Hard and NP-Complete, Example Problems- Vertex Cover problem, Clique Problem. **Network Flows**: Flow Networks and Network Flow, Max- Flow Min Cut Theorem, Ford Fulkerson method, Bipartite matching (*Analysis not required)*
Module 5: (10 Hours)
Introduction to Approximation Algorithms: Approximation Ratio, 2-approximation algorithm for Vertex Cover problem, Vertex Cover Approximation using Linear Programming and LP Rounding Algorithm. **Introduction to Randomised Algorithms**: Review of Basic Probability, Schwartz-Zippel Lemma and Polynomial Identity Testing, Randomized Quick Sort (*Proof of Expected Worst Case Analysis not required)*

Text Books

1. Thomas H. Cormen, et al., "Introduction to Algorithms", Prentice Hall, 3rd Edition (2010)
2. Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran, "Fundamentals of Computer Algorithms", Orient Longman, Universities Press, 2nd Edition (2008)

Reference Books

1. Richard Neapolitan, Kumarss Naimipour, "Foundations of Algorithms", Jones and Bartlett Publishers, Inc, 4th Edition (2011).
2. Sara Baase, Allen Van Gelder, "Computer Algorithms: Introduction to Design and Analysis", Pearson India, 3rd Edition (2002).
3. A. Levitin, "Introduction to the Design & Analysis of Algorithms", Pearson Education, 3rd Edition (2008).

Course Contents and Lecture Schedule

Sl. No.	Topic	No. of Lectures
1	**Review of Algorithm Analysis and Divide & Conquer**	**8 Hours**
1.1	Time and Space Complexity	1
1.2	Asymptotic Notations	1
1.3	Recurrence Equations, Solving Recurrence Equations- Substitution method	1
1.4	Iteration method	1
1.5	Divide and Conquer: Control Abstraction, Merge Sort, Merge Sort Analysis	2
1.6	Quick Sort, Quicksort analysis	1
1.7	Matrix Multiplication	1
2	**Greedy Strategy and Dynamic Programming**	**9 Hours**
2.1	Greedy Strategy: Control Abstraction, Knapsack Problem	1
2.2	Minimum Cost Spanning Tree	1
2.3	Prim's algorithm	1
2.4	Kruskal's algorithm	1
2.5	Job Scheduling with deadlines	1
2.6	Dynamic Programming: Control Abstraction, Principle of Optimal substructure	1
2.7	All Pairs shortest path problem	1
2.8	Travelling Salesman Problem	1
2.9	Bellman-Ford Algorithm	1

3	**Backtracking, Branch & Bound, Lower Bounds**	**7 Hours**
3.1	Backtracking: Control Abstraction N- Queens problem	1
3.2	Sum of subsets problem	1
3.3	Branch and Bound: Control Abstraction 8- Puzzle problem	1
3.4	Lower Bounds: The Decision Tree method	2
3.5	Lower Bounds for Comparison based Sorting	1
3.6	Lower bounds for searching	1
4	**Computational complexity, Network Flows**	**11 Hours**
4.1	Class P, NP	1
4.2	Polynomial Time Reductions	1
4.3	Class NP-Hard and NP-Complete	2
4.4	Vertex Cover Problem	1
4.5	Clique problem	1
4.6	Flow Networks and Network Flows	2
4.7	Max Flow Min Cut Theorem	1
4.8	Ford Fulkerson’s method	1
4.9	Bipartite matching	1
5	**Approximation & Randomised Algorithms**	**10 Hours**
5.1	Approximation algorithms- introduction, Approximation Ratio	1
5.2	2- approximation algorithm for Vertex Cover problem	1
5.3	Vertex Cover Approximation using Linear Programming and LP Rounding Algorithm	2
5.4	Randomized Algorithms: introduction, Review of Basic Probability	1
5.5	Review of Basic probability	2
5.6	Schwartz-Zippel Lemma and Polynomial Identity Testing	2
5.7	Randomized Quick Sort	1