CS1105: Design and Analysis of Algorithms

Course Title and Code: Design and Analysis of Algorithms: CS1105	
Hours per Week	**L-T-P: 3-0-4**
Credits	**4 (CSE)**

Course Objective:

This course introduces an understanding of the design and analysis of algorithms. The course aims to develop a familiarity with important algorithms and data structures and an ability to analyze the asymptotic performance of algorithms. It will equip the students to apply important algorithmic design paradigms and methods of analysis to develop efficient algorithms in common engineering design situations.

Course Outcome:

On successful completion of this course, the students should be able to:

CS1105.1. Analyze the complexity of different algorithms using asymptotic analysis.
CS1105.2. Analyze and select an appropriate data structure for a computing problem.
CS1105.3. Differentiate between different algorithm designs technique: Divide and Conquer Technique, Greedy, Backtracking, and Dynamic Programming. Also, recognize when an algorithmic design situation calls for using these.
CS1105.4. Develop algorithm and programs using Divide and Conquer technique to solve various computing problems, e.g., Sorting, Strassen's matrix multiplication, and Closest pair.
CS1105.5. Develop energy-efficient algorithms and programs using Greedy approach to solve various computing problems, e.g., Minimum Spanning Trees, Shortest Path, Knapsack, Job scheduling, Graph coloring etc.
CS1105.6. Develop algorithms and programs using Backtracking technique to solve various computing problems, e.g., N queen, Hamiltonian Cycle detection, Travelling salesman, and Network flow.
CS1105.7. Develop algorithms and programs using Dynamic Programming technique to solve various computing problems, e.g., Knapsack, Shortest path, Coinage, Matrix Chain Multiplication, Longest common subsequence.
CS1105.8. Apply Query optimization algorithms using Greedy and Dynamic programming approaches.
CS1105.9. Apply various search-based problem-solving methods e.g., Uninformed search (BFS, DFS, DFS with iterative deepening), Heuristics, and Informed search (hill-climbing, generic best-first, A*).
CS1105.10. Evaluate and apply appropriate energy efficient algorithmic design technique for solving complex computing problem.
CS1105.11. Explain the ways to analyze randomized algorithms (expected running time, probability of error).
CS1105.12. Differentiate between P, NP, NP-Complete, and NP-Hard problems.

Prerequisites: Nil

Sr. No	Specifications	Marks
1	Attendance	Nil
2	Assignment	10
3	Class Participation	10

4	Quiz	Nil
5	Theory Exam-I	Nil
6	Theory Exam-II	10
7	Theory Exam-III	30
8	Report-I	Nil
9	Report-II	Nil
10	Report-III	Nil
11	Project-I	Nil
12	Project-II	Nil
13	Project-III	20
14	Lab Evaluation-I	Nil
15	Lab Evaluation-II (lab exam)	20
16	Course Portfolio	Nil
17	Presentation	Nil
18	Viva	Nil
	Total (100)	100

Retest Evaluation Scheme		
1	Theory Exam–III	30
	Total (30)	**30**

Syllabus (Theory):

UNIT I: Introduction: Algorithms, Analyzing algorithms, Complexity of algorithms, Growth of functions, Performance measurements, Types of approaches.

UNIT II: Selection sort, Bubble sort, Insertion Sort, Shell sort, Quick sort, Merge sort, Heap sort, sorting in linear time: Radix sort, Counting Sort, Comparison of sorting algorithms, Divide and Conquer with examples such as Sorting, Matrix Multiplication, Convex hull and Searching

UNIT III: Greedy methods with examples such as Optimal Reliability Allocation, Knapsack, Minimum Spanning trees – Prim's and Kruskal's algorithms, Single-source shortest paths - Dijkstra's and Bellman-Ford algorithms.

UNIT IV: Dynamic programming with examples such as Knapsack, all pair shortest paths – Warshal's and Floyd's algorithms, Resource allocation problem, Backtracking, Branch and Bound with examples such as Travelling Salesman Problem.

UNIT V: Selected Topics: String Matching, Huffman Coding, Theory of NP-completeness, Approximation algorithms and Randomized algorithms.

Text Book(s)

1. Thomas H. Coreman, Charles E. Leiserson and Ronald L. Rivest, "Introduction to Algorithms", Prentice Hall of India. 2002.
2. S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, "Algorithms", McGraw-Hill Education, 2006.

Reference Book(s)

1. RCT Lee, SS Tseng, RC Chang and YT Tsai. Introduction to the Design and Analysis of Algorithms. Mc Graw Hill, 2005.
2. E. Horowitz & S Sahni. Fundamentals of Computer Algorithms. 1984
3. Berman, Paul. Algorithms. Cengage Learning. 2002
4. Aho, Hopcraft, Ullman, The Design and Analysis of Computer Algorithms. Pearson Education, 2008.

Syllabus (Practical):

1. **SEARCHING AND SORTING BASED PROBLEMS**
   - I. Implement an algorithm to find an element in a matrix in which each row and each column is sorted.
   - II. Implement an efficient algorithm to find a majority element in an array. A majority element is one whose number of occurrences is more than half the size of the array.
   - III. Given an array [a1 to an] and we must construct another array [b1 to bn] where bi = a1*a2*...*an/ai. You are allowed to use only constant space and the time complexity is O(n). No divisions are allowed
   - IV. Implement the following sorting algorithms: Insertion, Selection, Bubble, Count, Shell, Radix
2. **DIVIDE AND CONQUER**
   - I. Write a program to implement the merge sort using recursive and non-recursive procedures.
   - II. To implement finding greatest common divisor between two positive integers.
   - III. To implement Matrix Multiplication and analyze its time complexity.
   - IV. To implement Quick sort on the given list of elements by considering pivot as the median of the 3 values first, middle and last value.
3. **GREEDY AND DYNAMIC PROGRAMMING**
   - I. To implement Longest Common Subsequence problem and analyze its time complexity.
   - II. To implement minimum spanning tree using Kruskal's and Prim's algorithms.
   - III. To implement Dijkstra's algorithm and analyze its time complexity.
   - IV. To implement Job sequencing problem using greedy approach
   - V. To find whether a set of integers can be divided into two subsets such that the sum of elements in each set is equal using dynamic programming.
   - VI. To implement 0/1 knapsack using dynamic programming.
4. **BACKTRACKING AND BRANCH-BOUND TECHNIQUES**
   - I. To implement graph coloring problem using backtracking
   - II. To implement DFS graph search algorithm
   - III. To implement Travelling Salesman problem using backtracking.
5. **STRING MATCHING**
   - I. To implement naïve String-Matching algorithm.
   - II. To implement Rabin Karp algorithm using.
   - III. To implement Knuth Morris Pratt algorithm and analyze its time complexity.
6. **PROBLEM SOLVING BY SEARCH**
   - I. To implement uninformed and informed search techniques for problem solving
   - II. To solve 8 puzzle problem

III. To solve n-queen problem

NPTEL Swayam Course:

1. https://nptel.ac.in/courses/106/106/106106127/
2. https://nptel.ac.in/courses/106/102/106102064/
3. http://www.nptelvideos.in/2012/11/data-structures-and-algorithms.html

Course Outcomes	PO 1	PO2 a	PO2 b	PO2 c	PO3 a	PO3 b	PO3 c	PO4 a	PO4 b	PO4 c	PO5 a	PO5 b	PO 6	PO7 a	PO7 b	PSO 1	PSO 2
CS1105.1	2		1		2											2	2
CS1105.2	2		1		2				1							2	2
CS1105.3	2		1		2				1							2	2
CS1105.4	2		1		1				1							2	2
CS1105.5	1		1		1				1							2	2
CS1105.6	1															2	2
CS1105.7	1		1		1				2							2	2
CS1105.8	1							1						1			2
CS1105.9	1				1			1	1					1	1	2	2
CS1105.10								1						1		2	2
CS1105.11	1		1		1			1						1			1
CS1105.12	1		1		1			1						1			1