Course Title: Analysis & Design of Algorithm

Course Level: UG

Course Code: CSE303

Course Objectives:
The designing of algorithm is an important component of computer science. The objective of this course is to make students aware of various techniques used to evaluate the efficiency of a particular algorithm. Students eventually should learn to design efficient algorithm for a particular program.

Pre-requisites: Data Structures

Course Contents/Syllabus:

<table>
<thead>
<tr>
<th>Module I:</th>
<th>Weightage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction: Algorithm Design paradigms - motivation, concept of algorithmic efficiency, run time analysis of algorithms, Asymptotic Notations. Recurrences- substitution method, recursion tree method, master method</td>
<td>20%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module II:</th>
<th>Weightage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Divide and Conquer: Structure of divide-and-conquer algorithms: examples; Binary search, quick sort, Merge sort, Strassen matrix multiplication; Run time analysis of divide and conquer and recurrence relations. Greedy Method: Overview of the greedy paradigm examples of exact optimization solution (minimum cost spanning tree), approximate solution (Knapsack problem), Single source shortest paths problems, traveling salesman problem</td>
<td>20%</td>
</tr>
</tbody>
</table>
### Module III:

**Dynamic Programming:** Overview, difference between dynamic programming and divide and conquer technique. Applications: Shortest path in graph, chain matrix multiplication, Traveling salesman Problem, longest Common sequence problem, knapsack problem

### Module IV:

**Graph Searching and Traversal:** Overview, Representation of graphs, strongly connected components, Traversal methods (depth first and breadth first search) and its analysis  
**Back tracking:** Overview, 8-queen problem, and Knapsack problem  
**Branch and bound:** LC searching Bounding, FIFO branch and bound, LC branch and bound application: 0/1 Knapsack problem, Traveling Salesman Problem

### Module V:

**Computational Complexity:** Complexity measures, Polynomial Vs non-polynomial time complexity; NP-hard and NP-complete classes, examples.

### Student Learning Outcomes:

- Ability to apply knowledge of mathematics, science, engineering and computing appropriate to the discipline.
- Ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- Ability to design, implement, and evaluate a computer-based system, process, component, or programmer to meet desired needs.
- Ability to use current techniques, skills, and tools necessary for computing practice.

### Pedagogy for Course Delivery:

The class will be taught using lecture and practical based method. In addition to giving the assignments and tutorials, the course instructor will spend considerable time in transforming theoretical concepts in practical oriented approach. The lab programs will be solve by using C/C++.

### Analysis and Design of Algorithm Lab

**Mandatory Programs**

1. Implement Recursive Binary search and determine the time taken to search an element. Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n.
2. Sort a given set of elements using Quick sort method and determine the time taken to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n.
4. Implement 0/1 Knapsack Problem using Dynamic Programming method.
5. From a given starting node in a digraph, print all the nodes reachable by using BFS/DFS method.
6. Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s/Kruskal’s algorithm.
7. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra’s algorithm.
8. Consider the problem of N queen on an (NxN) chessboard. Two queens are said to attack each other if they are on the same row, column, or diagonal. Implement backtracking algorithm to solve the problem i.e. place N non-attacking queens on the board.
10. Implement Traveling Salesman problem based on Branch and Bound technique.

1. Develop a Hamiltonian Path in an undirected graph is a path that visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a Hamiltonian Path such that there is an edge (in graph) from the last vertex to the first vertex of the Hamiltonian Path. Develop a program to implement the solution of Travelling Salesman Problem by considering the Hamiltonian cycle approach.

2. Given two sequences, a subsequence is a sequence that appears in the same relative order, but not necessarily contiguous. For example, “abc”, “abg”, “bdf”, “aeg”, „acefg”, etc are subsequences of “abcdefg”. So a string of length n has 2^n different possible subsequences. It is a classic computer science problem, the basis of file comparison programs and has applications in bioinformatics. Develop a program to implement the solution of Longest Common Subsequence problem.

3. A road network can be considered as a graph with positive weights. The nodes represent road junctions and each edge of the graph is associated with a road segment between two junctions. The weight of an edge may correspond to the length of the associated road segment, the time needed to traverse the segment or the cost of traversing the segment. Using directed edges it is also possible to model one-way streets. Such graphs are special in the sense that some edges are more important than others for long distance travel (e.g. highways). This property has been formalized using the notion of highway dimension. There are a great number of algorithms that exploit this property and are therefore able to compute the shortest path a lot quicker than would be possible on general graphs. Develop a program to find the shortest path from each node to solve the road network problem.

4. There are a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent to each task and exactly one task to each agent in such a way that the total cost of the assignment is minimized. If the numbers of agents and tasks are equal and the total cost of the assignment for all tasks is equal to the sum of the costs for each agent (or the sum of the costs for each task, which is the same thing in this case), then the problem is called the linear assignment problem. Develop a program to solve the job assignment problem.
5. Suppose that we are designing a program to simulate the storage and search in a dictionary. Words appear with different frequencies, however, and it may be the case that a frequently used word such as "the" appears far from the root while a rarely used word such as "conscientiousness" appears near the root. We want words that occur frequently in the text to be placed nearer to the root. Moreover, there may be words in the dictionary for which there is no definition. Write a program to organize an optimal binary search tree that simulates the storage and search of words in a dictionary.

Assessment/Examination Scheme:

<table>
<thead>
<tr>
<th></th>
<th>Theory L/T (%)</th>
<th>Lab/Practical/Studio (%)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>80%</td>
<td>20%</td>
<td>100</td>
</tr>
</tbody>
</table>

Theory Assessment (L&T):

<table>
<thead>
<tr>
<th>Components (Drop down)</th>
<th>Continuous Assessment/Internal Assessment</th>
<th>End Term Examination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attendance</td>
<td>Class Test</td>
</tr>
<tr>
<td>Weightage (%)</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

Lab Assessment

<table>
<thead>
<tr>
<th>Components (Drop down)</th>
<th>Continuous Assessment/Internal Assessment</th>
<th>End Term Examination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attendance</td>
<td>Lab Record</td>
</tr>
<tr>
<td>Weightage (%)</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>
Text & References:

Text Books:

Reference Books:
2. J.E Hopcroft, J.D Ullman, “Design and analysis of algorithms” TMH Publication