UNIVERSITY OF PETROLEUM & ENERGY STUDIES

2022-23 Batch

CSEG 2021	Design and Analysis of Algorithms	L	T	P	C
Version 1.0		3	0	0	3
Pre-requisites/Exposure	Basic knowledge Mathematics and data structure				
Co-requisites	–				

Course Objectives

1. To understand the necessity of the algorithm design.
2. To write the algorithm to solve a problem.
3. To analyze the performance of the algorithm.
4. To implement the algorithm in C/C++.

Course Outcomes

On completion of this course, the students will be able to

CO1. Apply mathematical techniques to find the complexity of an algorithm.
CO2. Analyze algorithms and express asymptotically different case behavior.
CO3. Demonstrate good principles of algorithm designs.
CO4. Design appreciate data structures to reduce the complexity of an algorithm.
CO5. Differentiate among P, NP Hard and NP Complete problems.

Catalog Description

This course deals with various aspects of designing algorithms and their mathematical characteristics. The broad focus lies on computational complexity, divide-and-conquer approach, dynamic programing, greedy approach and backtracking algorithms. The clear distinction among P, NP Hard and NP Complete problems are covered in detail.

Course Content

UNIT I:

Introduction **9 Lecture Hours**

Algorithm, Psuedo code, Performance Analysis- Space complexity, Time complexity, Asymptotic Notation- Big oh notation, Omega notation, Theta notation with numerical, different algorithm design techniques, recurrence relation, solving methods: substitution, recursion tree, master theorem with numerical.

UNIT II

Divide And Conquer **6 Lecture Hours**

Binary search, Quick sort: best case & worst case analysis, Merge sort, Strassen's matrix multiplication

UNIT III:

Greedy Method **6 Lecture Hours**

Activity selection problem, knapsack problem, Minimum cost spanning trees: Prims and kruskal, Single source shortest path problem: Bellman ford, dijkstra's, Huffman codes.

UNIT IV

DYNAMIC PROGRAMMING **5 Lecture Hours**

Matrix chain multiplication, 0/1 knapsack problem, All pairs shortest path problem, largest common subsequence.

UNIT V

Sorting In Linear Time **6 Lecture Hours**

Lower Bounds For Sorting, Counting Sort, Radix Sort, bucket sort

Backtracking: N-queen problem, sum of subsets problem

UNIT VI

Branch and Bound Method And Its Applications **4 Lecture Hours**

Travelling salesman problem

NP-Hard and NP-Complete problem and concepts

Text Books

1. Thomas H. Cormen (2009) Introduction to Algorithm (Third Edition), The MIT Press. ISBN: 978-0-262-03384-8
2. John Kleinberg and Eva Tardos (2005), Algorithm Design, ISBN: 0-321-29535-8

Reference Books

CSEG 2103	Design and Analysis of Algorithms Lab	L	T	P	C
Version 1.0		0	0	2	1
Pre-requisites/Exposure	Programming knowledge and fundamentals of data structures.				
Co-requisites	--				

Course Objectives

1. Learn how to analyze a problem and design the solution for the problem.
2. Design and implement efficient algorithms for a specified application.
3. Strengthen the ability to identify the suitable data structure and design strategy for the given real world problem.

Course Outcomes

Upon completion of this course the learners will be able to:

CO1. To learn the importance of designing an algorithm in an effective way by considering space and time complexity

CO2. Apply divide and conquer strategy based algorithms.

CO3. Apply Greedy and Dynamic Programming designing approach to problem solution

CO4. Application of Backtracking and brand and bound strategies over different problems

Catalog Description

The important aspects of algorithm design include creating an efficient algorithm to solve a problem in an efficient way using minimum time and space. To solve a problem, different approaches can be followed. Some of them can be efficient with respect to time consumption, whereas other approaches may be memory efficient. However, one has to keep in mind that both time consumption and memory usage cannot be optimized simultaneously.

List of Experiments

Experiment-1: Brute Force Techniques

1. Sort a given set of elements using bubble and selection sort and hence find the time required to sort elements

2. Perform linear search and find the time required to search an element.
3. Given a string called TEXT with 'n' characters and another string called PATTERN with 'm' characters (m<=n) .Write a program which implements brute force string matching to search for a given pattern in the text. If the pattern is present then find the position of first occurrences of Pattern in that Text.

Experiment-2: Divide and Conquer-I

1. Implement Binary search and linear search and determine the time required 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. Search a elements using the Binary search method and determine the time required to search the element. Repeat the experiment for different values of n, to search for the element in the list and plot a graph of the time taken versus n.
3. Sort a given set of elements using the Merge Sort algorithm to sort a given set of elements and determine the time required 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. The elements can be read from a file or can be generated using the random number generator.

Experiment-3: Divide and Conquer-II

1. Sort a given set of elements using the Quicksort method and determine the time required 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. The elements can be read from a file or can be generated using the random number generator.
2. Sort a given set of elements using the insertion Sort algorithm to sort a given set of elements and determine the time required 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. The elements can be read from a file or can be generated using the random number generator.

3. Implement Strassen's matrix multiplication and compare the complexity with normal matrix multiplication

EXPERIMENT-4: Title: GREEDY METHODS-I

1. Implement the activity-selection problem (You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time.
   Example: Consider the following 6 activities.
   Start [] = {1, 3, 0, 5, 8, 5}; finish [] = {2, 4, 6, 7, 9, 9};
   The maximum set of activities that can be executed by a single person is {0, 1, 3, 4}).

2. Consider the following scheduling problem. You are given n jobs. Job i is specified by an earliest start time si, and a processing time pi. We consider a preemptive version of the problem where a job's execution can be suspended at any time and then completed later. For example if n = 2 and the input is s1 = 2, p1 = 5 and s2 = 0, p2 = 3, then a legal preemptive schedule is one in which job 2 runs from time 0 to 2 and is then suspended. Then job 1 runs from time 2 to 7 and secondly, job 2 is completed from time 7 to 8. The goal is to output a schedule that minimizes $\Sigma Cj = 1$, where Cj is the time when job j is completed and j runs from 1 to n. In the example schedule given above, C1 =7 and C2=8.
3. To find Optimal solution for a Knapsack Problem using Greedy Method
4. Implement the file or code compression using Huffman's algorithm.

EXPERIMENT-5: Title: GREEDY METHODS-II

1. To find Optimal solution for a Knapsack Problem using Greedy Method
2. Implement the file or code compression using Huffman's algorithm.

EXPERIMENT-6: Title: Graph Algorithms-I

1. Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm

2. Find Minimum Cost Spanning Tree of a given undirected graph using prim's algorithm
3. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm

4. From a given vertex in a weighted connected graph, find shortest paths to other vertices negative weights (using Bellman-Ford algorithm).
5. Print all the nodes reachable from a given starting node in a digraph using BFS method. Check whether a given graph is connected or not using DFS method.

EXPERIMENT-7: Title: Graph Algorithms-II

1. Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm
2. Find Minimum Cost Spanning Tree of a given undirected graph using prim's algorithm
3. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm

4. From a given vertex in a weighted connected graph, find shortest paths to other vertices negative weights (using Bellman-Ford algorithm).
5. Print all the nodes reachable from a given starting node in a digraph using BFS method. Check whether a given graph is connected or not using DFS method.

EXPERIMENT-8 Title: DYNAMIC PROGRAMMING-I

1. Implement 0/1 Knapsack problem using Dynamic Programming
2. Given a rod of length n inches and an array of prices that contains prices of all pieces of size smaller than n. Determine the maximum value obtainable by cutting up the rod and selling the pieces. For example, if length of the rod is 8 and the values of different pieces are given as following, then the maximum obtainable value is 22 (by cutting in two pieces of lengths 2 and 6)

Length	1	2	3	4	5	6	7	8

Price	1	5	8	9	10	17	17	20

3. The order in which we parenthesize the product affects the number of simple arithmetic operations needed to compute the product, or the *efficiency* .(using matrix chain multiplication)

EXPERIMENT-9: Title: DYNAMIC PROGRAMMING-II

1. Implement Floyd's algorithm for the All-Pairs-Shortest-Paths problem.
2. Compute the transitive closure of a given directed graph using Warshall's algorithm
3. Given two sequences, find the length of longest subsequence present in both of them. 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".

EXPERIMENT-10: Title: SORTING IN LINEAR TIME LOWER BOUNDS

1. Write a program to perform count sort
2. Write a program to perform radix sort

EXPERIMENT-11: Title: BACKTRACKING

1. Find a subset of a given set S = {$s_1, s_2,.....,s_n$} of n positive integers whose sum is equal to a given positive integer d. For example, if S= {1, 2, 5, 6, 8} and d = 9 there are two solutions{1,2,6}and{1,8}.A suitable message is to be displayed if the given problem instance doesn't have a solution.

2. Implement N Queen's problem using Back Tracking.
3. Implementation of GRAPH COLORING
4. Implement Sudoku problem
5. Implement the Hamilton cycle problem.

EXPERIMENT-12: Title: Branch and Bound

1. implement the 1/0 knapsack problem using branch and bound
2. The Hamiltonian cycle problem is NP-complete.
3. The sum of subset problem using NP-complete.

Text Books

1. Introduction to Algorithms, Third Edition, By Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, 2009.

Reference Books

1. Introduction to the Design and Analysis of Algorithms, Dr. Anany Levitin

Continuous Evaluation

There will be continuous evaluation for all practical subjects of SoCS during the semester. The performance of a student in a Practical subject will be evaluated as per component of evaluation given below:

- Viva voce / Quiz (50%)
- Performance & Records (50%).

Lab performance and record evaluation shall be a continuous process throughout the semester. Minimum two Viva-voce and two Quizzes based on practical sessions shall be conducted during the semester.

Relationship between Program Outcomes (POs), Program Specific Outcomes (PSOs) and Course Outcomes (COs)

PO/CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO 12	PSO 1	PSO 2	PSO 3
CO1	2														2
CO2		1	2		3									2	
CO3	1	2			2										2
CO4			3	2									2		2
CO5			2	2										2	
CO6					2									2	2

1=weak 2= moderate 3=strong