17CS003 DATA STRUCTURES AND ALGORITHMS

L	T	P	C
3	-	3	5

Course description and objectives:
This course introduces students to the analysis and design of computer algorithms. The course is intended to provide the foundations of the practical implementation and usage of Algorithms and Data Structures. One objective is to ensure that the student evolves into a competent programmer capable of designing and analysing implementations of algorithms and data structures for different kinds of problems. The second objective is to expose the student to the algorithm analysis techniques, to the theory of reductions, and to the classification of problems into complexity classes like NP.

Course Outcomes:
Upon completion of this course, students will be able to do the following:

- ✓ Analyze the asymptotic performance of algorithms.
- ✓ Demonstrate a familiarity with major algorithms and data structures.
- ✓ Apply important algorithmic design paradigms and methods of analysis.
- ✓ Synthesize efficient algorithms in common engineering design situations.

SKILLS:

- ✓ Be able to Design and Analyse programming problem statements.
- ✓ Choose appropriate data structures and algorithms, understand the ADT/libraries, and use it to design algorithms for a specific problem.
- ✓ Be able to understand the necessary mathematical abstraction to solve problems.
- ✓ Be able to come up with analysis of efficiency and proofs of correctness
- ✓ To be able to comprehend and select algorithm design approaches in a problem specific manner.

Unit – I

Elementary Data Structures: Trees, binary heaps, Hashing, Balanced Search Trees - Properties and Abstract Data Types (ADT) of AVL, Red-Black and Splay Trees, Disjoint set data structure: Union-find

Unit – II

Introduction to Algorithm Analysis: Algorithm, Asymptotic Notation Recurrences: Substitution, Iteration and master method
Divide and Conquer: General method, Applications - Binary search, Merge sort, Quick sort, Strassen's Matrix multiplication

Unit – III

Greedy Method: General method, Applications - Fractional knapsack problem, Minimum cost spanning trees, Single source shortest path problem
Graph Algorithms: BFS, Applications of BFS, bipartite graphs, Depth First Search(DFS), Application Of DFS like Topological Sort, Cycle Detection, Checking Whether a Digraph is Strongly connected or not,

Unit – IV
Dynamic Programming: General method, Applications - Matrix chain multiplication, Optimal binary search trees, 0/1 knapsack problem, All pairs shortest path problem, Travelling sales person problem and design, Longest Common Subsequence.

Unit – V

Backtracking: General method, Applications - N-queen problem, Sum of subsets problem
Intractability and NP-Completeness: The class NP, Satisfiability, NP-hard and NP-complete problems, proving a problem is NP-complete. Approximation algorithms for NP-hard problems

LABORATORY EXPERIMENTS

1. Implement stack ADT and write a program that reads an infix arithmetic expression of variables, constants, operators (+, -, *, /) and converts it into the corresponding postfix form. Extend the program to handle parenthesized expression also.
2. Write a program to implement the following operations:
    a) Traversal operation on a given binary search tree with given elements.
    b) Search for a key element in the above binary search tree.
    c) Delete an element from the above binary search tree.
3. Write a program to sort a given list of elements using quick sort and merge sort
4. Implement AVL Tree ADT and write a program that interactively allows
    a) Insertion b) Deletion c) Find_min d) Find_max
5. Consider the problem of eight queens on an (8x8) chessboard. Two queens are said to attack each other if they are on the same row, column, or diagonal. Write a C++ program that implements backtracking algorithm to solve the problem i.e. place eight non-attacking queens on the board.
6. Write a C++ program to implement dynamic programming algorithm to solve all pairs shortest path problem.
7. Write a C++ program to solve fractional 0/1 knapsack problem.
8. Write a C++ program to find the strongly connected components in a digraph.
9. Write a C++ program that uses dynamic programming algorithm to solve the optimal binary search tree problem.
10. Write a C++ program to solve traveling sales person problem.
11. Write a C++ program to find optimal ordering of matrix multiplication. (Note: Use Dynamic programming method).

Activities

1. Prepare a table for each sorting algorithm showing the elapsed system time (use standard clock function) of the sort for *at least* four different non-trivial array sizes. A non-trivial array size is one where the runtime is more than just a few milliseconds.
2. Design an array based data structure for two stacks called a DualStack The two stacks should share the same array in an efficient manner. If there are MaxSize entries in the array then the IsFull function should only return true is all the entries in the array are occupied. Your operations should all be constant time. Check such a nice data structure would be possible for 3 stacks or not
3. Design an algorithm based on depth-first search to determine if a graph is bipartite and if it is not return an odd length cycle in the graph. You algorithm should use the adjacency list representation of a graph. Your algorithm should run in linear time. Hint: there are two labels for marking 1 and 2. A new vertex visited from a vertex marked 1 is marked 2 and a new vertex visited from a vertex marked 2 is marked 1.
4. Some project planning applications use a labeled acyclic directed graphs to represent the jobs and job times on a project. A vertex in the graph represents a job and its label represents the time the job will take. A directed edge from one vertex to another represents the fact the job represented by the first vertex must be completed before the job represented by the second vertex. Assume we have a directed acyclic graph G =({1,2,...,n},E) with vertices labeled by non-negative integers c_1, c_2, ... ,c_n . The label

c_i represents the time job i will take. Assume futher that every vertex is reachable by some path from vertex 1, vertex 1 has in-degree 0, vertex n is reachable by some path from every vertex, and n has out degree 0. Vertex 1 represent the beginning of the project and vertex n represent the end of the project. The length of a path from 1 to n is the sum of the labels on the vertices along the path. Design an algorithm based on the topological sort algorithm to find the length of a longest path from 1 to n in the graph. The length of the longest path represents how long the entire project will take. Sometimes a longest path is called a critical path.

5. Design a Data Structure for web server to store history of visited pages. The server must maintain data for last n days. It must show the most visited pages of the current day first and then the most visited pages of next day and so on.
6. In biological applications, we often want to compare the DNA of two (or more) different organisms. A strand of DNA consists of a string of molecules called bases, where the possible bases are adenine, guanine, cytosine, and thymine. Representing each of these bases by their initial letters, a strand of DNA can be expressed as a string over the finite set A, C, G, T. One goal of comparing two strands of DNA is to determine how "similar" the two strands are, as some measure of how closely related the two organisms are. Similarity can be and is defined in many different ways. One way to measure the similarity of strands S1 and S2 is by finding a third strand S3 in which the bases in S3 appear in each of S1 and S2; these bases must appear in the same order, but not necessarily consecutively. The longer the strand S3 we can find, the more similar S1 and S2. Compute the similarity between ACCGGTCGAGTGCGCGGAAGCCGGCCGAA and GTCGTTCGGAATGCCGTTGCTCTGTAAA using dynamic programming approach.
7. Determine which of the following problems are NP-complete and which are solvable in polynomial time. In each problem you are given an undirected graph G = (V, E), along with:
   (a) A set of nodes $L \subseteq V$, and you must find a spanning tree such that its set of leaves includes the set L.
   (b) A set of nodes $L \subseteq V$, and you must find a spanning tree such that its set of leaves is precisely the set L.
   (c) A set of nodes $L \subseteq V$, and you must find a spanning tree such that its set of leaves is included in the set L.

TEXT BOOKS:

1. Ellis Horowitz, Sartaj Sahni and Rajasekaaran "Fundamentals of Computer Algorithms", second edition, University press.
2. Sartaj Sahni, "Data Structures, Algorithms and Applications in java", University Press.

REFERENCE BOOKS:

1. T.H.Cormen, C.E.Leiserson, R.L.Rivest,and C.Stein, "Introduction to Algorithms", second edition ,PHI Pvt. Ltd.
2. Aho, Ullman and Hopcroft, "Design and Analysis of algorithms", Pearson education.
3. Richard Johnson baugh and Marcus Schaefer , "Algorithm Design: Foundations, Analysis and Internet examples, Algorithms", Pearson Education.
4. Anany Levitin, "Introduction to the Design and Analysis of Algorithms", 3rd Edition by, Addison-Wesley.
5. Jon Kleinberg and Eva Tardos , "Algorithm Design", Pearson.