Data Structures and Algorithms - II

|  |  |
| --- | --- |
| **Teaching Scheme** | **Examination Scheme** |
| Lectures: 2 Hrs / Week | Assignments / Quizzes: 40 marks |
|  | End Semester Exam: 60 marks |

Course Outcomes

Students will be able to:

1. Write neat code by following coding standards, by selecting appropriate data structure and demonstrate a working solution for a given problem.
2. Think of all possible inputs to an application and handle all possible errors properly.
3. Analyze different possible solutions to a program and select the most efficient one.
4. Write an application requiring an effort of at least 1000 lines of code to demonstrate a good working solution.
5. Demonstrate the ability to write reusable code and abstract data types, using object-based way of thinking.

**Course Contents**

Linear Data Structures: Recall - Static and Dynamic memory allocation, Arrays, Linked List, Stack, Queue. Time Complexity Analysis. Verification of programs, invariants, assertions, proof of termination.

[3 Hrs]

Trees:Basic terminology. Binary Tree: Properties of a Binary Tree, ADT Binary trees and its representations, concept of Non- Linear Data Structures, Difference between Linear and Non-Linear data structure, Binary tree traversals (recursive and non-recursive) and various operations. Binary Search Tree (BST): Properties, Insertion and deletion of nodes. Complexity Analysis of all operations.

[6 Hrs]

Priority queues and Heap: Priority Queues. Max and Min Heap. Operations on heap, Heapsort. Applications of trees. AVL Trees: Introduction, Properties, Balance Factor, Operations like insert, rotate and delete, Red Black Trees: Properties, Operations like insert, delete and rotate.

**[5 Hrs]**

Graph: Representation of graphs using adjacency matrix, adjacency list. Implementation of algorithms for traversals; implementing Kruskal's or Prim's or Single source shortest paths using Dijkstra's algorithm. Applications of graphs for problems like shortest path on a map.

[6 Hrs]

Sparse Matrices. Matrix operations like addition and multiplication. Sparse matrices concept. Different implementations of sparse matrices. Operations like addition and multiplication. Time complexity comparisons.

**[4 Hrs]**

Applications. Design of data structures, modules, functions and algorithms for applications like Huffman Coding, Implementing a ‘heap’ manager (malloc/free library), programs like ‘tree’, ‘tar’, ‘diff’, ‘grep’, ‘find’, ‘zip’, ‘unzip’, ‘nano’, ‘vim’,  large matrix operations library,  map applications, air traffic simulator etc

**[4 Hrs]**

 Text Books

“Fundamentals of Data Structures in C”, E. Horowitz, S. Sahni, S. Anderson-freed, Second Edition, 2008, University Press, ISBN 978-81-7371-605-8

“The C Programming Language”, B. Kernighan, D. Ritchie, Second Edition, 2015, Pearson Education India; ISBN 81-203-0596-5

Reference Books:

“Data Structures using C”, Y. Langsam, M. Augenstin and A. Tannenbaum, First Edition, 2002, Pearson Education Asia, ISBN 978-81-317-0229-1

“C++: The Complete Reference”, Herbert Schildt, Fourth Edition, 2002, The McGraw-Hill company, ISBN 0-07-222680-3

 “Fundamentals of Data Structures in C++”, Ellis Horowitz, S. Sahni, D. Mehta, 2nd Edition, 2008, University Press, ISBN-10: 8173716064

“An introduction to data structures with Applications”, Jean-Paul Tremblay, Paul. G. Soresan, 2nd Edition, 1984, Tata Mc-Graw Hill International Editions, ISBN-0-07-462471-7

Data Structures and Algorithms – II Laboratory

|  |  |
| --- | --- |
| **Teaching Scheme** | **Examination Scheme** |
| Laboratory: 2 Hrs / Week | Continuous evaluation: 50 Marks |
|  | Mini Project: 25 marks |
|  | End Semester Exam: 25 Marks |

Course Outcomes

Students will be able to:

1. Write neat code by following coding standards, by selecting appropriate data structure and demonstrate a working solution for a given problem.
2. Think of all possible inputs to an application and handle all possible errors properly.
3. Analyze different possible solutions to a program and select the most efficient one.
4. Write an application requiring an effort of at least 1000 lines of code to demonstrate a good working solution.
5. Demonstrate the ability to write reusable code and abstract data types, using object-based way of thinking.

Note: Students will be expected to *reuse* the ADT list, stack, queue codes they have written in ‘Data Structures and Algorithms – I Laboratory’ course in III semester, whenever required.

Suggested List of Assignments:

1. Write the following functions for a binary search tree implementation: Searches the maximum value in the tree, preorder traversal without using recursion, Search the str in the tree and returns a pointer to the node, print the binary tree so that it looks like a tree.
2. Write a function to find transpose of a sparse matrix.
3. Write code to list leaf nodes, non-leaf nodes and level of all nodes in a given binary tree.
4. Write a code for level order traversal of a binary tree with and without stack.
5. Develop C functions to insert and delete into/from a max heap under the assumption that a dynamically allocated array is used, the initial capacity of this array is 1, and array doubling is done whenever we are to insert into a max heap that is full.
6. Implement Heap sort algorithm
7. Write a graph implementation, using adjacency lists.
8. Implement DFS and BFS on a Graph.
9. Find all connected components of a Graph.
10. Implement Dijkstra’s shortest path algorithm.
11. Develop a hash table implementation in which overflows are resolved using chaining.
12. Start with an empty AVL tree and perform series of insertions like : DECEMBER, JANUARY, APRIL, MARCH, JULY, AUGUST, OCTOBER, FEBRUARY, NOVEMBER, MAY, JUNE. Display the tree.
13. **Mini-project**:Write an application of your own demonstration your skills in defining a problem, writing down the requirements carefully, designing a modular solution with clear separation of abstract data types and their use, design of proper function prototypes and division of work among functions. The application can be a unix command re-implemented (e.g. find, tar,  , zip, unzip,  nano, vim,  large matrix operations library,  map applications, huffman coding, air traffic simulator etc.), reimplementation of C library functions, memory allocator, a simple game using libraries like n-curses or SDL, games like sudoku or chess, etc.