ITE5001 Advanced Data Structures and Algorithms

L T P J C
3 0 2 0 4

Version: 1.00
Pre-requisite: Nil

Objectives:

- To learn Algorithms and Data structures as more productive computer scientist.
- To facilitate the students to develop more efficient algorithms and employ appropriate Data structures for solving a problem and to aid them in implementing the same..

Expected Outcomes:

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

- Design an efficient algorithm for a problem using a specified paradigm along with a proper data structure.
- Choose an appropriate design paradigm that solves the given problem efficiently along with appropriate data structures.
- Map real-world problems to algorithmic solutions.
- Analyze algorithms asymptotically and compute the performance analysis of algorithms with the same functionality.
- Identify the existence of problems which defy algorithmic solution

Module	Topics	L Hrs	SLO
1	**Advanced Algorithm Design :** Dynamic Programming **-** Rod Cutting, Matrix chain multiplication, Longest Common Subsequence, Greedy Algorithms – Activity selection problem, Matroids and Greedy methods	6	1
2	**Primary Data Structures :** Sorting – Quick and Heap Sort, Radix Sort, AVL trees, Graph Traversals	6	1,14
3	**Time and Space Complexity Analysis :** Asymptotic notations, conditional asymptotic notations, Amortized analysis, NP complete and NP hard, Time and Space complexity analysis by solving recurrence equations	6	1,2
4	**Optimization Data structures** : Search Trees, building Optimal search trees, Height balanced and Weight balanced trees - B –trees, Red Black Trees and Splay trees	6	1,2,14
5	**Data Structures for sets of Intervals :** Interval Trees - Segment Trees, Trees for Weighted Intervals, Higher dimensional Segment Trees , Range Counting and Semi group model	6	1,5,14
6	**Geometric and Heap Structures :** K-d trees, Orthogonal Range trees, Leftist heap, Skew heap, Binomial heap and Fibonacci heaps	6	1,14
7	**Data structures for Strings & Transformations :** Dynamic Structures, Persistent Structures, Tries, Compressed Tries, Suffix Trees and Suffix Arrays	6	1,14
8	**Expert Talk on Recent Advancement in Data Structures and Algorithms**	3	14

Total Lecture Hours # Mode: Flipped Class Room, [Lecture to be videotaped], Use of physical and computer models to lecture, Visit to Industry, Min of 2 lectures by industry experts	**45**

Text Book:

1. Peter Brass, "Advanced Data Structures", Cambridge University Press, 2008.

Reference Books:

1. Thomas H.Cormen, Charles E.Leiserson, Ronald L.Rivest, Clifford Stein, "Introduction to Algorithms: Third Edition", The MIT Press, 2014.
2. Thomas H.Cormen, "Algorithms Unlocked", The MIT Press, 2013
3. Mark Allen Weiss, "Data structures and algorithm analysis in C++", Florida International University, 4th edition, Pearson education, 2014

Lab Exercises

1. Write a program to implement a 3-stacks of size 'm' in an array of size 'n' with all the basic operations such as IsEmpty(i), Push(i), Pop(i), IsFull(i) where 'i' denotes the stack number (1,2,3), m n/3. Stacks are not overlapping each other. Leftmost stack facing the left direction and other two stacks are facing in the right direction.

2. To implement 2 overlapping queues in an array of size 'N'. There are facing in opposite direction to eachother. Give IsEmpty(i), Insert(i), Delete(i) and IsFull(i) routines for ith queue

3. To implement Stack ADT using Linked list with the basic operations as Create(), Is Empty(), Push(), Pop(), IsFull() with appropriate prototype to a functions.

4. To implement Queue ADT using Linked list with the basic functions of Create(), IsEmpty(), Insert(), Delete() and IsFull() with suitable prototype to a functions

5. To implement Quick Sort on 1D array of Student structure (contains student_name, student_roll_no, total_marks), with key as student_roll_no. And count the number of s performed.

6. To implement Binary search on 1D array of Employee structure (contains employee_name, emp_no, emp_salary), with key as emp_no. And count the number of comparison happened.

7. To implement Radix Sort on 1D array of Faculty structure (contains faculty_name, faculty_ID, subject_codes, class_names), with key as faculty_ID. And count the number of s performed.

8. To store k keys into an array of size n at the location computed using a hash function, loc = key % n, where k<=n and k takes values from [1 to m], m>n. To handle the collisions use the following collision resolution techniques, a. Linear probing b. Quadratic probing c. Random probing d. Double hashing/rehashing e. Chaining

9. Binary Search Tree to implement following operations: a. Insertion b. Deletion i. Delete node with only child ii. Delete node with both children c. Finding an element d. Finding Min element e. Finding Max element f. Left child of the given node g. Right child of the given node h. Finding the number of nodes, leaves nodes, full nodes, ancestors, descendants.

10. AVL Tree to implement the insertion operations: (For nodes as integers) .Test the program for all cases (LL, RR, RL, LR rotation)

11. Implement Make_Heap, Insertion, Find_Min, Extract_Min, Union, Decrease_Key and Delete_Key operations in Fibonacci Heap for the given data as Student structures (contains student_name, student_roll_no, total_marks), with key as student_ roll_no

12. Write a program to insert the given segment into the Interval using Segment trees.

13. Write a program to construct the k-d trees

14. Write a program to construct the Tries

Compiled By: Dr. S.S. Manivannan

Date of approval by the Academic Council : 18.03.16