Course code	Data structures and Algorithms	L	T	P	J	C
CSE2003		2	0	2	4	4
Pre-requisite	-	Syllabus version				
		v. xx.xx				

Course Objectives:

- To stress the importance of Algorithms and Data structures in becoming a more productive computer scientist.
- To appreciate and understand the Algorithms and Data structures used for solving a problem are much more important than knowing the exact code for it in some programming language.
- To provide an insight into the intrinsic nature of the problem as well as possible solution techniques, independent of programming language, programming paradigms, computer hardware or any other implementation technique.

Expected Course Outcome:

On completion of this course, student should be able to

(1) Analyze the worst-case running time of algorithms
(2) Explain the major data structures and their analyses.
(3) Explain major algorithm design paradigms and their analyses.
(4) Explain the major graph algorithms and their analyses.
(5) Compare between different data structures and algorithmic techniques for a given problem and assess the tradeoffs involved.
(6) Synthesize efficient data structures and algorithms and provide program solutions in engineering design situations.
(7) Provide algorithmic solutions to real-world problems

Student Learning Outcomes (SLO):	1,5,9

Module:1	Introduction to Data structures and Algorithms	1 hour	SLO: 1

Overview and importance of algorithms and data structures, States of algorithm development for solving a problem. Describing the problem, Identifying a suitable technique, Design of an Algorithm, Proof of Correctness of the Algorithm, Computing the time complexity of the Algorithm.

Module:2	Analysis of Algorithms	3 hours	SLO: 1

Asymptotic notations and their significance, Running time of an algorithm, Time-complexity of an algorithm, Performance analysis of an algorithm, Analysis of iterative and recursive algorithms . Master theorem (without proof)

Module:3	Data Structures	7 hours	SLO: 1,5

Importance of data structures , Arrays , Stacks , Queues, Linked list, Trees, Hashing Table, Binary Search Tree, Heaps

Module:4	Algorithm Design Paradigms	8 hours	SLO:5,9

Divide and Conquer, Brute force, Greedy, Recursive Backtracking and Dynamic Programming.

Module:5	Graph Algorithms	4 hours	SLO: 5, 9

Breadth First Search (BFS), Deapth First Search (DFS), Minimum Spanning Tree (MST), Single Source Shortest Paths.

Module:6	Computational Complexity classes	5 hours	SLO: 5, 9

Tractable and Intractable Problems, Decidable and Undecidable problems, Computational complexity Classes: P,

NP and NP complete - Cook"s Theorem (without proof),3-CNF-SAT Problem, Reduction of 3-CNF-SAT to Clique Problem, Reduction of 3-CNF-SAT to Subset sum problem.			
Module:7	**Recent Trends**	**2 hours**	**SLO: 1,5,9**
	Total Lecture hours:	**30 hours**	

Text Book(s)	
1.	Introduction to Automata Theory, Languages, and Computation (3rd Edition), John E Hopcroft, Rajeev Motwani, Jeffery D. Ullman, Pearson education, 2013.
2.	Principles of Compiler Design, Alferd V. Aho and Jeffery D. Ullman, Addison Wesley,2006.
Reference Books	
1.	Introduction to Languages and the Theory of Computation, John Martin, McGraw-Hill Higher Education,2010
2.	Modern Compiler Implementation in Java, 2nd ed., Andrew W. AppelCambrdige University Press, 2012.
Mode of Evaluation:	

List of Challenging Experiments (Indicative)	**SLO: 14,17**	
	1. Array , loops 2. Stacks and Queues 3. Searching and Sorting 4. Linked List 5. Brute force technique 6. Greedy Technique 7. Backtracking 8. Dynamic Programming 9. Tree 10. BFS and DFS 11. Minimum Spanning Tree 12. Domain Specific Algorithms	30 hours **30**
1.	Design an algorithm for the following " closet pair problem" :Given n points x_1 , x_2 ,..., x_n on the real line, find the pair of points which are closest (in the sense of distance) of all such pairs. Implement your algorithm in any programming language.	
2.	Assume that a square matrix is called a Matrix Sorted Array, only if all the entries are in an increasing order both row and column wise. The below matrix is an example of the Matrix Sorted Array. Design an efficient algorithm to convert the given square matrix into a Matrix Sorted Array . Implement your algorithm into any programming language.	

		12	13	14	15	
		23	34	67	89	
		27	45	78	92	
		29	67	86	100	
3.	Given n points in a two dimensional plane, sort the n points in increasing order of the polar angles formed by those points.					
4.	Let S be the set of binary numbers (Strings on alphabet {0,1}) whose decimal value is divisible by 3. Write a program to sort the binary numbers in non-decreasing order of their decimal values.					
	In the situation where there are multiple users or a network computer system, you probably share a printer with other users. When you request to print a file, your request is added to the print queue. When your request reaches the front of the print queue, your file is printed. This ensures that only one person at a time has access to the printer and that this access is given on a first-come, first- served basis. Design an algorithm for this scenario and implement your algorithm in any programming language.					
	Implement an effective solution for Balanced parenthesis problem					
	You are making an iPod playlist to hear the songs. Assuming that shuffle functions are not applicable, choose an appropriate data structure that will add and delete songs onto you‟re your iPod in such a way that the recently inserted song will always be the first song currently on the iPod.					
	You have n coins, all of which are gold except one coin which appears to be a gold coin, but it is fake. All gold coins are of the same weight, the fake coin weighs less than the others. You have a balance scale, you can put any number of coins on each side of the scale at one time and it will tell you if the two sides weight the same, or which side is lighter if they don‟t weight the same. The problem is to identify the fake coin. Design al algorithm to find the fake coin in the given n coins.					
	Implement the following operations on the stack using linked list data structure. In addition to the usual operations of the stack and the linked list, your implementation should handle two more operations. • Split(p(i1,i2,i3…ip),q(j1,j2,…jq)) in the stack, where the stack of length n will be split in two stacks, each of length p and q such that p+q=n. Here the index i_k +1 need not be equal to i_{k+1}, where $1 \le k \le p$ and the index j_i+1 need not be equal to j_{i+1}, where $1 \le i \le q$. • Given two stacks p and q , Combine((p(i1,i2,i3…ip),q(j1,j2,…jq)) into one stack of length p+q=n. The new stack should contain the elements of the stacks p and q in any combination.					
	Consider the equation APPLE + LEMON = BANANA. Assume that each letter actually represents a digit from 0 to 9. Some conditions are imposed. The leftmost letter can't be zero in any word. There must be a one-to-one mapping between letters and digits. In other words, if you choose the digit 5 for the letter E, then all of the E's in the equation must be 5 and no other letter can be a 5. No digit can be repeated. Write a program to solve the above equation.					
	A village has a problem of frequent thefts. The sarpanch (elected head of the gram panchayat) decides to hire security guards to give protection to all streets of the village. Impement an efficient algorithm such that all the streets are protected with minimum number of security guards. Implement the problem using two different design techniques					