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					

	and compare their efficiencies.	
	Let P_1, P_2 , P_3 ,..., P_n be points on the two dimensional plane. Implement an efficient algorithm to find the longest pair of points (the distance between the points is maximum) . The distance the between the two points P_i and P_j is defined as $d(P_i, P_j) = \lvert x_i - x_j \rvert + \lvert y_i - y_j \rvert$ where operator "$\lvert\ \rvert$" is the mod function.	
	You have a standard 8x8 chessboard (if your are not familiar with game of chess, please get to know), empty but for a single knight on some square. Your task is to generate a series of legal knight moves that result in the knight visiting every square on the chessboard exactly once. In addition, the knight must end on a square that is one knight's move from the beginning square. The output of your program should indicate the order in which the knight visits the squares, starting with the initial position. Generalize your program for an n x n board where n >8.	
	The Binomial coefficients are defined as follows : $n_{C_k} = \begin{cases} & = \\ & - \quad + \quad - \end{cases}$ Implement an efficient algorithm to evaluate ${}^nC_k$. Let $M_1 \times M_2 \times M_3 \times \ldots \times M_n$ be a chain of matrix products. This chain may be evaluated in several different ways. The cost of any computation of $M_1 \times M_2 \times M_3 \times \ldots \times M_n$ is the number of multiplications used. Implement an efficient algorithm to compute the minimum as well as maximum cost to evaluate $M_1 \times M_2 \times M_3 \times \ldots \times M_n$. A "Matrix Sequence" $M_k = \begin{pmatrix} F_{k-1} \\ F_k \end{pmatrix}, where\ \ k \geq 2, F_k = F_{k-1} + F_{k-2},\ F_1 = 1\ F_0 = 0$ Implement an efficient algorithm to compute the number F_n in logarithmic time complexity.	
	Design a boolean circuit, which has n components (like OR, AND, XOR, NOT) connected by wire, in any order . Implement an efficient algorithm to compute the maximum and the minimum length of the wire(depth of the circuit) reqired for fabricating the boolean circuit for a given Boolean function.	
	Consider the problem of barricading n sleeping tigers by a fence of shortest length .Forest officials have tranquilized each tiger. Suggest an algorithm for the purpose. You are allowed to assume any information required for your algorithm. Implement your algorithm in any programming language.	
	Let (x_1,y_1), $(x_2, y_2)\ldots(x_n, y_n)$ be the coordinates of n villages located along a circular path. Government wants to open a post office to serve all these villages. Implement your algorithm to identify the location of the post office such that the post office is at an equal distance from all the villages.	
	Propose a real world of your choice and implement an algorithmic solution for that problem. Using any two different techniques along with two different data structures. Analyze the performace of the algorithms involved in the above process.	
	Implement the Quick-Sort Algorithm by choosing pivot element in five distinct measures and analyze the performance of all five algorithms. Based on your analysis give the best	

	measure for choosing the pivot element.	
	Sort the given n numbers in such way that every number is followed by its factors in increasing order .	
Project # Generally a team project [5 members] Projects may be given as group projects Review Report on the state of art of algorithms in a specified domain. For eg, A review report on the sorting algorithms To understand the search engines and compute its time-complexity To understand and implement any complex algorithm reported in a research article To analyse the best design technique deployed for the development of algorithms in a specified domain To develop an algorithm for a specific problem and implement. To give a programming solution for the problem which has defied theoretical proofs like Four Colour Map theorem. To reduce a real life problem into a computational model and then analyze.		60 (Non Contact Hours) SLO 1, 5

Text Books

1. Thomas H. Cormen, C.E. Leiserson, R L.Rivest and C. Stein, Introduction to Algorithms , Paper Back 2010, Third edition, MIT Press, 2010

Reference Books

1. Sanjoy Dasgupta, C.Papadimitriou and U.Vazirani , Algorithms , Tata McGraw-Hill, 2006.
2. A. V. Aho, J.E. Hopcroft and J. D. Ullman, Data Structures and Algorithms ,Pearson India, Ist Edition, 2002.
3. A. V. Aho, J.E. Hopcroft and J. D. Ullman, Design and Analysis of Computer Algorithms ,Pearson,1st edition, 2011.
4. Sara Baase , Allen Van Gelder, Computer Algorithms, Introduction to Design and Analysis, 3rd edition, Wesley Longman Publishing, 2000.
5. Mark Allen Weiss, Data Structures and Algorithm Analysis in C, 2nd edition, Pearson Education, 2007

Mode of evaluation:			
Recommended by Board of Studies	DD-MM-YYYY		
Approved by Academic Council	No. xx	Date	DD-MM-YYYY

CO-PO mapping and CO-SLO mapping

	PO1/SLO1	PO3/SLO5	PO6/SLO9
CO1	*	*	
CO2	*	*	
CO3	*	*	
CO4	*	*	
CO5	*	*	*
CO6	*	*	*
CO7	*	*	*

2. Knowledge Areas that contain topics and learning outcomes covered in the course

Knowledge Area	**Total Hours of Coverage [Theory+Practical]**
CE : ALG0 : History and overview	1+0
CS: AL : Basic Analysis CE: ALG1: Basic Algorithmic Analysis CS: DS: Basics of Computing CS: SDF : Algorithms and Design	3+ 1
CS: AL : Algorithmci Strategies CE: ALG2: Algorithmic Strategies CS: DS: Proof Techniques CS: SDF : Algorithms and Design	8+9
CS: AL: Fundamental Data Structures & Algorithms CE: ALG3 : Computing Algorithm CS: DS: Proof Techniques CS: DS: Graphs and Trees CS: SDF: Fundamental Data Structures	12 +10
CS : AL : Advanced Computational Complexity CE: ALG5 : Algorithmic Complexity	6 +0
CS: AL; Advanced Data Structures and Analysis	0+ 6

2.1 Body of Knowledge coverage

KA	Knowledge Unit	Topics Covered	Hours
CE : ALG0	**Introduction to Data structures and Algorithms**	Overview and the importance of algorithm and data structures , Stages of algorithm development for solving a problem.	1
CS: AL : CS:DS CE: ALG1:	**Analysis of Algorithms**	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 and Master theorem (without proof)	3
CS: AL CS: DS CS:SD F	**Data Structures**	Importance of data structures , Arrays , Stacks , Queues, Linked list ,Trees , Hashing table , Binary Search Tree , Heaps .	8
CS:AL	**Algorithm Design Paradigms**	Divide and Conquer, Brute force, Greedy, Recursive Backtracking, and Dynamic programming .	8
CS:DS CS:AL CE:AL G2	**Graph Algorithms**	BFS, DFS , MST , Single Source Shortest Path	4

CS:AL CE:AL G5	**Computational Complexity classes**	Tractable and intractable Problems, Decidable and undecidable problems, Computational complexity Classes: P, NP and NP complete class- 3SAT Problem, Clique Problem, Vertex cover problem) .	6

3. Where does the course fit in the curriculum?

This course is a program core course

Can be offered from the second semester onwards

Knowledge of any one programming language is preferred

4. What is covered in the course?

The course covers

- Basic techniques used to analyze the problems (Asymptotic performance of analysis)
- Basic techniques used to design an algorithm (including divide and conquer, Bruteforce, Recursive Backtracking , Greedy and Dynamic Programming)
- Choosing appropriate data structures for solving complex problem
- Standard classical algorithm (including sorting, searching, string matching, Geometric algorithms and Graph algorithms)

The main objective of this course is to equip the students to provide an algorithmic solution for the real world problems.

5. What is the format of the course?

The course includes two lectures per week along with 100 minutes of practicals (Lab session) per week . Total number of lectures : 30 hrs

Total number of Practicals : 30 hrs

6. How are students assessed?

Assessment includes two Continuous assessment tests, digital assignments, lab sessions, project and Final assessment test.

In the lab, students are expected to practice / implement the algorithm a set of challenging problems posed to them .

To ensure that the graduates can successfully apply the knowledge they have gained, in this course, all students should do one project related to the development of algorithm with appropriate data structures worth 60 hrs.

Assignments :
There will be 5 challenging assignments which requires 25 hrs of student's time.

7. Session wise plan

3 levels of depth: Familiarity, Usage, and Assessment

Familiarity: know what it means (eg : Different Loops)

Usage: can apply concept (e.g., write the code to use loops)

Assessment: can compare/contrast/select appropriate method/strategy for different situations (eg: when should for loop, while loop, etc)

Class hour and Lab hour describes that the given topic can be taken in the class room and lab respectively

Sl. No	Topic Covered	Class Hour	Lab Hour	levels of mastery	Reference Book	Remarks
1.	**Introduction to Data structures and Algorithms**	1		Familiarity	T1	
2.	**Analysis of Algorithms** Asymptotic notations and their significance and Running time of an algorithm	1		Familiarity	T1	
3	Time-complexity of an algorithm, Performance	1				

	Analysis of an algorithm,			Usage	T1	
4	Analysis of iterative and recursive algorithms and Master theorem (without proof)	1		Familiarity Usage	T1	
5	**Data Structures** Importance of data structures , Arrays ,	1	2	Familiarity Usage	T1	
6	Stacks and Queues.	1	2	Familiarity Usage	T1	
7	Linked list	2	4	Familiarity Usage	T1	
8	Trees, Binary Search Tree	1	2	Familiarity Usage	T1	
9	Hashing table	1				
10	Heaps	1		Familiarity Usage	T1	
11	**Algorithm Design Paradigms** Brute force	1	2	Familiarity Usage	T1	
12	Divide and Conquer,	1	2	Usage		
13	Greedy,	2	2	Familiarity Usage	T1	
14	Recursive Backtracking,	2	2	Familiarity Usage	T1	
15	Dynamic programming .	2	2	Familiarity Usage	T1	
16	**Graph Algorithms** BFS, DFS ,	2	2	Familiarity Usage	T1	
17	MST , Single Source Shortest Path	2	2	Familiarity Usage		
18	**Computational Complexity classes** Tractable and intractable Problems,	1		Familiarity	T1	

19	Decidable and undecidable problems,	1		Familiarity	T1	
20	Computational complexity Classes: P, NP and NP complete class.	1		Familiarity	T1	
21	3SAT Problem, Cook's theorem, Reduction of 3 CNF – SAT to Clique Problem, Reduction of 3 CNF –SAT to Subsetsum problem	2		Familiarity	T1	
22	Algorithms related to Search Engines .	2		Familarity		
23	Domain Specific Algorithms (String Matching and Computational Geometry)		6	Usage Assessment	T1	

Course code	**Network and Communication**	**L**	**T**	**P**	**J**	**C**
CSE1004		**3**	**0**	**2**	**0**	**4**
Pre-requisite	**-**	**Syllabus version**				
		v. xx.xx				

Course Objectives:

- Build an understanding of the fundamental concepts of computer networking, protocols, architectures and applications.
- Gain expertise in design, implement and analyze performance perspective of ISO-OSI layered Architecture.
- Deal with the major issues of the layers of the model.
- Implement new ideas in Networking through semester long projects
- Take advanced courses in Networking.