Algorithms Analysis and Design-1
Session: August'2024 - December'2024

1. Course Number and Name:

CSE 2631, Algorithms Analysis and Design1

2. Credits and Course Format:

4 Credits;
3 Classes/Week, 1 hr/Class;
1 Lab/Week, 2 hrs/Lab

3. Target Students:

Programme: B.Tech. (3rd Semester)
Branch: CSE, CS&IT, CSE (Data Science), CSE (AI&ML), CSE (IoT), CSE (Cyber Security)

4. Text Book and References:

Text book

1. (T1) Algorithm Design by Jon Kleinberg and Eva Tardos, Pearson Publication
2. (T2) Problem Solving in Data Structures & Algorithms Using Java by Hemant Jain

Reference book

1. (R1) The Algorithm Design Manual by Steven Skiena, Springer Publication
2. (R2) Introduction to Algorithms by CLRS, PHI Publication

5. Specific Course Information:

a. Course Description:

The course topics will include concepts of algorithm complexity, and various algorithmic design patterns like greedy approach, divide-and-conquer, and dynamic programming. Course will also cover major algorithms and data structures for searching and sorting, graphs, and some optimization techniques.

b. Prerequisites and/or Co-requisites:

Prerequisite: CSE 1001 (ICP), CSE 2001 (DSA), CSE 1004 (IGT)
Co-requisite: CSE 2141(CSW1)

6. Course Learning Outcomes (CLOs):

By the end of course through lectures, readings, home-works, lab assignments and exams, students will be able to demonstrate the abilities:

(1) to apply **knowledge of computing and mathematics** to algorithm design;

(i) to understand computational tractability condidering polynomial time as a definition of efficiency of an algorithm;
(ii) to analyze worst-case **running times of algorithms (both recursive and iterative)** using asymptotic analysis;

(2) to understand various types and aspects of basic data structures (array, linked list, stack, queue, binary tree) and **advanced data structures like priority queue (implementation using heap)**.
(3) to explain the major **graph algorithms** and their analyses. Employ graphs to model engineering problems, when appropriate.
(4) to describe the **greedy paradigm** and explain when an algorithmic design situation calls for it. Recite algorithms that employ this paradigm. Synthesize greedy algorithms. Derive and describe the performance of greedy algorithms.
(5) to describe the **divide-and-conquer paradigm** and explain when an algorithmic design situation calls for it. Recite algorithms that employ this paradigm. Synthesize divide-and-conquer algorithms. Derive and solve recurrences describing the performance of divide-and-conquer algorithms.
(6) to describe the **dynamic programming paradigm** and explain when an algorithmic design situation calls for it. Recite algorithms that employ this paradigm. Synthesize dynamic programming algorithms. Derive and solve recurrences describing the performance of dynamic programming algorithms.

7. B.TECH - PROGRAM OUTCOMES (POs)

PO-1	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems**(Engineering knowledge).**
PO-2	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences **(Problem analysis)**.
PO-3	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations **(Design/development of solutions)**.
PO-4	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions **(Conduct investigations of complex problems)**.
PO-5	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations **(Modern tool usage)**.
PO-6	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice **(The engineer and society)**.
PO-7	Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development **(Environment and sustainability)**.
PO-8	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice **(Ethics)**.

PO-9	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings **(Individual and team work)**.
PO-10	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions **(Communication)**.
PO-11	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments **(Project management and finance)**.
PO-12	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change **(Life-long learning)**.

8. Brief List of Topics to Be Covered: (L: Lecture, P: Practical)

- Introduction to algorithm design
- Basics of algorithm analysis
- Basic Concepts of Data Structures
- Graphs and related algorithms
- Greedy approach
- Divide-and-conquer
- Dynamic Programming

Contact Hour	Topics To Be Covered	Remarks(if any)	CO	PO
Week # 1:				
L 01	Introduction to the course/subject: Program Outcomes; Course Outcomes; Lesson plan; Teaching methodology; Evaluation strategy etc.	Course Overview with OBE awareness		
L 02	**Introduction to Algorithm Design:** Importance of problem solving using algorithms; Characteristic features of an algorithm(input, output, finiteness, definiteness, effectiveness, correctness, efficiency);			
L 03	**Introduction to Algorithm Design:** Expressing algorithms (pseudocode); Basic aspects of algorithms (correctness, design and analysis)			
P 01	*Approach to solve algorithm design problems* *1. Sum of n numbers* *2. Finding maximum and minimum* *3. Rotating array by k positions*	*To be referred from T2*		
Week # 2:				
L 04	**Computational tractability:** Polynomial time as a definition efficiency of an algorithm; Worst case Running times and Brute-Force Search	To be referred	CO1	

L 05	**Asymptotic order of growth** (Big-Oh, Big-Omega, Big-Theta) **[Page: (35-42)-(Section-2.2(Definition of asymptotic notations, 2.1 upto 2.9) and Class Notes]**	from T1		
L 06	**Asymptotic order of growth** (Big-Oh, Big-Omega, Big-Theta) **[Page: (35-42)-(Section-2.2(Definition of asymptotic notations, 2.1 upto 2.9) and Class Notes]**			
P 02	*Abstract data type (Array) – iterative implementation* *4. Finding the largest sum contiguous subarray* *5. Smallest positive missing number* *6. Maximum minimum array (I/p: 1 2 3 4 5, O/p: 5 1 4 2 3)* *7. Factorial of a number* *8. Generating nth fibonacci number*	*To be referred from T2*		
Week # 3:				
L 07	**Recurrences** (Iterative,Recursive Tree method, Substitution and Master method) **[Class Notes]**	To be referred from R2 and T1	CO1	
L 08	**Recurrences** (contd..) **[Class Notes]**			
L 09	**Priority Queue Implementation using Heap data structure** **[Section-2.5, Page: (57-65)] and Class Notes]**			
P 03	*Abstract data type (Array) – recursive implementation* *1. Sum of n numbers* *2. Finding maximum and minimum* *3. Factorial of a number* *4. Generating nth fibonacci number* *5. Find ing the GCD* *6. Conversion from decimal number to hexadecimal equivalent number* *7. Computing nth power of a number* *8. Smallest positive missing number*	*To be referred from T2*		
Week # 4:				
L 10	**Priority Queue Implementation using Heap data structure** **[Section-2.5, Page: (57-65)] and Class Notes]**	To be referred from T1	CO2 , CO3	
L 11	**Graph:** Basic definitions, applications and representations **[basic definition section-3.1 and Class Notes]**			
L 12	**Graph:** Basic definitions, applications and representations (contd..)			
P 04	*Sorting (Bubble sort, Insertion sort, Selection sort, etc..)* *1. Bubble sort* *2. Insertion sort* *3. Selection sort*	*To be referred from T2*		

Week # 5:				
L 13	**Graph:** Graph connectivity and graph traversal (BFS, DFS) **[section-3.2(3.3, 3.4, 3.5 no proof required), Implementing BFS (page:90-91 , time complexity) and Class Notes]**	To be referred from T1	CO3	
L 14	**Graph:** Graph connectivity and graph traversal (BFS, DFS) **[section-3.2(3.6, 3.7, 3.8 no proof required), Implementing DFS(page:92-93, time complexity) and Class Notes]**			
L 15	**Graph:** Testing bipartiteness – an application of BFS **[section-3.4(3.14, 3.15), no proof required and Class Notes]**		CO3	
P 05	*Sorting* *1. Array reduction* *2. Merging two sorted arrays* *3. check reverse*	*To be referred from T2*		
Week # 6:				
L 16	**Graph:** Connectivity in directed graph; Directed-Acyclic-Graph and Topological ordering. **[section-3.6(no proof required 3.18, 3.19, 3.20, algorithm (page:102-103)) and Class Notes]**	To be referred from T1	CO3	
L 17	**Graph:** Connectivity in directed graph; Directed-Acyclic-Graph and Topological ordering **[section-3.6(no proof required 3.18, 3.19, 3.20, algorithm (page:102-103)) and Class Notes]**			
L 18	**Graph:** MST using Kruskal's algorithm—the union-find data structure **[Class Notes]**			
P 06	*Searching* *1. Linear search without recursion* *2. Linear search using recursion* *3. Binary search without recursion* *4. Binary search using recursion*	*To be referred from T2*		
Week # 7:				
L 19	**Graph:** MST using Kruskal's algorithm—the union-find data structure (contd..) **[Class Notes]**	To be referred from T1 and R2	CO3 , CO4	
L 20	**Graph:** MST using Prim's algorithm **[Class Notes]**			
L 21	**Graph:** Shortest path problem (Dijkstra' algorithm) **[Class Notes]**			
P 07	*Searching (Linear Search, Binary Search, Hashing and Symbol tables)* *1. Finding first repeated elements in an array* *2. Print duplicates in a list*	*To be referred from T2*		

	3. Find the missing number in an array *4. Given an array of integers, find the element pair with minimum/maximum difference* *5. Given a list n numbers, find the element which appears maximum number of times.*			
Week # 8:				
L 22	**Greedy Method:** Interval Scheduling with proof of optimality using the Greedy Algorithm Stays Ahead **[Page: 158-163 (Section-4.1(4.1, 4.2, 4.3) with proof)]**	To be referred from T1(Chapter 4)	CO4	
L 23	**Greedy Method:** Interval Scheduling with proof of optimality using the Greedy Algorithm Stays Ahead **[Page: 158-163 (Section-4.1(4.1, 4.2, 4.3) with proof)]**			
L 24	**Greedy Method:** Scheduling to Minimize Lateness with proof of optimality using An Exchange Argument **[Page: 167-173 (Section-4.2(4.8, 4.9, 4.10) with proof)]**			
P 08	*Linked list* *1. Create* *2. Insertion (at any position including start and end)* *3. Delete (at any position including start and end)* *4. Traversal* *5. Reverse*	*To be referred from T2*		
Week # 9:				
L 25	**Greedy Method:** Optimal Caching: A More Complex Exchange Argument (no discussion on proof of optimality) **[Page: 173-176, Define the problem, Design the problem, Explain with an Example]**	To be referred from T1(Chapter 4)	CO4	
L 26	**Greedy Method:** Huffman Codes and Data Compression (no discussion on proof of optimality) **[page: 203-217 , Don't require proof of any lemma in the section-4.8]**			
L 27	**Greedy Method:** Huffman Codes and Data Compression (no discussion on proof of optimality) contd.. **[page: 203-217 , Don't require proof of any lemma in the section-4.8]**			
P 09	*Java collections*	*To be referred from T2*		

Week # 10:				
L 28	**Divide and Conquer:** Control abstraction	To be referred from T1	CO5	
L 29	**Divide and Conquer:** Merge sort **[Page: 116-119 (Section-5.1, Algorithm, Explain With an Example, Recurrence of Time complexity, Time Complexity)]**			
L 30	**Divide and Conquer:** Counting inversions **[Page: 127--131 , Section-5.3, Application of Merge Sort]**			
P 10	*Stack*	*To be referred from T2*		
Week # 11:				
L 31	**Divide and Conquer:** Quick sort **[Class Notes]**	To be referred from T1 and R2	CO5	
L 32	**Divide and Conquer:** Quick sort **[Class Notes]**			
L 33	**Divide and Conquer:** Fast integer multiplication (Karatsuba algorithm) **[Page: 137-140 (Section-5.5, Algorithm, Explain With an Example, Recurrence of Time complexity, Time Complexity)]**			
P 11	*Queue*	*To be referred from T2*		
Week # 12:				
L 34	**Dynamic Programming:** Principles of Dynamic Programming (Memoization or Iteration over Subproblems)	To be referred from T1	CO6	
L 35	**Dynamic Programming:** Weighted Interval Scheduling **[Page: 252--252 , Section-6.1(6.1, 6.2, 6.4) proof not required), Solve exapmles]**			
L 36	**Dynamic Programming:** Subset Sums and Knapsacks **[Page: 266--271 , Section-6.4 (6.8, 6.9, 6.10) proof not required), Solve exapmles]**			

9. Evaluation Strategy:

(Theory+Lab.) Assignments:	20%
Attendance:	5%
Mid-semester:	15%
End-semester(Lab. Test/Quiz):	15%
End-semester(Theory):	45%