Department of Computer Science and Engineering

Details of Lesson Plan

S.No.	Particulars	Details
1.	Course Name	Design and Analysis of Algorithms
2.	Course Code	COM-401
3.	Academic Year	2024-25
4.	Semester	4th
5.	Number of Lesson plans	46
6.	Faculty Assigned	Saurabh Sharma

Faculty Signature

Lesson Plan No. 1	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Design and Analysis of Algorithms: Introduction to the Course**
Objectives	At the end of the lesson the student shall be able to: a. Understand the importance and application of algorithms in computer science. b. Recognize the basic concepts and terminologies related to algorithms. c. Identify various career opportunities that involve algorithm design and analysis. d. Describe the structure of the course and the expected outcomes.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) *Explaining the Journey of an Algorithm to students in daily life with an example and then ask following 2 simple questions:* ✓ Can you name an algorithm that you use in your daily life? ✓ Why do you think algorithms are important in technology? 2. **Development** (30 minutes) a. Real-Life Applications of Algorithms • Search engines (e.g., Google Search) • Social media feeds (e.g., Facebook, Instagram) • Navigation apps (e.g., Google Maps) b. Why Study Algorithms? • Role of Algorithms in Computer Science ✓ Backbone of programming and software development. ✓ Enables efficient data processing and problem-solving. • Benefits ✓ Improves performance and efficiency ✓ Helps in optimizing resources ✓ Essential for innovation and technological advancement c. Career & Job Perspective: Career Roles • Algorithm Engineer • Data Scientist • Software Developer d. Industry Demand • High demand for skills in algorithm design and analysis e. Certifications and Courses Coursera Link: **https://www.coursera.org/programs/test-program-93zov/learn/analysis-of-algorithms?source=search**

	f. Course Overview: Key Units and Topics ✓ Time and Space Complexity ✓ Sorting Algorithms ✓ Greedy Approach & Dynamic Programming ✓ Graph Algorithms & Backtracking g. Explain the Learning Outcomes (CO1 to CO5) of the Course. 3. **Short Animation Video** (5 minutes) **Video Link:** https://www.youtube.com/watch?v=6hfOvs8pY1k&t=14s 4. **Interactive Session** (5 minutes) • Future Trends in Algorithm Design : Emerging Fields ✓ Machine Learning ✓ Artificial Intelligence ✓ Quantum Computing • Impact : Innovations driven by new algorithms
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 4th Edition. **b. Fundamentals of Computer Algorithms** by Ellis Horowitz, Sartaj Sahni, Universities Press, 2nd Edition. 3. **Next Lecture Topic : "Mathematical Preliminaries"**
Evaluation	Reflective Questions (What, Why, Who?). Allow students to answer and discuss. ✓ Can you explain why algorithms are fundamental to computer science? ✓ What are the benefits of using algorithms in technology? ✓ Describe a career role that involves working with algorithms. Spend 5 minutes to evaluate student assimilation of the lesson contents

End the lecture with following Note:

In this course, you'll get on an exciting journey through the world of algorithms. You'll learn how to design, analyze, and implement algorithms to solve real-world problems. From sorting and searching to dynamic programming and graph algorithms, you'll gain the skills that are the backbone of Computer Science and Engineering.

Example Elaborated: The Journey of an Algorithm

Imagine this:

You wake up in the morning and reach for your smartphone. The moment you unlock it, you start interacting with a world that runs on algorithms. Let me take you through a day where algorithms play a silent yet pivotal role in making your life easier and more efficient.

1. **Morning Routine**
   - **Weather Forecast:** You check the weather app to see if you need an umbrella. The app uses complex algorithms to predict the weather by analyzing vast amounts of meteorological data.
   - **Music Playlist:** You start your favorite playlist on Spotify. The recommendation engine uses algorithms to analyze your past listening habits and suggests songs you might like.
2. **Commute**
   - **Navigation:** As you drive to college, you use Google Maps. The app employs algorithms to provide real-time traffic updates and the shortest route to your destination, saving you time and fuel.
   - **Ride-Sharing:** If you take an Uber or any other service, algorithms match you with the nearest driver, calculate the optimal route, and even predict the fare.
3. **At College**
   - **Class Schedule:** Your college app reminds you of today's classes and any upcoming assignments. Algorithms help in scheduling and notifying you of important dates.
   - **Research and Learning:** When you search for information online, search engines use algorithms to rank pages and show the most relevant results.
4. **After College**
   - **Social Media:** You browse through Facebook, Instagram, or Twitter. Algorithms curate your feed, showing you posts and ads tailored to your interests based on your past interactions.
   - **Online Shopping:** You decide to buy a book on Amazon. Recommendation algorithms suggest books you might like based on your browsing history and purchases.
5. **Evening Relaxation**
   - **Streaming Services:** Watching a show on Netflix? The recommendation system uses algorithms to analyze your viewing history and preferences to suggest new shows or movies you might enjoy.
   - **Fitness Tracking:** If you go for a run, your fitness app tracks your progress using algorithms to analyze your speed, distance, and calories burned.

Lesson Plan No. 1.1	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Design and Analysis of Algorithms: Mathematical Preliminaries**
Objectives	At the end of the lesson the student shall be able to: a. Explain significance of Preliminary Terms b. Describe Mathematical Model of a Computer c. Understanding the concept of Demonstrating Incorrectness. d. Reasoning about Correctness of an algorithm.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) **Start by giving example of Online Shopping and Delivery Optimization :** Imagine you are shopping online during a big sale event. Thousands of orders are being placed every minute. The system needs to process these orders efficiently to ensure timely deliveries. This involves several steps: verifying payment, checking inventory, selecting the nearest warehouse, packing items, and choosing the fastest delivery route. All these steps need to be optimized to handle the high volume of orders and ensure customer satisfaction. **Engagement Questions:** • How do you think online shopping platforms manage to process thousands of orders efficiently during a big sale? • What might happen if the algorithm managing the order processing and delivery route optimization is incorrect? 2. **Development** (30 minutes) a. Understand Objectives of Learning Algorithms - Explain the tools to distinguish correct algorithms from incorrect ones. - Elaborate why Correct algorithms require careful exposition, and efforts to show both correctness and not incorrectnes. b. Describe the Mathematical Model of a Computer - Explain the significance of the RAM Model and how It works. - Working Principles of EM Model. - PRAM Model working criteria along with advantages and application areas. c. Expressing Algorithms - Reasoning about why an algorithm is impossible without a careful description of the sequence of steps to be performed.

	- Listing the most common forms of algorithmic notation. d. We use computers every day, but how often do we stop and think, "How do they do what they do?" **Short Animation Video: (3 minutes)** **https://www.youtube.com/watch?v=kM9ASKAni_s** **e.** Summarize the Objectives of Learning Algorithms and Working on Various Mathematical Models of a Computer.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. 3. **Next Lecture Topic : Time Complexity of an Algorithm**
Evaluation	Reflective Questions (What, Why, Who?). Allow students to answer and discuss. ✓ What is the significance of a mathematical model like the RAM or PRAM in designing and analyzing algorithms? ✓ Why is it important to prove the correctness of an algorithm before its implementation? ✓ Who would be responsible for ensuring that an algorithm is both correct and efficient in a software development team? ✓ What challenges might you face when trying to express an algorithm using a formal mathematical model? Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.2	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Time complexity**
Objectives	At the end of the lesson, the student shall be able to: a. Analyze the time complexity of common sorting algorithms. b. Evaluate the efficiency of an algorithm in terms of its time complexity. c. Implement algorithms in a programming language and measure their time complexity experimentally. d. Understanding the time-space trade-off involving a problem. e. Solve examples on finding Time Complexity.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) **Start by giving example of Traffic Navigation Systems** Imagine you're using a GPS navigation app like Google Maps during rush hour. The app needs to find the fastest route to your destination by considering multiple factors: current traffic conditions, road closures, and the distance of alternative routes. The app must quickly evaluate various routes to give you the best option, saving you time. **Engagement Questions:** • How do you think the GPS app determines the fastest route among so many possibilities? • What might happen if the algorithm used by the GPS app is not efficient in analyzing the routes? 2. **Development** (30 minutes) a. Understand Objectives of Time Complexity of an algorithm - How Time complexity is profoundly related to the input size. - Explain the thought process behind innovation and technology used to make life easier for people by providing ways to solve problems they may encounter. - Elaborate why it is always better to select the most efficient algorithm when a simple problem can be solved with different methods. b. Describing what it Takes to Create a Good Algorithm? c. What factors contribute to the creation of a good algorithm? d. Discuss the importance of time complexity in the long-term effectiveness of an algorithm.

	Importance of Efficiency: • Explain why efficiency is critical, particularly in applications where time and resources are limited, such as real-time systems, mobile applications, and large-scale data processing. • Discuss how even small differences in time complexity can lead to significant performance differences, especially for large datasets. e. Calculating Time Complexity in Analysis of Algorithm **Video Link:** https://www.youtube.com/watch?v=KXAbAa1mieU f. Summarize the Objectives of Time complexity.
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. 3. **Next Lecture Topic : Space complexity**
Evaluation	Reflective Questions (What, Why, Who?). Allow students to answer and discuss. ✓ What is the significance of analyzing the time complexity of an algorithm? ✓ Why is it essential to consider time-space trade-offs in algorithm design? ✓ Who should be involved in ensuring that an algorithm is optimized for both time and space complexity? Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.3	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Space complexity**
Objectives	At the end of the lesson the student shall be able to: a. Define space complexity and understand its importance in the context of algorithm design and analysis. b. evaluate how different data structures (e.g., arrays, linked lists, trees, graphs) impact the space complexity of algorithms. c. Analyze the space complexity of real-world algorithms and applications, such as those used in databases, networking, and machine learning. d. Identify the space complexity of basic algorithms, including sorting algorithms (e.g., bubble sort, merge sort) and searching algorithms (e.g., binary search). e. Solve examples of finding Space Complexity.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: What is the difference between an algorithm and a program? Define the term pseudocode. - Focus on the issues of Running Time of an Algorithm. - Have a discussion on the memory requirement of an algorithm. - Introducing Time- Space trade-off in an algorithm. 2. **Development** (30 minutes) a. Describe Space Complexity of an algorithm - Explain the significance of Auxiliary space and Input space. - Illustrating why the factor of time is usually more important than that of space. b. Describing what it Takes to Create a Good Algorithm? c. Explaining factors that play a significant role in the long-term usage of an algorithm. d. Space Complexity of an Algorithm https://www.youtube.com/watch?v=yOb0BL-84h8 3. **Exercise** (10 minutes) – - Have a discussion to summarize the lecture - Ask Questions Related to Topics

Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a.** **Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b.** **Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the Significance of Space Complexity of an algorithm. **Activity References**: **a.** https://towardsdatascience.com/space-and-time-complexity-in-computer-algorithms-a7fffe9e4683 **b.** https://www.hackerearth.com/practice/basic-programming/complexity-analysis/time-and-space-complexity/tutorial/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Google form Quiz on the topic Space Complexity of an algorithm. **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.4	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Worst-case analysis**
Objectives	At the end of the lesson, the student shall be able to: a. Understand how the worst case of an algorithm depends on the size of the user input value. b. Calculate the upper limit of the execution time of an algorithm. c. Understanding the concept of Worst-case analysis with the help of an example. d. Describe the concept of best-case analysis with the help of an example. e. Elaborate the concept of worst-case analysis with the help of an example.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: How do you determine the input that will produce the worst-case scenario for a given algorithm? - What is the worst-case time complexity of the bubble sort algorithm, and how is it derived? - Discuss why in real life, most of the time we do the worst-case analysis of an algorithm. - Solving examples based on Worst-case analysis of an algorithm. 2. **Development** (30 minutes) a. Understand Objectives of Analyzing Algorithms - how the worst, and best case of an algorithm depends on the size of the user input value. - Elaborate why the running time varies among different instances of the input. - Defining Best and Worst Case of an algorithm. - Understanding the examples based on worst-case Analysis b. Calculate the upper limit of the execution time of an algorithm. c. Solving Example of Linear Search and analyzing the searching problem. d. Understanding how the worst, case of an algorithm depends on the size of the user input value. e. Best Case and Worst-Case Analysis of an Algorithm Link: https://youtu.be/5g7K86jYto8 f. Summarize the Objectives of Best Case and Worst-Case

	Analysis of an Algorithm 3. **Exercise** (10 minutes) – - Have a discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: **Activity** : Solving Examples based on Worst Case Analysis of an Algorithm Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Asking open-ended questions on Worst-case analysis through nearpod. **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.5	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Average-case analysis**
Objectives	At the end of the lesson the student shall be able to: a. Understand how the average case of an algorithm depends on the size of the user input value. b. Explain the importance of average case analysis compared to worst-case and best-case analyses. c. perform average case analysis on various algorithmic problems. d. Apply average case analysis to real-world problems and algorithms. e. Understand how input distributions affect average case analysis.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: What mathematical tools and techniques are commonly used in average case analysis? - Can you explain the concept of probability distributions and how they are used in average case analysis? - How does the average case performance of a sorting algorithm differ from its worst-case performance? Use an example like QuickSort or MergeSort. - Solve examples based on Average-case analysis of an algorithm. 2. **Development** (30 minutes) a. Discuss a real-world scenario where average case analysis provides more meaningful insights than worst-case analysis. b. Can you analyze the average case performance of hash table operations such as insertion, deletion, and search? c. Understanding how the average case of an algorithm depends on the size of the user input value. d. Best Case, Worst Case and Average Case Analysis of an Algorithm Link: https://youtu.be/5g7K86jYto8 e. Summarize the Objectives of Average Case Analysis of an Algorithm 3. **Exercise** (10 minutes) – - Have a discussion to summarize the lecture - Ask Questions Related to Topics

Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Homework**: **Activity**: Solving Examples based on Average Case Analysis of an Algorithm Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Asking open-ended questions on Average-case analysis through nearpod. **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.6	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Use of order notations and related results**
Objectives	At the end of the lesson, the student shall be able to: a. Understanding the process of approaching a value or curve arbitrarily closely. b. Explain the significance of Resources for an algorithm that is usually expressed as a function regarding input. c. Describe the concept of Asymptotic Notation. d. Understanding the characteristics of an algorithm's efficiency. e. Solve examples on Big-oh, Omega, and Theta Notations.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: What do you understand by running time for an algorithm? How complexity of an algorithm can be analyzed? What is the best way to study Function growth efficiently? - Focus on the significance of Resources for an algorithm. - Have a discussion on the Asymptotic Notation and its applications. - Introducing three notations that are used to calculate the running time complexity of an algorithm 2. **Development** (30 minutes) a. Divide & Conquer Strategy - Explaining which function grows slowly with the input size as compared to others? - Elaborate ways of comparing different running times b. Describe the concept of Asymptotic Notations - Understand the ways to find the approximation in terms of upper, lower and tight bound. - Illustrating asymptotic tight bound with help of examples. - Calculating Asymptotic Upper and Lower Bound of a function. c. Describe why the use of O notations is preferable instead of Θ notations even though Θ could be more appropriate. d. Explaining factors that play a significant role in understanding the running time for an algorithm? e. Asymptotic Notations | Big O | Big Omega | Theta Notations Link: https://www.youtube.com/watch?v=7dz8Iaf_weM

	f. Summarize the Objectives of Use of order notations and related results. 3. **Exercise** (10 minutes) – - Have a discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the Significance of Big-oh , Omega and Theta Notations. **Activity References**: **a.** https://learnxinyminutes.com/docs/asymptotic-notation **b.** https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/asymptotic-notation Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Google form Quiz on solving examples of Big-oh , Omega and Theta Notations **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.7	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Recurrence Equations
Objectives	At the end of the lesson the student shall be able to: a. Define what a recurrence equation is and understand its role in analyzing algorithms. b. Identify the basic components of recurrence equations, including the initial conditions and recurrence relations. c. Explain the significance of recurrence equations in the analysis of recursive algorithms. d. Analyze the efficiency of recursive algorithms by solving their recurrence equations.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: What is a recurrence equation in the context of algorithm analysis? - Why are recurrence equations important in the design and analysis of algorithms? - How do recurrence equations help in understanding the performance of recursive algorithms? - What methods can be used to solve recurrence equations? - How does the substitution method work for solving recurrence equations? 2. **Development** (30 minutes) a. How can recurrence equations be applied to dynamic programming problems? b. What is the difference between homogeneous and non-homogeneous recurrence equations? c. How can you solve non-homogeneous recurrence equations in algorithm analysis?? d. Recurrence Equations Link:https://www.youtube.com/watch?v=G5B9on-JUoM&list=PL8tc66sMn9KiZiMN1D5DUue31QYTXVsXj e. Summarize the Objectives of Recurrence Equations. 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture

	- Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Google form Quiz on solving examples of Recurrence Equations **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.8	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Divide and conquer recurrences**
Objectives	At the end of the lesson the student shall be able to: a. Understand the Objectives of Divide-and-conquer algorithms. b. Explaining how Recurrences are used to analyze the computational complexity of an algorithm. c. Describe the process of Dividing the problem into smaller sub-problems. d. Understanding the concept of Solving those sub-problems. e. Combining the solutions for those smaller subproblems to solve the original problem.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: What are the resources for an algorithm that are usually expressed as a function regarding input? Elaborate the significance of Asymptotic Notation. Explain process of approaching a value or curve arbitrarily closely. - Focus on how we can describe a function in terms of its values on smaller inputs. - Have a discussion on Binary search to justify the concept of Divide and conquer recurrences. 2. **Development** (30 minutes) a. Understand Objectives of Learning Algorithms - Divide the problem into subproblems that are smaller instances of the same problem. - Conquer the subproblems by solving them recursively. If the subproblems are small enough, solve them trivially or by "brute force." - Combine the subproblem solutions to give a solution to the original problem. - Solving Linear Recurrences b. Understanding how the recursive nature of D&C leads to recurrences. c. Explaining how the Recurrences are used to analyze the computational complexity of divide-and-conquer algorithms.

	d. Solving the Binary search problem using D & C approach. e. Divide and Conquer Approach Link: https://www.youtube.com/watch?v=I8w2XN0w-fQ f. Summarize the Objectives of Recurrence Relation and How to Write Binary Search using Recurrence Relation. 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Divide and conquer recurrences **Activity References**: **a.** https://eng.libretexts.org/Bookshelves/Computer_Science/Programming_and_Computation_Fundamentals/Mathematics_for_Computer_Science_(Lehman_Leighton_and_Meyer)/05%3A_Recurrences/21%3A_Recurrences/21.04%3A_Divide-and-Conquer_Recurrences **b.** http://www.cs.cmu.edu/afs/cs/academic/class/15451-s14/www/LectureNotes/lecture1_supplement.pdf Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Asking Open ended questions on Objectives of Divide and conquer recurrences **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.9	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Recurrence relations: substitution method**
Objectives	At the end of the lesson the student shall be able to: a. Understand Objectives of condensed way of proving an asymptotic bound on a recurrence by induction. b. Explaining how to find a closed-form bound on the recurrence. c. Conclude that how substitution method is a powerful approach that is able to prove upper bounds for almost all recurrences. d. Understanding the use of induction in solving recurrence relations. e. Describing importance of adding additional terms to the upper bound. f. Solve Substitution method examples.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: How the recursive nature of D&C leads to recurrences. How the Recurrences are used to analyze the computational complexity of divide-and-conquer algorithms. Explain the process of Solving Linear Recurrences. - Focus on obtaining a function defined on the natural numbers that satisfies the recurrence. - Have a discussion on Using the mathematical induction to find the boundary condition and show that the guess is correct. 2. **Development** (30 minutes) a. Understand Objectives of the condensed way of proving an asymptotic bound on a recurrence by induction. - Process to get a recurrence relation for time complexity - Get the running time on an input of size n as a function of n and the running time on inputs of smaller sizes. - Guess for the solution and then use mathematical induction to prove the guess is correct or incorrect. - Determine a tight asymptotic lower bound for the example recurrence relation. b. Describing the importance of adding additional terms to the upper bound. c. Conclude that how substitution method is a powerful approach that is able to prove upper bounds for almost all recurrences. d. Substitution Method Link : https://youtu.be/icS-e8RaCyo

	e. Summarize the Objectives of learning Recurrence relations: substitution method 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a.** **Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b.** **Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Substitution Method **Activity References**: **a.** https://www.cs.cornell.edu/courses/cs3110/2014sp/recitations/24/using-the-substitution-and-master-method.html **b.** https://walkccc.me/CLRS/Chap04/4.3/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Nearpod activity on solving given example on substitution method **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.10	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Recurrence trees method
Objectives	At the end of the lesson, the student shall be able to: a. Understand the Objectives of drawing a recurrence tree and calculate the time taken by every level of the tree. b. Explaining how master theorem allows many recurrence relations of a form to be converted to Θ-notation directly, without doing an expansion of the recursive relation. c. Conclude how the master theorem always yields asymptotically tight bounds to recurrences from divide and conquer algorithms d. Understanding the way how recursion trees can be useful for gaining intuition about the closed form of a recurrence.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: - What is a recurrence tree, and how does it help in solving recurrence relations? - How do you construct a recurrence tree for a given recurrence relation? - What are the key components of a recurrence tree (e.g., root, nodes, levels)? - How do you determine the number of levels in a recurrence tree? 2. **Development** (30 minutes) a. Construct a recurrence tree for the recurrence relation $T(n)=2T(n/2)+nT(n) = 2T(n/2) + nT(n)=2T(n/2)+n$ and solve it. b. How would you solve the recurrence relation $T(n)=T(n-1)+nT(n) = T(n-1) + nT(n)=T(n-1)+n$ using a recurrence tree? c. Provide an example of a more complex recurrence relation and demonstrate how to solve it using a recurrence tree. d.Recursive Tree Method Link : https://www.youtube.com/watch?v=HBUFYO9gqvA Summarize the Objectives of learning Recurrence relations: Recursive Tree Method 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture

	- Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Recursive Tree Method **Activity References**: **a.** https://www.cs.cornell.edu/courses/cs3110/2012sp/lectures/lec20-master/lec20.html Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Google form Quiz on solving given examples on Recursive Tree Method. **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.11	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Master's theorem
Objectives	At the end of the lesson the student shall be able to: a. Classify the function f(n) and apply the appropriate case of Master's Theorem to determine the asymptotic behavior of the recurrence relation. b. Apply Master's Theorem to analyze the time complexity of real-world algorithms, such as Merge Sort, Binary Search, and Strassen's Algorithm for matrix multiplication. c. Conclude how the master theorem always yields asymptotically tight bounds to recurrences from divide and conquer algorithms. d. Identify recurrence relations of the form T(n)=aT(n/b)+f(n) and recognize when Master's Theorem is applicable.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: What is the Master's Theorem and in what context is it used in the design and analysis of algorithms? - Why is the Master's Theorem important for analyzing the time complexity of recursive algorithms? - What are the three cases of the Master's Theorem, and what conditions determine which case to use? - What is the standard form of a recurrence relation to which the Master's Theorem can be applied? - What are the three cases of the Master's Theorem, and what conditions determine which case to use? 2. **Development** (30 minutes) a. How do you identify the values of a, b, and f(n) in a recurrence relation for the application of the Master's Theorem? b. Can you apply the Master's Theorem to the recurrence T(n)=2T(n/2)+ n? Explain your steps and the result. c. Apply the Master's Theorem to the recurrence $T(n)=4T(n/2)+n^2$ and explain the result. d. Master Theorem Link : https://youtu.be/vny0ZVVLpbA 3. Summarize the Objectives of learning Recurrence relations: Master Theorem 4. **Exercise** (10 minutes) –

	- Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Master Theorem **Activity References**: **a.** https://www.programiz.com/dsa/master-theorem **b.** https://www.cs.cornell.edu/courses/cs3110/2012sp/lectures/lec20-master/lec20.html Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Google form Quiz on solving given examples on Master Theorem **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1.12	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Applications of Master's Theorem**
Objectives	At the end of the lesson, the student shall be able to: a. Apply the Master's Theorem to solve recurrence relations commonly encountered in algorithm design. b. Recognize recurrence relations that can be solved using the Master's Theorem. c. Use the Master's Theorem to determine the time complexity of various divide-and-conquer algorithms. d. Formulate recurrence relations for real-world problems and solve them using the Master's Theorem.
Teaching Aids (if any)	a. Presentation Slides b. YouTube Video
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: - How can Master's Theorem be used to analyze the time complexity of divide-and-conquer algorithms? - Can you provide a step-by-step example of applying Master's Theorem to a specific recurrence relation? - How does the theorem help simplify the process of finding the asymptotic complexity of recursive algorithms? - What are the conditions for each case of Master's Theorem? 2. **Development** (30 minutes) a. How does Master's Theorem relate to the recursion tree method and the iteration method? b. Can you derive the Master's Theorem from first principles and explain the intuition behind it? c. Explore the relationship between Master's Theorem and other methods for solving recurrence relations. d. Master Theorem Link : https://www.youtube.com/watch?v=4aePc0qOL70 e. Summarize the Objectives of learning Application of Master Theorem 3. **Exercise** (10 minutes) – - Have a discussion to summarize the lecture - Ask Questions Related to Topics

Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Application of Master Theorem **Activity References**: **a.** https://www.programiz.com/dsa/master-theorem **b.** https://www.cs.cornell.edu/courses/cs3110/2012sp/lectures/lec20-master/lec20.html Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Google form Quiz on solving given examples on Application of Master Theorem **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2.1	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Quick Sort and its analysis**
Objectives	At the end of the lesson the student shall be able to: a. Understand how Quick Sort uses a divide and conquer paradigm for sorting. b. Explaining execution trace of partition algorithm. c. Analyse the algorithm and doing its Worst-Case Analysis. d. Elaborating the Relational Formula for Worst Case e. Deriving Relational Formula for Randomized Quick Sort f. Doing Best and Average Case Analysis
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: How to calculate the time taken by every level of a recursion tree? How master theorem allows many recurrence relations of a form to be converted to Θ-notation directly? What is the main principal behind working of Divide & Conquer strategy? - Understand how quick sort consumes relatively fewer resources during execution. 2. **Development** (30 minutes) a. Discuss the Pseudo Code for recursive QuickSort function b. Explain the Pseudo code for partition () function. c. Understand how quick sort uses only a small auxiliary stack d. Elaborate the procedure of Partitioning the Array e. Understand about extremely short inner loop of algorithm. f. Performing a thorough mathematical analysis of the algorithm. g. Listing the advantages and disadvantages of Quick sort Algorithm. h. Analyzing the Worst Case, Best Case and Average Case Complexity of the algorithm. i. How to optimize QuickSort so that it takes O(Log n) extra space in worst case?

	j. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/ Quick Sort Algorithm - Divide and Conquer Link : https://youtu.be/IBnyUwCoY4I How Quick Sort Works | Performance of Quick Sort with Example Link: https://youtu.be/tWCaFVJMUi8 k. Summarize the Objectives of learning Quick Sort Algorithm and performing its analysis 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Quick Sort Algorithm and performing its analysis **Activity References**: **a.** https://www.gatevidyalay.com/tag/quick-sort-in-daa/ **b.** https://www.codesdope.com/course/algorithms-quicksort/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Nearpod / Mentimeter short Quiz on Quick Sort Algorithm and its analysis **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2.2	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Merge Sort recurrence**
Objectives	At the end of the lesson the student shall be able to: a. Understand how Merge Sort uses a divide and conquer paradigm for sorting. b. Explaining how algorithm divides the problem into sub problems and solves them individually. c. Elaborating the process of combining the results of sub problems to get the solution of the original problem d. Analyse the Time Complexity of Merge Sort(Best, Average and Worst Case). e. Explaining Merge Sort Applications.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: How Quick Sort uses a divide and conquer paradigm for sorting? What is Worst-Case Complexity of Quick Sort. How to derive the Relational Formula for Randomized Quick Sort? - Explaining the working principal of Merge Sort using suitable example. 2. **Development** (30 minutes) a. Discuss the Top-down Merge Sort Implementation and see how Merge sort repeatedly breaks down a list into several sub lists until each sub list consists of a single element. b. Consider an example to understand the approach better. c. Understand the process of Merging of two lists. d. Elaborating the Implementation of Merge Sort. e. Explaining the process how Bottom-Up merge sort approach uses iterative methodology. f. Analyzing the Time Complexity of Merge Sort recurrence. g. Explaining how Merge Sort is useful for sorting linked lists in O(nlogn) time. h. Listing the reasons behind the Slower processing comparative to the other sort algorithms for smaller tasks.

	i. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/ Recursion tree method: intuition | Merge Sort Link : https://youtu.be/C4JjXc0htp0 Solving the Merge Sort recurrence Link: https://youtu.be/LPfHV3xWwXg j. Summarize the Objectives of learning Merge Sort Algorithm and performing its analysis 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Merge Sort Algorithm and performing its analysis **Activity References**: **a.** https://www.gatevidyalay.com/tag/merge-sort-algorithm-in-daa/ **b.** https://www.interviewbit.com/tutorial/merge-sort-algorithm/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Mentimeter short Quiz on Merge Sort Algorithm and its analysis **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2.3	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Strassen's matrix multiplication**
Objectives	At the end of the lesson the student shall be able to: a. Understand the significance of Strassen's Algorithm for matrix multiplication. b. Explaining how this algorithm is faster than the naive matrix multiplication algorithm. c. Elaborating the conditions for Strassen's algorithm to work. d. Understanding the Procedure of Strassen matrix multiplication. e. Analyse the Time Complexity of Strassen's matrix multiplication. f. Explaining the Disadvantages of Strassen's matrix multiplication
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: How Merge Sort uses a divide and conquer paradigm for sorting? What are the Applications of Merge Sort ? What is the Time Complexity of Merge Sort algorithm. How Merge Sort uses a divide and conquer paradigm for sorting. - Explaining the basic working principal of Strassen's matrix multiplication. 2. **Development** (30 minutes) a. Discuss the Problem Statement of Naive method during matrix multiplication. b. Analyzing the Complexity for Naïve method of matrix multiplication. c. Understanding the idea of Strassen's method to reduce the number of recursive calls. d. Understanding the formula behind the working of algorithm. e. Calculate the product of A and B (matrix C) with the help of learned formulas. f. Describing the different conditions for Strassen's algorithm to work. g. Understanding how Strassen's matrix multiplication algorithm is asymptotically faster than the naive algorithm. h. Explaining the reasons why generally Strassen's Method is not preferred for practical applications.

	i. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/ Strassen's Matrix Multiplication Link : https://youtu.be/Te08SanNEN8 Strassen's Matrix Multiplication - Divide and Conquer - Analysis of Algorithm Link: https://youtu.be/UnpySHwAJsQ j. Summarize the Objectives of learning Strassen's matrix multiplication and performing its analysis 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Strassen's matrix multiplication and performing its analysis **Activity References**: **a.** https://iq.opengenus.org/strassens-matrix-multiplication-algorithm/ **b.** https://medium.com/swlh/strassens-matrix-multiplication-algorithm-936f42c2b344 Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Google form short Quiz on Strassen's Matrix Multiplication and its analysis **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2.4	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Fast multiplication of large integers**
Objectives	At the end of the lesson the student shall be able to: a. Analysing asymptotic efficiency by ingenious application of the divide-and-conquer technique. b. Explaining efficient manipulation of large integers. c. Elaborating the basic idea of the algorithm. d. Understanding the Procedure Fast multiplication e. Analyse the Complexity of Fast multiplication and its optimization.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: What is the significance of Strassen's Algorithm for matrix multiplication? What are the conditions for Strassen's algorithm to work. How Strassen's matrix multiplication algorithm is asymptotically faster than the naive algorithm? why generally Strassen's Method is not preferred for practical applications? - Understanding how much effort does it take to do long multiplication of two numbers. 2. **Development** (30 minutes) a. Discuss the Problem Statement of Naive method during matrix multiplication. b. Analyzing the process of addition of long numbers. c. Understanding the idea of short multiplication: a number times a digit. d. Understanding the formula behind the working of algorithm. e. Perform the analysis of long multiplication. f. Explaining Karatsuba's method for multiplying numbers of any length g. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/

	Large Integer Multiplication - Divide and Conquer - Analysis of Algorithm Link : https://youtu.be/BwHZXuzux44 Multiplying Long Integers Using Divide and Conquer Technique Link: https://youtu.be/_YIAcrBwpas h. Summarize the Objectives of learning Multiplication of Long Integers (Faster than Long Multiplication) 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Multiplication of Long Integers (Faster than Long Multiplication) **Activity References**: **a.** https://people.mpiinf.mpg.de/~mehlhorn/ftp/chapter2A-en.pdf **b.** https://www.brainkart.com/article/Multiplication-of-Large-Integers_8025/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Google form short Quiz on Multiplication of Long Integers (Faster than Long Multiplication) **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2.5	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Binary Search trees**
Objectives	At the end of the lesson the student shall be able to: a. Understanding the nodes arrangement in a Binary Search Tree. b. Explaining the basic operations associated with a tree. c. Implementation of Search, Delete and Insert operations in a BST. d. Understanding the architecture of a basic binary search algorithm. e. Analyse the Attributes of Binary Search Tree. f. Explaining the Types of Binary Trees g. Practice Problems Based on Binary Search Trees
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: What do you understand by a tree? What is the difference between a tree and a graph? How to Perform the analysis of long multiplication? How we can analyze asymptotic efficiency by ingenious application of the divide-and-conquer technique? - Understanding the importance of Traversal in Binary Search Trees. 2. **Development** (30 minutes) a. Discuss how we can recursively define a rooted binary tree. b. Explaining In-Order, PREORDER and POSTORDER Tree Walk traversals along with suitable examples. c. Understanding the idea of Querying a Binary Search Tree. d. Understanding the process of Tree Search and Iterative Tree Search. e. How we can define Successor and predecessor in a BST. f. Explaining algorithms for Tree Insertion and Deletion. g. Practice Problems Based on Binary Search Trees h. Explaining How Good Are Binary Search Trees? i. Understanding the concept of Balanced Search Trees.

	j. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/ Binary Search Tree | BST | Design & Algorithms Link : https://youtu.be/cvooTtquDlQ Insertion and Traversal in BST Link: https://youtu.be/sXABdGalFNg k. Summarize the Objectives of learning Binary Search trees and doing analysis. 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Binary Search trees **Activity References**: **a.** https://www.gatevidyalay.com/binary-search-trees-data-structures/ **b.** https://www.studytonight.com/data-structures/binary-search-tree Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Nearpod Activity on problems based on Binary Search trees. **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2.6	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Priority queues**
Objectives	At the end of the lesson the student shall be able to: a. Introduction to the concept of Priority queues b. Explaining the basic operations associated with a Priority queue. c. Understanding how to change priority of an arbitrary element. d. How to join two Priority queues. e. Explaining the basic operations associated with a Max Priority queue. f. Implementation of Priority queues with various techniques.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: What are the Attributes of Binary Search Tree? How we can recursively define a rooted binary tree? How we can define Successor and predecessor in a BST. What do you understand by Balanced Search Trees? - Understanding how the higher priority elements are served first in a priority queue. 2. **Development** (30 minutes) a. Understanding the process of Assigning Priority Value in a priority queue. b. Explaining Difference between Priority Queue and Normal Queue. c. Understanding the Implementation of Priority Queues using different Data Structures. d. Explaining the process of Inserting an Element into the Priority Queue. e. Describing the process of Deleting an Element from the Priority Queue. f. Understanding concept of Peeking from the Priority Queue (Find max/min). g. Explaining concept of Extract-Max/Min from the Priority Queue. h. Explaining the Priority Queue Applications.

	i. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/ Priority Queue Data Structure Link : https://www.youtube.com/watch?v=NlEwbC6Nt0c Priority Queue - Implementation with Example Link: https://youtu.be/OxhYCLWMdHs j. Summarize the Objectives of learning Binary Search trees and doing analysis. 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Understanding the working of Priority queues **Activity References**: **a.** http://www.infocobuild.com/education/audio-video-courses/computerscience/DesignAnalysisOfAlgorithms-CMI/lecture-34.html **b.** https://algs4.cs.princeton.edu/24pq/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Nearpod Activity on problems based on Priority queues **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2.7	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Heaps, Heap Sort and its analysis**
Objectives	At the end of the lesson the student shall be able to: a. Utilize the Heap to sort the given array. b. Understanding special heap properties. c. Describe the concept of Complete and incomplete Binary tree. d. Create a Heap data structure from unsorted list. e. Explaining the basic operations associated with Min- Heap and Max Heap. f. Analyse the Complexity of Heap Sort. g. Implementing Heap Sort Algorithm.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) - Ask questions: What are the basic operations associated with a Priority queue? Explain the different applications of Priority Queue. How we can assign Priority Value in a priority queue. How the higher priority elements are served first in a priority queue. How we can define Successor and predecessor in a BST. - Understanding how the heap is also the data structure that serves as a cornerstone of a theoretically important sorting algorithm called heapsort. 2. **Development** (30 minutes) a. Understanding the conditions required to form a heap. b. To check whether the binary tree is essentially complete. c. Understanding the process of implementing heap property. d. Explaining the process of Inserting an Element into the Priority Queue. e. Describing the list of important properties of heaps. f. Understanding concept of Heap and its array representation. g. How can we construct a heap for a given list of keys? h. Implementation of Heap Sort Algorithm using heapify function. i. Analyzing the Worst Case , Best Case and Average Case Complexity of an algorithm.

	j. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/ Heap Sort | Heapify Method | Build Max Heap Algorithm Link : https://youtu.be/Q_eia3jC9Ts Heap sort with Example | Heapify Method Link: https://youtu.be/nJ6FdAIr_6g k. Summarize the Objectives of learning Heaps, Heap Sort and performing its analysis 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Practice to implement Heap Sort and learn its analysis **Activity References**: **a.** https://www.kodnest.com/free-online-courses/algorithm-2/lessons/transform-and-conquer-approach/topic/heaps-and-heap-sort/ **b.** https://www.interviewbit.com/tutorial/heap-sort-algorithm/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** Nearpod quiz Activity on problems based on Heap Sort and its analysis **3.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3.1	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Data structures for disjoint sets**
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of Union and Find Operation. b. Keep track of a set of elements partitioned into several disjoint (non-overlapping) subsets. c. Analyze the worst-case running time of approach. d. Illustrate the significance of Union by rank and Path compression.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What are the basic operations associated with Min- Heap and Max Heap? - What are the properties of Complete Binary tree? - How we can analyze the Complexity of Heap Sort. 2. **Development** (30 minutes) a. Understanding the disjoint-set-union data structure along with examples. b. To manage the operations of union and find. c. Understanding the process of finding connected components from the graph. d. Explaining the algorithm for Union by Rank and Path compression. e. Analyzing the Brute force approach and its complexity with optimized approach. f. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/ Disjoint Set | UNION and FIND Link : https://www.youtube.com/watch?v=eTaWFhPXPz4 Disjoint Sets using union by rank and path compression Graph Algorithm Link: https://www.youtube.com/watch?v=ID00PMy0-vE

	g. Summarize the Objectives of learning union by rank and path compression Graph Algorithm. 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a.** **Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b.** **Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Practice to implement union by rank and path compression **Activity References**: **a.** https://www.hackerearth.com/practice/data-structures/disjoint-data-strutures/basics-of-disjoint-data-structures/tutorial/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3.2	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Concept of Path compression and Union by Rank**
Objectives	At the end of the lesson the student shall be able to: a. To check whether a given graph contains cycle or not b. To attach smaller depth tree under the root of the deeper tree. c. Understand the concept of Union and Path Compression. d. To articulate how size (in place of height) of trees can also be used as *rank*.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - Explain union() and find() operations for subsets. - How to flatten the tree when find() is called. - How we can analyze the Time Complexity of Find and Union Operations. 2. **Development** (30 minutes) a. Using a structure to represent an edge in the graph. b. Creates a graph with V vertices and E edges c. To find set of an element i using path compression technique. d. Understanding the disjoint-set-union data structure along with examples. e. To manage the operations of union and find. f. Understanding the process of finding connected components from the graph. g. Explaining the algorithm for Union by Rank and Path compression. h. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/ Disjoint Set | UNION and FIND Link : https://www.youtube.com/watch?v=eTaWFhPXPz4

	Disjoint Sets using union by rank and path compression Graph Algorithm Link: https://www.youtube.com/watch?v=ID00PMy0-vE i. Summarize the Objectives of learning union by rank and path compression Graph Algorithm. 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Practice to implement union by rank and path compression **Activity References**: **a.** https://www.hackerearth.com/practice/data-structures/disjoint-data-strutures/basics-of-disjoint-data-structures/tutorial/ **b.** https://www.geeksforgeeks.org/union-find-algorithm-union-rank-find-optimized-path-compression/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3.3	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Prim's and Kruskal's algorithms**
Objectives	At the end of the lesson the student shall be able to: a. Sort all the edges in non-decreasing order of their weight. b. Pick the smallest edge. Check if it forms a cycle with the spanning-tree. c. Build the Minimum Spanning Tree from any vertex in the graph using Prim's Algorithm. d. Build the Minimum Spanning Tree from the vertex carrying minimum weight in the graph. e. To Analyse the complexity of Both Algorithms.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - How we can check whether a given graph contains cycle or not - What do you understand by Union by Rank? - Explain the significance of Path Compression Algorithm. - How size of trees can also be used as rank? 2. **Development** (30 minutes) a. Create a set that keeps track of vertices already included in MST. b. Assign a key value to all vertices in the input graph. c. Traversing one node more than one time (prim's algorithm) to get the minimum distance. d. Traversing one node only once in case of Kruskal's algorithm. e. Analyzing the differences between Two Algorithms. f. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/ Prim's Algorithm for Minimum Cost Spanning Tree | Prims vs Kruskal Link : https://www.youtube.com/watch?v=_KX8GDvRzBc Kruskal Algorithm for Minimum Spanning TreeLink: https://www.youtube.com/watch?v=huQojf2tevI

	g. Summarize the Objectives of learning union by rank and path compression Graph Algorithm. 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Practice to implement Prim's and Kruskal's algorithms **Activity References**: 1. https://www.gatevidyalay.com/prims-and-kruskal-algorithm-difference/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3.4	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Huffman coding**
Objectives	At the end of the lesson the student shall be able to: a. Understand significance of lossless data compression algorithm. b. Illustrate the significance of using Fixed Length and Variable Length Codes. c. Creation and Traversal of Huffman Tree d. Process of Encoding and Decoding the stream of characters. e. Analysing the algorithm and finding Time Complexity.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What do you understand by Spanning Tree? - How we can build the Minimum Spanning Tree from any vertex in the graph using Prim's Algorithm. - Explain the differences between Prim's and Kruskal's algorithm. - How to Analyze the complexity of Both Algorithms. 2. **Development** (30 minutes) a. Build a Huffman Tree from input characters. b. Build a Huffman Tree from input characters. c. Create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. d. Extract two nodes with the minimum frequency from the min heap. e. Create a new internal node with a frequency equal to the sum of the two nodes frequencies. f. Time Complexity Analysis of Huffman Coding Algorithm. g. Applications of Huffman Coding. h. NPTEL Link: https://nptel.ac.in/courses/106/106/106106131/ Huffman Coding Algorithm | Greedy Technique Link : https://www.youtube.com/watch?v=uDS8AkTAcIU Huffman coding example -Greedy Method Link:

	https://www.youtube.com/watch?v=saofdNsZiYY i. Summarize the Objectives of learning union by rank and path compression Graph Algorithm. 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. **Home work**: 3. **Activity** : Practice to implement Huffman coding **Activity References**: 1. https://www.gatevidyalay.com/huffman-coding-huffman-encoding/ Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity on Huffman coding Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3.5	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Dijkstra's shortest path algorithm**
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of finding the shortest path in a graph. b. Learn the steps involved in Dijkstra's algorithm for finding the shortest path. c. Apply Dijkstra's algorithm to solve real-world problems related to network routing and optimization. d. Gain proficiency in implementing Dijkstra's algorithm in a programming language of choice.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is the significance of finding the shortest path in a network? - How does Dijkstra's algorithm differ from other graph traversal algorithms? - Can you identify scenarios where Dijkstra's algorithm would be most useful? - What data structures are crucial for the implementation of Dijkstra's algorithm? - How does the algorithm handle graphs with negative edge weights? 2. **Development** (30 minutes) a. Implement Dijkstra's algorithm in a programming language of your choice. b. Discuss any challenges faced during the implementation and how they were overcome. c. Experiment with different strategies to optimize Dijkstra's algorithm. d. Compare the performance of the original and optimized versions on various graph sizes. e. Research and present a case study where Dijkstra's algorithm was crucial for solving a real-world problem. f. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=pVfj6mxhdMw

	3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity on Dijkstra's shortest path algorithm Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3.6	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**LZW Coding**
Objectives	At the end of the lesson the student shall be able to: a. Understand the principles of LZW coding. b. Implement LZW compression and decompression algorithms. c. Recognize applications and use cases of LZW coding.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is LZW coding, and what problem does it aim to solve in data compression? - How does LZW differ from other compression algorithms, such as Huffman coding? - Can you explain the basic steps involved in the LZW compression process? - What challenges or limitations might be associated with LZW coding? - In what real-world scenarios or applications can LZW coding be particularly useful? 2. **Development** (30 minutes) a. Discuss the LZW compression process step by step. How does it handle repetitive patterns in data? b. What is the role of the dictionary in LZW coding, and how is it dynamically updated during compression? c. How does LZW decompression work? Can you outline the key steps involved? d. Compare LZW coding with other compression algorithms in terms of efficiency and complexity. e. Explore practical examples where LZW coding has been successfully applied. f. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=9gkiphD-VIY&t=53s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics

Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a.** **Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b.** **Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity on LZW Coding Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4.1	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Dynamic Programming: Basic concepts**
Objectives	At the end of the lesson the student shall be able to: a. Understand the fundamental concepts of Dynamic Programming. b. Explore the advantages of using Dynamic Programming in problem-solving. c. Learn how to identify problems suitable for Dynamic Programming.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - How can we break down a problem into subproblems to find an optimal solution?" - Why does identifying and solving overlapping subproblems lead to more efficient solutions? 2. **Development** (30 minutes) a. What characteristics make a problem suitable for a dynamic programming approach? b. How does memoization differ from tabulation in the context of Dynamic Programming? c. In what scenarios would you prefer a bottom-up approach over a top-down approach, and vice versa? d. Can you think of any real-world problems where Dynamic Programming could be a suitable solution? Why? e. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=0BhhiQGDbEA&t=97s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson,

	R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4.2	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Matrix-chain multiplication**
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of matrix chain multiplication. b. Learn the importance of optimal parenthesization in matrix multiplication. c. Explore dynamic programming techniques for optimizing matrix chain multiplication. d. Gain proficiency in implementing matrix chain multiplication algorithms.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is matrix chain multiplication, and why is it relevant in computer science? - Can you provide an example where matrix multiplication plays a crucial role? - Why is the order of multiplication crucial in matrix multiplication? - What challenges can arise if we don't optimize the parenthesizing of matrices? - What is dynamic programming, and how does it relate to solving problems like matrix chain multiplication? - Can you think of other problems where dynamic programming can be applied? - Discuss the time and space complexity of the naive matrix multiplication algorithm. - 2. **Development** (30 minutes) a. Break down the steps involved in solving matrix chain multiplication using dynamic programming. b. How does the concept of subproblems contribute to the efficiency of the algorithm? c. Demonstrate the process of finding the optimal parenthesization for a given set of matrices. d. How can we ensure that the chosen parenthesization is indeed optimal? e. Walk through the implementation of the matrix chain multiplication algorithm in a programming language of your choice.

	f. Can you identify potential optimizations or further improvements in the algorithm? g. Explore real-world applications of matrix chain multiplication. h. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=OnuQ_7l6lk0&t=105s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity on Matrix Chain Multiplication Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4.3	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	DP solution for traveling salesman problem
Objectives	At the end of the lesson the student shall be able to: a. Understand the Traveling Salesman Problem (TSP) and its significance in optimization. b. Learn the basics of Dynamic Programming (DP) and its application to solve combinatorial optimization problems. c. Explore the step-by-step approach of developing a DP solution for the Traveling Salesman Problem. d. Analyze the time and space complexity of the DP solution for TSP. e. Gain hands-on experience by implementing the DP solution in a programming environment.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is the Traveling Salesman Problem, and why is it considered a combinatorial optimization challenge? - How does Dynamic Programming differ from other optimization techniques? - Can you think of real-world scenarios where finding an optimal route is crucial? 2. **Development** (30 minutes) a. How does Dynamic Programming address the overlapping subproblems in the context of the TSP? b. Can you identify the optimal substructure in the Traveling Salesman Problem? c. How does memoization enhance the efficiency of the DP solution for TSP? d. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=3QiSyc7KyC4&t=64s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.

	2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4.4	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	DP solution for 0/1 Knapsack problems
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of the 0/1 Knapsack Problem. b. Explore the importance of dynamic programming in solving optimization problems. c. Learn the step-by-step approach of the dynamic programming solution for the 0/1 Knapsack Problem. d. Apply the acquired knowledge to solve real-world scenarios using dynamic programming techniques. e. Analyze the time and space complexity of the dynamic programming solution for the 0/1 Knapsack Problem.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is the 0/1 Knapsack Problem, and how does it relate to real-world scenarios? - Why is dynamic programming a suitable approach for solving optimization problems? - What are the key steps involved in the dynamic programming solution for the 0/1 Knapsack Problem? - How does the choice of subproblems contribute to the overall optimization solution? 2. **Development** (30 minutes) a. Break down the dynamic programming algorithm for the 0/1 Knapsack Problem step by step. b. Discuss the importance of defining and solving subproblems in this context. c. Demonstrate the implementation of the dynamic programming solution in a programming language of choice. d. Discuss how to handle different constraints and variations of the 0/1 Knapsack Problem. e. Explore ways to optimize the dynamic programming solution in terms of time and space complexity. f. Discuss the trade-offs involved in choosing specific optimization strategies.

	g. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=i8NqAEsZn54 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity on Knapsack problem Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4.5	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Concept of least common subsequences and Independent Sets**
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of Least Common Subsequences (LCS) and its applications. b. Explore the significance of Independent Sets in various mathematical and computational contexts. c. Learn the algorithms and techniques for finding LCS and Independent Sets. d. Apply the acquired knowledge to solve real-world problems related to sequence matching and graph theory.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is the purpose of finding the Least Common Subsequence in sequence analysis? - How does LCS differ from other sequence-related algorithms? - What is the role of Independent Sets in graph theory? - Can you provide examples of real-world scenarios where Independent Sets are applicable? - How do dynamic programming algorithms contribute to finding the LCS? - In what ways do graph algorithms play a role in identifying Independent Sets? - How can LCS be applied in bioinformatics or data analysis? - Why are Independent Sets important in network design or resource allocation problems? 2. **Development** (30 minutes) a. Demonstrate the step-by-step process of finding the LCS between two sequences. b. Discuss the time and space complexity of the algorithm. c. Explore how to identify Independent Sets in a given graph. d. Discuss real-world scenarios where finding Independent Sets is crucial. e. How does dynamic programming contribute to efficient LCS computation? f. Can you explain the concept of memoization and its role in LCS algorithms?

	g. Discuss graph traversal algorithms for identifying Independent Sets. h. Provide examples of graphs where finding Independent Sets is non-trivial. i. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=jHGgXV27qtk&t=237s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4.6	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Backtracking: General Method**
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of backtracking in algorithm design. b. Explore the general approach and application of backtracking. c. Learn to implement backtracking to solve combinatorial problems. d. Analyze the time and space complexity of backtracking algorithms.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is backtracking, and how does it differ from other algorithmic techniques? - Can you provide examples of real-world problems where backtracking is an effective solution strategy? - How does backtracking handle situations where decisions need to be revised? - What are the key characteristics of problems suitable for a backtracking approach? 2. **Development** (30 minutes) a. Discuss the basic structure of a backtracking algorithm. b. How does backtracking utilize recursion in problem-solving? c. Provide examples of problems that can be solved using the backtracking approach. d. Demonstrate how backtracking can be applied to combinatorial problems such as permutations and combinations. e. Discuss the space requirements and potential optimizations in backtracking. f. Compare the efficiency of backtracking with other problem-solving techniques. g. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=-47nFq_9_zk&t=38s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture

	- Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity on Backtracking algorithm Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4.7	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Breadth-First and Depth-First search algorithms**
Objectives	At the end of the lesson the student shall be able to: a. Understand the fundamental concepts of Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms. b. Differentiate between BFS and DFS and identify scenarios where each algorithm is applicable. c. Implement BFS and DFS algorithms to traverse graphs and analyze their time complexities. d. Recognize real-world applications of BFS and DFS in computer science and problem-solving.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is graph traversal, and why is it essential in computer science? - Can you name some real-world problems where graph traversal is applicable? - Can you give an example of a scenario where BFS would be more suitable than DFS? - How is recursion utilized in DFS, and what advantages does it offer? - In what situations might DFS be a preferred choice over BFS? - Can you think of a graph where BFS and DFS would produce different results? - Compare the time complexities of BFS and DFS. - When might one algorithm be more efficient than the other, and why? 2. **Development** (30 minutes) a. Develop a recursive algorithm for Depth-First Search. b. Code a DFS traversal for a given graph. c. Analyze the time complexity of your BFS and DFS implementations. d. Discuss the space complexity and memory requirements of both algorithms. e. Discuss the advantages and disadvantages of each algorithm in specific contexts. f. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/

	YouTubeLink: https://www.youtube.com/watch?v=N2P7w22tN9c&t=38s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.1	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**Topological Sort**
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of topological sorting in directed acyclic graphs (DAGs). b. Identify scenarios where topological sorting is applicable. c. Implement topological sorting algorithms. d. Analyze the time complexity of topological sorting algorithms.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is a directed acyclic graph (DAG), and why is it important in the context of topological sorting? - How does topological sorting help in identifying the order of tasks or events with dependencies? - Can you provide an example scenario where topological sorting would be beneficial? - What are the key differences between Kahn's algorithm and depth-first search (DFS) for topological sorting? - How does the presence of cycles in a graph affect the possibility of topological sorting? 2. **Development** (30 minutes) a. Define topological sorting and its significance. b. Walk through an example to illustrate the concept. c. Discuss the algorithm's time complexity and its suitability for different scenarios. d. Compare and contrast DFS with Kahn's algorithm. e. Provide hands-on exercises for implementing Kahn's algorithm and DFS for topological sorting. f. Explore real-world applications, such as task scheduling and course prerequisite determination. g. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=dis_c84ejhQ&t=35s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics

Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.2	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Recursive graph algorithms
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of recursion in the context of graph algorithms. b. Explore and implement recursive approaches for common graph operations. c. Analyze the time and space complexity of recursive graph algorithms. d. Apply recursive strategies to solve real-world graph-related problems.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is recursion, and how does it differ from iteration? - Why might recursive algorithms be advantageous in certain graph scenarios? - Can you name some common graph operations that could be implemented recursively? - How does the recursive approach impact the readability and simplicity of graph algorithms? 2. **Development** (30 minutes) a. What is the recursive strategy for implementing DFS on a graph? b. How can we ensure that every vertex is visited exactly once in a recursive DFS? c. Can you compare the recursive and iterative implementations of DFS in terms of simplicity and efficiency? d. Discuss the importance of directed acyclic graphs (DAGs) in topological sorting. e. How can recursion be employed to find connected components in an undirected graph? f. What are the advantages and limitations of using recursion in maze-solving algorithms? g. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=bJg_sv7PV-g&t=99s

	3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.3	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**String matching: KMP algorithm**
Objectives	At the end of the lesson the student shall be able to: a. Understand the importance of string-matching algorithms in computer science. b. Comprehend the inefficiencies of naive string-matching algorithms. c. Introduce the Knuth-Morris-Pratt (KMP) algorithm as an efficient string-matching solution. d. Learn the key components of the KMP algorithm and how they contribute to its efficiency. e. Implement the KMP algorithm in practical coding scenarios.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - Why is string matching important in computer science and real-world applications? - What are the limitations of naive string-matching algorithms, and why do we need more efficient solutions? - How does the KMP algorithm improve the efficiency of string matching? - What are the main components of the KMP algorithm, and how do they work together? - Can you outline the steps involved in implementing the KMP algorithm in a programming language of your choice? 2. **Development** (30 minutes) a. Discuss the concept of string matching and its applications in various fields. b. Explain the inefficiencies of naive string-matching algorithms, highlighting their time complexity. c. Introduce the Knuth-Morris-Pratt (KMP) algorithm and its historical significance. d. Break down the key components of the KMP algorithm, including the pre-processing step and the actual search process. e. Guide students through the implementation of the KMP algorithm in a step-by-step manner. f. Encourage discussions on real-world scenarios where the KMP algorithm could be advantageous. g. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/

	YouTubeLink: https://www.youtube.com/watch?v=GTJr8OvyEVQ&t=157s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.4	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Rabin-Karp algorithm
Objectives	At the end of the lesson the student shall be able to: a. Understand the basics of string searching algorithms. b. Explore the Rabin-Karp algorithm and its significance in pattern matching. c. Learn the key concepts of hash functions and their role in the Rabin-Karp algorithm. d. Implement the Rabin-Karp algorithm in a practical example.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - Why is string matching important in computer science and applications? - Can you think of real-world scenarios where efficient string matching is crucial? - How does it utilize hashing to expedite string matching? - What are the primary components of the Rabin-Karp algorithm? - How does the rolling hash function work in the context of this algorithm? - In what situations does the Rabin-Karp algorithm outperform other string matching algorithms? 2. **Development** (30 minutes) a. Demonstrate the step-by-step process of implementing the Rabin-Karp algorithm for a given string matching problem. b. How can you handle collisions in the hash function? c. How does the choice of the hash function impact the algorithm's performance? d. Explore possible optimizations or variations of the Rabin-Karp algorithm. e. Can you modify the algorithm to accommodate different types of patterns or text sources? f. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/

	YouTubeLink: https://www.youtube.com/watch?v=H4VrKHVG5qI&t=135s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity on Rabin-Carp Algorithm Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.5	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Polynomial Time Algorithms
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of polynomial time algorithms in the context of computational complexity theory. b. Recognize the significance of polynomial time algorithms in solving computational problems efficiently. c. Explore common examples of polynomial time algorithms and their applications. d. Analyze the difference between polynomial time and non-polynomial time algorithms. e. Evaluate the efficiency and scalability of algorithms through the lens of polynomial time complexity.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - How does polynomial time differ from non-polynomial time, and why is this classification important? - Can you give an example of a problem that is known to have a polynomial time solution? - In what scenarios would we prefer using polynomial time algorithms over non-polynomial time algorithms? 2. **Development** (30 minutes) a. Compare and contrast polynomial time algorithms with non-polynomial time algorithms. b. Explore examples of well-known polynomial time algorithms, such as sorting algorithms or graph algorithms. c. Analyze the implications of P vs NP problem in the context of polynomial time algorithms. d. Discuss the significance of polynomial time algorithms in practical problem-solving scenarios. e. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=ZSyZozw6JMQ&t=347s 3. **Exercise** (10 minutes) –

	- Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.6	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Basics of number Theory Algorithms
Objectives	At the end of the lesson the student shall be able to: a. Understand fundamental concepts of number theory. b. Explore basic number theory algorithms and their applications. c. Develop problem-solving skills related to number theory challenges. d. Gain proficiency in implementing number theory algorithms in programming.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is number theory, and why is it important in computer science? - Can you provide real-world examples where number theory is applied? - What is a prime number, and how can we determine if a number is prime? - How do factors and multiples relate to number theory? - What is the Euclidean algorithm, and how does it find the greatest common divisor (GCD)? - How can we use the Sieve of Eratosthenes to generate prime numbers efficiently? - In what ways are number theory algorithms used in cryptography? - Can you think of examples where modular arithmetic is applied in computing? 2. **Development** (30 minutes) a. Discuss the historical significance of number theory in the development of algorithms. b. Explore how number theory forms the foundation for various cryptographic techniques. c. Break down the steps involved in determining whether a number is prime or composite. d. Investigate the relationship between prime factorization and unique factorization in number theory. e. Implement the Euclidean algorithm in a programming language

	of your choice. f. Demonstrate the step-by-step process of the Sieve of Eratosthenes for finding prime numbers. g. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=69jsFIMINpI&t=35s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.7	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	**GCD and extended Euclidean algorithm**
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of Greatest Common Divisor (GCD) and its significance in number theory. b. Learn the basic algorithm for finding the GCD of two numbers. c. Explore the Extended Euclidean Algorithm and its application in finding modular inverses. d. Apply the Extended Euclidean Algorithm to solve linear Diophantine equations. e. Gain practical insights into the applications of GCD and Extended Euclidean Algorithm in computer science and cryptography.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is the significance of finding the Greatest Common Divisor (GCD) in number theory? - Can you provide real-world examples where GCD plays a crucial role? - How would you compute the GCD of two numbers using prime factorization? - Can you think of more efficient ways to find the GCD without prime factorization? - Why might we need an extended version of the Euclidean Algorithm? - How can the Extended Euclidean Algorithm be used to find modular inverses? - In what areas of computer science or cryptography might we encounter problems that involve the Extended Euclidean Algorithm? - How does the Extended Euclidean Algorithm help in solving linear Diophantine equations? 2. **Development** (30 minutes) a. Demonstrate the step-by-step process of finding the GCD of two numbers using the Euclidean Algorithm. b. How can you use the GCD to simplify fractions? c. Illustrate the Extended Euclidean Algorithm with a numerical example.

	d. Explain how the Extended Euclidean Algorithm can be applied to find modular inverses. e. Discuss a specific example where GCD is used in computer science, programming, or cryptography. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=yHwneN6zJmU&t=10s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.8	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Primality Testing
Objectives	At the end of the lesson the student shall be able to: a. Students will understand the concept of primality testing and be able to identify prime numbers. b. Familiarize students with various primality testing algorithms. c. Enable students to implement simple primality tests in programming languages. d. Develop critical thinking skills in analyzing the efficiency of different primality testing methods.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - Definition and characteristics of prime numbers. - Why are prime numbers important in cryptography? - What is the Sieve of Eratosthenes? - Can you think of real-world applications where primality testing is crucial? - Discuss applications in computer science, cryptography, and number theory. - How do primality testing algorithms differ in terms of efficiency? 2. **Development** (30 minutes) a. Break down the steps involved and discuss the time complexity. b. Discuss the underlying principles and conditions for Fermat's test. c. Address issues like false positives and false negatives in testing methods. d. Introduce the concept of probabilistic primality testing. e. Discuss algorithms like Miller-Rabin and their trade-offs between accuracy and efficiency. f. Encourage students to write a simple program to test primality and discuss optimization techniques. g. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/

	YouTubeLink: https://www.youtube.com/watch?v=sDrXeCs3ghQ&t=209s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity on LZW Coding Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.9	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Basic Concepts and Non-Deterministic Algorithms
Objectives	At the end of the lesson the student shall be able to: a. Understand the fundamental concepts of algorithms and their significance in problem-solving. b. Differentiate between deterministic and non-deterministic algorithms. c. Grasp the basic characteristics and applications of non-deterministic algorithms. d. Evaluate scenarios where non-deterministic algorithms may be more suitable.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is the role of algorithms in computer science and problem-solving? - How would you define a deterministic algorithm? - Can you provide an example of a problem where a deterministic algorithm might not be the most efficient solution? - What distinguishes non-deterministic algorithms from deterministic ones? - In what real-world scenarios can non-deterministic algorithms be beneficial? - What are some basic characteristics of non-deterministic algorithms? - How do they handle randomness or uncertainty in problem-solving? 2. **Development** (30 minutes) a. Explore the concept of non-deterministic algorithms and how they differ from deterministic counterparts. b. Investigate practical applications of non-deterministic algorithms in fields such as optimization, cryptography, or artificial intelligence. c. Discuss how non-deterministic algorithms leverage probability and randomness to arrive at solutions. d. Introduce the concept of NP-completeness and its relevance to non-deterministic algorithms.

	e. Examine situations where non-deterministic algorithms might offer efficiency advantages over deterministic alternatives. f. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=ZNe1ziMExGg&t=273s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.10	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Polynomial Time and algorithms
Objectives	At the end of the lesson the student shall be able to: a. Define and understand the concept of polynomial time in the context of algorithms. b. Differentiate between polynomial and non-polynomial time complexity. c. Identify algorithms with polynomial time complexity. d. Analyze the efficiency of algorithms using Big-O notation.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - Definition and implications. - Examples of polynomial time algorithms (e.g., bubble sort, merge sort). - Brief introduction to non-polynomial time complexity (e.g., exponential time complexity). - Understanding the implications of non-polynomial time. - Explanation of Big-O notation as a tool for describing algorithmic complexity. - Practical examples illustrating Big-O notation. 2. **Development** (30 minutes) a. Analyze the time complexity of a specific algorithm (e.g., binary search) and determine if it is polynomial. b. Compare the time complexity of two sorting algorithms, one with polynomial time complexity and the other with non-polynomial time complexity. c. Discuss a real-world problem and propose an algorithm to solve it, considering the importance of polynomial time. d. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=VVsStlZ30V8&t=36s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics

Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No.5.11	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	NP-hard & NP –complete classes
Objectives	At the end of the lesson the student shall be able to: a. Understand the concepts of NP-Hard and NP-Complete problems. b. Differentiate between problems that belong to P, NP, NP-Hard, and NP-Complete classes. c. Analyze the implications of NP-Hard and NP-Complete problems on computational complexity. d. Explore strategies for dealing with NP-Complete problems.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What are complexity classes, and why are they essential in computational theory? - Can you provide examples of problems that are easy to solve (in P) and those that are hard to solve (beyond P)? - What is the definition of an NP-Hard problem? - How does an NP-Hard problem differ from an NP-Complete problem? - What does it mean for a problem to be NP-Complete? - Can you explain how Cook's theorem is related to NP-Completeness? - How does the verification of a solution relate to NP problems? - Why is it easier to verify a solution than to find one for NP problems? 2. **Development** (30 minutes) a. Discuss examples of problems that are known to be NP-Hard. b. How can reductions be used to show that a problem is NP-Hard? c. Analyze real-world problems that fall into the NP-Complete category. d. Can you identify situations where dealing with NP-Complete problems is unavoidable? e. Explain the significance of Cook's theorem in establishing NP-Completeness. f. How does Cook's theorem impact the way we understand the complexity of computational problems? g. Explore heuristic methods and approximation algorithms for solving NP-Complete problems.

	h. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=sJT1Zzubz5A&t=11s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.12	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Cook's Theorem
Objectives	At the end of the lesson the student shall be able to: a. Understand the fundamental concepts of Cook's Theorem in computational complexity theory. b. Explore the significance of Cook's Theorem in the context of the theory of computation. c. Gain insights into the relationship between the complexity classes P and NP. d. Apply the key ideas from Cook's Theorem to analyze and categorize computational problems.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is the primary focus of Cook's Theorem? - How does Cook's Theorem contribute to our understanding of computational complexity? - Can you explain the concept of polynomial-time reducibility? 2. **Development** (30 minutes) a. How does Cook's Theorem relate to the P versus NP problem? b. Discuss any real-world implications or applications of Cook's Theorem. c. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=JYBoaoPkaQM&t=63s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta, C. Papadimitriou, Umesh

	Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments **3.** Nearpod Activity Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5.13	Course Name: Design and Analysis of Algorithms	Course No.: COM-401

Topics	Introduction to Approximation Algorithms
Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of approximation algorithms and their significance in solving optimization problems. b. Explore the trade-off between accuracy and efficiency in algorithm design. c. Recognize real-world applications where approximation algorithms play a crucial role. d. Analyze the performance guarantees and bounds associated with approximation algorithms.
Teaching Aids (if any)	a. PPTs. b. Green board (Chalk and Talk). c. Video Lectures by NPTEL / Youtube d. Use of Nearpod / Mentimeter / Kahoot tools.
Teaching Development	1. **Introduction** (5 minutes) Ask questions: - What is the primary goal of optimization problems in computer science? - Why are exact solutions often impractical for NP-hard problems? - Can you provide an example of a real-world problem that can be modeled as an optimization problem? - How do approximation algorithms strike a balance between optimality and efficiency? - What are the challenges associated with designing and analyzing approximation algorithms? 2. **Development** (30 minutes) a. How do we measure the performance of an approximation algorithm? b. Discuss the concepts of approximation ratio and approximation factor. c. How do greedy algorithms contribute to approximation algorithms? d. Provide examples where greedy strategies can be applied. e. Explain the concept of NP-hardness. f. Discuss why finding exact solutions for NP-hard problems is often infeasible. g. Explore specific approximation algorithms for well-known problems (e.g., Traveling Salesman Problem, Knapsack Problem). h. Discuss their applications and limitations.

	i. NPTEL Link: https://archive.nptel.ac.in/courses/106/106/106106131/ YouTubeLink: https://www.youtube.com/watch?v=u5rqrQqGk3E&t=16s 3. **Exercise** (10 minutes) – - Have discussion to summarize the lecture - Ask Questions Related to Topics
Closure	1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. 2. Suggested Reading books: **a. Introduction to Algorithms** by T.Cormen, C. Lieserson, R.Rivest, C.Steina. Prentice-Hall/India, 3rd Edition. **b. Algorithms** by S. Dasgupta,C. Papadimitriou, Umesh Vazirani. McGraw Hill Education, 1st Edition. Spend 5 minutes to wrap up and consolidate the leanings.
Evaluation	**1.** Reflective Questions (What, Why, Who?). Allow students to answer and discuss. **2.** MCQ / Sessional Test / Assignments Spend 5 minutes to evaluate student assimilation of the lesson contents