Faculty of Technology and Engineering

Academic Year : 2024-25 **Semester** : 4th

Course code : AIML207 **Course name** : Design and Analysis of Algorithms

Date: 16th Dec, 2024

Practical List

Sr. No.	Aim		Hours	CO
1	**Implement and analyze algorithms using iterative and recursive approaches for the problems given below. (Annexure 1,2)**		2	
	1.1	Find the factorial of a given number.		
	1.2	Print the Fibonacci sequence up to the given number.		
	Questions that students need to answer : (for learning outcomes) 1. Compare the time complexity of the recursive and iterative versions of the above problems. Which one is more efficient in terms of time and space? 2. What is stack overflow in the context of recursion? How does this problem arise, and how can it be mitigated?			
2	**Implement and analyze the best case, average case and worst case of the algorithms for problems given below. (Annexure 1,2)**		2	
	2.1	Given a **sorted array** of integers, find the **first occurrence** of a target element **x**. If the target element is not found, return **-1**. Explore alternatives of searching such elements and analyze.		
	2.2	Given an array of integers, use **Insertion Sort** algorithm to sort the array in ascending order.		

	2.3	Given an array of integers **num** and an integer **target**, return indices of the two numbers such that they add up to the target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.		

		Can you come up with an algorithm that is less than O(n^2) time complexity? https://leetcode.com/problems/two-sum/		
	Questions that students need to answer for conclusion: 1. How can you identify the most appropriate algorithm for a given problem based on its constraints, input size, and time/space requirements? 2. What steps would you take to further optimize any of the algorithms you implemented for scenarios with very large input sizes? 3. Compare insertion sort with other comparison based sorting algorithms.			
3	**Divide and Conquer(Annexure 2)**		2	
	3.1	You are managing a logistics company that delivers parcels across various cities. The parcels are stored in a large warehouse in an unsorted array, where each element represents the weight of a parcel in kilograms. To optimize the delivery routes, you need to sort the parcels by weight before loading them onto trucks. Your task is to implement **Divide and Conquer** algorithms to sort the parcel weights. Additionally, based on the sorted weights, determine the number of trucks required to transport the parcels, given that each truck has a maximum capacity of C kilograms. Compare implemented Sorting algorithm's time complexity analysis: worst case, average case and best case.		
	Questions that students need to answer for conclusion: 1. What are the key advantages of using a Divide and Conquer approach in algorithm design compared to traditional iterative approaches? 2. How selection of pivot elements affects the performance of quick sort algorithms? 3. When implementing a Divide and Conquer algorithm, what are some potential pitfalls or challenges you may face in terms of performance or memory usage? How would you address these issues?			
4	**Greedy Approach(Annexure 2)**		8	

	4.1	There is an online media platform that sells advertisement slots during a popular live-streaming event. Each ad slot has a certain duration in seconds, and advertisers are willing to pay different amounts for these slots. However, you have a limited total ad duration available for the event, and you want to maximize the revenue generated by selling these slots.		

		To maximize revenue, you can sell entire ad slots or fractions of a slot(e.g., selling half of a 10-second slot). Your task is to select the ad slots in such a way that the total revenue is maximized without exceeding the available time.		
	4.2	A city is installing an **emergency response system**. The city has multiple hospitals, fire stations, and police stations that need to be reached as quickly as possible during emergencies. Each road in the city has a specific travel time, and during emergencies, it's crucial to send the fastest available response team to the incident location. Your job is to find the **shortest travel time** from any emergency station (hospital, fire station, or police station) to a specific incident location, ensuring the response is as quick as possible. You are given the locations of all emergency stations and the incident site, and you need to compute the **shortest time** for any emergency station to reach the incident.		
	4.3	A telecom company tasked with connecting several cities using fiber optic cables. Each pair of cities can be connected by a cable, but the cost of laying the cable between two cities varies depending on the distance and terrain. The goal is to connect **allthe cities** with the **minimum total cost**. The network should not form any cycles. The list of possible connections between cities **cannot be represented as an adjacency matrix** due to memory constraints (this would be inefficient for sparse graphs). Return the total minimum cost required to connect all cities, or **-1** if it is not possible to connect all cities.		
	4.4	Structured batch wise Lab Tasks on Greedy Algorithm Techniques: Task Disclosure During Lab Sessions.		
	Questions that students need to answer for conclusion: 1. How does the **greedy algorithm** help in solving this problem optimally? Could there be any case where the greedy approach might not work if fractional items were not allowed in the first problem? 2. Compare greedy approach with divide and conquer approach.			

5	**Dynamic Programming Approach(Annexure 2)**		6	
	5.1	You are organizing **a music festival** and need to pack supplies in a limited number of trucks for transport. Each truck has a maximum capacity, and you have a variety of supplies, each with a specific weight and value. Your goal is to maximize the total value of the supplies transported without exceeding the truck's weight limit. You are given:		
		• An integer array **weights[]** where **weights[i]** represents the weight of the ith supply item. • An integer array **values[]** where **values[i]** represents the value of the ith supply item. • An integer **capacity** representing the maximum weight the truck can carry. You need to determine the maximum total value of supplies that can be packed into the truck.		
	5.2	In a **bioinformatics lab** where you need to compare two DNA sequences to find the longest common DNA subsequence. This is essential for identifying similarities and differences between the genetic material of different organisms, which can help in various biological studies. Given two strings, **s1** and **s2**, representing the DNA sequences, your task is to find the length of the longest common DNA subsequence between them.		

	5.3	An image processing system that applies a series of transformations to a collection of images. Each transformation is represented as a matrix operation. To process an image, these transformations must be applied in sequence, with the output of one operation serving as the input to the next. The efficiency of the image processing pipeline depends heavily on the order in which these matrix operations are performed, as the computational cost of multiplying matrices varies with their dimensions. Your goal is to determine the optimal sequence of matrix operations that minimizes the total computational cost(number of scalar multiplications) required to process the image. Given a chain < A1, A2,...,An> of n matrices, where for i=1,2,...,n matrix Ai with dimensions. Implement the program to fully parenthesize the product A1,A2,...,An in a way that minimizes the number of scalar multiplications. Also calculate the number of scalar multiplications for all possible combinations of matrices. **Test Case n Matrices with dimensions** 1 3 A1: 3*5, A2: 5*6, A3: 6*4 2 6 A1: 30*35, A2: 35*15, A3: 15*5, A4: 5*10, A5: 10*20, A6: 20*25		
	5.4	Structured batch wise Lab Tasks on Dynamic programming: Task Disclosure During Lab Sessions.		

	Questions that students needs to answer for conclusion: 1. What key concepts did you learn about dynamic programming, and how does it differ from other problem-solving approaches like divide and conquer or greedy algorithms? 2. Can you explain the importance of **overlapping subproblems** and **optimal substructure** in dynamic programming? How did these concepts influence your approach to problem-solving? 3. Compare memoization and tabulation methods for any of the above mentioned dynamic programming tasks.		
6	**Backtracking and Branch & Bound(Annexure 2)**	2	

	6.1	You are the event manager for a large festival, and you have the opportunity to secure sponsorship from various companies. Each company offers a sponsorship amount, but their support comes with certain conditions that affect the overall budget. Each sponsorship deal has a weight (representing the budget requirement or resources needed) and a value (representing the financial contribution or visibility provided). Your goal is to maximize the total sponsorship value while staying within the festival's budget constraints. This means selecting a combination of sponsorship deals that will provide the highest overall contribution without exceeding the available budget. By employing **backtracking as well as branch-and-bound** methods, effectively decide which sponsorship deals to pursue, ensuring the festival's financial success.		
	Questions that students need to answer for conclusion: 1. Backtracking can often have an exponential time complexity. Can you explain the factors that contribute to the time complexity in the problems you solved? 2. How does pruning (cutting off branches) reduce the search space, and how did this impact the efficiency of your backtracking solutions? 3. What is the key principle behind the **Branch and Bound** approach? How does it differ from backtracking?			
7	**String Matching(Annexure 2)**		2	
	7.1	You are working on an **anti-plagiarism tool** for an educational platform. The platform allows students to submit their essays, and you need to detect if a specific **reference phrase** from a popular source material appears in the submitted essay. The goal is to find all occurrences of the **reference phrase** within the **essay** text. To ensure optimal performance, particularly for large essays or numerous reference phrases, you are required to implement the **Rabin-Karp algorithm** for string matching. **Example:**		

		Essay[] = "Academic honesty is important. Plagiarism is a serious offense. Academic integrity matters." Phrase[] = "Academic" **Output:** Pattern found at index 0 Pattern found at index 58		

	Questions that students need to answer for conclusion: 1. For long texts with short patterns, which string matching algorithm would you choose and why? 2. In a situation where multiple patterns need to be searched in a text, which algorithm might be preferable?		
8	**Real-Time Algorithmic Challenge**	2	

General Instruction

1. Analysis of the Program should contain the following:

1. Impact of Input Size on the Performance of Program. Make Table and Draw graph of Input Size vs Running Time/Total No of Instructions. Take at least Five Input of Different Size. 2. Impact of Input Quality on the Performance of Program. Make Table and Draw graph of Best Case, Worst Case and Average Case Input Quality vs Running Time/ Total No. of Instructions.
3. Rate of Growth of Program. Make Table and Draw Graph of Input Size vs Instruction(s) Running Maximum No of Time in the Program**.**

2. End-of-Topic Questions:

At the end of each topic, students are required to answer specific questions. These answers must be hand written on designated file pages for submission.

3. Annexure 1 and 2:

Students must complete analysis tasks provided in Annexure 1 and Annexure 2. Students must fill in the details in annexures and print the necessary analysis for their labwork.

4. Practical Tasks 4.4 and 5.4:

These are on-the-spot batch-wise individual tasks designed to assess understanding of the topic. These tasks can be part of the internal evaluation and should be completed during the lab session.

5. Practical Task 8:

This is a final batch-wise group task to be completed in groups of 2 or 3 students. The task will consist of moderate-level questions and must be submitted on file pages as part of the internal evaluation.

Annexure 1

Algorithm: - Approach: -

Input	Best Case			Average Case			Worst Case		
5									
10									
15									
20									
25									

Graph:

Annexure 2

Algorithm: -

Approach: -

COMPLEXITY: -

Operations									
	BEST CASE	AVERAGE CASE	WORST CASE	BEST CASE	AVERAGE CASE	WORST CASE	BEST CASE	AVERAGE CASE	WORST CASE
Time Complexity									
Space Complexity									