Devang Patel Institute of Advance Technology & Research

Department of Computer Engineering

Subject Name: Design and Analysis of Algorithms Semester: 5th
Subject Code: CE355 Year: 2022-23

Practical List

A. Course Outcomes:

After completion of the course, Students will be able to:

CO1	Analyze the asymptotic performance of algorithms.
CO2	Derive time and space complexity of different sorting algorithms and compare them to choose application specific efficient algorithm.
CO3	Understand and analyze the problem to apply design technique from divide and conquer, dynamic programming, backtracking, branch and bound techniques and understand how the choice of algorithm design methods impact the performance of programs.
CO4	Understand and apply various graph algorithms for finding shorted path and minimum spanning tree.
CO5	Understand the notations of P, NP, NP-Complete and NP-Hard.

Analysis of Program should contain following sub heading(s).

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.
4. Conclusion from the above graph or Data Table
5. For all Test cases, add column for output, calculate the answer and write the answer in the output column and verify with the output of the program.

Exp. No.	Name of Experiment		Hours	CO
1.	**Implement and analyze algorithms given below.**		**04**	1,5
	1.1	Factorial (Iterative and Recursive)		
	1.2	Fibonacci (Iterative and Recursive)		
	1.3	Matrix Addition and Matrix Multiplication (Iterative)		
	1.4	Find a subset of a given set S = {s1,s2,....., sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S= {1, 2, 5, 6, 8} and d = 9 there are two solutions {1,2,6} and {1,8}.A suitable message is to be displayed if the given problem instance doesn't have a solution.		

<table>
<tr><td></td><td colspan="3"></td><td></td></tr>
<tr><td>2.</td><td colspan="2">Implement and analyze algorithms given below. (Compare them)</td><td>02</td><td>1,2</td></tr>
<tr><td></td><td>2.1</td><td>Bubble Sort</td><td></td><td></td></tr>
<tr><td></td><td>2.2</td><td>Selection Sort</td><td></td><td></td></tr>
<tr><td></td><td>2.3</td><td>Insertion Sort</td><td></td><td></td></tr>
<tr><td></td><td colspan="3"></td><td></td></tr>
<tr><td>3.</td><td colspan="2">Divide and Conquer Strategy</td><td>04</td><td>1,3,5</td></tr>
<tr><td></td><td>3.1</td><td>Implement and perform analysis of worst case of Merge Sort and Quick sort. Compare both algorithms.</td><td></td><td></td></tr>
<tr><td></td><td>3.2</td><td>Implement the program to perform Linear Search and Binary Search. Also compare Time complexity of both.</td><td></td><td></td></tr>
<tr><td></td><td colspan="3"></td><td></td></tr>
<tr><td>4.</td><td colspan="2">Greedy Approach</td><td>04</td><td>1,2, 3,4</td></tr>
<tr><td></td><td>4.1</td><td>Program to implement Kruskal’s algorithm using greedy method.</td><td></td><td></td></tr>
<tr><td></td><td>4.2</td><td>Let S be a collection of objects with profit-weight values. Implement the fractional knapsack problem for S assuming we have a sack that can hold objects with total weight W. Check the program for following test cases:
<table>
<tr><th>Test Case</th><th>S</th><th>profit-weight values</th><th>W</th></tr>
<tr><td>1</td><td>{A,B,C}</td><td>Profit:(1,2,5)
Weight: (2,3,4)</td><td>5</td></tr>
<tr><td>2</td><td>{A,B,C,D,E,F,G}</td><td>Profit:(10,5,15,7,6,18,3)
Weight: (2,3,5,7,1,4,1)</td><td>15</td></tr>
<tr><td>3</td><td>{A,B,C,D,E,F,G}</td><td>A:(12,4),B:(10,6),
C:(8,5),D:(11,7),
E:(14,3),F:(7,1), G:(9,6)</td><td>18</td></tr>
</table></td><td></td><td></td></tr>
<tr><td></td><td>4.3</td><td>Suppose you want to schedule N activities in a Seminar Hall. Start time and Finish time of activities are given by pair of (si,fi) for ith activity.
Implement the program to maximize the utilization of Seminar Hall. (Maximum activities should be selected.)
<table>
<tr><th>Test Case</th><th>Number of activities (N)</th><th>(si,f)</th></tr>
<tr><td>1</td><td>9</td><td>(1,2), 1,3),(1,4),(2,5),(3,7),
(4,9), (5,6), (6,8), (7,9)</td></tr>
<tr><td>2</td><td>11</td><td>(1,4),(3,5),(0,6),(3,8),(5,7),
(5,9), (6,10), (8,12),(8,11)
(12,14), (2,13)</td></tr>
</table></td><td></td><td></td></tr>
<tr><td></td><td colspan="3"></td><td></td></tr>
<tr><td>5.</td><td colspan="2">Dynamic Programming</td><td>06</td><td>1,2, 3,5</td></tr>
</table>

	5.1	A cashier at any mall needs to give change of an amount to customers many times in a day. Cashier has multiple number of coins available with different denominations which is described by a set C. Implement the program for a cashier to find the minimum number of coins required to find a change of a particular amount A. Output should be the total number of coins required of given denominations. Check the program for following test cases:		

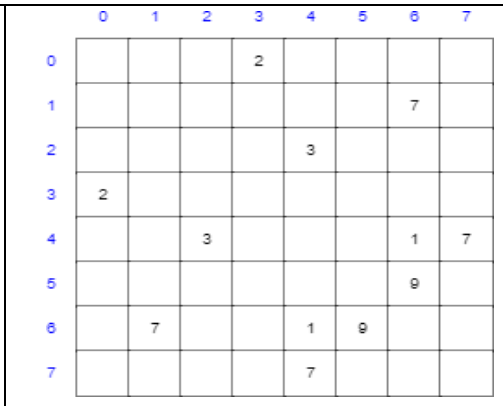
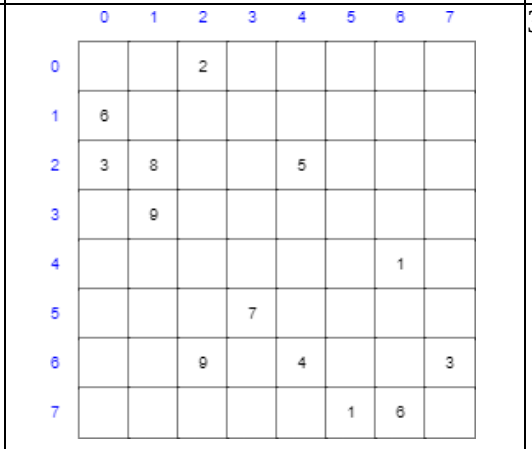
Test Case	**n**	**k**
1	₹1, ₹2, ₹3	₹ 5
2	₹18, ₹17, ₹5, ₹1	₹ 22
3	₹100, ₹25, ₹10, ₹5, ₹1	₹ 289

5.2	Implement the program 4.2 using Dynamic Programing. Compare Greedy and Dynamic approach.			
5.3	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	Program to implement all pairs shortest path.		
Graph		**06**	1,2,3, 4
6.1	Write a program to implement BFS and DFS in Graph. Compare Time Complexity of both algorithms.		
6.2	From a given vertex in a weighted graph, implement a program to find shortest paths to other vertices using Dijkstra's algorithm.		

Test Case	**Adjacency Matrix of graph**	**Start Vertex**

		1		1		
		2		3		
7.	**Backtracking**				**02**	1,2,3 ,5
	7.1	Program to implement 8-Queen's problem using Backtracking.				
8.	**String Matching Algorithm**				**02**	1,2,3
	8.1	Suppose you are given a source string S[0 ..n − 1] of length n, consisting of symbols a and b. Suppose that you are given a pattern string P[0 ..m − 1] of length m < n, consisting of symbols a, b, and *, representing a pattern to be found in string S. The symbol * is a "wild card" symbol, which matches a single symbol, either a or b. The other symbols must match exactly. The problem is to output a sorted list M of valid "match positions", which are positions j in S such that pattern P matches the substring S [j..j + |P|− 1]. For example, if S = ababbab and P = ab*, then the output M should be [0, 2]. Implement a straightforward, naive algorithm to solve the problem.					
	8.2	Implement Rabin karp algorithm and test it on the following test cases:				

Test Case	String	Pattern
1	2359023141526739921	31415 q=13
2	ABAAABCDBBABCDDEBCABC	ABC q=101