USN | | | | | | | | | | |

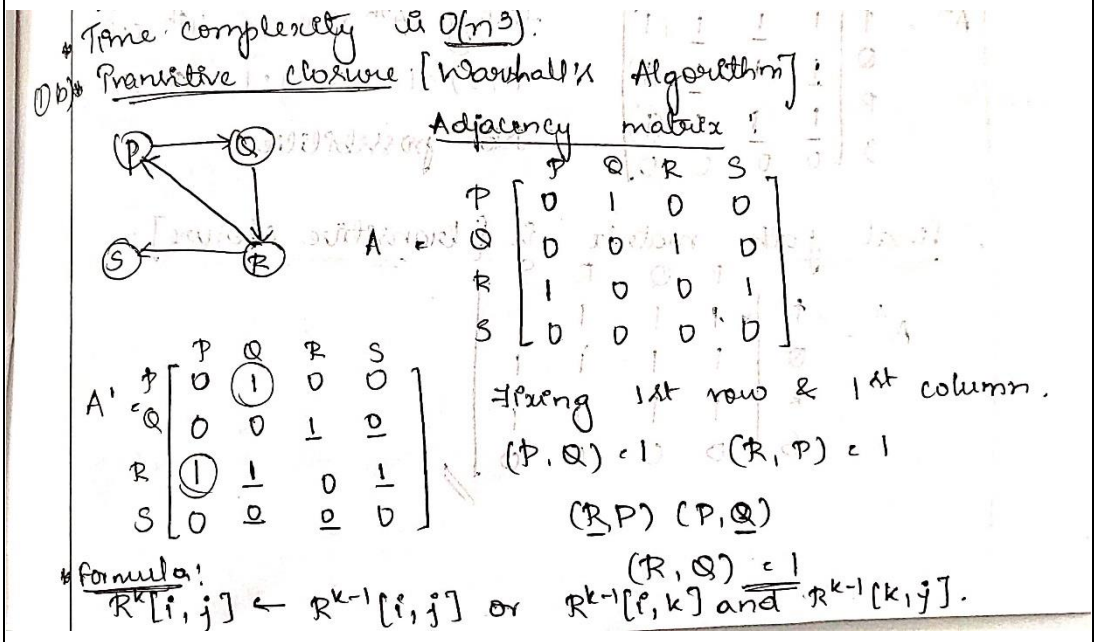
Internal Assessment Test 3 – September 2023

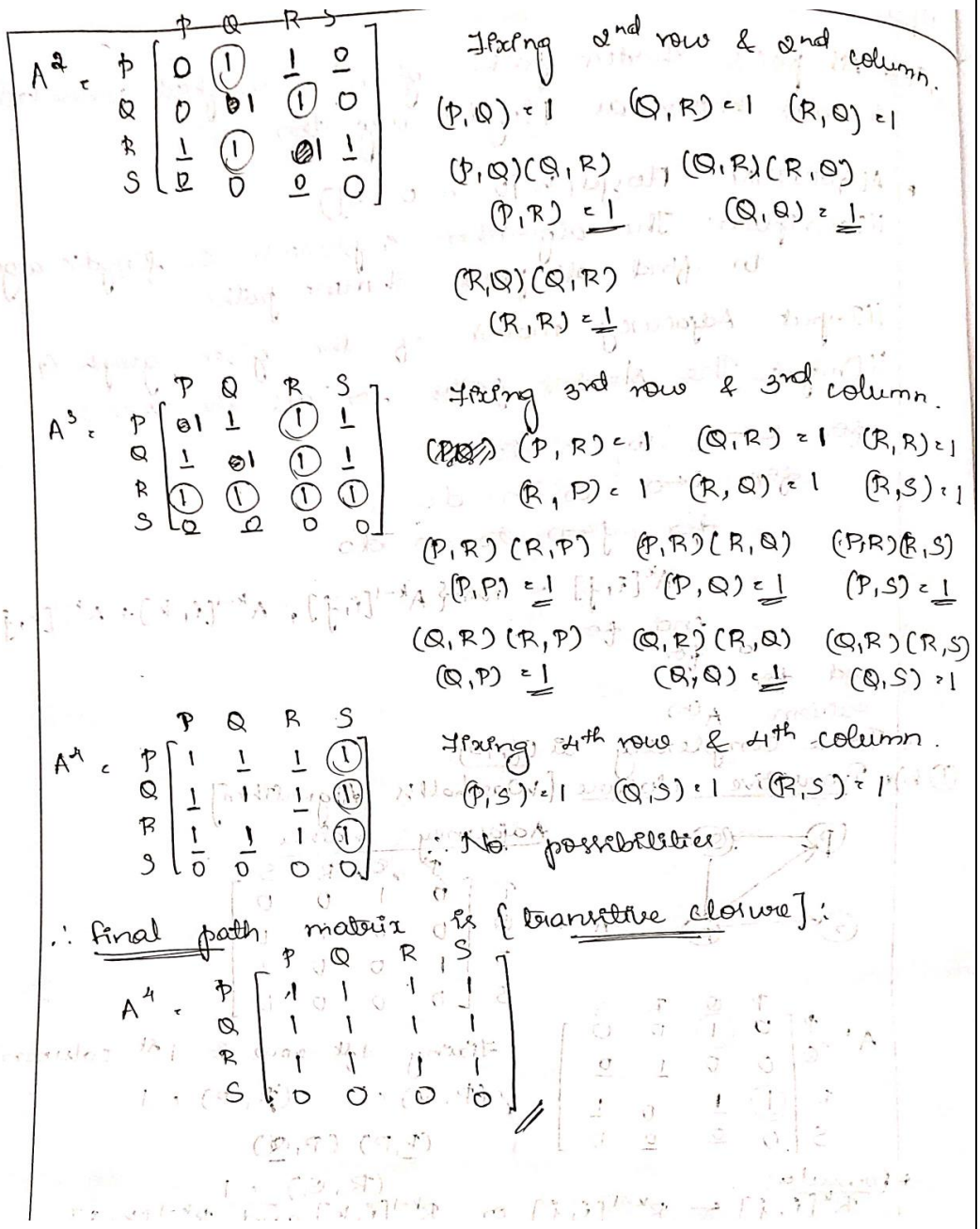
Sub:	Design and Analysis of Algorithms					Sub Code:	21CS42	Branch:	CSE
Date:	12-09-2023	Duration:	90 mins	Max Marks:	50	Sem / Sec:	IV(A, B & C)		OBE

Answer any FIVE FULL Questions SOLUTION

		MARKS	CO	RBT

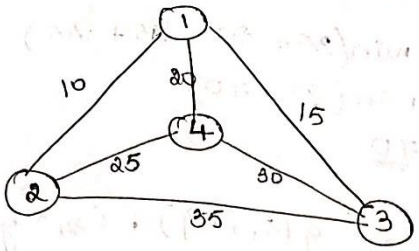
1 (a)	Design an algorithm to find all pairs of shortest paths given a weighted connected path using dynamic programming technique
	
	ALGORITHM *Floyd*$(W[1..n, 1..n])$
	//Implements Floyd's algorithm for the all-pairs shortest-paths problem
	//Input: The weight matrix W of a graph with no negative-length cycle
	//Output: The distance matrix of the shortest paths' lengths
	$D \leftarrow W$ //is not necessary if W can be overwritten
	for $k \leftarrow 1$ **to** n **do**
	  for $i \leftarrow 1$ **to** n **do**
	  for $j \leftarrow 1$ **to** n **do**
	  $D[i, j] \leftarrow \min\{D[i, j], D[i, k] + D[k, j]\}$
	return D
		5	CO4	L2

1 (b)	Find the transitive closure of the given digraph using Warshall's algorithm.  	5	CO4	L3



2 (a)	A Student **XYZ** from the CMRIT college forgot to get the Hall ticket on the day of exam and he has reached college already. Before 2 hours of his exam he realized about the issue. He has to go back to home to get Hall ticket. While travelling there are 3 cities, he has to visit exactly once and comeback to the college as early as possible to attend the exam. Analyze the Scenario and Identify the algorithm which suits and apply for the below graph. Assume Vertex 1=CMRIT, Vertices 2,3 and 4 are 3 cities.	10	CO4	L3

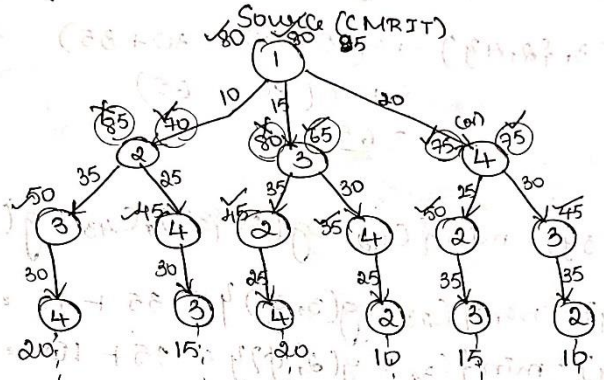
Given:



Adjacency matrix:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 10 & 15 & 20 \\ 10 & 0 & 35 & 25 \\ 15 & 35 & 0 & 30 \\ 20 & 25 & 30 & 0 \end{bmatrix} \end{array}$$

The given problem of visiting all the vertices exactly once and with minimum cost represents travelling salesman problem.



The paths can be: 1-2-4-3-1 (or) 1-3-4-2-1

From the formula of Travelling salesman problem:

$$g(i, S) = \min\{c_{ik} + g(k, \{S\} - \{k\})\}$$

$$g(1, \{2,3,4\}) = \min\{c_{12} + g(2, \{3,4\}),\ c_{13} + g(3, \{2,4\}),\ c_{14} + g(4, \{2,3\})\} \quad \text{— (1)}$$

$$g(2, \{3,4\}) = \min\{c_{23} + g(3, \{4\}),\ c_{24} + g(4, \{3\})\} \quad \text{— (2)}$$

$$g(3, \{4\}) = \min\{c_{34} + g(4, \Phi)\} = 30 + 20 = 50.$$

$$g(4, \{3\}) = \min\{c_{43} + g(3, \Phi)\} = 30 + 15 = 45.$$

② ⇒ $g(2, \{3,4\}) = \min(35+50, 25+45)$
$= \min(85, 70)$
$= 70$

$g(3, \{2,4\}) = \min\{C_{32} + g(2, \{4\}), C_{34} + g(4, \{2\})\}$ ③

$g(2, \{4\}) = \min\{C_{24} + g(4, \phi)\} = 25 + 20 = 45$

$g(4, \{2\}) = \min\{C_{42} + g(2, \phi)\} = 25 + 10 = 35$

③ ⇒ $g(3, \{2,4\}) = \min(35+45, 30+35)$
$= \min(80, 65)$
$= 65$

$g(4, \{2,3\}) = \min\{C_{42} + g(2, \{3\}), C_{43} + g(3, \{2\})\}$ ④

$g(2, \{3\}) = \min\{C_{23} + g(3, \phi)\} = 35 + 15 = 50$

$g(3, \{2\}) = \min\{C_{32} + g(2, \phi)\} = 35 + 10 = 45$

④ ⇒ $g(4, \{2,3\}) = \min(25+50, 30+45)$
$= \min(75, 785)$
$= 75$

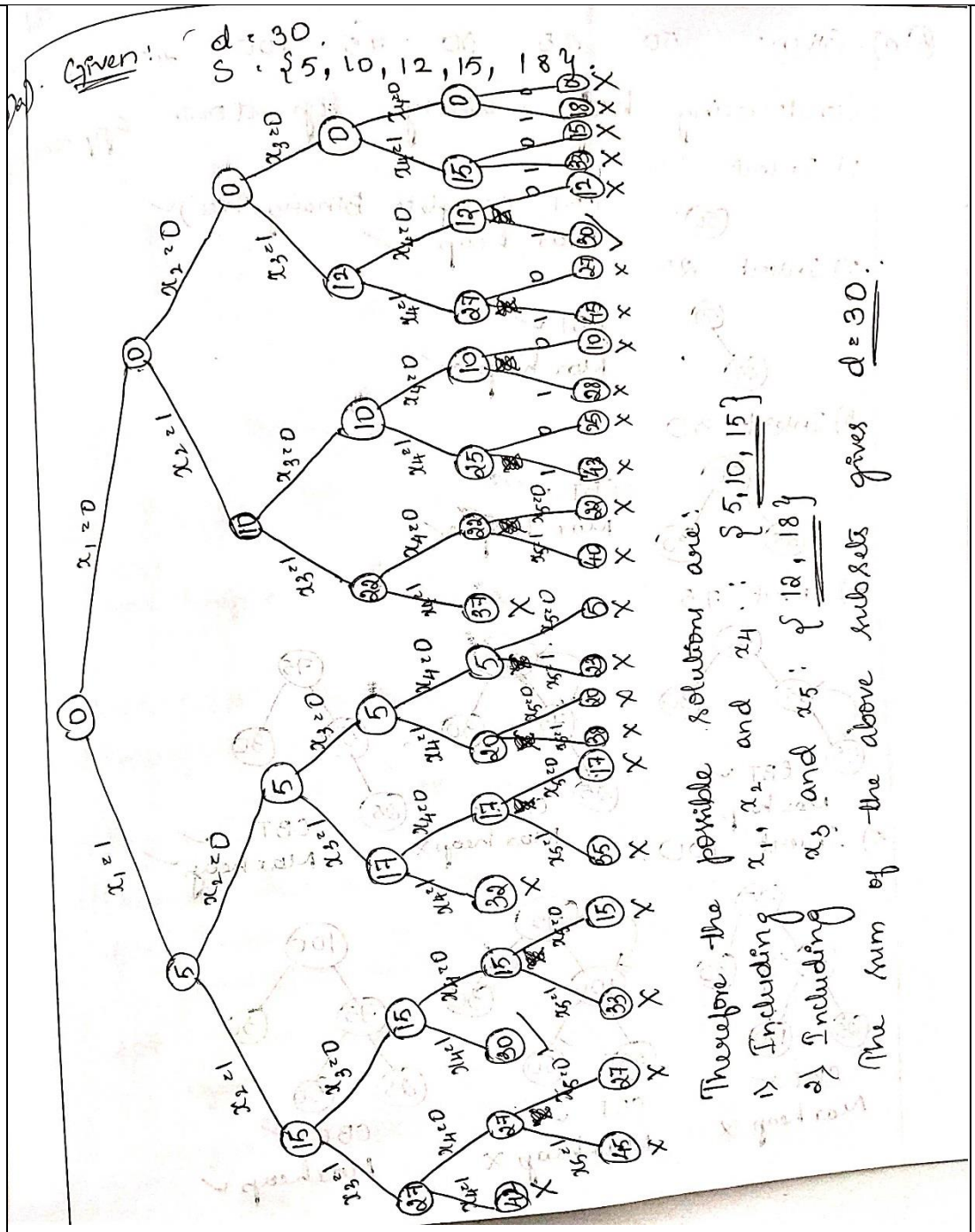
Substituting eqn ②, ③ & ④ in eqn ①;

$g(1, \{2,3,4\}) = \min(10+70, 15+65, 20+75) = \min(80, 80, 95)$

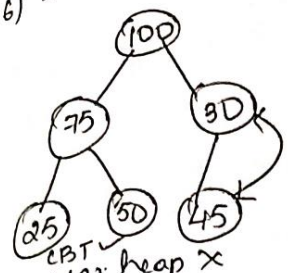
$\therefore g(1, \{2,3,4\}) = 80$

Path is: $1-2-4-3-1$ (or) $1-3-4-2-1$

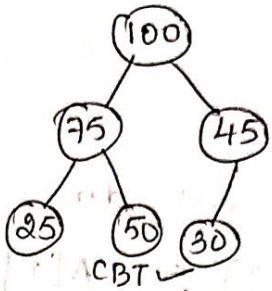
3(a)	Apply Back Tracking method to solve sum of subset problem for the instance d=30, S={5,10,12,15,18}.Give all possible solution with state space tree construction.	10	CO5	L3

				
4(a)	Construct the heap for the list 50 25 30 75 100 45 80 using top down approach.	6	CO3	L3

6) Insert 45.

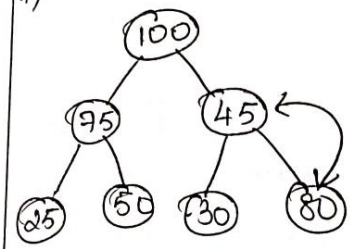


CBT ✓
Max heap ✗

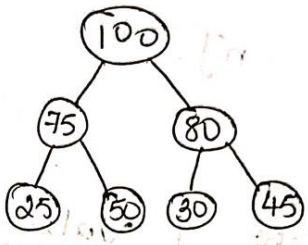


CBT ✓
Max heap ✓

7) Insert 80.

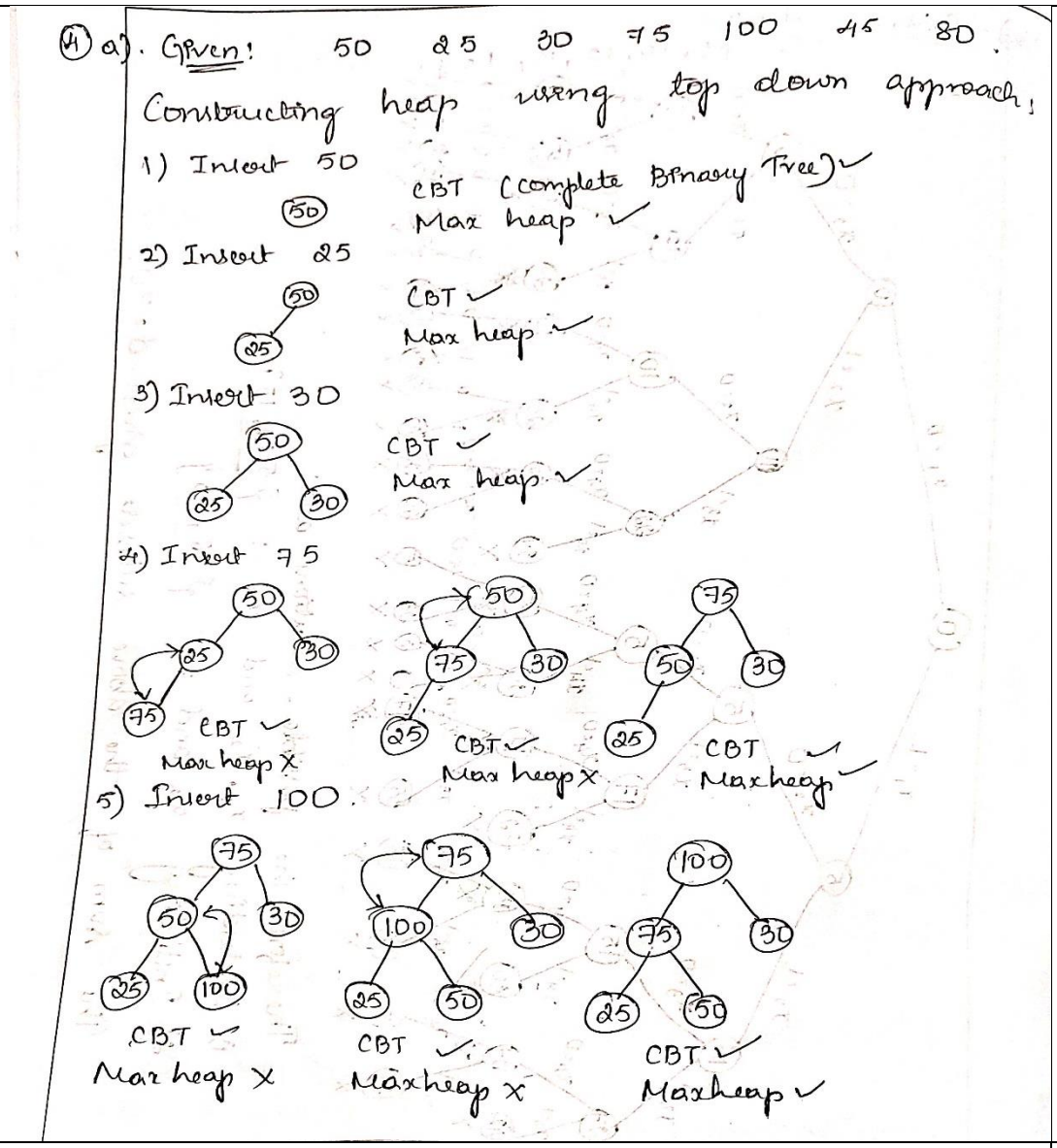


CBT ✓
Max heap ✗



CBT ✓
Max heap ✓

∴ Max heap is constructed.

4(b)	Design an algorithm to sort the elements using Sorting by Counting.
	
	ALGORITHM *DistributionCountingSort*$(A[0..n-1], l, u)$
	//Sorts an array of integers from a limited range by distribution counting
	//Input: An array $A[0..n-1]$ of integers between l and u $(l \le u)$
	//Output: Array $S[0..n-1]$ of A's elements sorted in nondecreasing order
	for $j \leftarrow 0$ **to** $u-l$ **do** $D[j] \leftarrow 0$ //initialize frequencies
	for $i \leftarrow 0$ **to** $n-1$ **do** $D[A[i]-l] \leftarrow D[A[i]-l]+1$ //compute frequencies
	for $j \leftarrow 1$ **to** $u-l$ **do** $D[j] \leftarrow D[j-1]+D[j]$ //reuse for distribution
	for $i \leftarrow n-1$ **downto** 0 **do**
	$j \leftarrow A[i]-l$
	$S[D[j]-1] \leftarrow A[i]$
	$D[j] \leftarrow D[j]-1$
	return S
		4	CO4	L2

5(a)	Apply the dynamic programming to the following instance of the Knapsack problem W=5	10	CO4	L3

Item	Weight	Value
1	2	12
2	1	10
3	3	20
4	2	15

$V[i,j] = \max\{V[i-1,j],\ v_i + V[i-1, j-w_i]\}$.

Also, if $j < w_i \Rightarrow V[i,j] = V[i-1,j]$.

$V[1,2] = \max\{V[0,2],\ 12 + V[0,0]\}$
$= \max\{0, 12+0\} = \underline{12}$

$V[1,3] = \max\{V[0,3],\ 12 + V[0,1]\} = \underline{12}$
$V[1,4] = \max\{V[0,4],\ 12 + V[0,2]\} = \underline{12}$
$V[1,5] = \max\{V[0,5],\ 12 + V[0,3]\} = \underline{12}$

$V[2,1] = \max\{V[1,1],\ 10 + V[1,0]\}$
$= \max\{0, 10+0\} = \underline{10}$
$V[2,2] = \max\{V[1,2], 10 + V[1,1]\} = \underline{12}$
$V[2,3] = \max\{V[1,3], 10 + V[1,2]\} = \underline{22}$
$V[2,4] = \max\{V[1,4], 10 + V[1,3]\} = \underline{22}$
$V[2,5] = \max\{V[1,5], 10 + V[1,4]\} = \underline{22}$

$V[3,3] = \max\{V[2,3],\ 20 + V[2,0]\}$
$= \max\{22, 20+0\} = \underline{22}$
$V[3,4] = \max\{V[2,4], 20 + V[2,1]\} = \underline{30}$
$V[3,5] = \max\{V[2,5], 20 + V[2,2]\} = \underline{32}$

$V[4,2] = \max\{V[3,2],\ 15 + V[3,0]\}$
$= \max\{12, 15+0\} = \underline{15}$
$V[4,3] = \max\{V[3,3], 15 + V[3,1]\} = \underline{25}$
$V[4,4] = \max\{V[3,4], 15 + V[3,2]\} = \underline{30}$
$V[4,5] = \max\{V[3,5], 15 + V[3,3]\} = \underline{37}$

x_1	x_2	x_3	x_4
1	1	0	1

$x_4 = 37 - 15 = \underline{22}$ $\quad x_2 = 10 - 10 = \underline{0}$
$x_1 = 22 - 12 = \underline{10}$

⑤ a). W = 5

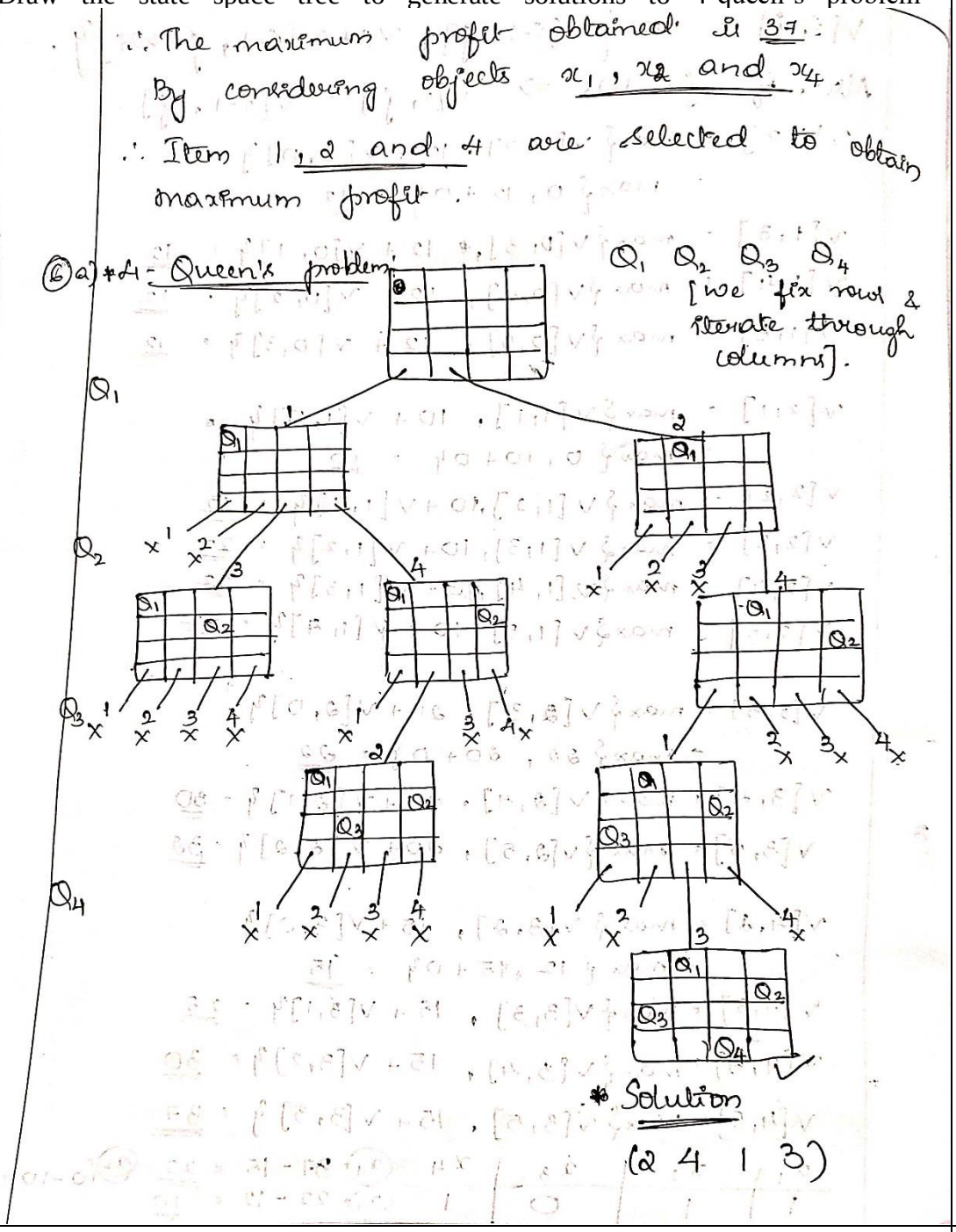
Item	weight	Value
1	2	12
2	1	10
3	3	20
4	2	15

Constructing the table for memorization in dynamic programming.

n \ W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	12	12	12	12
2	0	10	12	22	22	22
3	0	10	12	22	30	32
4	0	10	15	25	30	37

n ↓ W →

Formula for dynamic programming approach of knapsack is given by:

6 (a)	Draw the state space tree to generate solutions to 4-queen's problem  	6	CO4	L2
6(b)	Write a short note on the following a. Dynamic programming Dynamic programming is a technique that breaks the problems into sub-problems, and saves the result for future purposes so that we do not need to compute the result again. The sub problems are optimized to optimize the overall solution is known as optimal substructure property. The main use of dynamic programming is to solve optimization problems. Here, optimization problems mean that when we are trying to find out the minimum or the maximum solution of a problem. The dynamic	2+2	CO4 &CO 5	L2

programming guarantees to find the optimal solution of a problem if the solution exists.

The definition of dynamic programming says that it is a technique for solving a complex problem by first breaking into a collection of simpler sub problems, solving each sub problem just once, and then storing their solutions to avoid repetitive computations.

b. Back tracking

Backtracking is one of the techniques that can be used to solve the problem. We can write the algorithm using this strategy. It uses the Brute force search to solve the problem, and the brute force search says that for the given problem, we try to make all the possible solutions and pick out the best solution from all the desired solutions. This rule is also followed in dynamic programming, but dynamic programming is used for solving optimization problems. In contrast, backtracking is not used in solving optimization problems. Backtracking is used when we have multiple solutions, and we require all those solutions.

Backtracking name itself suggests that we are going back and coming forward; if it satisfies the condition, then return success, else we go back again. It is used to solve a problem in which a sequence of objects is chosen from a specified set so that the sequence satisfies some criteria.

When to use a Backtracking algorithm?

When we have multiple choices, then we make the decisions from the available choices. In the following cases, we need to use the backtracking algorithm:

- A piece of sufficient information is not available to make the best choice, so we use the backtracking strategy to try out all the possible solutions.
- Each decision leads to a new set of choices. Then again, we backtrack to make new decisions. In this case, we need to use the backtracking strategy.

CI **CCI** **HOD**

PO Mapping

CO-PO and CO-PSO Mapping																				
Course Outcomes		Blooms Level	Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Analyze the performance of the algorithms, state the efficiency using asymptotic notations, and analyze mathematically the complexity of the algorithm.	L2	M1	3	3	2	3	2	-	-	-	-	-	-	2	-	-	-	-
CO2	Apply divide and conquer approaches and decrease and conquer approaches in solving the problems and analyze the same	L3	M2	3	3	2	3	2	-	-	-	-	-	-	2	-	-	-	-
CO3	Apply the appropriate algorithmic design technique like the greedy method, transform and conquer approaches and compare the efficiency of algorithms to solve the given problem.	L3	M3	3	3	2	3	2	-	-	-	-	-	-	2	-	-	-	-
CO4	Apply and analyze dynamic programming approaches to solve some problems. and improve an algorithm's time efficiency by sacrificing space.	L3	M4	3	3	2	3	2	-	-	-	-	-	-	2	-	-	-	-
CO5	Apply and analyze backtracking, branch and bound methods to describe P, NP, and NP-complete problems.	L3	M5	3	2	2	3	2	-	-	-	-	-	-	2	-	-	-	-

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS

L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				