

Internal Assessment Test 2 – May 2025

Sub:	Analysis and Design of Algorithms					Sub Code:	BCS401	Branch:	CSE																		
Date:	26-05-2025	Duration:	90 mins	Max Marks:	50	Sem / Sec:	IV (A, B & C)		OBE																		
<u>Answer any FIVE FULL Questions</u>								MARKS	CO	RBT																	
1(a)	Define AVL tree. Construct an AVL tree of the list of keys: 3, 6, 5, 1, 2, 4 indicating each step of key insertion and rotation.						5	CO3	L2																		
1(b)	Consider the problem of searching for genes in DNA sequences using Horspool’s algorithm. A DNA sequence consists of a text on the alphabet {A, C, G, T} and the gene or gene segment is the pattern. a. Construct the shift table for the following gene segment of your chromosome 10: TCCTATTCTT b. Apply Horspool’s algorithm to locate the above pattern in the following DNA sequence: TTATAGATCTCGTATTCTTTTATAGATCTCCTATTCTT						5	CO3	L3																		
2(a)	Define Heap. Constructing a heap for the list 1, 8, 6, 5, 3, 7, 4 by the bottom-up algorithm						5	CO3	L2																		
2(b)	Construct a 2-3 tree for the list C, O, M, P, U, T, I, N, G. Use the alphabetical order of the letters and insert them successively starting with the empty tree.						5	CO3	L3																		
3(a)	Define transitive closure. Compute transitive closure for the following directed graph. $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$						5	CO4	L3																		
3(b)	Construct a Huffman Code. We are given the following characters and probabilities: <table border="1"><tr><td>Character</td><td>A</td><td>B</td><td>C</td><td>D</td><td>–</td></tr><tr><td>Probability</td><td>0.4</td><td>0.1</td><td>0.2</td><td>0.15</td><td>0.15</td></tr></table> Encode the text ABACABAD using the code						Character	A	B	C	D	–	Probability	0.4	0.1	0.2	0.15	0.15	5	CO4	L3						
Character	A	B	C	D	–																						
Probability	0.4	0.1	0.2	0.15	0.15																						
4 (a)	Apply the bottom-up dynamic programming algorithm to the following instance of the knapsack problem with capacity =6: <table border="1"><tr><td>Item</td><td>Weight</td><td>Value</td></tr><tr><td>1</td><td>3</td><td>\$25</td></tr><tr><td>2</td><td>2</td><td>\$20</td></tr><tr><td>3</td><td>1</td><td>\$15</td></tr><tr><td>4</td><td>4</td><td>\$40</td></tr><tr><td>5</td><td>5</td><td>\$50</td></tr></table>						Item	Weight	Value	1	3	\$25	2	2	\$20	3	1	\$15	4	4	\$40	5	5	\$50	5	CO5	L2
Item	Weight	Value																									
1	3	\$25																									
2	2	\$20																									
3	1	\$15																									
4	4	\$40																									
5	5	\$50																									

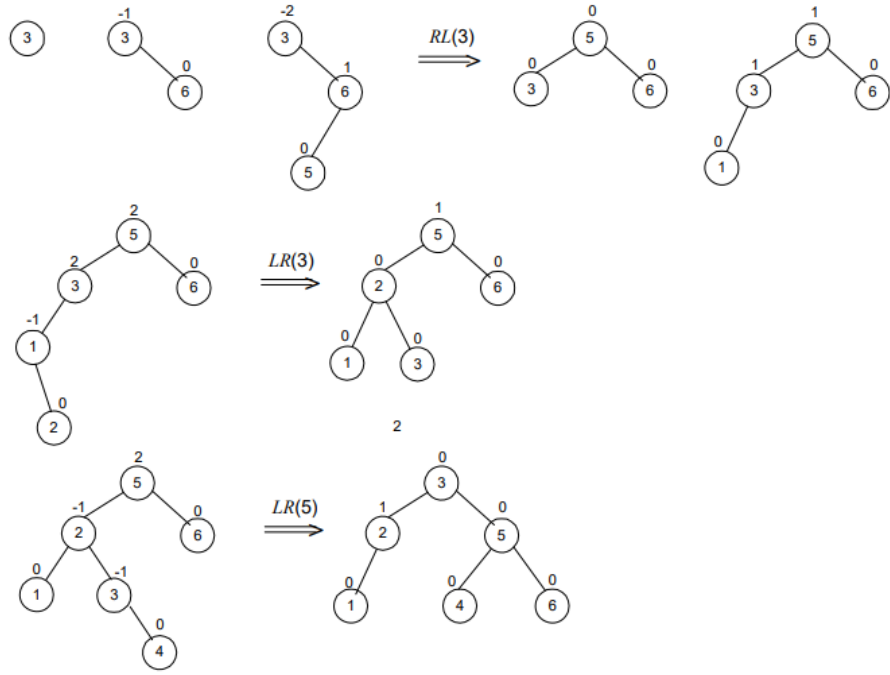
4 (b)	Explain Dynamic Programming using the Memory Function approach. Write the algorithm for the 0/1 Knapsack problem using the Memory Function method.	5	CO5	L2															
5(a)	Apply the greedy method to obtain optimal solution to continuous and discrete the knapsack problem given capacity, M=60. Weight, W={5,10,20,30,40} and Profit P={30,20,100,90,60} Find the total profit earned	5	CO5	L3															
5(b)	Solve the following instance of the knapsack problem by the branch-and-bound algorithm with capacity W=16 <table border="1"><thead><tr><th>Item</th><th>Weight</th><th>Value</th></tr></thead><tbody><tr><td>1</td><td>10</td><td>\$100</td></tr><tr><td>2</td><td>7</td><td>\$63</td></tr><tr><td>3</td><td>8</td><td>\$56</td></tr><tr><td>4</td><td>4</td><td>\$12</td></tr></tbody></table>	Item	Weight	Value	1	10	\$100	2	7	\$63	3	8	\$56	4	4	\$12	5	CO5	L3
Item	Weight	Value																	
1	10	\$100																	
2	7	\$63																	
3	8	\$56																	
4	4	\$12																	
6(a)	Draw the state space tree to generate solutions to 4-queen's problem.	5	CO5	L2															
6(b)	Apply backtracking to solve the following instance of the subset-sum problem: A = {1, 3, 4, 5} and d = 11 .	5	CO5	L3															

CI

CCI

HOD

Internal Assessment Test 2 – May 2025
SOLUTIONS

Sub:	Analysis and Design of Algorithms	Sub Code:	BCS401	Branch:	CSE	MARKS	C O	RB T
1(a)	<p>Define AVL tree. Construct an AVL tree of the list of keys: 3, 6, 5, 1, 2, 4 indicating each step of key insertion and rotation.</p> <p>An AVL tree is a self-balancing binary search tree (BST) where the difference in height between the left and right subtrees (called the balance factor) of any node is at most 1.</p> <p>Key Properties:</p> <ul style="list-style-type: none">For every node in the tree: Balance Factor = Height of Left Subtree - Height of Right Subtree and it must be -1, 0, or +1.Whenever an insertion or deletion operation causes the balance factor to go outside this range, the tree performs rotations (single or double) to restore balance. 	5	CO3	L2				
1(b)	<p>Consider the problem of searching for genes in DNA sequences using Horspool's algorithm. A DNA sequence consists of a text on the alphabet {A, C, G, T} and the gene or gene segment is the pattern.</p> <p>a. Construct the shift table for the following gene segment of your chromosome 10: TCCTATTCTT</p>	5	CO3	L3				

	<p>b. Apply Horspool's algorithm to locate the above pattern in the following DNA sequence: TTATAGATCTCGTATTCTTTTATAGATCTCCTATTCTT</p> <p>Solution:</p> <p>a. For the pattern TCCTATTCTT and the alphabet {A, C, G, T}, the shift table looks as follows:</p> <table border="1"> <tr> <td>c</td><td>A</td><td>C</td><td>G</td><td>T</td></tr> <tr> <td>$t(c)$</td><td>5</td><td>2</td><td>10</td><td>1</td></tr> </table>	c	A	C	G	T	$t(c)$	5	2	10	1			
c	A	C	G	T										
$t(c)$	5	2	10	1										
2(a)	<p>Define Heap. Constructing a heap for the list 1, 8, 6, 5, 3, 7, 4 by the bottom-up algorithm</p> <p>A heap is a complete binary tree that satisfies the heap property.</p> <p>Types of Heap:</p> <p>Max-Heap: In a max-heap, the value of each parent node is greater than or equal to the values of its children. -The largest element is at the root.</p> <p>Min-Heap: In a min-heap, the value of each parent node is less than or equal to the values of its children. The smallest element is at the root.</p> <p>algorithm (a root of a subtree being heapified is shown in bold):</p> <p>1 8 6 5 3 7 4 \Rightarrow 1 8 7 5 3 6 4</p> <p>1 8 7 5 3 6 4</p> <p>1 8 7 5 3 6 4 \Rightarrow 8 5 7 1 3 6 4</p>	5	CO3	L2										
2(b)	<p>Construct a 2-3 tree for the list C, O, M, P, U, T, I, N, G. Use the alphabetical order of the letters and insert them successively starting with the empty tree.</p>	5	CO3	L3										

3(a)	<p>Define transitive closure. Compute transitive closure for the following directed graph.</p> <p>Transitive Closure Definition:</p> <p>The transitive closure of a relation R on a set is the smallest transitive relation that contains R.</p> <p>For a directed graph $G = (V, E)$, the transitive closure of G is a graph $G^+ = (V, E^+)$ such that for every pair of vertices (u, v):</p> <ul style="list-style-type: none"> $(u, v) \in E^+$ if and only if there is a path from u to v in G. $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	5	CO4	L3

$$R^{(0)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^{(1)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^{(2)} = \begin{bmatrix} 0 & 1 & \mathbf{1} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

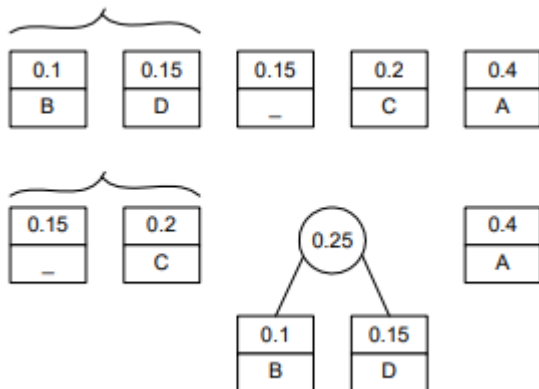
$$R^{(3)} = \begin{bmatrix} 0 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 1 & \mathbf{1} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^{(4)} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = T$$

3(b) a. Construct a Huffman Code. We are given the following characters and probabilities:

Character	A	B	C	D	–
Probability	0.4	0.1	0.2	0.15	0.15

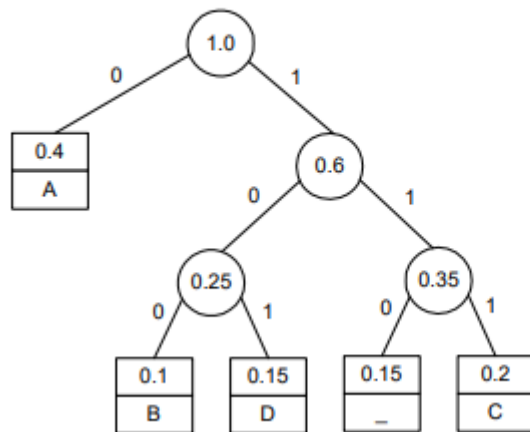
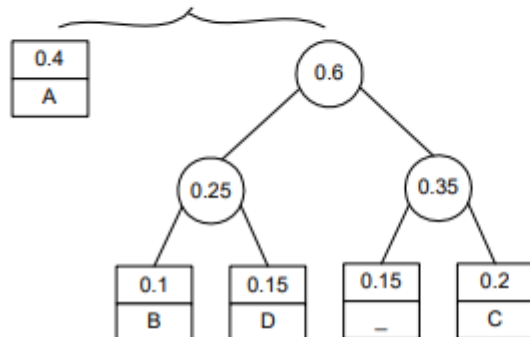
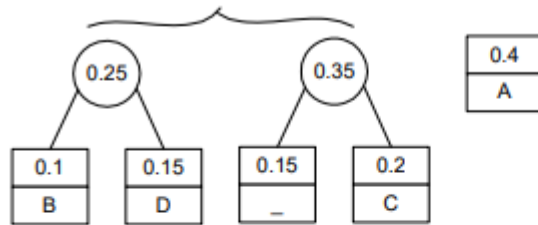
Encode the text ABACABAD using the code



5

CO4

L3



character	A	B	C	D	
probability	0.4	0.1	0.2	0.15	0.15
codeword	0	100	111	101	110

b. The text ABACABAD will be encoded as 0100011101000101.

4 (a) Apply the bottom-up dynamic programming algorithm to the following instance of the knapsack problem with capacity =6:

Item	Weight	Value
1	3	\$25
2	2	\$20
3	1	\$15
4	4	\$40
5	5	\$50

Solution:

5

CO5

L2

	<div>1. a.</div> <table><tr><td></td><td></td><td colspan="7">capacity j</td></tr><tr><td></td><td>i</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>$w_1 = 3, v_1 = 25$</td><td>1</td><td>0</td><td>0</td><td>0</td><td>25</td><td>25</td><td>25</td><td>25</td></tr><tr><td>$w_2 = 2, v_2 = 20$</td><td>2</td><td>0</td><td>0</td><td>20</td><td>25</td><td>25</td><td>45</td><td>45</td></tr><tr><td>$w_3 = 1, v_3 = 15$</td><td>3</td><td>0</td><td>15</td><td>20</td><td>35</td><td>40</td><td>45</td><td>60</td></tr><tr><td>$w_4 = 4, v_4 = 40$</td><td>4</td><td>0</td><td>15</td><td>20</td><td>35</td><td>40</td><td>55</td><td>60</td></tr><tr><td>$w_5 = 5, v_5 = 50$</td><td>5</td><td>0</td><td>15</td><td>20</td><td>35</td><td>40</td><td>55</td><td>65</td></tr></table> <div>The maximal value of a feasible subset is $F[5, 6] = 65$. The optimal subset is {item 3, item 5}.</div>			capacity j								i	0	1	2	3	4	5	6		0	0	0	0	0	0	0	0	$w_1 = 3, v_1 = 25$	1	0	0	0	25	25	25	25	$w_2 = 2, v_2 = 20$	2	0	0	20	25	25	45	45	$w_3 = 1, v_3 = 15$	3	0	15	20	35	40	45	60	$w_4 = 4, v_4 = 40$	4	0	15	20	35	40	55	60	$w_5 = 5, v_5 = 50$	5	0	15	20	35	40	55	65			
		capacity j																																																																										
	i	0	1	2	3	4	5	6																																																																				
	0	0	0	0	0	0	0	0																																																																				
$w_1 = 3, v_1 = 25$	1	0	0	0	25	25	25	25																																																																				
$w_2 = 2, v_2 = 20$	2	0	0	20	25	25	45	45																																																																				
$w_3 = 1, v_3 = 15$	3	0	15	20	35	40	45	60																																																																				
$w_4 = 4, v_4 = 40$	4	0	15	20	35	40	55	60																																																																				
$w_5 = 5, v_5 = 50$	5	0	15	20	35	40	55	65																																																																				
4 (b)	<div>Explain Dynamic Programming using the Memory Function approach. Write the algorithm for the 0/1 Knapsack problem using the Memory Function method.</div> <div>Problem with Direct Top-Down Approach:</div> <div><ul style="list-style-type: none">• Solves the same subproblems multiple times, especially in recursive solutions.• Leads to inefficient algorithms—often exponential time complexity.<div>◇ Classic Bottom-Up Dynamic Programming:</div><div><ul style="list-style-type: none">• Fills a table from smallest to largest subproblem.• Every subproblem is solved once.• Drawback: All subproblems are solved, even if some are not needed for the final result.<div>◇ Combining Strengths: Memoization (Top-Down + Table)</div><div><ul style="list-style-type: none">• Aims to solve only the necessary subproblems, and only once.• Uses a top-down recursive approach (like naive recursion).• Adds a memory table (like in bottom-up DP).<div>◇ How It Works:</div><div><div>1. A table (cache) is created and initialized with a special value (e.g., null or -1) indicating “not yet computed.”</div><div>2. When a recursive function is called:</div><div><ul style="list-style-type: none">○ It first checks if the solution is already in the table.○ If yes, it returns the cached result.○ If no, it computes the value recursively, stores it in the table, and returns it.</div><div>◇ Benefits:</div><div><ul style="list-style-type: none">• Avoids redundant calculations.• Saves time compared to naive recursion.• More space-efficient than bottom-up DP in some cases (solves only needed subproblems).</div></div></div></div></div>	5	CO5	L2																																																																								

ALGORITHM *MFKnapsack*(*i*, *j*)

//Implements the memory function method for the knapsack problem

//Input: A nonnegative integer *i* indicating the number of the first// items being considered and a nonnegative integer *j* indicating

// the knapsack capacity

//Output: The value of an optimal feasible subset of the first *i* items//Note: Uses as global variables input arrays *Weights*[1..*n*], *Values*[1..*n*],//and table *F*[0..*n*, 0..*W*] whose entries are initialized with -1's except for

//row 0 and column 0 initialized with 0's

if *F*[*i*, *j*] < 0 if *j* < *Weights*[*i*] value ← *MFKnapsack*(*i* - 1, *j*)

else

 value ← max(*MFKnapsack*(*i* - 1, *j*), *Values*[*i*] + *MFKnapsack*(*i* - 1, *j* - *Weights*[*i*])) *F*[*i*, *j*] ← valuereturn *F*[*i*, *j*]

- 5(a) Apply the greedy method to obtain optimal solution to continuous and discrete the knapsack problem given capacity, *M*=60. Weight, *W*={5,10,20,30,40} and Profit *P*={30,20,100,90,60} Find the total profit earned

5

CO5

L3

Sol: Greedy method (Knapsack problem)

M = 60

Item	Weight	Profit	$\frac{V_i}{W_i}$
1	5	30	6
2	10	20	2
3	20	100	5
4	30	90	3
5	40	60	1.5

Arrange in descending order based on $\frac{V_i}{W_i}$ value

Item	Weight	Profit	$\frac{V_i}{W_i}$
5	30	90	3
3	20	100	5
1	5	30	6
2	10	20	2
4	40	60	1.5

Discrete knapsack problem =

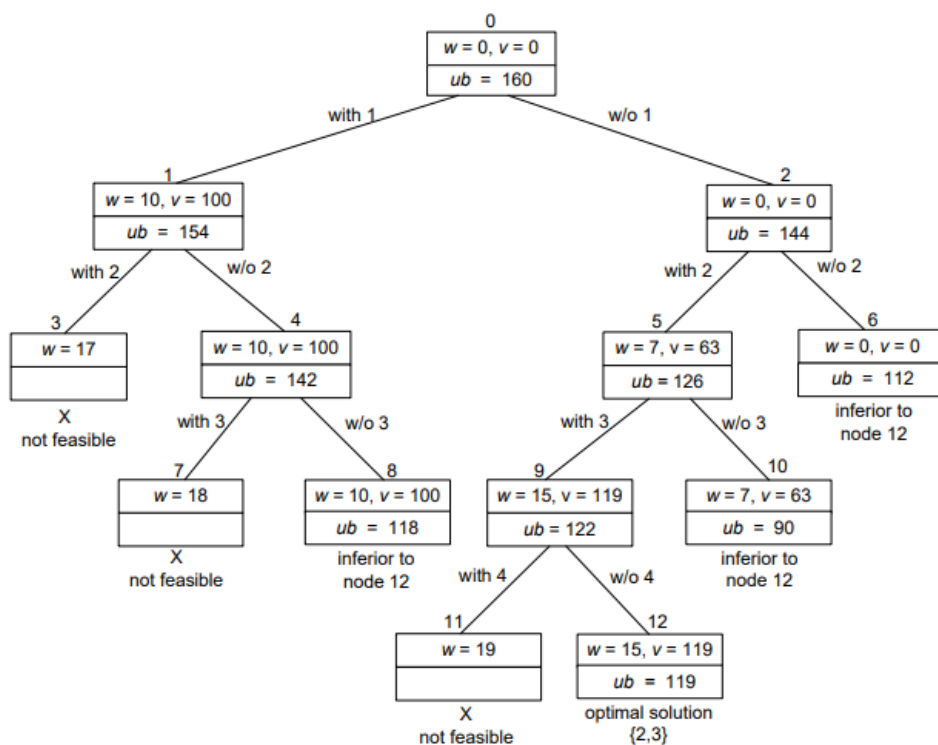
$$30 + 100 + 90 = 220. \text{ (profit)}$$

Continuous knapsack problem.

$$30 + 100 + 90 + 10 = 230 \text{ (profit)}$$

5(b) Solve the following instance of the knapsack problem by the branch-and-bound algorithm with capacity $W=16$

Item	Weight	Value
1	10	\$100
2	7	\$63
3	8	\$56
4	4	\$12



The found optimal solution is {item 2, item 3} of value \$119.

6(a) Draw the state space tree to generate solutions to 4-queen's problem.

5

CO5

L2

6(b)	<p>Apply backtracking to solve the following instance of the subset-sum problem: $A = \{1, 3, 4, 5\}$ and $d = 11$.</p> <div data-bbox="236 208 1000 609"> </div> <p>There is no solution to this instance of the problem.</p>	5	CO5	L3
------	--	---	-----	----

CI

CCI

HOD