SRM Institute of Science and Technology
College of Engineering and Technology
School of Computing
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu
Academic Year: 2023-24 (EVEN)

SET D

Test: CLA-2 T3 **Date:30 /04/2024**
Course Code & Title: 21CSC204J Design and Analysis of Algorithms **Duration:** 1 hour 40 min
Year & Sem: II Year / IV Sem **Max. Marks:** 50

Course Articulation Matrix:

Course Outcome	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	Program Specific Outcomes		
													PSO-1	PSO-2	PSO-3
CO1	2	1	2	1	-	-	-	-		3	-	3	3	1	-
CO2	2	1	2	1	-	-	-	-		3	-	3	3	1	-
CO3	2	1	2	1	-	-	-	-		3	-	3	3	1	-
CO4	2	1	2	1	-	-	-	-		3	-	3	3	1	-
CO5	2	1	2	1	-	-	-	-		3	-	3	3	1	-

Part – A (8 x 1 = 8 Marks) Instructions: Answer all						
Q. No	**Question**	**Marks**	**BL**	**CO**	**PO**	**PI Code**
1	Which of the following is false about the Kruskal's algorithm? a) It constructs MST by selecting edges in increasing order of their weights b) It is a greedy algorithm c) It uses union-find data structure d) It can accept cycles in the MST	1	L1	3	2	2.5.2
2	If an optimal solution can be created for a problem by constructing optimal solutions for its subproblems, the problem possesses _____ property. a) Overlapping subproblems b) Optimal substructure c) Memoization d) Greedy	1	L1	3	2	2.5.2
3	When a top-down approach of dynamic programming is applied to a problem, it usually _____ a) Decreases both, the time complexity, and the space complexity b) Decreases the time complexity and increases the space complexity c) Increases the time complexity and decreases the space complexity d) Increases both, the time complexity, and the space complexity	1	L1	3	2	2.5.2
4	Consider a scenario where you are implementing a Sudoku-solving algorithm using backtracking. You have a partially completed Sudoku grid and need to fill in the empty cells such that every row, column, and 3x3 sub grid contains all the digits from 1 to 9. Which of the following best describes the role of	1	L2	4	2	2.5.2

	backtracking in this algorithm? **a) Backtracking is used to randomly fill in the empty cells until a solution is found.** **b) Backtracking is employed to check each cell in the grid sequentially until a valid solution is reached.** **c) Backtracking is used to systematically explore possible solutions, backtracking from choices that prove to be invalid** **d) Backtracking is applied to optimize the search process by considering heuristic techniques to prioritize cell filling.**					
5	**In which of the following scenarios would the Branch and Bound algorithm be most appropriate?** **a) Sorting a list of integers in ascending order.** **b) Finding the shortest path between two vertices in a graph.** **c) Solving the Knapsack problem to maximize the value of items without exceeding the weight limit** **d) Implementing a hash table for fast key-value lookups.**	**1**	**L1**	**4**	**2**	**2.5.2**
6	**In the 0/1 Knapsack Problem, what is the significance of the "knapsack capacity"?** **a) It represents the total number of items available for selection.** **b) It represents the maximum value that can be obtained by selecting items.** **c) It represents the maximum weight that the knapsack can hold.** **d) It represents the total value of the items available for selection.**	**1**	**L1**	**4**	**2**	**2.5.2**
7	**Let X be a problem that belongs to class NP. Then which one of the following is TRUE?** **a) There is no polynomial time algorithm for X.** **b) If X can be solved deterministically in polynomial time, then P = NP.** **c) If X is NP-hard, then it is NP-complete.** **d) X must be undecidable.**	**1**	**L2**	**5**	**2**	**2.5.2**
8	**Which of the following statements are TRUE?** **(1) The problem of determining whether there exists a cycle in an undirected graph is in P.** **(2) The problem of determining whether there exists a cycle in an undirected graph is in NP.** **(3) If problem A is NP-Complete, there exists a non-deterministic polynomial time algorithm to solve A.** **a) 1, 2 and 3** **b) 1 and 3** **c) 2 and 3** **d) 1 and 2**	**1**	**L2**	**5**	**2**	**2.5.2**

SET D

SRM Institute of Science and Technology
College of Engineering and Technology
School of Computing
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu

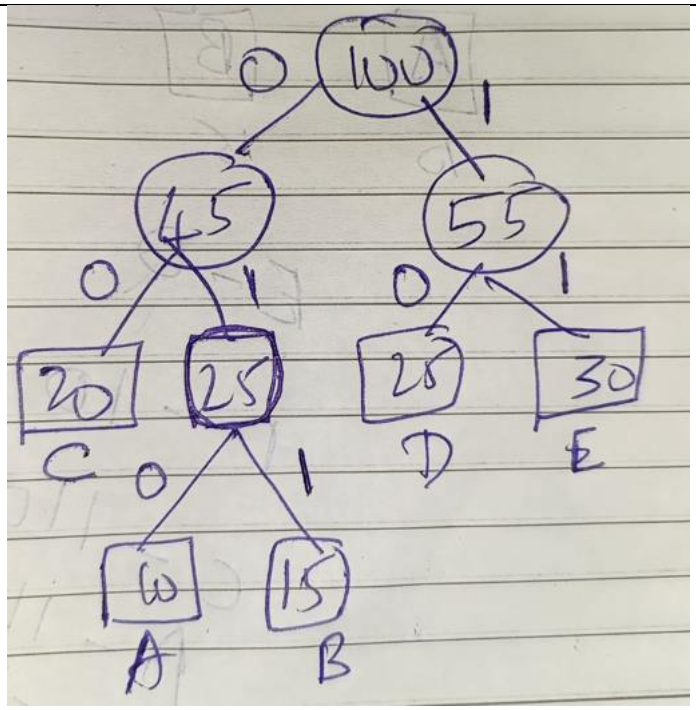
Academic Year: 2023-24 (EVEN)

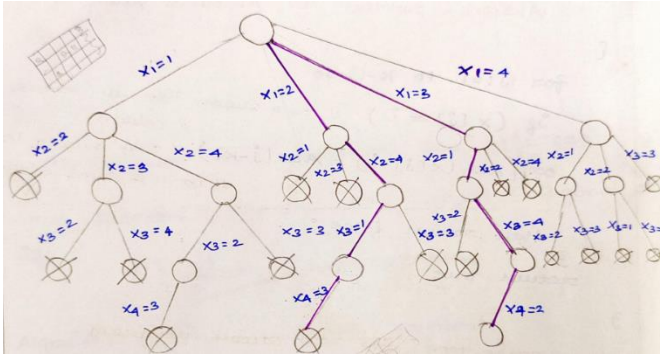
Test: CLA-2 T3 **Date: 30/04/2024**
Course Code & Title: 21CSC204J Design and Analysis of Algorithms **Duration:** 1 hour 40 min
Year & Sem: II Year / IV Sem **Max. Marks:** 50

Course Articulation Matrix:

Course Outcome	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	Program Specific Outcomes		
													PSO-1	PSO-2	PSO-3
CO1	*2*	*1*	*2*	*1*	-	-	-	-		*3*	-	*3*	*3*	*1*	-
CO2	*2*	*1*	*2*	*1*	-	-	-	-		*3*	-	*3*	*3*	*1*	-
CO3	*2*	*1*	*2*	*1*	-	-	-	-		*3*	-	*3*	*3*	*1*	-
CO4	*2*	*1*	*2*	*1*	-	-	-	-		*3*	-	*3*	*3*	*1*	-
CO5	*2*	*1*	*2*	*1*	-	-	-	-		*3*	-	*3*	*3*	*1*	-

Part – B (3 x 5 = 15 Marks) Instructions: Answer all						
9	**You are a data compression engineer working for a video streaming service. Your task is to compress a set of video files efficiently to minimize bandwidth usage and storage requirements. Apply a greedy based compression technique, known for its effectiveness in compressing data with varying frequencies.** **Given the following table showing the frequencies of characters in a sample video file:** **Character Frequency** **A 10** **B 15** **C 20** **D 25** **E 30** **a) Using the greedy method, construct a Huffman tree for encoding the characters based on their frequencies.** **b) Provide the Huffman codes for each character.**	**5**	**L2**	**3**	**2**	**2.6.3**

	 **A – 010** **B – 011** **C – 00** **D – 10** **E – 11**					

10	**Apply a Backtracking strategy to place 4 queens on a 4x4 chessboard such that no two queens threaten each other and evaluate its computational Complexity.**
	Algorithm Place (k, i)
	{
	For j ← 1 to k - 1
	do if (x [j] = i)
	or (Abs x [j]) - i) = (Abs (j - k))
	then return false;
	return true;
	}
	Algorithm NQueens (k, n)
	{
	For i ← 1 to n do
	if Place (k, i) then {
	x [k] ← i;
	if (k ==n) then
	write (x [1....n));
	else
	NQueens (k + 1, n);
	}
	}
	 
	Time Complexity: O(n!)
		5	**L3**	**4**	**2**	**2.6.3**

11	**You are tasked with developing a sorting algorithm for a social media platform to display posts in users' feeds. The posts are continually updated and need to be sorted by their timestamp to ensure the most recent ones appear at the top. However, due to the dynamic nature of the platform, the timestamps can occasionally be identical. How would you utilize the randomized quicksort algorithm to efficiently sort these posts, ensuring that the ordering remains stable, and that the algorithm performs well even in the presence of identical timestamps?** **To utilize the randomized quicksort algorithm to efficiently sort posts by their timestamps.** RANDOMIZED-PARTITION(A, p, r) 1 i = RANDOM(p, r) 2 exchange $A[r]$ with $A[i]$ 3 **return** PARTITION(A, p, r) RANDOMIZED-QUICKSORT(A, p, r) 1 **if** $p < r$ 2 q = RANDOMIZED-PARTITION(A, p, r) 3 RANDOMIZED-QUICKSORT$(A, p, q-1)$ 4 RANDOMIZED-QUICKSORT$(A, q+1, r)$ **Choose a Random Pivot: Instead of always selecting the first or last element as the pivot, randomly select a pivot element from the array. This helps prevent worst-case scenarios where the algorithm's performance degrades due to sorted or almost sorted input.**	**5**	**L2**	**5**	**2**	**2.5.2**
	Part – C (3 x 9 = 27 Marks)					
12. A	**A company is optimizing its production process and needs to determine the best combination of components to include in a product, considering both their cost and effectiveness. How could Dynamic programming approach be applied to find the optimal combination of components, ensuring maximum performance within a budget constraint for its effective computation complexity?** **Consider the following items available for you to pack:** **Components Weight (kg) Value (Rs)** **A 2 10** **B 3 15** **C 5 20** **D 7 25** **E 8 30** **Assume your product has a weight capacity of 10 kg.** **a. Solve the 0/1 knapsack problem using dynamic programming to determine the maximum total value of items you can carry in your backpack without exceeding its weight capacity.** **b. Provide the list of items you should pack to achieve this maximum total value.**	**9**	**L2**	**3**	**2**	**2.6.3**

Result: $M[n, W]$

$M[0, w] \leftarrow 0, \forall w\ 0\ to\ W$

$M[i, 0] \leftarrow 0, \forall i\ 0\ to\ n$

for $i \leftarrow 1\ to\ n$ do

  for $w \leftarrow 1\ to\ W$ do

    if $w_i \leq w$ then

      $M[i, w] = max(M[i-1, w-w_i] + v_i,\ M[i-1, w])$

    else

      $M[i, w] = M[i-1, w]$

    end

  end

end

return $M[n, W]$

Weights: 0 1 2 3 4 5 6 7 8 9 10

0	0	0	0	0	0	0	0	0	0	0	0
A 1	0	0	10	10	10	10	10	10	10	10	10
B 2	0	0	10	15	15	25	25	25	25	25	25
C 3	0	0	10	15	15	25	25	30	35	35	45
D 4	0	0	10	15	25	25	25	30	35	35	45
E 5	0	0	10	15	25	25	25	30	35	35	45

List of items- (A, B, C)

(5,10) →(4,10) → (3,10) →(2,5) → (1,2) →(0,0)

(or)

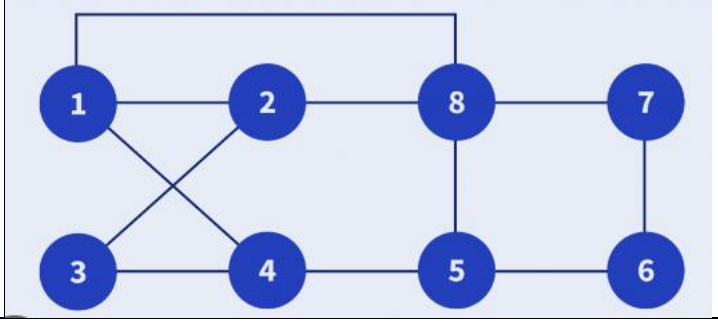
12. B	**You are working on a plagiarism detection system for a university's computer science department. Your task is to develop an algorithm that can efficiently identify similarities between pairs of documents submitted by students. Apply a dynamic programming approach to find the longest common sequence of characters between two documents.** **Consider two documents represented as strings:** **Document 1: "algorithm"** **Document 2: "logarithm"** **Using dynamic programming, determine the length of the longest common subsequence between the two documents.**	**9**	**L2**	**3**	**2**	**2.6.3**

Pseudocode for LCS

```
LCS-LENGTH(X, Y)
 1  m = X.length
 2  n = Y.length
 3  let b[1..m, 1..n] and c[0..m, 0..n] be new tables
 4  for i = 1 to m
 5      c[i, 0] = 0
 6  for j = 0 to n
 7      c[0, j] = 0
 8  for i = 1 to m
 9      for j = 1 to n
10          if x_i == y_j
11              c[i, j] = c[i − 1, j − 1] + 1
12              b[i, j] = "↖"
13          elseif c[i − 1, j] ≥ c[i, j − 1]
14              c[i, j] = c[i − 1, j]
15              b[i, j] = "↑"
16          else c[i, j] = c[i, j − 1]
17              b[i, j] = "←"
18  return c and b
```

		l	o	g	a	r	i	t	h	m
	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	1	1	1	1	1	1
l	0	1	1	1	1	1	1	1	1	1
g	0	1	1	2	2	2	2	2	2	2
o	0	1	2	2	2	2	2	2	2	2
r	0	1	2	2	2	3	3	3	3	3
i	0	1	2	2	2	3	4	4	4	4
t	0	1	2	2	2	3	4	5	5	5
h	0	1	2	2	2	3	4	5	6	6
m	0	1	1	2	2	3	4	5	6	7

LCS = 7

13. A	**You are a computer scientist working on a project to optimize delivery routes for a logistics company. One of the challenges you face is finding the most efficient way for delivery trucks to visit a set of locations without visiting any location more than once. Implement a backtracking algorithm to find a cycle that visits each vertex exactly once in a graph. Your task is to find all possible Hamiltonian circuits for a given graph representing the locations.** 	**9**	**L2**	**4**	**2**	**2.6.3**

```
1   Algorithm Hamiltonian(k)
2   // This algorithm uses the recursive formulation of
3   // backtracking to find all the Hamiltonian cycles
4   // of a graph. The graph is stored as an adjacency
5   // matrix G[1 : n, 1 : n]. All cycles begin at node 1.
6   {
7       repeat
8       { // Generate values for x[k].
9           NextValue(k); // Assign a legal next value to x[k].
10          if (x[k] = 0) then return;
11          if (k = n) then write (x[1 : n]);
12          else Hamiltonian(k + 1);
13      } until (false);
14  }

Algorithm NextValue(k)
// x[1 : k − 1] is a path of k − 1 distinct vertices. If x[k] = 0, then
// no vertex has as yet been assigned to x[k]. After execution,
// x[k] is assigned to the next highest numbered vertex which
// does not already appear in x[1 : k − 1] and is connected by
// an edge to x[k − 1]. Otherwise x[k] = 0. If k = n, then
// in addition x[k] is connected to x[1].
{
    repeat
    {
        x[k] := (x[k] + 1) mod (n + 1); // Next vertex.
        if (x[k] = 0) then return;
        if (G[x[k − 1], x[k]] ≠ 0) then
        { // Is there an edge?
            for j := 1 to k − 1 do if (x[j] = x[k]) then break;
                                    // Check for distinctness.
            if (j = k) then // If true, then the vertex is distinct.
                if ((k < n) or ((k = n) and G[x[n], x[1]] ≠ 0))
                    then return;
        }
    } until (false);
}
```

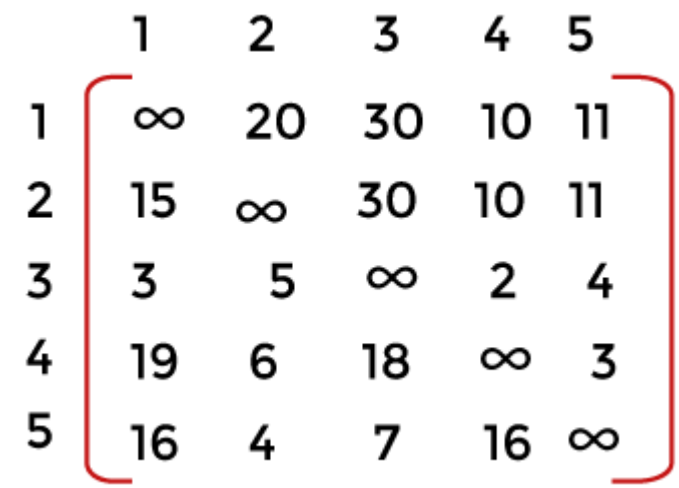
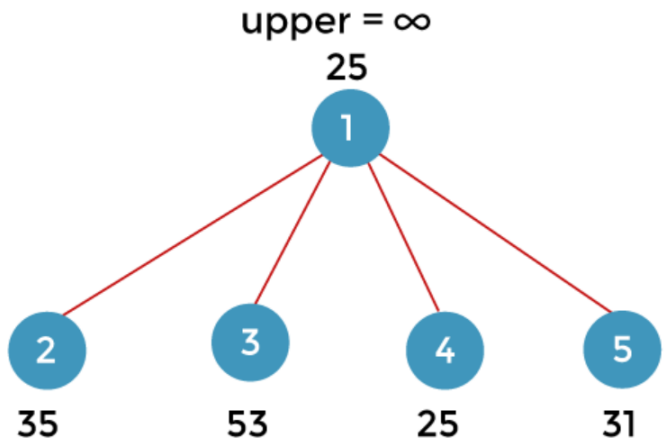
Hamiltonian Circuit

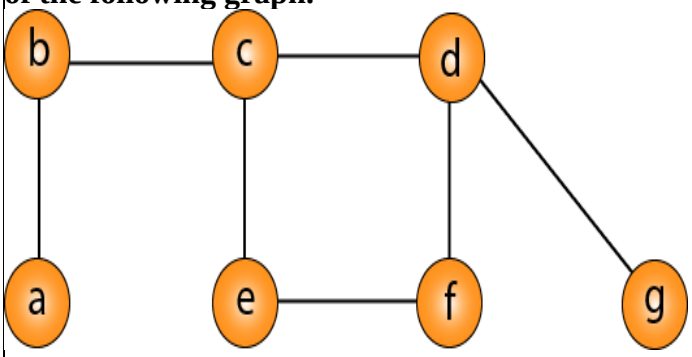
1 – 2 - 3 – 4 - 5 - 6 -7 -8 - 1

1 – 8 - 7 – 6 - 5 - 4 -3 -2 - 1

(or)

13. B	**Imagine you're a logistics manager for a distribution company responsible for planning delivery routes for your fleet of trucks. Your company operates in a city with multiple warehouses and delivery locations. Your goal is to minimize the total distance traveled by your trucks while ensuring that each delivery location is visited exactly once. Using the branch and bound algorithm, outline the steps you would take to find the optimal delivery route for your trucks. Consider factors such as the number of delivery locations, the distances between them, and how you would efficiently explore the solution space to find the best route. Additionally, discuss any strategies you would employ to reduce the computational complexity of the problem and speed up the solution process.**	**9**	**L2**	**4**	**2**	**2.6.3**

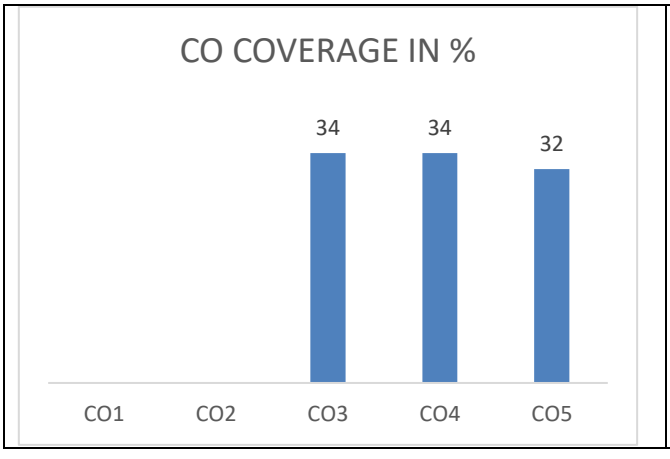
	  **Since node 4 is minimal, perform** **1→4→2 = 28** **1→4→3 = 50** **1→4→5 = 36** **Since node 2 is minimal, perform** **1→4→2→3 = 52** **1→4→2→5 = 28** **The optimal tour of the travelling Salesperson is defined by,** **1->4->2->5->3**					
14. A	**Describe about the class P and Class NP problems with suitable examples.** **Class P Problems:** **P stands for "Polynomial Time." Problems in class P are those for which there exists an algorithm that can solve them in polynomial time, meaning the time taken by the algorithm is bounded by a polynomial function of the input size.** **In simpler terms, P problems are those that can be solved efficiently with algorithms whose running time grows polynomially with the size of the input.** **Examples of problems in class P include:** **Sorting an array of integers (e.g., using algorithms like merge sort, quicksort) - O (n log n) - time complexity.** **Finding the shortest path in a graph with non-negative**	**9**	**L2**	**5**	**2**	**2.5.2**

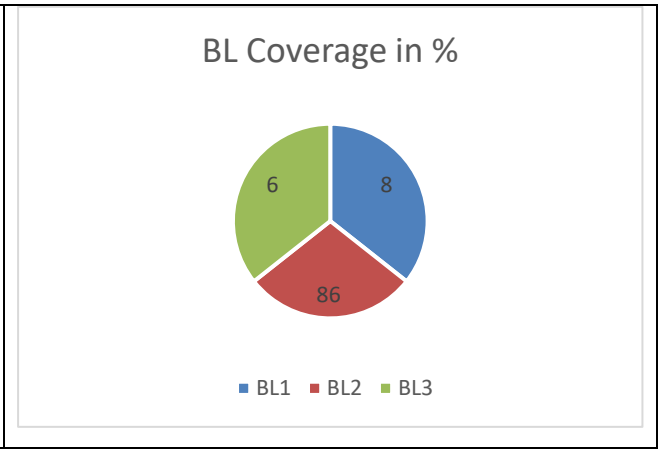
	edge weights (e.g., Dijkstra's algorithm) - O(V^2) or O (E log V) time complexity. **Strassen Matrix Multiplication - O(n^2.7) time complexity.** **NP stands for "Nondeterministic Polynomial Time." Problems in class NP are those that cannot be solved in polynomial time, for which a given solution can be verified in polynomial time.** **Examples of problems in class NP include:** **The Traveling Salesman Problem (TSP) - Given a list of cities and the distances between each pair of cities, find the shortest possible route that visits each city exactly once and returns to the original city.** **The Boolean Satisfiability Problem (SAT) - Given a Boolean formula, is there an assignment of truth values to the variables that makes the formula evaluate to true?** **The Subset Sum Problem - Given a set of integers and a target sum, can any subset of the integers sum to the target sum?** **The Knapsack Problem - Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.** **Graph Coloring- The realm of NP (Nondeterministic Polynomial Time) problems for finding the chromatic number (minimum number of colors needed to color a graph) is in the class of NP problems.** **Any one NP problem can be solved with an example.**					
		(or)				
14. B	**Explain Vertex Cover Problem briefly. Develop a greedy based approach to determine the minimum size vertex cover of a graph with n nodes and m edges of the following graph.**  **Given an undirected graph, a vertex cover is a subset of vertices such that every edge in the graph is incident to at least one vertex in the subset. The objective of the Vertex Cover Problem is to find the minimum size vertex cover, i.e., the smallest possible subset of vertices that covers all edges in the graph.**	**9**	**L2**	**5**	**2**	**2.6.3**

	1. Approx-Vertex-Cover (G = (V, E))
	2. {
	3. C = empty-set;
	4. E'= E;
	5. While E' is not empty do
	6. {
	7. Let (u, v) be any edge in E': (*)
	8. Add u and v to C;
	9. Remove from E' all edges incident to
	10. u or v;
	11. }
	12. Return C;
	13. }
	
	Greedy Vertex Cover (Graph G):
	Initialize an empty set C (vertex cover)
	For each edge (u, v) in G:
	If neither u nor v is in C:
	Add either u or v to C
	Return C
	
	Min Vertex Cover = {b, d, e}					

***Program Indicators are available separately for Computer Science and Engineering in AICTE examination reforms policy.**

Course Outcome (CO) and Bloom's level (BL) Coverage in Questions





Approved by the Audit Professor/Course Coordinator