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 B

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 strategy is used in the greedy approach for solving the Knapsack problem? a) Selecting items randomly **b) Choosing items with the highest value-to-weight ratio** c) Prioritizing items based on their weight only d) Selecting items with the lowest value-to-weight ratio	1	L1	3	2	2.5.2
2	What is the primary advantage of solving the optimal binary search tree problem using dynamic programming? a) It guarantees the shortest tree height b) It ensures the optimal placement of nodes c) It reduces the overall memory usage **d) It optimizes search operations by reducing the number of comparisons**	1	L1	3	2	2.5.2
3	The minimum number of scalar multiplications required for multiplying three matrices A, B, and C, where the dimensions are A(10x20), B(20x30), and C(30x40) is ________. **a) 2400** b) 3000 c) 3600 d) 4800 **Note: As there is a typo error in this question, provide a grace mark to the students.**	1	L2	3	2	2.6.3
4	What is the time complexity of the backtracking approach to solve the N Queens problem for an N x N chessboard? **a) O(N!)** b) O(2^N) c) O(N^2) d) O(N)	1	L2	4	2	2.6.3

5	What does the Floyd Warshall algorithm compute in a graph? **a) shortest path for each pair of vertices** b) Minimum spanning tree c) Longest path d) Topological sort	1	L1	4	2	2.5.2
6	In a complete graph with n vertices, how many Hamiltonian circuits exist? a) n b) (n-1)! **c) n!** d) 2^n	1	L2	4	2	2.1.2
7	In the Rabin-Karp algorithm, which factor can affect the accuracy of pattern matching? a) Size of the text b) Length of the pattern **c) Choice of hash function** d) Number of characters in the alphabet	1	L2	5	2	2.5.2
8	Consider the problem of finding the shortest path in a weighted graph. Is this problem NP-hard or NP-complete? a) NP-hard b) NP-complete **c) Neither NP-hard nor NP-complete** d) Both NP-hard and NP-complete	1	L1	5	2	2.5.2

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

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 B

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

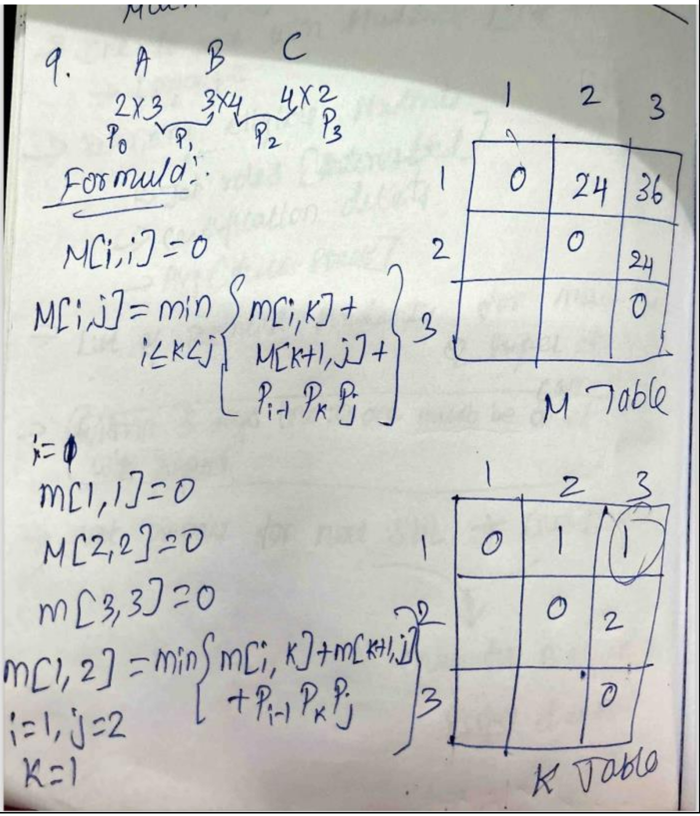
Course Articulation Matrix:

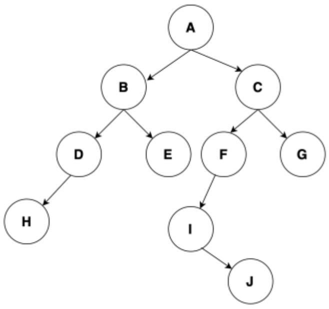
Course Outcome	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	Program Specific Outcomes		
													PSO-1	PSO-2	PSO-3
CO1	2	1	2	1	-	-	-	-		3	-	3	3	1	-
CO2	2	1	2	1	-	-	-	-		3	-	3	3	1	-
CO3	2	1	2	1	-	-	-	-		3	-	3	3	1	-
CO4	2	1	2	1	-	-	-	-		3	-	3	3	1	-
CO5	2	1	2	1	-	-	-	-		3	-	3	3	1	-

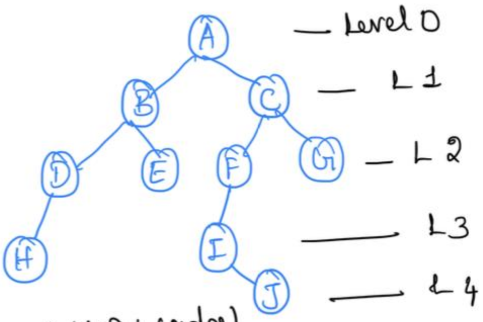
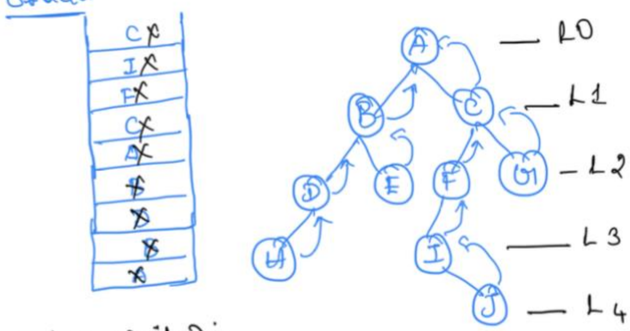
Part – B (3 x 5 = 15 Marks) Instructions: Answer all

Q. No	Question	Marks	BL	CO	PO	PI Code
9	Given three matrices A, B, and C with dimensions as follows: Matrix A: 2 x 3 Matrix B: 3 x 4 Matrix C: 4 x 2 Using dynamic programming, determine the minimum number of scalar multiplications required to multiply these matrices together efficiently. Describe the steps and discuss the time complexity of your algorithm.	5	L2	3	2	2.4.1

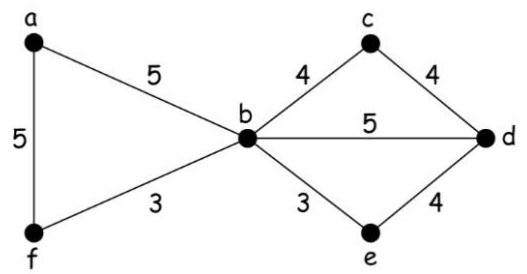
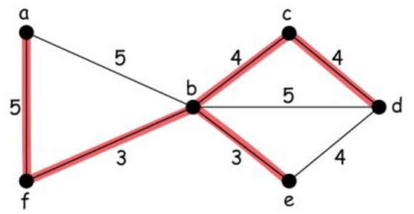
Solution: 4 Marks



	$= \min\{m[1,1]+m[1+1,2]+P_{1-1}P_1P_2\}$ $= m[1,1]+m[2,2]+P_0P_1P_2$ $= 0+0+2*3*4$ $= 24$ $m[2,3] = \min\{m[2,2]+m[2+1,3]+P_{2-1}P_2P_3$ $i=2,\ k=2,\ j=3$ $= \min\{m[2,2]+m[3,3]+P_1P_2P_3$ $= m[2,2]+m[3,3]+P_1P_2P_3$ $= 0+0+3*4*2$ $= 24$ $m[1,3] = \min\begin{cases} m[1,1]+m[1+1,3]+P_{1-1}P_1P_3 \\ m[1,2]+m[2+1,3]+P_{1-1}P_2P_3 \end{cases}$ $i=1, j=3$ $k=1,2$ $= \min\begin{cases} m[1,1]+m[2,3]+P_0P_1P_3 \\ m[1,2]+m[3,3]+P_0P_2P_3 \end{cases}$ $= \min\begin{cases} 0+24+2*3*2 \\ 24+0+2*4*2 \end{cases}$ $= \min\{36, 40\} = 36$ (when $k=1$) $K[1,3]=1$ $K[2,3]=2$ $((A)BC)$ $((A)\ B\ C)$ $O(n^3)$. Time Complexity: $O(n^3)$ (1 Mark)					
10	Consider the following tree structure. Implement Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms to explore this tree. Using appropriate data structures, provide the order in which the nodes are visited for both BFS and DFS. Demonstrate the traversal step-by-step. (5 Marks) 	5	L3	4	2	2.1.2

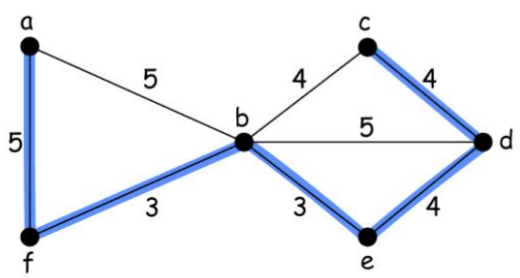
	1) BFS -> Level order traversal. -> Queue data structure -> visit & explore all the adjacent vertices. Queue:   Vertex visited: (order) A B C D E F G H I J 2) DFS -> Pre-order traversal -> stack data structure -> visit one vertex & explore Stack:  Vertex visited: A B D H E C F I J G					

11	In the bustling kingdom of Eldoria, there exists a prestigious guild of skilled adventurers who seek employment opportunities in various quests and missions. Each adventurer possesses unique abilities and strengths, making them suitable candidates for different quests. However, the guild master faces the challenge of efficiently selecting the most qualified adventurers for each quest. Write a pseudocode that employs a randomized approach to select the most qualified adventurers for a given quest, ensuring fairness and impartiality in the selection process. Also, provide the time complexity for your algorithm.
	Pseudocode: 4 Marks
	function selectAdventurers(adventurers, questRequirements):
	    selectedAdventurers = []
	    while (questRequirements not met):
		5	**L3**	**5**	**2**	**2.4.1**

	randomly select an adventurer from the pool if (adventurer meets questRequirements): add adventurer to selectedAdventurers return selectedAdventurers Time Complexity: $O(C_n \ln n)$ (1 Mark)					
	Part – C (3 x 9 = 27 Marks)					
12. A	Suppose you are tasked with designing an efficient transportation network at minimum cost for a city with multiple neighbourhoods. Each neighborhood needs to be connected to every other neighbourhood in the city to ensure seamless travel for residents and visitors. However, building roads between all pairs of neighbourhoods would be costly and unnecessary. Your goal is to devise a strategy to connect the neighbourhoods using the minimum number of roads while ensuring that every neighbourhood remains accessible from any other. Using a greedy approach, describe how you would go about selecting the roads to build to connect the neighbourhoods efficiently. Provide a detailed explanation of your strategy and justify your decisions at each step to achieve the most cost-effective transportation network for the city. Additionally, provide a pseudocode and time complexity outlining the steps of your greedy approach to constructing the transportation network.  Expected Answer: Marks can be awarded for both Prim's and Kruskal's algorithms. Students must have solved correctly using any one algorithm, providing proper pseudocode and mentioning the time complexity. Example: 4 Marks Pseudocode: marks Time complexity: 1 Mark Prims algorithm:  Total Cost = 5 + 3 + 3 + 4 + 4 = 19 Pseudocode: Prim (G, start): Initialize an empty set MST to store the minimum spanning tree Initialize a priority queue PQ to store vertices and	**9**	**L3**	**3**	**2**	**3.6.2**

```
their weights
    Add start vertex to MST and PQ with weight 0
    while PQ is not empty:
        u = extract_min(PQ)
        Add u to MST
        for each neighbor v of u:
            if v is not in MST:
                Add v to PQ with weight of edge (u, v)
    return MST
```

Kruskal Algorithm



Total Cost = 3 + 3 + 4 + 4 + 5 = 19

Pseudocode:

```
Kruskal(G):
    Sort the edges of G in non-decreasing order of their weights
    Initialize an empty set MST to store the minimum spanning tree
    for each vertex v in G:
        Make a set containing only v
    for each edge (u, v) in sorted edges:
        if find(u) is not equal to find(v):  // check if adding the edge creates a cycle
            Add (u, v) to MST
            Union the sets containing u and v
    return MST
```

Time Complexity: O(n^3)

(or)

12. B	Imagine you are a puzzle enthusiast who loves solving word puzzles. You have two sets of word tiles, each containing a sequence of letters arranged in a particular order. Your goal is to find the longest sequence of letters that appears in the same order in both sets of word tiles. However, you can only remove letters from the tiles, not rearrange them, to form the common sequence. **Example Problem: 5 Marks** Consider two sets of word tiles: Set A: {M, Z, J, A, W, X, U} Set B: {X, M, J, Y, A, U, Z} Determine the length of the longest sequence of letters that appears in the same order in both sets of word tiles using a dynamic Programming Approach. Provide the pseudocode for the same.	**9**	**L3**	**3**	**2**	**3.6.2**

Solution:

Y[] ↓	X[] →	0 ø	1 M	2 Z	3 J	4 A	5 W	6 X	7 U
0	ø	0	0	0	0	0	0	0	0
1	X	**0**	0	0	0	0	0	1	1
2	M	0	**1**	**1**	1	1	1	1	1
3	J	0	1	1	**2**	2	2	2	2
4	Y	0	1	1	**2**	2	2	2	2
5	A	0	1	1	2	**3**	**3**	**3**	3
6	U	0	1	1	2	3	3	3	**4**
7	Z	0	1	2	2	3	3	3	**4**

Pseudocode: 4 Marks

If(a[i]=b[j])

LCS [I,j] = 1+LCS[i-1, j-1]

Else

LCS[I,j] = max (LCS [i-1,j], LCS [i, j-1])

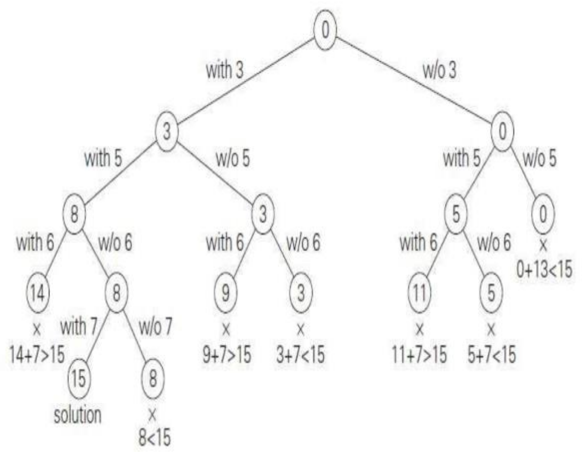
13. A	Imagine you are embarking on an adventure through a magical forest filled with treasure chest of variable sizes. The treasure chest contains a collection of coins, with each coin having a different value. Your goal is to find the combination of coins from the treasure chest that adds up to a specific target value, allowing you to unlock a hidden path deeper into the forest. However, you can only select coins from the treasure chest without exceeding the target value. Determine the combination of coins from the treasure chests that add up to the target value with state space tree and pseudocode using the Backtracking Approach.	**9**	**L3**	**4**	**2**	**2.1.2**

Using the example treasure chest and target value:

Treasure Chest: {3, 5, 6, 7}

Your target value is 15.

Solution: 5 Marks



Pseudocode: 4 Marks

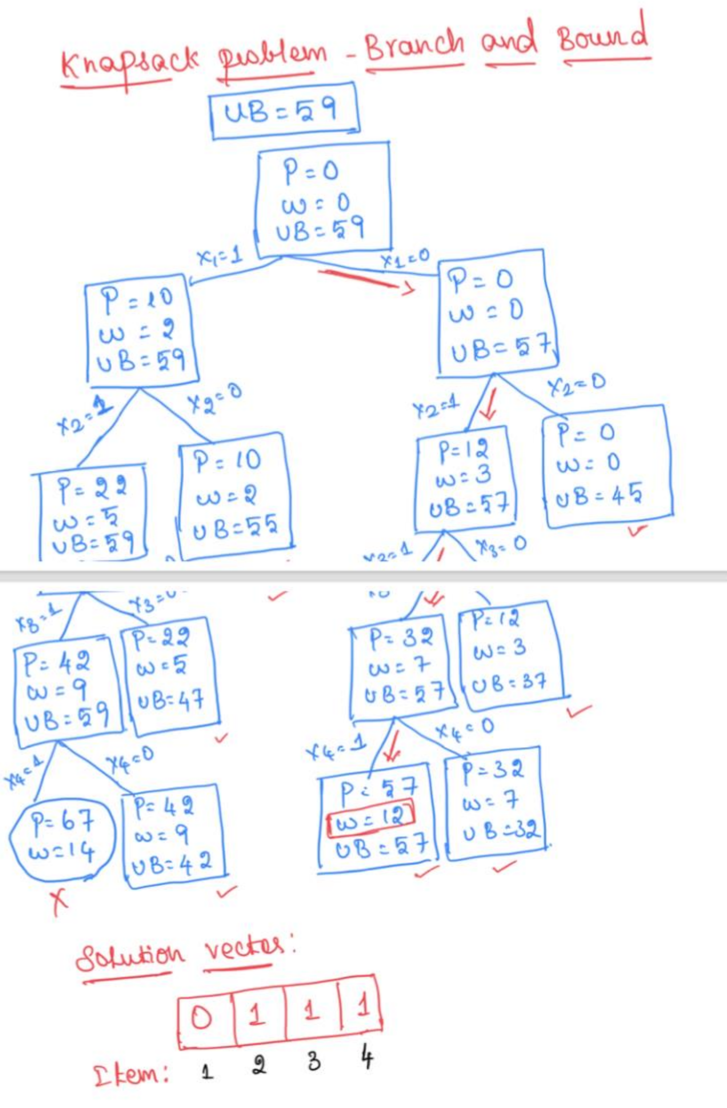
```
Alg sumofsets (s, k, r)
                 └─ index of weight in w[]        weight vector
// find out all subsets of w[1:n] that sum to M
// s = Σ_{j=1}^{k-1} w[j]*x[j] & r = Σ_{j=k}^{n} w[j]
{
// generate left child
  x[k] = 1
  If (s + w[k] == M) then write (x[1:k])
                                  // subset found
  else if (s + w[k] + w[k+1] ≤ M) then
     sumofsets (s + w[k], k+1, r - w[k]);
// Generate right child
  If ((s + r - w[k] ≥ M) and (s + w[k+1] ≤ M)) then
  {
    x[k] = 0
    sumofsets (s, k+1, r - w[k]);
  }
}
```

(or)

13. B	Alice is preparing for a wilderness expedition where she needs to carry a limited weight of supplies in her backpack. She has a list of items with their weights and values. However, her backpack has a strict weight limit. Alice wants to maximize the total value of the items she carries while ensuring she doesn't exceed the weight limit of her backpack. Derive the following example and provide the pseudocode for the same.	**9**	**L3**	**4**	**2**	**3.6.2**

Items	Tent	Sleeping Bag	Portable Stove	First Aid Kit
Profit	$10	$12	$20	$25
Weight	2 kg	3 kg	4 kg	5 kg
Backpack weight limit: 12 kg				

Example: 5 marks



Pseudocode: 4 Marks

Algorithm BB_KNAPSACK(cp, cw, k)
// Description : Solve knapsack problem using branch and bound
// Input:
cp: Current profit, initially 0
cw: Current weight, initially 0
k: Index of item being processed, initially 1

// Output:
fp: Final profit
Fw: Final weight
X: Solution tuple

cp ← 0
cw ← 0
k ← 1

if (cw + w[k] ≤ M) then
  Y[k] ← 1
  if (k < n) then

	BB_KNAPSACK(cp + p[k], cw + w[k], k + 1)
	end
	if (cp + p[k] > fp) && (k == n) then
	fp ← cp + c[k]
	fw ← cw + w[k]
	X ← Y
	end
	end
	
	if BOUND(cp, cw, k) ≥ fp then
	Y[k] ← 0
	if (k < n) then
	BB_KNAPSACK(cp, cw, k + 1)
	end
	if(cp>fp) && (k == n) then
	fp ← cp
	fw ← cw
	X ← Y
	end
	end
	Upper bound is computed as follow :
	Function BOUND(cp, cw, k)
	
	b ← cp
	c ← cw
	for i ← k + 1 to n do
	if (c + w[i] ≤ M) then
	c ← c + w[i]
	b ← b – p[i]
	end
	end
	return b					

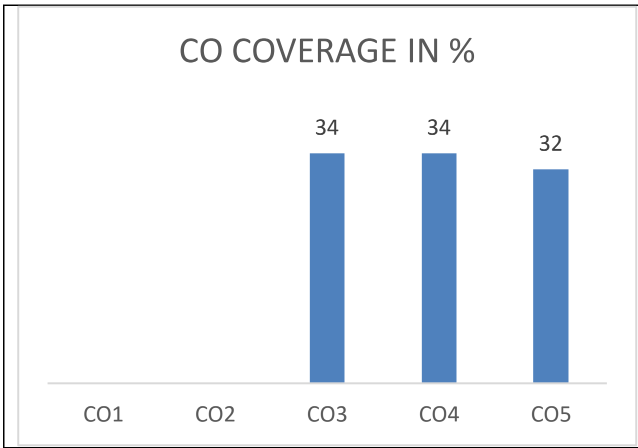
14. A	Considering the complexities inherent in sorting diverse datasets efficiently, explain what strategic approach can be employed to ensure that quicksort maintains an average-case time complexity of O(nlogn) when dealing with large datasets. Provide a pseudocode for the approach and solve the sample dataset given below.
	The dataset consists of the following elements:
	[10, 25, 30, 45, 50, 65, 70, 85].
	
	Example: 4 Marks
	Use Randomized quick sort to sort the array.
	Sorted Array: 10, 25, 30, 45, 50, 65, 70, 85
	
	Pseudocode: 5 Marks
	function randomizedQuicksort(array):
	if length(array) <= 1:
	return array
	
	pivotIndex = randomIndex(length(array)) // Choose a random index as the pivot
	pivot = array[pivotIndex]
	
	// Partition the array around the pivot
	lessThanPivot = []
	equalToPivot = []
		9	**L3**	**5**	**2**	**2.4.1**

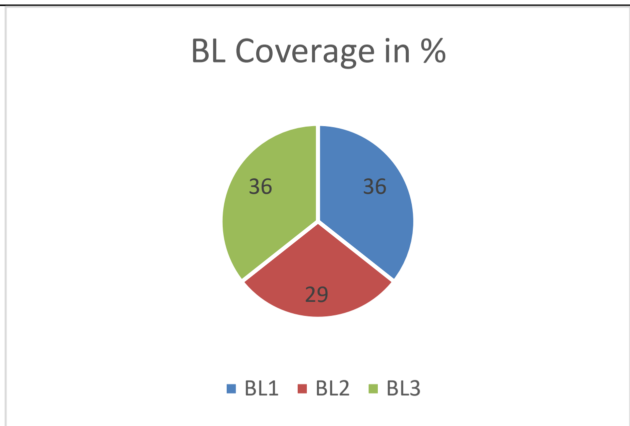
	greaterThanPivot = []
	for element in array:
	if element < pivot:
	append element to lessThanPivot
	else if element == pivot:
	append element to equalToPivot
	else:
	append element to greaterThanPivot
	
	// Recursively sort the subarrays
	sortedLess = randomizedQuicksort(lessThanPivot)
	sortedGreater = randomizedQuicksort(greaterThanPivot)
	
	// Concatenate the sorted subarrays with the pivot
	return concatenate (sortedLess, equalToPivot, sortedGreater)					

(or)						
14. B	Write an algorithmic approach to solve the Traveling Salesman Problem (TSP), a well-known NP-complete problem, using dynamic programming. Analyze the time complexity of your proposed solution and compare it with the Brute Force algorithm. **Algorithm: Traveling-Salesman-Problem (5 Marks)** C ({1}, 1) = 0 for s = 2 to n do for all subsets S ϵ {1, 2, 3, … , n} of size s and containing 1 C (S, 1) = ∞ for all j ϵ S and j ≠ 1 C (S, j) = min {C (S – {j}, i) + d(i, j) for i ϵ S and i ≠ j} Return minj C ({1, 2, 3, …, n}, j) + d(j, i) **Time Complexity Analysis: (1 Mark)** There are 2^n subsets of cities, and for each subset, we iterate over all cities, resulting in a time complexity of O(n^2 * 2^n). Each DP state takes O(n) time to compute. Overall, the time complexity of the dynamic programming solution is O(n^2 * 2^n). **Comparison with Brute Force: (3 Marks)** Brute Force: The brute force approach generates all permutations of cities and calculates the total distance for each permutation. The time complexity of brute force is O(n!), significantly worse than dynamic programming for larger n. Dynamic Programming: The dynamic programming approach optimally solves the problem in O(n^2 * 2^n) time, making it much more efficient than brute force for larger instances of the problem. In summary, while the dynamic programming solution for the Traveling Salesman Problem has a high time complexity, it is much more efficient than the brute force approach for larger instances of the problem due to its polynomial time complexity.	**9**	**L3**	**5**	**2**	**2.1.1**

***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