SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF COMPUTER APPLICATIONS

QUESTION BANK

I SEMESTER

MC4161 -ADVANCED DATA STRUCTURES AND ALGORITHMS

Regulation – 2024

Academic Year 2026-2027 (ODD Semester)

Prepared by

S.SHAKILABANU AP(O/G)

Department of Computer Applications

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203.

DEPARTMENT OF COMPUTER APPLICATIONS

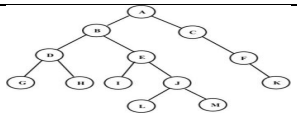
QUESTION BANK

SUBJECT : MC4161 -ADVANCED DATA STRUCTURES AND ALGORITHMS

SEM / YEAR : I Sem / I Year

UNIT I - LINEAR DATA STRUCTURES

Abstract Data Types (ADTs) – List ADT – Array-Based Implementation – Linked List Implementation Doubly-Linked Lists – Circular Linked Lists – Stack ADT: Implementation of Stacks – Queue ADT: Implementation of Queues – Applications.

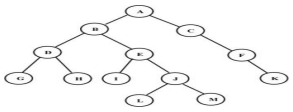
Q.No.	Question	Bloom's Taxonomy Level	Competence	Course Outcome
	PART – A			
1	Define Abstract Data Type. Give any two examples.	BTL -1	Remembering	CO1
2	Define data structure.	BTL -2	Understanding	CO1
3	Define list ADT.	BTL -2	Understanding	CO1
4	Distinguish between linear and nonlinear data structures.	BTL -1	Remembering	CO1
5	Compare between linear linked list and circular linked list.	BTL -2	Understanding	CO1
6	Define an array. Give an example.	BTL -1	Remembering	CO1
7	List any two applications of linked lists.	BTL -2	Understanding	CO1
8	Examine a doubly linked list with a neat diagram.	BTL -1	Remembering	CO1
9	Interpret the advantages and disadvantages of linked lists over arrays.	BTL -2	Understanding	CO1
10	Define stack ADT. 	BTL -1	Remembering	CO1
11	Given the prefix for an expression. Write its postfix: ++A*BCD and +*AB*CD	BTL -1	Remembering	CO1
12	Given an infix expression convert it into postfix expression using stack a+b*(c^d-e)^(f+g*h)-i	BTL -1	Remembering	CO1
13	Write the postfix and prefix forms of the expression: A+B*(C-D)/(P-R)	BTL -2	Understanding	CO1
14	Give the purpose of top and pop with suitable examples.	BTL -1	Remembering	CO1
15	What are the two kinds of dequeue?	BTL -2	Understanding	CO1

16	How to implement stack using a linked list?	BTL -2	Understanding	CO1
17	What are priority queues? What are the ways to implement priority queue?	BTL -2	Understanding	CO1
18	What is a circular queue?	BTL -1	Remembering	CO1
19	Circular queue is better than standard linear queue, why?	BTL -1	Remembering	CO1
20	Convert A*(B+D)/E-F*(G+H/K) into a postfix expression.	BTL -1	Remembering	CO1
21	Show a routine to perform enqueue operation in a queue.	BTL -2	Understanding	CO1
22	Differentiate between double ended queue and circular queue.	BTL -1	Remembering	CO1
23	Write an algorithm for deleting an element in a double ended queue	BTL -2	Understanding	CO1
24	List any four applications of stack.	BTL -1	Remembering	CO1
25	Difference between stack and queue	BTL -1	Remembering	CO1
	PART – B			
1	Demonstrate the PUSH and POP operations of a stack with a simple example.	BTL3	Applying	CO1
2	Implement a singly linked list and show how to check whether the list is empty, whether a node is the last node, and how to find an element.	BTL3	Applying	CO1
3	Implement the List ADT using a linked list and demonstrate its operations with an example.	BTL3	Applying	CO1
4	Compare the insert, delete and search operations of the List ADT in array and linked-list implementations with examples.	BTL4	Analyzing	CO1
5	Examine how elements are created, inserted, deleted and displayed in a circular linked list with an example.	BTL4	Analyzing	CO1
6	Demonstrate insertion and deletion in a doubly linked list with an example.	BTL3	Applying	CO1
7	Examine the advantages and disadvantages of a doubly linked list over a singly linked list with a suitable example.	BTL4	Analyzing	CO1
8	Analyze how a circular linked list can be used to implement a circular queue, with an example.	BTL4	Analyzing	CO1
9	Implement PUSH and POP operations of a stack using a linked list and trace them with an example.	BTL3	Applying	CO1
10	Apply a stack to convert the infix expression (A+B)*C-D into postfix and show each step.	BTL3	Applying	CO1
11	Demonstrate how a stack converts the infix expression A+B*(C-D)/E into postfix, showing each step.	BTL3	Applying	CO1

12	Apply a stack to convert ((A+B)-C*(D/E))+F into postfix and prefix expressions and show the steps.	BTL3	Applying	CO1
13	Demonstrate insertion and deletion at both ends of a double-ended queue (deque) with an example.	BTL3	Applying	CO1
14	Examine how a priority queue works and compare it with an ordinary queue with an example.	BTL4	Analyzing	CO1
15	Demonstrate the enqueue and dequeue operations of a queue with a simple example.	BTL3	Applying	CO1
16	Implement a stack using an array and demonstrate PUSH and POP, including overflow and underflow.	BTL3	Applying	CO1
17	Examine the insertion and deletion operations of a circular queue using an array with an example.	BTL4	Analyzing	CO1
18	Compare stack and queue in terms of operations, order of access and applications with examples.	BTL4	Analyzing	CO1

UNIT II - ALGORITHMS IN COMPUTING

Introductions to Algorithms – Iterative and Recursive Algorithms – Designing Algorithms – Analyzing Algorithms – Growth of Functions: Asymptotic Notations – Standard Notations and Common Functions – Recurrences: The Substitution Method – The Recursion – Tree Method

Q.No.	Question	Bloom's Taxonomy Level	Competence	Course Outcome
	PART – A			
1	What do you mean by algorithm?	BTL-2	Understanding	CO2
2	Define an Iterative Algorithm.	BTL-2	Understanding	CO2
3	What are the types of algorithm efficiencies?	BTL-2	Understanding	CO2
4	What is space complexity?	BTL-1	Remembering	CO2
5	Define asymptotic notations.	BTL-2	Understanding	CO2
6	Define the asymptotic notation "Big oh" (0)	BTL-1	Remembering	CO2
7	Difference between Best Case and Worst-Case Complexities	BTL-1	Remembering	CO2
8	Define the asymptotic notation "Omega" (Ω).	BTL-2	Understanding	CO2
9	Define the asymptotic notation "theta" (θ) 	BTL-2	Understanding	CO2
10	Design an algorithm for computing area and circumference of the circle.	BTL-2	Understanding	CO2

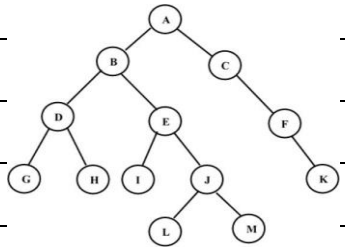
11	How do you measure the efficiency of an algorithm?	BTL-2	Understanding	CO2
12	Write down the properties of asymptotic notations?	BTL-1	Remembering	CO2
13	What is a basic operation?	BTL-2	Understanding	CO2
14	What is validation of algorithm?	BTL-1	Remembering	CO2
15	What is recursive algorithm?	BTL-2	Understanding	CO2
16	Define recurrence	BTL-1	Remembering	CO2
17	Give the general plan for analyzing recursive algorithm	BTL-2	Understanding	CO2
18	What are all the methods available for solving recurrence relations?	BTL-1	Remembering	CO2
19	Define Substitution Method	BTL2	Understand	CO2
20	What are the applications of Algorithm Visualization?	BTL2	Understand	CO2
21	List the reasons for choosing an approximate algorithm.	BTL2	Understand	CO2
22	What is the time complexity of linear search and binary search?	BTL1	Remember	CO2
23	What are the types of recurrence relations?	BTL1	Remember	CO2
24	Name any four common functions used to compare the growth of algorithms.	BTL1	Remember	CO2
25	What is a recurrence relation? Give one example of a recurrence relation used to analyze a recursive algorithm.	BTL-2	Understanding	CO2
	PART –B			
1	Demonstrate the steps of designing and analyzing an algorithm using a simple problem such as finding the largest number in a list.	BTL3	Applying	CO2
2	Demonstrate best-case, average-case and worst-case analysis using a simple algorithm with an example.	BTL3	Applying	CO2
3.	Differentiate Big-O, Omega and Theta notations with simple examples	BTL4	Analyzing	CO2
4	Apply Big-O notation to compare and rank the growth of n, log n, n log n, n^2 and 2^n.	BTL3	Applying	CO2
5	Examine the time and space efficiency of a simple algorithm, such as finding the sum of n numbers, with an example.	BTL4	Analyzing	CO2
6	Apply the iterative method to the recurrence T(n) = T(n-1) + 1, where T(1) = 1, and find its time complexity.	BTL3	Applying	CO2
7	Apply Theta notation to find the order of growth of the function $3n^2 + 2n + 1$.	BTL3	Applying	CO2
8	Compare the iterative and recursive versions of finding the	BTL4	Analyzing	CO2

	factorial of a number and analyze their time complexity.			
9	Calculate the time complexity of a simple loop-based algorithm step by step.	BTL3	Applying	CO2
10	Compare the time and space efficiency of two simple algorithms with examples.	BTL4	Analyzing	CO2
11	Apply the substitution method to the recurrence T(n) = T(n/2) + 1, T(1) = 1 and find its Big-O complexity.	BTL3	Applying	CO2
12	Apply backward substitution and forward substitution to the recurrence T(n) = T(n-1) + n, T(1) = 1.	BTL3	Applying	CO2
13	Apply the recursion-tree method to the recurrence T(n) = 2T(n/2) + n and find its complexity.	BTL3	Applying	CO2
14	Examine how the recursion-tree method works by finding the complexity of T(n) = T(n/2) + n.	BTL4	Analyzing	CO2
15	Compare the iterative, substitution and recursion-tree methods for recurrence relations with an example.	BTL4	Analyzing	CO2
16	Calculate the best-case, average-case and worst-case time complexity and space complexity of linear search with an example.	BTL3	Applying	CO2
17	Apply the substitution method to the recurrence T(n) = 2T(n/2) + n, T(1) = 1 and find its Big-O complexity.	BTL3	Applying	CO2
18	Compare the substitution method and the recursion-tree method by applying both to T(n) = 2T(n/2) + 1.	BTL4	Analyzing	CO2

UNIT III - HIERARCHICAL DATA STRUCTURES & HASHING

Trees: Preliminaries – Implementation of Trees – Tree Traversals with an Application – Binary Trees: Implementation – Expression Trees – Search Tree ADT – Binary Search Trees – Applications of Trees - Fundamentals of Hashing – Hash Function – Separate Chaining – Open Addressing.

Q.No.	Question	Bloom's taxonomy Level	Competence	Course Outcome
	PART – A			
1	What is tree traversal?	BTL 2	Understand	CO3
2	Mention the type of traversals.	BTL 2	Understand	CO3
3	Define a binary tree. Give an example.	BTL 1	Remember	CO3
4	Create an expression tree for the expression. 4+((7+9) *2).	BTL 2	Understand	CO3
5	Differentiate AVL tree and Binary search tree.	BTL 1	Remember	CO3
6	Give the various types of rotations in AVL tree during the insertion of a node?	BTL 1	Remember	CO3

7	For the given tree i. List the siblings for node E ii. Compute the height.  	BTL 2	Understand	CO3
8	What is the time complexity of searching an element in a binary search tree in the best case and the worst case?	BTL 1	Remember	CO3
9	Define the balance factor of AVL Tree.	BTL 1	Remember	CO3
10	Define Separate Chaining.	BTL 2	Understand	CO3
11	Define an expression tree. Give an example of it.	BTL 1	Remember	CO3
12	Define AVL Tree.	BTL 1	Remember	CO3
13	Identify the properties of Binary Search Tree.	BTL 1	Remember	CO3
14	Identify the type of traversal that gives the data in ascending order.	BTL2	Understand	CO3
15	Identify the properties of AVL Tree.	BTL2	Understand	CO3
16	List out the various operations that can be performed on Binary Search Tree to make it a height balanced one.	BTL 1	Remember	CO3
17	Identify the three cases for deleting a node from a binary search tree.	BTL 1	Remember	CO3
18	Define leaves in a tree.	BTL1	Remember	CO3
19	What do you mean by level of the tree?	BTL2	Understand	CO3
20	Identify the properties of a binary tree.	BTL1	Remember	CO3
21	Define Hashing.	BTL2	Understand	CO3
22	List the Different ways of Open Addressing.	BTL1	Remember	CO3
23	List out the applications of Trees.	BTL2	Understand	CO3
24	What do you mean by Heap?	BTL1	Remember	CO3
25	What is a hash function? Mention any two characteristics of a good hash function.	BTL-2	Understanding	CO3
	PART –B			
1	Demonstrate preorder, inorder and postorder traversals of a binary tree with an example.	BTL3	Applying	CO3

2	Demonstrate the search, minimum and maximum operations on a Binary Search Tree (BST) with an example.	BTL3	Applying	CO3
3	Apply the insertion steps to build a BST from 50, 30, 70, 20, 40, 60, 80 and show each step.	BTL3	Applying	CO3
4	Examine the insert, search and delete operations of a BST and trace each with an example.	BTL4	Analyzing	CO3
5	Compare array and linked representations of a binary tree with their advantages and disadvantages.	BTL4	Analyzing	CO3
6	Apply BST insertion to construct a BST using 45, 39, 56, 12, 34, 78, 32, 10, 89, 54, 67, 81.	BTL3	Applying	CO3
7	Demonstrate the three cases of deleting a node from a BST (leaf, one child, two children) with an example.	BTL3	Applying	CO3
8	Examine whether a given binary search tree is an AVL tree and apply rotations to balance it if needed.	BTL4	Analyzing	CO3
9	Apply AVL-tree insertion and rotations for the sequence 63, 9, 19, 27, 18, 108, 99, 81 and show each step.	BTL3	Applying	CO3
10	Apply BST properties to find the minimum and maximum values and perform a post order traversal with an example.	BTL3	Applying	CO3
11	Apply the BST deletion steps to delete 20, 30 and 50 one by one from the BST built using 50, 30, 70, 20, 40, 60, 80.	BTL3	Applying	CO3
12	Apply the steps to construct an expression tree for (a+b)*(c-d) and obtain its traversals.	BTL3	Applying	CO3
13	Compare the deletion of a node from a binary tree using the merging and copying techniques with examples.	BTL4	Analyzing	CO3
14	Examine the time complexity of search, insert and delete operations in a BST for the best case and the worst case (skewed tree) with an example.	BTL4	Analyzing	CO3
15	Apply the expression-tree construction steps for (a+b*c)+((d*e+f)*g) and obtain its inorder, preorder and postorder traversals.	BTL3	Applying	CO3
16	Apply linear probing to insert 12, 45, 67, 88, 27, 78, 20, 62, 36, 55 into a hash table of size 10 and trace every step.	BTL3	Applying	CO3
17	Apply quadratic probing to insert 20, 5, 10, 22, 33, 40, 50, 30, 51, 31 into a hash table of size 11 and trace every step.	BTL3	Applying	CO3
18	Compare separate chaining and open addressing with suitable examples.	BTL4	Analyzing	CO3

UNIT IV - SORTING AND GRAPHS

Sorting Algorithms: Insertion Sort, Quick Sort, Heap Sort - Graphs: Representation of Graphs – Graph Traversals – Topological Sort – Shortest Path Algorithms: Dijkstra's Algorithm – Minimum Spanning

Tree: Prim's and Kruskal's Algorithm.				
Q.No.	**Question**	**Bloom's Taxonomy Level**	**Competence**	**Course Outcome**
PART – A				
1	Define Heap sort?	BTL 1	Remember	CO 4
2	Define Quick Sort.	BTL 2	Understand	CO 4
3	What is the difference between insertion sort and quick sort?	BTL 2	Understand	CO 4
4	Define graph.	BTL 2	Understand	CO 4
5	Create an undirected graph and its adjacency matrix for the following specification of a graph G. V(G)=1,2,3,4 E(G) ={ (1,2),(1,3),(3,3),3,4),(4,1)}	BTL 2	Understand	CO 4
6	What is meant by bi-connected graph?	BTL 1	Remember	CO 4
7	Give the purpose of Dijkstra's algorithm.	BTL 2	Understand	CO 4
8	Differentiate cyclic and acyclic graph	BTL 1	Remember	CO 4
9	Classify strongly connected and weakly connected graph.	BTL 1	Remember	CO 4
10	What is an articulation point? Give example.	BTL 1	Remember	CO 4
11	What are the representations of the graph?	BTL 1	Remember	CO 4
12	Define minimum spanning tree. Give an example	BTL 1	Remember	CO 4
13	State the principle of Topological sorting.	BTL 1	Remember	CO 4
14	Explain procedure for Depth first search algorithm.	BTL 2	Understand	CO 4
15	What is the time complexity of quick sort in the best case and the worst case?	BTL 2	Understand	CO 4
16	Prove that the number of edges in a complete graph of n vertices in n (n-1)/2.	BTL 2	Understand	CO 4
17	Assess the minimum number spanning tree possible for a complete graph with n vertices.	BTL 2	Understand	CO 4
18	Give two applications of graphs.	BTL 2	Understand	CO 4
19	What is visiting and traversing in a graph?	BTL 1	Remember	CO 4
20	What is a complete graph? Give an example.	BTL 2	Understand	CO 4
21	What is insertion sort?	BTL 1	Remember	CO 4
22	What is in-degree of a graph. Give suitable example.	BTL 2	Understand	CO 4
23	What is meant by out-degree of a graph?	BTL 1	Remember	CO 4

24	What are the two basic parts of Heap sort?	BTL2	Understand	CO4
25	List any two differences between BFS and DFS.	BTL2	Understand	CO4
PART –B				
1	Apply insertion sort to arrange 25, 10, 40, 5, 30, 15 in ascending order and show each pass.	BTL3	Applying	CO4
2	Compare Quick Sort and Heap Sort, and apply Quick Sort to arrange 18, 29, 68, 32, 43, 37, 87, 24, 47, 50 in ascending order.	BTL4	Analyzing	CO4
3	Apply Quick Sort to arrange 12, 24, 8, 71, 4, 23, 6, 89, 56 in ascending order and show all partition steps.	BTL3	Applying	CO4
4	Examine heap sort by building a max heap and sorting 40, 10, 30, 50, 20 step by step.	BTL4	Analyzing	CO4
5	Compare the adjacency-matrix and adjacency-list representations of a graph with an example.	BTL4	Analyzing	CO4
6	Apply topological sort to a given directed acyclic graph and write its adjacency list.	BTL3	Applying	CO4
7	Apply Dijkstra's algorithm to find the shortest paths from a source vertex to all other vertices of a given graph.	BTL3	Applying	CO4
8	Compare insertion sort, quick sort and heap sort in terms of best-case, worst-case time complexity and space.	BTL4	Analyzing	CO4
9	Differentiate Depth-First Search (DFS) and Breadth-First Search (BFS) with examples.	BTL4	Analyzing	CO4
10	Apply DFS on an undirected graph to find its connected components and state the time complexity.	BTL3	Applying	CO4
11	Examine Breadth-First Search (BFS) and trace its execution on a simple graph.	BTL4	Analyzing	CO4
12	Apply DFS to find the articulation points of a connected graph and trace the steps with an example.	BTL3	Applying	CO4
13	Apply Prim's algorithm to find the minimum spanning tree of a given weighted graph and show each step.	BTL3	Applying	CO4
14	Examine Kruskal's algorithm and trace the construction of a minimum spanning tree for a simple weighted graph.	BTL4	Analyzing	CO4
15	Examine biconnectivity and categorize graphs (directed, undirected, weighted, connected) with examples.	BTL4	Analyzing	CO4
16	Examine the applications of graphs and relate them to graph representations and traversal techniques.	BTL4	Analyzing	CO4
17	Apply heap sort to build a max heap and sort 15, 40, 25, 10, 35, 5 in ascending order and show each step.	BTL3	Applying	CO4

18	Compare Prim's and Kruskal's algorithms for finding a minimum spanning tree using a suitable graph.	BTL4	Analyzing	CO4
UNIT V - ALGORITHM DESIGN TECHNIQUES Greedy Algorithms: Huffman Codes – Divide and Conquer: Merge Sort – Dynamic Programming: Using a Table instead of Recursion – Ordering Matrix Multiplications – Introduction to NP Completeness				
Q.No.	**Question**	**Bloom's Taxonomy Level**	**Competence**	**Course Outcome**
PART – A				
1	Mention different algorithm design techniques.	BTL 1	Remember	CO 5
2	Mention the two properties of sorting algorithms.	BTL 2	Understand	CO 5
3	State greedy technique.	BTL 2	Understand	CO 5
4	Define dynamic programming.	BTL 2	Understand	CO 5
5	Define divide and conquer design technique	BTL 2	Understand	CO 5
6	State the Principle of Optimality.	BTL 1	Remember	CO 5
7	What is Huffman trees?	BTL 2	Understand	CO 5
8	List the advantage of Huffman's encoding?	BTL 1	Remember	CO 5
9	What do you mean by Huffman code?	BTL 1	Remember	CO 5
10	List any two examples of problems solved using the greedy method.	BTL 1	Remember	CO 5
11	What do you mean by row major and column major?	BTL 1	Remember	CO 5
12	Show the general procedure of dynamic programming.	BTL 1	Remember	CO 5
13	Define the matrix chain multiplication problem.	BTL 1	Remember	CO 5
14	List the features of dynamic programming?	BTL 2	Understand	CO 5
15	How Dynamic Programming is used to solve Knapsack Problem?	BTL 2	Understand	CO 5
16	List two major parts in Huffman Coding.	BTL 2	Understand	CO 5
17	Define NP Problem.	BTL 2	Understand	CO 5
18	Differentiate P and NP.	BTL 2	Understand	CO 5
19	How NP-hard problems are different from NP-Complete?	BTL 1	Remember	CO 5
20	Give some examples of non-polynomial problem.	BTL 2	Understand	CO 5
21	Give some examples of Polynomial problem.	BTL 1	Remember	CO 5
22	What is the time complexity of the Huffman coding algorithm?	BTL 2	Understand	CO 5
23	Define Polynomial (P) problem.	BTL 1	Remember	CO 5
24	On what basis problems are classified?	BTL2	Understand	CO 5

25	What is NP-Completeness? Mention the two important classes associated with NP-Completeness.	BTL-1	Remembering	CO 5
	PART –B			
1	Examine the table-filling method of dynamic programming by computing the binomial coefficient C(5, 2) using a table.	BTL4	Analyzing	CO5
2	Examine the divide-and-conquer approach and trace it on a simple problem such as finding the maximum of an array.	BTL4	Analyzing	CO5
3	Examine Merge Sort as a divide-and-conquer algorithm and trace it for 38, 27, 43, 3, 9, 82, 10.	BTL4	Analyzing	CO5
4	Examine how a greedy algorithm works using the coin-change problem and trace the choices made.	BTL4	Analyzing	CO5
5	Examine the dynamic programming approach and trace it by computing Fibonacci numbers using a table.	BTL4	Analyzing	CO5
6	Compare the greedy method and dynamic programming with a suitable example.	BTL4	Analyzing	CO5
7	Apply dynamic programming to the 0/1 knapsack problem with items of weight 1, 3, 4, values 15, 20, 30 and capacity 4, and fill the table.	BTL3	Applying	CO5
8	Apply the greedy method to solve the fractional knapsack problem for a given set of items and trace the choices.	BTL3	Applying	CO5
9	Compare dynamic programming and divide-and-conquer with a suitable example.	BTL4	Analyzing	CO5
10	Examine Floyd's algorithm and trace the shortest paths for a small weighted graph.	BTL4	Analyzing	CO5
11	Apply the Huffman coding algorithm to build a Huffman tree for A:5, B:9, C:12, D:13, E:16, F:45 and state its time complexity.	BTL3	Applying	CO5
12	Examine the dynamic programming method for the Optimal Binary Search Tree problem and find its time complexity.	BTL4	Analyzing	CO5
13	Examine the two stages of Huffman coding (building the tree and assigning codes) with an example.	BTL4	Analyzing	CO5
14	Examine the advantages of Huffman coding over fixed-length coding with an example.	BTL4	Analyzing	CO5
15	Examine the Matrix-Chain Multiplication problem using dynamic programming and trace the solution for matrices of order 10×30, 30×5 and 5×60.	BTL4	Analyzing	CO5
16	Differentiate P and NP problems with simple examples.	BTL4	Analyzing	CO5
17	Apply the idea of NP-hard problems to examples such as the Traveling Salesman and Hamiltonian cycle problems.	BTL3	Applying	CO5

18	Differentiate NP, NP-hard and NP-complete problems with suitable examples.	BTL4	Analyzing	CO5

****