

SRM VALLIAMMAI ENGINEERING COLLEGE
(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF INFORMATION TECHNOLOGY

QUESTION BANK



I SEMESTER

**CP3161 – ADVANCED DATA STRUCTURES AND
ALGORITHMS**

Regulation – 2023

Academic Year 2026 – 2027(odd Semester)

Prepared by

Mrs.G.Santhiya, Assistant Professor, (Sr.G)/IT



SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203.



DEPARTMENT OF INFORMATION TECHNOLOGY QUESTION BANK

SUBJECT : CP3161 – ADVANCED DATA STRUCTURES AND ALGORITHMS

SEM / YEAR: I Semester/ I Year

UNIT I FUNDAMENTALS			
Properties of Big-oh Notation – Conditional Asymptotic Notation – Algorithm Analysis – Amortized Analysis – Introduction to NP-Completeness/NP-Hard – Recurrence Equations – Solving Recurrence Equations – Time-Space Tradeoff.			
PART – A			
Q. No	Questions	BT Level	Competence
1	Define Big-Oh notation.	BTL1	Remembering
2	List the properties of Big-Oh notation.	BTL1	Remembering
3	Differentiate between Big-O, Big-Theta, and Big-Omega notations.	BTL2	Understanding
4	Define conditional asymptotic notation.	BTL1	Remembering
5	What is algorithm analysis?	BTL1	Remembering
6	Define amortized analysis and give its significance.	BTL2	Understanding
7	What are NP problems?	BTL1	Remembering
8	Differentiate between NP-Complete and NP-Hard problems.	BTL2	Understanding
9	Define recurrence relation and give one example.	BTL1	Remembering
10	What is a time-space tradeoff?	BTL2	Understanding
11	What is asymptotic analysis?	BTL1	Remembering
12	Define the best case, average case, and worst case of an algorithm.	BTL2	Understanding
13	Define the order of growth of an algorithm.	BTL1	Remembering
14	What is the role of mathematical induction in recurrence relations?	BTL2	Understanding
15	Define algorithmic efficiency.	BTL1	Remembering
16	What is a polynomial-time algorithm?	BTL2	Understanding
17	Define exponential-time algorithm with example.	BTL2	Understanding
18	List the various methods of solving recurrence relations.	BTL1	Remembering
19	Define deterministic and non-deterministic algorithms.	BTL2	Understanding
20	What is meant by computational complexity?	BTL2	Understanding
21	Define the substitution method used for solving recurrences.	BTL1	Remembering

22	What is the master theorem used for?	BTL2	Understanding
23	Define algorithmic optimality.	BTL1	Remembering
24	What are the performance analysis techniques of an algorithm?	BTL2	Understanding
PART – B			
1	Describe in detail about asymptotic notations used in algorithm analysis.	BTL1	Remembering
2	Discuss briefly the sequence of steps in designing and analyzing an algorithm.	BTL1	Remembering
3	Explain in detail about divide and conquer approach .	BTL1	Remembering
4	Describe in detail about merge sort with an example.	BTL1	Remembering
5	State the best, average and worst case complexities of binary search for successful and unsuccessful search.	BTL2	Understanding
6	Solve the recurrence relation using substitution method $T(n) = \begin{cases} T(1) & n=1 \\ aT(n/b) + f(n) & n>1 \end{cases}$, where $a=5, b=4$, and $f(n)=cn^2$	BTL2	Understanding
7	Demonstrate the use of Big-O properties to determine the efficiency of insertion sort.	BTL3	Applying
8	Illustrate the concept of conditional asymptotic notation with suitable examples.	BTL3	Applying
9	Solve the recurrence relation $T(n) = 2T(n/2) + n$ using the Master Theorem.	BTL3	Applying
10	Show how amortized analysis is used to evaluate stack operations using the accounting method.	BTL3	Applying
11	Examine the difference between worst-case and amortized analysis using examples.	BTL4	Analyzing
12	Differentiate between merge sort and quicksort based on their asymptotic complexities.	BTL4	Analyzing
13	Interpret the recurrence relation for binary search and discuss its implications on performance.	BTL4	Analyzing
14	Compare and contrast the characteristics of P, NP, NP-Complete, and NP-Hard problems.	BTL4	Analyzing
15	Assess how the time-space tradeoff affects the performance of algorithms.	BTL5	Evaluating
16	Judge the significance of amortized analysis compared to average-case analysis.	BTL5	Evaluating
17	Critique the feasibility of developing polynomial-time algorithms for NP-Complete problems.	BTL5	Evaluating
18	Design an algorithm that demonstrates time-space tradeoff using lookup tables.	BTL6	Creating
19	Formulate and solve the recurrence relation for the Tower of Hanoi problem.	BTL6	Creating
20	Develop a pseudo-code to identify whether a given problem belongs to the NP class.	BTL6	Creating
21	Construct a comparative chart showing Big-O, Big-Theta, and Big-Omega notations with examples.	BTL6	Creating
22	Propose a strategy to approximate NP-Complete problems in polynomial time.	BTL6	Creating
23.	Create a case study demonstrating the application of amortized analysis in algorithm design.	BTL6	Creating
24.	Assess how recurrence equations can be solved with suitable example.	BTL5	Evaluating
25.	Analyze the asymptotic notations used for best case ,average case and worst case analysis of algorithms and Write an algorithm for finding maximum element of an array perform best , worst and average case complexity with appropriate order notations	BTL5	Analyzing
26.	Prove that the circuit-satisfiability problem belongs to the class NP and also NP-Hard.	BTL5	Evaluating
27.	Analyze the best, average and worst case complexity of quick sort.	BTL4	Analyzing

UNIT II SEARCH STRUCTURES

Binary Search Trees – AVL Trees – Red-Black trees – Multi-way Search Trees –B-Trees –Splay Trees – Tries

PART – A

Q. No.	Questions	BT Level	Competence
1	Why don't we allow a minimum degree of $t = 1$ in B-trees ?	BTL1	Remembering
2	Define a Binary Search Tree (BST).	BTL1	Remembering
3	State the binary-search-tree property.	BTL1	Remembering
4	List the operations that can be performed on a BST.	BTL1	Remembering
5	What is the height of a binary search tree?	BTL1	Remembering
6	Define an AVL tree.	BTL1	Remembering
7	What is the balance factor in an AVL tree?	BTL1	Remembering
8	List the types of rotations used to balance an AVL tree.	BTL1	Remembering
9	Define a Red-Black Tree and mention its properties.	BTL2	Understanding
10	What are the rules for coloring in a Red-Black Tree?	BTL2	Understanding
11	Differentiate between AVL Tree and Red-Black Tree.	BTL2	Understanding
12	What is a B-Tree?	BTL1	Remembering
13	Define the term “minimum degree” in a B-Tree.	BTL1	Remembering
14	What is a Multiway Search Tree?	BTL2	Understanding
15	What is the need for using a Multiway Search Tree?	BTL2	Understanding
16	Define a Splay Tree.	BTL1	Remembering
17	What is the purpose of the splaying operation?	BTL2	Understanding
18	What are the advantages of using splay trees?	BTL2	Understanding
19	Define a Trie data structure.	BTL1	Remembering
20	List the applications of tries.	BTL2	Understanding
21	Differentiate between Trie and Binary Search Tree.	BTL2	Understanding
22	What is the time complexity of searching in a B-Tree?	BTL2	Understanding
23	Define insertion and deletion in AVL trees.	BTL1	Remembering
24	What is the difference between internal and external nodes in B-Trees?	BTL2	Understanding
25	What is the disadvantage of using splay trees?	BTL2	Remembering
26	When we have red-black trees and AVL trees that can perform most of operations in logarithmic times, then what is the need for splay trees?	BTL6	Understanding
27	Why to prefer splay trees?	BTL4	Remembering

PART – B			
1	Demonstrate the procedure for deleting a given node z from a Binary Search Tree and illustrate the RIGHT-ROTATE operation in Red-Black Trees.	BTL3	Applying
2	Explain the issues with AVL Trees and justify how Red-Black Trees resolve them; describe their key properties with an example.	BTL4	Analyzing
3	Implement the code for deletion in a Binary Search Tree, covering all three cases (leaf, one child, two children).	BTL3	Applying
4	Illustrate the structure and construction rules of a Binary Search Tree with a clear example.	BTL3	Applying
5	Demonstrate preorder, in-order and post-order traversals in a Binary Tree and construct insertion code.	BTL3	Applying
6	Summarize the fundamental operations of Binary Search Trees and interpret their working with examples.	BTL4	Analyzing
7	Illustrate the structure and working of Splay Trees with a neat diagram showing rotations.	BTL3	Applying
8	Demonstrate the working principle of 2–3 Trees and analyze their insertion and deletion processes with an example.	BTL4	Analyzing
9	Compare the balancing strategies of Red-Black Trees and AVL Trees using examples.	BTL4	Analyzing
10	Implement the operations of B-Trees and illustrate insertion and deletion steps with figures.	BTL3	Applying
11	Examine the characteristics and usefulness of Binary Trees through suitable examples.	BTL4	Analyzing
12	Analyze the insertion and deletion operations of AVL Trees and interpret how rotations restore balance.	BTL4	Analyzing
13	Assess the performance of Binary Search Trees for various types of inputs and justify their applications.	BTL5	Evaluating
14	Illustrate all possible AVL rotations and explain the algorithmic steps for each case with examples.	BTL4	Analyzing
15	Design algorithms for the following AVL Tree routines: (a) Insertion (b) Deletion (c) Single Rotation (d) Double Rotation.	BTL6	Creating
16	Construct an AVL Tree using the values {11, 5, 35, 8, 4, 2, 3, 77, 1} in an initially empty tree and write the insertion algorithm.	BTL6	Creating
17	Evaluate the time complexity for searching the keys 90 and 80 in a BST containing {50, 30, 60, 40, 35, 80, 90}; compare its performance when converted into AVL and Splay Trees and recommend the most suitable structure with justification.	BTL5	Evaluating

UNIT III HEAP STRUCTURES			
Min/Max heaps – Heaps – Leftist Heaps – Binomial Heaps – Fibonacci Heaps – Skew Heaps – Lazy Binomial Heaps			
PART – A			
Q. No.	Questions	BT Level	Competence
1	Define a heap.	BTL1	Remembering
2	Differentiate between a min-heap and a max-heap.	BTL2	Understanding
3	State the heap-order property.	BTL1	Remembering
4	What is a complete binary tree?	BTL1	Remembering
5	List the basic operations supported by a heap.	BTL1	Remembering
6	Define a deap.	BTL1	Remembering
7	What is a leftist heap?	BTL1	Remembering
8	List the advantages of leftist heaps.	BTL2	Understanding
9	Define a binomial heap.	BTL1	Remembering
10	List the structural properties of a binomial heap.	BTL2	Understanding
11	What is a Fibonacci heap?	BTL1	Remembering
12	State any two operations supported by Fibonacci heaps.	BTL2	Understanding
13	Define a skew heap.	BTL1	Remembering
14	Explain lazy binomial heaps.	BTL2	Understanding
15	What is the use of a priority queue?	BTL1	Remembering
16	Define the meld (merge) operation on heaps.	BTL1	Remembering
17	Differentiate between leftist and skew heaps.	BTL2	Understanding
18	List applications of heap structures.	BTL2	Understanding
19	What is the time complexity of heapify operation?	BTL2	Understanding
20	Define decrease-key operation in a heap.	BTL1	Remembering
21	Mention the advantages of Fibonacci heaps over binomial heaps.	BTL2	Understanding
22	Explain the structure of a binomial tree.	BTL2	Understanding
23	Define amortized cost in heap operations.	BTL2	Understanding
24	What is Lazy merging Heap?	BTL2	Understanding

PART – B			
1	Demonstrate the procedure for constructing a max-heap from the keys {5, 3, 17, 10, 84, 19, 6, 22, 9}. Show all intermediate steps.	BTL3	Applying
2	Illustrate the insertion and deletion operations in a min-heap with suitable examples. Show how the heap property is maintained.	BTL3	Applying
3	Implement the heap-sort algorithm and determine its time complexity for best, average, and worst cases.	BTL3	Applying
4	Examine the process of building a heap using the bottom-up approach. Compare its complexity with repeated insertion.	BTL4	Analyzing
5	Differentiate between binary, leftist, and skew heaps in terms of structure, height, and merge performance.	BTL4	Analyzing
6	Investigate how binomial heaps are formed from binomial trees and analyze their merge and delete-min operations.	BTL4	Analyzing
7	Assess the relative efficiency of binomial and Fibonacci heaps for priority-queue operations.	BTL5	Evaluating
8	Evaluate the impact of lazy merging in lazy binomial heaps on amortized running time.	BTL5	Evaluating
9	Judge the importance of amortized analysis in evaluating the performance of Fibonacci heaps.	BTL5	Evaluating
10	Design an efficient priority-queue algorithm using binomial heaps and explain each operation.	BTL6	Creating
11	Construct pseudocode for merging two leftist heaps and explain how rank ensures balancing.	BTL6	Creating
12	Develop an algorithm to convert a max-heap into a min-heap, and analyze its complexity.	BTL6	Creating
13	Apply Fibonacci heaps to implement Dijkstra's shortest-path algorithm and analyze its time efficiency.	BTL3	Applying
14	Analyze how skew heaps maintain balance without storing rank information.	BTL4	Analyzing
15	Assess the suitability of different heap structures (binary, binomial, Fibonacci) for real-time applications.	BTL5	Evaluating
16	Propose a memory-management technique that utilizes heap structures for efficient allocation and deallocation.	BTL6	Creating
17	Design an experimental comparison of the insertion operation times among binary, binomial, and Fibonacci heaps and interpret the results.	BTL6	Creating

UNIT IV – GEOMETRIC ALGORITHMS

(Segment Trees – 1-Dimensional Range Searching – k-d Trees – Line Segment Intersection – Computing the Overlay of Two Subdivisions – Range Trees – Voronoi Diagram)

Q. No.		BT Level	Competence
1	Define a geometric algorithm.	BTL1	Remembering
2	What is a segment tree?	BTL1	Remembering
3	List the operations supported by a segment tree.	BTL1	Remembering
4	Define range searching.	BTL1	Remembering
5	What is a 1-dimensional range query?	BTL2	Understanding
6	Explain the concept of 1-dimensional range searching with an example.	BTL2	Understanding
7	Define a k-d tree.	BTL1	Remembering
8	List the properties of a k-d tree.	BTL1	Remembering
9	What is the difference between a binary search tree and a k-d tree?	BTL2	Understanding
10	Define line segment intersection.	BTL1	Remembering
11	Mention applications of line segment intersection algorithms.	BTL2	Understanding
12	What is meant by the overlay of two subdivisions?	BTL1	Remembering
13	State the use of sweep line technique in computational geometry.	BTL2	Understanding
14	Define a range tree.	BTL1	Remembering
15	List the advantages of range trees in multidimensional searching.	BTL2	Understanding
16	What is a Voronoi diagram?	BTL1	Remembering
17	Mention the applications of Voronoi diagrams.	BTL2	Understanding
18	Define nearest neighbor searching.	BTL1	Remembering
19	What is meant by computational geometry?	BTL1	Remembering
20	Differentiate between static and dynamic geometric data structures.	BTL2	Understanding
21	Define plane sweep algorithm.	BTL1	Remembering
22	What is the significance of event points in line segment intersection algorithms?	BTL2	Understanding
23	Define spatial data structure.	BTL1	Remembering
24	What is the importance of geometric algorithms in computer graphics?	BTL2	Understanding

PART-B			
1	Demonstrate the steps for constructing a segment tree for a given set of intervals and illustrate its range query operation.	BTL3	Applying
2	Illustrate the process of 1-dimensional range searching using segment trees with an example.	BTL3	Applying
3	Implement the insertion and deletion operations in a k-d tree and analyze their complexities.	BTL3	Applying
4	Examine the procedure for building a k-d tree and analyze its use in multidimensional searching.	BTL4	Analyzing
5	Differentiate between segment trees and range trees in terms of structure and query performance.	BTL4	Analyzing
6	Analyze the working of line segment intersection algorithms using the sweep line method.	BTL4	Analyzing
7	Interpret how event points are used in detecting line segment intersections with examples.	BTL4	Analyzing
8	Compare the efficiency of k-d trees and range trees for multidimensional range queries.	BTL4	Analyzing
9	Assess the role of segment trees in solving real-time range query problems such as network data analysis or graphics rendering.	BTL5	Evaluating
10	Evaluate the performance of sweep line algorithms in computing the overlay of two subdivisions.	BTL5	Evaluating
11	Judge the efficiency and limitations of Voronoi diagram algorithms in nearest neighbor searching.	BTL5	Evaluating
12	Design an algorithm for detecting line segment intersections using the plane sweep technique.	BTL6	Creating
13	Construct a k-d tree for the given 2D points and perform range searching operations.	BTL6	Creating
14	Develop pseudocode for computing the overlay of two subdivisions and explain its complexity.	BTL6	Creating
15	Propose a range tree-based algorithm for 2D range searching and justify its efficiency.	BTL6	Creating
16	Design a Voronoi diagram construction algorithm for a given set of points and discuss its applications in computational geometry.	BTL6	Creating
17	Formulate a comparative study between segment trees, range trees, and k-d trees based on their applications, space, and time efficiency.	BTL6	Creating

UNIT V – ADDITIONAL TOPICS			
(Approximation Algorithms – Vertex Cover & Euclidean Travelling Salesperson Problem – Randomized Algorithms – Closest Pair Problem & Minimum Spanning Trees – Online Algorithm: Euclidean Spanning Tree)			
PART – A			
Q. No.	Questions	BT Level	Competence
1	Define an approximation algorithm.	BTL1	Remembering
2	What is meant by a randomized algorithm?	BTL1	Remembering
3	Define the term “vertex cover problem.”	BTL1	Remembering
4	What is the Euclidean Travelling Salesperson Problem (TSP)?	BTL1	Remembering
5	List the characteristics of approximation algorithms.	BTL2	Understanding
6	Explain the importance of approximation algorithms in NP-hard problems.	BTL2	Understanding
7	What is an online algorithm?	BTL1	Remembering
8	Define the Euclidean spanning tree.	BTL1	Remembering
9	What is the closest pair problem?	BTL1	Remembering
10	Differentiate between exact algorithms and approximation algorithms.	BTL2	Understanding
11	What is a randomized quicksort?	BTL1	Remembering
12	Define probabilistic analysis of an algorithm.	BTL2	Understanding
13	Explain the term “competitive ratio” in online algorithms.	BTL2	Understanding
14	What is the difference between deterministic and randomized algorithms?	BTL2	Understanding
15	Define the Monte Carlo algorithm.	BTL1	Remembering
16	Define the Las Vegas algorithm.	BTL1	Remembering
17	What are the applications of randomized algorithms?	BTL2	Understanding
18	State the approximation ratio of a vertex cover algorithm.	BTL1	Remembering
19	What is the purpose of a minimum spanning tree algorithm?	BTL2	Understanding
20	Define greedy algorithms in the context of MST.	BTL1	Remembering
21	Explain why approximation algorithms are used for NP-hard problems.	BTL2	Understanding
22	Define the probabilistic method in randomized algorithms.	BTL2	Understanding
23	What is the relationship between approximation and randomized algorithms?	BTL2	Understanding
24	Explain the role of online algorithms in dynamic environments.	BTL2	Understanding

PART – B			
1	Demonstrate the working of an approximation algorithm for the vertex cover problem with an example.	BTL3	Applying
2	Illustrate how an approximation algorithm can be applied to the Travelling Salesperson Problem (TSP).	BTL3	Applying
3	Implement a randomized algorithm to find the closest pair of points in a plane and discuss its efficiency.	BTL3	Applying
4	Apply the concept of randomized quicksort and compare it with the deterministic version.	BTL3	Applying
5	Examine the factors influencing the approximation ratio of vertex cover and TSP algorithms.	BTL4	Analyzing
6	Analyze the performance of randomized algorithms in comparison to deterministic algorithms.	BTL4	Analyzing
7	Differentiate between Monte Carlo and Las Vegas algorithms with appropriate examples.	BTL4	Analyzing
8	Investigate the competitive ratio concept in online algorithms using a suitable problem example.	BTL4	Analyzing
9	Assess the importance of randomization in improving the expected performance of algorithms.	BTL5	Evaluating
10	Evaluate the trade-off between accuracy and efficiency in approximation algorithms.	BTL5	Evaluating
11	Judge the effectiveness of online algorithms for real-time optimization problems.	BTL5	Evaluating
12	Design an approximation algorithm for the Euclidean TSP and determine its approximation ratio.	BTL6	Creating
13	Develop a randomized algorithm to find the minimum spanning tree of a given weighted graph.	BTL6	Creating
14	Construct pseudocode for an online Euclidean spanning tree algorithm.	BTL6	Creating
15	Propose a hybrid approach combining randomization and approximation for solving NP-hard problems.	BTL6	Creating
16	Formulate an evaluation metric to compare deterministic, randomized, and online algorithms.	BTL6	Creating
17	Design a simulation to demonstrate the impact of randomization on algorithm performance in large datasets.	BTL6	Creating