Reg. No. : [| | | | | | | | | | |]

Question Paper Code : 90918

B.E./B.Tech. DEGREE EXAMINATIONS, APRIL/MAY 2025.

Third Semester

Computer Science and Engineering

CS 3301 – DATA STRUCTURES

(Regulations 2021)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What is Radix sort and in what cases is it used?

2. How does a doubly linked list provide advantages over a singly linked list?

3. How does a stack manage function calls in programming?

4. What is the advantage of circular queue over linear queues?

5. What is a binary heap? State its properties and types.

6. List down the three types of tree traversal methods.

7. Define Euler circuits and state the conditions for a graph to have an Euler circuit.

8. What is a B+ Tree? How does it differ from a B-Tree?

9. What is selection sort? Is it a stable sort?

10. What is separate chaining in hashing? How does it handle collisions?

PART B — (5 × 13 = 65 marks)

11. (a) (i) Compare and construct linked list and array. (5)

(ii) Discuss the Abstract Data Type (ADT) concept in detail. Write down the ADT for polynomial representation and write algorithm for polynomial addition. (8)

Or

(b) (i) Write down the algorithms for insert, delete and search in a single linked list. (6)

(ii) What are Multilists? Explain the concept of multilists with an example, and discuss some practical applications where multilists are useful. (7)

12. (a) (i) Describe the applications of a stack data structure. Explain how a stack can be used for balancing symbols and evaluating arithmetic expressions. (6)

(ii) Define the Queue Abstract Data Type (ADT). Explain its main operations implementation using a linked representation (7)

Or

(b) (i) Describe the process of balancing symbols using a stack. Provide an example to illustrate how this is done. (6)

(ii) Define a deque (double-ended queue). Explain its operations and applications in real-world scenarios, giving examples of where it can be used effectively. (7)

13. (a) (i) Explain the Binary Search Tree Abstract Data Type (ADT) along with its key oeprations. Provide pseudocode and examples to illustrate each operation. (6)

(ii) Discuss the properties and implementation of AVL trees. Explain how they maintain balance and the rotations involved during insertion and deletion. (7)

Or

(b) (i) Define the priority queue and explain how it can be implemented using heaps. Discuss its applications in real-world scenarios. (6)

(ii) Discuss the concept of expression trees and describe the step-by-step process for constructing an expression tree from a given infix expression. (7)

14. (a) (i) Discuss the properties and advantages of B-Trees. (6)

(ii) Compare and contrast Prim's algorithm and Kruskal's algorithm for finding the minimum spanning tree. (7)

Or

(b) (i) Discuss the significance of topological sorting and how it can be implemented using depth-first search (DFS). (6)

(ii) Explain Dijkstra's algorithm in detail. Provide a step-by-step example to illustrate how it finds the shortest path in a weighted graph. (7)

15. (a) (i) Discuss shell sort, including its methodology and how it improves upon insertion sort. (6)

(ii) Explain the concept of open addressing in hashing. Describe its methods, including linear probing, quadratic probing, and double hashing and discuss their advantages and disadvantages. (7)

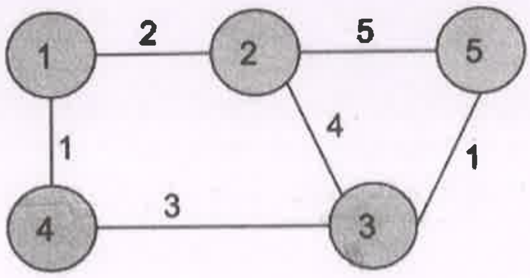
Or

(b) (i) Explain the concept of rehashing in hash tables. (6)

(ii) Compare and contrast linear search and binary search. Discuss the time complexities of both algorithms, including their best, average, and worst-case scenarios. Provide examples to illustrate when each search method would be most appropriate. (7)

PART C — (1 × 15 = 15 marks)

16. (a) Consider the weighted graph. Using Dijkstra's algorithm, determine the shortest path from vertex A to all other vertices. Illustrate the steps involved and show the final shortest path distances from vertex A to each vertex.



Or

(b) Construct an AVL Tree by inserting the following keys in the given order : 30, 20, 10, 25, 5, 15, and 35. Illustrate the structure of the AVL Tree after each insertion and indicate any necessary rotations to maintain the AVL property.