Chapter 1 : Introduction to Data Structures 1-1 to 1-28

Syllabus : Introduction to Data Structures : Linear and non Linear Data Structures, Static and Dynamic Data Structures.

1.1 Data 1-1
1.1.1 Data Types 1-1
1.1.2 Abstract Data Types (ADT) 1-1
1.1.3 Data Object 1-3
1.2 Data Structures 1-3
1.2.1 Types of Data Structures 1-3
1.2.1(A) Primitive and Non-Primitive 1-3
1.2.1 (B) Linear and Non-Linear 1-3
1.2.1(C) Static and Dynamic 1-4
1.3 Relationship among Data Object, Data Type, Data Structure and Data Representation 1-5
1.3.1 Operations on Data Structure 1-5
1.4 Algorithm Analysis 1-5
1.4.1 Measuring the Running Time of a Program (Time Complexity) 1-7
1.4.2 Measurement of Growth Rate (Asymptotic Growth Rate) 1-7
1.4.2(A) Asymptotic Consideration 1-7
1.4.2(B) Constant Factor in Complexity Measure 1-7
1.4.3 Notation O : (Pronounced as Big-Oh), (O(n2) is Pronounced as Big-Oh of n2) 1-8
1.4.4 Best Case, Worst Case and the Average Case Behaviour 1-9
1.5 Introduction to Arrays 1-10
1.6 Representation and Analysis 1-11
1.7 One-Dimensional Arrays 1-12
1.8 Operations with Arrays 1-13
1.8.1 Deletion 1-13
1.8.2 Insertion 1-14
1.8.3 Search 1-15
1.8.4 Merging of Sorted Arrays 1-16
1.9 Two-Dimensional Arrays 1-18
1.9.1 Initializing Two-Dimensional Arrays 1-19
1.9.2 Address Calculation 1-19
1.10 Multi-Dimensional Arrays 1-20
1.11 Application of Arrays 1-21
1.11.1 Addition of Two 2-D Matrices 1-21
1.11.2 Transpose of Square Matrix 1-23
1.11.3 Finding whether a given Square Matrix is Symmetrical 1-24
1.11.4 Multiplication of Two Matrices $A_{m \times n}$ and $B_{n \times p}$ 1-25
1.12 University Questions and Answers 1-27

Chapter 2 : Stacks and Queues 2-1 to 2-63

Syllabus : Concept of stack and queue. Array Implementation of Stack and Queue, Circular Queue, Double Ended Queue, priority Queue.

2.1 Introduction 2-1
2.2 Operations on Stacks 2-1
2.3 Array Representation 2-1
2.3.1 'C' Functions for Primitive Operations on a Stack 2-2
2.3.2 Program Showing Stack Operations 2-2
2.3.3 Well-Formedness of Parenthesis 2-5
2.3.4 Operations on Stack Considering Overflow and Underflow 2-6
2.3.5 Stack as an ADT 2-7
2.4 Applications of Stack 2-7
2.4.1 Expression Representation 2-7
2.4.2 Evaluation of a Postfix Expression using a Stack 2-8
2.4.3 Conversion of an Expression from Infix to Postfix 2-12
2.5 Expression Conversion (A Fast Method) 2-22
2.5.1 Infix to Postfix 2-22
2.5.2 Algorithm to Check Well-Formedness of Parenthesis 2-23
2.6 Introduction to Recursion 2-27

2.7 Converting a Recursive Function to an Equivalent C-Function..2-27

2.7.1 Finding Factorial of an Integer Number...................2-27

2.7.2 Finding nth Term of Fibonacci Sequence Recursive Definition 2-27

2.7.3 Finding GCD of given Numbers................................2-28

2.7.4 Calculation of xn using Recursion............................2-28

2.7.5 Calculation of Sum of Digits.....................................2-29

2.8 Examples of Recursion 2-29

2.8.1 Finding Sum of the Elements Stored in an Array...2-29

2.8.1(A) 'C' Function for Finding Sum of the Elements of an Array..2-29

2.8.2 Finding Length of a String..2-29

2.8.3 Reversing a String..2-29

2.8.4 Searching a Number in an Array..............................2-30

2.8.5 Finding Largest Element in an Array........................2-30

2.8.6 Binary Search..2-30

2.8.7 Tower of Hanoi Problem 2-30

2.9 Solved Examples...2-32

2.10 Removal of Recursion...2-35

2.11 Tail Recursion 2-35

2.12 Array and Linked Representation and Implementation of Queues...............................2-37

2.12.1 Definition..2-37

2.12.2 Application of Queues 2-38

2.12.3 Array Representation and Implementation of Queues 2-38

2.13 Operations on Queue 2-39

2.13.1 Operations on Queue Implemented using Array......2-39

2.14 Circular Queues 2-43

2.14.1 Queue using a Circular Array 2-43

2.14.1(A) Implementation of a Circular Movement Inside a Linear Array 2-44

2.15 Applications of Queue 2-51

2.15.1 Categorizing Data 2-51

2.15.2 Job Scheduling...2-51

2.15.3 Queue Simulation..2-52

2.16 Priority Queue 2-52

2.16.1 Implementation of Priority Queues.........................2-53

2.16.1(A) Implementation of a Priority Queue using a Circular Array 2-53

2.16.2 Dequeues 2-57

2.16.3 Implementation of Dequeue using a Circular Array...2-59

2.17 University Questions and Answers 2-62

Chapter 3 : Linked List 3-1 to 3-60

Syllabus : Concept of Linked Lists, Singly linked lists, doubly linked lists and circular linked lists. Insertion, deletion, update and copying operations with Singly linked lists, doubly linked lists and circular linked lists. Reversing a singly linked list.

3.1 Representation and Implementation of Singly Linked Lists..3-1

3.1.1 Comparison between Array and Linked Lists.............3-1

3.1.2 Representation..3-1

3.1.3 Implementation..3-2

3.1.4 Types of Linked List 3-3

3.1.4(A) Singly Linked List...3-3

3.1.4(B) Doubly Linked List 3-3

3.1.4(C) A Circular Linked List 3-3

3.1.5 Differences between Singly Linked List and Doubly Linked List 3-3

3.2 Basic Linked List Operations......................................3-4

3.2.1 Creating a Linked List..3-4

3.2.2 Traversing a Linked List 3-5

3.2.3 Counting Number of Nodes in a Linked List through Count Function......................................3-5

3.2.4 Printing a List through Print Function........................3-6

3.2.5 Inserting an Item 3-6

3.2.5(A) Inserting an Item at the End of a Linked List 3-7

3.2.5(B) Inserting a Data 'x' at a given Location 'LOC' in a Linked List, Referenced by 'head' 3-8

3.2.5(C) Inserting an Element in a Priority Linked List...........3-10

3.2.6 Deleting an Item 3-10

3.2.6(A) Deletion of the Last Node of a Linked List 3-11

3.2.6(B) Deletion of a Node at Location 'LOC' from a Linked List..3-12

3.2.6(C) Delete a Linked List, Referenced by the Pointer Head..3-13

3.2.7 Concatenation of Two Linked Lists 3-13

3.2.8 Inversion of Linked List...3-13

3.2.9 Searching a Data 'x' in a Linked List, Referenced by the Pointer Head 3-15

3.2.10 Searching an Element x in a Sorted Linked List.........3-16

3.2.11 New Linear Linked List by Selecting Alternate Element 3-16

3.2.12 Handling of Records through Linked List 3-17

3.2.13 Merging of Sorted Linked Lists 3-17

3.2.14 Splitting a Linked List at the Middle and Merge with Second Half as First Half..................................3-18

3.2.15 Removing Duplicate Elements from a Linked List.....3-19

3.3 Circular Linked List..3-23

3.3.1 Applications of Circular Linked List..........................3-29

3.4 Doubly Linked List...3-29

3.4.1 Creation of a Doubly Linked List..............................3-29

3.4.2 Deletion of a Node 3-32

3.5 Doubly Linked Circular List 3-37

3.6 Applications of Linked Lists 3-38

3.6.1 Polynomials as Linked Lists.....................................3-38

3.6.2 Addition of Two Polynomials..................................3-39

3.7 Linked Representation of a Stack 3-46

3.7.1 Functions for Stack Operations 3-48

3.8 Linked Representation of a Queue 3-50

3.8.1 Comparison between Array Representation and the Linked Representation of a Queue 3-51

3.8.2 Operations on Queue Implemented using Linked Structure 3-51

3.9 Queue using a Circular Linked List 3-55

3.9.1 Implementation of a Priority Queue using a Linked List 3-59

3.10 University Questions and Answers 3-60

Chapter 4 : Trees 4-1 to 4-97

Syllabus : Introduction to Trees : Terminology, Types of Binary trees. Non recursive Preorder, in-order and post-order traversal. Creation of binary trees from the traversal of binary trees. Binary search tree: Traversal, searching, insertion and deletion in binary search tree. Threaded Binary Tree: Finding in-order successor and predecessor of a node in threaded tree. Insertion and deletion in threaded binary tree. AVL Tree: Searching and traversing in AVL trees. Tree Rotations: Right Rotation, Left Rotation. Insertion and Deletion in an AVL Tree. B-tree: Searching, Insertion, Deletion from leaf node and non-leaf node. B+ Tree, Digital Search Tree, Game Tree & Decision Tree

4.1 Basic Terminology 4-1

4.1.1 Introduction 4-1

4.1.2 Basic Terms 4-1

4.2 Binary Tree 4-2

4.3 Representation of a Binary Tree using an Array 4-2

4.4 Linked Representation of a Binary Tree 4-3

4.4.1 Program for Creation of a Sample Binary Tree..........4-4

4.4.2 'C' Function for Creation of a Binary Tree 4-4

4.5 A General Tree..4-5

4.5.1 Node Declaration for a Tree 4-5

4.6 Types of Binary Tree...4-6

4.6.1 Full Binary Tree...4-6

4.6.2 Complete Binary Tree...4-7

4.6.3 Skewed Binary Tree 4-7

4.6.4 Strictly Binary Tree 4-7

4.6.5 Extended Binary Tree (2-Tree)..................................4-7

4.7 Binary Tree Traversal..4-7

4.7.1 Preorder Traversal (Recursive) 4-8

4.7.1(A) 'C' Function for Preorder Traversal 4-8

4.7.2 Inorder Traversal (Recursive) 4-9
4.7.2(A) 'C' Function for Inorder Traversal..............................4-9
4.7.3 Postorder Traversal (Recursive) 4-9
4.7.3(A) 'C' Function for Postorder Traversal........................4-10
4.7.4 Non-Recursive Preorder Traversal...........................4-10
4.7.4(A) 'C' Function for Non-Recursive Preorder of Tree Along with the ADT Stack............................4-11
4.7.5 Non-Recursive Inorder Traversal.............................4-11
4.7.5(A) 'C' Function for Non-Recursive Inorder Traversal of a Binary Tree.......................................4-12
4.7.6 Non-Recursive Postorder Traversal.........................4-12
4.7.6(A) 'C' Function for Non-Recursive Postorder Traversal 4-14
4.7.7 Tree Traversal Examples..4-15
4.8 Basic Tree Operations...4-17
4.8.1 'C' Function for Counting of Nodes in a Tree...........4-17
4.8.2 'C' Function for Counting of Leaf Nodes in a Tree (Recursive)....................................4-18
4.8.3 'C' Function for Counting of Nodes of Degree 1 (Recursive) 4-18
4.8.4 'C' Function for Counting of Nodes of Degree 2 (Recursive) 4-18
4.8.5 'C' Function to Create an Exact Copy of a Tree (Recursive).......................................4-18
4.8.6 'C' Function for Checking Equivalence of Two Binary Trees..4-19
4.8.7 'C' Function for Finding Height of a Tree (Recursive)..4-19
4.8.8 'C' Function for Swapping of Left and Right Children of Every Node (Mirror)..............................4-19
4.8.9 Finding Width of a Tree 4-19
4.8.10 Function to List the DATA Fields of the Node of a Binary Tree T by Level. Within Levels Nodes are Listed Left to Right.......................4-20
4.8.11 Non-Recursive Algorithm for Height of a Binary Tree..4-21
4.8.11(A) 'C' Function for Height of a Tree (Non-Recursive)..4-21
4.9 Creation of a Binary Tree from Traversal Sequence...4-22
4.9.1 Creation of Binary Tree from Preorder and Inorder Traversals 4-22
4.9.2 Creation of Tree from Postorder and Inorder Traversal 4-22
4.9.3 Examples on Tree Creation from Traversal Sequence...4-22
4.10 Binary Search Tree (BST) 4-26
4.10.1 Definition...4-26
4.10.2 Operations on a Binary Search Tree 4-26
4.10.2(A) Initialize Operation..4-26
4.10.2(B) Find Operation 4-26
4.10.2(C) Make Empty Operation 4-27
4.10.2(D) Insert Operation 4-27
4.10.2(E) Example on Creation of a BST 4-28
4.10.2(F) Delete Operation...4-28
4.10.2(G) Create..4-30
4.10.2(H) Find Min 4-31
4.10.2(I) Find Max...4-31
4.10.3 Program for Various Operations on BST..................4-31
4.11 Threaded Binary Trees (TBT) 4-35
4.11.1 Preorder Traversal (TBT) 4-39
4.11.2 Inorder Traversal of a TBT 4-39
4.11.3 Postorder Traversal of a TBT 4-40
4.11.3(A) 'C++' Function for Postorder Traversal of TBT...4-40
4.11.4 Insertion of Nodes in Inorder Threaded Binary Tree 4-41
4.11.5 Advantages and Disadvantages of a TBT 4-42
4.11.6 Deletion of a Node from TBT...................................4-42
4.11.6(A) Algorithm for Deletion of Node from TBT 4-43

4.11.7 Algorithm for Threading an Existing Tree (BST) 4-43

4.11.8 Class Definition of TBT 4-44

4.12 AVL Trees 4-47

4.12.1 Height Balanced Tree 4-48

4.12.2 Balance Factor 4-48

4.12.3 Structure of a Node in AVL Tree 4-49

4.12.4 'C' Function for Finding the Balance Factor of a Node 4-49

4.12.5 Insertion of a Node into an AVL Tree 4-49

4.12.5(A) Rotate Left 4-50

4.12.5(B) Rotate Right 4-51

4.12.5(C) Single Rotation and Double Rotation 4-52

4.12.5(D) 'C' Function for Insertion of an Element into an AVL Tree 4-72

4.12.5(E) 'C' Function to Find Height of AVL Tree 4-72

4.12.5(F) 'C' Function to Rotate Right 4-72

4.12.5(G) 'C' Function to Rotate Left 4-73

4.12.5(H) 'C' Function for RR 4-73

4.12.5(I) 'C' Function for LL 4-73

4.12.5(J) 'C' Function for LR 4-73

4.12.5(K) 'C' Function for RL 4-73

4.13 Application of Trees 4-73

4.13.1 Expression Trees 4-73

4.13.2 Program on Expression Tree from Postfix Expression 4-74

4.13.3 Conversion of an Expression into Binary Tree 4-75

4.13.4 Construction of an Expression Tree from Infix Expression 4-77

4.14 B-Trees 4-79

4.14.1 Insertion of a Key into a B-tree 4-80

4.14.2 Deleting a Value from a B-tree 4-86

4.14.3 B-tree as an ADT 4-90

4.15 B+ Trees 4-94

4.16 University Questions and Answers 4-96

Chapter 5 : Graphs 5-1 to 5-23

Syllabus : Introduction to Graphs: Undirected Graph, Directed Graph, graph terminology, Connectivity in Undirected and Directed Graphs. Spanning tree. Representation of graph: adjacency matrix, adjacency list, Transitive closure of a directed graph and path matrix. Traversals: Breadth First Search, Depth First Search.

5.1 Terminology and Representation 5-1

5.1.1 Definition 5-1

5.1.2 Undirected Graph 5-1

5.1.3 Directed Graph 5-1

5.1.4 A Complete Graph 5-2

5.1.5 Weighted Graph 5-2

5.1.6 Adjacent Nodes 5-2

5.1.7 Path 5-2

5.1.8 Cycle 5-2

5.1.9 Connected Graph 5-2

5.1.10 Subgraph 5-3

5.1.11 Component 5-3

5.1.12 Degree of a Vertex 5-3

5.1.13 Self Edges or Self Loops 5-3

5.1.14 Multigraph 5-3

5.1.15 Tree 5-3

5.1.16 Spanning Trees 5-4

5.1.17 Minimal Spanning Tree 5-4

5.2 Representation of Graphs 5-4

5.2.1 Adjacency Matrix 5-4

5.2.2 Adjacency List 5-5

5.2.3 Path Matrix 5-11

5.3 Traversal of Graphs 5-12

5.3.1 Depth First Search (DFS) 5-12

5.3.1(A) Algorithm for Depth First Search (Recursive) 5-12

5.3.1(B) Non-Recursive DFS Traversal 5-15

5.3.2 Breadth First Search(BFS) 5-17

5.3.2(A) Algorithm for BFS 5-17
5.4 University Questions and Answers 5-23

Chapter 6 : Recursion and Storage Management 6-1 to 6-8

Syllabus : Recursion: Writing a recursive function, Flow of control in recursive functions, Winding and unwinding phase, Recursive data structures, Implementation of recursion. Tail recursion. Indirect and Direct Recursion. Storage Management: Sequential Fit Methods: First Fit, Best Fit and Worst Fit methods. Fragmentation, Freeing Memory, Boundary Tag Method. Buddy Systems: Binary Buddy System, Fibonacci Buddy System. Compaction, Garbage Collection.

6.1 Recursion 6-1
6.2 Converting a Recursive Function to an Equivalent C-Function 6-1
6.2.1 Finding Factorial of an Integer Number 6-1
6.2.2 Finding nth term of Fibonacci Sequence 6-1
6.3 Examples of Recursion 6-1
6.3.1 Finding Sum of the Elements Stored in an Array 6-1
6.3.1(A) 'C' Function for Finding Sum of the Elements of An Array 6-1
6.3.2 Finding Length of a String 6-1
6.3.3 Reversing a String 6-1
6.3.4 Searching a Number in an Array 6-1
6.3.5 Finding Largest Element in an Array 6-1
6.3.6 Binary Search 6-1
6.3.7 Tower of Hanoi Problem 6-1
6.4 Backtracking 6-2
6.5 Variants of Recursion 6-2
6.6 Backtracking Algorithmic Strategy 6-3
6.6.1 Use of Stack in Backtracking 6-3
6.6.2 Removal of Recursion 6-3
6.7 Storage Management 6-3
6.7.1 Static Storage Management 6-3
6.7.2 Dynamics Storage Management 6-4
6.8 Memory Representation 6-4
6.8.1 Boundary Tag System 6-4
6.9 Storage Allocations 6-5
6.9.1 First - Fit Allocation 6-5
6.9.1(A) Algorithm for First - Fit 6-6
6.9.2 Best – Fit Allocation 6-6
6.9.2(A) Algorithm for Best Fit 6-7
6.9.3 Worst Fit 6-8
6.9.4 Nest Fit 6-8
6.10 Garbage Collection and Compaction 6-8

Chapter 7 : Searching and Sorting 7-1 to 7-53

Syllabus : Searching: Sequential Search, Binary Search. Hashing: Hash Functions: Truncation, Mid-square Method, Folding Method, Division Method. Collision Resolution: Open Addressing: Linear Probing, Quadratic Probing, Double Hashing, Separate Chaining Bucket Hashing. Analysis of all searching techniques Sorting: Insertion sort, Selection sort, Merge sort, Quick sort and Radix sort. Analysis of all sorting techniques

7.1 Searching 7-1
7.2 Sequential Search 7-1
7.2.1 Sequential Search on a Sorted Array 7-2
7.3 Binary Search 7-3
7.4 Hash Tables 7-8
7.4.1 What is Hashing ? 7-8
7.4.2 Hash Table Data Structure 7-8
7.4.2(A) Open Hashing Data Structure 7-9
7.4.2(B) Closed Hashing Data Structure 7-9
7.4.3 Hashing Functions 7-9
7.4.3(A) Characteristics of a Good Hash Function 7-9
7.4.3(B) Division-Method 7-10
7.4.3(C) Midsquare Methods 7-10
7.4.3(D) Folding Method 7-10
7.4.3(E) Digit Analysis 7-10
7.4.3(F) Length Dependent Method 7-10
7.4.3(G) Algebraic Coding 7-10
7.4.3(H) Multiplicative Hashing 7-11

7.4.4 Collision Resolution Strategies (Synonym Resolution) 7-11
7.4.4(A) Separate Chaining....7-11
7.4.4(B) Open Addressing 7-12
7.4.4(C) Primary Clustering 7-21
7.5 Sorting 7-21
7.5.1 Sort Stability 7-21
7.5.2 Sort Efficiency....7-22
7.5.3 Passes 7-22
7.6 Insertion Sort....7-22
7.6.1 Sorting an Array of Strings using Insertion Sort....7-24
7.6.2 Sorting an Array of Records on the given key using Insertion Sort....7-25
7.7 Bubble Sort....7-26
7.8 Selection Sort 7-28
7.9 Quick Sort....7-29
7.9.1 Picking a Pivot 7-30
7.9.2 Partitioning....7-30
7.9.3 Running Time of Quick Sort....7-38
7.9.3(A) Worst-Case Analysis 7-38
7.9.3(B) Best-Case Analysis 7-38
7.9.3(C) Average-Case Analysis 7-38
7.9.4 Role of Pivot in Efficiency of Quick Sort....7-41
7.10 Two-Way Merge Sort 7-41
7.10.1 Merging 7-43
7.10.2 Analysis of Merge Sort....7-45
7.10.3 Non-Recursive Merge Sort 7-46
7.11 Radix Sort 7-47
7.11.1 Algorithm for Radix Sort....7-48
7.11.2 C- Function for Radix Sort....7-48
7.11.3 Analysis of Radix Sort 7-50
7.12 Comparison of Sorting Algorithms 7-50
7.13 Best-Case, Worst-Case and Average-Case Analysis of Sorting Algorithm....7-50
7.14 External Vs Internal Sorting....7-51
7.15 University Questions and Answers 7-52

Chapter 8 : Applications of Data Structures 8-1 to 8-49

Syllabus : Applications of Linked Lists: Addition of 2 Polynomials and Multiplication of 2 polynomials. Applications of Stacks: Reversal of a String, Checking validity of an expression containing nested parenthesis, Function calls, Polish Notation: Introduction to infix, prefix and postfix expressions and their evaluation and conversions. Application of Queues: Scheduling, Round Robin Scheduling Applications of Trees: Huffman Tree and Heap Sort. Applications of Graphs: Dijkstra's Algorithm, Minimum Spanning Tree: Prim's Algorithm, Kruskal's Algorithm.

8.1 Applications of Linked Lists 8-1
8.1.1 Polynomials as Linked Lists....8-1
8.1.2 Addition of Two Polynomials....8-1
8.2 Applications of Stack 8-1
8.2.1 Expression Representation....8-1
8.2.2 Evaluation of a Postfix Expression using a Stack 8-1
8.2.3 Conversion of an Expression from Infix to Postfix....8-1
8.2.4 Conversion of an Expression from Infix to Prefix....8-1
8.2.5 Expression Conversion (A Fast Method)....8-3
8.2.5(A) Infix to Postfix....8-3
8.2.5(B) Infix to Prefix 8-3
8.2.5(C) Postfix to Prefix 8-3
8.2.5(D) Prefix to Infix 8-4
8.2.6 Algorithm to Check Well-Formedness of Parenthesis....8-4
8.3 Applications of Queue 8-4
8.3.1 Categorizing Data 8-4
8.3.2 Job Scheduling....8-4
8.3.3 Queue Simulation....8-4
8.4 Application of Trees 8-4
8.4.1 Expression Trees....8-4
8.4.2 Program on Expression Tree from Postfix Expression....8-4
8.4.3 Conversion of an Expression into Binary Tree....8-4

8.4.4 Construction of an Expression Tree from Infix Expression 8-4

8.5 Huffman Algorithm 8-4

8.5.1 Huffman Codes 8-4

8.5.2 Representation of Binary Codes as a Binary Tree 8-5

8.5.3 Huffman's Algorithm 8-6

8.5.4 Program for Huffman Tree 8-10

8.6 Heap Sort 8-11

8.6.1 Heaps 8-11

8.6.2 Types of Heaps 8-12

8.6.3 Basic Heap Operations 8-12

8.6.3(A) Basic Operations for Max Heap 8-12

8.6.3(B) Heap Creation – A Better Approach 8-16

8.6.4 Applications of Heap 8-18

8.6.4(A) Sorting in Ascending Order (Algorithm) 8-18

8.6.4(B) Sorting in Ascending Order (Program) 8-18

8.6.4(C) Timing Complexity of Heap Sort 8-19

8.7 Minimum Cost Spanning Tree 8-23

8.7.1 Prim's Algorithm 8-23

8.7.1(A) 'C' Function for Finding the Minimum Cost Spanning Tree 8-25

8.7.1(B) Program for Prim's Algorithm 8-26

8.7.1(C) Timing Complexity of Prim's Algorithm 8-27

8.7.1(D) Examples on Prim's Algorithm 8-27

8.7.2 Kruskal's Algorithm 8-33

8.7.2(A) 'C' Function find() 8-34

8.7.2(B) 'C' Function for Finding Minimum Cost Spanning Tree of a Graph using Kruskal's Algorithm 8-35

8.7.2(C) Program for Kruskal's Algorithm 8-35

8.7.2(D) Timing Complexity of Kruskal's Algorithm 8-37

8.7.2(E) Comparison of Time Complexity of Prim's and Kruskal's Algorithm 8-37

8.7.2(F) Examples on Kruskal's Algorithm 8-37

8.8 Shortest Path Algorithm 8-43

8.8.1 Dijkstra's Algorithm 8-43

8.8.1(A) Timing Complexity 8-44

8.8.2 Program on Dijkstra's Algorithm 8-44

8.8.3 Examples on Dijkstra's Algorithm 8-45

8.9 University Questions and Answers 8-48

❑❑❑