



Unit - I

Chapter 1 : An Overview of C Language 1-1 to 1-7

1.1	C Language : An Overview	1-1
1.2	C Programming	1-1
1.2.1	Constants in C	1-1
1.2.2	Variables	1-2
1.2.3	Keywords in C	1-3
1.2.4	Comments	1-3
1.2.5	Use of \n	1-3
1.2.6	C Statements	1-3
1.2.7	Storing Data	1-4
1.2.8	Reading a Keyboard Input	1-4
1.2.9	C- Operators	1-5
1.2.10	Decision Making with if Statement	1-5
1.2.11	Looping in C	1-5
1.3	Overview of Data Types	1-6
1.3.1	Primary or Built in Data Types	1-6
1.3.1(A)	Size and Range of Basic Data Types	1-6
1.3.2	User - Defined Type	1-6
1.3.3	Derived Data Types	1-7
1.3.3(A)	Structure	1-7
1.3.3(B)	Union	1-7

Chapter 2 : Operators and Expressions in 'C'

2-1 to 2-9

2.1	Operators and Expressions	2-1
2.2	Arithmetic Operators	2-1
2.2.1	Converting Algebraic Expression to 'C' Expression	2-1
2.2.2	Integer Arithmetic	2-2
2.2.3	Real Arithmetic	2-2
2.2.4	Mixed-Mode Arithmetic	2-2
2.3	Relational Operators	2-2

2.3.1	An Arithmetic Expression Containing Relational Expression	2-2
2.4	Logical Operators	2-2
2.5	Assignment Operator	2-3
2.6	Increment and Decrement Operators	2-3
2.7	Conditional Operator	2-4
2.8	Type Casting Operators	2-4
2.9	Bitwise Operators	2-5
2.10	Special Operators	2-7
2.10.1	Comma Operator	2-7
2.10.2	sizeof Operator	2-8
2.11	Operator Precedence and Associativity	2-8

Chapter 3 : Control Structure in 'C' 3-1 to 3-14

3.1	Control Flow in 'C'	3-1
3.2	if () Statement	3-1
3.3	if () else Statement	3-2
3.3.1	Nested if Statements	3-3
3.4	Switch Statement	3-3
3.5	Conditional Operator Statement	3-4
3.6	goto Statement	3-4
3.7	while Statement	3-5
3.8	do while Statement	3-7
3.9	for Statement	3-7
3.10	Nesting of Loops	3-7
3.11	Break Statement to End the Loop	3-8
3.12	Using the Continue Statement	3-8
3.13	Functions and Program Structure in C	3-9
3.13.1	Program Structure in C	3-11
3.13.2	Form of a C Function	3-11
3.13.3	Declaring Functions using Prototypes	3-11
3.13.4	Passing Arguments to Function	3-12
3.13.5	Returning Values from Functions	3-12
3.13.6	Scope of a Function	3-12



3.14	Inline Functions	3-12	5.4.1	Deletion	5-4
3.15	Storage Class	3-13	5.4.2	Insertion	5-4
3.16	Macros	3-14	5.4.3	Search	5-6
3.16.1	Comparison between Macro and Function	3-14	5.4.4	Merging of Sorted Arrays	5-6
Chapter 4 : Algorithm and its Analysis 4-1 to 4-12			5.5	Two-Dimensional Arrays	5-8
4.1	Introduction to Algorithms	4-1	5.5.1	Initializing Two-Dimensional Arrays	5-9
4.1.1	Definition and Characteristics	4-1	5.5.2	Address Calculation	5-9
4.1.2	Algorithm Design Tools	4-3	5.6	Multi-Dimensional Arrays	5-11
4.1.2(A)	Flowcharting	4-3	5.7	Application of Arrays	5-11
4.1.2(B)	Pseudo-Language	4-3	5.7.1	Addition of Two 2-D Matrices	5-11
4.1.3	Relation between Data Structure and Algorithm	4-5	5.7.2	Transpose of Square Matrix	5-13
4.2	Program Development	4-5	5.7.3	Finding whether a given Square Matrix is Symmetrical	5-14
4.2.1	Different phases of creating program	4-5	5.7.4	Multiplication of Two Matrices $A_{m \times n}$ and $B_n \times p$	5-16
4.3	Algorithm Analysis	4-6	Chapter 6 : ADT and Recursion 6-1 to 6-17		
4.3.1	Measuring the Running Time of a Program (Time Complexity)	4-8	6.1	Data	6-1
4.3.2	Measurement of Growth Rate (Asymptotic Growth Rate)	4-8	6.1.1	Data Types	6-1
4.3.2(A)	Asymptotic Consideration	4-8	6.1.2	Abstract Data Types (ADT)	6-1
4.3.2(B)	Constant Factor in Complexity Measure	4-8	6.1.3	Data Object	6-3
4.3.3	Notation O : (Pronounced as Big-Oh), ($O(n^2)$ is Pronounced as Big-Oh of n^2)	4-9	6.2	Data Structures	6-3
4.3.4	Best Case, Worst Case and the Average Case Behaviour	4-11	6.2.1	Types of Data Structures	6-3
4.3.5	Different Algorithm Asymptotic Notation	4-12	6.2.1(A)	Primitive and Non-Primitive	6-3
4.3.5(A)	The Notation O (Big-Oh)	4-12	6.2.1(B)	Linear and Non-Linear	6-4
4.3.5(B)	The Notation Ω (Omega)	4-12	6.2.1(C)	Static and Dynamic	6-4
4.3.5(C)	The Notation θ (Theta)	4-12	6.3	Passing Parameters by Value	6-5
Chapter 5 : Arrays 5-1 to 5-18			6.3.1	Relationship among Data Object, Data Type, Data Structure and Data Representation	6-6
5.1	Introduction to Arrays	5-1	6.4	Introduction to Recursion	6-6
5.2	Representation and Analysis	5-1	6.5	Converting a Recursive Function to an Equivalent C- Function	6-6
5.3	One-Dimensional Arrays	5-2	6.5.1	Finding Factorial of an Integer Number	6-6
5.4	Operations with Arrays	5-3	6.5.2	Finding n^{th} Term of Fibonacci Sequence Recursive Definition	6-7



6.5.3	Finding GCD of given Numbers.....	6-8	7.6.2	Reversing a String.....	7-11
6.5.4	Calculation of x^n using Recursion.....	6-8	7.6.3	String Searching.....	7-11
6.6	Examples of Recursion.....	6-8	7.6.4	String Concatenation.....	7-12
6.6.1	Finding Sum of the Elements Stored in an Array.....	6-8	7.6.5	A Complete Program for String Manipulation (without Pointers).....	7-14
6.6.2	Finding Length of a String.....	6-8	7.6.6	A Complete Program for String Manipulation (with Pointers).....	7-18
6.6.3	Reversing a String.....	6-9	7.7	A Function Returning a Pointer.....	7-23
6.6.4	Searching a Number in an Array.....	6-9	7.7.1	Pointer to a Function.....	7-24
6.6.5	Finding Largest Element in an Array.....	6-9	7.8	Command Line Arguments.....	7-24
6.6.6	Binary Search.....	6-9	7.9	Structures.....	7-25
6.6.7	Tower of Hanoi Problem.....	6-10	7.9.1	Difference between Array and Structure.....	7-25
6.7	Solved Examples.....	6-12	7.9.2	Declaring Structures.....	7-25
6.8	Dynamic Memory Management.....	6-16	7.9.3	Structure as a Function Argument.....	7-26
6.8.1	Memory Management Function during Runtime.....	6-16	7.9.4	Array of Structures.....	7-26
Chapter 7 : Pointers and Structures in C 7-1 to 7-56			7.9.5	Nested Structure.....	7-29
7.1	Pointers in C.....	7-1	7.10	Unions.....	7-30
7.2	Declaring a Pointer.....	7-1	7.10.1	Structure Vs. Union.....	7-30
7.3	Accessing a Variable Through a Pointer.....	7-1	7.11	Enumerations.....	7-30
7.4	Advantages of Pointers.....	7-2	7.12	Pointer to Structure.....	7-31
7.5	Pointers to Array and Pointer Arithmetic.....	7-2	7.13	Dynamic Memory Allocation.....	7-36
7.5.1	Adding an Integer Value to a Pointer.....	7-3	7.14	An Array of Pointers.....	7-37
7.5.2	Subtracting an Integer Value from a Pointer.....	7-4	7.15	Ordered List.....	7-38
7.5.3	Difference of Two Pointers.....	7-4	7.15.1	Polynomial as an Ordered List.....	7-39
7.5.4	Operations not Permitted on Pointers.....	7-4	7.15.1(A)	Function for Initialisation of a Polynomial.....	7-39
7.5.5	Comparison of Two Pointers.....	7-4	7.15.1(B)	Insert a Term.....	7-39
7.5.6	Pointers as Function Arguments.....	7-4	7.15.1(C)	Reading a Polynomial.....	7-40
7.5.6(A)	Result of Passing Parameter by Value and by Reference.....	7-6	7.15.1(D)	Printing a Polynomial.....	7-40
7.5.7	Unidimensional Integer Arrays as Function Arguments.....	7-7	7.15.1(E)	Addition of Two Polynomials.....	7-40
7.5.8	Unidimensional Character Array as Function Arguments : String Functions.....	7-8	7.15.1(F)	Multiplication of Two Polynomials.....	7-41
7.6	Character String in C-Language.....	7-9	7.15.1(G)	Evaluation of a Polynomials.....	7-42
7.6.1	String Comparison.....	7-10	7.15.1(H)	Polynomial Representation in an Array.....	7-45
			7.15.2	Representing a Polynomial $A(x, y)$	7-45



7.15.3	Representing a Polynomial A (x, y, z).....	7-46	9.2.3	Sorting an Array of Strings using Insertion Sort.....	9-4
7.15.4	Various Methods of Representation of a Polynomial in 1 Variable.....	7-46	9.2.4	Sorting an Array of Records on the given Key using Insertion Sort.....	9-5
7.16	Introduction to Files.....	7-48	9.3	Bubble Sort.....	9-9
7.17	Files I/O in 'C'.....	7-48	9.4	Selection Sort.....	9-16
7.18	Compare Text File and Binary File.....	7-50	9.5	Two-Way Merge Sort.....	9-18
7.19	Basic File Operations.....	7-50	9.5.1	Merging.....	9-20
7.20	Sequential and Random Access Files.....	7-52	9.5.2	Analysis of Merge Sort.....	9-22
7.21	Comparison between Sequential File and Random Access File.....	7-53	9.5.3	Non-Recursive Merge Sort.....	9-23
7.22	Program for Various Operations on a Sequential File.....	7-54	9.6	Hash Tables.....	9-25
7.23	Binary Files.....	7-55	9.6.1	What is Hashing ?.....	9-25
			9.6.2	Hash Table Data Structure.....	9-26
			9.6.2(A)	Open Hashing Data Structure.....	9-26
			9.6.2(B)	Closed Hashing Data Structure.....	9-26
			9.7	Hashing Functions.....	9-26
			9.7.1	Characteristics of a Good Hash Function.....	9-27
			9.7.2	Division-Method.....	9-27
			9.7.3	Midsquare Methods.....	9-27
			9.7.4	Folding Method.....	9-27
			9.7.5	Digit Analysis.....	9-27
			9.7.6	Length Dependent Method.....	9-27
			9.7.7	Algebraic Coding.....	9-27
			9.7.8	Multiplicative Hashing.....	9-28
			9.8	Collision Resolution Strategies (Synonym Resolution).....	9-28
			9.8.1	Separate Chaining.....	9-28
			9.8.2	Open Addressing.....	9-29
			9.8.2(A)	Linear Probing.....	9-29
			9.8.2(B)	Quadratic Probing.....	9-32
			9.8.2(C)	Double Hashing.....	9-33
			9.8.3	Primary Clustering.....	9-34

Unit - II

Chapter 8 : Searching **8-1 to 8-10**

8.1	Searching.....	8-1
8.2	Sequential - Linear Search.....	8-1
8.2.1	Sequential Search on a Sorted Array.....	8-2
8.3	Binary Search.....	8-3
8.4	Fibonacci Search.....	8-8
8.4.1	Binary Search versus Fibonacci Search.....	8-8
8.4.2	Selection of Searching Algorithm.....	8-9
8.4.3	Timing Complexity of Fibonacci Search.....	8-10

Chapter 9 : Sorting **9-1 to 9-34**

9.1	Sorting.....	9-1
9.1.1	Sort Stability.....	9-1
9.1.2	Sort Efficiency.....	9-2
9.1.3	Passes.....	9-2
9.2	Insertion Sort.....	9-3
9.2.1	'C' Function for Insertion Sort.....	9-3
9.2.2	Time Complexity of Insertion Sort in Best Case and Worst Case.....	9-4



Unit - III

Chapter 10 : Stack	10-1 to 10-33
---------------------------	----------------------

10.1	Introduction	10-1
10.2	Operations on Stacks	10-1
10.3	Array Representation.....	10-1
10.3.1	'C' Functions for Primitive Operations on a Stack.....	10-1
10.3.2	Program Showing Stack Operations (Reversing Data)	10-2
10.3.3	Well Formedness of Parenthesis.....	10-5
10.3.4	Operations on Stack Considering Overflow and Underflow	10-5
10.3.5	Stack as an ADT	10-6
10.4	Applications of Stack.....	10-6
10.4.1	Expression Representation.....	10-6
10.4.2	Evaluation of a Postfix Expression using a Stack.....	10-7
10.4.3	Evaluation of a Prefix Expression	10-12
10.4.3(A)	Conversion of an Expression from Infix to Postfix.....	10-12
10.4.4	Conversion of an Expression from Infix to Prefix.....	10-22
10.4.5	Conversion of Expression from Postfix to Infix.....	10-24
10.4.6	Conversion of Expression from Postfix to Prefix	10-24
10.4.7	Expression Conversion (A Fast Method)	10-24
10.4.7(A)	Infix to Postfix.....	10-24
10.4.7(B)	Infix to Prefix.....	10-25
10.4.7(C)	Postfix to Prefix	10-25
10.4.7(D)	Prefix to Infix.....	10-25
10.4.8	Algorithm to Check Well-Formedness of Parenthesis	10-25
10.5	Removal of Recursion	10-31
10.6	Tail Recursion	10-31

Chapter 11 : Queue	11-1 to 11-19
---------------------------	----------------------

11.1	Array and Linked Representation and Implementation of Queues	11-1
11.1.1	Definition	11-1
11.1.2	Application of Queues	11-1
11.1.3	Array Representation and Implementation of Queues	11-1
11.1.4	Linked Representation of a Queue	11-2
11.1.5	Comparison between Array Representation and the Linked Representation of a Queue	11-3
11.2	Operations on Queue	11-3
11.2.1	Operations on Queue Implemented using Array	11-4
11.2.1(A)	Realization of Queue Operations through 'C' Function	11-4
11.2.1(B)	'C' Function to initialize()	11-4
11.2.1(C)	'C' Function to Check whether Queue is Empty or Not	11-4
11.2.1(D)	'C' Function to Check whether Queue is Full or Not	11-4
11.2.1(E)	'C' Function for Insertion in a Queue using Array	11-4
11.2.1(F)	'C' Function for Insertion in a Queue	11-5
11.2.1(G)	'C' Function for Deletion in a Queue using Array	11-5
11.2.1(H)	'C' Function for Deletion in a Queue	11-5
11.3	Circular Queues	11-8
11.3.1	Queue using a Circular Array	11-8
11.3.1(A)	Implementation of a Circular Movement Inside a Linear Array.....	11-9
11.4	Applications of Queue	11-13
11.4.1	Categorizing Data.....	11-13
11.4.2	Job Scheduling.....	11-13
11.4.3	Queue Simulation.....	11-14
11.5	Priority Queue	11-14
11.5.1	Implementation of Priority Queues	11-15
11.5.1(A)	Implementation of a Priority Queue using a Circular Array	11-15



Unit - IV

Chapter 12 : Linked List**12-1 to 12-51**

<p>12.1 Representation and Implementation of Singly Linked Lists..... 12-1</p> <p>12.1.1 Comparison between Array (Sequential) and Linked Lists 12-1</p> <p>12.1.2 Representation 12-1</p> <p>12.1.3 Implementation..... 12-2</p> <p>12.1.4 Types of Linked List 12-3</p> <p>12.1.4(A) Singly Linked List 12-3</p> <p>12.1.4(B) Doubly Linked List 12-3</p> <p>12.1.4(C) Circular Linked List..... 12-3</p> <p>12.2 Basic Linked List Operations 12-4</p> <p>12.2.1 Creating a Linked List..... 12-4</p> <p>12.2.2 Traversing a Linked List 12-5</p> <p>12.2.3 Counting Number of Nodes in a Linked List through Count Function 12-5</p> <p>12.2.3(A) A Recursive C-Function to Count Nodes in DLL/SLL..... 12-6</p> <p>12.2.4 Printing a List through Print Function..... 12-7</p> <p>12.2.5 Inserting an Item 12-7</p> <p>12.2.5(A) Inserting an Item at the End of a Linked List..... 12-8</p> <p>12.2.5(B) Inserting a Data 'x' at a given Location 'LOC' in a Linked List, Referenced by 'Head'..... 12-8</p> <p>12.2.5(C) Inserting an Element in a Priority Linked List..... 12-9</p> <p>12.2.6 Deleting an Item 12-10</p> <p>12.2.6(A) Deletion of the Last Node of a Linked List 12-10</p> <p>12.2.6(B) Deletion of a Node at Location 'LOC' from a Linked List 12-11</p> <p>12.2.6(C) Delete a Linked List, Referenced by the Pointer Head..... 12-12</p> <p>12.2.6(D) C Function to Traverse a Linked List in Reverse Order..... 12-12</p>	<p>12.2.6(E) C Function to Sort Data Stored in a Linked List (SLL / DLL)..... 12-12</p> <p>12.2.7 Concatenation of Two Linked Lists 12-13</p> <p>12.2.8 Inversion of Linked List..... 12-14</p> <p>12.2.9 Searching a Data 'x' in a Linked List, Referenced by the Pointer Head..... 12-15</p> <p>12.2.10 Searching an Element x in a Sorted Linked List 12-16</p> <p>12.2.11 New Linear Linked List by Selecting Alternate Element 12-16</p> <p>12.2.12 Handling of Records through Linked List 12-17</p> <p>12.2.13 Merging of Sorted Linked List 12-17</p> <p>12.2.14 Splitting a Linked List at the Middle and Merge with Second Half as First Half 12-19</p> <p>12.2.15 Removing Duplicate Elements from a Linked List.... 12-20</p> <p>12.3 Linked List in Array 12-21</p> <p>12.4 Circular Linked List..... 12-21</p> <p>12.4.1 'C' Function for Inserting a Number at the Rear of a Circular Linked List..... 12-22</p> <p>12.4.2 'C' Function for Inserting a Number at the Front of the Circular Linked List..... 12-23</p> <p>12.4.3 'C' Function for Traversing a Circular Linked List..... 12-23</p> <p>12.4.4 'C' Function to Insert and Delete a Nth Element in Circular Linked List..... 12-23</p> <p>12.5 Doubly Linked List 12-25</p> <p>12.5.1 Creation of a Doubly Linked List..... 12-25</p> <p>12.5.1(A) 'C' Function for Inserting a Node Pointed to by P after a Node Pointed to by q..... 12-27</p> <p>12.5.1(B) 'C' Function for Inserting a Node Pointed to by P before a Node Pointed to by q..... 12-27</p> <p>12.5.1(C) 'C' Function for Inserting a Value x, at the Beginning of Doubly Linked List 12-28</p> <p>12.5.1(D) 'C' Function for Inserting a Value x, at the End of a Doubly Linked List 12-28</p> <p>12.5.1(E) 'C' Function for Inserting a Node at a given Position in a DLL 12-28</p> <p>12.5.2 Deletion of a Node..... 12-28</p>
---	---



12.5.2(A) 'C' Function for Deleting a Node from the given Position in a DLL	12-29	13.6.2 Inorder Traversal	13-10
12.5.2(B) 'C' Function to Delete a Node of DLL with Data Value x.....	12-29	13.6.3 Postorder Traversal	13-11
12.5.3 Application of Doubly Linked List.....	12-30	13.6.4 Non-recursive Preorder Traversal.....	13-11
12.6 Doubly Linked Circular List.....	12-31	13.6.5 Non-recursive Inorder Traversal	13-12
12.7 Applications of Linked Lists	12-33	13.6.6 Non-Recursive Postorder Traversal.....	13-13
12.7.1 Polynomials as Linked Lists	12-33	13.6.7 Tree Traversal Examples	13-16
12.8 Linked Representation of a Stack.....	12-40	13.7 Basic Tree Operations.....	13-20
12.8.1 Functions for Stack Operations	12-42	13.7.1 'C' Function for Counting of Nodes in a Tree	13-20
12.8.2 Operations on Queue Implemented using Linked Structure.....	12-43	13.7.2 'C' Function for Creation of a Tree.....	13-20
12.8.3 Queue using a Circular Linked List.....	12-47	13.7.3 'C' Function for Counting of Leaf Nodes in a Tree.....	13-20
		13.7.4 'C' Function for Counting of Nodes of Degree 1.....	13-20
		13.7.5 Non-Recursive Function to Count the Number of Leaf Nodes	13-21
		13.7.6 'C' Function for Counting of Nodes of Degree 2.....	13-21
		13.7.7 'C' Function to Create an Exact Copy of a Tree.....	13-21
		13.7.8 'C' Function for Checking Equivalence of Two Binary Trees	13-21
		13.7.9 'C' Function for Finding Height of a Tree	13-21
		13.7.10 Function to List the DATA Fields of the Node of a Binary Tree Level Wise	13-22
		13.7.11 Creation of Binary Tree from Preorder and Inorder Traversals	13-23
		13.8 Binary Search Tree (BST)	13-25
		13.8.1 Definition	13-25
		13.8.2 Operations on a Binary Search Tree	13-25
		13.8.2(A) Initialize Operation.....	13-26
		13.8.2(B) Find Operation.....	13-26
		13.8.2(C) Makeempty Operation	13-26
		13.8.2(D) Insert Operation.....	13-26
		13.8.2(E) Delete Operation	13-28
		13.8.2(F) Create	13-30
		13.8.2(G) FindMin	13-33
		13.8.2(H) FindMax	13-33

Unit - V

Chapter 13 : Trees

13-1 to 13-47

13.1 Basic Terminology.....	13-1
13.1.1 Introduction to Trees	13-1
13.1.2 Basic Terms	13-1
13.2 Binary Tree.....	13-2
13.2.1 Representation of a Binary Tree using an Array	13-2
13.3 Linked Representation of a Binary Tree.....	13-4
13.3.1 Height of a Node	13-5
13.3.2 Height of a Tree.....	13-5
13.4 A General Tree (Forest)	13-6
13.5 Types of Binary Tree	13-7
13.5.1 Full Binary Tree	13-7
13.5.2 Complete Binary Tree.....	13-8
13.5.3 Skewed Binary Tree	13-8
13.5.4 Strictly Binary Tree	13-8
13.5.5 Extended Binary Tree (2-Tree)	13-8
13.6 Binary Tree Traversal	13-8
13.6.1 Preorder Traversal.....	13-9



13.8.3	Binary Search Tree as an ADT	13-33
13.9	Application of Trees	13-45
13.9.1	Expression Trees	13-45
13.9.1(A)	Conversion of an Expression into Binary Tree	13-46
13.9.2	Game Tree	13-47

Unit - VI

Chapter 14 : Graphs **14-1 to 14-63**

14.1	Terminology and Representation	14-1
14.1.1	Definition of Graph	14-1
14.1.2	Undirected Graph	14-1
14.1.3	Directed Graph	14-1
14.1.4	A Complete Graph	14-2
14.1.5	Weighted Graph	14-2
14.1.6	Adjacent Nodes	14-2
14.1.7	Path	14-2
14.1.8	Cycle	14-2
14.1.9	Connected Graph	14-2
14.1.10	Subgraph	14-3
14.1.11	Component	14-3
14.1.12	Degree of a Vertex	14-3
14.1.13	Self Edges or Self Loops	14-3
14.1.14	Multigraph	14-3
14.1.15	Tree	14-3
14.1.16	Spanning Trees	14-4
14.1.17	Minimal Spanning Tree	14-4

14.2	Representation of Graphs	14-5
14.2.1	Adjacency Matrix	14-5
14.2.2	Adjacency List	14-6
14.2.3	Path Matrix	14-12
14.3	Traversal of Graphs	14-14
14.3.1	Depth First Search (DFS)	14-14
14.3.1(A)	Algorithm for DFS (Recursive)	14-14
14.3.1(B)	Non-Recursive DFS Traversal	14-17
14.3.2	Breadth First Search (BFS)	14-19
14.3.2(A)	Algorithm for BFS	14-20
14.3.2(B)	BFS and DFS Comparison	14-26
14.4	Minimum Cost Spanning Tree	14-26
14.4.1	Prim's Algorithm	14-26
14.4.1(A)	Timing Complexity of Prim's Algorithm	14-28
14.4.2	Kruskal's Algorithm	14-37
14.4.2(A)	'C' Function for Finding Minimum Cost Spanning Tree of a Graph using Kruskal's Algorithm	14-40
14.4.2(B)	Timing Complexity of Kruskal's Algorithm	14-56
14.4.2(C)	Comparison of Time Complexity of Prim's and Kruskal's Algorithm	14-56
14.5	Shortest Path Algorithm	14-56
14.5.1	Dijkstra Algorithm	14-56
14.5.1(A)	Timing Complexity	14-57
14.5.2	Program on Dijkstra's Algorithm	14-57
14.5.3	Examples on Dijkstra's Algorithm	14-58