

**UNIT - I****Chapter 1 : Hashing 1-1 to 1-21**

1.1	Hash Tables	1-1
1.1.1	What is Hashing ?	1-1
1.1.2	Hash Table Data Structure	1-2
1.1.2(A)	Open Hashing Data Structure.....	1-2
1.1.2(B)	Closed Hashing Data Structure	1-2
1.2	Terminology	1-2
1.2.1	Bucket	1-2
1.2.2	Collision.....	1-2
1.2.3	Probe	1-2
1.2.4	Synonym	1-3
1.2.5	Overflow	1-3
1.2.6	Perfect Hash Function.....	1-3
1.2.7	Load Density	1-3
1.2.8	Full Table	1-3
1.3	Hashing Functions.....	1-3
1.3.1	Characteristics of a Good Hash Function	1-4
1.3.2	Division-Method.....	1-4
1.3.3	Midsquare Methods.....	1-4
1.3.4	Folding Method.....	1-4
1.3.5	Digit Analysis.....	1-4
1.3.6	Length Dependent Method	1-5
1.3.7	Algebraic Coding	1-5
1.3.8	Multiplicative Hashing.....	1-5
1.4	Collision Resolution Strategies (Synonyms Resolutions).....	1-5
1.4.1	Separate Chaining.....	1-5
1.4.2	Open Addressing.....	1-6

1.4.2(A)	Linear Probing.....	1-6
1.4.2(i)	Linear Probing with Chaining (without Replacement)...	1-9
1.4.2(ii)	Linear Probing with Chaining (with Replacement).....	1-13
1.4.2(B)	Quadratic Probing	1-17
1.4.2(C)	Double Hashing.....	1-18
1.5	Rehashing	1-19
1.6	Extendible Hashing.....	1-19
1.7	Dictionary as ADT	1-20
1.8	Skip List	1-20
1.9	University Questions and Answers	1-21

UNIT - II**Chapter 2 : Trees 2-1 to 2-59**

2.1	Basic Terminology.....	2-1
2.1.1	Introduction	2-1
2.1.2	Basic Terms	2-1
2.2	Binary Tree.....	2-1
2.3	Representation of a Binary Tree using an Array	2-2
2.4	Linked Representation of a Binary Tree.....	2-2
2.4.1	Program for Creation of a Sample Binary Tree.....	2-3
2.4.2	C++ Function for Creation of a Binary Tree	2-4
2.5	A General Tree.....	2-4
2.5.1	Node Declaration for a Tree	2-4
2.6	Types of Binary Tree	2-6
2.6.1	Full Binary Tree	2-7
2.6.2	Complete Binary Tree.....	2-7
2.6.3	Skewed Binary Tree	2-7
2.6.4	Strictly Binary Tree	2-7
2.6.5	Extended Binary Tree (2-Tree)	2-7
2.7	Binary Tree Traversal (DFS).....	2-7
2.7.1	Preorder Traversal (Recursive).....	2-8



2.7.1(A) 'C++' Function for Preorder Traversal.....	2-8	2.9	Creation of a Binary Tree from Traversal Sequence	2-20
2.7.2 Inorder Traversal (Recursive)	2-8	2.9.1	Creation of Binary Tree from Preorder and Inorder Traversals	2-20
2.7.2(A) 'C++' Function for Inorder Traversal	2-9	2.9.2	Creation of Tree from Postorder and Inorder Traversal	2-20
2.7.3 Postorder Traversal (Recursive)	2-9	2.9.3	Examples on Tree Creation from Traversal Sequence	2-20
2.7.3(A) 'C++' Function for Postorder Traversal	2-9	2.9.4	Properties of a Tree (Solved Examples)	2-22
2.7.4 Non-Recursive Preorder Traversal	2-9	2.10	Binary Search Tree (BST)	2-28
2.7.4(A) 'C++' Function for Non-Recursive Preorder of Tree along with the ADT Stack.....	2-10	2.10.1	Definition	2-28
2.7.5 Non-Recursive Inorder Traversal.....	2-10	2.10.2	Operations on a Binary Search Tree	2-28
2.7.5(A) 'C++' Function for Non-Recursive Inorder Traversal of a Binary Tree.....	2-10	2.10.2(A)	Initialize Operation.....	2-28
2.7.6 Non-Recursive Postorder Traversal.....	2-11	2.10.2(B)	Find Operation	2-28
2.7.6(A) 'C++' Function for Non-Recursive Postorder Traversal	2-12	2.10.2(C)	Makeempty Operation.....	2-29
2.7.7 Tree Traversal Examples.....	2-13	2.10.2(D)	Insert Operation	2-29
2.8 Basic Tree Operations.....	2-15	2.10.2(E)	Delete Operation.....	2-31
2.8.1 'C++' Function for Counting of Nodes in a Tree	2-15	2.10.2(F)	Create	2-33
2.8.2 'C++' Function for Counting of Leaf Nodes in a Tree (Recursive).....	2-16	2.10.2(G)	FindMin	2-33
2.8.3 'C++' Function for Counting of Nodes of Degree 1 (Recursive).....	2-16	2.10.2(H)	FindMax	2-34
2.8.4 'C++' Function for Counting of Nodes of Degree 2 (Recursive).....	2-16	2.10.3	Class Definition for BST	2-34
2.8.5 'C++' Function to Create an Exact Copy of a Tree (Recursive).....	2-16	2.11	Threaded Binary Trees (TBT).....	2-41
2.8.6 'C++' Function for Checking Equivalence of Two Binary Trees.....	2-16	2.11.1	Preorder Traversal (TBT).....	2-44
2.8.7 'C++' Function for Finding Height of a Tree (Recursive).....	2-16	2.11.2	Inorder Traversal of a TBT	2-45
2.8.8 'C++' Function for Swapping of Left and Right Children of Every Node (Mirror)	2-17	2.11.3	Postorder Traversal of a TBT	2-45
2.8.9 Binary Tree Traversal (BFS).....	2-18	2.11.3(A)	'C++' Function for Postorder Traversal of TBT	2-46
2.8.10 Non-Recursive Algorithm for Height of a Binary Tree.....	2-19	2.11.4	Insertion of Nodes in Inorder Threaded Binary Tree.....	2-46
2.8.10(A) 'C' Function for Height of a Tree (Non-Recursive)	2-19	2.11.5	Advantages and Disadvantages of a TBT.....	2-47
		2.11.6	Deletion of a Node from TBT.....	2-47
		2.11.6(A)	Algorithm for Deletion of Node from TBT.....	2-48
		2.11.7	Algorithm for Threading an Existing Tree (BST)	2-48
		2.11.8	Class Definition of TBT.....	2-49
		2.12	Application of Trees.....	2-52



2.12.1	Expression Trees	2-52	3.2.1	Adjacency Matrix	3-5
2.12.2	Program on Expression Tree from Postfix Expression	2-53	3.2.2	Adjacency List	3-6
2.13	Huffman Algorithm.....	2-54	3.2.3	Multilist Representation of Graph	3-7
2.13.1	Huffman Codes	2-54	3.2.4	Inverse Adjacency List	3-8
2.13.2	Representation of Binary Codes as a Binary Tree	2-55	3.2.5	Examples on Graph Representation.....	3-8
2.13.3	Huffman's Tree.....	2-55	3.3	Traversal of Graphs.....	3-10
2.13.4	Program for Huffman Tree.....	2-56	3.3.1	Depth First Search (DFS)	3-10
2.14	University Questions and Answers	2-58	3.3.1(A)	Algorithm for DFS.....	3-10
UNIT - III			3.3.1(B)	Program for DFS using Adjacency Matrix	3-11
Chapter 3 : Graphs			3.3.1(C)	Program for DFS using Adjacency List.....	3-12
3-1 to 3-56			3.3.1(D)	Non-Recursive DFS Traversal	3-13
3.1	Terminology and Representation.....	3-1	3.3.1(E)	Example on DFS	3-13
3.1.1	Definition	3-1	3.3.1(F)	DFS Spanning Tree.....	3-14
3.1.2	Undirected Graph	3-1	3.3.1(G)	'C++' Function for Checking whether the given Graph is Connected	3-15
3.1.3	Directed Graph	3-1	3.3.1(H)	'C++' Function for Finding Components of a Graph	3-15
3.1.4	A Complete Graph.....	3-2	3.3.2	Breadth First Search (BFS)	3-15
3.1.5	Weighted Graph	3-2	3.3.2(A)	Algorithm for BFS.....	3-15
3.1.6	Adjacent Nodes	3-2	3.3.2(B)	Examples on BFS.....	3-16
3.1.7	Path.....	3-2	3.3.2(C)	Program for BFS using Adjacency Matrix	3-17
3.1.8	Cycle	3-2	3.3.2(D)	Program for BFS using Adjacency List.....	3-18
3.1.9	Connected Graph	3-2	3.3.2(E)	BFS Spanning Tree	3-20
3.1.10	Subgraph.....	3-3	3.4	Introduction of Greedy Strategy	3-21
3.1.11	Component.....	3-3	3.4.1	Knapsack Problem	3-21
3.1.12	Degree of a Vertex	3-3	3.4.2	Job Sequencing with Deadlines.....	3-22
3.1.13	Self Edges or Self Loops	3-3	3.4.3	Graph Colouring Problem	3-22
3.1.14	Multigraph	3-3	3.5	Connected Components.....	3-23
3.1.15	Tree.....	3-3	3.5.1	'C' Function for Printing Connected Components of a Graph	3-23
3.1.16	Spanning Trees	3-4	3.5.2	Transitive Closure and Connectedness	3-23
3.1.17	Eulerian Walk	3-4	3.5.3	Examples on Transitive Closure	3-24
3.1.18	Real World Applications of Graph.....	3-4	3.5.4	Warshall's Algorithm for Computing Transitive Closure of a Graph G	3-24
3.1.19	Minimal Spanning Tree.....	3-4			
3.2	Representation of Graphs.....	3-5			



<p>3.5.4(A) 'C' Function for Computing Transitive Closure of a Graph using Warshall's Algorithm3-24</p> <p>3.6 Minimum Cost Spanning Tree3-25</p> <p>3.6.1 Prim's Algorithm3-25</p> <p>3.6.1(A) 'C++' Function for Finding the Minimum Cost Spanning Tree.....3-26</p> <p>3.6.1(B) Program for Prim's Algorithm3-27</p> <p>3.6.1(C) Timing Complexity of Prim's Algorithm3-28</p> <p>3.6.1(D) Examples on Prim's Algorithm3-28</p> <p>3.6.2 Kruskal's Algorithm.....3-33</p> <p>3.6.2(A) 'C' Function find()3-35</p> <p>3.6.2(B) 'C' Function for Finding Minimum Cost Spanning Tree of a Graph using Kruskal's Algorithm3-35</p> <p>3.6.2(C) Program for Kruskal's Algorithm3-35</p> <p>3.6.2(D) Timing Complexity of Kruskal's Algorithm.....3-37</p> <p>3.6.2(E) Comparison of Time Complexity of Prim's and Kruskal's Algorithm3-37</p> <p>3.6.2(F) Examples on Kruskal's Algorithm3-37</p> <p>3.7 Shortest Path Algorithm3-42</p> <p>3.7.1 Dijkstra Algorithm3-42</p> <p>3.7.1(A) Timing Complexity3-42</p> <p>3.7.2 Program on Dijkstra's Algorithm3-42</p> <p>3.7.3 Examples on Dijkstra's Algorithm3-44</p> <p>3.8 Topological Sorting.....3-50</p> <p>3.8.1 Program for Topological Sorting3-50</p> <p>3.9 Floyd-warshall Algorithm3-54</p> <p>3.10 Data Structure in Webgraph and Google Map3-55</p> <p>3.11 University Questions and Answers3-55</p>	<p>4.2.1 Static Tree Tables4-2</p> <p>4.2.2 Dynamic Tree Tables4-2</p> <p>4.3 Introduction to Dynamic Programming4-2</p> <p>4.3.1 Triangulation Problem4-2</p> <p>4.3.2 0/1 Knapsack Problem4-3</p> <p>4.3.3 Other Examples of Dynamic Programming.....4-4</p> <p>4.4 Optimal Binary Search Tree (OBST)4-5</p> <p>4.4.1 OBST with Failure Nodes4-5</p> <p>4.4.2 Equation of the Root for OBST4-7</p> <p>4.4.3 C-Algorithm for Finding Minimum Cost Binary Search Tree.....4-8</p> <p>4.4.4 Program for Constructing an OBST4-9</p> <p>4.5 AVL Trees4-11</p> <p>4.5.1 Height Balanced Tree.....4-11</p> <p>4.5.2 Balance Factor4-11</p> <p>4.5.3 Structure of a Node in AVL Tree.....4-11</p> <p>4.5.4 'C' Function for Finding the Balance Factor of a Node.....4-12</p> <p>4.5.5 Insertion of a Node into an AVL Tree.....4-12</p> <p>4.5.5(A) Rotate Left.....4-13</p> <p>4.5.5(B) Rotate Right4-13</p> <p>4.5.5(C) Single Rotation and Double Rotation4-14</p> <p>4.5.5(D) 'C++' Function for Insertion of an Element into an AVL Tree4-17</p> <p>4.5.5(E) 'C++' Function to Find Height of AVL Tree4-18</p> <p>4.5.5(F) 'C++' Function to Rotate Right.....4-18</p> <p>4.5.5(G) 'C++' Function to Rotate Left.....4-18</p> <p>4.5.5(H) 'C++' Function for RR4-18</p> <p>4.5.5(I) 'C++' Function for LL4-18</p> <p>4.5.5(J) 'C++' Function for LR4-18</p> <p>4.5.5(K) 'C++' Function for RL4-18</p>
---	--

UNIT - IV

Chapter 4 : Search Trees 4-1 to 4-37

<p>4.1 Symbol Tables4-1</p> <p>4.2 Representation of Symbol Table4-1</p>
--



4.5.5(L)	'C++' Function for Deletion	4-19
4.5.6	Program for AVL Tree (AVL tree as an ADT).....	4-20
4.6	Weight Balanced Tree.....	4-32
4.7	Splay Tree.....	4-32
4.7.1	Bottom up Splaying	4-32
4.7.2	Top Down Splaying	4-33
4.8	Red-Black Tree	4-34
4.9	K-dimensional Tree	4-36
4.10	AA Tree.....	4-36
4.11	University Questions and Answers	4-37

Unit - V

Chapter 5 : Indexing & Multiway Trees 5-1 to 5-50

5.1	Indexing.....	5-1
5.1.1	Advantages of Indexing	5-1
5.2	Indexing Techniques	5-2
5.2.1	Cylinder Surface Indexing.....	5-2
5.2.2	Hashed Indexes	5-2
5.2.3	Tree Indexing	5-3
5.2.4	Trie Indexing.....	5-3
5.2.4(A)	Compact Trie.....	5-4
5.3	Multiway Search Tree.....	5-4
5.4	B-Trees	5-4
5.4.1	Insertion of a Key into a B-tree	5-5
5.4.2	Deleting a Value from a B-tree	5-12
5.4.3	B-tree as an ADT.....	5-16
5.5	B ⁺ Trees	5-20
5.6	Set ADT	5-23
5.6.1	UNION and FIND Operations for Disjoint Sets	5-23
5.6.2	The Disjoint Set ADT.....	5-24
5.6.2(A)	C-declaration of ADT	5-24

5.6.3	Smart Union Algorithms	5-25
5.6.3(A)	Union-by-size	5-25
5.6.3(B)	Union-by-height	5-26
5.6.4	Path Compression.....	5-27
5.7	Binary Heaps.....	5-28
5.7.1	Introduction	5-28
5.7.2	Types of Heaps	5-28
5.7.3	Basic Heap Operations	5-29
5.7.3(A)	Basic Operations for Max Heap.....	5-29
5.7.3(B)	'C++' Function for Upadjustment	5-30
5.7.3(C)	'C++' Function for Inserting an Element in a Heap	5-30
5.7.3(D)	'C++' Function for Downadjust.....	5-30
5.7.3(E)	'C++' Function for Delete Max	5-31
5.7.3(F)	'C++' Function for Creation of a Max Heap (in $N \log N$)	5-31
5.7.3(G)	Heap Creation-A Better Approach (In Linear Time) ..	5-32
5.7.3(H)	Heap as an ADT	5-39
5.7.4	Application of Heaps	5-41
5.7.4(A)	Sorting in Ascending Order (Algorithm)	5-41
5.7.4(B)	Sorting in Ascending Order (Program).....	5-42
5.7.4(C)	Timing Complexity of Heap Sort.....	5-44
5.7.4(D)	Priority Queues using a Heap Data Structure	5-48
5.8	University Questions and Answers	5-50

Unit - VI

Chapter 6 : File Organization 6-1 to 6-33

6.1	Introduction to Files (in 'C').....	6-1
6.1.1	Files I/O in 'C'	6-1
6.1.2	Comparison of Text File and Binary File.....	6-3
6.2	Basic File Operations (in 'C')	6-3
6.3	Sequential and Random Access Files (in 'C').....	6-5
6.3.1	Binary Files	6-7
6.4	Introduction to Files (In C++)	6-8



6.4.1	Creating a File.....	6-8	6.6.2(g)	Packing	6-15
6.4.2	Selecting File Open Mode	6-8	6.7	Indexing and Hashing.....	6-18
6.4.3	Reading from a File	6-8	6.7.1	Indexing.....	6-18
6.4.4	Writing to a File	6-8	6.7.1(A)	Advantages of Indexing over Sequential File.....	6-18
6.4.5	Reading and Writing of Objects.....	6-8	6.7.2	Hashing (Direct File Organization).....	6-18
6.4.6	Random Access Files.....	6-9	6.7.3	Operations on a Direct File.....	6-19
6.5	Organization of Records into Blocks.....	6-9	6.8	Types of Indices	6-21
6.6	File Organization	6-10	6.8.1	Primary Indices (Indexed Sequential File)	6-21
6.6.1	Primitive Operations on a File.....	6-10	6.8.1(A)	Operations on an Indexed Sequential File	6-22
6.6.1(A)	Creation.....	6-10	6.8.2	Clustering Indices.....	6-25
6.6.1(B)	Reading.....	6-10	6.8.3	Secondary Indices (Simple Index File)	6-25
6.6.1(C)	Insertion	6-10	6.8.3(A)	Operations on a Simple Index File.....	6-25
6.6.1(D)	Deletion.....	6-11	6.9	Indexing (or Sequential) and Hashing Comparison...	6-28
6.6.1(E)	Updation.....	6-11	6.10	Linked Organization of a File.....	6-29
6.6.2	Sequential Files.....	6-11	6.11	Inverted File Organisation	6-30
6.6.2(A)	Operations on a Sequential File	6-11	6.12	Coral Rings	6-30
6.6.2(a)	Creation.....	6-12	6.13	Cellular Partitions	6-30
6.6.2(b)	Reading (Printing).....	6-12	6.14	Consequential Processing.....	6-30
6.6.2(c)	Insertion	6-12	6.14.1	Merging of Two Sorted List.....	6-31
6.6.2(d)	Deletion.....	6-13	6.14.2	K-way Merge Algorithm	6-32
6.6.2(e)	Searching	6-14	6.15	University Questions and Answers	6-32
6.6.2(f)	Updation (Modifications).....	6-14		• Lab Manual.....	L-1 to L-21