Unit1

29 April 2025 20:51

1. What is a hash table?

Answer:
A hash table is a data structure that stores key-value pairs. It uses a hash function to compute an index (also called a hash code) into an array of buckets or slots, from which the desired value can be found.

2. What is a hash function?

Answer:
A hash function takes an input (or key) and returns an integer (hash code), which is used as an index in the hash table. A good hash function distributes keys uniformly and minimizes collisions.

3. What are the basic operations of a hash table?

Answer:
The basic operations include:

- **Insertion**: Add a key-value pair.
- **Search**: Find the value associated with a key.
- **Deletion**: Remove a key-value pair.

4. What is a collision in hashing?

Answer:
A collision occurs when two different keys produce the same hash index. This needs to be handled using collision resolution strategies.

5. Explain the terms 'bucket', 'probe', and 'synonym' in hashing.

Answer:

- **Bucket**: A container at a hash index where key-value pairs are stored.
- **Probe**: A process of checking alternate locations when the intended location is occupied.
- **Synonym**: Two keys that hash to the same index are called synonyms.

6. What is the difference between open hashing and closed hashing?

Answer:

- **Open Hashing (Separate Chaining)**: Uses linked lists to store multiple elements at the same index.
- **Closed Hashing (Open Addressing)**: Stores all elements within the hash table array, finding alternative locations when a collision occurs.

7. What is load factor? How is it calculated?

Answer:
Load factor is the ratio of the number of elements in the hash table to the number of buckets.
Load Factor=Number of elementsSize of hash table\text{Load Factor} = \frac{\text{Number of elements}}{\text{Size of hash table}}Load Factor=Size of hash tableNumber of elements

8. What are the properties of a good hash function?

Answer:

- Uniform distribution
- Determinism

- Efficient computation
- Minimization of collisions

9. Describe different hash functions.

Answer:

- **Division Method**: h(k) = k mod m
- **Multiplication Method**: Uses multiplication and fractional parts
- **Mid-Square Method**: Squares the key and takes middle digits
- **Extraction**: Extracts specific digits from the key
- **Folding**: Breaks the key into parts and combines them
- **Universal Hashing**: Randomly chooses from a class of hash functions

10. What is rehashing? When is it done?

Answer:

Rehashing is the process of creating a new hash table with a larger size and reinserting all elements. It is done when the load factor exceeds a certain threshold.

11. Describe open addressing techniques.

Answer:

- **Linear Probing**: Check next slots sequentially
- **Quadratic Probing**: Check at intervals increasing quadratically
- **Double Hashing**: Uses a second hash function to determine the step size

12. What is separate chaining?

Answer:

A method of collision resolution where each index contains a linked list of all elements hashed to that index.

13. What is extendible hashing?

Answer:

A dynamic hashing technique that allows the hash table to grow and shrink gracefully using a directory of pointers to buckets.

14. What is a perfect hash function?

Answer:

A hash function that maps each key to a unique index with no collisions.

15. What is a skip list?

Answer:

A skip list is a probabilistic data structure that allows fast search, insertion, and deletion operations, similar to a balanced tree, using multiple levels of linked lists.

16. How are insertion and removal performed in a skip list?

Answer:

- **Insertion**: Insert at the lowest level, then with probability (like 0.5), insert in upper levels.
- **Removal**: Remove the node from all levels it appears in.

Unit 2

29 April 2025 20:57

1. What is a tree in data structures?

Answer:
A tree is a hierarchical data structure consisting of nodes, with a single root node and potentially many levels of additional nodes. Each node has a value and links (edges) to child nodes.

2. Define the basic terminology used in trees.

Answer:

- **Root**: Topmost node.
- **Leaf**: Node with no children.
- **Parent**: Node with one or more child nodes.
- **Child**: Node that descends from a parent.
- **Subtree**: A tree formed by a node and its descendants.
- **Height**: Length of the longest path to a leaf.
- **Depth**: Distance from root to the node.

3. What is a general tree and how is it represented?

Answer:
A general tree is a tree where each node can have an arbitrary number of children. It can be represented:

- Using **sequential representation** (array-like)
- Using **linked representation** (each node has a pointer to the first child and next sibling)

4. What is a binary tree?

Answer:
A binary tree is a tree where each node has at most two children, commonly referred to as the left and right child.

5. How do you convert a general tree to a binary tree?

Answer:
By using the **left-child right-sibling** representation:

- The first child becomes the left child.
- Subsequent siblings become the right child of the previous one.

6. Explain different types of binary tree traversals.

Answer:

- **Inorder (LNR)**: Left, Node, Right
- **Preorder (NLR)**: Node, Left, Right
- **Postorder (LRN)**: Left, Right, Node
- **Level Order**: Breadth-first traversal

Both recursive and non-recursive (using stacks or queues) methods are used.

7. What is depth-first and breadth-first traversal in trees?

Answer:

- **Depth-First**: Explores as far as possible along each branch before backtracking (includes inorder, preorder, postorder).
- **Breadth-First**: Visits all nodes at the current depth before moving to the next level (uses a queue).

8. What are the operations on a binary tree?

Answer:

- **Insertion**
- **Deletion**
- **Traversal (inorder, preorder, postorder, level order)**
- **Searching**

9. What is a Huffman Tree?

Answer:

A Huffman Tree is a binary tree used for data compression. It builds an optimal prefix code based on the frequency of symbols. The most frequent symbols have the shortest codes.

10. What is a Binary Search Tree (BST)?

Answer:

A BST is a binary tree where each node's left subtree contains only nodes with keys less than the node's key, and the right subtree only keys greater than the node's key.

11. What are the basic operations on BSTs?

Answer:

- **Insertion**: Place the new key maintaining BST properties.
- **Deletion**: Remove a node and adjust the tree:
    - No child: delete directly
    - One child: replace node with child
    - Two children: replace with inorder successor or predecessor
- **Search**: Traverse down using the BST property.

12. What is a threaded binary tree?

Answer:

A binary tree where NULL pointers are replaced with references (threads) to in-order predecessor or successor, improving traversal without recursion or stack.

13. What is an in-order threaded binary search tree?

Answer:

In this tree, all the NULL pointers are replaced by threads pointing to the in-order predecessor and successor, enabling in-order traversal in linear time.

14. Explain the concept of threading in trees.

Answer:

Threading is used to make tree traversal faster and more memory efficient by using otherwise unused NULL pointers to point to adjacent nodes in traversal order.

15. How is in-order traversal performed on an in-order threaded binary tree?

Answer:

Starting from the leftmost node, follow the threads to visit the next in-order node until the end is reached.

Unit 3

29 April 2025 20:59

1. What is a graph in data structures?

Answer:
A graph is a collection of nodes (called vertices) connected by edges. It can be directed or undirected and may have weights on edges.

2. What are the different types of graphs?

Answer:

- **Undirected Graph**
- **Directed Graph (Digraph)**
- **Weighted Graph**
- **Unweighted Graph**
- **Cyclic / Acyclic Graph**
- **Connected / Disconnected Graph**

3. How are graphs represented in memory?

Answer:

- **Adjacency Matrix**: 2D matrix to represent edges; space-consuming but fast.
- **Adjacency List**: Each vertex has a list of its adjacent vertices; more space-efficient.
- **Adjacency Multi-list**: Used for directed graphs with multiple edges.
- **Inverse Adjacency List**: Stores nodes that point to a given node.

4. What are depth-first and breadth-first traversals in graphs?

Answer:

- **Depth-First Search (DFS)**: Explores as far as possible along each branch before backtracking (uses stack or recursion).
- **Breadth-First Search (BFS)**: Explores all neighbors at current depth before moving deeper (uses queue).

5. What is a Minimum Spanning Tree (MST)?

Answer:
An MST is a subset of the edges that connects all vertices with the minimum possible total edge weight and without cycles.

6. Which algorithms are used to find MSTs?

Answer:

- **Prim's Algorithm**: Starts with a single vertex and grows the tree.
- **Kruskal's Algorithm**: Sorts edges and adds the smallest ones without forming cycles.

7. What are greedy algorithms and how are they used in graph problems?

Answer:
Greedy algorithms make locally optimal choices at each step. Prim's and Kruskal's algorithms are greedy approaches to find MSTs.

8. What is Dijkstra's Algorithm?

Answer:
It finds the shortest path from a single source to all other vertices in a graph with non-negative edge

weights using a priority queue or min-heap.

9. What is the difference between Dijkstra's and Bellman-Ford algorithms?

Answer:

- **Dijkstra's**: Faster, but doesn't work with negative weights.
- **Bellman-Ford**: Handles negative weights but slower.

10. What is the Floyd-Warshall algorithm?

Answer:

A dynamic programming algorithm to find the shortest paths between all pairs of vertices. It works for both positive and negative weights (but no negative cycles).

11. What is topological sorting?

Answer:

It is a linear ordering of vertices in a directed acyclic graph (DAG) such that for every directed edge u → v, u comes before v in the ordering.

12. What are the applications of graphs in real-world scenarios?

Answer:

- Navigation systems (shortest path)
- Network routing
- Scheduling problems (topological sort)
- Social network modeling
- Resource allocation

13. What is an adjacency matrix and when should it be used?

Answer:

A 2D matrix where A[i][j] = 1 if there's an edge between vertices i and j. Use it when the graph is dense.

14. What is an adjacency list and when is it preferred?

Answer:

Each node has a list of its neighbors. It's space-efficient and preferred for sparse graphs.

15. Can BFS or DFS detect cycles in a graph?

Answer:

Yes, both can detect cycles:

- In **DFS**, if a node is visited and not yet fully explored (back edge), a cycle is present.
- In **BFS**, if a node is visited again through a non-parent node, a cycle is present.

Unit 4

29 April 2025 21:01

1. What is a symbol table?

Answer:
A symbol table is a data structure used by a compiler to store information about variables, functions, objects, etc., and their attributes (like scope, type, etc.).

2. What are the types of symbol table representations?

Answer:

- **Static Tree Table**: The structure of the tree doesn't change once created.
- **Dynamic Tree Table**: Allows insertions and deletions; adjusts the structure dynamically.

3. What is a weight-balanced tree?

Answer:
A binary search tree in which the balance of the tree is based on the number of nodes (weight) in the subtrees instead of height.

4. What is an Optimal Binary Search Tree (OBST)?

Answer:
An OBST is a binary search tree that provides the minimum expected search cost when different keys have different access probabilities.

5. How is OBST related to dynamic programming?

Answer:
OBST construction is solved using dynamic programming by building solutions to subproblems and using them to construct an optimal solution.

6. What is a height-balanced tree?

Answer:
A binary tree where the height of the left and right subtrees of every node differs by at most one.

7. What is an AVL tree?

Answer:
An AVL tree is a self-balancing binary search tree where the balance factor (height difference) of each node is between -1 and +1.

8. What are the rotation types in AVL trees?

Answer:

- **Left Rotation (LL)**
- **Right Rotation (RR)**
- **Left-Right Rotation (LR)**
- **Right-Left Rotation (RL)**

9. What is a Red-Black Tree?

Answer:
A Red-Black Tree is a self-balancing BST where each node has a color (red or black) and follows specific rules to ensure the tree remains balanced.

10. What are the properties of a Red-Black Tree?

Answer:

1. Each node is either red or black.
2. The root is always black.
3. Red nodes cannot have red children.
4. Every path from a node to its descendant null nodes must have the same number of black nodes.

11. What is an AA Tree?

Answer:

An AA tree is a balanced BST similar to a Red-Black Tree but simpler, using a single balancing operation (skew and split) to maintain balance.

12. What is a K-dimensional tree (k-d tree)?

Answer:

A k-d tree is a space-partitioning data structure for organizing points in a k-dimensional space, commonly used in range and nearest neighbor searches.

13. What is a Splay Tree?

Answer:

A self-adjusting binary search tree where recently accessed elements are moved to the root using tree rotations. This improves access time for frequently accessed elements.

14. What are the advantages of splay trees?

Answer:

- Frequently accessed nodes are faster to access.
- Simple to implement compared to AVL or Red-Black Trees.
- Amortized time complexity is O(log n).

15. Compare AVL Tree vs Red-Black Tree.

Answer:

Feature	**AVL Tree**	**Red-Black Tree**
Balancing Strict	More Strict	Less Strict
Rotations	More (frequent)	Fewer
Lookup	Faster	Slightly slower
Insertion/Deletion	Slower	Faster

Unit 5

29 April 2025 21:04

1. What is indexing in data structures?

Answer:
Indexing is a technique used to optimize the performance of database queries by minimizing the number of disk accesses required.

2. What are the different types of indexing techniques?

Answer:

- **Primary Indexing**: Based on the primary key of the table.
- **Secondary Indexing**: Based on non-primary key attributes.
- **Dense Indexing**: Index contains an entry for every record.
- **Sparse Indexing**: Index contains entries only for some records.

3. What is a multiway search tree?

Answer:
A multiway search tree is a tree data structure where each node can have more than two children, used to maintain sorted data and allow fast insertion, deletion, and search.

4. What is a B-Tree?

Answer:
A B-Tree is a self-balancing search tree in which nodes can have multiple children and is optimized for systems that read and write large blocks of data.

5. What are the properties of a B-Tree?

Answer:

- All leaves appear at the same level.
- A node with n keys has n+1 children.
- Keys in a node are sorted.
- Supports efficient insertion, deletion, and search in O(log n) time.

6. How does insertion work in a B-Tree?

Answer:
Start at the root, find the appropriate leaf node, insert the key, and if the node overflows (i.e., exceeds max keys), split it and propagate the middle key upwards.

7. How does deletion work in a B-Tree?

Answer:
Find the key, remove it, and if a node underflows (has fewer keys than minimum), borrow from a sibling or merge nodes and adjust the tree accordingly.

8. What is a B+ Tree?

Answer:
A B+ Tree is a variation of B-Tree where all values are stored at the leaf level, and internal nodes only store keys used for routing.

9. How is a B+ Tree different from a B-Tree?

Answer:

Feature	B-Tree	B+ Tree
Data Storage	Internal + Leaf nodes	Only in Leaf nodes
Leaf Structure	Unlinked	Linked sequentially
Range Queries	Slower	Faster

10. Why is B+ Tree preferred in database indexing?

Answer:
Because all data is stored at leaf nodes and leaves are linked, it allows faster range queries and efficient disk-based retrieval.

11. What are the use cases of B+ Trees in indexing?

Answer:

- Used in file systems and databases to index large datasets.
- Supports efficient range queries and sequential access.

12. What is a Trie Tree (Prefix Tree)?

Answer:
A Trie is a tree-like data structure used to store a dynamic set of strings where keys are usually strings (like in dictionary-based search).

13. How does insertion work in a Trie?

Answer:
Each character of the input string is inserted as a child node, forming a path from the root to a leaf where the full word is represented.

14. What are the advantages of Trie over Hash Tables?

Answer:

- Supports prefix search.
- No need for hash functions.
- Predictable and sorted retrieval of keys.

15. Where is Trie Tree used?

Answer:

- Autocomplete systems
- Spell checkers
- IP routing (Longest prefix match)
- Dictionary implementations

Unit 6

29 April 2025 21:05

1. What is a file in data structures?

Answer:
A file is a collection of data stored in a secondary storage device, organized in a specific format for efficient access and manipulation.

2. Why is file organization important?

Answer:
It determines how data is stored, retrieved, and updated efficiently, impacting performance, storage usage, and data retrieval time.

3. What are primitive file operations?

Answer:

- Create
- Open
- Read
- Write
- Close
- Delete

4. What is sequential file organization?

Answer:
In sequential file organization, records are stored and accessed in a linear sequence, usually sorted by a key field.

5. What are the advantages and disadvantages of sequential file organization?

Answer:

- **Advantages:** Simple to implement, efficient for batch processing.
- **Disadvantages:** Slow for random access, inefficient for frequent updates.

6. What is direct access file organization?

Answer:
Records are stored in a way that allows them to be accessed directly using a key or address, typically with the help of a hash function.

7. When is direct access file organization preferred?

Answer:
When fast and frequent retrieval of individual records is required, such as in databases and search operations.

8. What is indexed sequential file organization?

Answer:
It combines sequential and direct access by maintaining an index that maps keys to record locations, allowing both ordered and direct access.

9. What are the types of indices in indexed sequential files?

Answer:

- **Single-level index**
- **Multi-level index**
- **Dense index**
- **Sparse index**

10. What is the structure of an indexed sequential file?

Answer:
It includes a data file (sequentially organized) and one or more index files, where the index helps locate data records quickly.

11. What is linked file organization?

Answer:
Records are stored in different locations and connected through pointers, forming a logical sequence (like linked lists).

12. What are multi-list files in linked organization?

Answer:
Files where records are linked based on multiple keys, allowing different logical views of the same data.

13. What are coral rings in file organization?

Answer:
A file organization where records form circular lists or rings connected by multiple pointers, useful in networks or circular referencing.

14. What is an inverted file?

Answer:
A file structure that maintains a list of record pointers for each key, used in full-text search and indexing systems.

15. What are cellular partitions in file organization?

Answer:
A method of organizing files by dividing data into cells (or blocks), each responsible for a specific range of data or keys, improving access efficiency.