# Unit 1: Introduction to data structure and algorithms

1. **What is a data structure?**
    - **Answer:** A data structure is a way of organizing and storing data to perform operations efficiently.

2. **Define an algorithm.**
    - **Answer:** An algorithm is a step-by-step procedure or a set of rules for solving a problem.

3. **What is the purpose of a data structure?**
    - **Answer:** The purpose of a data structure is to organize and manage data efficiently, enabling effective manipulation and retrieval.

4. **Explain time complexity.**
    - **Answer:** Time complexity measures the amount of time an algorithm takes to complete as a function of the input size.

5. **What is an array?**
    - **Answer:** An array is a contiguous memory space that stores elements of the same type accessed using an index.

6. **Define a linked list.**
    - **Answer:** A linked list is a data structure where elements are stored in nodes, and each node points to the next in the sequence.

7. **Differentiate between a stack and a queue.**
    - **Answer:** A stack follows Last In, First Out (LIFO), while a queue follows First In, First Out (FIFO) ordering.

8. **What is recursion?**
    - **Answer:** Recursion is a programming concept where a function calls itself to solve a smaller instance of the same problem.

9. **Explain the concept of Big O notation.**

- **Answer:** Big O notation describes the upper bound of the time complexity of an algorithm in the worst-case scenario.

10. **Define a tree data structure.**

- **Answer:** A tree is a hierarchical data structure consisting of nodes connected by edges.

11. **What is a binary search?**

- **Answer:** Binary search is an efficient algorithm to find a target element in a sorted array by repeatedly dividing the search space in half.

12. **Define hashing.**

- **Answer:** Hashing is a technique to map data to a fixed-size array, typically for efficient data retrieval.

13. **Differentiate between breadth-first search and depth-first search.**

- **Answer:** Breadth-first search explores all neighbor nodes before moving to the next level, while depth-first search explores as far as possible before backtracking.

14. **What is dynamic programming?**

- **Answer:** Dynamic programming is a method for solving problems by breaking them down into smaller overlapping sub-problems and solving each subproblem only once.

15. **Define an AVL tree.**

- **Answer:** An AVL tree is a self-balancing binary search tree, where the heights of the two child sub-trees of every node differ by at most one.

16. **Explain the concept of amortized analysis.**

- **Answer:** Amortized analysis provides the average time per operation, considering both expensive and cheap operations over a sequence of operations.

17. **What is the purpose of a hash table?**

- **Answer:** A hash table is used to implement an associative array and maps keys to values using a hash function.

18. **Define Dijkstra's algorithm.**

- **Answer:** Dijkstra's algorithm is a shortest-path algorithm that finds the shortest path between two nodes in a weighted graph.

19. **Explain the concept of greedy algorithms.**

- **Answer:** Greedy algorithms make locally optimal choices at each stage with the hope of finding a global optimum.

20. **Define a priority queue.**

- **Answer:** A priority queue is a data structure that stores elements with associated priorities and supports efficient retrieval of the element with the highest priority.

# Unit 2: Linked Lists

1. **What is a linked list?**
    - **Answer:** A linked list is a linear data structure where elements are stored in nodes, and each node points to the next node in the sequence.

2. **Differentiate between an array and a linked list.**
    - **Answer:** An array is a fixed-size data structure with contiguous memory, while a linked list is dynamic in size and uses non-contiguous memory.

3. **What is a node in a linked list?**
    - **Answer:** A node is a basic building block of a linked list, containing data and a reference (or link) to the next node in the sequence.

4. **Explain singly linked list and doubly linked list.**
    - **Answer:** In a singly linked list, each node points to the next node. In a doubly linked list, each node has pointers to both the next and the previous nodes.

5. **Define the term "head" in a linked list.**
    - **Answer:** The "head" is the first node in a linked list, serving as the starting point for traversal.

6. **What is the "tail" of a linked list?**
    - **Answer:** The "tail" is the last node in a linked list, where the next reference is typically null.

7. **How is an empty linked list represented?**
    - **Answer:** An empty linked list is represented by a null reference (or a null head).

8. **What is the time complexity for inserting an element at the beginning of a linked list?**
    - **Answer:** O(1) or constant time.

9. **Explain the term "null pointer" in the context of linked lists.**
    - **Answer:** A "null pointer" is a pointer that does not point to any valid node or memory location.

10. **What is the time complexity for searching an element in a linked list?**
    - **Answer:** O(n), where n is the number of nodes in the linked list.

11. **Define the term "circular linked list."**
    - **Answer:** In a circular linked list, the last node points back to the first node, forming a loop.

12. **What is the purpose of a dummy node in a linked list?**
    - **Answer:** A dummy node can be used as a placeholder to simplify edge cases or avoid special handling for an empty list.

13. **Explain the concept of a "sentinel node."**
    - **Answer:** A sentinel node is a dummy node placed at the beginning or end of a list to simplify algorithms and reduce special cases.

14. **What is the difference between "iteration" and "recursion" in the context of linked lists?**
    - **Answer:** Iteration involves using loops to traverse a linked list, while recursion involves using a function that calls itself.

15. **Define "in-place reversal" of a linked list.**
    - **Answer:** In-place reversal means reversing the order of nodes within the existing linked list without using additional data structures.

16. **Explain the term "cycle" in the context of linked lists.**

- **Answer:** A cycle occurs when a node in the linked list points to a previous node, creating a loop.

17. **What is the difference between a singly linked list and a doubly linked list in terms of memory usage?**

- **Answer:** Doubly linked lists use more memory due to the additional pointers for backward traversal.

18. **Define the term "intersection point" in the context of linked lists.**

- **Answer:** The intersection point is the node where two linked lists merge.

19. **How is a self-loop different from a cycle in a linked list?**

- **Answer:** A self-loop occurs when a node points to itself, creating a loop of length one.

20. **What is the purpose of a hash table in the context of linked lists?**

- **Answer:** A hash table can be used to efficiently store and retrieve elements in a linked list by mapping keys to positions in the list.

# Unit 3: Stack

1. **What is a stack?**
    - **Answer:** A stack is a linear data structure that follows the Last In, First Out (LIFO) principle.

2. **Name the two primary operations on a stack.**
    - **Answer:** Push (inserting an element onto the stack) and Pop (removing the top element from the stack).

3. **What is the role of the top pointer in a stack?**
    - **Answer:** The top pointer points to the top element of the stack, indicating the last element that was added.

4. **Explain the dynamic implementation of a stack.**
    - **Answer:** Dynamic implementation uses a linked list to allow the stack size to grow or shrink dynamically

5. **What is the minimum number of stacks needed to implement a queue?**
    - **Answer:** Two stacks are required to implement a queue efficiently.

6. **Define the term "stack overflow."**
    - **Answer:** A stack overflow occurs when the stack is full, and a push operation is attempted.

7. **What is the significance of parentheses matching using a stack?**
    - **Answer:** Stacks are often used to check if parentheses in an expression are balanced.

8. **Explain the use of a stack in function calls.**
    - **Answer:** The call stack is used to manage function calls, storing local variables and return addresses.

9. **What is the time complexity of push and pop operations in a stack?**

- **Answer:** O(1) time complexity for both push and pop operations.

10. **Can a linked list be used to implement a stack?**

- **Answer:** Yes, a linked list is a common way to implement a stack dynamically.

11. **What is the real-world analogy of a stack?**

- **Answer:** A real-world analogy of a stack is a stack of plates in a cafeteria.

12. **Define the term "stack underflow."**

- **Answer:** A stack underflow occurs when a pop operation is attempted on an empty stack.

13. **How is the stack used in the depth-first search algorithm?**

- **Answer:** The call stack is used to keep track of vertices during recursive traversal in depth-first search.

14. **Explain the role of a temporary stack in sorting a stack.**

- **Answer:** A temporary stack is used to store intermediate results during stack sorting algorithms.

15. **What is the difference between a stack and a queue?**

- **Answer:** A stack follows LIFO, while a queue follows FIFO (First In, First Out) order.

16. **Why is a stack often used in the undo mechanism of applications?**

- **Answer:** The stack helps keep track of changes, allowing easy reversal in the undo mechanism.

17. **What is the purpose of a sentinel node in a stack implementation?**

- **Answer:** A sentinel node is a dummy node used to simplify stack implementation and handle edge cases.

18. **Explain the concept of a call stack in programming.**

- **Answer:** The call stack is a data structure that stores information about active functions and their local variables during program execution.

19. **Can a stack be implemented using arrays?**

- **Answer:** Yes, a stack can be implemented using arrays with a fixed or dynamic size.

20. **How does a stack help in evaluating expressions?**

- **Answer:** Stacks are used to convert infix expressions to postfix form, facilitating efficient evaluation.

# Unit 4: Queues

1. **What is a queue?**
    - **Answer:** A queue is a linear data structure that follows the First In, First Out (FIFO) principle.

2. **How are elements added to a queue?**
    - **Answer:** Elements are added to a queue at the rear (or back).

3. **How are elements removed from a queue?**
    - **Answer:** Elements are removed from a queue at the front.

4. **What is the main difference between a stack and a queue?**
    - **Answer:** A stack follows Last In, First Out (LIFO), while a queue follows First In, First Out (FIFO).

5. **Explain the terms "enqueue" and "dequeue."**
    - **Answer:** "Enqueue" is the operation to add an element to the rear of the queue, and "dequeue" is the operation to remove an element from the front.

6. **What is the front of a queue?**
    - **Answer:** The front of a queue is the position where elements are removed.

7. **What is the rear of a queue?**
    - **Answer:** The rear of a queue is the position where elements are added.

8. **Why is a queue often implemented using linked lists?**
    - **Answer:** Linked lists allow dynamic memory allocation and efficient insertion and deletion at both ends, making them suitable for queues.

9. **What is the time complexity of enqueue and dequeue operations in a basic queue?**
    - **Answer:** Both enqueue and dequeue operations in a basic queue have a time complexity of O(1).

10. **What is a circular queue, and why is it used?**
    - **Answer:** A circular queue is a queue where the rear and front are connected. It avoids wasting space and allows the queue to wrap around.

11. **Explain the term "empty queue."**
    - **Answer:** An empty queue is a queue with no elements in it.

12. **How is a priority queue different from a regular queue?**
    - **Answer:** A priority queue assigns a priority to each element and removes the element with the highest priority first.

13. **What is the significance of the "peek" operation in a queue?**
    - **Answer:** The "peek" operation returns the front element of the queue without removing it.

14. **Define "circular buffer" in the context of queues.**
    - **Answer:** A circular buffer is a data structure that uses a fixed-size array as if it were connected end-to-end.

15. **Why is a double-ended queue (deque) useful?**
    - **Answer:** A deque allows insertion and deletion at both ends, providing more flexibility than a basic queue.

16. **What is the purpose of a blocking queue?**
    - **Answer:** A blocking queue is designed to block or wait when attempting to dequeue from an empty queue or enqueue to a full queue.

17. **How can a queue be implemented using stacks?**

- **Answer:** A queue can be implemented using two stacks, simulating the enqueue and dequeue operations.

18. **Explain the term "priority inversion" in the context of queues.**

- **Answer:** Priority inversion occurs when a higher-priority task is waiting for a lower-priority task to release a resource.

19. **What is the difference between a regular queue and a deque?**

- **Answer:** A deque allows insertion and deletion at both ends, while a regular queue only allows operations at one end.

20. **When would you choose to use a queue over other data structures?**

- **Answer:** Queues are particularly useful when the order of processing elements is crucial, and the FIFO principle needs to be maintained.

# Unit 5: Recursion

1. **What is recursion?**
    - **Answer:** Recursion is a programming concept where a function calls itself to solve a smaller instance of the same problem.

2. **What is the base case in recursion?**
    - **Answer:** The base case is the condition that stops the recursive calls and provides a result directly.

3. **Explain the difference between recursion and iteration.**
    - **Answer:** Recursion involves a function calling itself, while iteration involves loops and repeated execution.

4. **What is the significance of a base case in recursive functions?**
    - **Answer:** The base case prevents infinite recursion by defining a condition where the function returns a result without making further recursive calls.

5. **Define tail recursion.**
    - **Answer:** Tail recursion occurs when the recursive call is the last operation in a function.

6. **What is the call stack in recursion?**
    - **Answer:** The call stack is a data structure that stores information about the active functions in a program, including local variables and return addresses.

7. **Explain the concept of indirect recursion.**
    - **Answer:** Indirect recursion occurs when two or more functions call each other in a circular manner.

8. **What is the role of a recursive helper function?**

- **Answer:** A recursive helper function is used to encapsulate the recursive logic and may have additional parameters to assist in the computation.

9. **Define tree recursion.**

- **Answer:** Tree recursion happens when a function makes multiple recursive calls, creating a branching structure similar to a tree.

10. **How can you optimize recursive algorithms for better performance?**

- **Answer:** Memorization (caching previous results) and dynamic programming are common techniques for optimizing recursive algorithms.

11. **What is the significance of the call stack in recursion?**

- **Answer:** The call stack keeps track of function calls and allows the program to return to the correct point after a recursive call.

12. **What is the Fibonacci sequence, and how can you compute it using recursion?**

- **Answer:** The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding ones. Recursion can be used to compute it by expressing the nth Fibonacci number in terms of the (n-1) and (n-2) Fibonacci numbers.

13. **Explain the concept of "head" recursion.**

- **Answer:** Head recursion occurs when the recursive call is the first operation in a function.

14. **How does recursion work for problems involving backtracking?**

- **Answer:** Recursion is often used in backtracking to explore all possible solutions by trying different choices and undoing them if they lead to a dead end.

15. **What is the difference between direct and indirect recursion?**

- **Answer:** Direct recursion involves a function calling itself, while indirect recursion involves multiple functions calling each other in a circular manner.

16. **How can you prevent stack overflow in recursive programs?**

- **Answer:** Optimizing tail recursion, using proper base cases, and increasing the stack size are ways to prevent stack overflow.

17. **Define the concept of "mutual recursion."**

- **Answer:** Mutual recursion occurs when two or more functions call each other in a cyclic manner.

18. **How can you convert a recursive solution to an iterative one?**

- **Answer:** You can use a loop and a stack or queue to mimic the call stack and convert a recursive solution to an iterative one.

19. **What is the role of a termination condition in recursion?**

- **Answer:** The termination condition, also known as the base case, defines when the recursion should stop and provides the result directly without further recursive calls.

20. **How does recursion relate to the "divide and conquer" paradigm?**

- **Answer:** Recursion often implements the "divide and conquer" paradigm by breaking a problem into smaller sub-problems, solving them recursively, and combining the results to obtain the final solution.

# Unit 6: Trees

1. **What is a tree in data structures?**
    - **Answer:** A tree is a hierarchical data structure composed of nodes, each having a value and zero or more child nodes.

2. **Define the root of a tree.**
    - **Answer:** The root is the topmost node in a tree, from which all other nodes are descended.

3. **What is a leaf node in a tree?**
    - **Answer:** A leaf node is a node in a tree that has no children; it is a terminal node.

4. **Differentiate between a binary tree and a binary search tree.**
    - **Answer:** A binary tree is a tree in which each node has at most two children, while a binary search tree maintains the property that the left child is less than the parent, and the right child is greater.

5. **Explain the term "height" in a tree.**
    - **Answer:** The height of a tree is the length of the longest path from the root to a leaf. It is the number of edges on the longest downward path.

6. **Define a full binary tree.**
    - **Answer:** A full binary tree is a binary tree in which every node has either 0 or 2 children.

7. **What is an ancestor in a tree?**
    - **Answer:** An ancestor of a node in a tree is any node on the path from the root to that node.

8. **Define a sub-tree.**
    - **Answer:** A sub-tree is a tree formed by a node and all its descendants in the original tree.

9. **What is a binary heap?**

- **Answer:** A binary heap is a complete binary tree that satisfies the heap property, where the value of each node is less than or equal to the values of its children.

10. **Explain the terms "preorder," "inorder," and "postorder" traversal.**

- **Answer:** Preorder: Visit the root node before its children. Inorder: Visit the left child, then the root, and finally the right child. Postorder: Visit the children before the root.

11. **Define a balanced tree.**

- **Answer:** A balanced tree is a tree in which the heights of the left and right sub-trees of every node differ by at most one.

12. **What is the purpose of AVL trees?**

- **Answer:** AVL trees are self-balancing binary search trees, maintaining balance to ensure efficient operations.

13. **Differentiate between a complete binary tree and a perfect binary tree.**

- **Answer:** A complete binary tree is a tree in which every level, except possibly the last, is completely filled, while a perfect binary tree is a complete binary tree with all levels completely filled.

14. **Define a binary tree traversal algorithm without using recursion.**

- **Answer:** Use a stack to simulate the recursive call stack during traversal.

15. **What is the lowest common ancestor (LCA) in a binary tree?**

- **Answer:** The lowest common ancestor of two nodes in a binary tree is the lowest (deepest) node that has both nodes as descendants.

16. **Explain the term "sub-tree sum."**

- **Answer:** Sub-tree sum is the sum of all values in a sub-tree rooted at a specific node.

17. **Define a threaded binary tree.**

- **Answer:** A threaded binary tree is a binary tree in which all the null pointers are replaced with pointers to successor or predecessor nodes in an inorder traversal.

18. **What is a trie data structure used for?**

- **Answer:** A trie is used for efficient retrieval of keys in a dynamic set, typically strings.

19. **Define the terms "height-balanced" and "weight-balanced" trees.**

- **Answer:** A height-balanced tree is a tree where the heights of the two child sub-trees of every node differ by at most one. A weight-balanced tree is a tree where, for every node, the size of its left and right sub-trees differ by at most one.

20. **What is the importance of tree traversal algorithms?**

- **Answer:** Tree traversal algorithms are important for visiting and processing all nodes in a tree efficiently and are fundamental to many tree-related operations and applications.

# Unit 7: Graphs

1. **What is a graph?**
    - **Answer:** A graph is a collection of nodes (vertices) and edges that connect pairs of nodes.

2. **Differentiate between a directed and an undirected graph.**
    - **Answer:** In a directed graph, edges have a direction, while in an undirected graph, edges have no direction.

3. **Define a path in a graph.**
    - **Answer:** A path is a sequence of edges that connect a sequence of vertices in a graph.

4. **What is a cycle in a graph?**
    - **Answer:** A cycle is a path that starts and ends at the same vertex, passing through other vertices only once.

5. **Explain the term "connected graph."**
    - **Answer:** A connected graph is a graph in which there is a path between every pair of vertices.

6. **What is a tree in the context of graphs?**
    - **Answer:** A tree is an acyclic, connected, and undirected graph.

7. **Define the degree of a vertex.**
    - **Answer:** The degree of a vertex is the number of edges incident to it.

8. **What is a weighted graph?**
    - **Answer:** A weighted graph assigns weights or costs to the edges.

9. **Differentiate between a sparse and a dense graph.**

- **Answer:** A sparse graph has fewer edges relative to the number of vertices, while a dense graph has more edges.

10. **Define an adjacency matrix.**

- **Answer:** An adjacency matrix is a 2D array where each entry `matrix[i][j]` represents an edge between vertices `i` and `j`.

11. **Explain the concept of an adjacency list.**

- **Answer:** An adjacency list is a collection of lists, where each list represents the neighbors of a vertex in a graph.

12. **What is a bipartite graph?**

- **Answer:** A bipartite graph is a graph whose vertices can be divided into two disjoint sets such that no two vertices within the same set are adjacent.

13. **Define the term "topological sort."**

- **Answer:** Topological sort is an ordering of the vertices of a directed acyclic graph (DAG) such that for every directed edge `uv`, vertex `u` comes before `v` in the ordering.

14. **What is a minimum spanning tree (MST)?**

- **Answer:** A minimum spanning tree is a tree that spans all the vertices in a connected, undirected graph while minimizing the sum of edge weights.

15. **Explain the concept of graph traversal.**

- **Answer:** Graph traversal is the process of visiting all the vertices and edges in a graph systematically.

16. **Define the term "graph cycle detection."**

- **Answer:** Graph cycle detection is the process of determining whether a graph contains cycles.

17. **What is Dijkstra's algorithm used for?**

- **Answer:** Dijkstra's algorithm finds the shortest paths from a source vertex to all other vertices in a weighted graph.

18. **Define the term "graph coloring."**

- **Answer:** Graph coloring is the assignment of labels (colors) to the vertices of a graph such that no two adjacent vertices have the same color.

19. **What is the purpose of the Floyd-Warshall algorithm?**

- **Answer:** The Floyd-Warshall algorithm is used to find the shortest paths between all pairs of vertices in a weighted graph.

20. **Define the term "Eulerian graph."**

- **Answer:** An Eulerian graph is a graph that contains an Eulerian circuit, a closed path that includes every edge exactly once.

# Unit 8: Sorting

1. **What is sorting?**
    - **Answer:** Sorting is the process of arranging elements in a specific order, often in ascending or descending order.

2. **Name a few sorting algorithms.**
    - **Answer:** Examples include Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort.

3. **Define time complexity in the context of sorting algorithms.**
    - **Answer:** Time complexity measures the amount of time an algorithm takes to complete as a function of the input size.

4. **Explain stable sorting.**
    - **Answer:** Stable sorting preserves the relative order of equal elements in the sorted output as they were in the original input.

5. **What is the worst-case time complexity of Bubble Sort?**
    - **Answer:** O(n^2), where n is the number of elements.

6. **Define in-place sorting.**
    - **Answer:** In-place sorting algorithms use a constant amount of extra memory to rearrange elements.

7. **What is the key operation in Quick Sort?**
    - **Answer:** Partitioning is the key operation in Quick Sort.

8. **Define comparison-based sorting.**
    - **Answer:** Comparison-based sorting algorithms determine the order of elements by comparing them using comparison operators (e.g., <, >, ==).

9. **Name a linear-time sorting algorithm.**

- **Answer:** Counting Sort is an example of a linear-time sorting algorithm.

10. **What is the time complexity of Merge Sort?**

- **Answer:** O(n log n), where n is the number of elements.

11. **Explain the concept of stability in sorting.**

- **Answer:** Stability in sorting means equal elements maintain their relative order in the sorted output.

12. **Define external sorting.**

- **Answer:** External sorting involves the sorting of data that is too large to fit into memory.

13. **What is the main advantage of Heap Sort?**

- **Answer:** Heap Sort has a consistent O(n log n) time complexity for both worst and average cases.

14. **Explain the term "inversion" in the context of sorting.**

- **Answer:** An inversion is a pair of indices (i, j) in an array where i < j but A[i] > A[j].

15. **Define Radix Sort.**

- **Answer:** Radix Sort is a non-comparative sorting algorithm that works by distributing elements into buckets based on their digits.

16. **What is the main disadvantage of Bubble Sort?**

- **Answer:** Bubble Sort has poor performance on large datasets and is generally not suitable for practical use.

17. **What is the primary benefit of Insertion Sort?**

- **Answer:** Insertion Sort is efficient for small datasets and partially sorted data.

18. **Define a stable sorting algorithm.**

- **Answer:** A sorting algorithm is stable if the relative order of equal elements remains unchanged in the sorted output.

19. **What is the time complexity of Selection Sort?**

- **Answer:** O(n^2), where n is the number of elements.

20. **Explain the term "in-place" in the context of sorting algorithms.**

- **Answer:** An in-place sorting algorithm rearranges elements using only a constant amount of extra memory, making it memory-efficient.

# Unit 9: Searching and Hashing

1. **What is searching in data structures?**
   - **Answer:** Searching involves finding a specific element in a collection of data.

2. **What is linear search?**
   - **Answer:** Linear search is a sequential search algorithm that checks each element in order until a match is found.

3. **Explain binary search.**
   - **Answer:** Binary search is an efficient search algorithm that divides the search space in half at each step.

4. **What is the time complexity of binary search?**
   - **Answer:** The time complexity of binary search is O(log n).

5. **Define hashing.**
   - **Answer:** Hashing is a technique that maps data to a fixed-size array, typically for efficient data retrieval.

6. **What is a hash function?**
   - **Answer:** A hash function is a function that takes an input (or "key") and produces a fixed-size string of characters.

7. **Explain collision in hashing.**
   - **Answer:** Collision occurs in hashing when two different keys hash to the same index.

8. **How is collision resolution handled in hashing?**
   - **Answer:** Collision resolution methods include chaining, open addressing, and double hashing.

9. **What is open addressing in hashing?**

- **Answer:** Open addressing is a collision resolution technique where the system searches for another empty location in the hash table.

10. **Define linear probing.**

- **Answer:** Linear probing is a type of open addressing where the system searches for the next available slot linearly.

11. **What is chaining in hashing?**

- **Answer:** Chaining is a collision resolution technique where each hash table slot points to a linked list of elements that hash to the same index.

12. **Explain the concept of load factor in hashing.**

- **Answer:** The load factor is the ratio of the number of elements stored in the hash table to the size of the hash table.

13. **What is rehashing?**

- **Answer:** Rehashing involves creating a new hash table and transferring elements from the old table to the new one when the load factor is too high.

14. **Define probing in hashing.**

- **Answer:** Probing is the process of searching for the next available slot when a collision occurs in open addressing.

15. **Explain the concept of perfect hashing.**

- **Answer:** Perfect hashing is a technique where no collisions occur, ensuring each key maps to a unique index.

16. **What is a hash collision attack?**

- **Answer:** A hash collision attack is a deliberate attempt to find two different inputs that hash to the same output.

17. **Define cuckoo hashing.**

- **Answer:** Cuckoo hashing is a hashing algorithm that uses two hash functions and multiple hash tables to resolve collisions.

18. **What is the purpose of a bloom filter?**

- **Answer:** A bloom filter is a data structure designed to test whether a given element is a member of a set.

19. **Explain the concept of double hashing.**

- **Answer:** Double hashing is a collision resolution technique where a secondary hash function is used to find an alternative index.

20. **What is the time complexity of searching in a hash table?**

- **Answer:** The time complexity of searching in a hash table is typically O(1) on average, assuming a good hash function and minimal collisions.