100 Most Asked Data Structure QnA

Made By:

Yadneyesh (Curious Coder)
CodWithCurious.com

1. **What is a data structure?**
   a. A data structure is a way of organizing and storing data in a computer so that it can be accessed and manipulated efficiently.
2. **What is the difference between an array and a linked list?**
   a. An array stores elements of the same type in contiguous memory locations, allowing random access to elements. A linked list stores elements in separate objects called nodes, which are connected through pointers, providing dynamic memory allocation and efficient insertion/deletion at any position.
3. **What is the time complexity of accessing an element in an array and a linked list?**
   a. Accessing an element in an array takes constant time O(1) since the position is known. In a linked list, accessing an element takes linear time O(n) as you need to traverse the list.
4. **What is the difference between a stack and a queue?**
   a. A stack follows the Last-In-First-Out (LIFO) principle, where the last element inserted is the first one to be removed. A queue follows the First-In-First-Out (FIFO) principle, where the first element inserted is the first one to be removed.
5. **What is the difference between a stack and a heap?**
   a. A stack is used for local variables and function calls, and its memory allocation and deallocation are handled automatically. A heap is used for dynamically allocated memory and needs explicit memory management.
6. **What is a binary tree?**
   a. A binary tree is a data structure where each node can have at most two children. The left child is generally smaller, and the right child is greater than the parent.
7. **What is the difference between a binary tree and a binary search tree (BST)?**
   a. A binary tree can have any values in its nodes, whereas a BST follows the property that for any node, all elements in its left subtree are smaller, and all elements in its right subtree are larger.
8. **What is the time complexity of searching for an element in a binary search tree (BST)?**
   a. The time complexity of searching for an element in a BST is O(log n) in the average case and O(n) in the worst case if the tree is skewed.

9. **What is a hash table?**
    a. A hash table is a data structure that stores key-value pairs using a hash function. It provides constant-time average-case complexity for insertion, deletion, and search operations.
10. **What is collision resolution in a hash table?**
    a. Collision resolution is the process of handling situations where two different keys map to the same hash value. Common techniques include chaining (using linked lists) and open addressing (probing neighboring locations).
11. **What is the difference between a stack and a heap memory allocation?**
    a. Stack memory allocation is used for static memory allocation and follows a Last-In-First-Out (LIFO) structure, while heap memory allocation is used for dynamic memory allocation and allows flexible memory management.
12. **What is a doubly linked list?**
    a. A doubly linked list is a type of linked list where each node contains a reference to both the next and previous nodes, allowing traversal in both directions.
13. **What is a circular linked list?**
    a. A circular linked list is a type of linked list where the last node points back to the first node, creating a circular structure. It can be singly or doubly linked.
14. **What is a priority queue?**
    a. A priority queue is an abstract data type that allows elements to be inserted with a priority and removes the highest-priority element first.
15. **What is the difference between a shallow copy and a deep copy?**
    a. A shallow copy creates a new object with references to the same memory locations as the original object. A deep copy creates a new object with its own copy of the data, recursively copying all the referenced objects.
16. **What is the time complexity of various operations in a priority queue implemented as a binary heap?**
    a. The time complexity of insertion and deletion (extracting the minimum or maximum) in a binary heap-based priority queue is O(log n), while searching for an element takes O(n).
17. **What is the difference between BFS (Breadth-First Search) and DFS (Depth-First Search)?**
    a. BFS explores all the vertices of a graph or tree level by level, while DFS explores a branch as far as possible before backtracking.
18. **What is a trie?**
    a. A trie, also known as a prefix tree, is a tree-like data structure used for efficient retrieval of keys that share prefixes. It is often used for implementing dictionary-like structures.
19. **What is the time complexity of searching in a trie?**
    a. The time complexity of searching in a trie is O(m), where m is the length of the search string. It does not depend on the number of keys stored in the trie.
20. **What is an AVL tree?**

a. An AVL tree is a self-balancing binary search tree where the heights of the left and right subtrees of any node differ by at most one. It ensures that the tree remains balanced, providing efficient search, insertion, and deletion operations.

21. **What is the time complexity of searching in an AVL tree?**
    a. The time complexity of searching in an AVL tree is O(log n), as the tree is balanced.
22. **What is a B-tree?**
    a. A B-tree is a self-balancing search tree that maintains sorted data and allows efficient search, insertions, and deletions. It is commonly used in databases and file systems.
23. **What is the difference between a B-tree and a binary search tree?**
    a. A B-tree can have multiple keys per node and multiple children, while a binary search tree can have at most two children per node. B-trees are designed for efficient disk access and can store a larger number of keys per node.
24. **What is the time complexity of searching in a B-tree?**
    a. The time complexity of searching in a B-tree is O(log n), as the tree is balanced and ensures efficient search operations.
25. **What is the difference between a graph and a tree?**
    a. A tree is a connected acyclic graph with a unique root, while a graph can have cycles and may not have a root.
26. **What is a spanning tree?**
    a. A spanning tree of a graph is a subgraph that includes all the vertices of the graph with the minimum possible number of edges. It does not contain cycles.
27. **What are the different types of graph traversals?**
    a. The different types of graph traversals are Breadth-First Search (BFS) and Depth-First Search (DFS).
28. **What is Dijkstra's algorithm?**
    a. Dijkstra's algorithm is a popular algorithm for finding the shortest path between two nodes in a graph with non-negative edge weights.
29. **What is Kruskal's algorithm?**
    a. Kruskal's algorithm is a greedy algorithm used to find a minimum spanning tree in a weighted undirected graph.
30. **What is the time complexity of Dijkstra's algorithm?**
    a. The time complexity of Dijkstra's algorithm is O((V + E) log V), where V is the number of vertices and E is the number of edges in the graph.
31. **What is the time complexity of Kruskal's algorithm?**
    a. The time complexity of Kruskal's algorithm is O(E log E), where E is the number of edges in the graph.
32. **What is memoization?**
    a. Memoization is a technique used to optimize recursive algorithms by caching the results of function calls and reusing them when the same inputs occur again, avoiding redundant computations.

33. **What is the difference between a linear search and a binary search?**
    a. Linear search sequentially checks each element in a list until a match is found, while binary search divides a sorted list into halves and compares the middle element to the target value, reducing the search space in each iteration.
34. **What is the time complexity of a linear search?**
    a. The time complexity of a linear search is O(n), where n is the number of elements in the list.
35. **What is the time complexity of a binary search?**
    a. The time complexity of a binary search is O(log n), where n is the number of elements in the sorted list.
36. **What is a self-balancing binary search tree?**
    a. A self-balancing binary search tree is a binary search tree that automatically maintains its balance during insertions and deletions, ensuring efficient search operations.
37. **What are some examples of self-balancing binary search trees?**
    a. Examples of self-balancing binary search trees include AVL tree, Red-Black tree, and Splay tree.
38. **What is the time complexity of insertions and deletions in a self-balancing binary search tree?**
    a. The time complexity of insertions and deletions in a self-balancing binary search tree is O(log n), as the tree ensures balance after each operation.
39. **What is the difference between a hash set and a hash map?**
    a. A hash set is a collection of unique elements, while a hash map is a collection of key-value pairs, where each key is unique.
40. **What is the difference between a graph and a tree?**
    a. A tree is a special type of graph with no cycles, while a graph can have cycles.
41. **What is a self-loop in a graph?**
    a. A self-loop is an edge in a graph that connects a vertex to itself.
42. **What is a directed graph?**
    a. A directed graph, also known as a digraph, is a graph where edges have a direction. The edges indicate a one-way relationship between vertices.
43. **What is the difference between a breadth-first search (BFS) and a depth-first search (DFS) in a graph?**
    a. BFS explores all the vertices at the same level before moving to the next level, while DFS explores as far as possible along each branch before backtracking.
44. **What is a cyclic graph?**
    a. A cyclic graph is a graph that contains at least one cycle, i.e., a path that starts and ends at the same vertex.
45. **What is a topological sort?**
    a. A topological sort is an ordering of the vertices of a directed acyclic graph (DAG) such that for every directed edge (u, v), vertex u comes before v in the ordering.
46. **What is the time complexity of finding a cycle in a graph?**

a. The time complexity of finding a cycle in a graph is O(V + E), where V is the number of vertices and E is the number of edges.

47. **What is the time complexity of a depth-first search (DFS) in a graph?**
    a. The time complexity of a depth-first search in a graph is O(V + E), where V is the number of vertices and E is the number of edges.
48. **What is the time complexity of a breadth-first search (BFS) in a graph?**
    a. The time complexity of a breadth-first search in a graph is O(V + E), where V is the number of vertices and E is the number of edges.
49. **What is Huffman coding?**
    a. Huffman coding is a lossless data compression algorithm that assigns variable-length codes to characters based on their frequencies, with more frequent characters having shorter codes.
50. **What is the time complexity of the merge sort algorithm?**
    a. The time complexity of the merge sort algorithm is O(n log n), where n is the number of elements to be sorted.
51. **What is a red-black tree?**
    a. A red-black tree is a self-balancing binary search tree with additional properties that ensure it remains balanced. It guarantees a worst-case time complexity of O(log n) for search, insert, and delete operations.
52. **What is the difference between a binary tree and a binary search tree (BST)?**
    a. In a binary tree, there are no specific rules for the order of elements, while in a BST, the left subtree of a node contains values smaller than the node, and the right subtree contains values greater than the node.
53. **What is a trie and how is it different from a binary search tree (BST)?**
    a. A trie, also known as a prefix tree, is a tree-like data structure primarily used for efficient retrieval of keys that share prefixes. Unlike a BST, a trie does not use comparisons to store or retrieve elements but instead relies on the structure of the keys themselves.
54. **What is a skip list?**
    a. A skip list is a probabilistic data structure that allows efficient search, insert, and delete operations. It consists of multiple layers of linked lists, with higher layers skipping elements, providing faster access to desired elements.
55. **What is the time complexity of searching in a skip list?**
    a. The time complexity of searching in a skip list is O(log n), where n is the number of elements stored in the list.
56. **What is a self-balancing binary search tree?**
    a. A self-balancing binary search tree automatically maintains its balance during insertions and deletions to ensure efficient search operations. Examples include AVL tree, Red-Black tree, and Splay tree.
57. **What is a self-balancing binary search tree rotation?**
    a. A rotation is an operation performed on a self-balancing binary search tree to maintain or restore balance. It involves moving nodes around to restructure the

tree while preserving the order of elements.

58. **What is the difference between a complete binary tree and a full binary tree?**
    a. A complete binary tree is a tree where all levels, except possibly the last, are fully filled, and all nodes are as left as possible. A full binary tree is a tree where all nodes have either 0 or 2 children.
59. **What is an in-order traversal of a binary tree?**
    a. In an in-order traversal, the left subtree is visited first, followed by the current node, and then the right subtree. This traversal visits the nodes in ascending order in a binary search tree.
60. **What is a post-order traversal of a binary tree?**
    a. In a post-order traversal, the left subtree is visited first, followed by the right subtree, and then the current node. This traversal is commonly used to delete nodes from a binary search tree.
61. **What is a pre-order traversal of a binary tree?**
    a. In a pre-order traversal, the current node is visited first, followed by the left subtree and then the right subtree. This traversal is useful for creating a copy of the tree.
62. **What is the difference between BFS (Breadth-First Search) and DFS (Depth-First Search) in a tree?**
    a. In a tree, both BFS and DFS visit each node exactly once. BFS explores the tree level by level, while DFS explores as deep as possible along each branch before backtracking.
63. **What is an adjacency matrix?**
    a. An adjacency matrix is a two-dimensional array that represents a graph. It indicates the presence or absence of an edge between two vertices using a 0 or 1.
64. **What is an adjacency list?**
    a. An adjacency list is a collection of linked lists or arrays used to represent a graph. Each vertex has a list of its neighboring vertices.
65. **What is a heap data structure?**
    a. A heap is a complete binary tree where each node is greater than or equal to (in a max heap) or less than or equal to (in a min heap) its child nodes. It is commonly used for efficient priority queue operations.
66. **What is the difference between a stack and a queue?**
    a. A stack follows the Last-In-First-Out (LIFO) principle, where the last element inserted is the first one to be removed. A queue follows the First-In-First-Out (FIFO) principle, where the first element inserted is the first one to be removed.
67. **What is the difference between a stack and a linked list?**
    a. A stack is an abstract data type that can be implemented using various data structures, including a linked list. A linked list is a linear data structure that consists of nodes, each containing a reference to the next node.
68. **What is the difference between a queue and a linked list?**

a. A queue is an abstract data type that can be implemented using various data structures, including a linked list. A linked list is a linear data structure that consists of nodes, each containing a reference to the next node.

69. **What is a hash table?**
    a. A hash table, also known as a hash map, is a data structure that uses a hash function to map keys to values. It provides fast insertion, deletion, and lookup operations.
70. **What is a collision in a hash table?**
    a. A collision occurs in a hash table when two or more keys map to the same index in the underlying array. Collision resolution techniques are used to handle such cases.
71. **What are some collision resolution techniques used in hash tables?**
    a. Common collision resolution techniques include chaining (using linked lists or arrays to store multiple values at the same index) and open addressing (probing for an alternative index when a collision occurs).
72. **What is a heapify operation in a heap?**
    a. Heapify is an operation that reorders the elements in a heap to maintain the heap property (e.g., max heap or min heap). It ensures that the parent node is greater (or smaller) than its children.
73. **What is the time complexity of heapify operation in a heap?**
    a. The time complexity of the heapify operation in a heap is O(log n), where n is the number of elements in the heap.
74. **What is a disjoint set data structure?**
    a. A disjoint set data structure is a data structure that keeps track of a partitioning of a set into disjoint subsets. It supports efficient operations to merge sets and determine whether elements belong to the same set.
75. **What is the time complexity of the union-find operation in a disjoint set data structure?**
    a. The time complexity of the union-find operation in a disjoint set data structure is typically O(log n), where n is the number of elements.
76. **What is the difference between a doubly linked list and a singly linked list?**
    a. In a doubly linked list, each node contains references to both the next and previous nodes, allowing traversal in both directions. In a singly linked list, each node only has a reference to the next node.
77. **What is a circular linked list?**
    a. A circular linked list is a linked list where the last node points back to the first node, creating a loop. It can be singly or doubly linked.
78. **What is a self-loop in a linked list?**
    a. A self-loop occurs in a linked list when a node points to itself, creating a loop within the list.
79. **What is the time complexity of inserting an element at the beginning of a linked list?**

a. The time complexity of inserting an element at the beginning of a linked list is O(1), as it only involves updating a few references.

80. **What is the time complexity of searching for an element in a linked list?**
    a. The time complexity of searching for an element in a linked list is O(n), where n is the number of elements in the list.
81. **What is the time complexity of deleting an element from a linked list?**
    a. The time complexity of deleting an element from a linked list depends on the position of the element. For deletion at the beginning, it is O(1), while for deletion at the end or a specific position, it is O(n).
82. **What is a self-balancing binary search tree rotation?**
    a. A rotation is an operation performed on a self-balancing binary search tree to maintain or restore balance. It involves rearranging nodes to restructure the tree while preserving the order of elements.
83. **What is a B-tree?**
    a. A B-tree is a self-balancing search tree that maintains sorted data and allows efficient insertions, deletions, and searches. It is commonly used in databases and file systems.
84. **What is the time complexity of searching in a B-tree?**
    a. The time complexity of searching in a balanced B-tree is O(log n), where n is the number of elements in the tree.
85. **What is a priority queue?**
    a. A priority queue is an abstract data type that stores elements with associated priorities and allows retrieval of the highest-priority element. It can be implemented using various data structures, such as heaps or binary search trees.
86. **What is the difference between a singly linked list and a doubly linked list?**
    a. In a singly linked list, each node contains a reference to the next node, while in a doubly linked list, each node contains references to both the next and previous nodes.
87. **What is the time complexity of inserting an element at the end of a linked list?**
    a. The time complexity of inserting an element at the end of a linked list is O(1) if there is a reference to the tail, and O(n) otherwise, as it requires traversing the entire list.
88. **What is the time complexity of reversing a linked list?**
    a. The time complexity of reversing a linked list is O(n), where n is the number of elements in the list, as each node needs to be visited once.
89. **What is a hash function?**
    a. A hash function is a function that takes an input (such as a key) and computes a fixed-size value (hash code or hash value) that represents the input. It is used in hashing-based data structures like hash tables.
90. **What are the characteristics of a good hash function?**
    a. A good hash function should produce a uniform distribution of hash codes, minimize collisions, and be computationally efficient.

91. **What is the difference between linear probing and quadratic probing?**
    a. Linear probing and quadratic probing are collision resolution techniques used in hash tables. Linear probing checks the next available slot in a linear manner, while quadratic probing checks slots based on a quadratic function.
92. **What is a sparse matrix?**
    a. A sparse matrix is a matrix that contains a significant number of zero elements compared to the total number of elements. It is often represented more efficiently using data structures like linked lists or hash tables.
93. **What is a graph traversal?**
    a. Graph traversal is the process of visiting all the vertices or nodes in a graph. Common traversal algorithms include depth-first search (DFS) and breadth-first search (BFS).
94. **What is the difference between a graph and a tree?**
    a. A tree is a type of graph that has no cycles, whereas a graph can have cycles. Additionally, a tree has a single root node, while a graph can have multiple disconnected components.
95. **What is the time complexity of searching in a binary search tree (BST)?**
    a. The time complexity of searching in a balanced binary search tree is O(log n), where n is the number of elements in the tree. In the worst case, an unbalanced BST can have a time complexity of O(n).
96. **What is a topological sort?**
    a. A topological sort is an ordering of the vertices of a directed graph such that for every directed edge (u, v), vertex u comes before vertex v in the ordering. It is used in scheduling and dependency resolution.
97. **What is the difference between a stack and a heap?**
    a. A stack is a region of memory used for local variables and function call information, while a heap is a region of memory used for dynamic memory allocation.
98. **What is the time complexity of inserting an element into a binary search tree (BST)?**
    a. The time complexity of inserting an element into a balanced binary search tree is O(log n), where n is the number of elements in the tree. In the worst case, an unbalanced BST can have a time complexity of O(n).
99. **What is a self-balancing binary search tree?**
    a. A self-balancing binary search tree is a binary search tree that automatically maintains balance after insertions and deletions to ensure efficient search operations. Examples include AVL tree, Red-Black tree, and Splay tree.
100. **What is the time complexity of the merge sort algorithm?**
    - The time complexity of the merge sort algorithm is O(n log n), where n is the number of elements to be sorted.

.