Quiz Performance Report

Name: Computer

Topic: Data structures and Algorithms

Date: 2025-09-16 10:54:23

Final Score: 7 / 10

AI Performance Summary:

Could not generate a detailed analysis. Your final score was 7/10. Please try again later.

AI Recommendations & Resources:

No recommendations available.

Question Breakdown:

Question 1: Which data structure follows the Last-In, First-Out (LIFO) principle?
Your Answer: Stack (Correct ■)
Correct Answer: Stack
Difficulty: easy

Question 2: Which data structure follows the First-In-First-Out (FIFO) principle?
Your Answer: Linked List (Incorrect ■)
Correct Answer: Queue
Difficulty: easy

Question 3: Which data structure follows the Last-In, First-Out (LIFO) principle?
Your Answer: Stack (Correct ■)
Correct Answer: Stack
Difficulty: easy

Question 4: Which data structure follows the Last-In, First-Out (LIFO) principle?
Your Answer: Stack (Correct ■)
Correct Answer: Stack
Difficulty: easy

Question 5: Which of the following data structures would be MOST efficient for implementing a LRU (Least Recently Used) cache with a fixed capacity?
Your Answer: A doubly linked list combined with a hash table. (Correct ■)
Correct Answer: A doubly linked list combined with a hash table.
Difficulty: medium

Question 6: Which data structure would be MOST efficient for implementing a Last-In-First-Out (LIFO) queue that needs to support frequent insertions and deletions at the same end?
Your Answer: A stack, utilizing an array or linked list to efficiently manage LIFO operations. (Correct ■)
Correct Answer: A stack, utilizing an array or linked list to efficiently manage LIFO operations.
Difficulty: medium

Question 7: Which of the following statements BEST describes the time complexity of searching for a specific element in a sorted linked list?
Your Answer: O(1) - Constant time, as the element can be accessed directly if its index is known. (Incorrect ■)
Correct Answer: O(n) - Linear time, as in the worst case, the entire list needs to be traversed.
Difficulty: medium

Question 8: You are designing a system for real-time stock trading where extremely fast lookups of stock prices are crucial. The system needs to handle millions of stock symbols and their corresponding prices, which are constantly updated. Which data structure would be MOST efficient for minimizing the average-case time complexity of both insertion and retrieval operations, while also considering memory overhead and potential for concurrent updates from multiple threads?
Your Answer: A hash table (with separate chaining or open addressing) utilizing a high-quality hash function and optimized for concurrent access with techniques like lock-free data structures. (Correct ■)
Correct Answer: A hash table (with separate chaining or open addressing) utilizing a high-quality hash function and optimized for concurrent access with techniques like lock-free data structures.
Difficulty: hard

Question 9: You are tasked with designing a data structure to efficiently handle a stream of incoming data points (x, y coordinates) that requires fast nearest-neighbor queries within a specific radius. The data points are constantly updated (insertions and deletions). Which data structure and algorithm combination would offer the best overall performance in terms of both query time and update time, considering that the number of data points can reach millions?
Your Answer: An R-tree, leveraging its spatial indexing capabilities for efficient nearest-neighbor searches and updates. (Correct ■)
Correct Answer: An R-tree, leveraging its spatial indexing capabilities for efficient nearest-neighbor searches and updates.
Difficulty: hard

Question 10: You are tasked with designing a data structure to efficiently handle a stream of incoming data points, each consisting of a timestamp and a value. The data points must be queried efficiently for ranges of timestamps, and the structure must support insertions and deletions in O(log n) time, where n is the number of data points. Which data structure offers the best asymptotic time complexity for both insertion/deletion and range queries?
Your Answer: A skip list, which provides probabilistic guarantees of O(log n) time complexity for insertion, deletion, and range queries, but might exhibit worse performance in worst-case scenarios compared to balanced trees. (Incorrect ■)
Correct Answer: A self-balancing binary search tree (e.g., AVL tree or red-black tree) augmented with a size field for each node to enable efficient range queries.
Difficulty: hard