Mock Test

Topic: javascript
Difficulty: Advanced
Total Questions: 5
Time Allowed: 10 minutes

Instructions:

1. Attempt all questions
2. Each question carries equal marks
3. Time allowed: 10 minutes

1. Implement a function that efficiently finds the longest palindrome substring within a given string. Consider edge cases and optimize for performance.

2. Given a binary tree, write a function to determine if it is a valid binary search tree (BST). Handle potential edge cases and optimize for time and space complexity.

3. Design a LRU (Least Recently Used) cache using JavaScript. The cache should support `get(key)` and `put(key, value)` operations with a fixed capacity. Explain your chosen data structures and time/space complexity.

4. Implement a function that checks if a given graph is strongly connected using Depth-First Search (DFS). Handle both directed and undirected graphs. Explain your approach and its time complexity.

5. Write a function that takes an array of integers and returns the kth smallest element in linear time (O(n)) using the QuickSelect algorithm. Handle edge cases such as empty arrays and k exceeding the array length.

Answer Key

1. Correct Answer: null
Explanation: Efficient solutions typically involve dynamic programming or Manacher's algorithm. Edge cases include empty strings, strings with single characters, and strings with multiple palindromes of equal length.

2. Correct Answer: null
Explanation: Solutions often involve recursive in-order traversal to check if the nodes are sorted. Efficient solutions avoid redundant checks.

3. Correct Answer: null
Explanation: Efficient implementations typically use a combination of a doubly linked list and a hash map to achieve O(1) time complexity for both `get` and `put` operations.

4. Correct Answer: null
Explanation: For directed graphs, perform two DFS traversals: one from an arbitrary node and another from its transpose. For undirected graphs, a single DFS is sufficient.

5. Correct Answer: null
Explanation: QuickSelect is based on the partitioning logic of Quicksort but only recursively processes one partition, significantly improving performance for finding the kth smallest element. Randomized pivot selection helps ensure better average-case performance.