Questions for data structures and algorithms

Question

1. **Analyze the time and space complexity of a solution to the k-th smallest element problem in an unsorted array, comparing and contrasting the efficiency of using a min-heap versus a quickselect algorithm. Consider both average-case and worst-case scenarios, detailing the specific conditions that lead to worst-case performance in each approach.** (This question probes deep understanding of algorithmic analysis, heap data structures, and divide-and-conquer strategies.)
5. **Develop an algorithm to find the longest palindromic substring within a given string. Compare the time and space complexity of your approach to other potential algorithms (e.g., dynamic programming, Manacher's algorithm). Discuss the trade-offs between different algorithmic approaches and their suitability for various string lengths.** (This assesses problem-solving skills and understanding of algorithmic optimization techniques.)
2. **Describe a scenario where a trie (prefix tree) would be a significantly more efficient data structure than a hash table for storing and searching a large vocabulary. Quantify the advantage in terms of Big O notation, considering specific operations like prefix searching and autocompletion. Justify your choice with concrete examples and limitations of each data structure in this context.** (This necessitates understanding the strengths and weaknesses of different data structures in specific applications.)
3. **Design an algorithm to detect cycles in a directed graph using depth-first search (DFS) or breadth-first search (BFS). Explain your chosen approach and its time and space complexity. Furthermore, modify your algorithm to not only detect cycles but also to return the nodes that constitute the cycle.** (This demands understanding graph traversal algorithms and cycle detection techniques within graphs.)
6. **Explain how consistent hashing can solve the problem of data redistribution when adding or removing nodes from a distributed hash table (DHT). Discuss the benefits of consistent hashing over simpler approaches like modulo hashing. Analyze the impact of node failures and the mechanisms to handle them efficiently within a consistent hashing scheme.** (This requires understanding of distributed systems, hashing techniques, and strategies for
4. **Compare and contrast the performance characteristics of different self-balancing binary search trees (AVL trees, red-black trees, B-trees) in the context of database indexing. Discuss their advantages and disadvantages in terms of search, insertion, and deletion operations, considering scenarios with both high and low update frequencies. Explain how the choice of tree impacts database query performance.** (This tests knowledge of advanced tree structures and their application in real-world systems.)