Questions for data structures

Question
5. **Question:** Compare and contrast the use of arrays, linked lists, and stacks in implementing a recursive algorithm for solving the Tower of Hanoi problem. Discuss the trade-offs between memory usage, execution time, and stack overflow prevention in each data structure. **Hint:** You may need to know the concepts of recursive function calls, stack frames, and memory allocation in each data structure. I hope these questions help!
4. **A graph database needs to support highly efficient shortest path queries between nodes, even with millions of nodes and edges. Compare and contrast the suitability of Dijkstra's algorithm implemented using different graph representations (adjacency matrix, adjacency list, etc.). Analyze the space and time complexity of each implementation, including the impact of sparse vs. dense graphs. Discuss potential optimizations for improving performance in the context of real-world applications.** (This requires deep knowledge of graph algorithms and their relationship to data structure choices).
4. **Question:** Describe the process of converting a linked list of nodes to a doubly linked list, including the necessary updates to each node's "next" and "prev" pointers. How does this conversion affect the time complexity of common operations like insertion, deletion, and traversal? **Hint:** You may need to know the differences between singly linked lists and doubly linked lists, as well as the implications for node traversal and memory management.
2. **Question:** Compare and contrast the efficiency of binary search trees (BSTs) and B-trees for storing a large dataset of integers, considering factors such as search, insertion, and deletion operations. Assume the dataset is sorted in ascending order. **Hint:** You may need to know the trade-offs between node depth, search time, and disk storage in BSTs and B-trees.
1. **Discuss the trade-offs involved in choosing between a B-tree and a Trie for indexing a large dataset of strings, considering factors like search speed, storage efficiency, and update complexity. Provide specific scenarios where one would be demonstrably superior to the other, justifying your answer with concrete examples and analytical reasoning.** (This requires understanding of both data structures, their complexities, and ability to apply that knowledge to real-world scenarios.)