Red-Black Tree (RB Tree)

1. What are the properties that define a Red-Black Tree?
2. Why is a Red-Black Tree considered a self-balancing binary search tree?
3. Explain the insertion process in a Red-Black Tree and the different cases of rebalancing.
4. What is the significance of color changes and rotations during the insertion in a Red-Black Tree?
5. Compare the time complexity of insertion in a Red-Black Tree with that of an AVL tree.

Simple Insertion Questions

1. Insert the following elements into an initially empty Red-Black Tree: **10, 20, 30**. Show the tree after each insertion and explain any rotations or color changes.
2. Insert the elements **15, 40, 50, 25** into an empty Red-Black Tree. Explain how the tree maintains its balance.
3. Insert **7, 3, 18, 10, 22, 8, 11, 26** into an empty Red-Black Tree and describe the transformations (rotations and recoloring) that occur.
4. What happens when inserting a node into a Red-Black Tree when its parent is red? Explain with an example.
5. Insert the numbers **5, 15, 25, 35, 45, 55** into an empty Red-Black Tree and illustrate the balancing process.

B-Tree

1. What are the main applications of B-Trees in database indexing and file systems?
2. Explain the insertion process in a B-Tree. What happens when a node overflows?
3. Describe the deletion process in a B-Tree. How is underflow handled?
4. How does a B-Tree maintain balance compared to an AVL or Red-Black Tree?
5. Why is a B-Tree preferred for disk-based storage systems?

Simple Insertion Questions

(Assume a **B-tree of order 3 (minimum degree t = 2)**, meaning each node can have **at most 3 children** and **at least 2 children**.)

1. Insert the numbers **10, 20, 5** into an initially empty B-tree of order 3. Show the structure after each step.
2. Insert the elements **30, 40, 50, 60** into a B-tree of order 3. Explain when and how the node splits.
3. Consider a B-tree of order 3. Insert **15, 25, 35, 45, 55, 65** into an empty B-tree. Show the tree at each step.

4. Insert **8, 9, 10, 11, 12, 13, 14** into a B-tree of order 3. How many splits occur, and what does the final tree look like?
5. Insert **1, 2, 3, 4, 5, 6, 7** into an empty B-tree of order 3. Show the tree and explain how balancing is maintained.

Binomial Heap

1. What is a binomial heap, and how does it differ from a binary heap?
2. Explain the process of merging two binomial heaps.
3. What are the steps involved in performing a union operation on two binomial heaps?
4. How does the structure of a binomial heap enable efficient merge operations?
5. What is the time complexity of merging two binomial heaps, and why?