Heap in Trees – Summary

A **Heap** is a specialized tree-based data structure that follows strict rules to keep data organized efficiently. It is mainly used when fast access to the minimum or maximum element is required.

1. Structural Property (Complete Binary Tree)

A heap is always a **complete binary tree**. This ensures the tree remains balanced, which guarantees logarithmic time complexity for major operations.

2. Heap Property

1. **Max Heap:** Parent node value ≥ children values.
2. **Min Heap:** Parent node value ≤ children values.

3. Representation

Although a heap is conceptually a tree, it is commonly implemented using an **array**. For a node at index i: left child = 2i + 1, right child = 2i + 2, parent = (i – 1) / 2.

4. Heap Operations

1. **Insertion:** Insert at the last position, then heapify up.
2. **Deletion:** Remove root, replace with last node, heapify down.
3. **Heapify:** Process of restoring heap property.

Additional Notes (Quick Revision)

1. Heap height is O(log n) because it is a complete binary tree.
2. The root node always contains the minimum or maximum element.
3. Heap does not guarantee sorted order between siblings.
4. Heap Sort is not stable but works in O(n log n) time.
5. Building a heap from an array takes O(n), not O(n log n).

Common Mistakes

1. Confusing Heap with Binary Search Tree (BST).
2. Thinking heaps store elements in sorted order.
3. Ignoring the complete binary tree property.