Heap

A **heap** is a special **tree-based** data structure that satisfies the **heap property**:

- **Max-Heap**: The value of each parent node is **greater than or equal to** its children.
- **Min-Heap**: The value of each parent node is **less than or equal to** its children.

It is usually implemented as a **binary tree** (specifically, a **complete binary tree**), and stored as an **array.**

Complete Binary Tree-

=> All Levels are completely filled except the last levels.

=> At last level, Node should be filled from left side

Why Do We Use Heap

1. **Priority Queues**
   Heaps are used to implement **priority queues**, where we need fast access to the highest or lowest priority element.
2. **Efficient Operations**
   - Insertion: `O(log n)`
   - Deletion (max/min): `O(log n)`
   - Get max/min: `O(1)`
3. **Heap Sort**
   Sorting algorithm that uses heaps to sort an array in `O(n log n)` time.

4. **Finding Top K Elements**
   Efficiently find the **k largest/smallest elements** from a large dataset.

5. **Used in Graph Algorithms**
   Algorithms like **Dijkstra's shortest path** and **Prim's minimum spanning tree** use heaps to pick the next minimum-cost edge or node.

6. **Scheduling Tasks**
   Operating systems use heaps for **task scheduling**, giving priority to more important tasks.

Basic Operations on Heap

(i) Insertion (O(log n))

1. Insert new element at the **end** of the heap (last position).
2. Perform **Heapify Up (Bubble Up)**: compare with parent and swap until heap property is restored.

(ii) Deletion (Remove Root) (O(log n))

1. Root element is removed (min in Min Heap, max in Max Heap).
2. Replace root with the **last element**.
3. Perform **Heapify Down (Bubble Down)** to restore heap property.

(iii) Heapify

- Process of **converting an array into a valid heap**.
- Done in **O(n)** time.

Applications of Heap

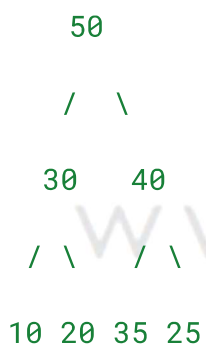
1. **Heap Sort** → Efficient sorting algorithm with O(n log n).
2. **Priority Queues** → Tasks scheduled based on priority.
3. **Graph Algorithms**
    - Dijkstra's shortest path
    - Prim's Minimum Spanning Tree
4. **Median Finding** (using two heaps).
5. **Order Statistics** (Kth largest/smallest element).

Types of Heaps

1. Max Heap

- Root contains the **largest element.**

Example:



2. Min Heap

- Root contains the **smallest element.**

Example:

```
      10
     /  \
    20   15
   / \   / \
  30 40 50 60
```

Operation	Time Complexity
Insertion	O(log n)
Deletion (extract min/max)	O(log n)
Get Min/Max	O(1)
Build Heap (Heapify)	O(n)

Heap Sort	O(n log n)