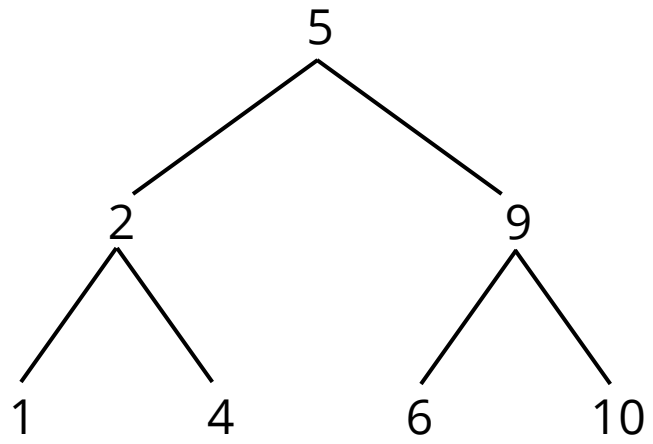
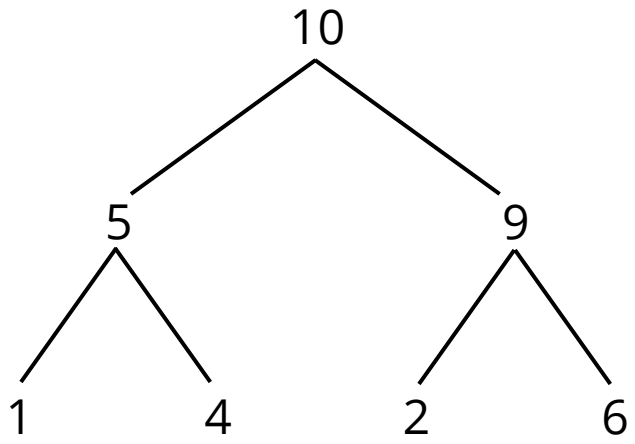
Binary Search Tree

left value < root < right value



Heap

parent ≥ children (max-heap)



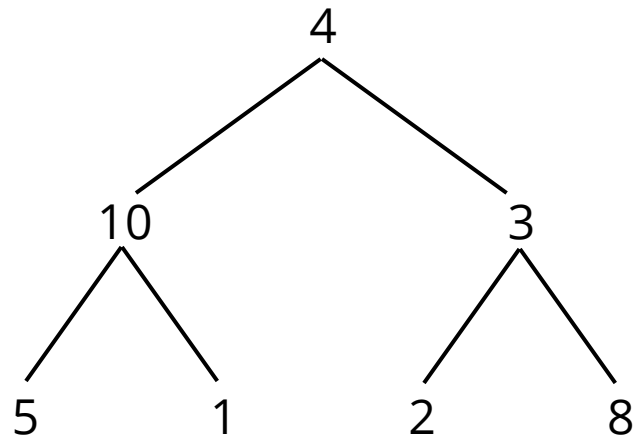
- Ordered binary tree
- PRIORITY QUEUE: urgent tasks (high priority) are handled before others

Heap Sort

Steps:

1. Build a heap from the array.
2. Repeatedly remove the top (max), move it to end, re-heapify (Step 1)

Example: Perform Heap Sort for [4, 10, 3, 5, 1, 2 ,8]



Hashing

- map data (keys) to fixed-size values (hashes) for fast access
- Search time: O(1) on average

Example:

Hash function: $h(x) = x \% 5$

Hash Table	
0	
1	
2	

Hashing

- map data (keys) to fixed-size values (hashes) for fast access
- Search time: O(1) on average

Example:

Hash function: $h(x) = x \% 5$

Insert 10 $\rightarrow h(10) = 0 \rightarrow$ Store at index 0
Insert 17 $\rightarrow h(17) = 2 \rightarrow$ Store at index 2

Hash Table	
0	10
1	
2	17

Hashing

- map data (keys) to fixed-size values (hashes) for fast access
- Search time: O(1) on average

Example:

Hash function: $h(x) = x \% 5$

Insert 10 $\rightarrow h(10) = 0 \rightarrow$ Store at index 0
Insert 17 $\rightarrow h(17) = 2 \rightarrow$ Store at index 2

Hash Table	
0	10
1	
2	17

Common Problem: **Collisions**

- A collision happens when two keys map to the same index

Insert 2 $\rightarrow h(2) = 2 \rightarrow$ Store at index 2 ⚡

Solving Strategies: Cuckoo hashing, Linear probing, ...