CS2x1:Data Structures and Algorithms

Koteswararao Kondepu

k.kondepu@iitdh.ac.in

Motivation

Motivation

- Sequential Data Structures
    - Linked List
    - Stack
    - Queue

→

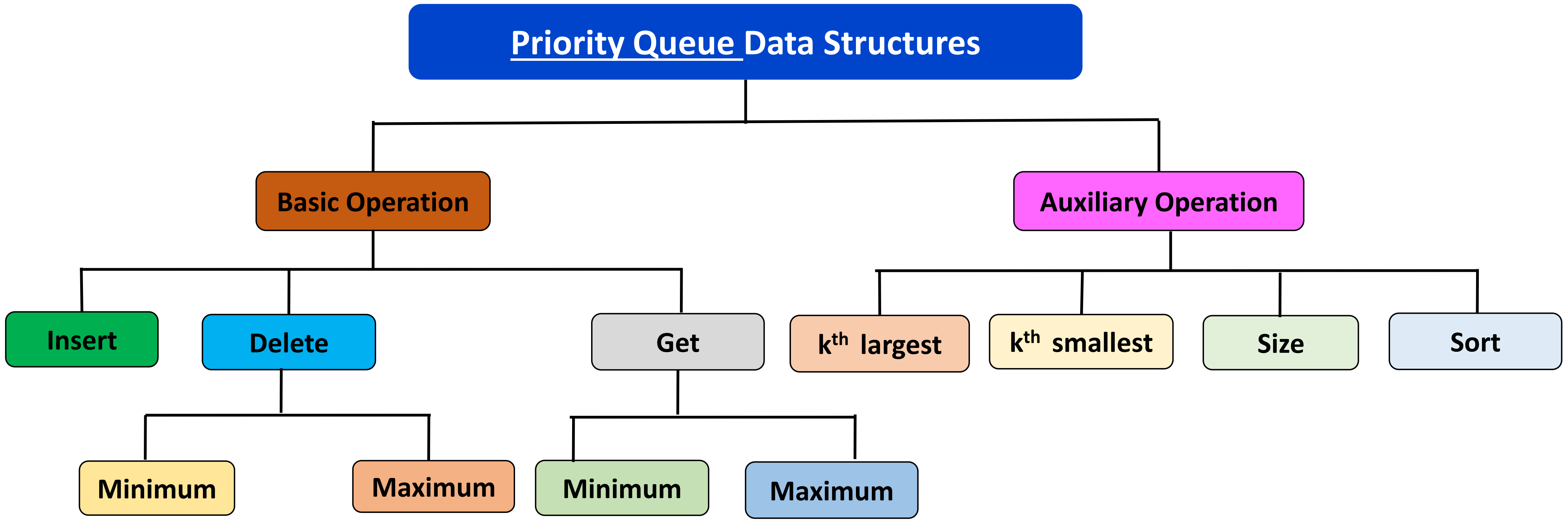
Restriction of access to the elements
1 element at a time for each operation

- Data structure → total ordering of elements i.e., ***which maintains all the elements in sorted order*** → ***their implementation and design might be difficult***

- ***Data structure → Restriction of access to the elements + maintain an implied ordering of elements (i.e., not necessary total ordering)***

Priority Queue (1)

Define: A *Priority Queue* (like a Queue) has a restriction in accessing elements, but each element has an implicit "*priority*", such that the element with *the highest ("max") priority* is always *DeQueued*, regardless of the order in which it was *EnQueued*.



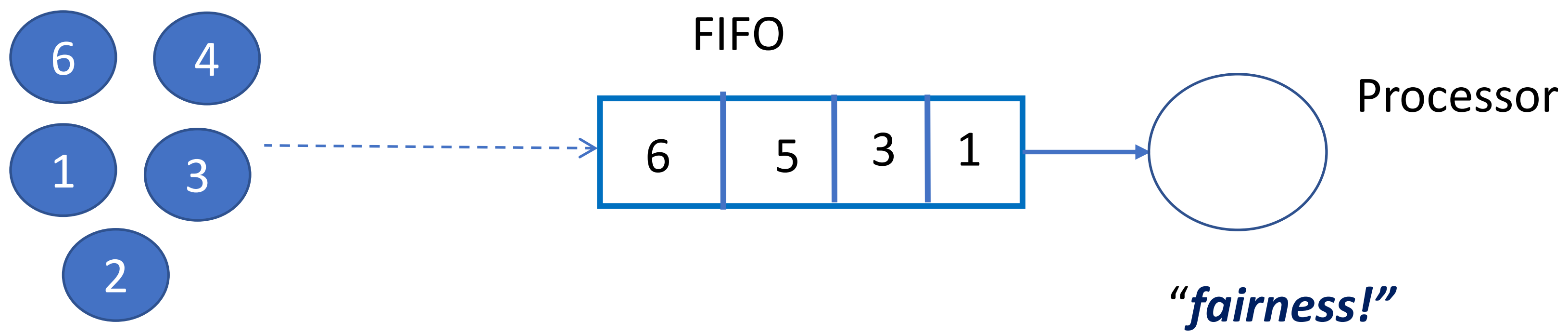
Priority Queue (2)

Define: A *Priority Queue* (like a Queue) has a restriction in accessing elements, but each element has an implicit "*priority*", such that the element with *the highest ("max") priority* is always *DeQueued*, regardless of the order in which it was *EnQueued*.

- The order in which the elements enter the queue may not the same in which they were processed
- Application → job scheduling

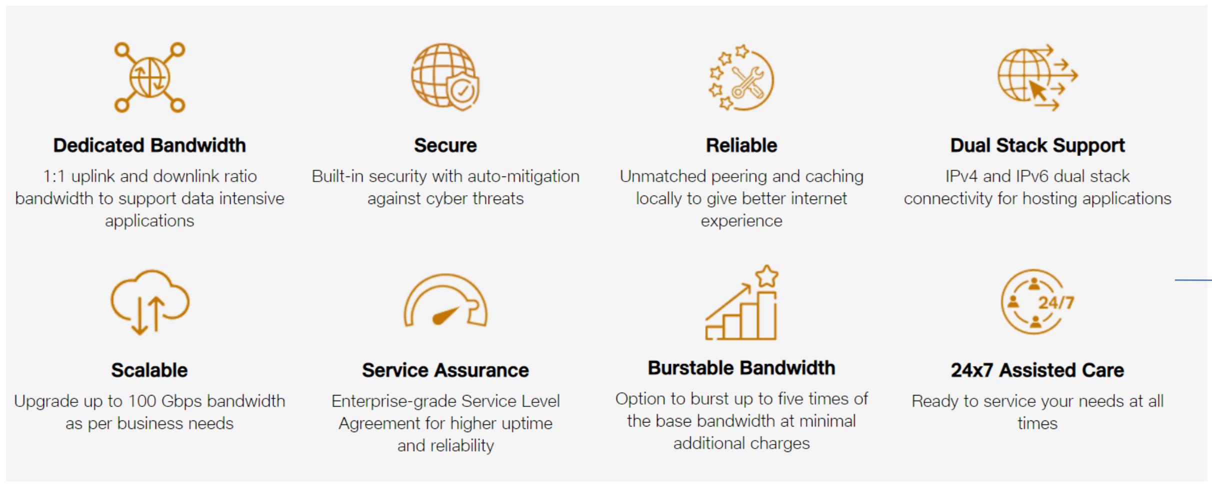
- *Ascending – Priority Queue:* smallest key has the highest priority → delete smallest element always
- *Descending – Priority Queue:* largest key has the highest priority → delete largest element always

Priority Queue (3)



"fairness!"

- Business Leased Line
- Point-to-Point Leased Line
- Virtual Leased Line



Dedicated Bandwidth
1:1 uplink and downlink ratio bandwidth to support data intensive applications

Secure
Built-in security with auto-mitigation against cyber threats

Reliable
Unmatched peering and caching locally to give better internet experience

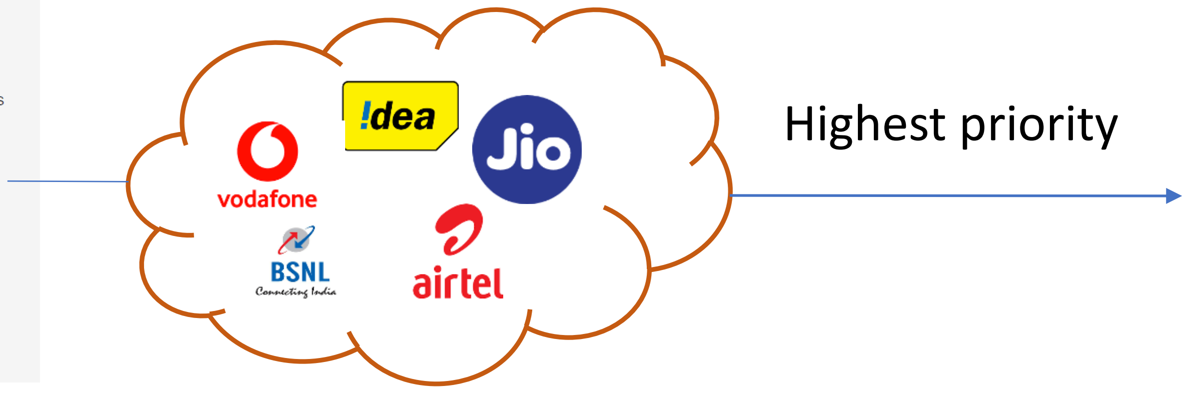
Dual Stack Support
IPv4 and IPv6 dual stack connectivity for hosting applications

Scalable
Upgrade up to 100 Gbps bandwidth as per business needs

Service Assurance
Enterprise-grade Service Level Agreement for higher uptime and reliability

Burstable Bandwidth
Option to burst up to five times of the base bandwidth at minimal additional charges

24x7 Assisted Care
Ready to service your needs at all times



Priority Queue

Applications

- *Data compression: Huffman Coding algorithm*
- *Shortest path algorithm: Dijkstra's algorithm*
- *Minimum Spanning tree algorithm: Prim's algorithm*
- *Event-driven simulation: customers in a line*
- *Finding k^{th} Smallest element*

Heap (1)

__Heap Properties:__

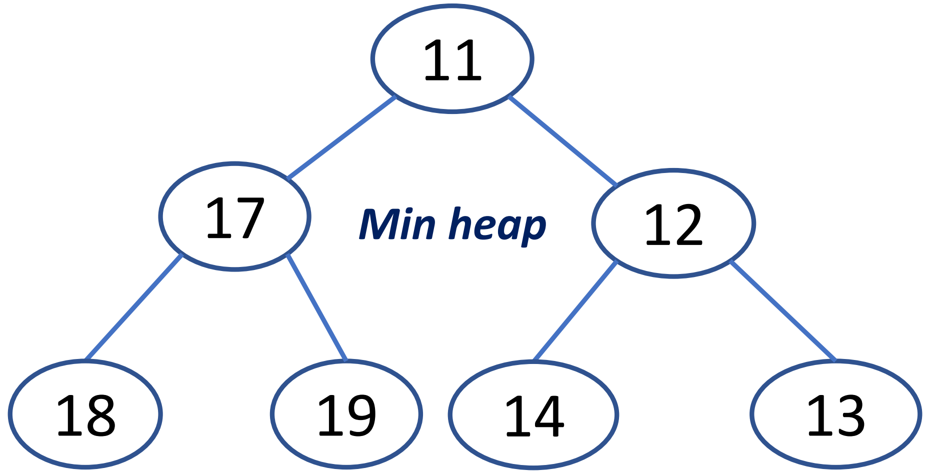
A heap is a tree with some special properties:

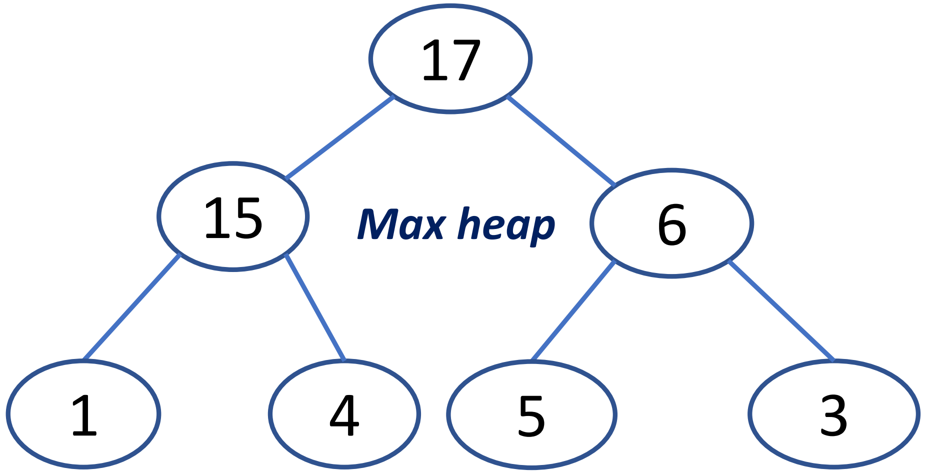
(i) the values of a node must be ≥ or ≤ to the values of its child

- ✓ ***Min heap***: The value of a node must be less than or equal (≤) to the value of its children
- ✓ ***Max heap:*** The value of a node must be greater than or equal (≥) to the values of its children

(ii) the leaf nodes of a heap should be at *h* or *h-1*, where *h is* the height of the tree → *complete binary tree*

After operations (*add, max, pop- max*) → it must be a complete binary tree



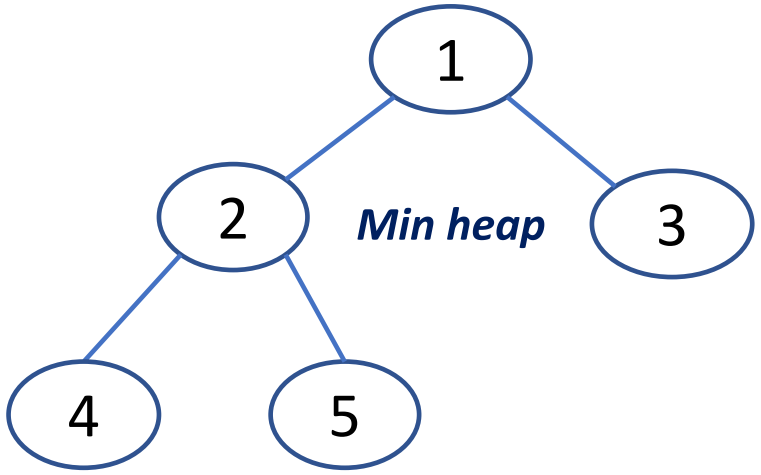


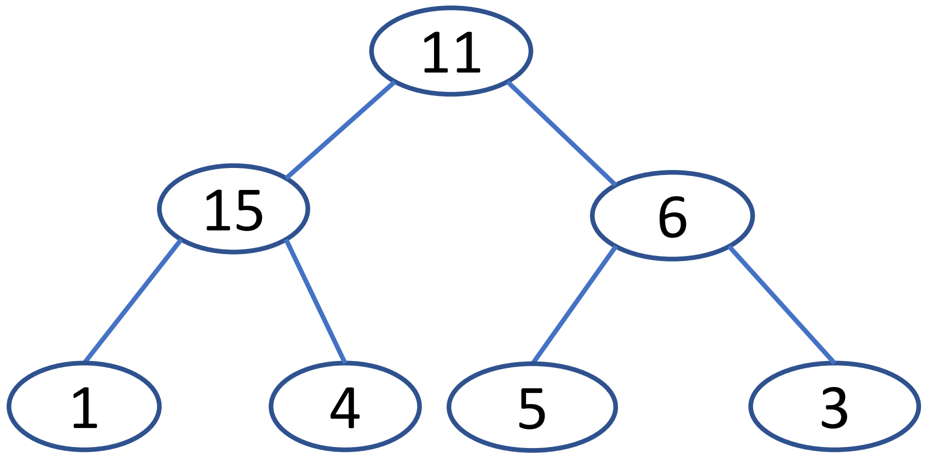
In any subtree, *__the maximum values should be found at the root__*

Heap (2)

Heap Properties:

(ii) the leaf nodes of a heap should be at *h* or *h-1*, where *h is* the height of the tree → *complete binary tree*





Heap?????

Not a Heap

Max-Heap – Insertion

Construct <u>Max-heap</u> with the following elements: Elements are inserted in sequential order :

25, 45, 30, 60, 70, 5, 9, 1

- ✓ ***Max heap:*** The value of a node must be greater than or equal (≥) to the values of its children

Min-Heap - Insertion

Construct Min-heap with the following elements: Elements are inserted in sequential order:

25, 45, 30, 60, 70, 5, 9, 1

✓ ***Min heap***: The value of a node must be less than or equal (≤) to the value of its children

Exercise: Heap- Insertion (1)

A priority-Queue is implemented as a max-heap. Initially, it has 5 elements. The level order traversal of the heap is: 10, 8, 5, 3, 2. The new element 1 and 7 are inserted into heap in that order. The level order traversal of the heap after the insertion of the element is:

A. *10, 8, 7, 3, 2, 1, 5*
B. *10, 8, 7, 2, 3, 1, 5*
C. *10, 8, 7, 1 2, 3, 5*
D. *10, 8, 7, , 3, 2, 1*

Exercise: Heap- Insertion (2)

Suppose the elements {27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0} are inserted in that order into the valid 3-ary max heap. Construct 3-ary max heap.?

Heap: Deletion (1)

Heap: Deletion Properties:

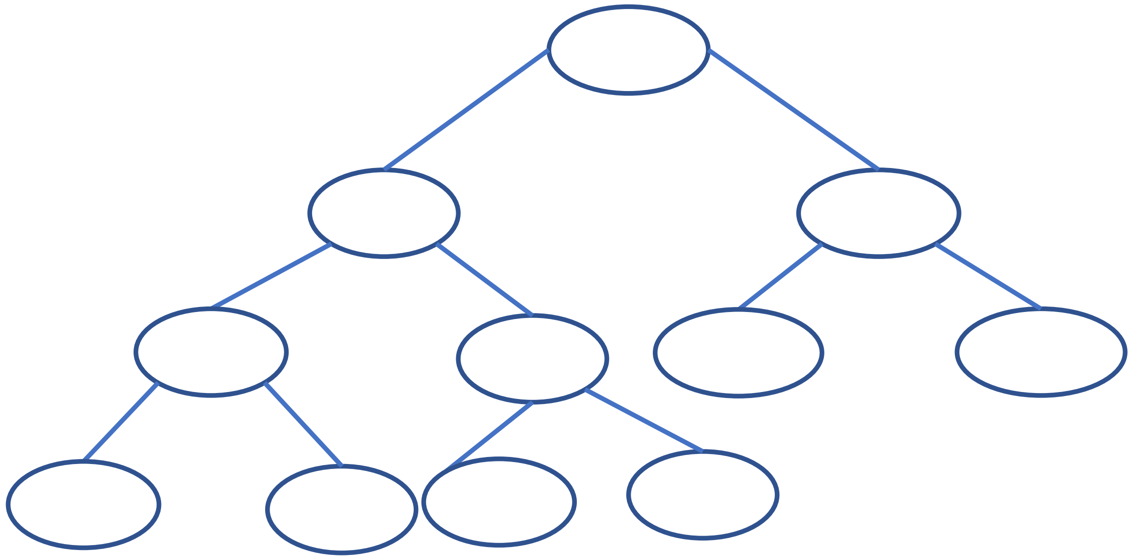
(i) Deleting the root node in the heap

(ii) Deleting any node in the heap

(i) Deleting the root node in the heap (*i.e., consider Max-heap*)

- Step#1: read the root node
- Step#2: replace the root node by the last node in the heap tree
- Step#3: reconstruct the modified tree by following the heap properties, recursively.

Example: *99, 45, 63, 35, 29, 57, 42, 27, 12, 24, 26*

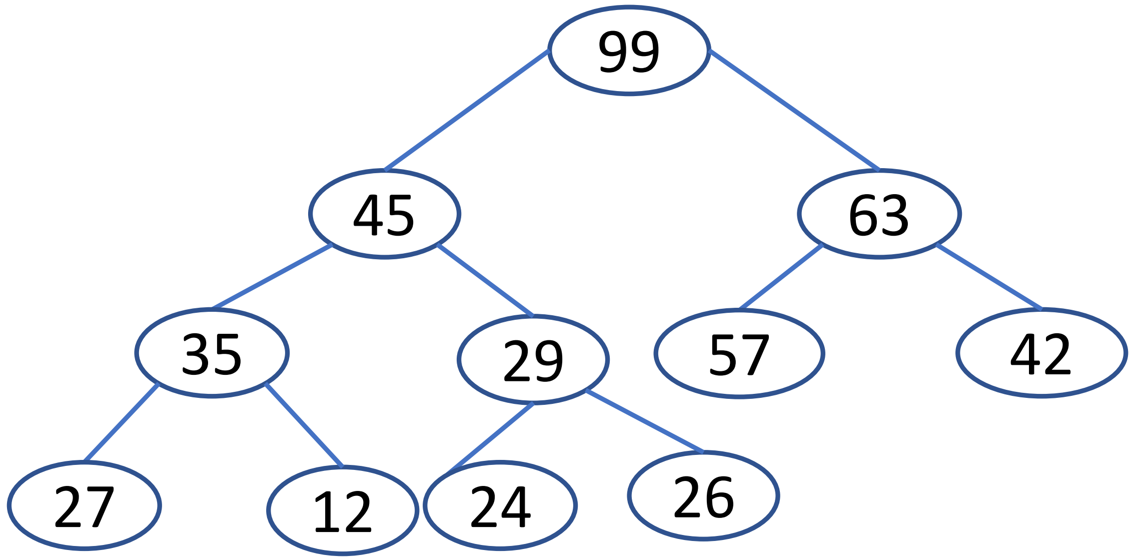


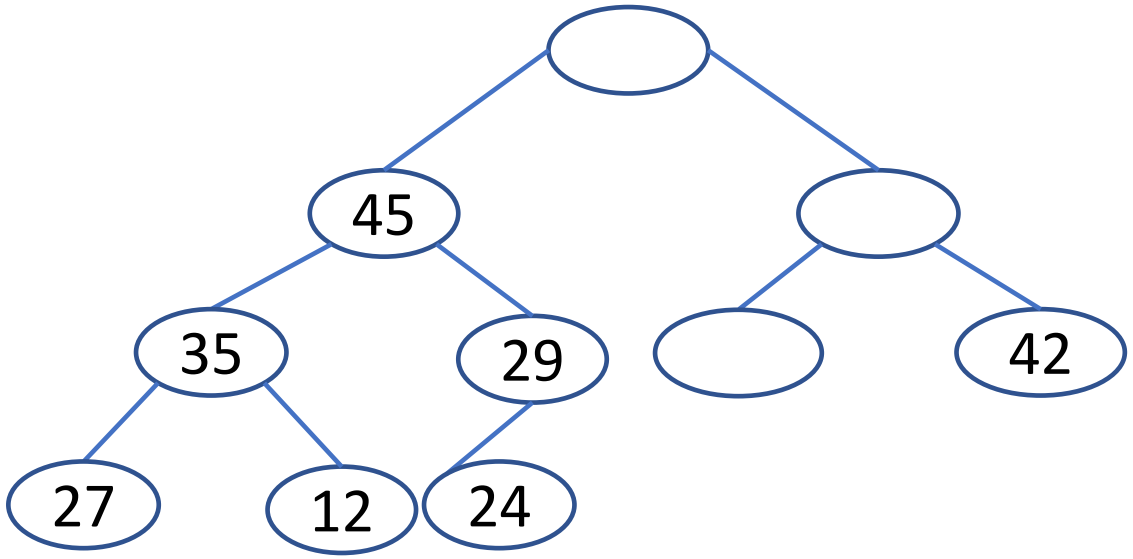
Heap: Deletion (2)

(i) Deleting the root node in the heap (*i.e., consider <u>Max-heap</u>*)

- Step#1: read the root node
- Step#2: replace the root node by the last node in the heap tree
- Step#3: reconstruct the modified tree by following the heap properties, recursively.

Example: *99, 45, 63, 35, 29, 57, 42, 27, 12, 24, 26*



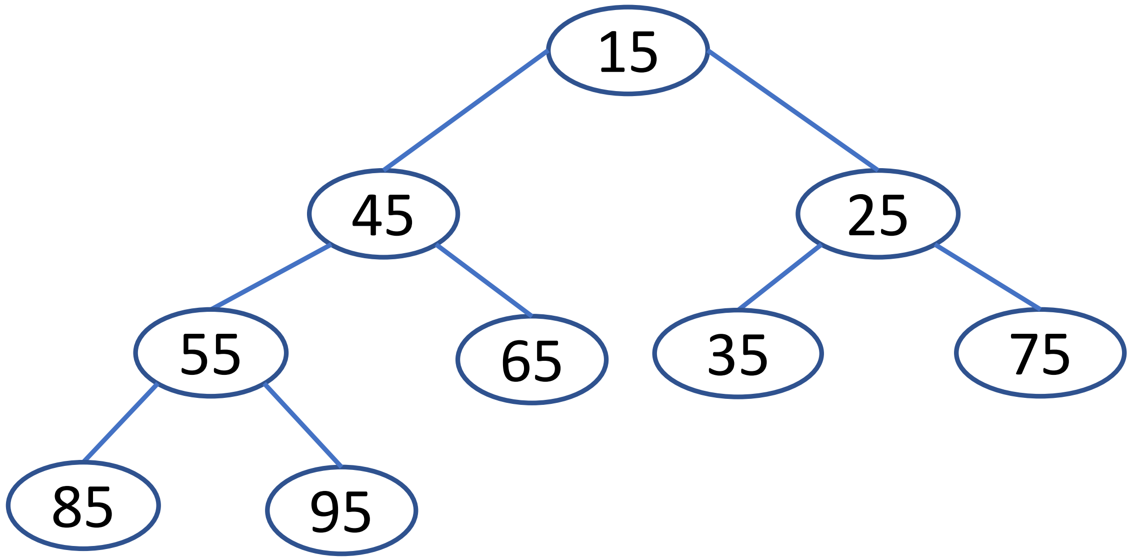


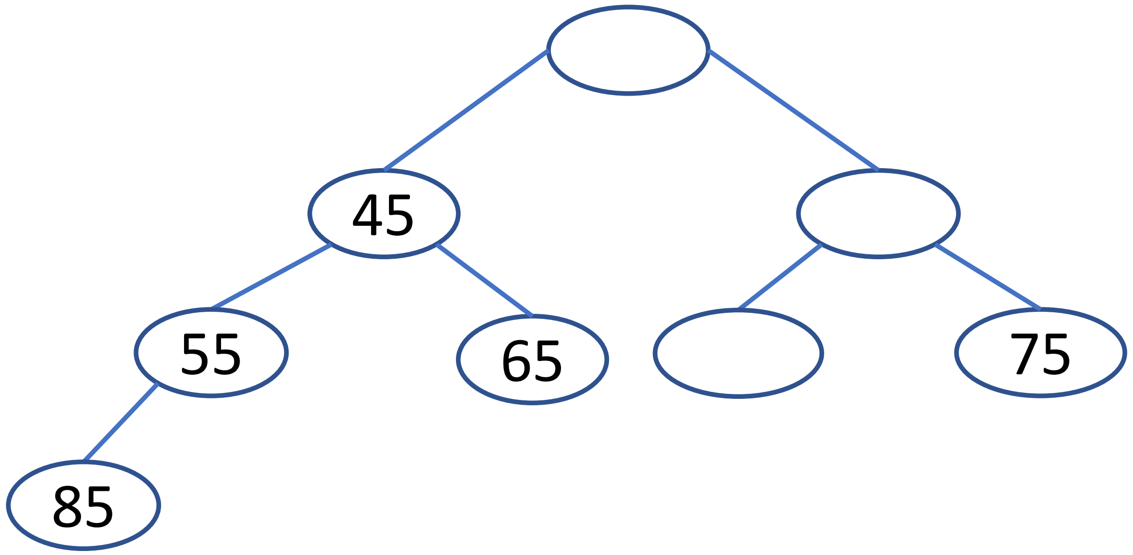
After deleting root node 99

Heap: Deletion (3)

(i) Deleting the root node in the heap (*i.e., consider <u>Min-heap</u>*)

- Step#1: read the root node
- Step#2: replace the root node by the last node in the heap tree
- Step#3: reconstruct the modified tree by following the heap properties, recursively.





After deleting root node 15

Heap: Deletion

A binary max heap has its level order traversal as 140, 90, 70, 46, 64, 16, 42, 10, 24, 9, 20. If the element 90 is deleted from the max heap, then the post-order traversal of the binary max heap will be?

A. 140, 64, 46, 10, 24, 20, 9, 70, 16, 42
B. 10, 46, 24, 64, 9, 20, 16, 70, 42, 140
C. 10, 24, 46, 9, 20, 64, 16, 42, 70, 140

thank you!

email:
k.kondepu@iitdh.ac.in