PRIORITY QUEUE

A **Priority Queue** is a data structure in which:

- Each element has a **priority**
- Element with **highest priority** (or lowest) is removed first

Types of Priority Queue

1. **Max Priority Queue** – highest priority element is removed first
2. **Min Priority Queue** – lowest priority element is removed first

Basic Operations

- **Insert** – add an element
- **Delete** – remove highest/lowest priority element
- **Peek** – view highest/lowest priority element

IMPLEMENTATIONS OF PRIORITY QUEUE

1. Priority Queue using Array

a) Unsorted Array

- Elements stored without order

Operations & Time Complexity

Operation	**Time**
Insert	O(1)
Delete (highest priority)	O(n)
Peek	O(n)

Disadvantage

- Slow deletion

b) Sorted Array

- Elements stored in sorted order

Operations & Time Complexity

Operation	**Time**
Insert	O(n)
Delete	O(1)
Peek	O(1)

Disadvantage

- Slow insertion

2. Priority Queue using Linked List

a) Unsorted Linked List

- No priority order

Time Complexity

Operation	**Time**
Insert	O(1)
Delete	O(n)
Peek	O(n)

b) Sorted Linked List

- Nodes arranged by priority

Time Complexity

Operation	**Time**
Insert	O(n)
Delete	O(1)
Peek	O(1)

3. Priority Queue using Binary Heap (Binary Tree)

Binary Heap

- Complete Binary Tree

- Two types:
  - **Max Heap**
  - **Min Heap**

Properties

- Parent has higher (or lower) priority than children
- Height is **log n**

Operations & Time Complexity

Operation	**Time**
Insert	O(log n)
Delete	O(log n)
Peek	O(1)

Advantage

- Efficient insertion and deletion

COMPARISON SUMMARY

Implementation	**Insert**	**Delete**
Unsorted Array / LL	O(1)	O(n)
Sorted Array / LL	O(n)	O(1)
Binary Heap	O(log n)	O(log n)

Applications

- CPU Scheduling
- Dijkstra’s Algorithm
- Job Scheduling
- Huffman Coding

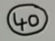
Om Sairam

Min Heap Insertion and Deletion

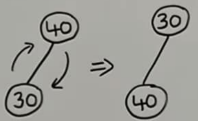
Construct a Min Heap for the following Elements

40 30 20 10 15 16 17 8 4

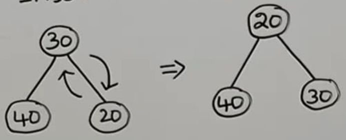
(a) Insert 40



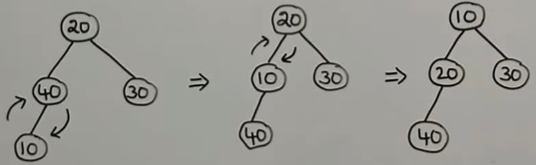
(b) Insert 30



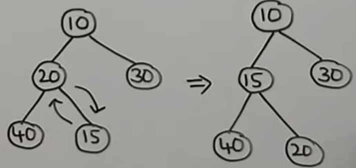
(C) Insert 20



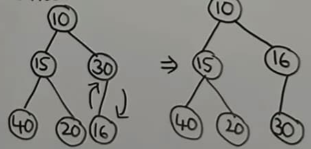
(d) Insert 10



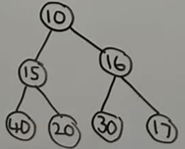
(e) Insert 15



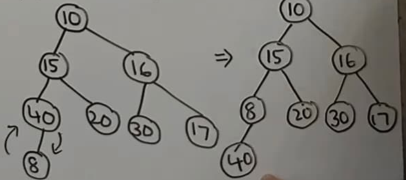
(f) Insert 16



(G) Insert 17



(H) Insert 8

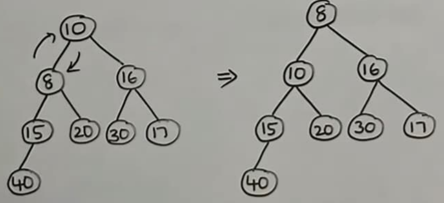


Om Sairam

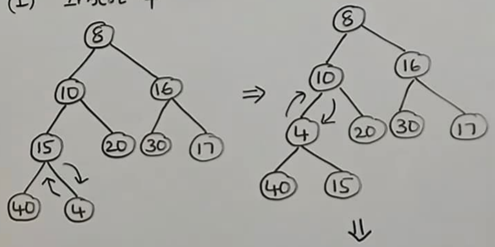
Min Heap Insertion and Deletion

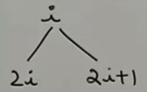
Construct a Min Heap for the following Elements

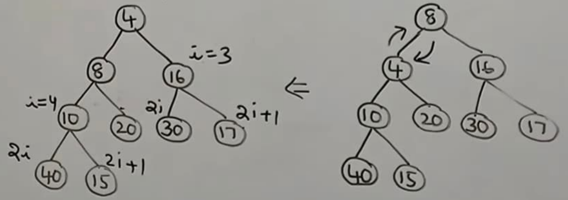
40 30 20 10 15 16 17 8 4



(I) Insert 4





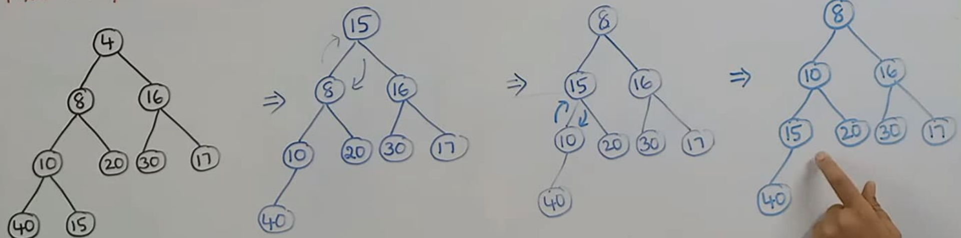


4	8	16	10	20	30	17	40	15
1	2	3	4	5	6	7	8	9

Min Heap Insertion and Deletion

Om Sairam

Min Heap Deletion :-

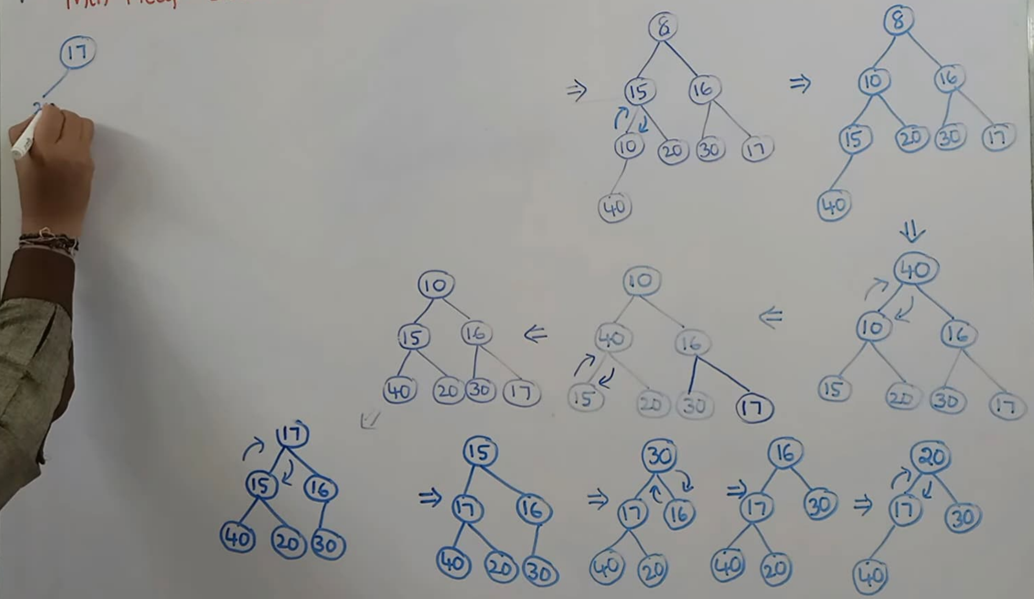


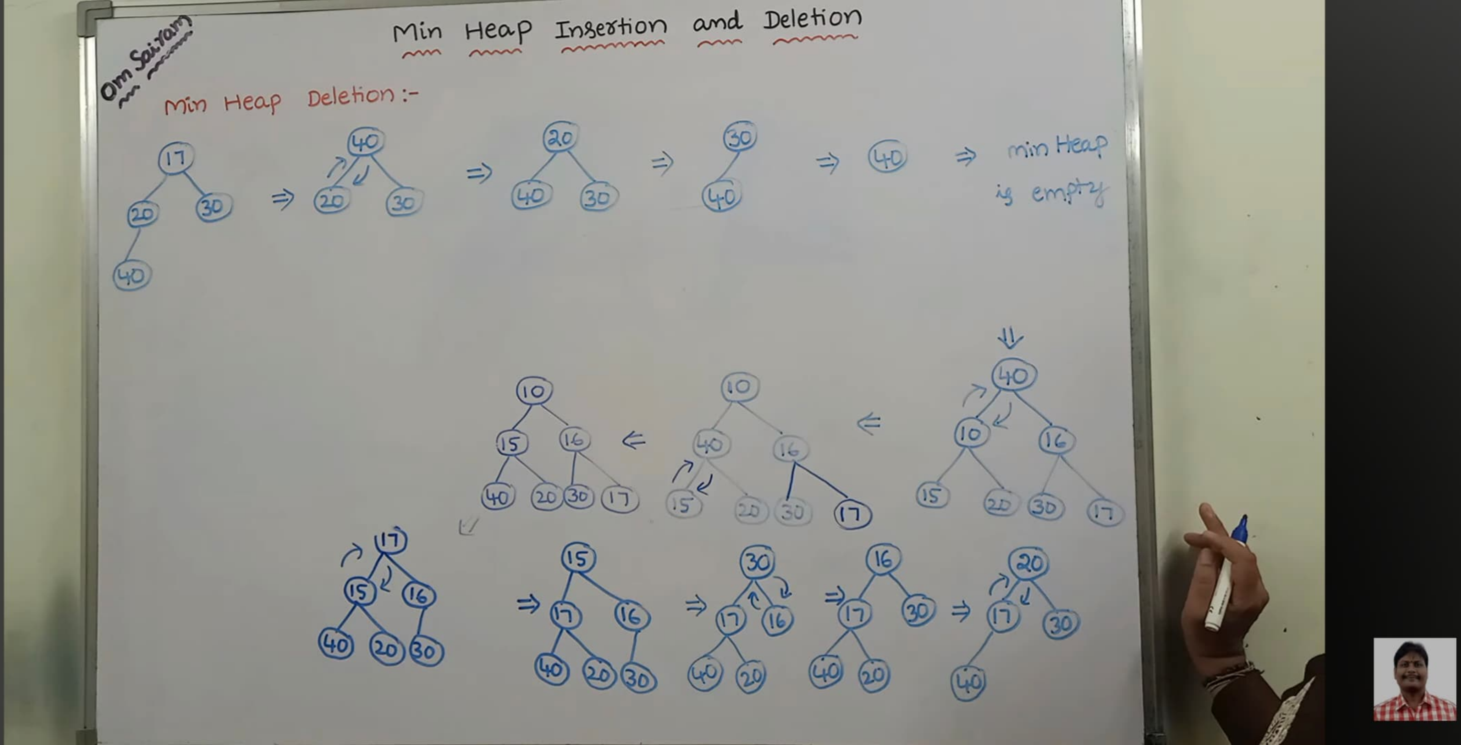
Delete root node 4 that can be replaced with right most node in the last level of Heap

Om Sairam

Min Heap Insertion and Deletion

Min Heap Deletion :-





Om Sairam

Min Heap Insertion and Deletion

Min Heap Deletion :-

min Heap is empty

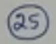
Om Sairam

Max Heap Insertion and Deletion

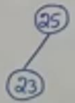
Construct Max Heap for the Following Elements

25 23 32 20 14 19 27 34 26 21 9

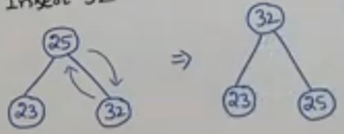
(a) Insert 25



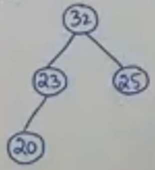
(b) Insert 23



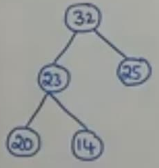
(c) Insert 32



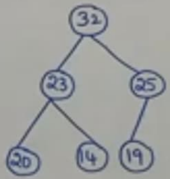
(d) Insert 20



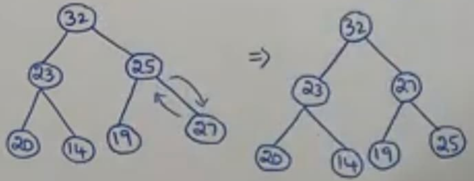
(e) Insert 14



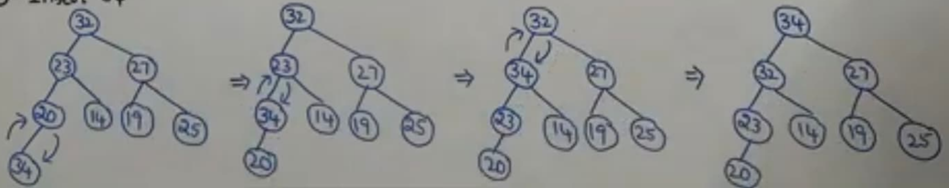
(f) Insert 19

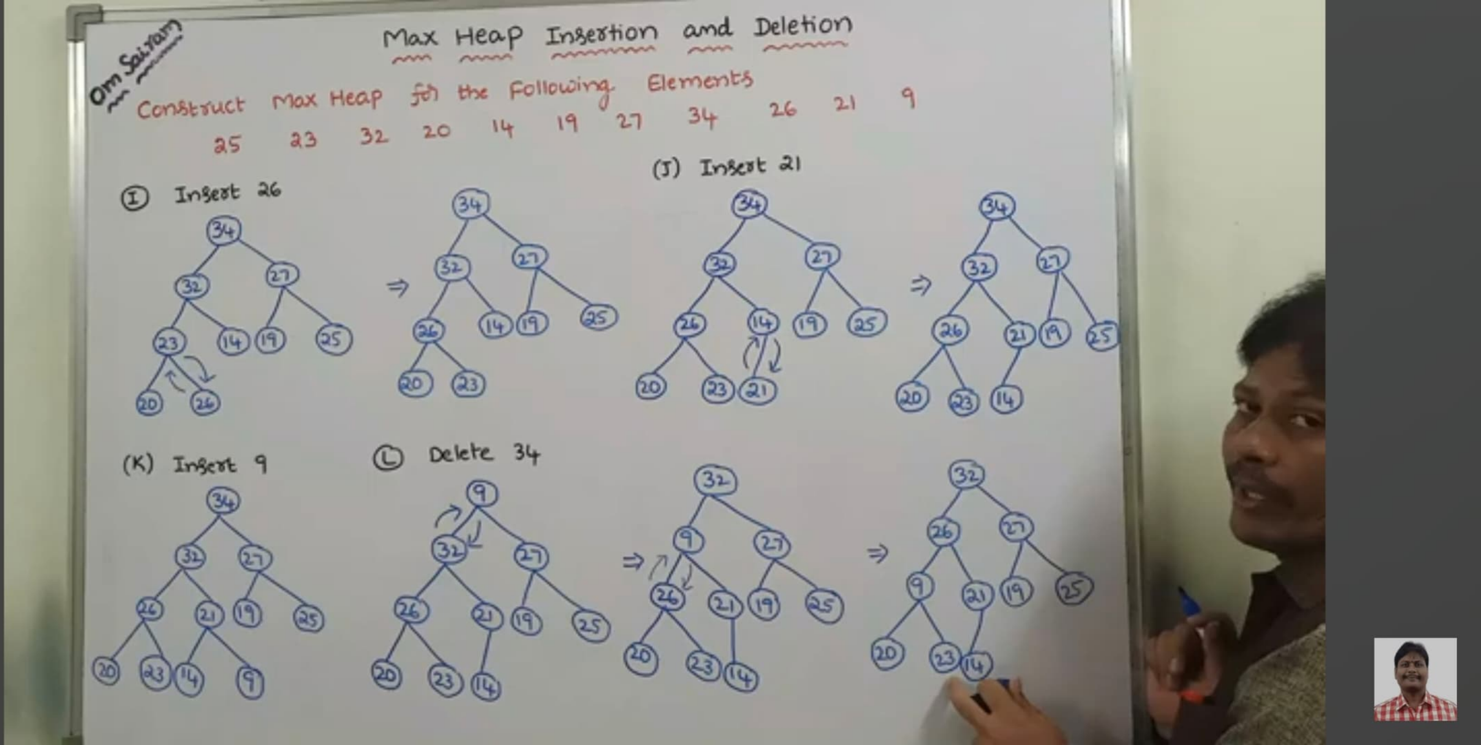


(G) Insert 27



(H) Insert 34





Om Sairam

Max Heap Insertion and Deletion

Construct Max Heap for the Following Elements

25 23 32 20 14 19 27 34 26 21 9

(I) Insert 26

(J) Insert 21

(K) Insert 9

(L) Delete 34

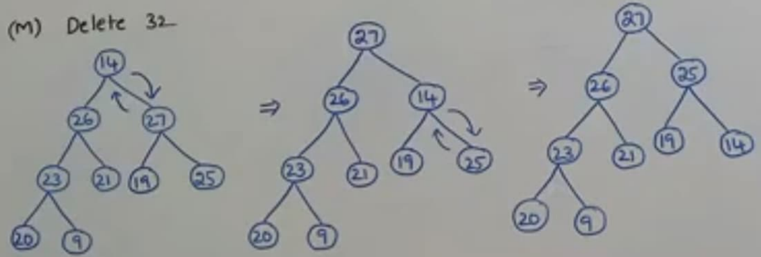
Om Sairam

Max Heap Insertion and Deletion

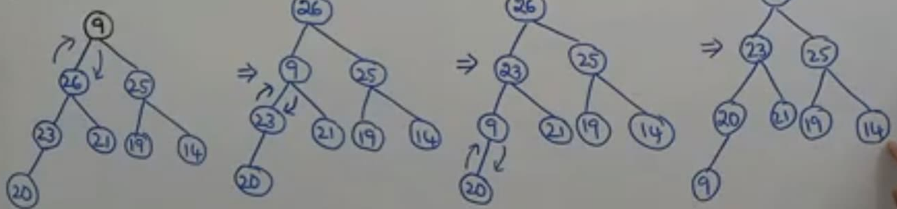
Construct Max Heap for the Following Elements

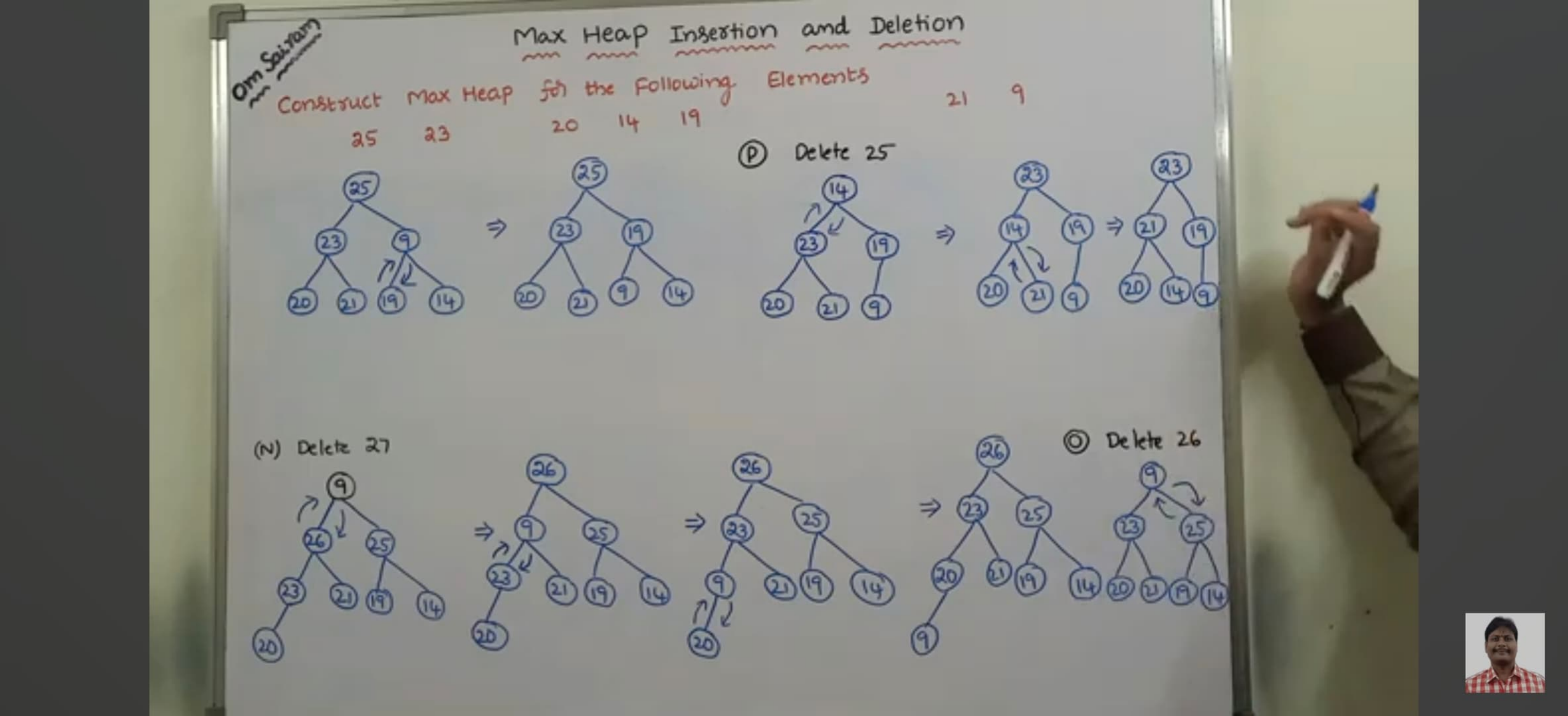
25 23 20 14 19 27 26 21 9

(M) Delete 32



(N) Delete 27





Om Sairam

Max Heap Insertion and Deletion

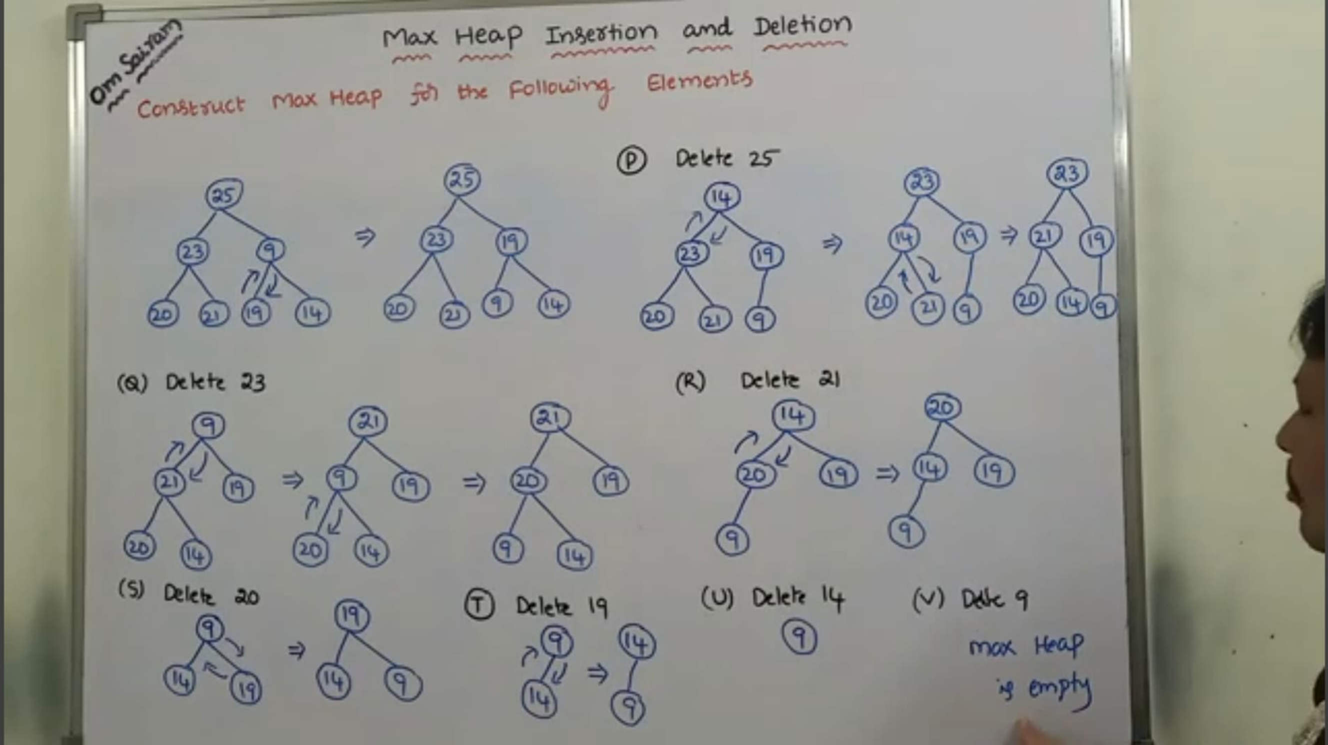
Construct Max Heap for the Following Elements

25 23 20 14 19 21 9

(P) Delete 25

(N) Delete 27

(O) Delete 26



Om Sairam

Max Heap Insertion and Deletion

Construct Max Heap for the Following Elements

(P) Delete 25

(Q) Delete 23

(R) Delete 21

(S) Delete 20

(T) Delete 19

(U) Delete 14

(V) Delte 9

max Heap is empty