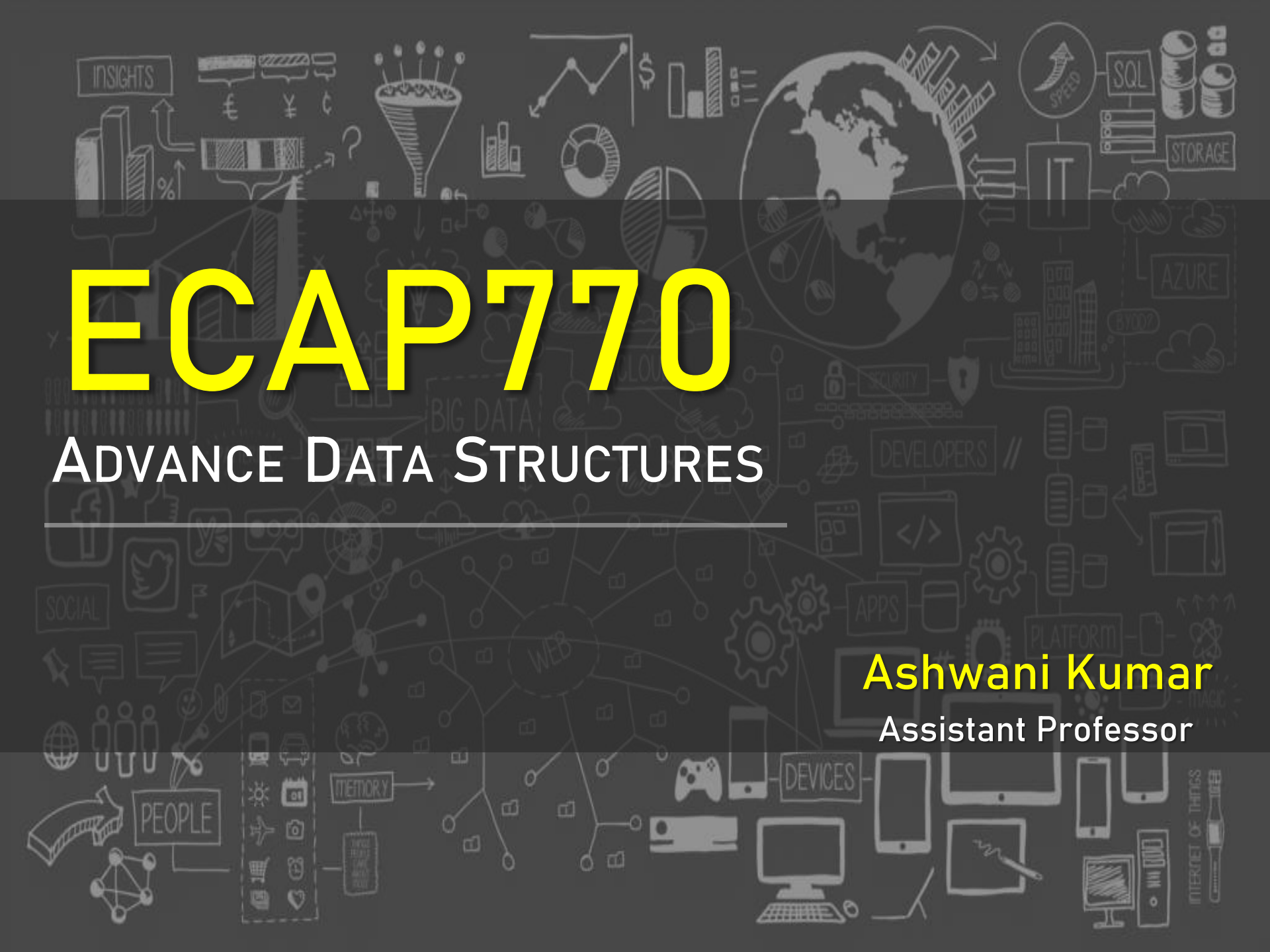


ECAP770

Advance Data Structures

Ashwani Kumar

Assistant Professor

Learning Outcomes

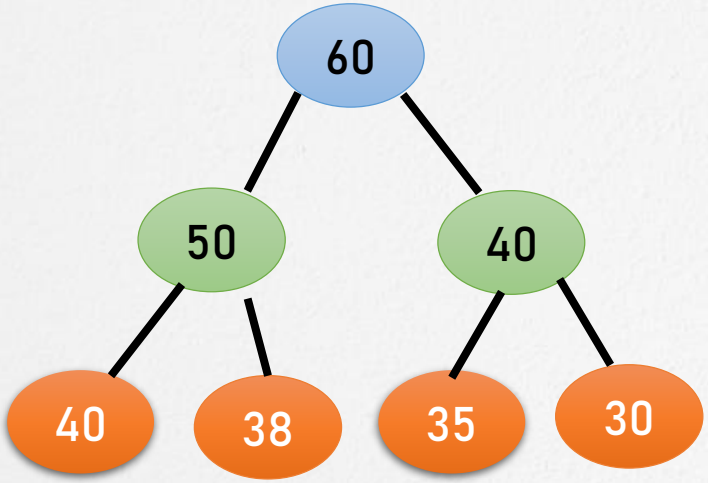
After this lecture, you will be able to

- Understand applications of heap
- Priority queue implementation

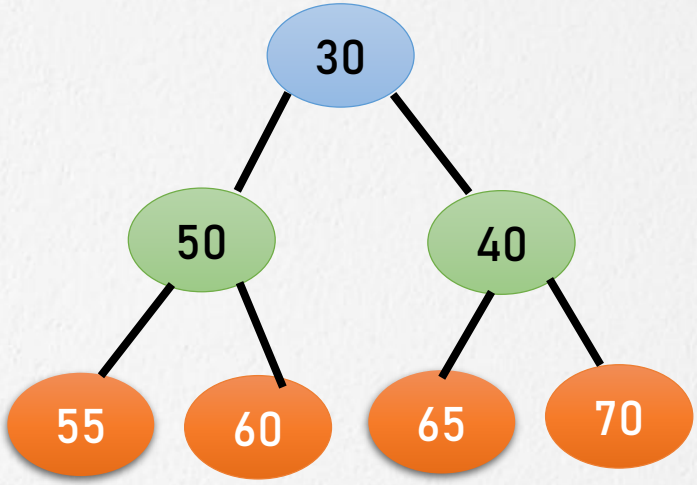
Heap

- Heap data structure is a complete binary tree that satisfies the heap property.
- Structural property
- Ordering property
- In a complete binary tree, all levels are full except the last level, i.e., nodes in all levels except the last level will have two children and all the nodes should be left-justified.

Heap Tree



Max heap



Min heap

Applications of heap

- Priority queue implementation
- Heap sort
- Order statistics

Priority queue

- In priority queue key is associated with every element.
- The element with highest priority will be moved to the front of the queue and one with lowest priority will move to the back of the queue.
- Queue returns the element according to priority.
- However, if elements with the same priority occur, they are served according to their order in the queue.

Priority queue

- A max-priority queue returns the element with maximum key first. A max-heap is used for a max-priority queue.
- A min-priority queue returns the element with the smallest key first. A min-heap is used for a min-priority queue.

Priority queue

- Ascending order priority queue: In ascending order priority queue, a lower priority number is given as a higher priority in a priority.
- Descending order priority queue: In descending order priority queue, a higher priority number is given as a higher priority in a priority.

Analysis of complexities

Implementation	add	Remove	peek
Linked list	O(1)	O(n)	O(n)
Binary heap	O(logn)	O(logn)	O(1)
Binary search tree	O(logn)	O(logn)	O(1)

Priority queue operations

- Insert
- Delete
- Peeking from the Priority Queue (Find max/min)
- Extract-Max/Min from the Priority Queue

Inserting operation: algorithm

- If there is no node,

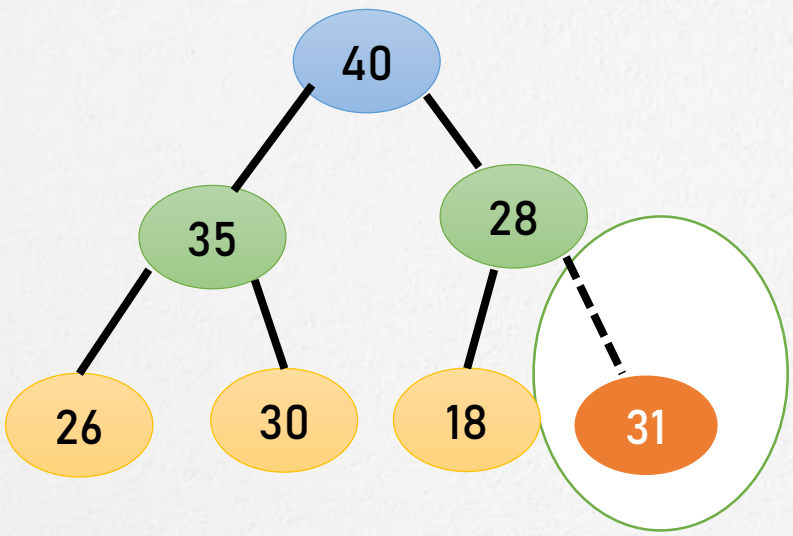
  create a newNode.

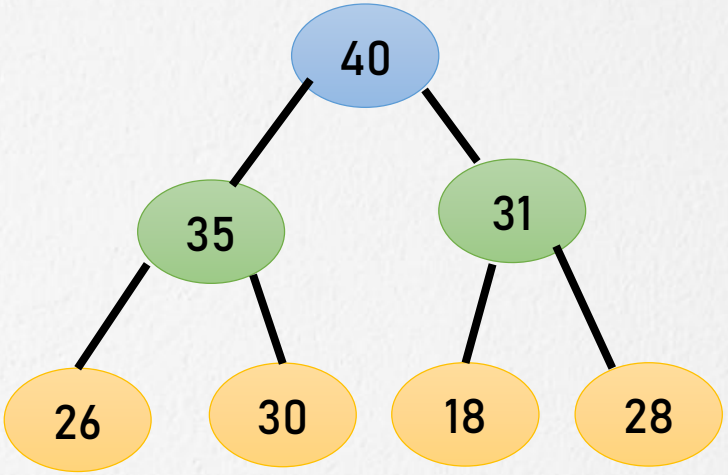
- else (a node is already present)

  insert the newNode at the end (last node from left to right.)

- Heapify the array

Inserting operation





Delete operation: Algorithm

- If nodeToBeDeleted is the leafNode

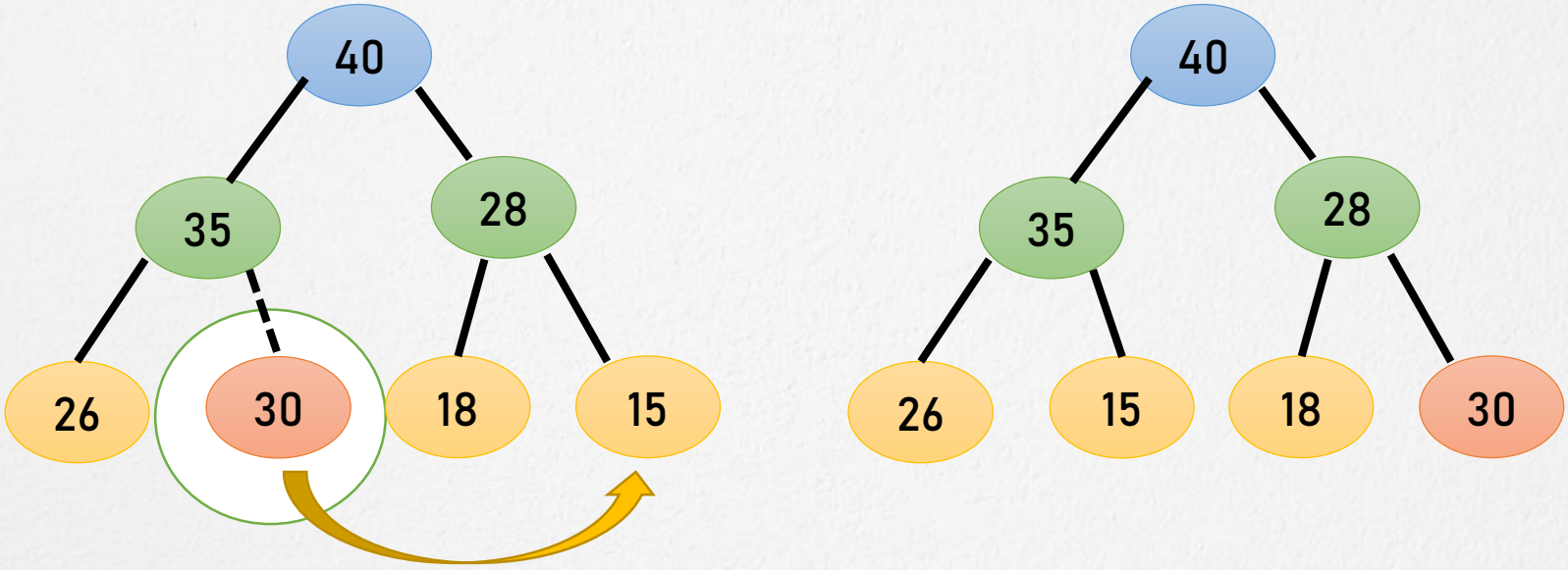
  remove the node

- Else swap nodeToBeDeleted with the lastLeafNode

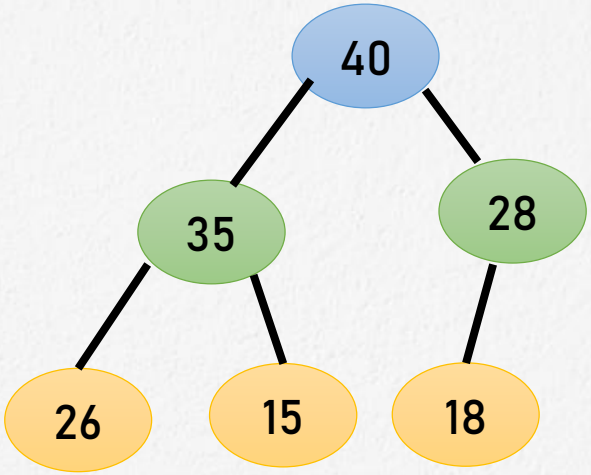
  remove noteToBeDeleted

- heapify the array

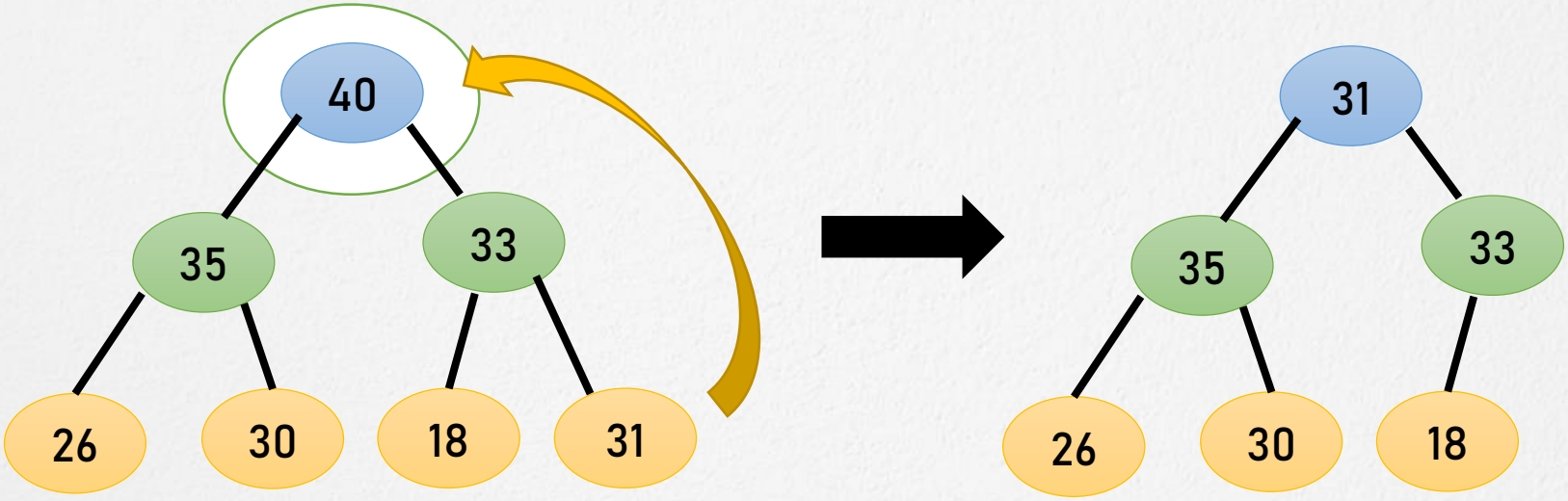
Delete operation



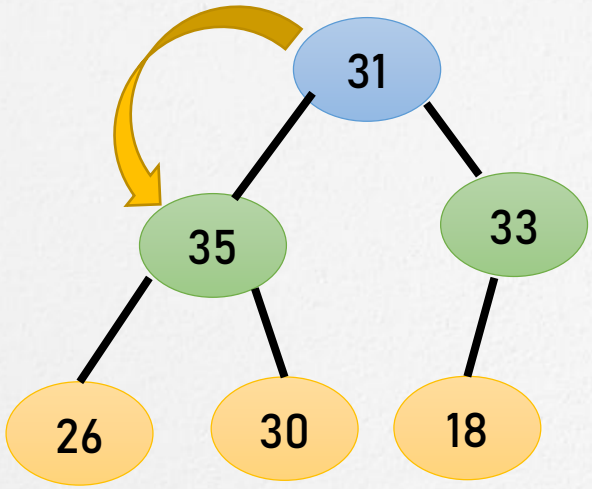
Delete operation



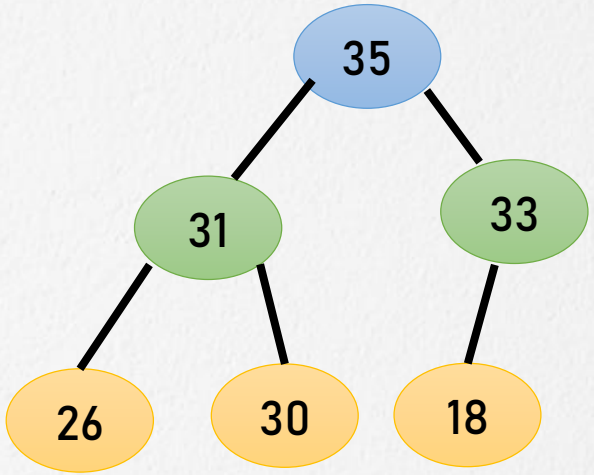
Extract Maximum (Max heap)



Extract Maximum



Heapify the tree



Applications of Priority Queue

- Dijkstra's algorithm for shortest path
- Load balancing and interrupt handling in an operating system
- Data compression in Huffman code

That's all for now...