# Heap

## Data Structures – CS2001 – Fall 2021

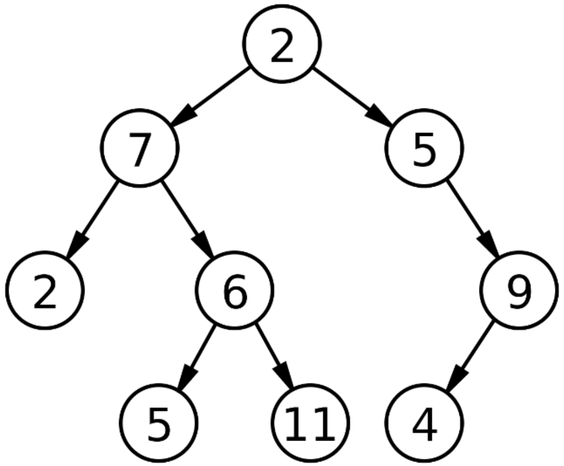
Dr. Syed Ali Raza
School of Computer Science
National University of Computing & Emerging Sciences
Karachi Campus

# Outlines

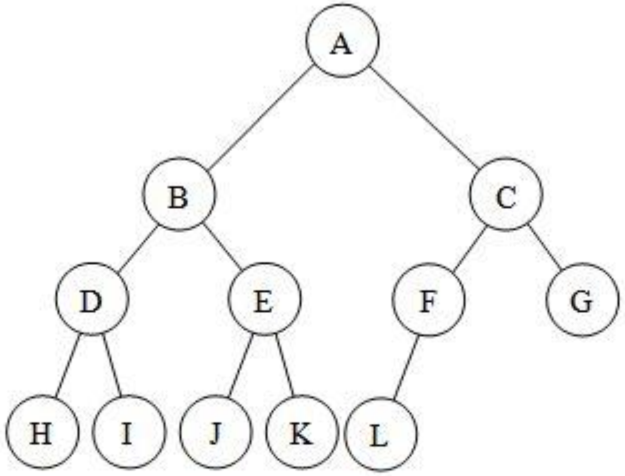
- Binary Tree
- Heap using Array
- Heap sort

# Binary Tree

- A tree data structure in which each node has at most two children.

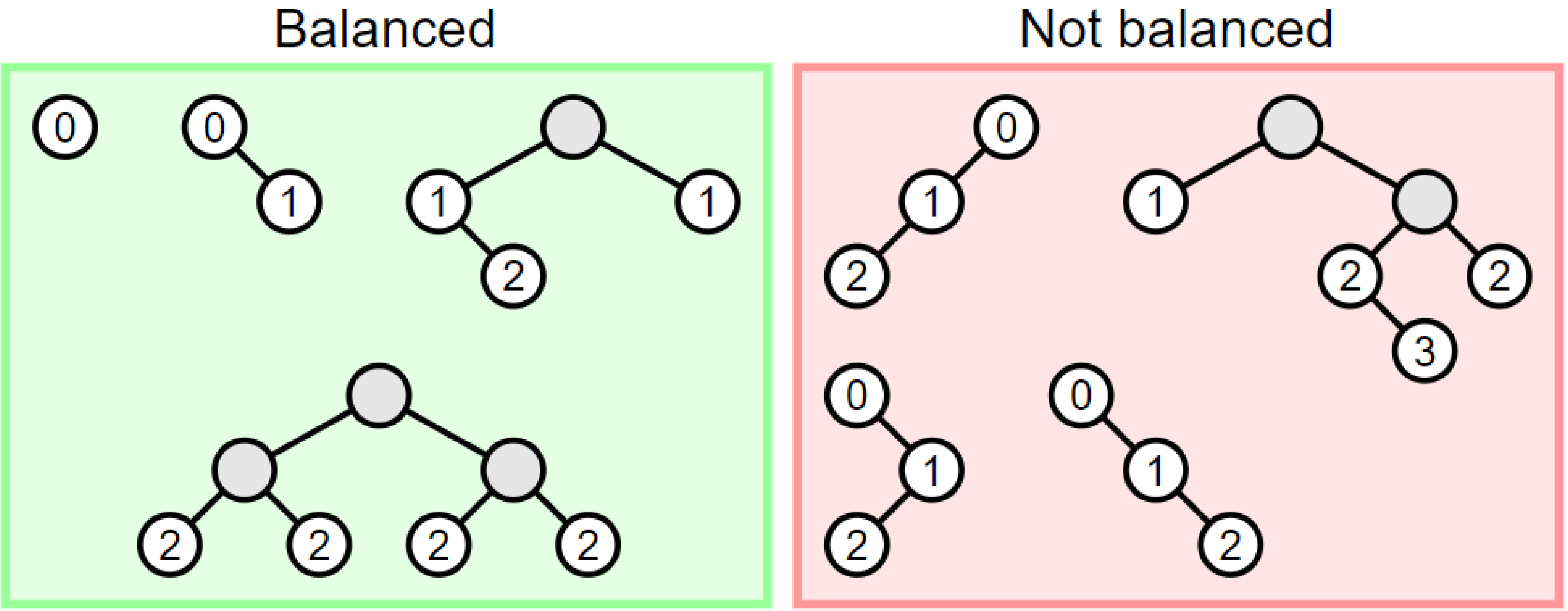



- A complete binary tree is a binary tree in which all the levels are completely filled except possibly the lowest one, which is filled from the left.



# Binary Tree

- (Height-) Balanced binary tree: a binary tree in which the left and right subtrees of every node differ in height by no more than 1.
- Every complete binary tree is balanced but not the other way around.



# Heap

- A particular kind of binary tree, called a *heap,* has the following two properties:

1. The value of each node is greater than or equal to the values stored in each of its children.
2. The tree is perfectly balanced (till the second last node), and the leaves in the last level are all in the leftmost positions (i.e., complete binary tree).

- These two properties define a *max heap.* If “greater” in the first property is replaced with “less,” then the definition specifies a *min heap.*

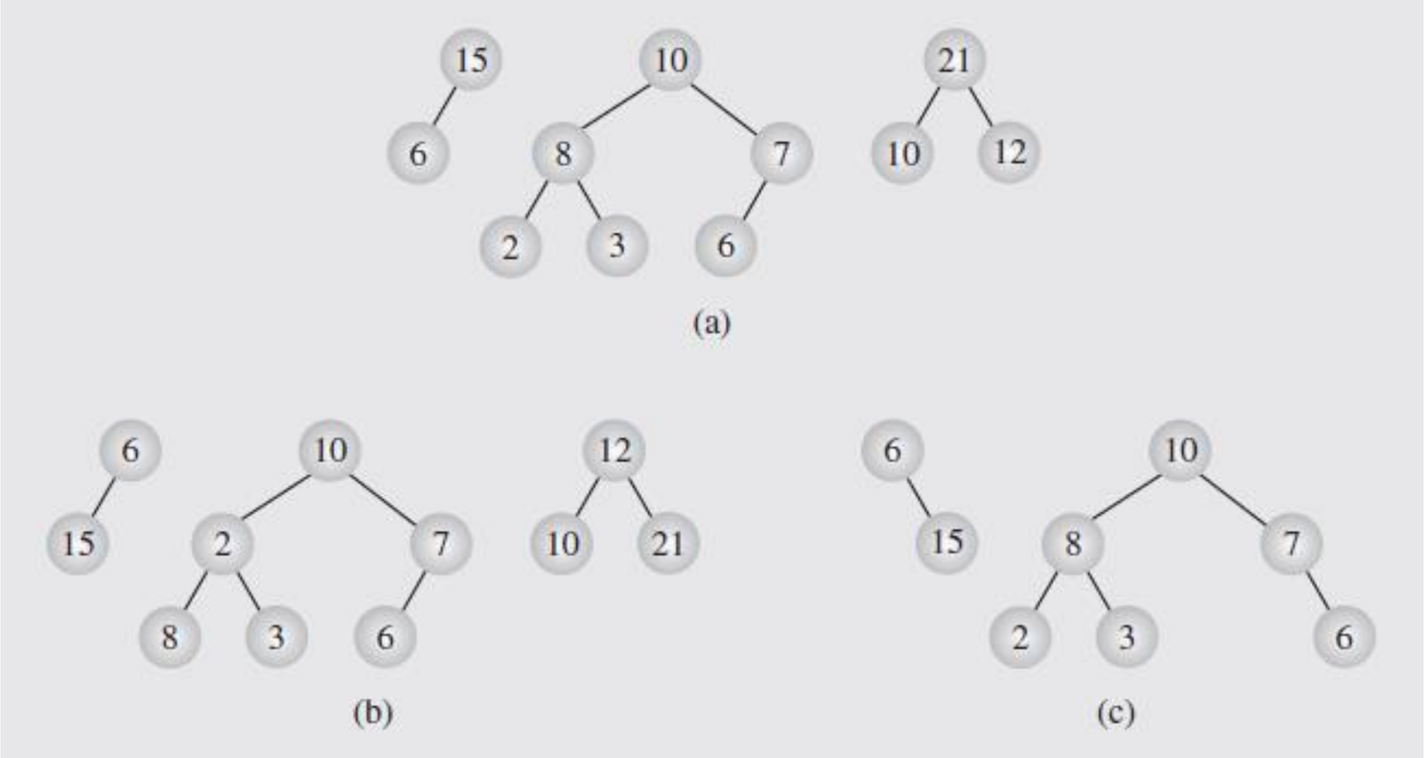
# Heap

- In heap, we keep a binary tree *complete* (to fulfil the second condition).
- It has two advantages:

1. It keeps height of binary tree shallow. Hence, it allows faster operations (E.g., insert).
   - The height of any complete binary tree with n nodes is O(log n).
   - A complete binary tree with n nodes has the minimum possible height over all binary trees with n nodes.
2. Allows to store heap as array.

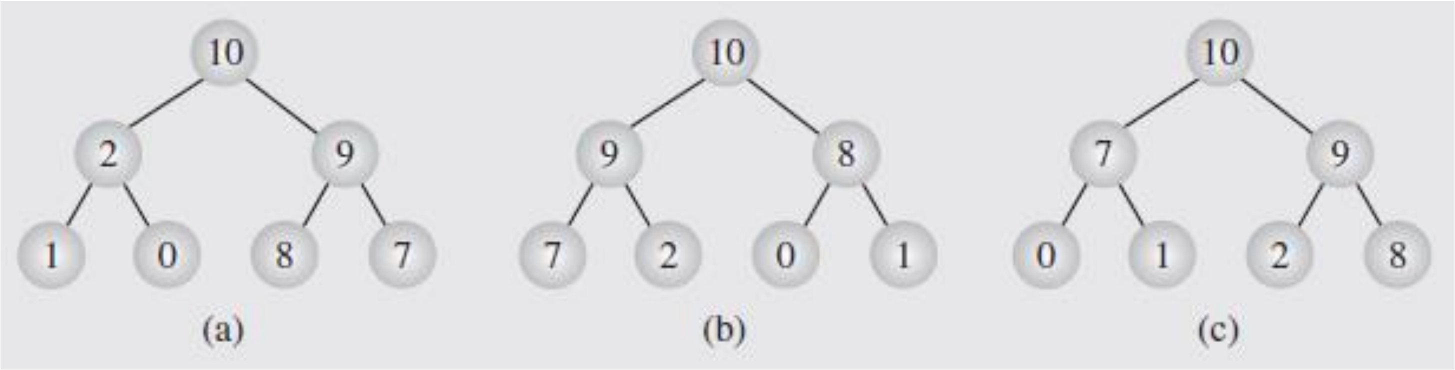
# Heap

- Examples of Heap (a) and nonheap (b and c)



# Heap

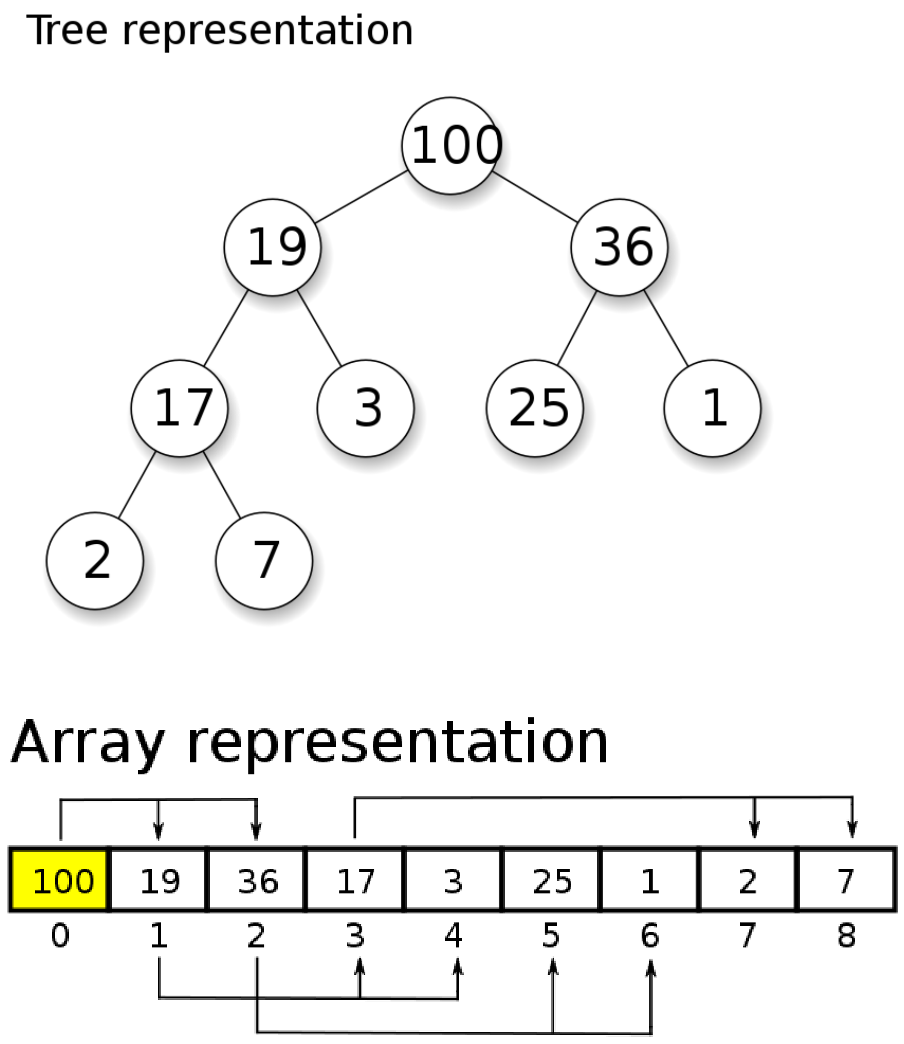
- Elements in a heap my not be perfectly ordered, but must follow the above two properties.




Different heaps constructed with the same elements.

# Heap

- Heaps can be implemented by arrays.
- The elements are placed at sequential locations representing the nodes from top to bottom and in each level from left to right.
- The root element is at zeroth index.




Example of max Heap representation as array (Wikipedia).

# Heap

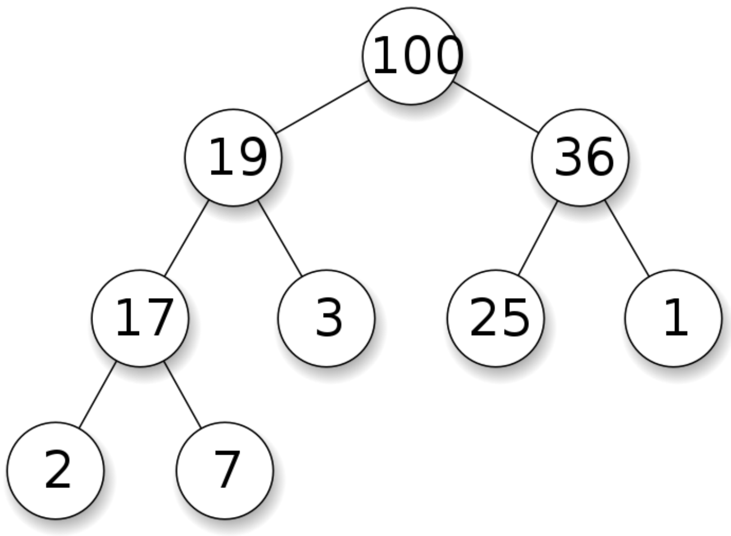
- For any other ith node, the indexes of its children and parent are given as follow,
- Left child’s index = [2*i + 1]
- Right child’s index = [2*i + 2]
- Parent’s index = [(i -1)/2]

# Max Heap

**Operations on Max Heap:**

- **getMax:** return the value at root.
- **Insert:** attach at the left most vacant position in the last level and call sift up.
- ***Sift up:*** swap larger node with its parent until the second property is satisfied. Note, this property is violated at a time on at most one edge.
- Time complexity is O(tree height).
- Practice: insert 2, 8, 6, 1, 10, 15, 3 in an empty Max heap.

Tree representation



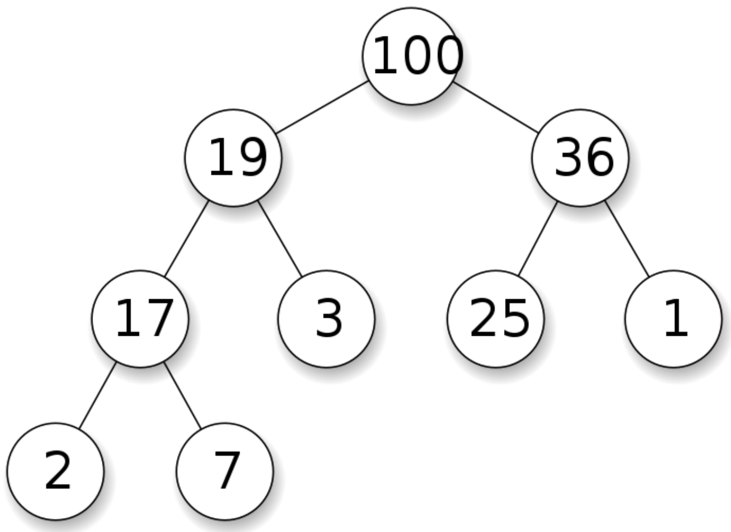

Insert 95 as left child of 3

# Max Heap

**Operations on Heap:**

- **extractMax:** replace the root with the last leaf and call sift down.
- ***Sift down:*** swap smaller node with its larger child until the second property is satisfied. Note, this property is violated at a time on at most one edge.
- Time complexity is O(tree height) or O(logn).

Tree representation

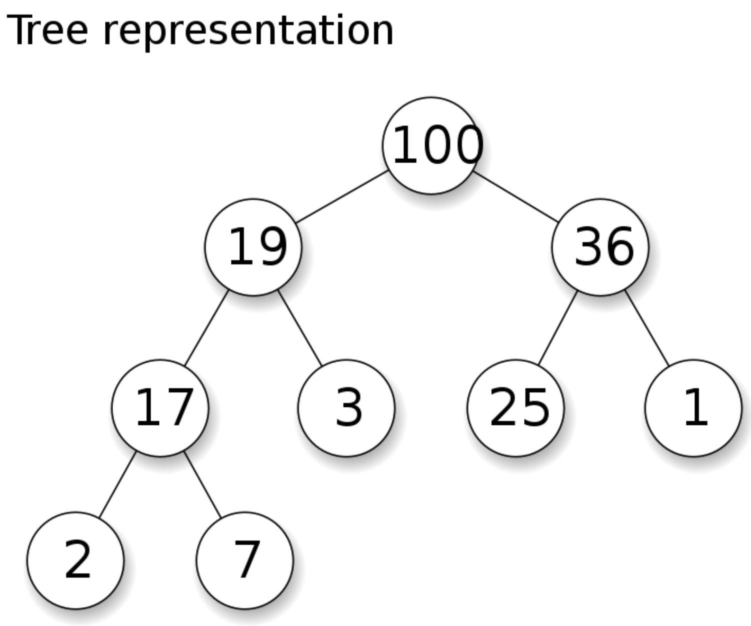



Replace 100 with 7, extract 100, and sift down 7.

# Max Heap

**Operations on Heap:**

- **Remove:** change the value of the element to a very high number (say, positive infinity), then call sift up, and then call extractMax.
- Time complexity is O(tree height) or O(logn).

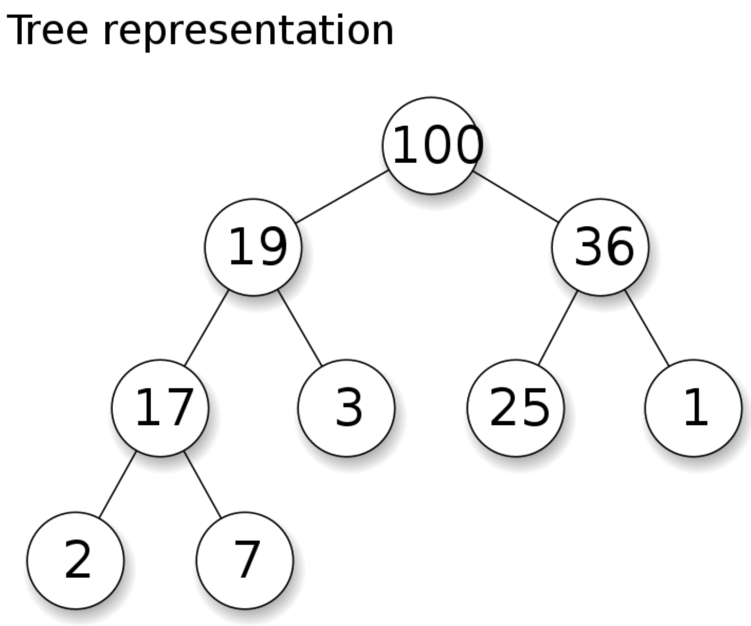



Remove node with priority 17

# Max Heap

**Operations on Heap:**

- **ChangePriority:** change the priority and if the priority is increased then call sift up, otherwise, if the priority is decreased then call sift down.
- Time complexity is O(tree height) or O(logn).




Change priority of node 7 to 25.

# Heap Applications

- They essentially implement a priority queue.
- Implement heap sort.
- Used as internal traversal data structure in graph based algorithms like Dijkstra's shortest-path algorithm.

# Heap Sort

- It attempts to improve selection sort.
- Recall that selection sort is efficient in terms of swaps (maximum n) but inefficient in terms of comparisons (it makes n^2 comparisons to select the small element in each iteration).
- In Heap sort, the selection part is replaced with a heap data structure.
- We use max heap to select (extact) maximum element and place it at the end of the heap (array).
- We can use extractMax operation to get and extract the maximum element existing in the heap which creates an empty space at the end of the array.

# Heap Sort

- The idea is to iteratively extract the max and place it at before the already sorted elements.
- Heap sort can have both in-place and out-of-place implementations.
- In the in-place implementation, we use the same array which is given as input to sort (i.e., no additional space is used).
- In the out-of-place implementation, we use an additional space (array) to sort.
- Heap sort is not a stable sort algorithm; the order of equal elements is not preserved in the sorted output.

# Heap Sort

- Implementation of Heap sort has two parts:

1. Building heap: given an array, build a heap.
2. Sorting using heap: extractMax and place at the end.

- In Build heap, we start from the last internal node and go through all the nodes till the root node. For each node, we call sift_down. This way we rearrange the elements in the given array to form a heap.
- The last element of a binary heap is given by [(n – 2) / 2] where n is the number of elements in heap.

# Heap Sort

- Time complexity of Build heap looks like O(nlogn) because we do sift down on n node.
- However, it turns out that it completes in O(n).
- The intuition is that for the nodes in the second last layer the tree height is 1 so maximum 1 swaps will occur. Likewise, for the nodes in the third last level, maximum two swaps occur.
- In fact, only for the roor node, we have Logn number of sift-down operations.
- Also, in the second last node we have n/4 nodes, so most of the nodes need few sift-down operations to maintain heap property.

# Heap Sort

- Even if the build heap is done in O(n), the sort part takes O(nlogn).
- In the sort part, we do extractMax which is O(logn) n times.
- Finally, Heap sort is a comparison based sorting which completes in worst time of O(nlogn) and takes no additional space.