HEAP SORT ALGORITHM

I/ DEFINITION:

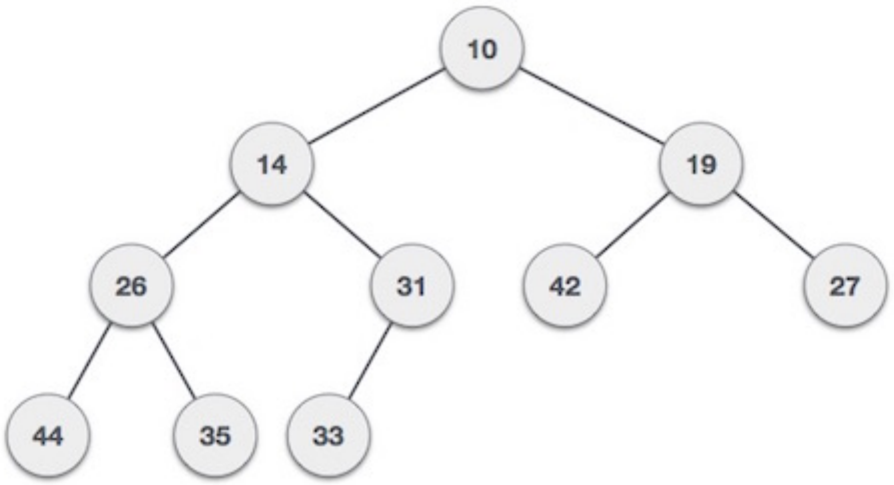
- **Heap** is a special case of balanced binary tree data structure where the root-node key is compared with its children and arranged accordingly. If **α** has child node **β** then –

$$key(\alpha) \geq key(\beta)$$

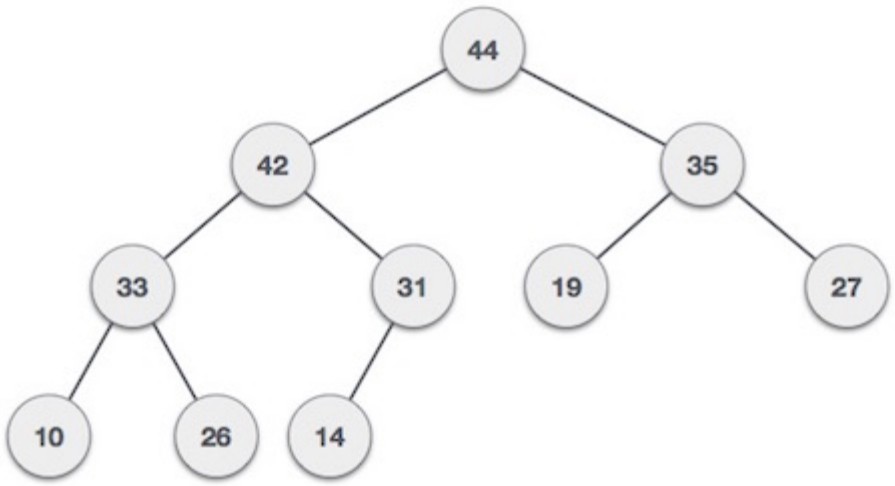
- As the value of parent is greater than that of child, this property generates **Max Heap**. Based on this criteria, a heap can be of two types –

```
For Input → 35 33 42 10 14 19 27 44 26 31
```

- **Min-Heap** – Where the value of the root node is less than or equal to either of its children.



- **Max-Heap** – Where the value of the root node is greater than or equal to either of its children.



Both trees are constructed using the same input and order of arrival.

==

II/ MAX HEAP DELETION ALOGORITHM:

- Let us derive an algorithm to delete from max heap. Deletion in Max (or Min).Heap always happens at the root to remove at the Maximum (or Minimum) value.

Step 1: Remove root mode

Step 2: Move the last element of last level to root

Step 3: Compare the value of this node with its parent.

Step 4: If value of parent is less than child, then swap them.

Step 5: Repeat step 3&4 until Heap property holds.

```
Heapsort(A as array)
   BuildHeap(A)
   for i = n to 1
        swap(A[1], A[i])
        n = n - 1
        Heapify(A, 1)

BuildHeap(A as array)
    n = elements_in(A)
    for i = floor(n/2) to 1
        Heapify(A,i,n)

Heapify(A as array, i as int, n as int)
    left = 2i
    right = 2i + 1

    if (left ≤ n) and (A[left > A[i])
        max = left
    else
        max = i

    if (right ≤ n) and (A[right > A[i])
        max = right

    if (max != i)
```

```
        swap(A[i], A[max])
        Heapify(A, max)
```