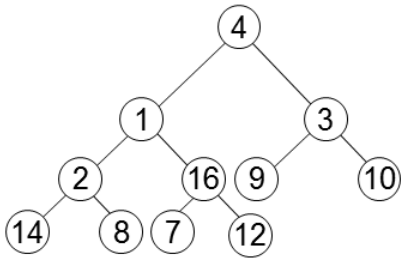
Heap Tree

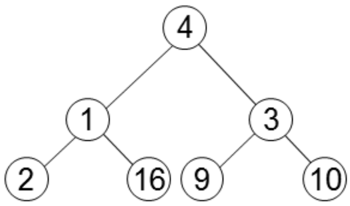
Special Types of Trees

- *Def:* Full binary tree = a binary tree in which each node is either a leaf or has degree exactly 2.



Full binary tree

- *Def:* Complete binary tree = a binary tree in which all leaves are on the same level and all internal nodes have degree 2.

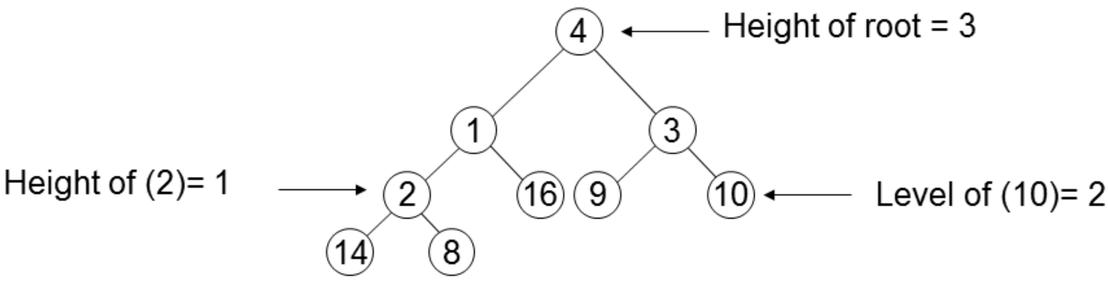


Complete binary tree

O(n) – How? Each node: *O(1)*, *n* nodes: *O(n)*

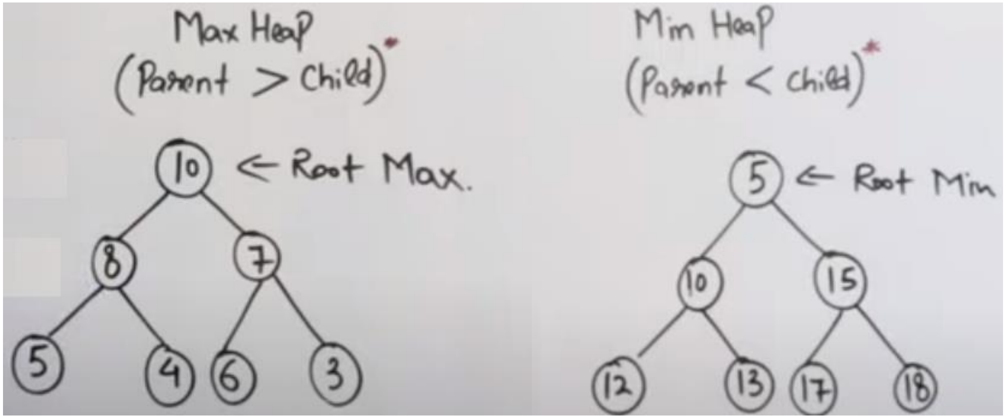
Definitions

- Height of a node = the number of edges on the longest simple path from the node down to a leaf
- Level of a node = the length of a path from the root to the node
- Height of tree = height of root node



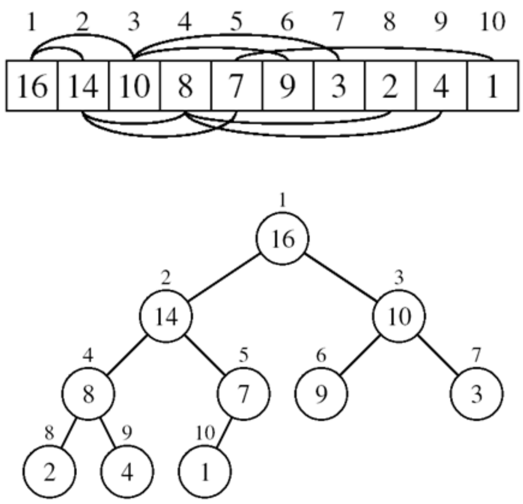
Must follow

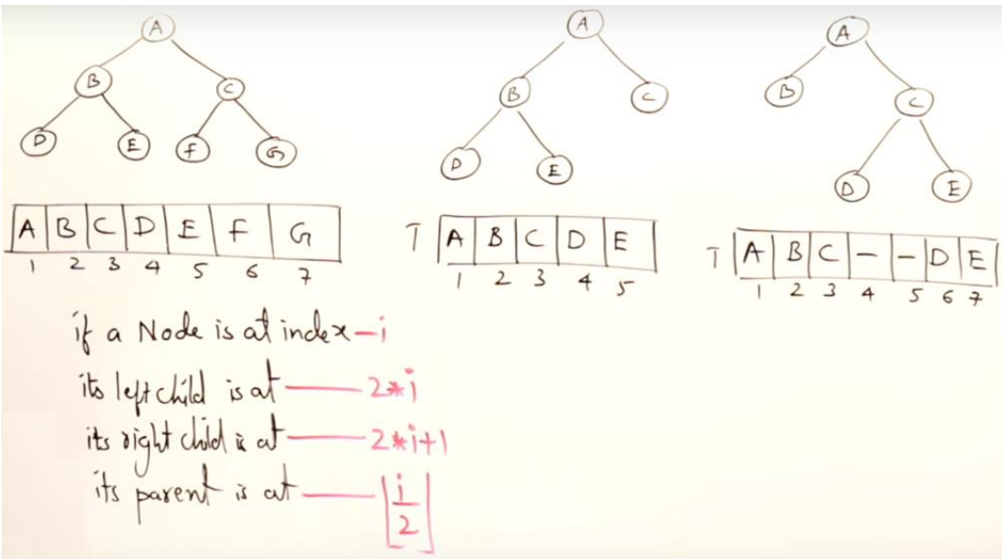
- Structural Property: It must be almost complete binary tree.
- Ordering Property: Max Heap or Min Heap



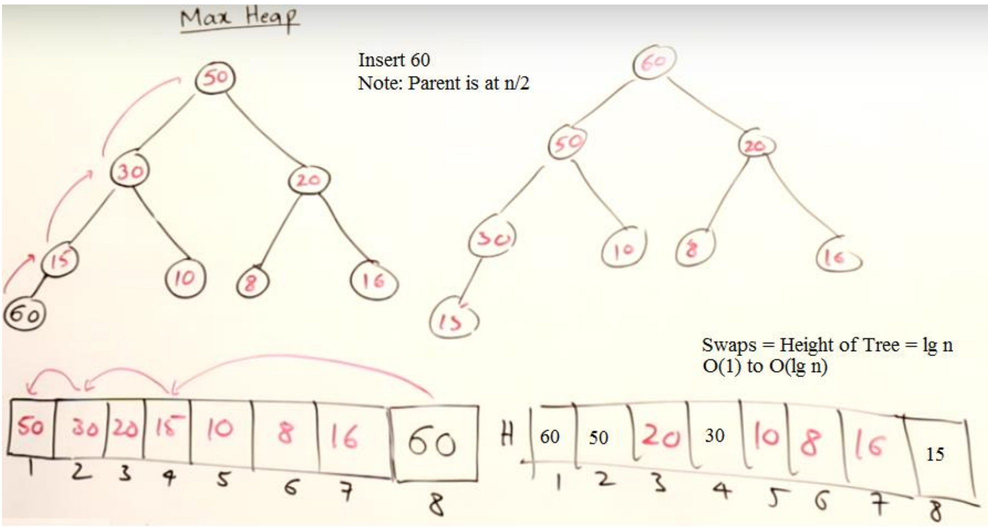
Array Representation of Heaps

- A heap can be stored as an array *A*.
  - Root of tree is A[1]
  - Left child of A[i] = A[2i]
  - Right child of A[i] = A[2i+1]
  - Parent of A[i] = A[⌊i/2⌋]
  - Heapsize[A] ≤ length[A]
- The elements in the subarray A[(⌊n/2⌋+1) .. n] are leaves





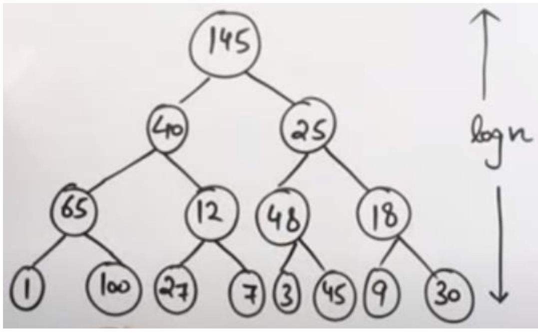
Insertion: New nodes are always inserted at the bottom level (left to right).



Insert Key one by one in the given order

- Insert an element in an empty tree: *O(1)*
- Insert an element in an existing tree. Worst case: Height of the binary tree **(HOW?)**. The complexity function is 2log n, So, *O(log n)*
- Total elements: n, So, *O(n log n)*

Heapify: 145, 40,25, 65, 12, 48, 18, 1, 100, 27, 7, 3, 45, 9, 30



Solution?

Total Swaps = S

$$S = \frac{n}{2^0} * 0 + \frac{n}{2^1} * 1 + \frac{n}{2^2} * 2 + \frac{n}{2^3} * 3 \ldots\ldots + \frac{n}{2^{\log n}} * \log n$$

$$S = n \left[\frac{1}{2} + \frac{2}{2^2} + \frac{3}{2^3} + \frac{4}{2^4} + \ldots + \frac{\log n}{2^{\log n}} \right] \ldots\ldots (1)$$

Multiply eqn 1 by 1/2

$$\frac{S}{2} = n \left[\frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \frac{4}{2^5} + \ldots + \frac{\log n - 1}{2^{\log n}} + \frac{\log n}{2^{\log n + 1}} \right] \ldots (2)$$

Multiply eqn 1 by 1/2

$$\frac{S}{2} = n\left[\frac{1}{2^2}+\frac{2}{2^3}+\frac{3}{2^4}+\frac{4}{2^5}+\dots+\frac{\log n-1}{2^{\log n}}+\frac{\log n}{2^{\log n+1}}\right] \dots (2)$$

Subtract (2) from (1)

$$\frac{S}{2} = n\left[\left[\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+\frac{1}{2^4}+\dots\frac{1}{2^{\log n}}\right]-\frac{\log n}{2^{\log n+1}}\right]$$

Use GP formula $r<1$

$$\frac{S}{2} = n\left[\left(\frac{\frac{1}{2}\cdot\left(1-\frac{1}{2^{\log n}}\right)}{1/2}\right)-\frac{\log n}{2^{\log n+1}}\right] \qquad \boxed{S=\frac{a\cdot(1-r^n)}{1-r}}$$

Complexity: *O(n)*

Delete Node:

- Best Case, Rightmost at the lowest level: *O(1)*
- Worst Case, Root is deleted: *O(log n)*, For n elements: *O(n log n)*

Heapsort: *O(n) + O(n log n)* **means** *O(n log n)*

- Construct Min-Heap: *O(n)*
- Remove root again and again till the tree is empty: *O(n log n)*
- Example: 4, 6, 10, 9, 2