Q. Given length of N ropes

[2, 5, 2, 6, 3]

Cost of connecting any 2 ropes = sum of length of both the ropes

Find the minimum cost of connecting all the ropes

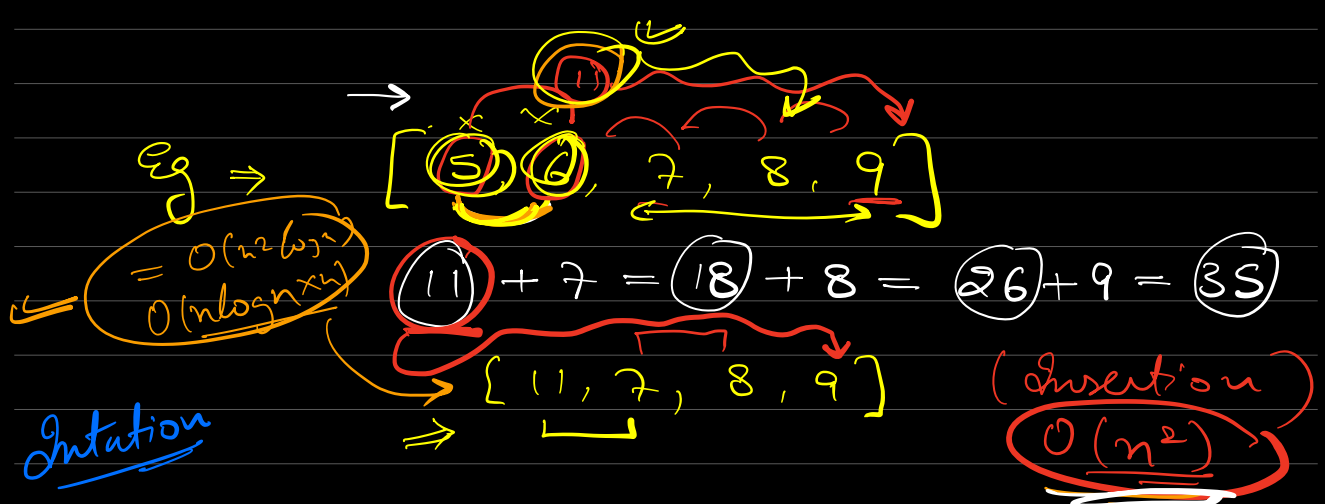
C1 C2 C3 C4

1) 2+5 = 7 + 2 = 9 + 6 = 15 + 3 = 18

7 + 9 + 15 + 18 = 49

2) 2+2 = 4 + 3 = 7 5 + 6 = 11 = 18

4 + 7 + 11 + 18 = 40



⇒ x < y < z

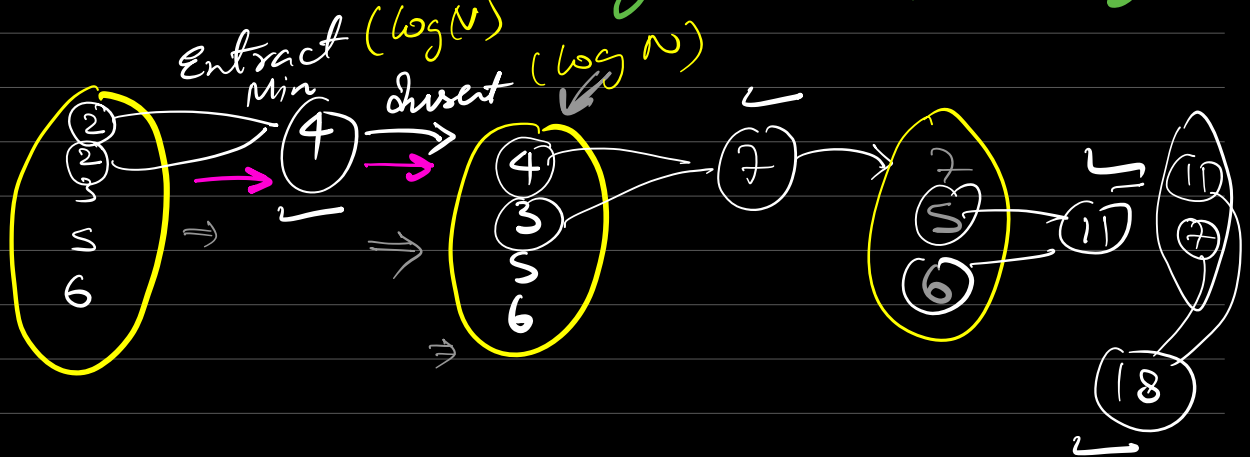
x min

C1 (1) x+y y+z x+z

C2 (2) (x+y)+z (y+z)+x (x+z)+y

⇒ At any given time, my goal is to select the 2 ropes with min length.

Q What if we had a DS where we can insert & getMin optimally



T.C. = (2 Extract + One insert)

$N($ 2 Extract + One insert $)$

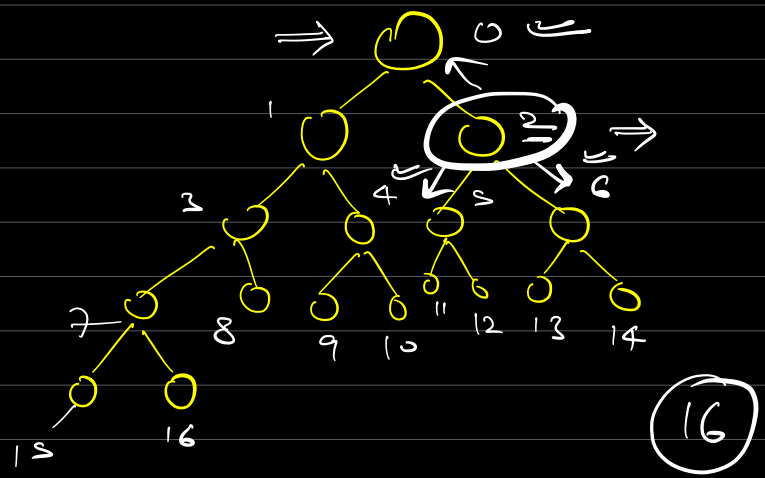
$N($ 2 log N + log N $)$

$= O(N \log N)$

Binary Heap

Properties : 1) Structure is Complete Binary

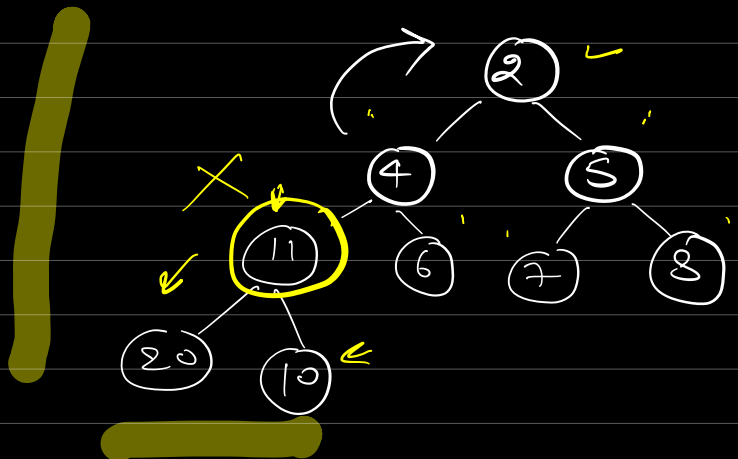
Tree.



$\lfloor (i-1)/2 \rfloor$

2) Types → Max Heap
→ Min Heap

∀ nodes, node.data ≤ node.left.data
node.data ≤ node.right.data



Is this a min Heap ⇒ NO (10 > 11)

$i^0 \Rightarrow$ LC = 2 * i + 1

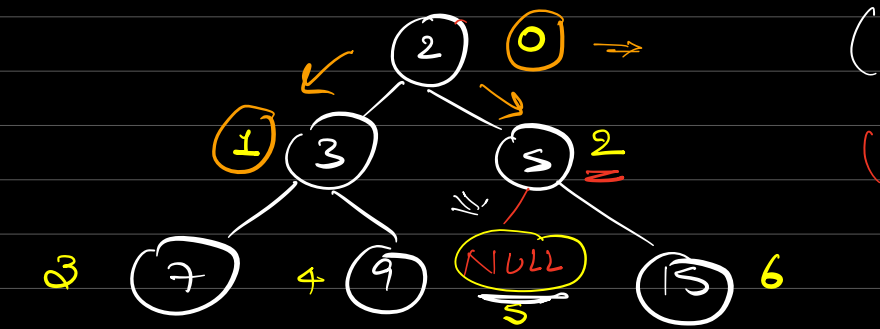
RC = 2 * i + 2

Parent ⇒ ⌊(i−1)/2⌋

Max Heap

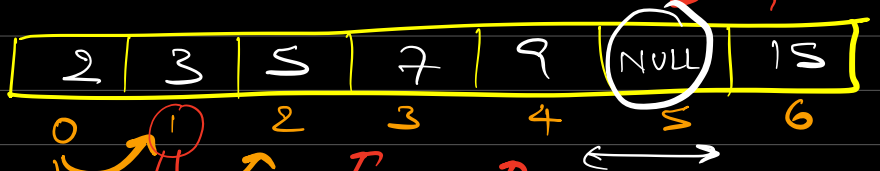
node.data ≥ node.left.data

node.data ≥ node.right.data



(6)

(2×2 + 0) = 6 ✓



(7)

0 Based indexing

i ⇒ LC (2 * i + 1)

RC (2 * i + 2)

Parent ⌊(i−1)/2⌋

1 Based

i ⇒ LC = 2i

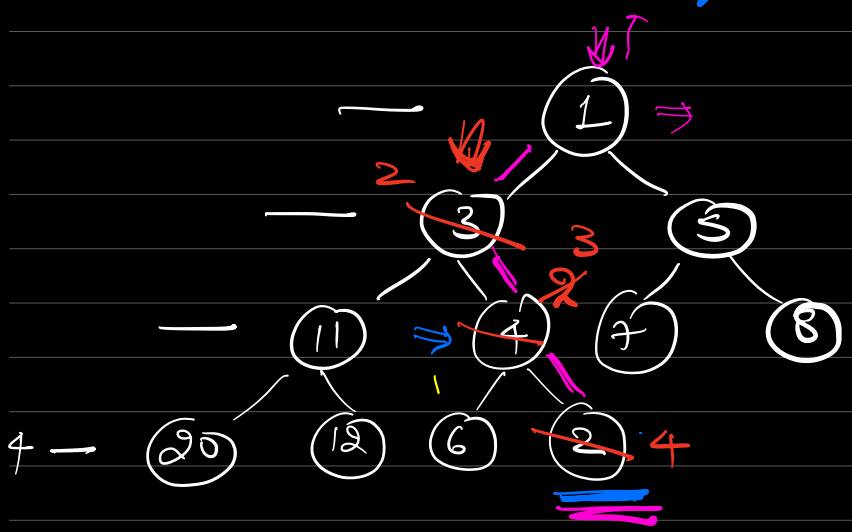
RC = 2i + 1

Parent = ⌊i/2⌋

① Insertion

NOTE

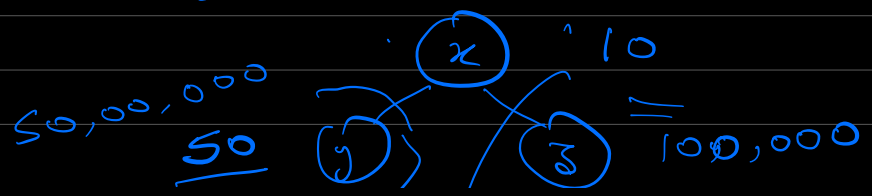
⇒ Whenever you perform any operation on the heap.
We need to make sure after completing the operation, the 2 properties of a heap are still being followed

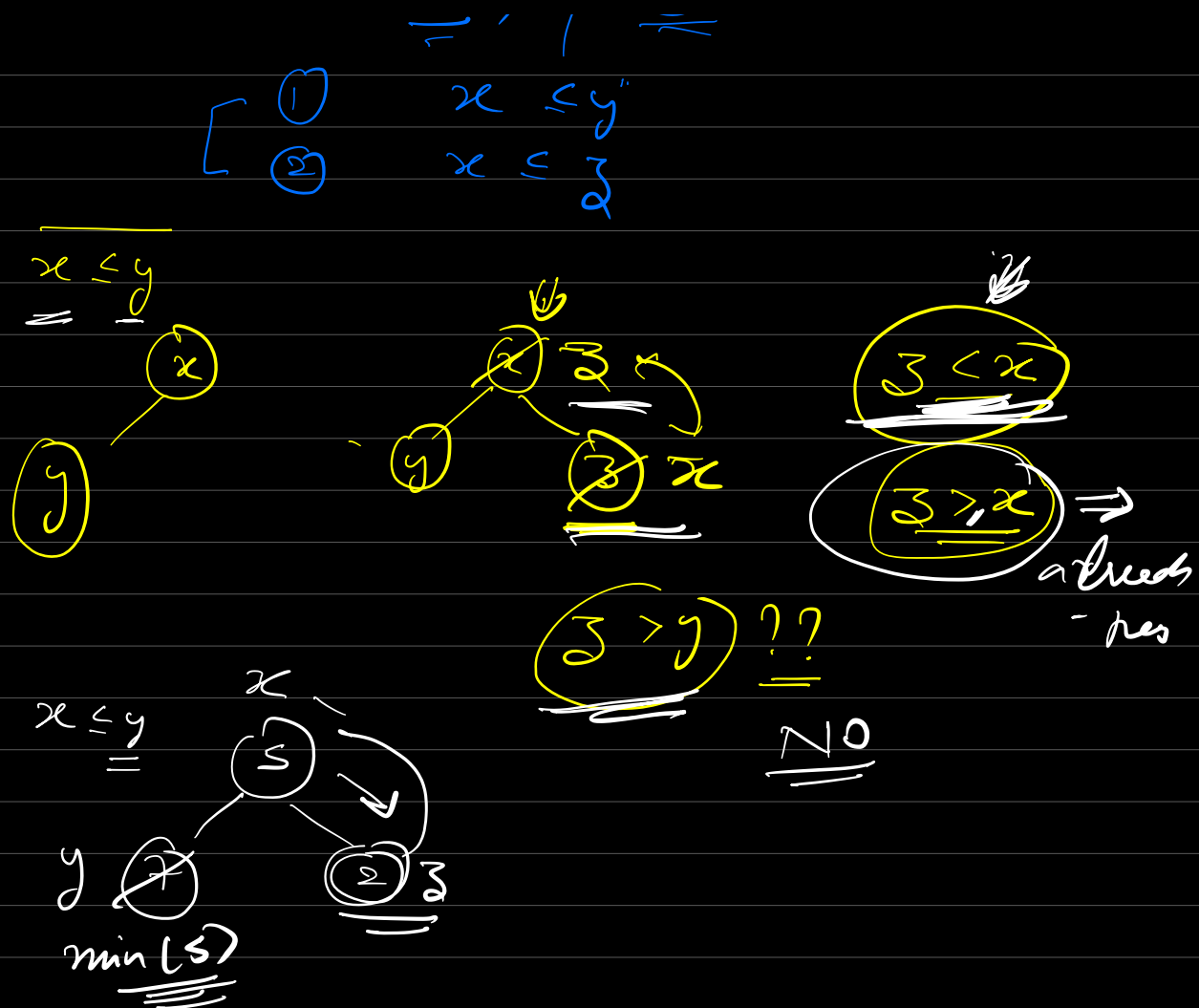


Heapify

⇒ Maintain the property of heap after insertion / deletion

Prop³ No relation b/w the left & the right subtree of each node





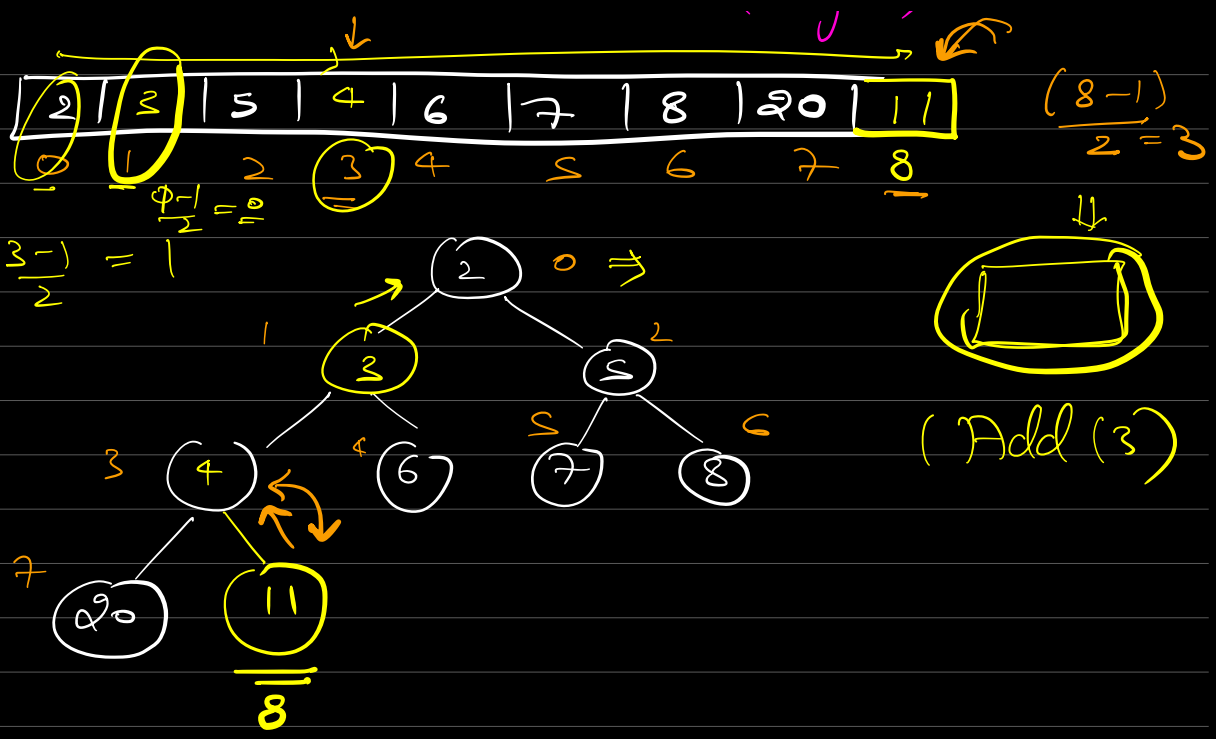
Steps to Insert

1) Insert at the last level in the left most empty position

2) Up Heapify

↳ if (parent.data > curr.data) {
swap (parent.data, curr.data);
}

T.C = O(H) = O(log N)



INIT arr[]

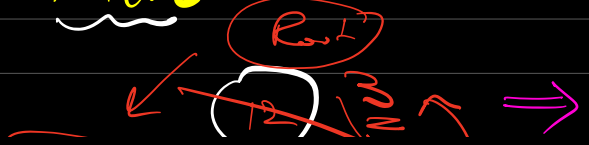
Node arr[]

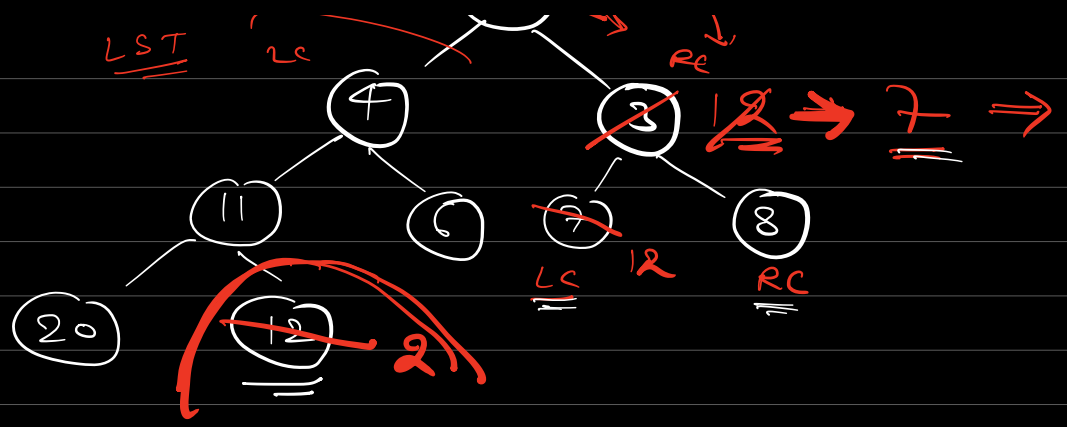
UP-HEAPIFY

curr ⇒ parent = ⌊(curr-1)/2⌋

```
while ( curr != 0 && (parent.value > curr.value) {
    swap ( curr, parent) ;
    curr = parent;
}
```

2) Delete Min





⇒ Element can be deleted only from the end of the array
(To maintain CBT property)

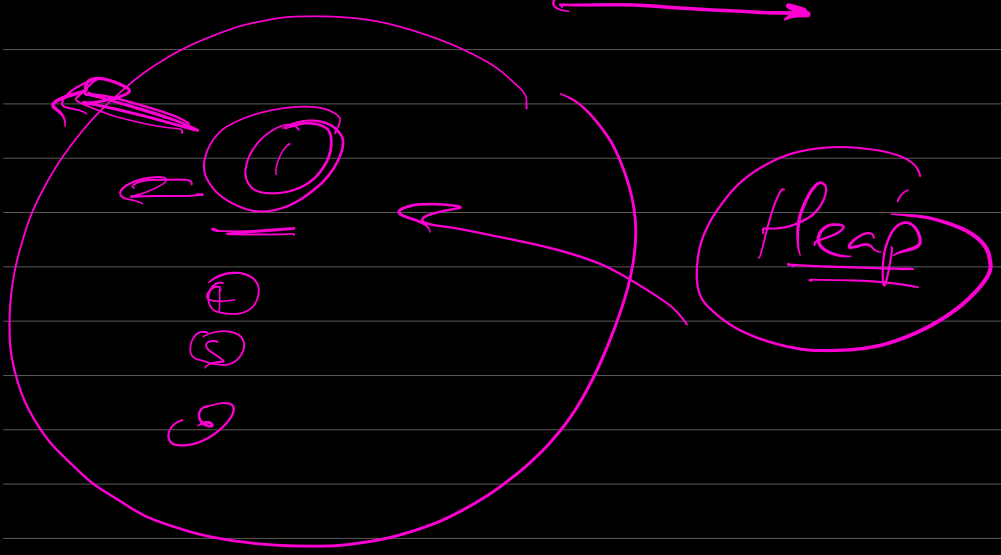
⇒ swap root with the last element
A[0] A[N-1]

⇒ Down heapify till the property of heap is satisfied.

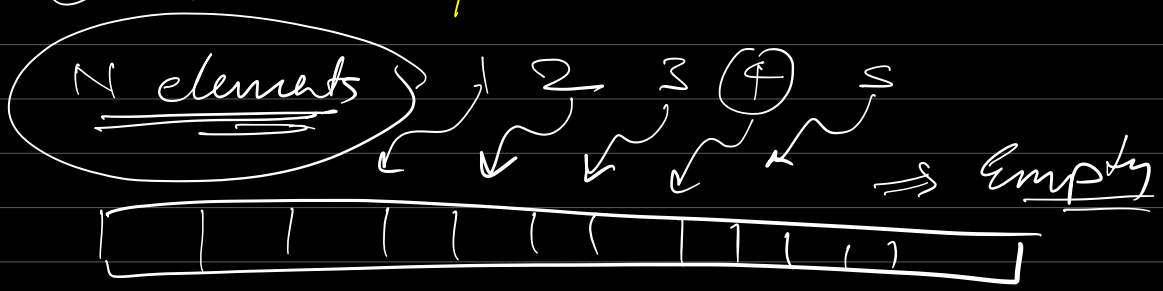
Code

```
down-heapify (0) {
    curr = 0;
    while ( curr < size && ( curr.data > root.left.data
        || curr.data > root.right.data ) {
        // If Else   H.W.
    }
}
```

T.C. = $O(\log N)$ (down heapify)



Build Heap

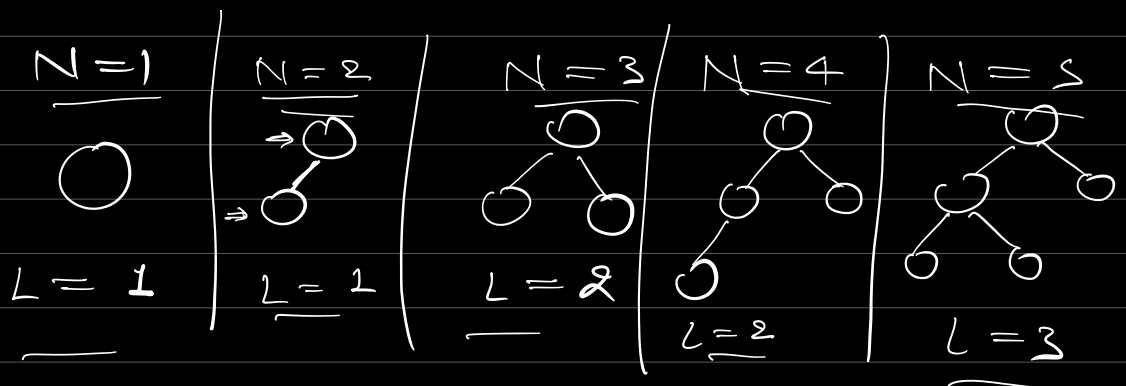


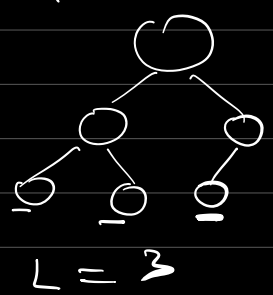
size = 0

[size++;
insert an element] ⇒ N times

T.C. = $O(N \log N)$

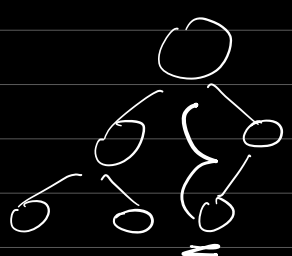
② What is the no. of leaf nodes in a complete BT with N nodes.





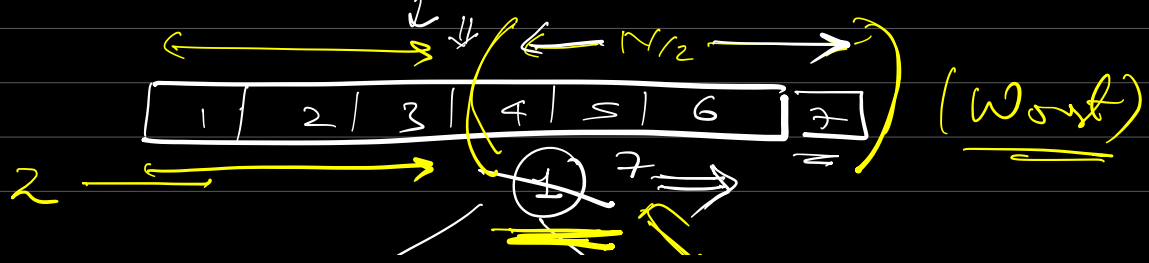
7 8 9 10
4 4 5 5

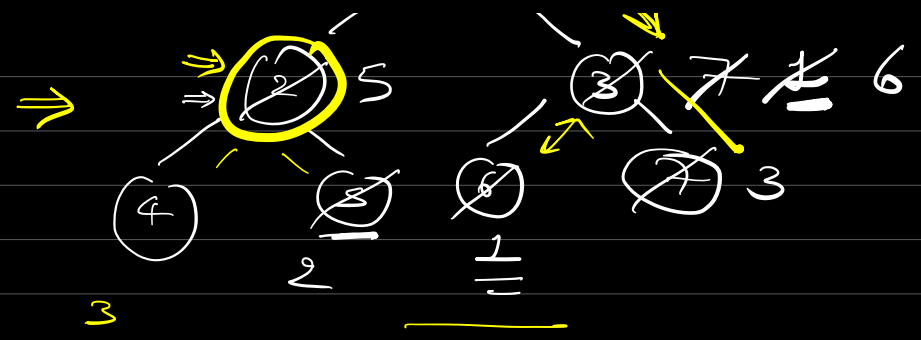
no. of leaf nodes = $\lceil \frac{N}{2} \rceil$

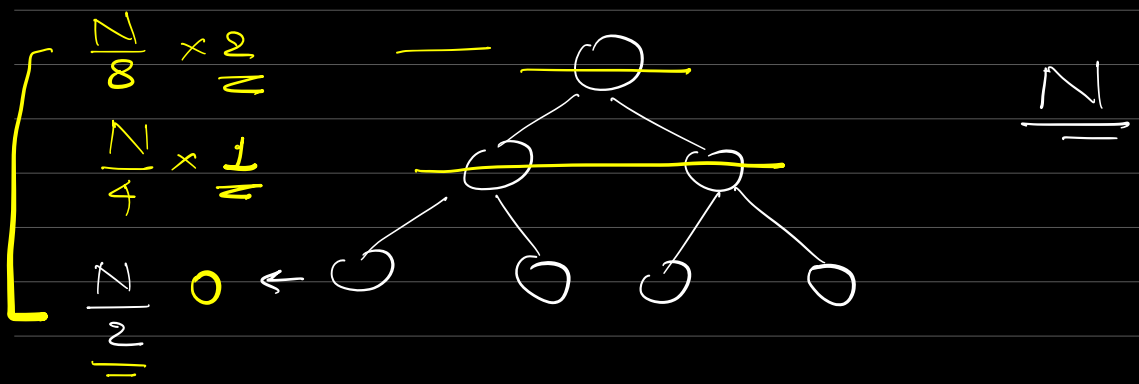


$\frac{7}{2} = 3$

Max Heap







$\frac{N}{8} \times 2$

$\frac{N}{4} \times 1$

$\frac{N}{2} \times 0$

N

Total no. of swaps $= \frac{N}{2}(0) + \frac{N}{4} \times 1 + \frac{N}{8} \times 2$

$+ \frac{N}{16} \times 3 \ldots\ldots$ ∞

$$\sum_{i=0}^{\infty} \frac{N}{2^{i+1}} (i)$$

$$\frac{N}{2}\left(\sum_{i=0}^{\infty} \frac{i}{2^i}\right) \qquad \sum_{i=0}^{\infty} \left(\frac{i}{2^i}\right) = 2$$

Total no. of swaps $= \frac{N}{2} \times 2 = N$

T.C. = O(N)

Code

Array A of size N to heap.

```
for(i = N/2 ; i ≥ 0 ; i--) {
    downHeapify(i);
}
```

T.C. = O(N)