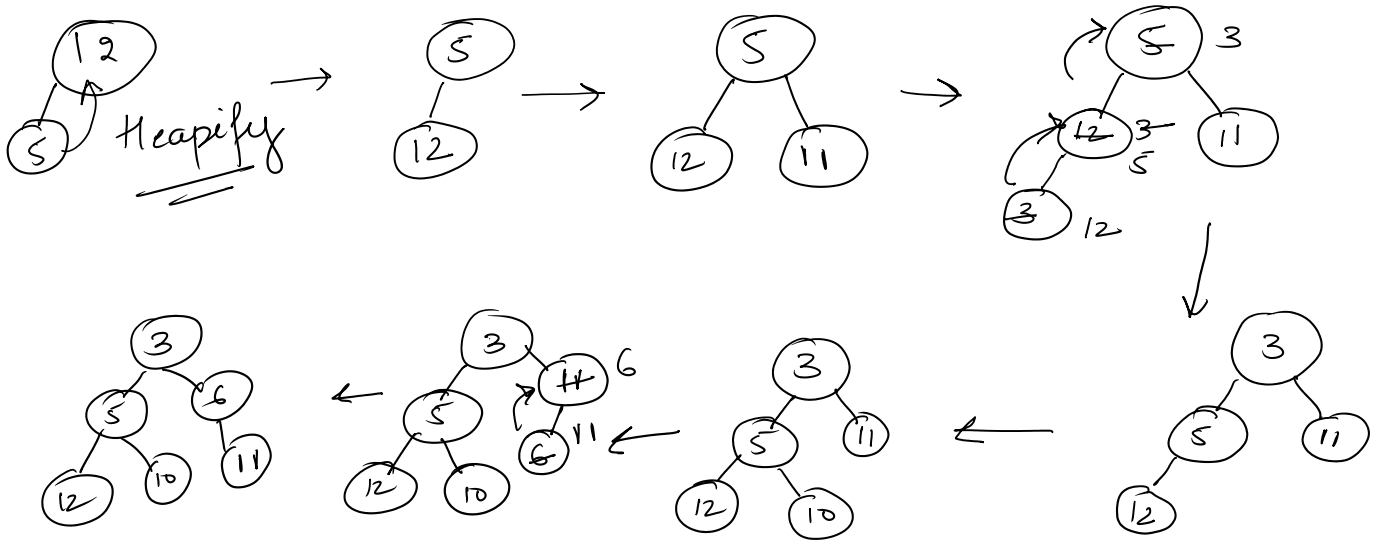
#g Insert - 12, 5, 11, 3, 10, 6, 9, 4, 8, 1, 7, 2



Insert - $O(\log n)$

Insert "n" elements $\Rightarrow O(n \log n)$ — Time

Space = $O(n)$

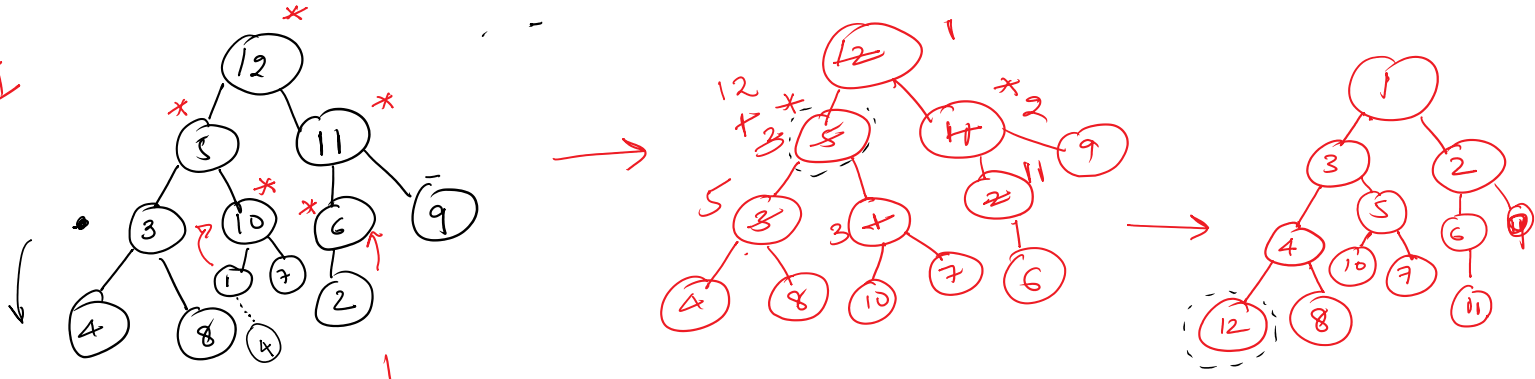
1. FLOYD'S METHOD - To build heap of n elements

- 1 Construct a complete binary tree
- 2 Mark the nodes which donot satisfy the heap property
- 3 Fix the order/property for each of the

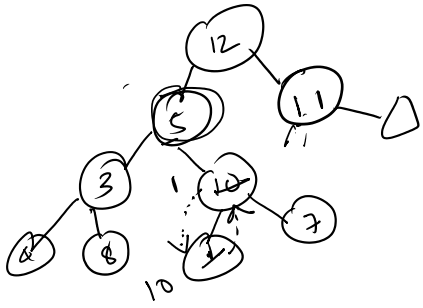
marked nodes

#g 12, 5, 11, 3, 10, 6, 9, 4, -8, 1, 7, 2

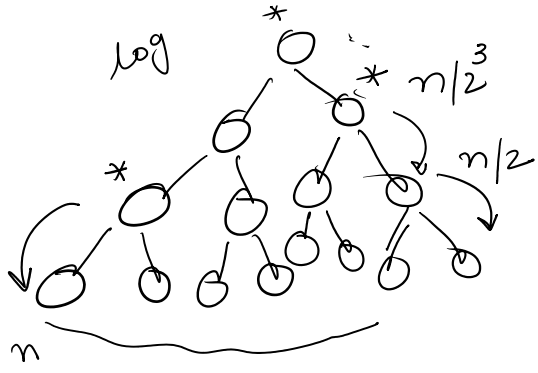
I

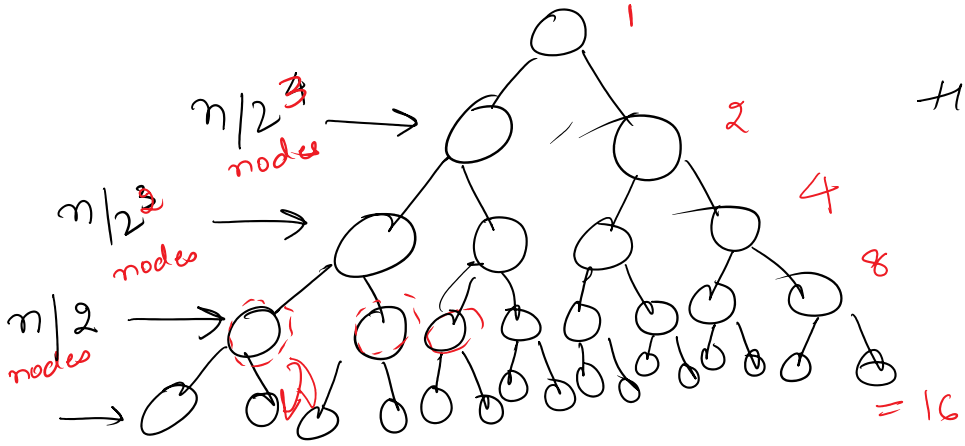


Bubble up / Sink down



$T(n) =$





How many Sink down operations?

$$T(n) = \frac{n}{2} \times 1 + \frac{n}{4} \times 2 + \frac{n}{8} \times 3 + \cdots \log n$$

$$= n\left(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots \exp \log n\right)$$

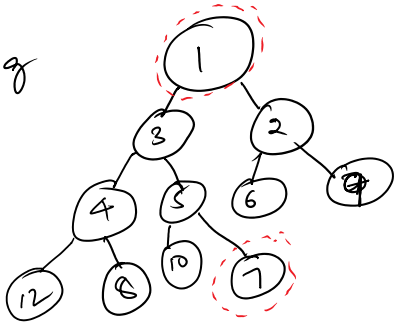
$= n \left(\sum_{i=1}^{\log n} \frac{i}{2^i} \right) \leq 2$ (geometric sum) $= n \times 2 = O(n)$

1. Heap Sort

- 1 Replace root with last leaf node
- 2 Heapify - correct the tree to maintain the heap order

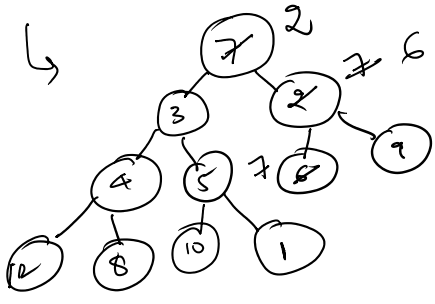
(* not consider the nodes which have been sorted)

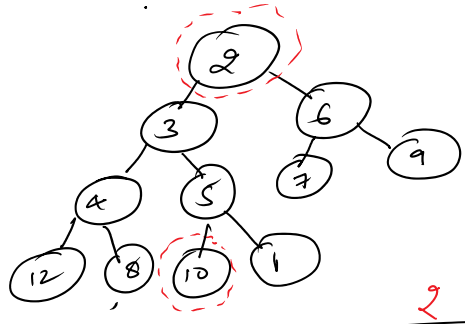
#q



	7											1
x	~~1~~	3	2	4	5	6	9	12	8	10	~~7~~	

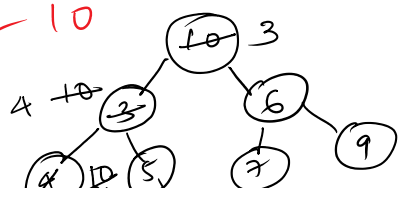
Replace 1 & 7

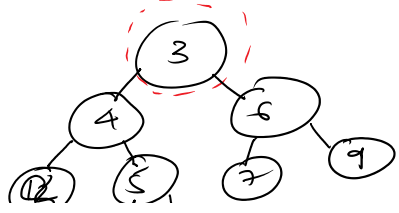


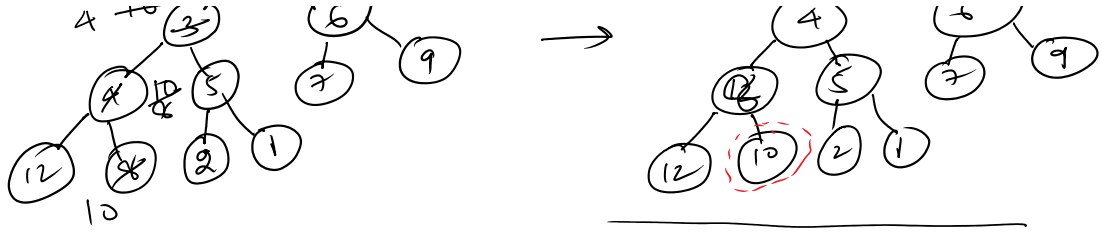


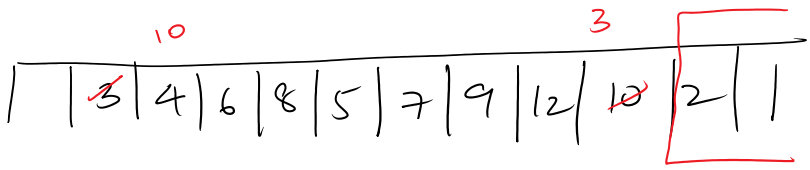
	10									2	
~~2~~	3	6	4	5	7	9	12	8	~~10~~	1	

Replace 2 & 10









(logn) → for each oper^n

Sorting "n" elements ⇒ O(n log n)