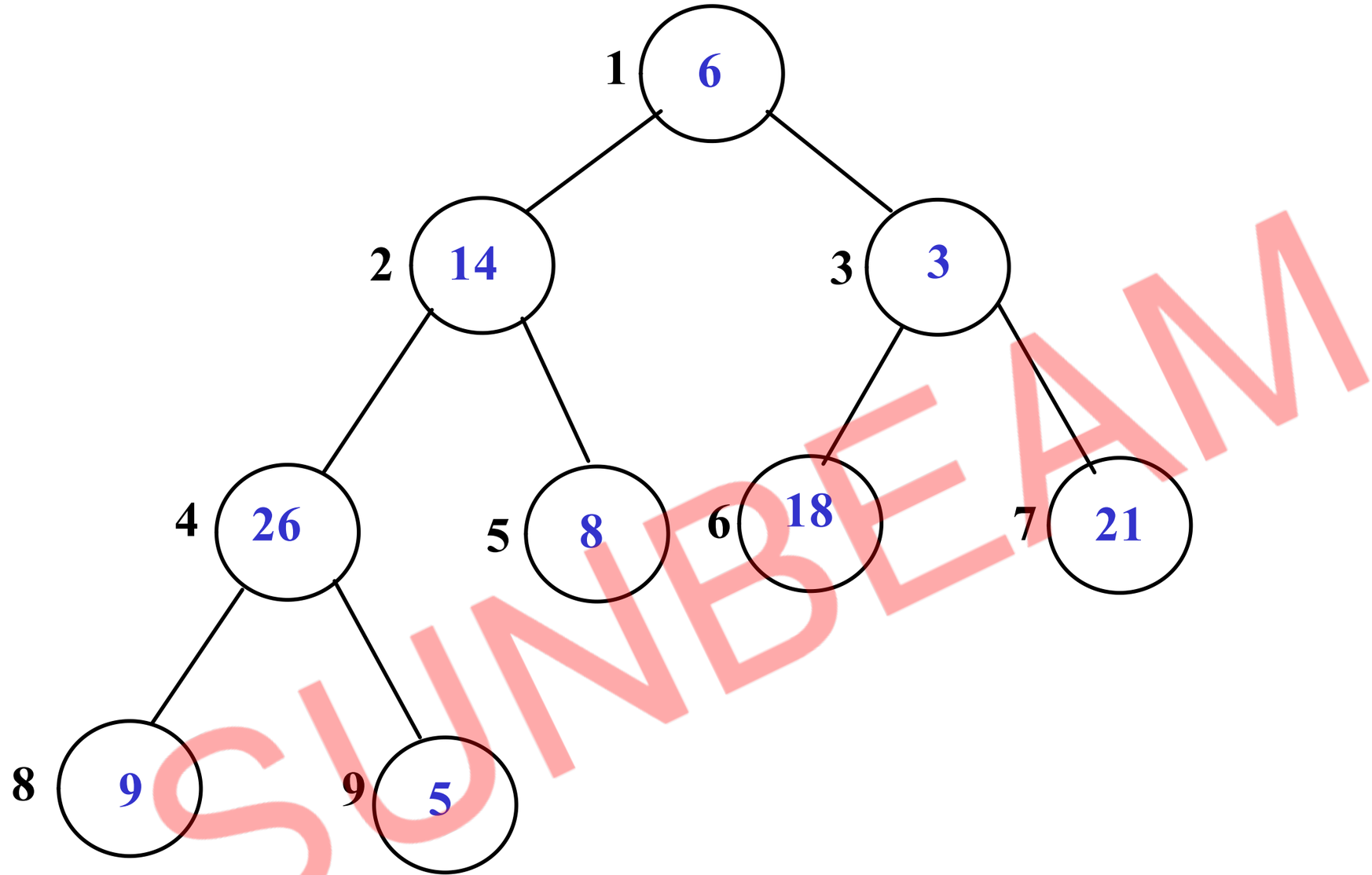
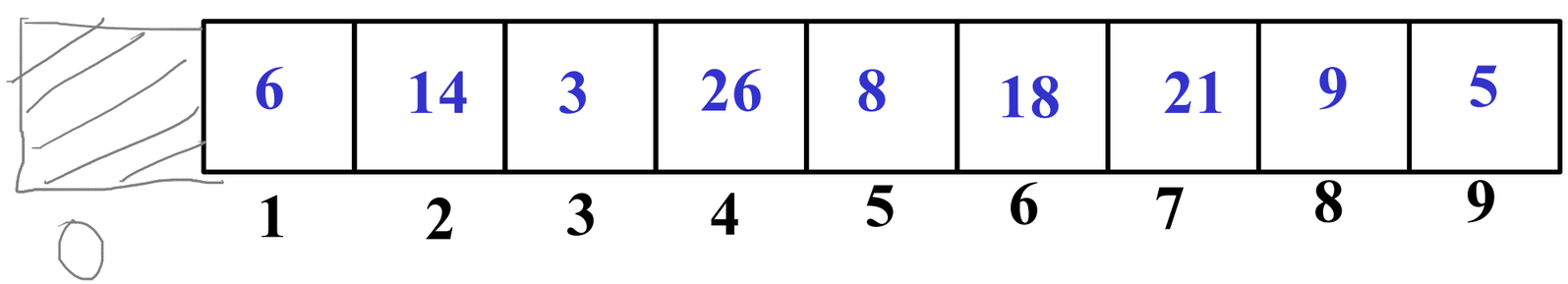
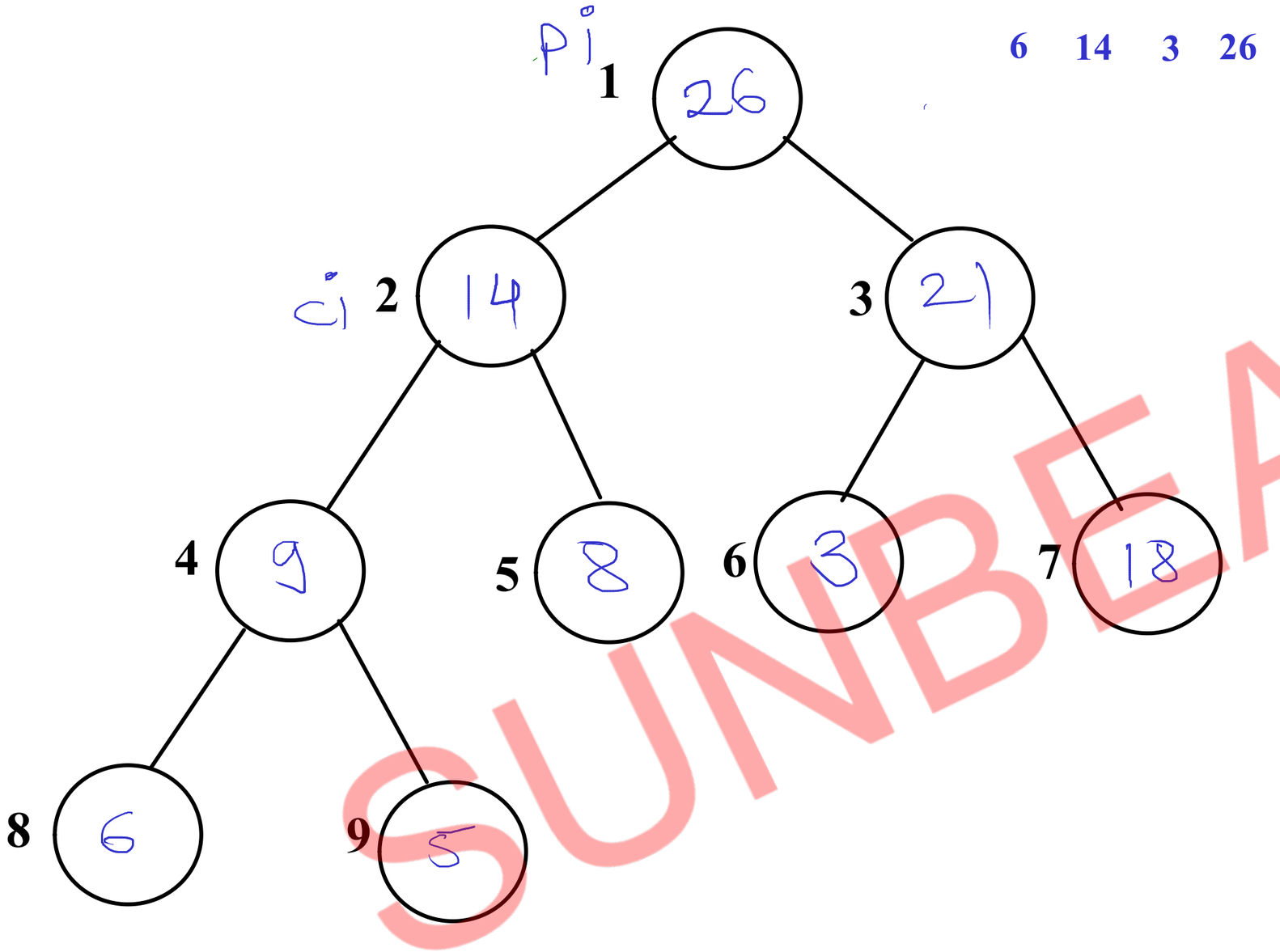
Almost Complete Binary Tree





Max Heap -add



6 14 3 26 8 18 21 9 5

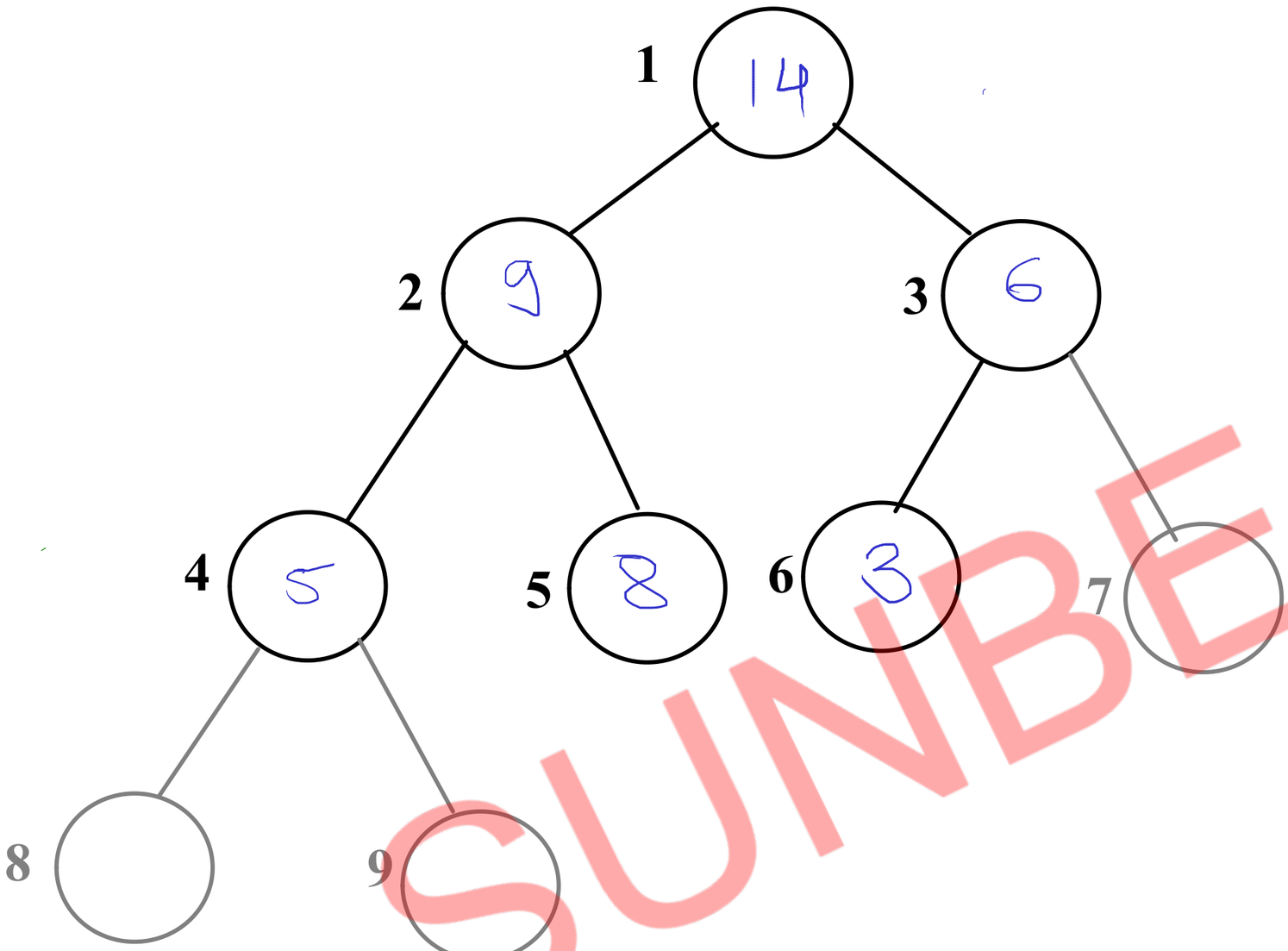
1) add new value at first empty index from left side

2) adjust the position of newly added element into array by compairing it with all its ancestors

$T(n) = O(\log n)$

26	14	21	9	8	3	18	6	5
1	2	3	4	5	6	7	8	9

Max Heap - delete



14	9	6	5	8	3			
1	2	3	4	5	6	7	8	9

1) always root element (max) is deleted from heap.

max = 26, 27, 18

2) place last element of heap at empty place of root node.

3) Adjust position of root so that it will be max heap by comparing with all it descendents.

$T(n) = O(\log n)$

Heap Sort

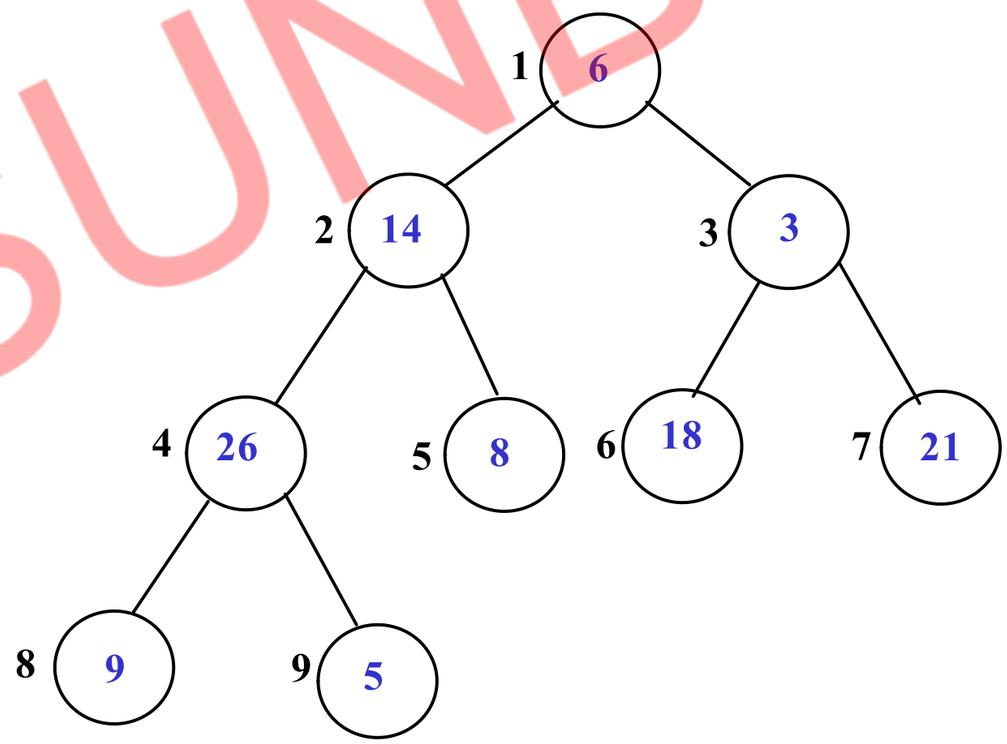
- Heap sort is a two step process
//1. create min/max heap
//2. delete all elements from heap

arr length = n

n.log n + = 2n log n
n. log n

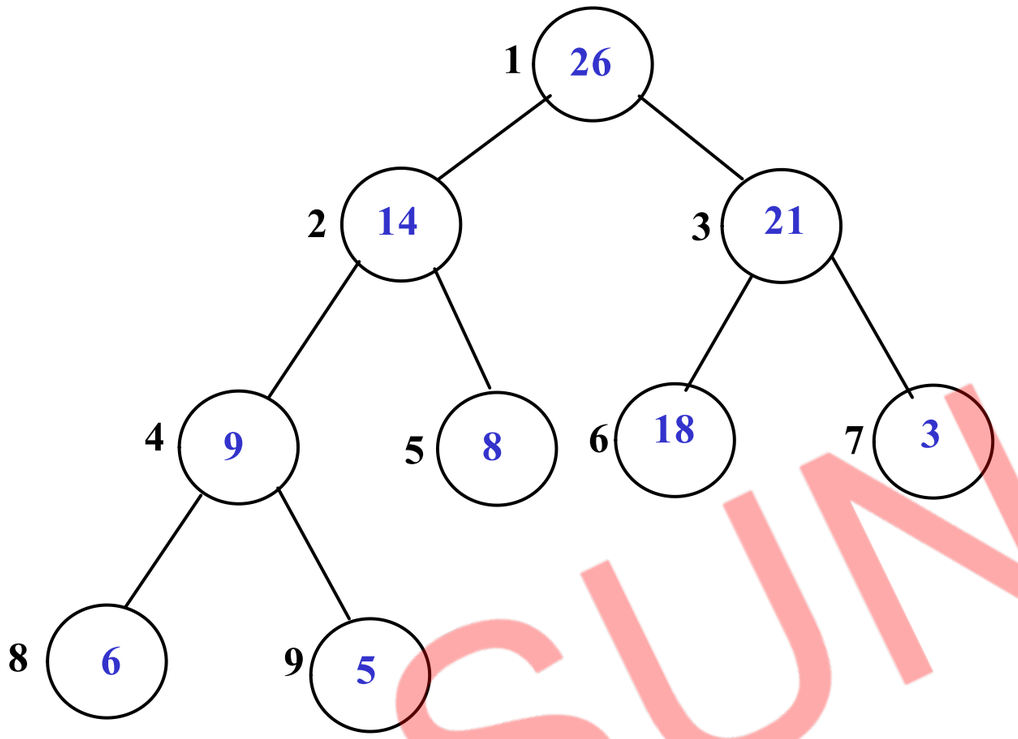
T(n) = O(n log n)

6	14	3	26	8	18	21	9	5
1	2	3	4	5	6	7	8	9



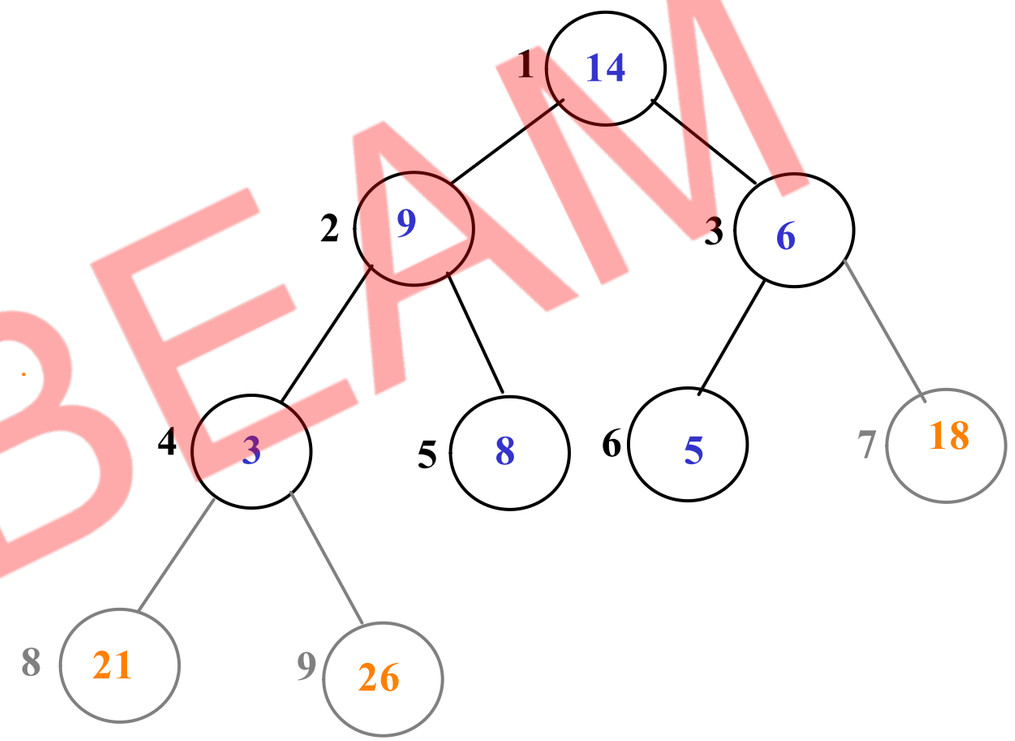
Heapify

26	14	21	9	8	18	3	6	5
1	2	3	4	5	6	7	8	9



Delete

14	9	6	3	8	5	18	21	26
1	2	3	4	5	6	7	8	9



size = 9

9/2 = 4 — first parent from right side

$T(n) = O(n)$

Merge Sort

//1. divide array into two parts
//2. sort both partitions individually
//3. merge sorted partitions into temp array in such way that, temp arr is sorted
//4. overwrite temp array into original array