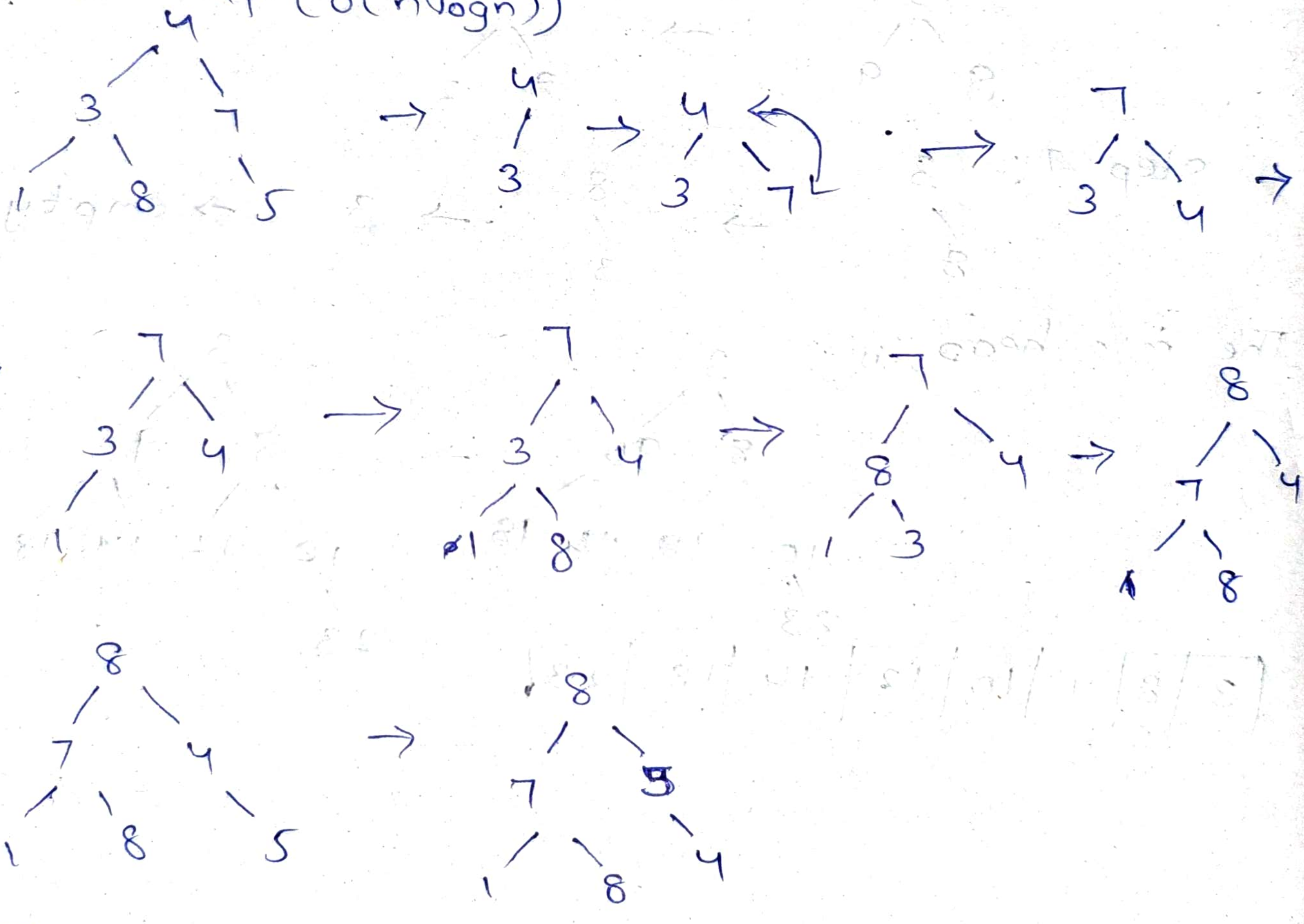
Array

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

First apply Max heap.general (o(nlogn)) and another method heapify (o(n)) best and fast as time complexity o(n).

Max heap	Max heap(heapify.)
* Time complexity O(n logn) * It check with parent node and child node from top.	* Time complexity O(n). * It checks from right and also with nonleaf node (has child min '1' for parent). $(\frac{n}{2}+1)$n for index starts '1'. $(\frac{n}{2}-1)$n for index starts '0'.

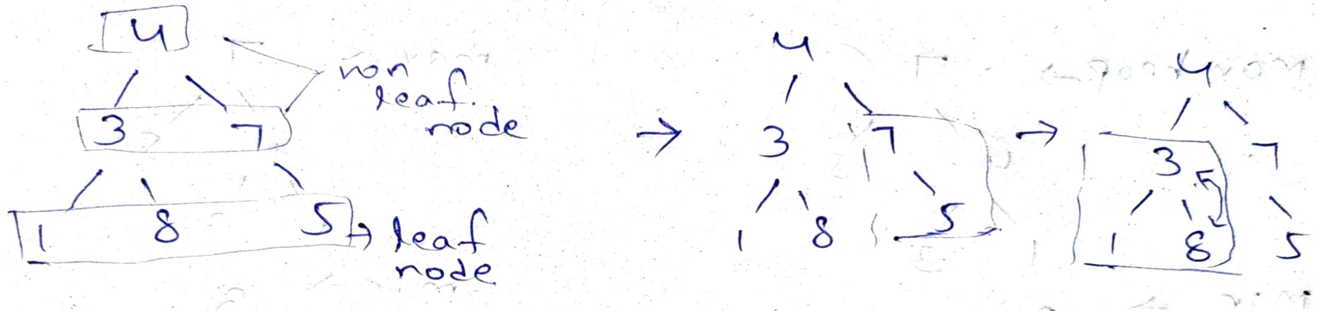
Max heap (o(nlogn))

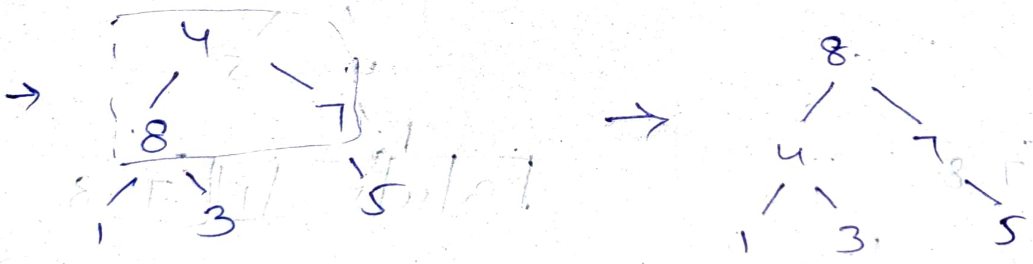


* The building of max heap in normal follows parent node is $\geq$ to child nodes and arrange to the max number with node and same for all. The Tc → o(nlogn).

Heapify method:-

* It is precise and different it follows leaf node and non leaf node.
* The array has total 'n' indexs is '6' then we see
* $(\frac{n}{2}+1)n \rightarrow (\frac{6}{2}+1)n \rightarrow (3+1)n \rightarrow 4(n)$: It is '4' to n i.e 6.
* 4 to 6. are leaf nodes. that are max nodes. which has no child nodes.
* Now 1, 2, 3 are maxheap to perform heapify. in the '3' is max and starts from right.
* The TC is o(n). so, we prefer this.





→ (8 is max, swap with 3).

→ again with parent (4) is swap with '8'.

Now for min heap.

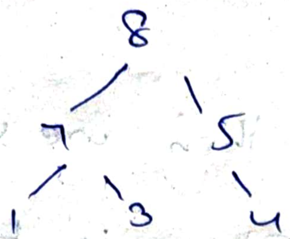
Steps:-

→ The first top (or) first node is extracted (removed) then swap with last node. It same, after check it is max heap with all parent and child nodes.

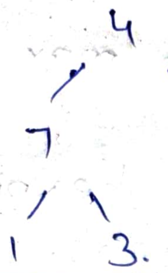
→ If it is rearrange and swap. After this again min heap remove first & swap with last.

→ Same follows for all.

Max heap

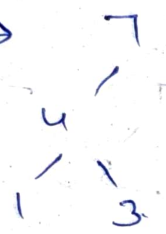


Step 1: min heap →

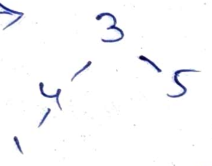


| 4 | 7 | 5 | 1 | 3 | [8

max heap →

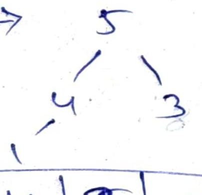


Min →

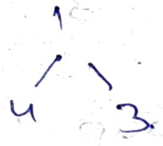


| 3 | 4 | 5 | [7 8

max →

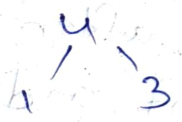


→

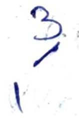


| 1 | 4 | 3 | 5 7 8

→ max



→ min



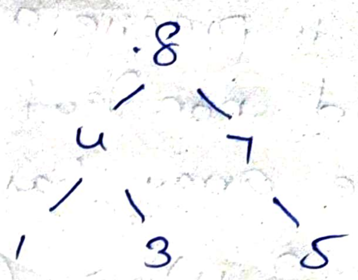
| 3 | 1 | 4 | 5 7 8

→ 1 [3 4 5 7 8

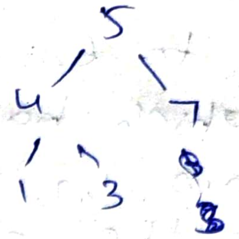
→ empty. [1 3 4 5 7 8

| 1 | 3 | 4 | 5 | 7 | 8 |

Heapify max heap.

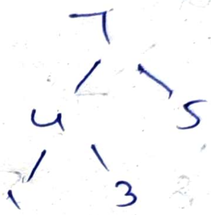


Step: min heap →

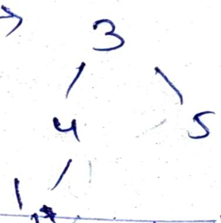


| 5 | 4 | 7 | 1 | 3 | [8

max →

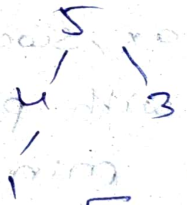


min →

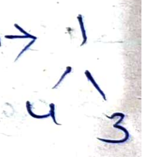


| 3 | 4 | 5 | 1 | [7 8

max →

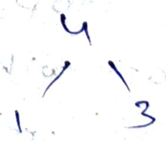


min →

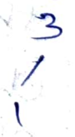


| 1 | 4 | 3 | [5 7 8

→ max



→ min



| 3 | 1 | [4 5 7 8

→ 1 [3 4 5 7 8

→ empty. [1 3 4 5 7 8

| 1 | 3 | 4 | 5 | 7 | 8 |