# Priority Queue

Airplane - booked - 10 people waiting - 1 seat vacated - which passenger to choose - ~~by~~

Ways -

1) First Come First serve
2) VIP in waiting list.

Priority Queue is an alternative to Queue, which is FIFO.

1) Insert
2) getMax / getMin
3) removeMax / removeMin

Let's consider the elements to be their priority value.

| | insert | getMin / getMax | removeMin / removeMax |
|---|---|---|---|
| Arrays (unsorted) | $O(1)$ | $O(n)$ | $O(n)$ finding + $O(n)$ shifting = $O(n)$ |
| Array sorted | $O(n)$ finding & shifting | $O(1)$ | $O(1)$, Shifting takes time |
| LL (unsorted) | $O(1)$ | $O(n)$ | $O(n)$ |
| LL (sorted) | $O(n)$ | $O(1)$ | $O(1)$ |
| BST | $O(H)$ = n in worst case | $O(H)$ | $O(H)$ |

n in worst case

| Balanced BST → $O(\log n)$ | $O(\log n)$ | $O(\log n)$ |
|---|---|---|

BST mein worst case mein height $O(n)$ ho sakti hai, but balanced BST humesha height ka difference ko 1 se jyada nhi hone deta. So height is always $O(\log n)$.

So in all except Balanced BST, if n elements are to be inserted or removed, the time complexity will be $n^2$.

But in Balanced BST that would be $(n \log n)$.

| Hashmap $O(1)$ | $O(n)$ | $O(n)$ | $= O(n^2)$ |
|---|---|---|---|

Issues with Balanced BST-

1) Balancing factors
2) Storing is complicated.

So we have a new DS designed called heaps.

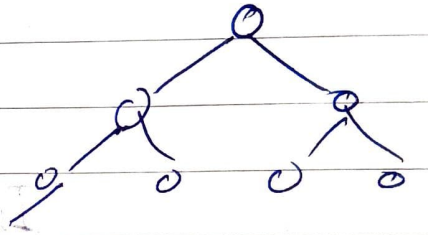
Heaps Major Property

1) Complete binary tree
2) Heap order property.

Complete Binary Tree - : All levels except the last level should be completely filled -
And the last level too should be filled left to right.

CBT

Minimum no. of nodes with height h CBT



$2^0 + 2^1 + \cdots - 2^{h-2} + 1$

$= \boxed{2^{h-1} - 1 + 1} \Rightarrow$ Minimum

$= \boxed{2^{h-1}} \Rightarrow$ Minimum no. of nodes of a tree of height h.

- Maximum No. of nodes for height $h = \boxed{2^h - 1}$

$2^{h-1} \leq N \leq 2^h - 1$

$2^{h-1} \leq N$

$h - 1 \leq \log_2 N$

$\boxed{H \leq \log_2 N + 1}$

$2^h - 1 \geq N$

$2^h \geq N + 1$

$H \geq \log(N+1)$

$\log_2(N+1) \leq H \leq \log_2 N + 1$

$O(\log_2 N) \leq h \leq O(\log_2 N)$

CBT height = log(N)

| 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|

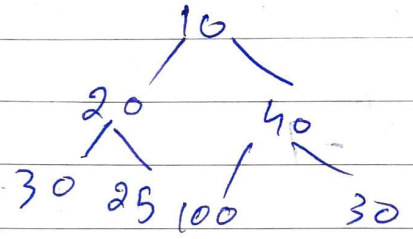
For parent - $\frac{index-1}{2}$

i
0 → 1, 2
1 → 3, 4

$(2i+1, 2i+2)$

## Insert & Delete in Heaps

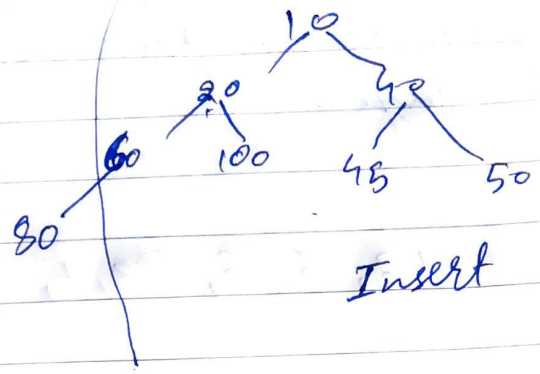
Heap Order Property < MinHeap / MaxHeap.




MinHeap mein root ki value uske dono children se choti honi chahiye

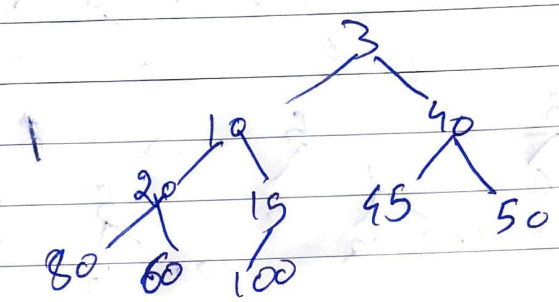
Validity of a Heap is checked by whether its a valid CBT or not or if it satisfies Heaporder property.

MaxHeap mein bas root bada hoga then both children.




Insert 15

3



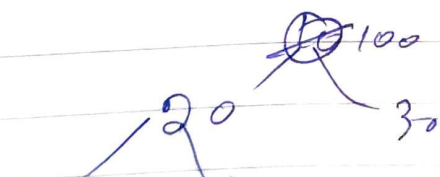

· 2 Stop conditions when parent's value is less than child, & again when we have reached root.

Insertion is taking $O(h) = O(\log n)$

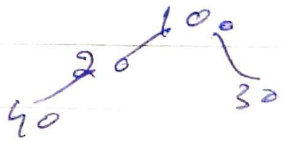
Is toggling & swapping is called upheapyfy.

## Delete

Remove the element having highest or lowest priority for Max Heap & Min Heap.



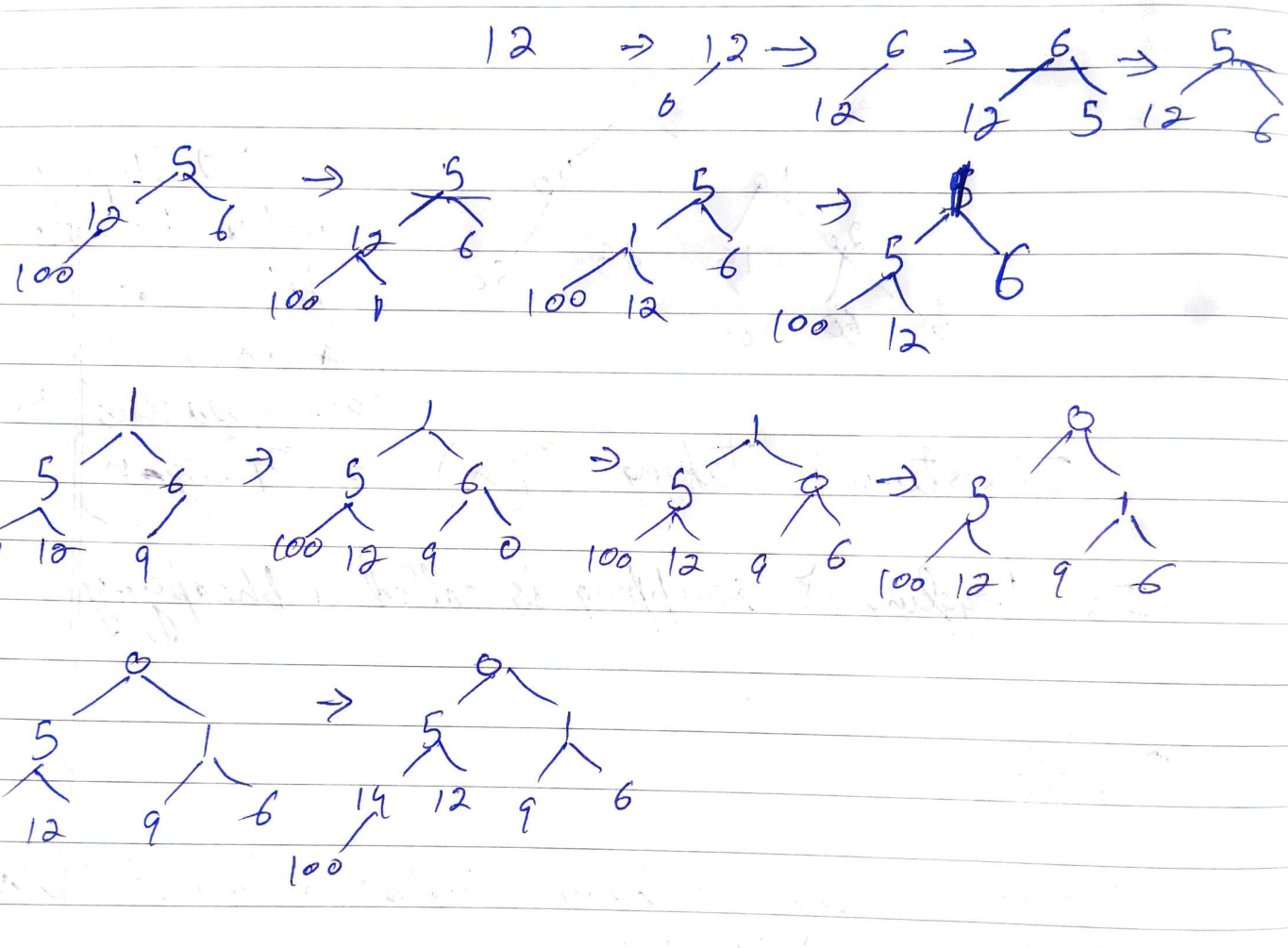

Delete the element.

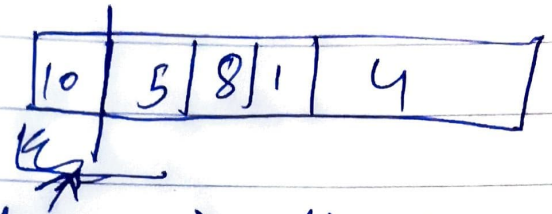



Downheapify mein apne dono children ko compare karke, minimum wale se swapkana hai.

Jab tak karenge jab tak apne correct position pe na pahunch jaye.

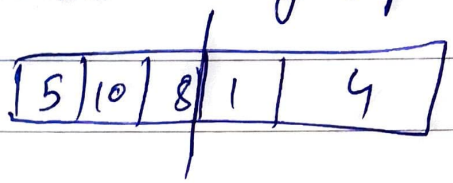
12, 6, 5, 100, 1, 9, 0, 14






Heap has 1 element & array has n-1 elements.

Pehle 5 insert kiya, phir heapify kiya




Then