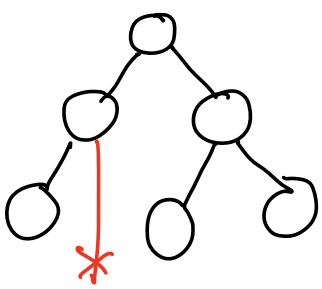
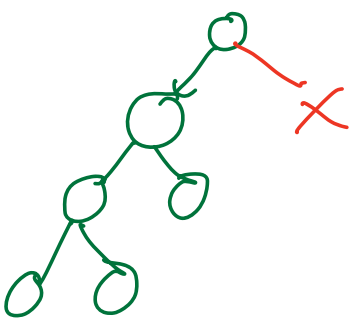
Heaps

Complete Binary Tree (Poc - req)

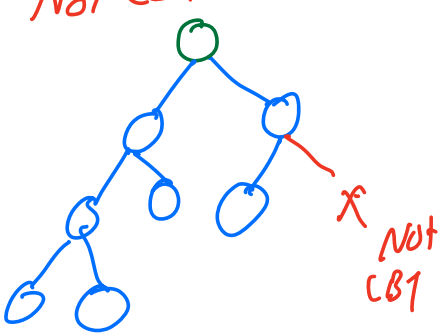
A BT is a CBT if :

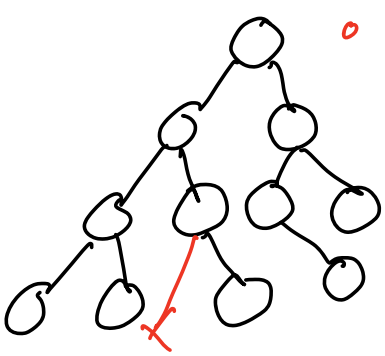
1) A BT is said to be CBT if
   i) All nodes have to be filled level by level from left → right
   ii) At all levels it should be completely filled except last level (we can either fill it or not)

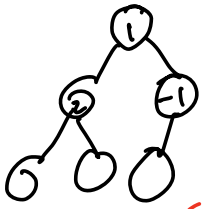




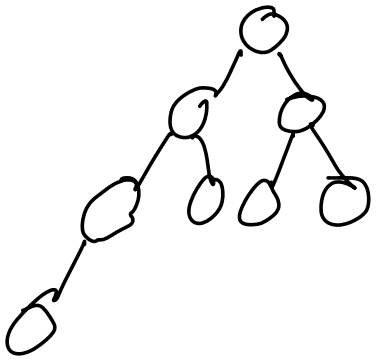
Not CBT







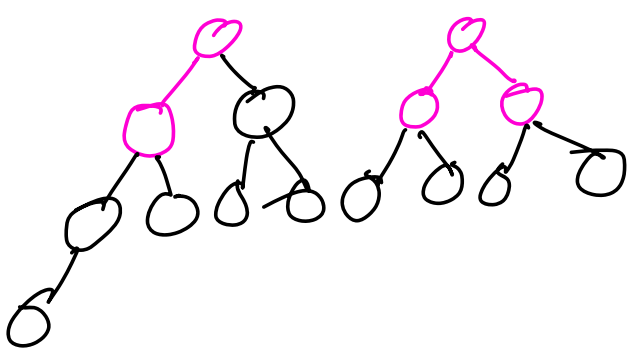
CBT ✓



CBT ✓

Balance Binary Tree: CBT for every Node.

Height of CBT	Min Node	Max Node
1	2 : 2^1	3 : 2^2-1
2	4 : 2^2	7 : 2^3-1
3	8 : 2^3	15 : 2^4-1
h	2^h	$2^{h+1}-1$



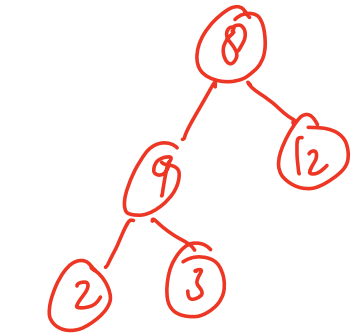
$2^h = N \Rightarrow h = O(?)$ ↗ $\log_2 N$

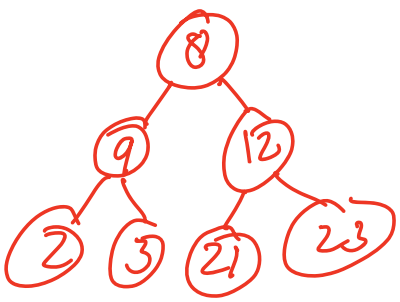
H

$2^{h+1}-1 = N \Rightarrow h = O(?)$

Height of N nodes CBT is $O(\log_2 N)$

Implementation of CBT





8, 9, 12, 2, 3, 21, 23, 26

Whenever a new node is created insert in a queue, & delete from front of queue only if both left & right child are filled

TC: $O(N)$

SC: $O(N)$

2 Problems:

→ Extra space

→ Node → parent is tough

Arrays

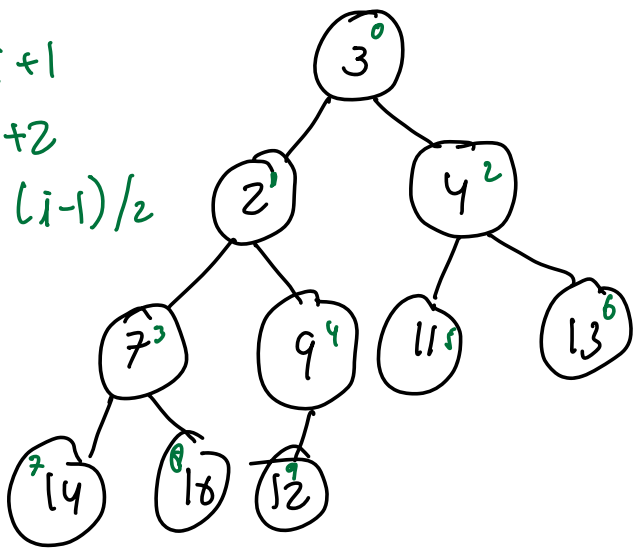
	0	1	2	3	4	5	6	7	8	9
list[int]l:	3	2	4	7	9	11	13	14	15	12

l: 2i+1

r: 2i+2

parent: (i-1)/2

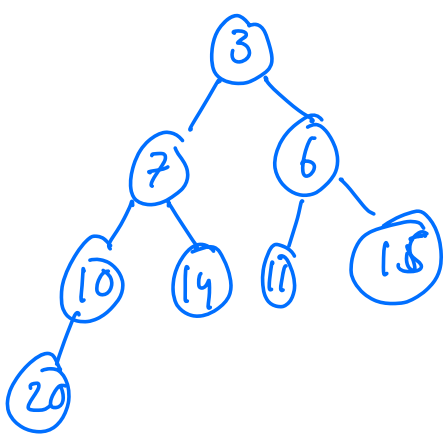
node	l	right
0	1	2
1	3	4
2	5	6



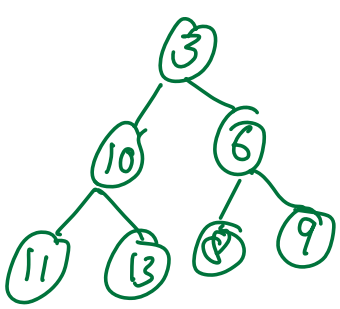
i/2

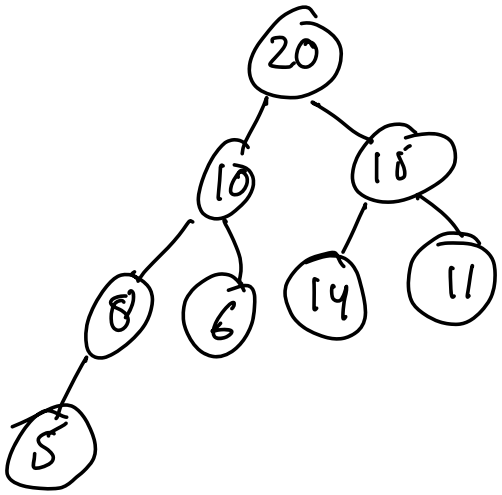
Heap:

1) CBT
2) (i) for any node > both child : Maxheap
   (ii) for any node < both child : Minheap



: min heap





: maxHeap

Min heap:

10 , 3 , 8 , 24 , 0 , -10 , 26

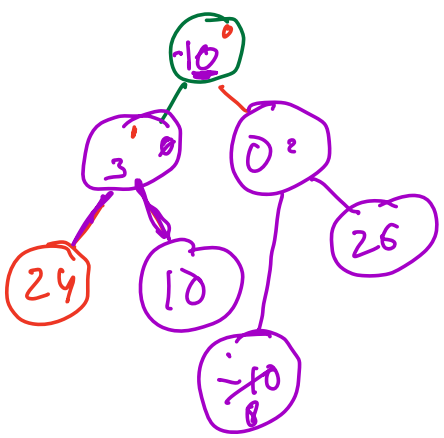
0 1 2 3 4 5 6

~~10~~ , ~~3~~ , ~~8~~ , 24 , ~~0~~ , ~~-10~~ , 26

~~3~~ ~~10~~ 0 10 8 Min heap

~~0~~ 3

-10

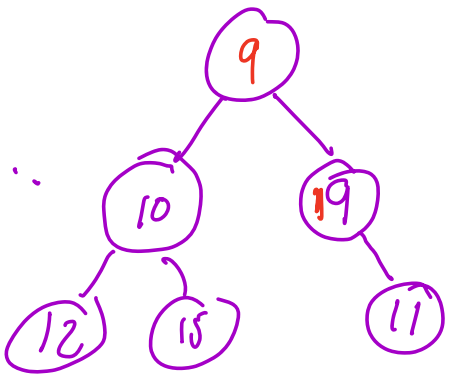


0 1 2 3 4 5 6

10 , 3 , 8 , 24 , 0 , -10 , 26

10 3 0

min Heap



Break : 8 : 52

10 , 3 , 8 , 24 , 0 , -10 , 26

arr → 10, 3

Pseudo Code

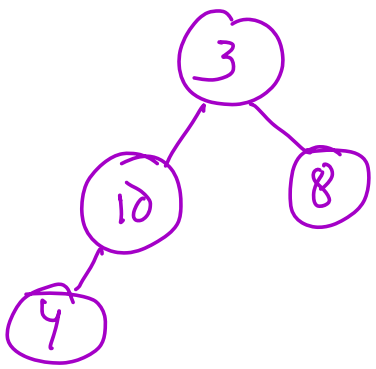
```
void Insert ( List<int> arr, int ele )
{
    arr.add(ele)
    int ind = arr.lenght - 1;
    int par = (ind - 1) / 2;
    while( ind != 0 && arr[par] > arr[ind])
    {
        swap( arr[ind], arr[par])
        ind = par;
        par = (ind - 1) / 2 ;
```

}
}

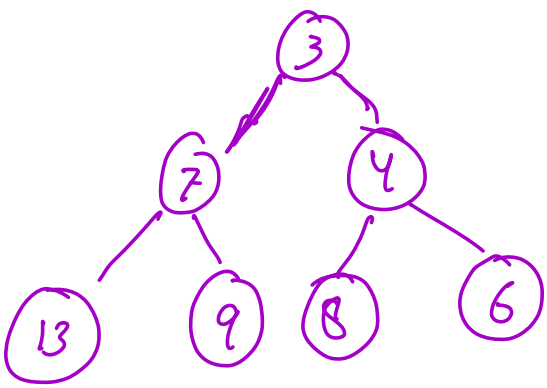
10

arr → 0: 3, 1: 10, 2: 8, 3: 10

l = 3



insert → O(log N)
getMin → O(1)
search → O(N)
deleteMin / Max → O(log N)
delete Random → O(N)



Delete

O(log N)

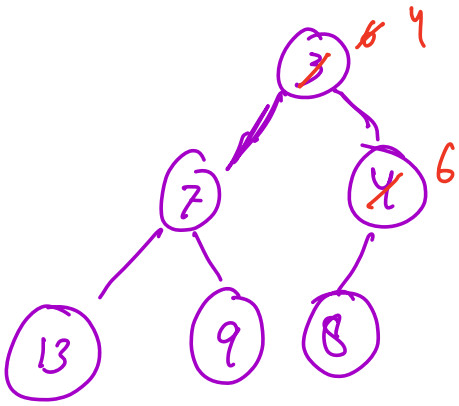
3, 7, 4, 13, 9, 8, 6

① Swap root & last ele
② Delete last element
③ Propogate down

N → n log N

Not optimal

insert Heap → O(N)



4 6 6

3, 7, 4, 13, 9, 8,