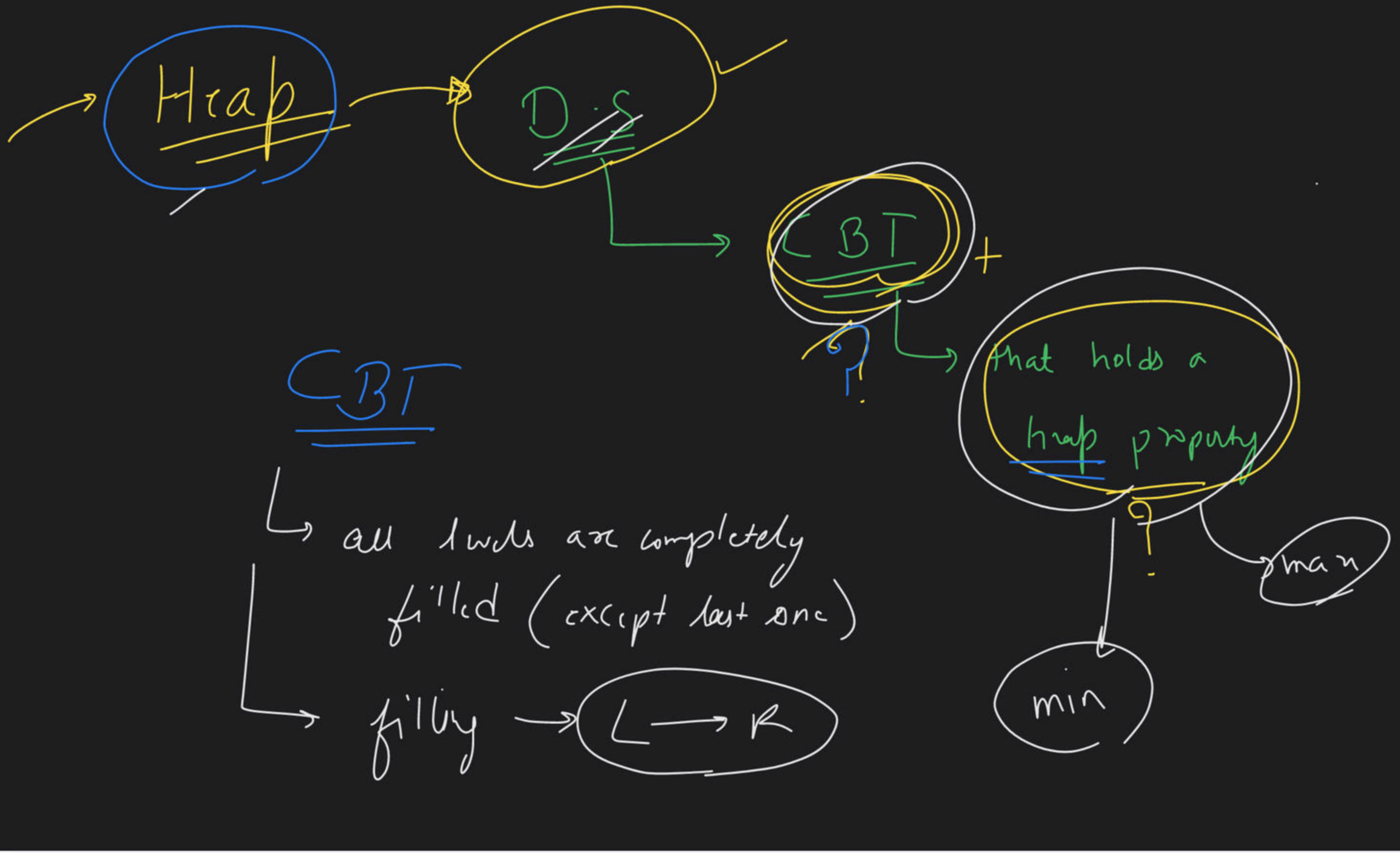
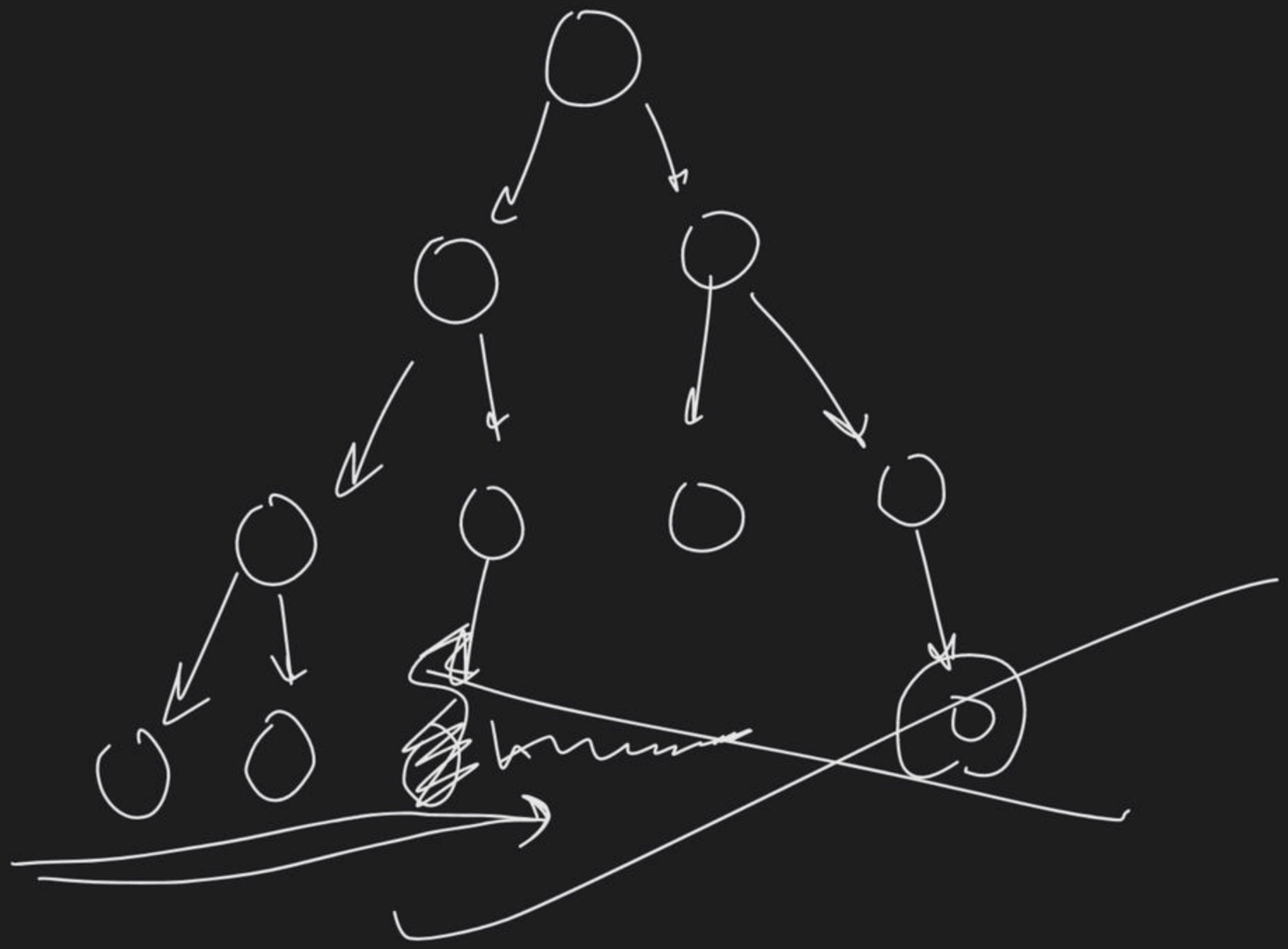
Heaps Class - 1

Special class

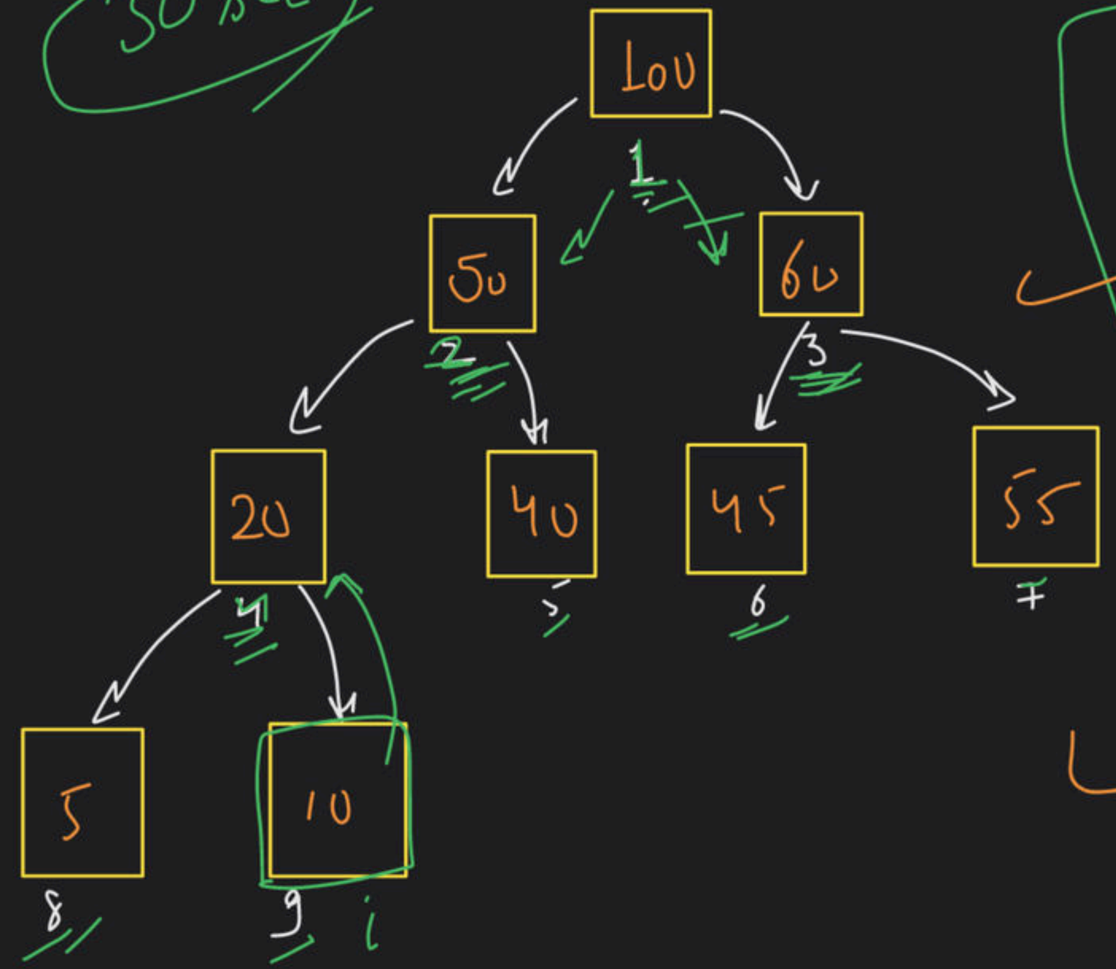
Love Babbar • May 21, 2023


Heap
D.S
CBT
+
that holds a
heap property
?
?
CBT
all levels are completely
filled (except last one)
filling
L → R
min
max



Heap (Using Arrays)

30 sec



indexing → 0

parent = i

leftChild = 2i+1

rightChild = 2i+2

parent = i

leftChild = 2*i

rightChild = 2i+1

index = i

parent = i/2

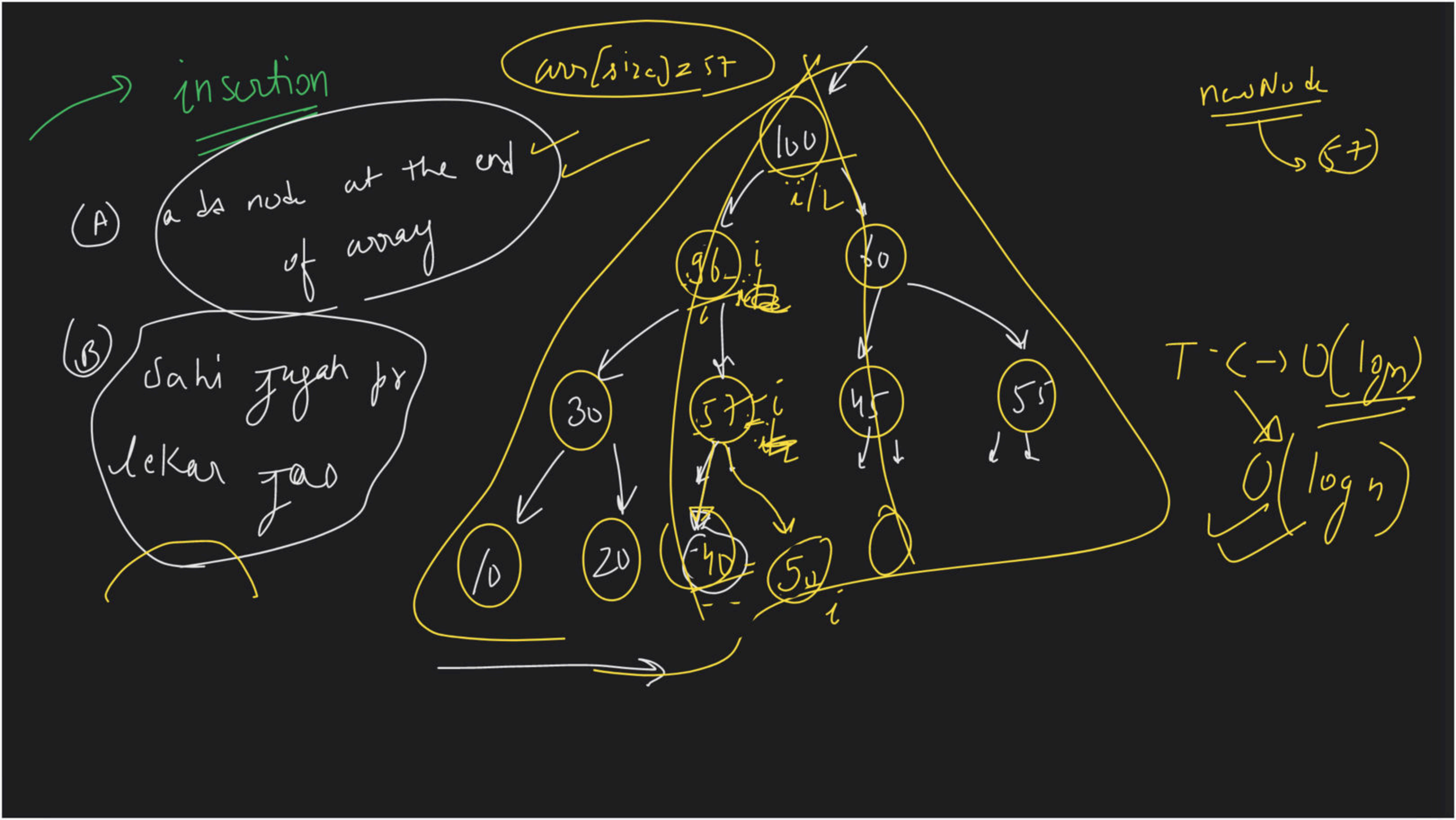
-1	100	50	60	20	40	45	55	5	10
0	1	2	3	4	5	6	7	8	9

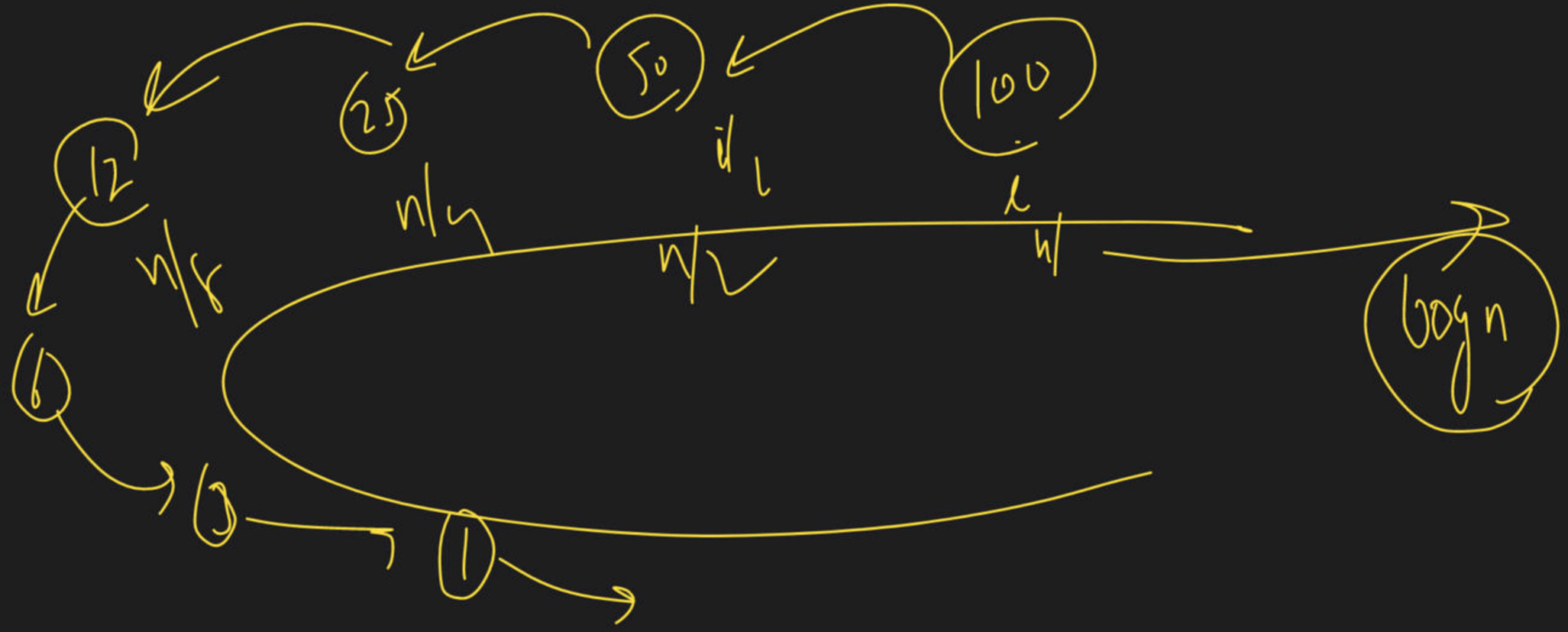
i → 9 → 4 → 2 → 1

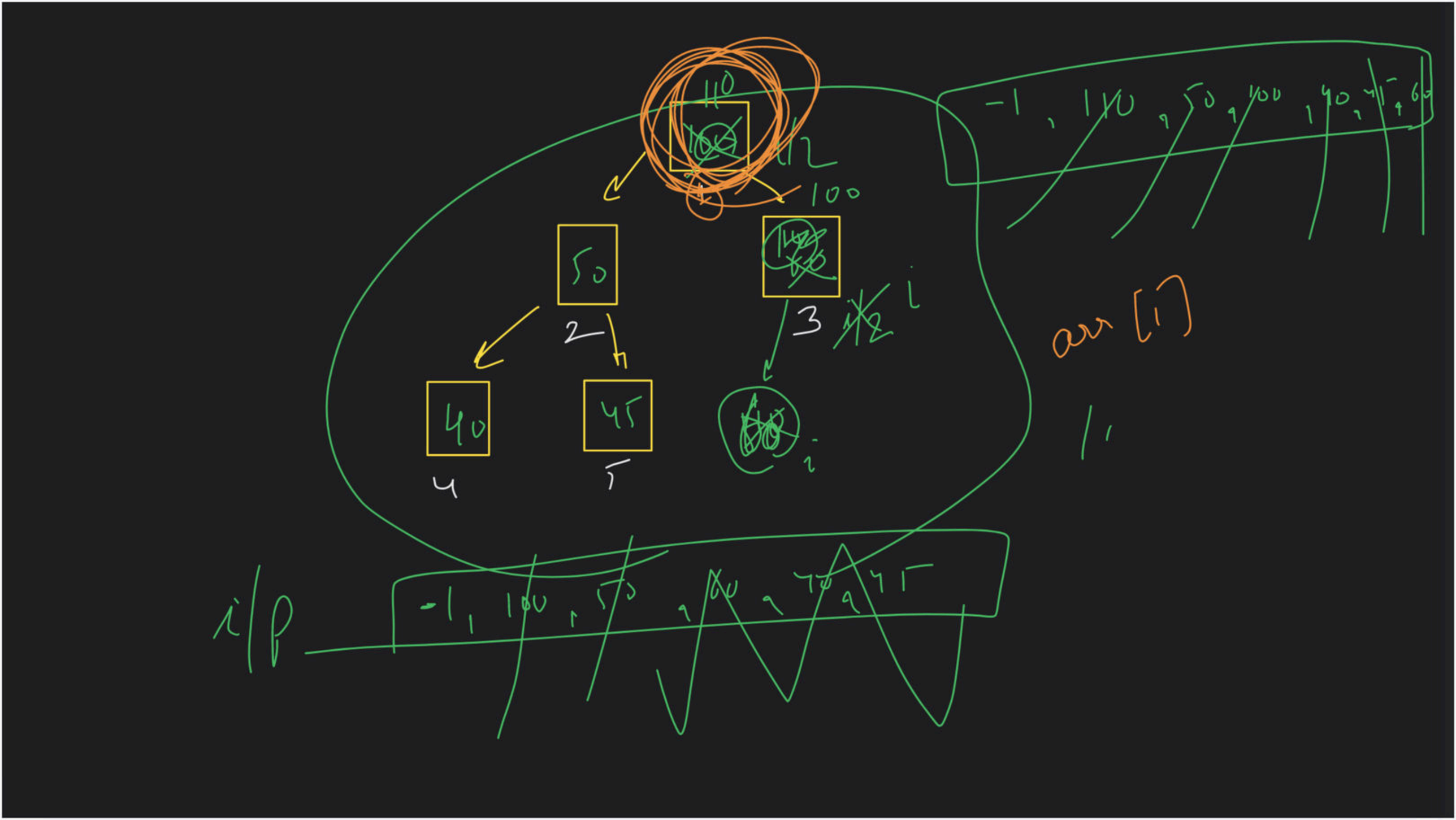
i/2

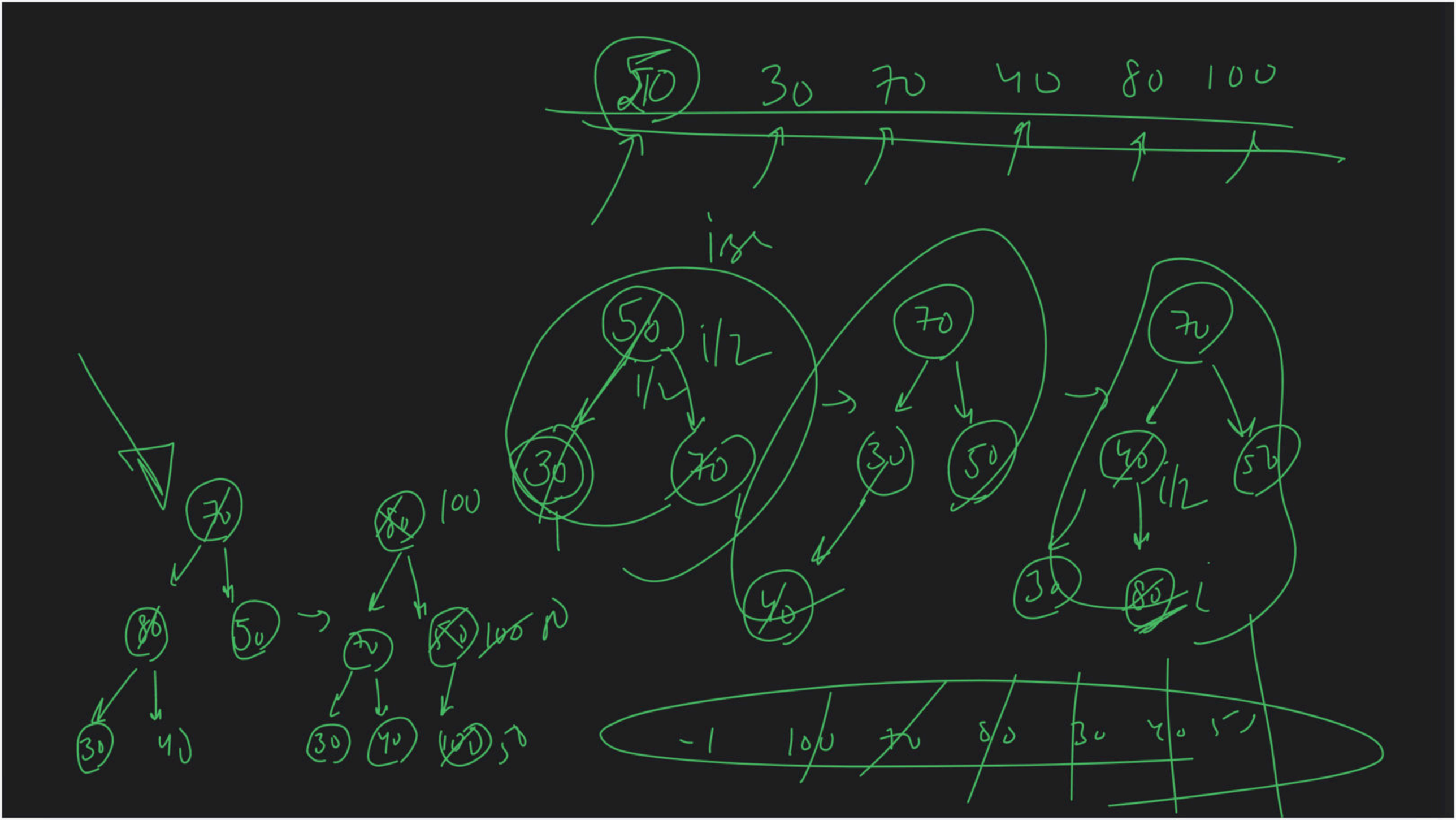
1-indexed

2i 2i+1


insertion
arr[size] = 57
(A) add node at the end of array
(B) sahi jagah par lekar jao
100
96
50
30
57
45
55
10
20
40
50
newNode(57)
T.C → O(log n)
O(log n)


12
25
50
100
n/4
n/2
n
log n
6
5
1

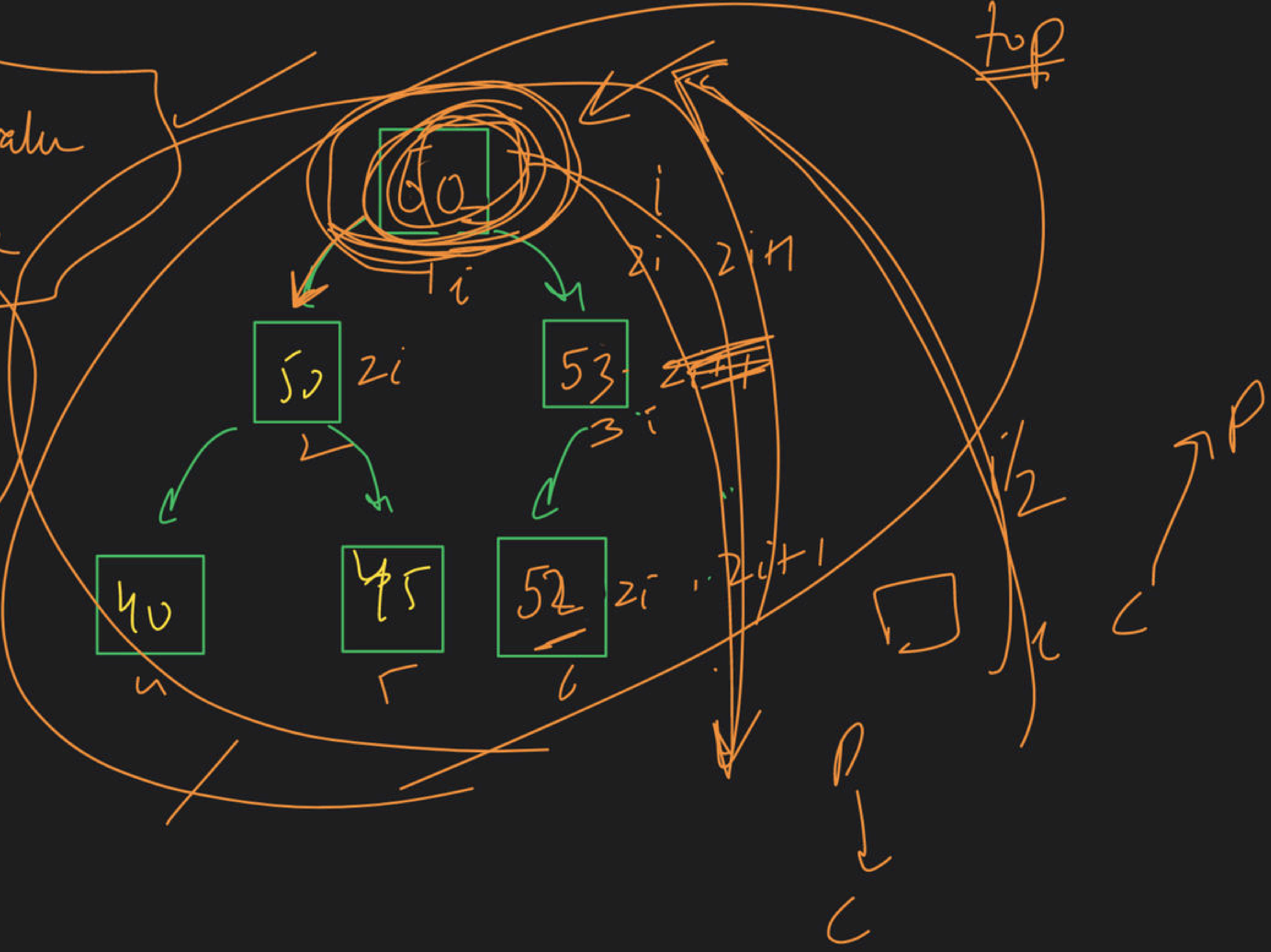

arr[i]
i/p

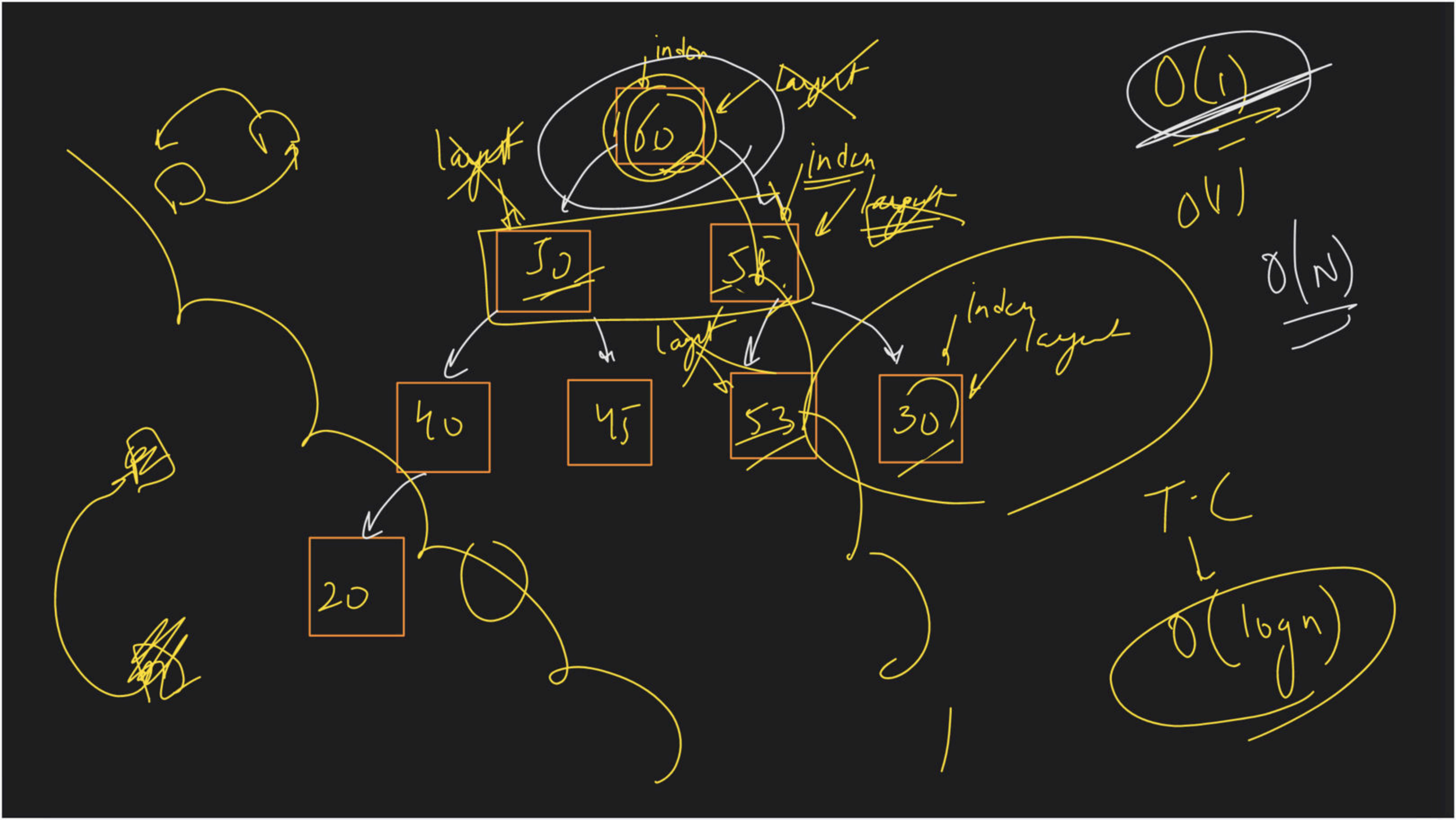

10
30
70
40
80
100
i/2
i/2
i/2
i/2
-1
100
70
80
30
40
-1

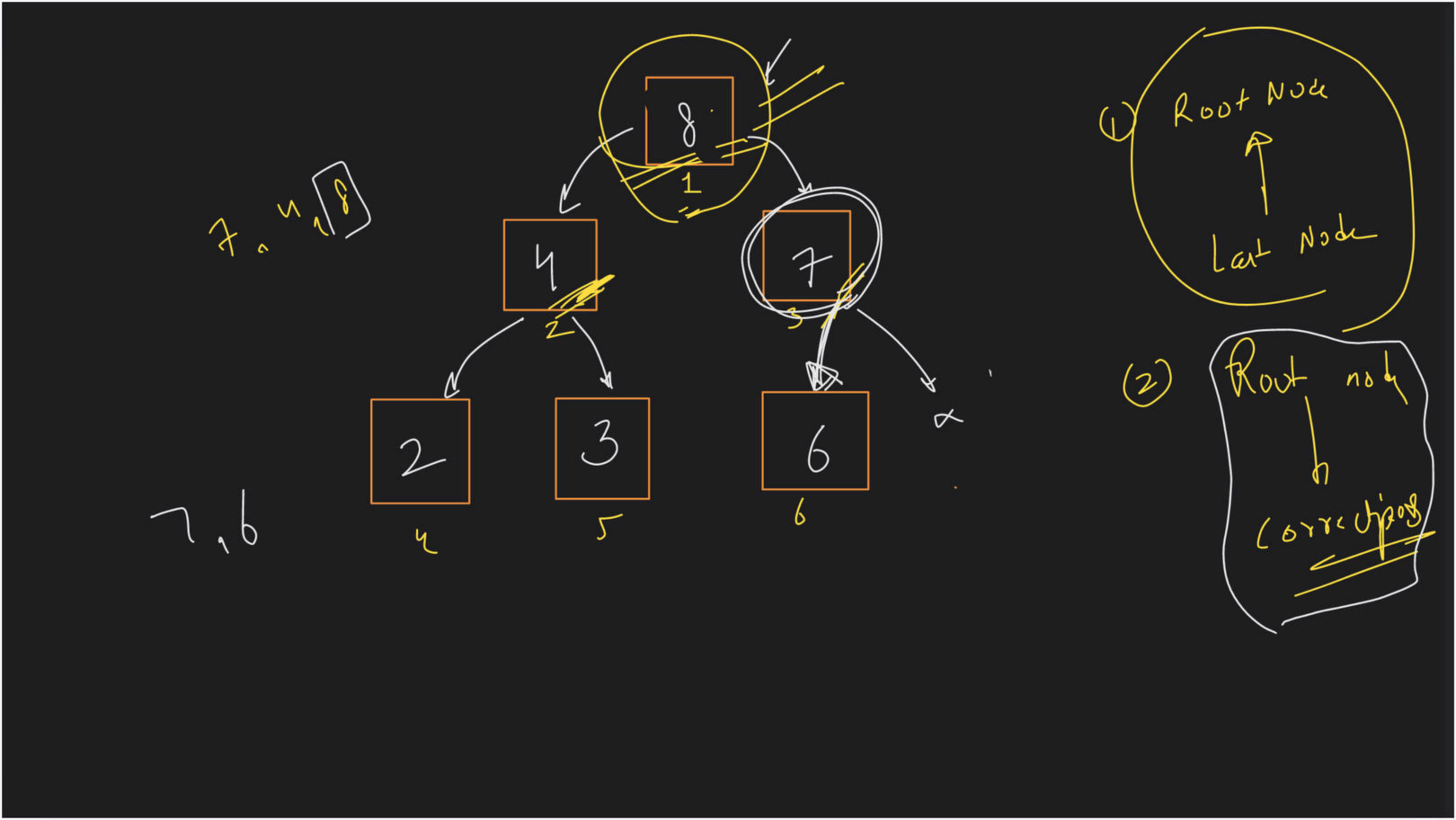
Deletion

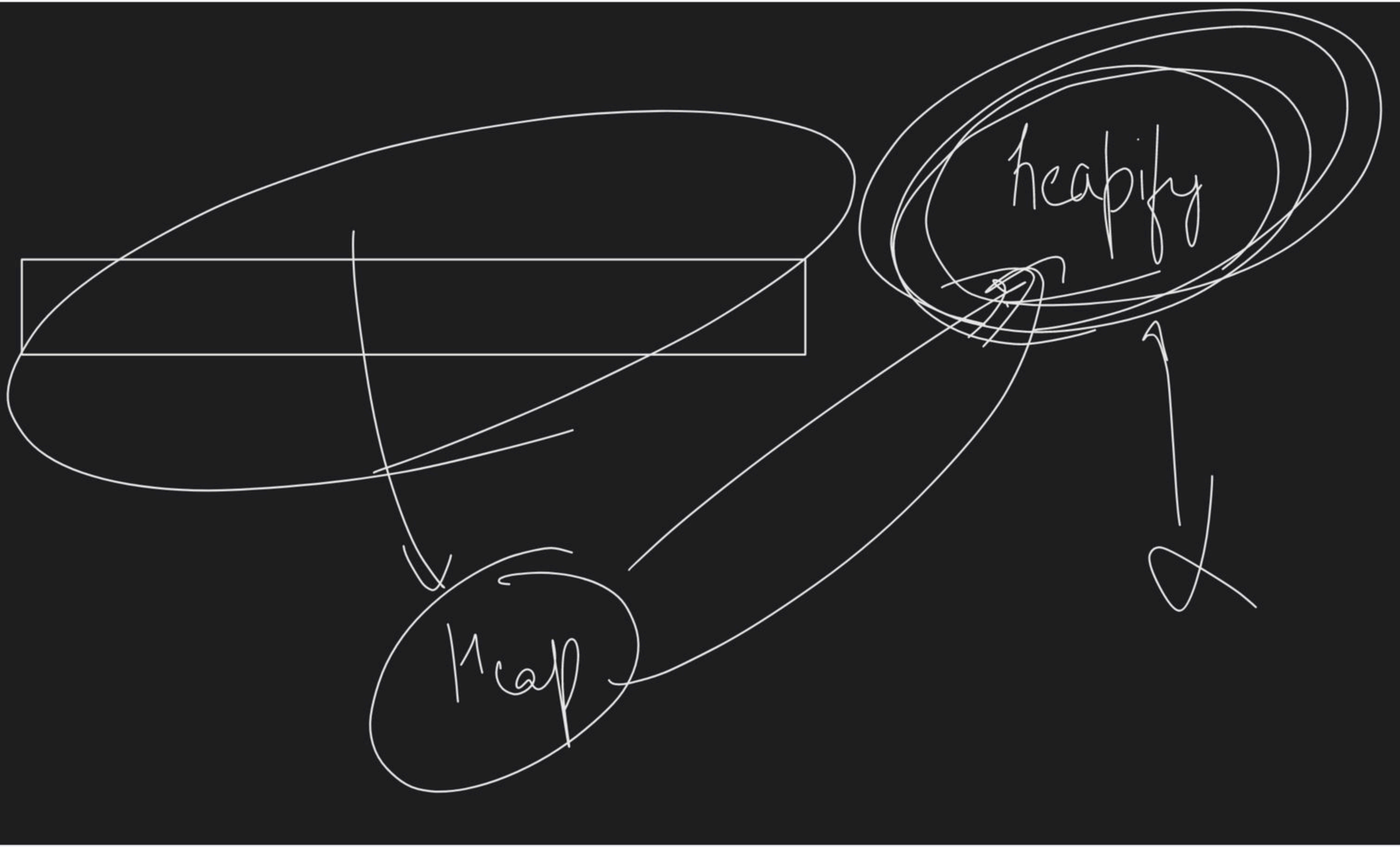
① Replace last value with root node

② Root node → correct order



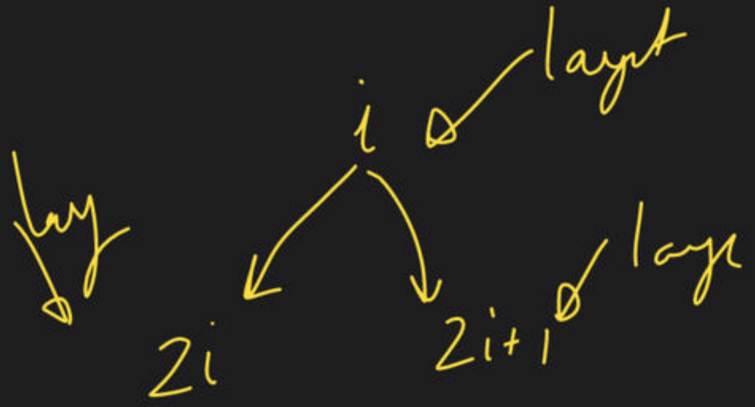

index
layer
60
layer
index
layer
50
55
index
layer
40
45
layer
53
30
20
O(1)
O(N)
T.C
O(logn)


8
4
7
2
3
6
1
2
3
4
5
6
7, 4, 8
7, 6
(1) Root Node
Last Node
(2) Root node
(correct)


heapify
Heap

Heapify

heapify (arr, n, i)



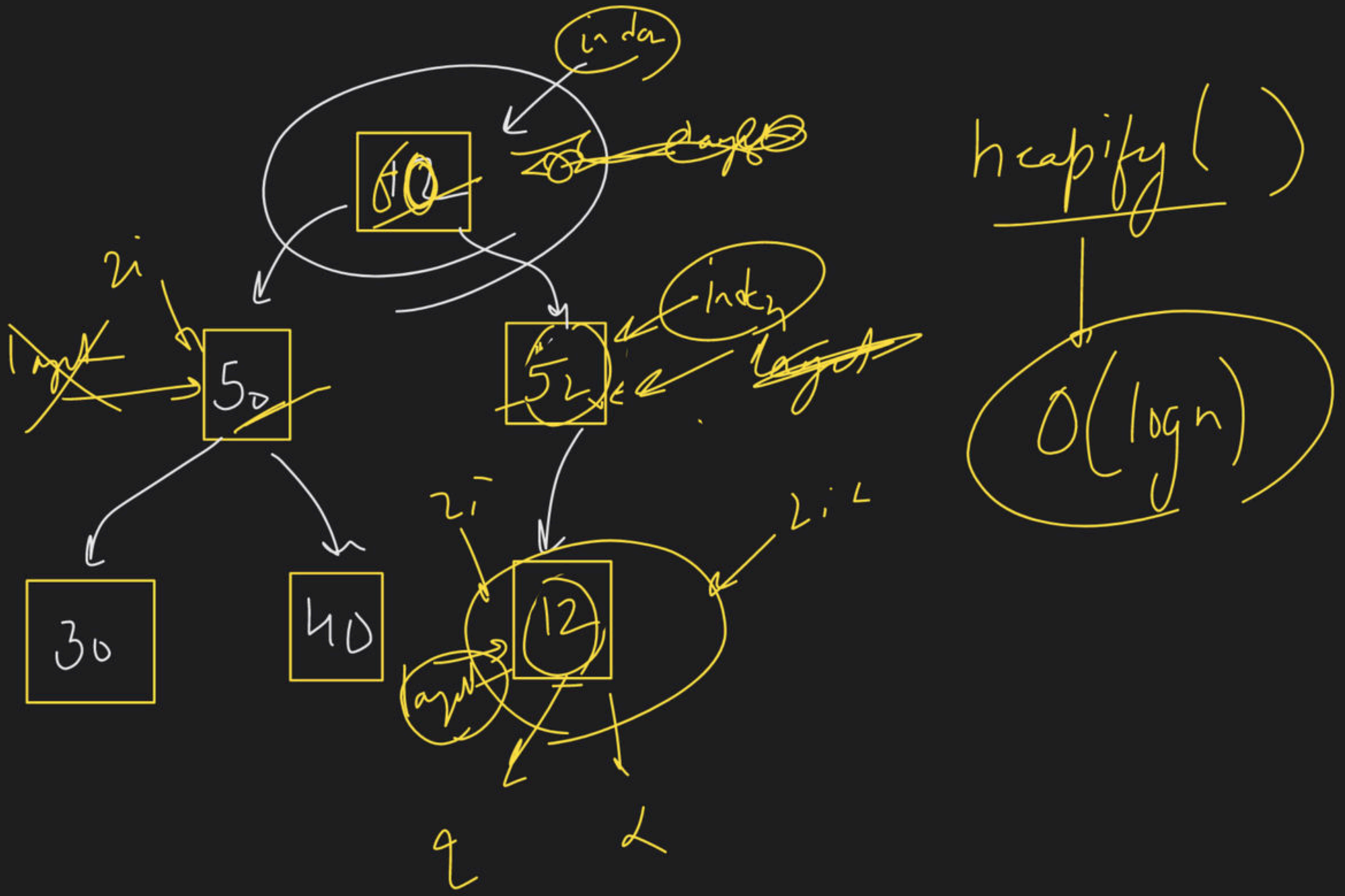
i == largest

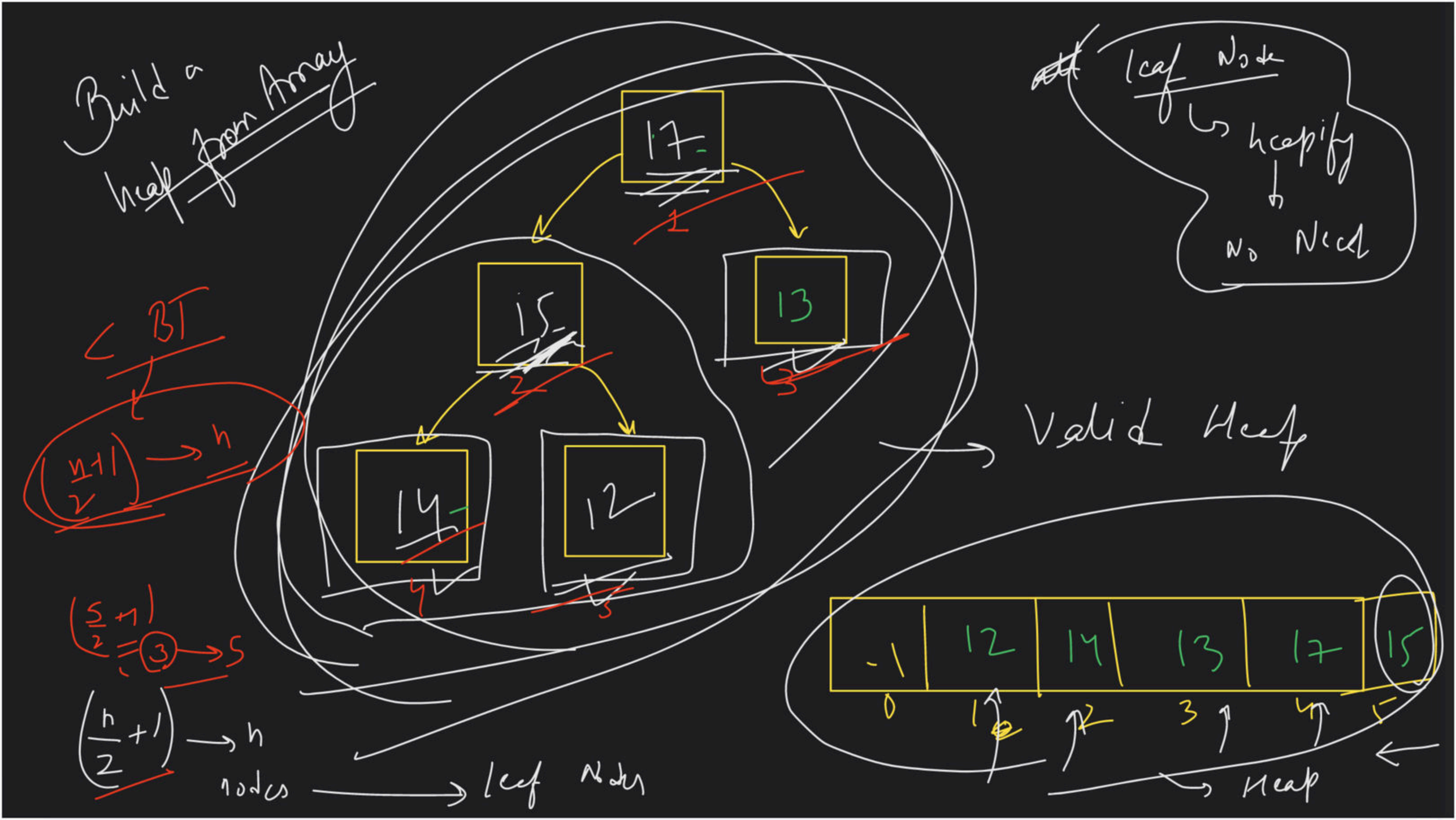
i != largest

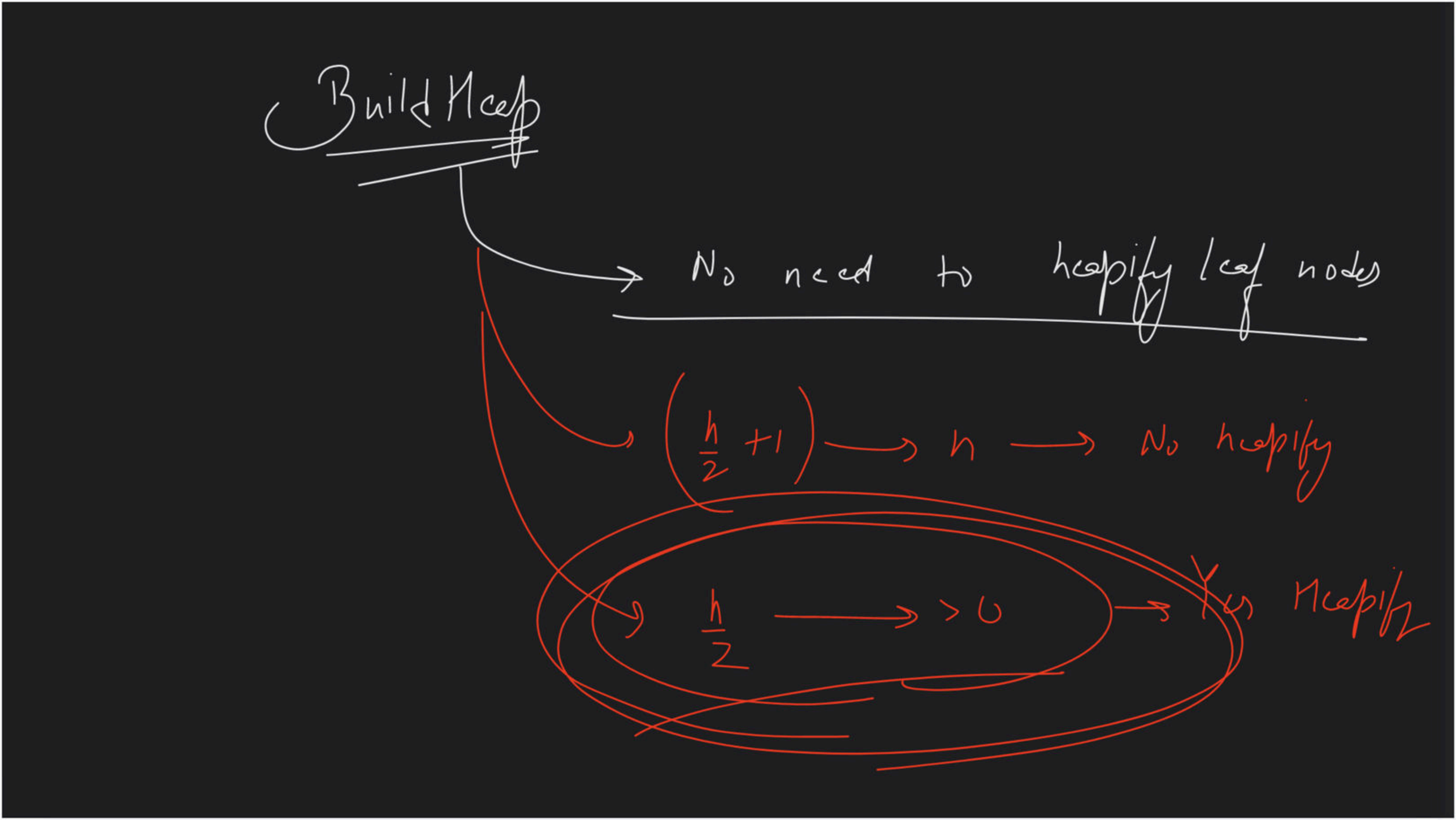
swap

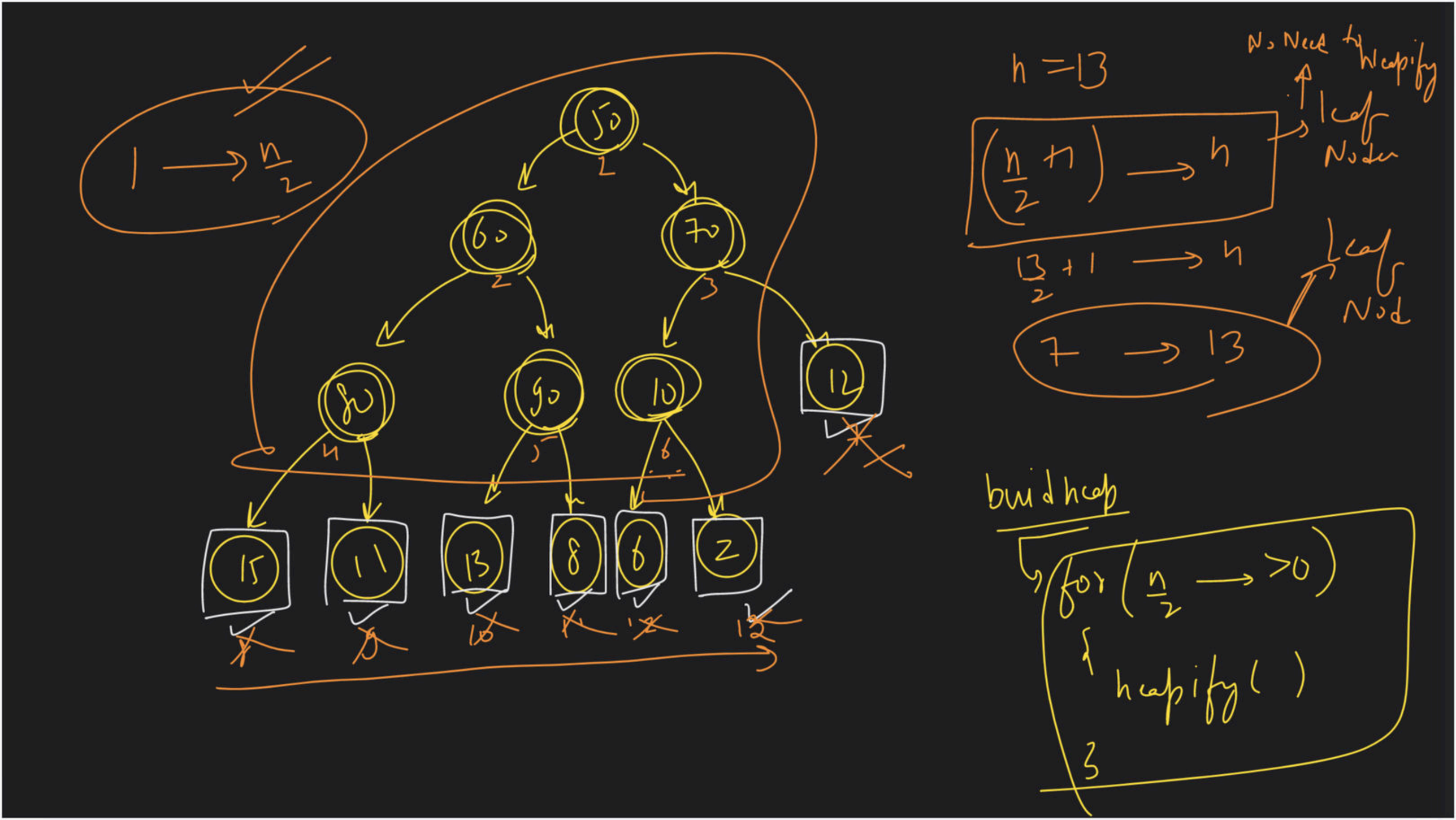
i = largest

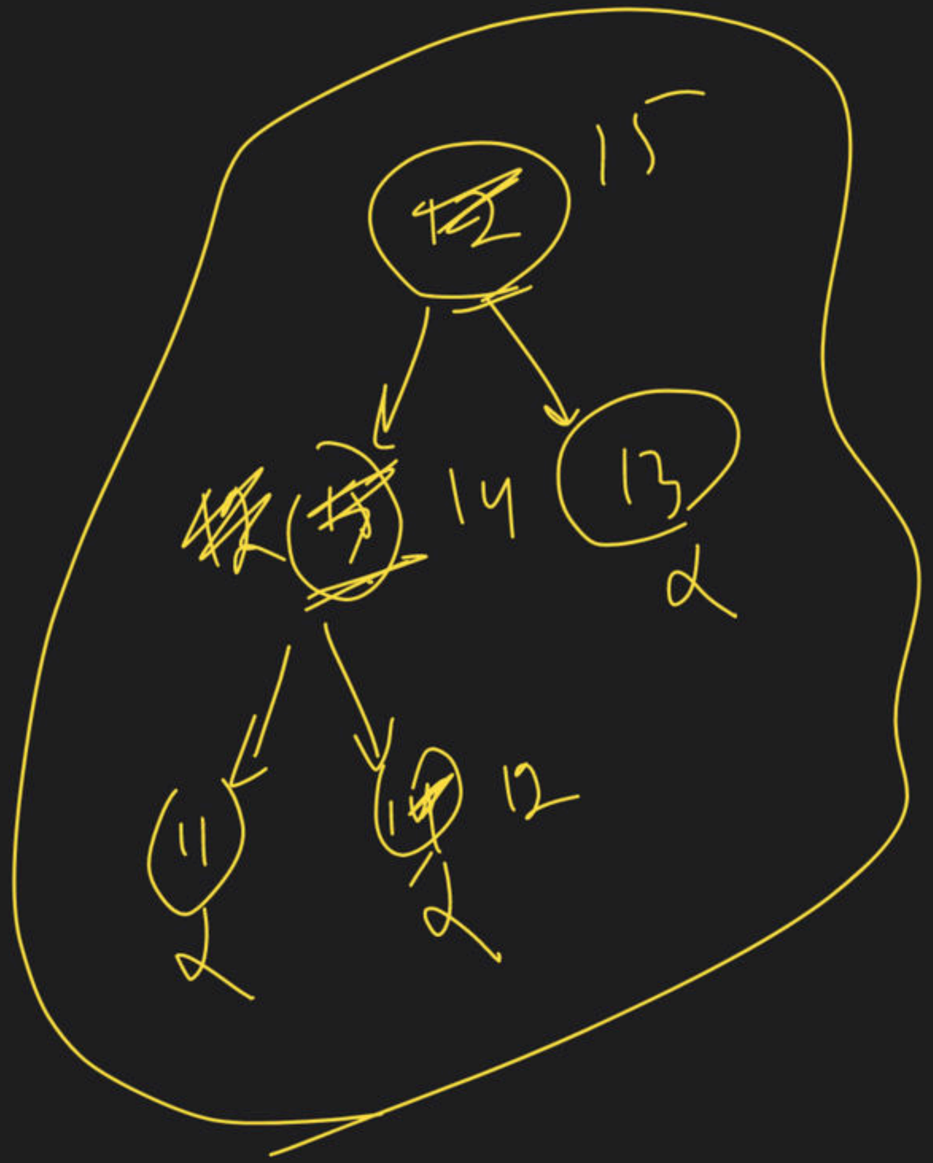
heapify

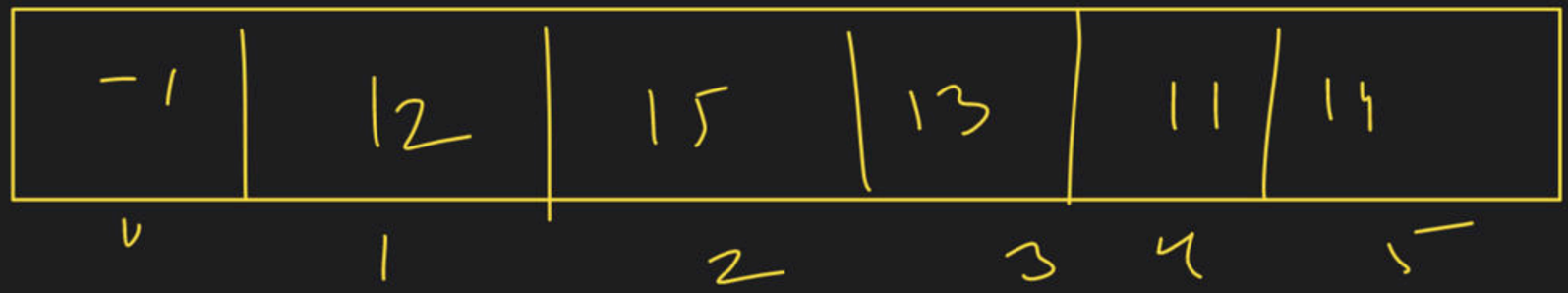

index
40
2i
50
52
index
2i
2i+1
30
40
12
left
l
r
heapify()
O(log n)

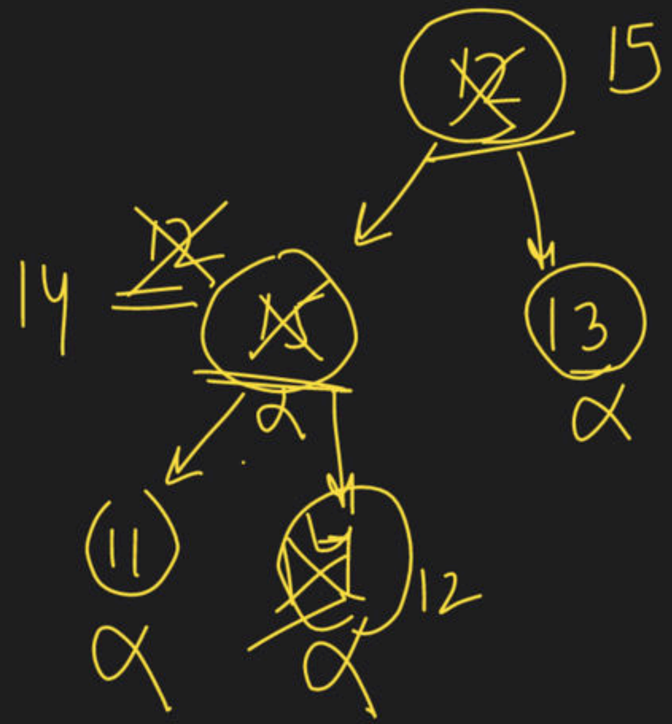

Build a heap from Array
17
15
13
14
12
Leaf Node
heapify
No Need
BT
h
Valid Heap
-1
12
14
13
17
15
0
1
2
3
4
5
heap
leaf Nodes

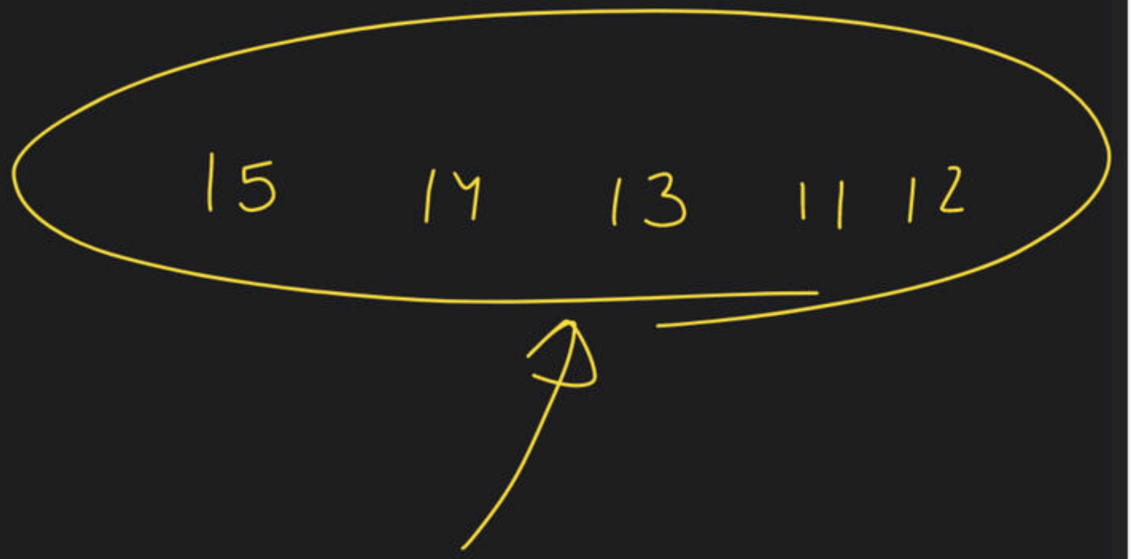

Build Heap
No need to heapify leaf nodes
$\left(\frac{n}{2}+1\right)$ → n → No heapify
$\frac{n}{2}$ → > 0 → Yes Heapify

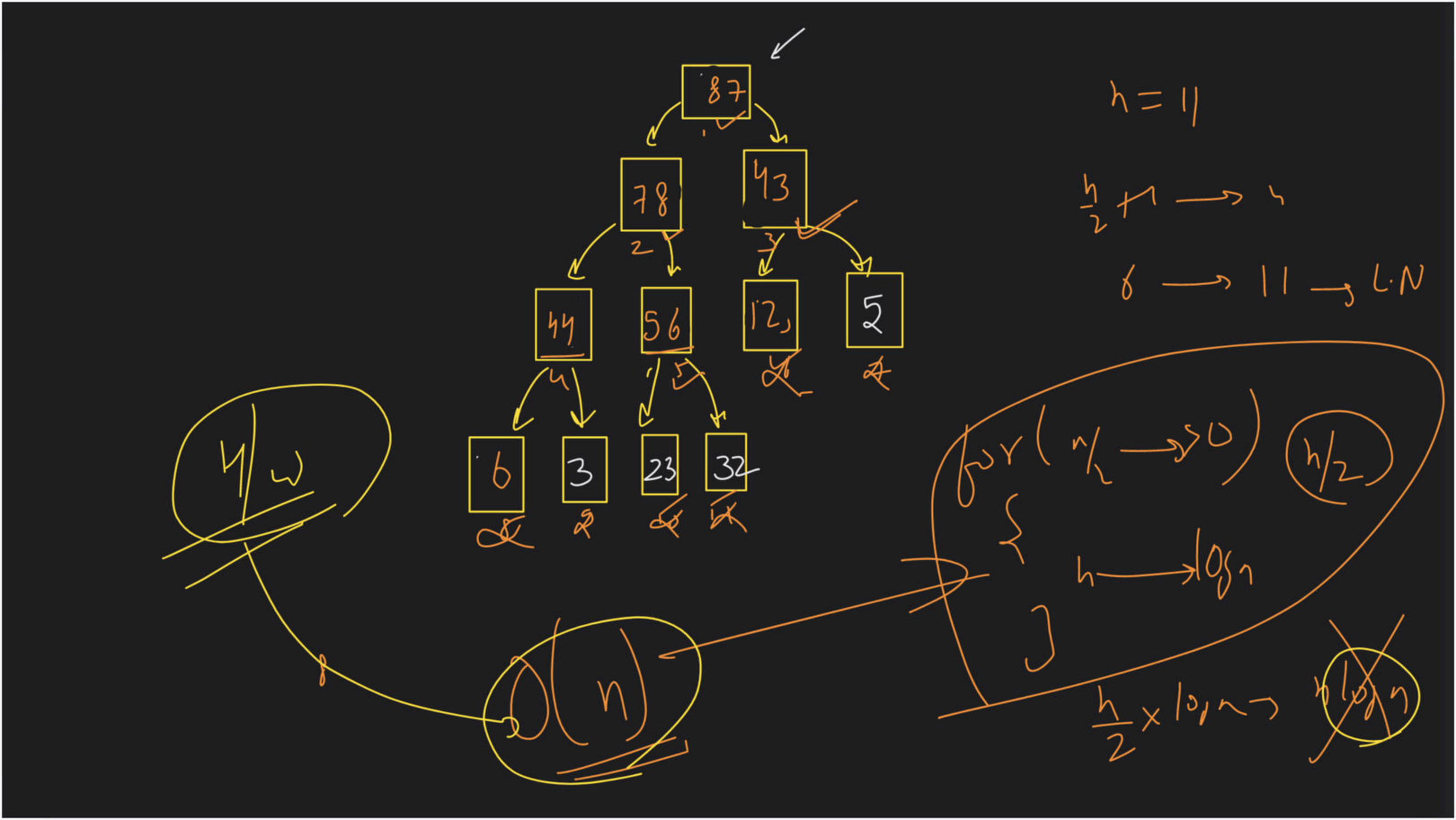

1 → n/2
50
60
70
80
90
10
12
15
11
13
8
6
2
h = 13
No Need to Heapify
Leaf Node
(n/2 + 1) → h
13/2 + 1 → h
Leaf Node
7 → 13
build heap
for (n/2 → >0)
{
heapify( )
}


15
14
13
12
11


-1
12
15
13
11
14
0
1
2
3
4
5


12
15
14
13
11
12

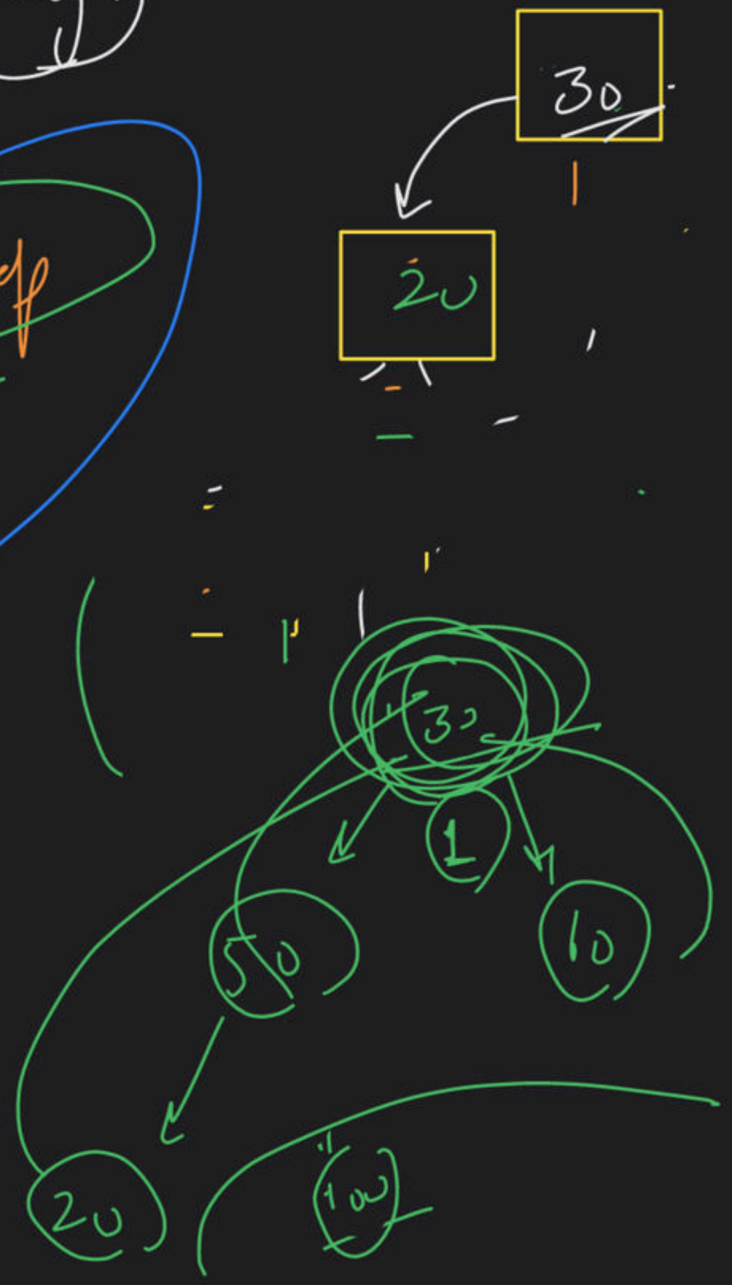

15
14
13
11
12


87
78
43
44
56
12
5
6
3
23
32
n=11
n/2+1 → n
1 → 11 → LN
for(n → 0)
{
n → log n
}
n/2
O(n)
n/2 x log n → n log n

Heap Sort

$n \log n$

1. first elem | last → swap
   size - -
2. heapify



n = 5

index

100 50 60 20 30

index

60 50 30 20 | 100

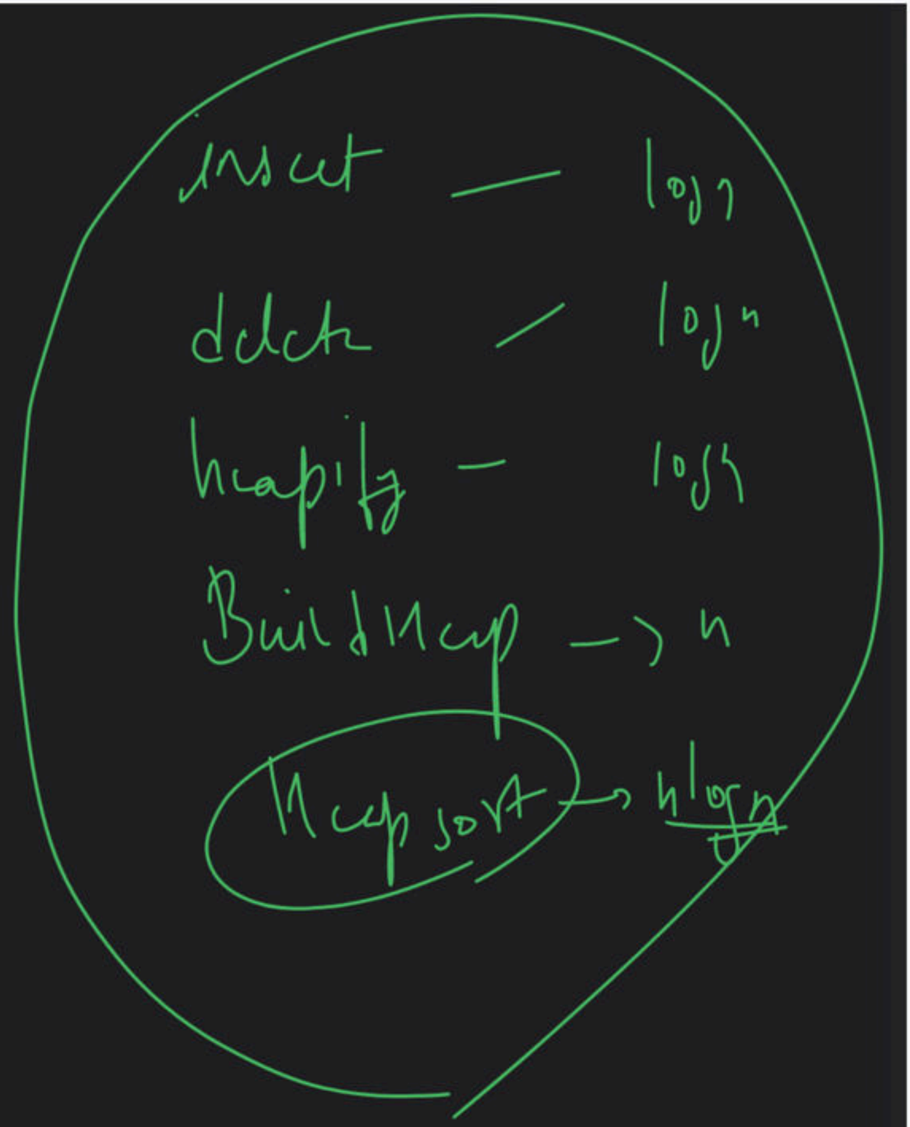
heap

50 20 30 | 60 100

30 20 | 50 60 100

20 | 30 50 60 100

20 30 50 60 100


insert — log n
delete — log n
heapify — log n
Build Heap → n
Heap sort → n log n