# [Heaps]

- Prateek Narang
- SDE III @ Google
- 150K Learners.

(Q) N students took part in contest Find out Top-K students based upon Score

scores = [80, 95, 566, 761, 300, 480, 500]

761 566 95 80

Solution

[761, 566, 500, 480, 300, 95, 80]

Top-3

O(N log N) (faster most likely) — MergeSort / Randomised Quicksort

[761, 566, 500 _ _ _ _ _ _]

Unsorted

O(NK) — Selection Sort

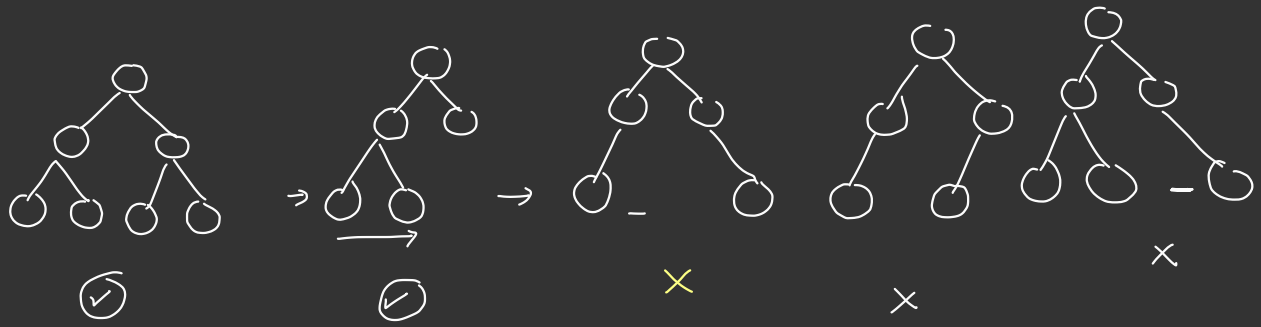
# "Heap Data Structure"

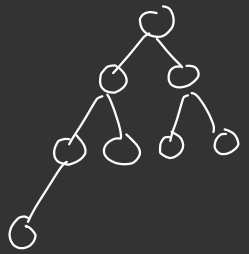
① → Complete Binary Tree

② → Heap order property

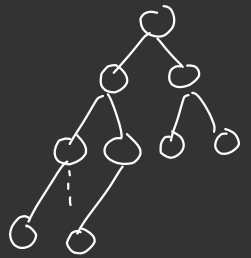
## (I) CBT

Complete Binary Tree is a tree in which all levels are completely filled except last level which may be partially filled but the filling must be in Left to Right order.

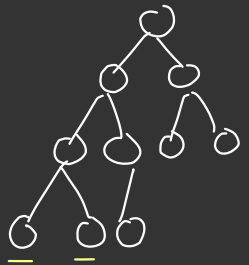




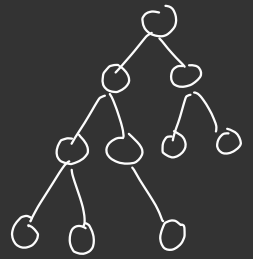
Yes



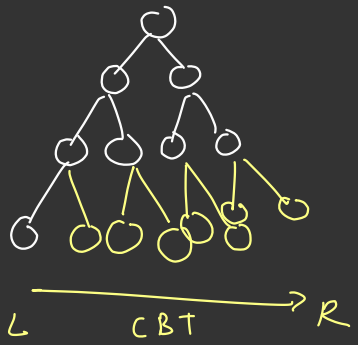
No



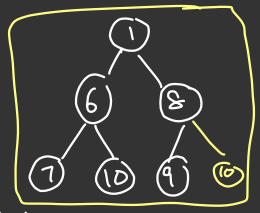
Yes



No.



## (II) Heap order property




Heap

- Min Heap
- Max Heap

- Min Heap Property

Node.data ≤ children.data

for all nodes

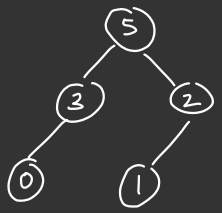
Min Heap. [ 1) CBT
2) MinHeap Prop

OR

- Max Heap Prop

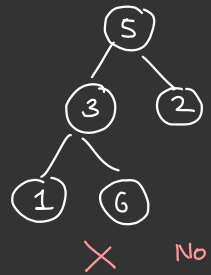
Node.data ≥ children.data

for all nodes.




Not CBT

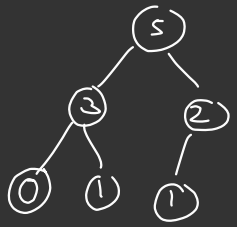
↳ Not a heap




5 ≥ 3  5 ≥ 2

3 ≥ 1  3 ≥ 6 ← Violating max heap prop

✕ No heap



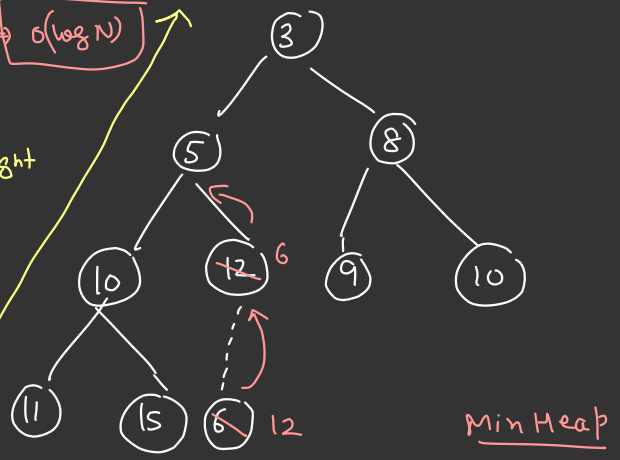

→ CBT
→ Max Heap Property
] Max Heap

# Special Methods with Heap

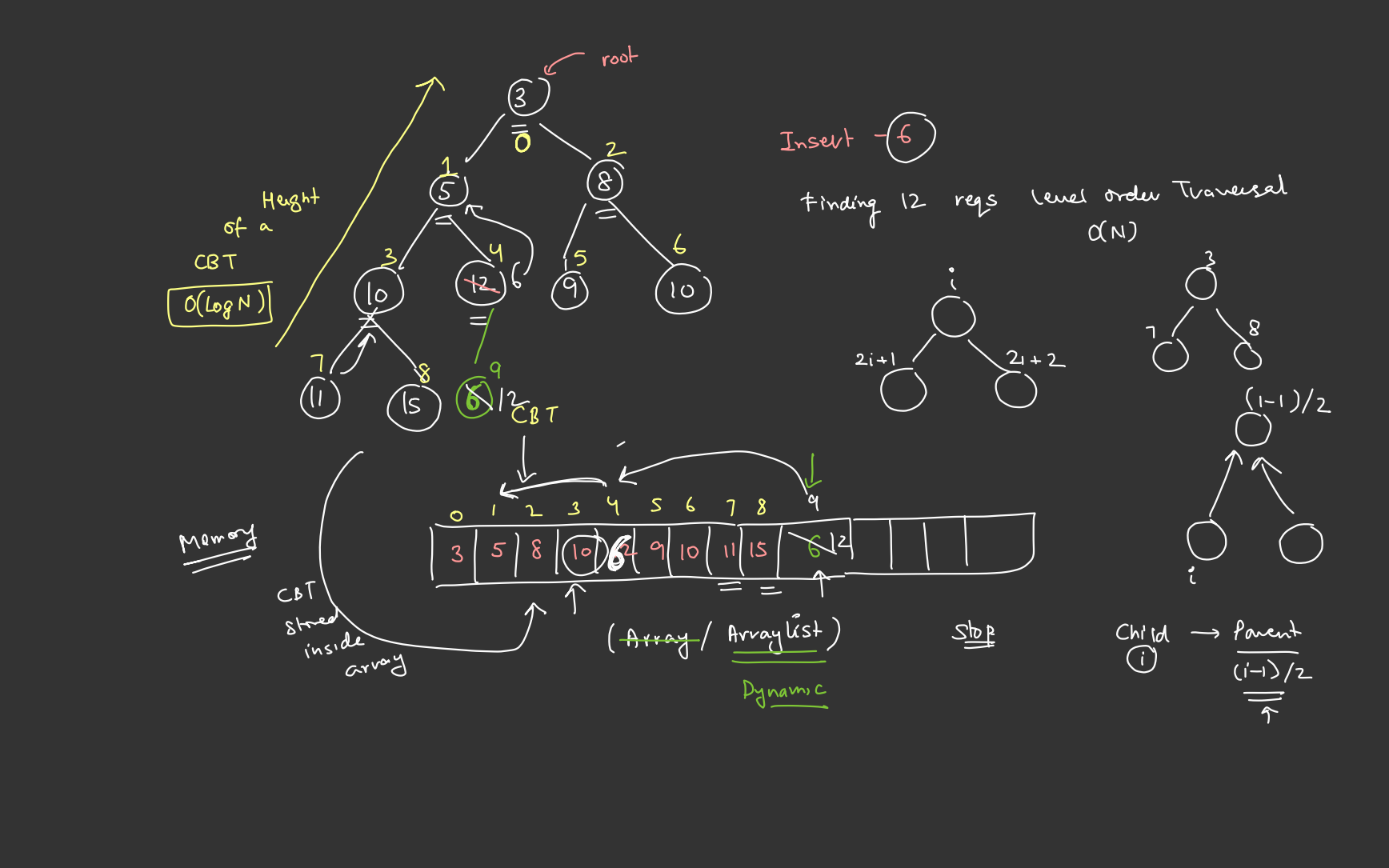
1. Insert (data) → O(LogN)
2. Remove Min() / Remove Max() → O(LogN)
3. Get Min() / Get Max()
   ↓
   O(1)

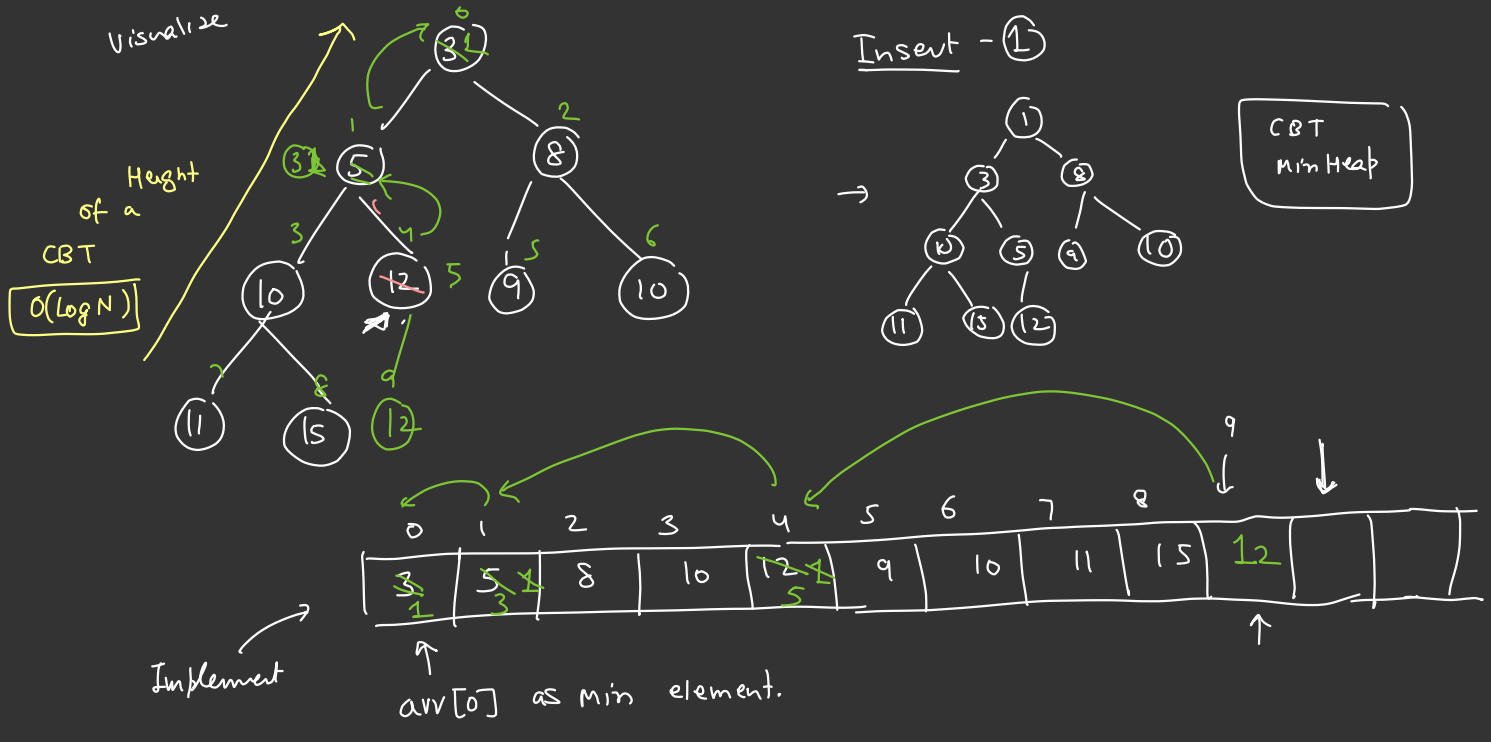
Height of a CBT → O(LogN)

Insert 6

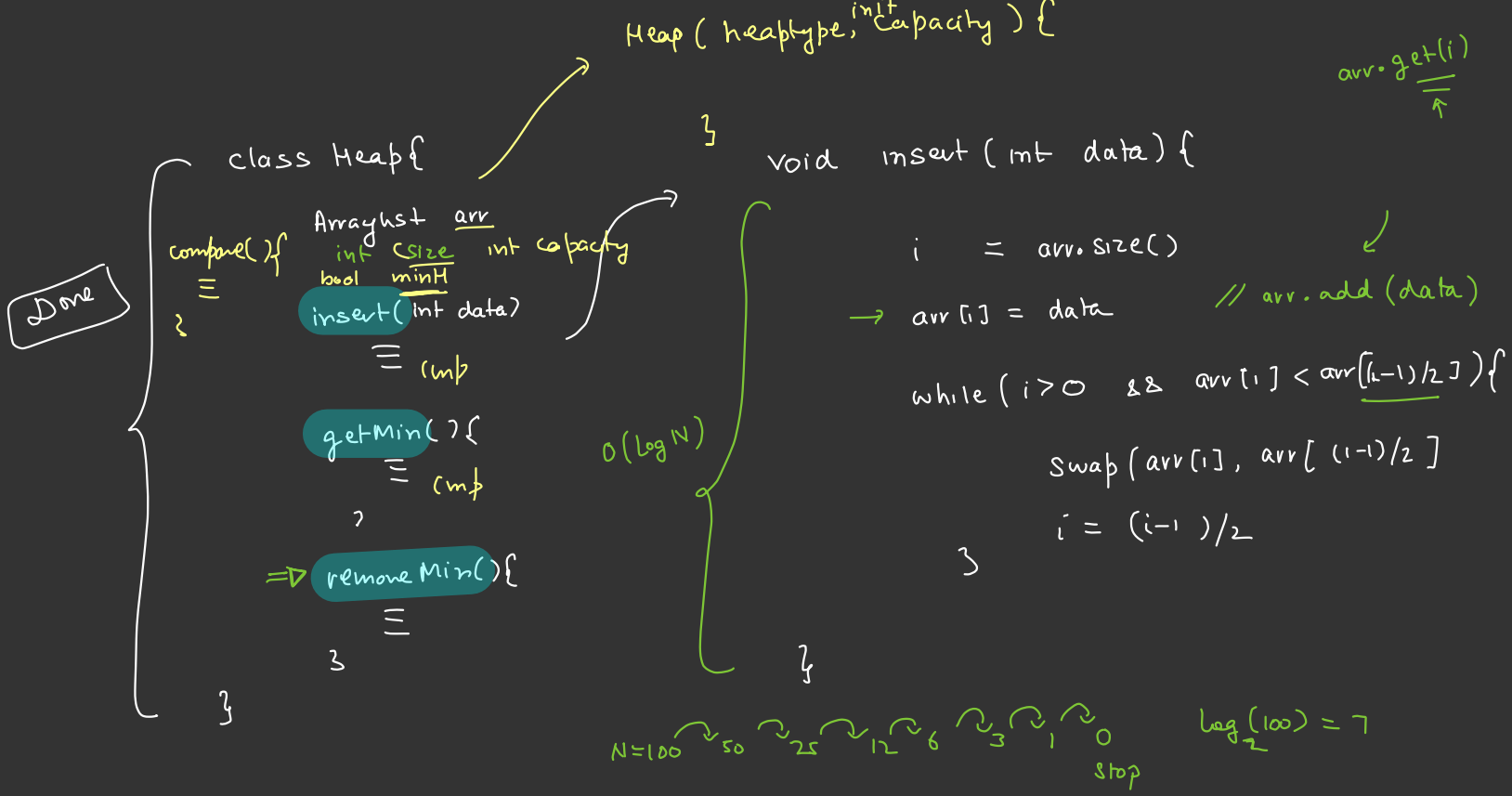


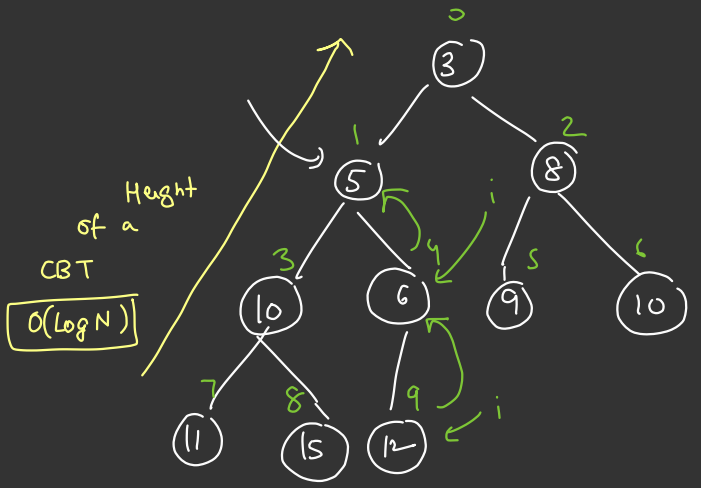

Min Heap
[Yes]


root
Height of a CBT
O(Log N)
Insert 6
Finding 12 reqs level order Traversal
O(N)
2i+1
2i+2
(i-1)/2
CBT
Memory
CBT stored inside array
(Array / ArrayList)
Dynamic
Stop
Child (i) → Parent (i-1)/2




CBT should be implemented using array / arraylist.


Heap ( heaptype, int capacity ) {
}
class Heap{
Arrayhst arr
int csize int capacity
bool minH
compare( ){
insert( int data)
cmp
getMin( ){
cmp
removeMin( ){
Done
}
Void insert ( int data) {
i = arr.size( )
arr[i] = data
// arr.add (data)
while ( i>0 && arr[i] < arr[(i-1)/2] ){
Swap ( arr[i], arr[ (i-1)/2 ]
i = (i-1 )/2
}
}
O(Log N)
arr.get(i)
N=100 50 25 12 6 3 1 0
stop
Log2(100) = 7



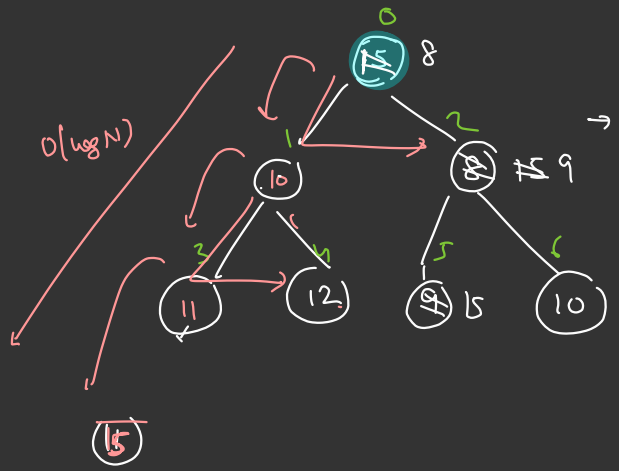

size = 9

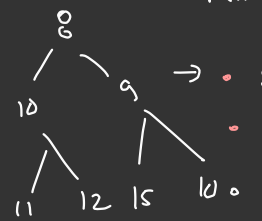
$(i-1)/2$

O(1)

```
int getMin() {
    return arr[0]
}
```

// arr.get(0)



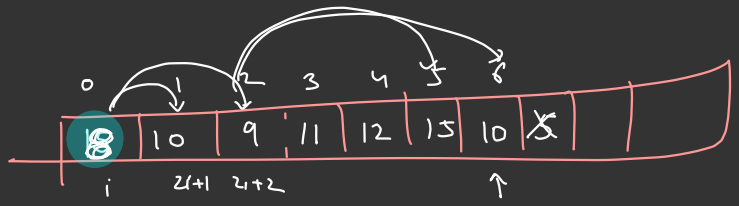




remove Min() {

- Swap first and last idx O(1)
- Remove last Node O(1)
- Fix the tree starting from Node 0 for min Heap

Heapify (0) → O(log N)

}




ArrayList<Integer>

void removeMin ( ) {

last = arr.size() - 1

Swap ( arr[0], arr[last] )

arr.remove (last)

i = 0 , minIdx = i

while (true) {

left = 2i + 1

right = 2i + 2

if (left < size && a[i] > a[left])

{ minIdx = left }

if (right < size && a[minIdx] > a[right]){

{ minIdx = right }

if (minIdx != i) {

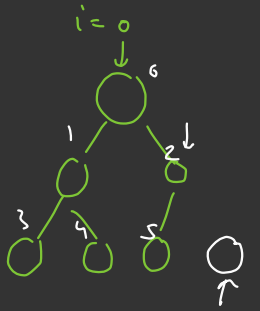
swap ( a[minIdx], a[i])

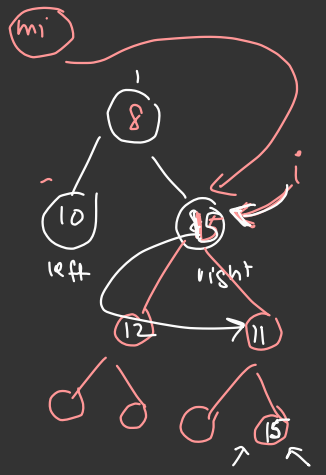
i = minIdx, ← update i after Swap.

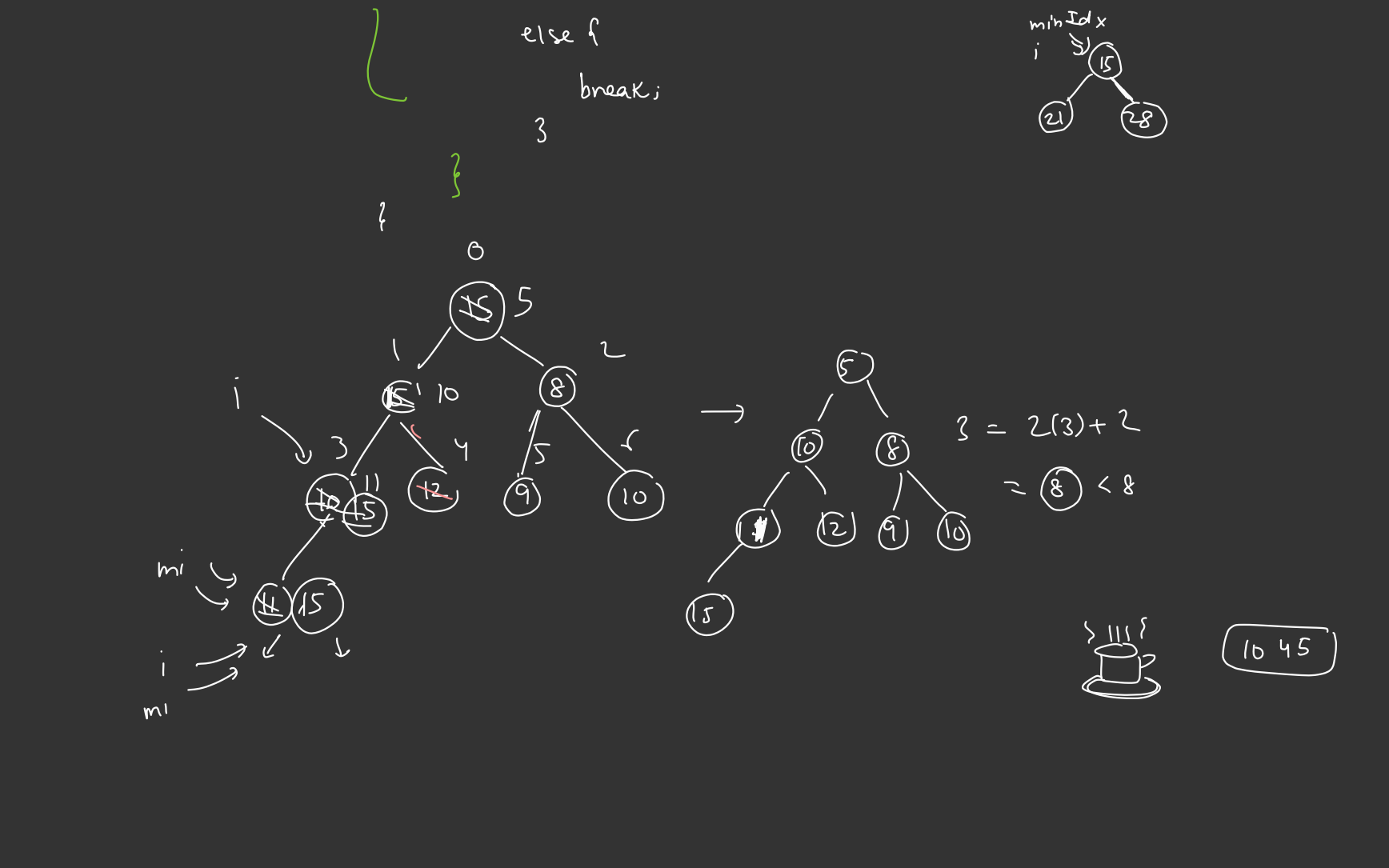
}

O(log N)

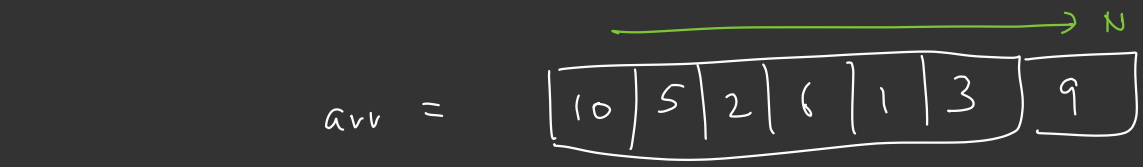
heapify




else {
break;
}
}
minIdx
i
15
21
28
0
5
1
2
i
3
4
5
6
10
11
12
9
10
mi
i
mi
3 = 2(3) + 2
≃ 8 < 8
10 45

Build a heap (inplace) from a given array)




Heap h → additional storage (array list)

```
for(every x. arr){
    h.insert(x)
}
```

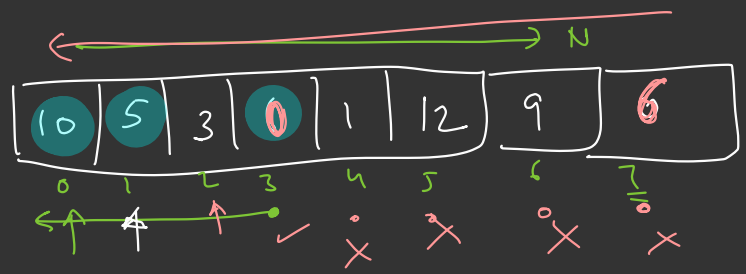
log N

O(N log N)

TRICK:

Convert a given array into heap in O(N) time.

arr =

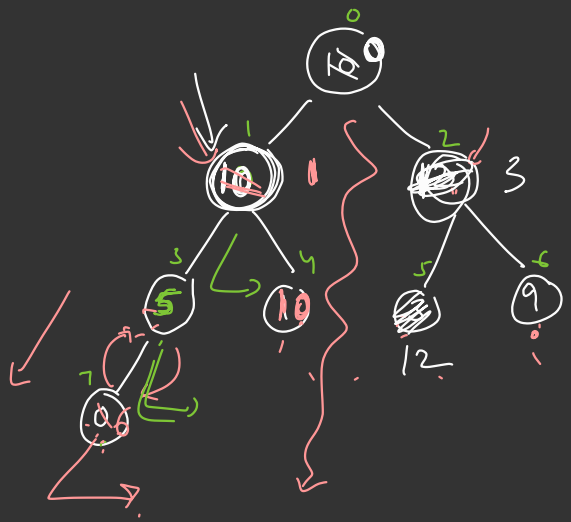



⇒ Last non-leaf node = $\frac{(\text{lastIdx} - 1)}{2}$

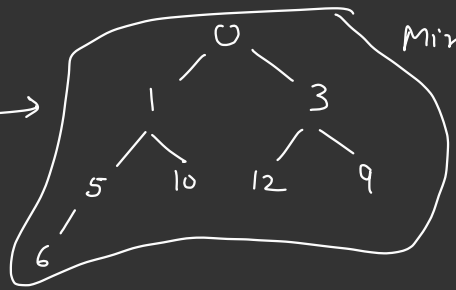
3 units O(H)

2 unit time for each node at this level

1 unit time " " " -

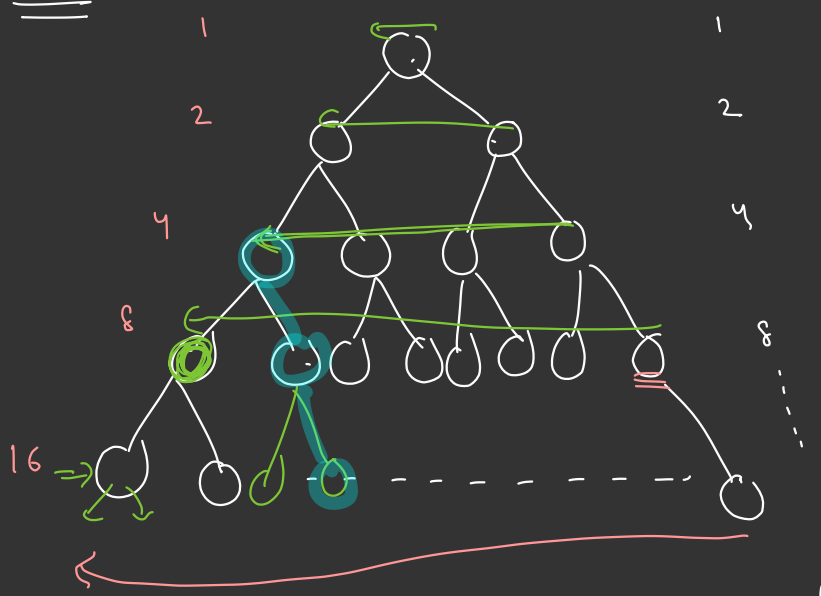



=




Min Heap.

Proof




$\frac{1}{16}$ Node $\times 3$

$\frac{N}{8}$ --- $\times 2$

$\frac{N}{4}$ Nodes $\times 1$

$\frac{N}{2}$ Nodes $\times 0$

total $N = 31$

Total work

$$\Rightarrow S = \frac{N}{4} + \frac{N}{8} \times 2 + \frac{N}{16} \times 3 + \frac{N}{32} \times 4 + \cdots$$

$$= \frac{N}{2^2} + \frac{N}{2^3} \times 2 + \frac{N}{2^4} \times 3 + \frac{N}{2^5} \times 4 + \cdots$$

$$= \frac{N}{2}\left[\frac{1}{2} + \frac{2}{2^2} + \frac{3}{2^3} + \frac{4}{2^4} + \cdots\cdots\right]$$

AGP

$$= \frac{N}{2}(2)$$

School way — class 12

$$S = \frac{1}{2} + \left(\frac{2}{2^2} - \frac{1}{2^2}\right) + \left(\frac{3}{2^3} - \frac{2}{2^3}\right) + \left(\frac{4}{2^4} - \frac{3}{2^4}\right) + \cdots\cdots \text{ to } \infty \text{ terms}$$

$$\frac{S}{2} = \qquad\qquad + \left(\frac{1}{2^2}\right) + \left(\frac{1}{2^3}\right) + \cdots\cdots \text{ to } \infty \text{ terms}$$

$$S - \frac{S}{2} = \frac{1}{2} + \left(\frac{1}{2^2}\right) + \left(\frac{1}{2^3}\right) + \left(\frac{1}{2^4}\right) + \cdots\cdots \text{ to } \infty \text{ terms.}$$

$$= \frac{1}{2}\left(1 - \frac{1}{2}\right) = 1$$

$$S = \frac{a}{1-r}$$

$$\left\lfloor \frac{S}{2} \right\rfloor = 1$$

$$\Rightarrow \boxed{S = 2}$$

firstNonLeafNode = (lastIdx - 1) / 2

```
for (i = firstNonLeafNode, i >= 0; i--) {
    heapify (arr, i)
}
```

↳ array into heap inplace

O(N)

# Assignment — Merge K-sorted arrays into single sorted array

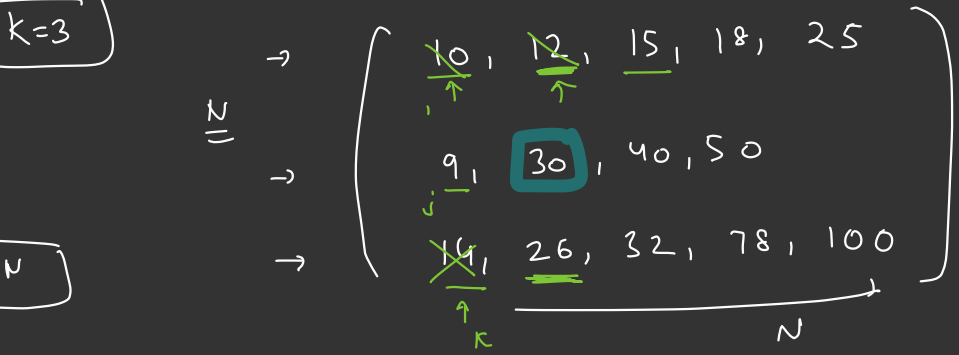
K=3

$E = KN$

$E \log E$

$KN \log KN$

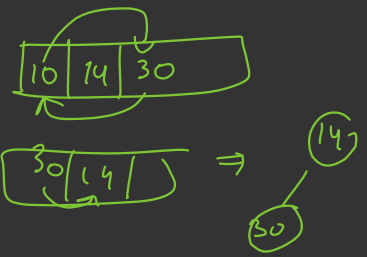
$N = ?$



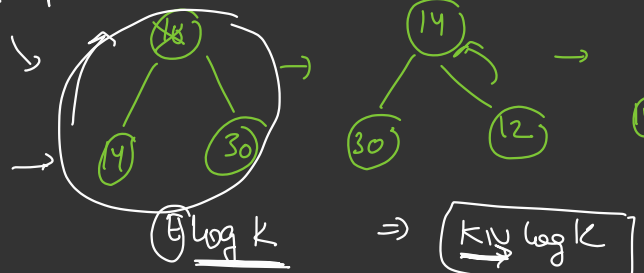

9, 10, 12, 14, 15, ...

Hunt

Heap of size K




Heap




$E \log K \Rightarrow KN \log K$

Insert / Remove