# CS & IT ENGINEERING


## Algorithms

### Miscellaneous

Lecture No.- 02


By- Aditya Jain sir

# Recap of Previous Lecture

Topic

Topic

Heaps Intro

# Topics to be Covered

- Topic
- Topic
- Topic

Heaps

Questions

# About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 12,000+ students & working professions in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

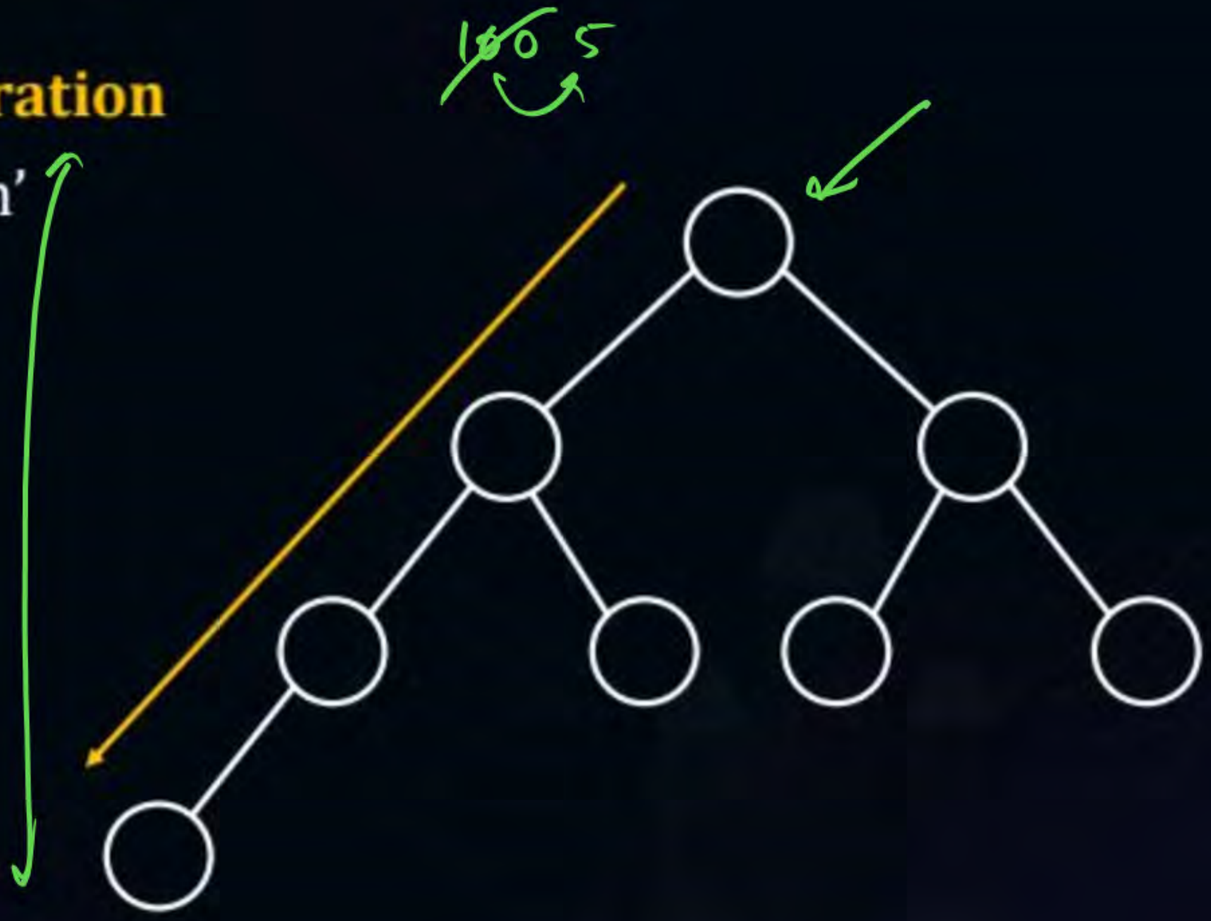

Telegram Link for Aditya Jain sir: https://t.me/AdityaSir_PW

# Topic : Miscellaneous

## 1. Increase / Decrease key operation

Heap ⟶ 'n'

TC ⇒ $O(\log_2 n)$



# Topic : Miscellaneous

V. V. Imp

Root

## Delete Operation in a Heap:

- For a max-Heap ⟶ maximum element.
- For a min-Heap ⟶ minimum element.

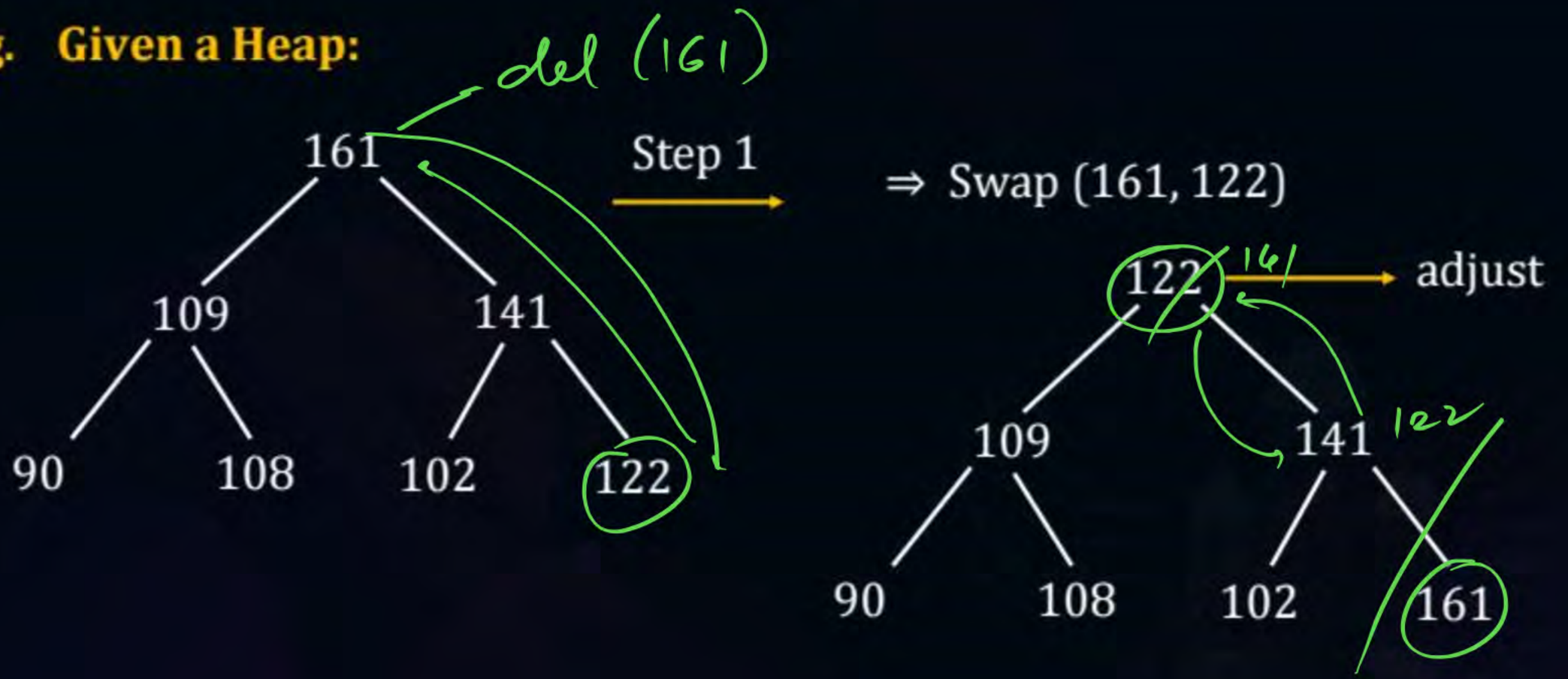
} The root is getting deleted

Steps:

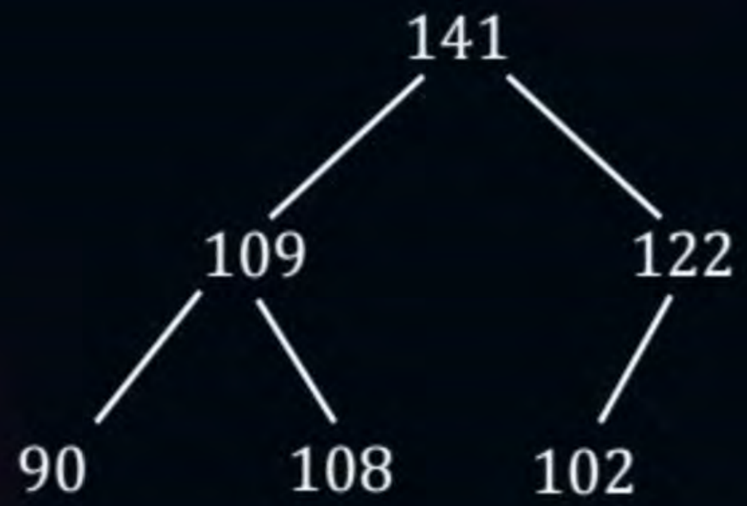
1. Swap the root with the last element
   swap (A[1], A[n])
2. Remove A[n]
3. Adjust A[1]

# Topic : Miscellaneous

**E.g. Given a Heap:**



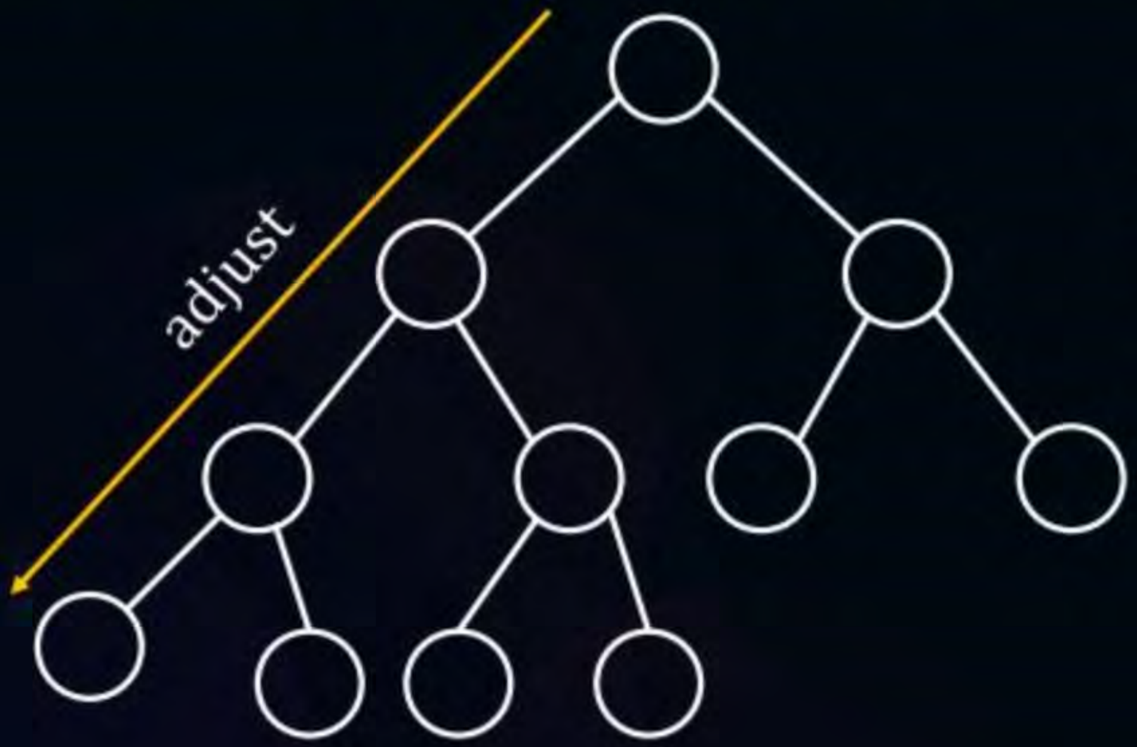
Final Heap after deleting 161




Valid Heap

# Topic : Miscellaneous

## Time complexity of a Delete Operation:




After removing you have to adjust the new root to make sure its back to a valid Heap.

TC: $O(\log_2 n)$

**[MCQ]**

#Q. Which Array Representation is a valid Binary Max-Heap.

A <25, 12, 16, 13, 10, 8, 14>

B <25, 14, 16, 13, 10, 8, 12>

C <25, 14, 13, 16, 10, 8, 12>

D <25, 14, 12, 13, 10, 8, 16>

[MCQ]

#Q. Which Array Representation is a valid Binary Max-Heap.

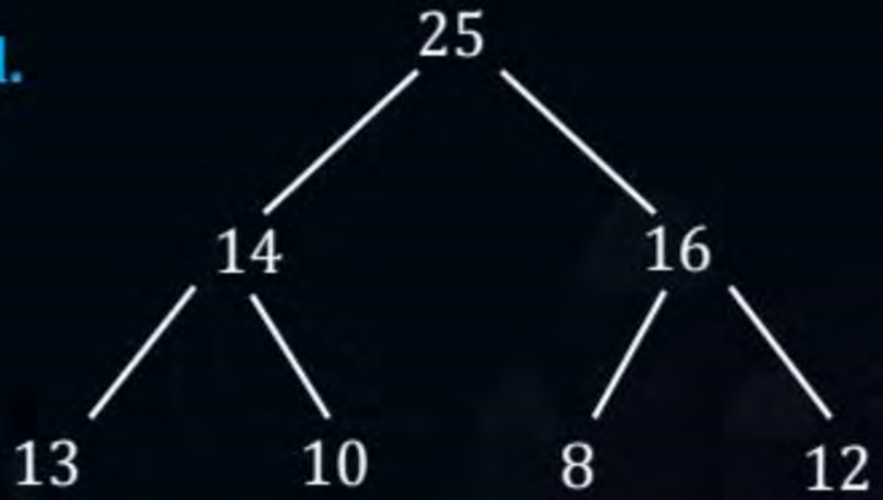
A <25, 12, 16, 13, 10, 8, 14>

B <25, 14, 16, 13, 10, 8, 12>

C <25, 14, 13, 16, 10, 8, 12>

D <25, 14, 12, 13, 10, 8, 16>

Sol.



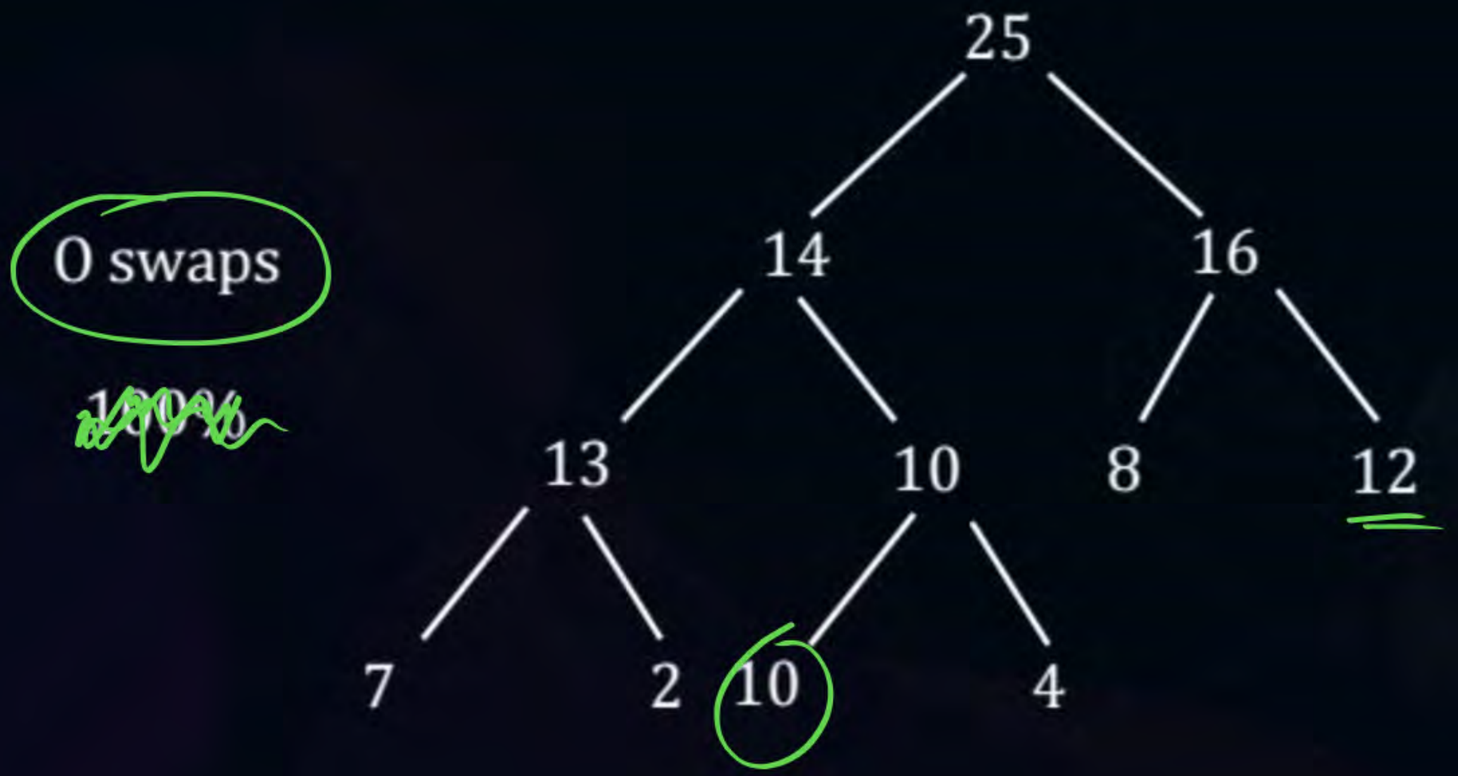
**[NAT]**

#Q. To the valid Heap of Previous Question insert elements <7 2 10 4>. Indicate the resultant Heap in Array
How many total swaps after insertion?

**[NAT]**

#Q. To the valid Heap of Previous Question insert elements <7 2 10 4>. Indicate the resultant Heap in Array
How many total swaps after insertion?

**Sol.**



[MCQ]

#Q. Which one is valid ~~3-ary~~ ternary Maximum Heap Array representation

A <1, 3, 5, 6, 8, 9>

B <9, 6, 3, 1, 8, 5>

C <9, 3, 6, 8, 5, 1>

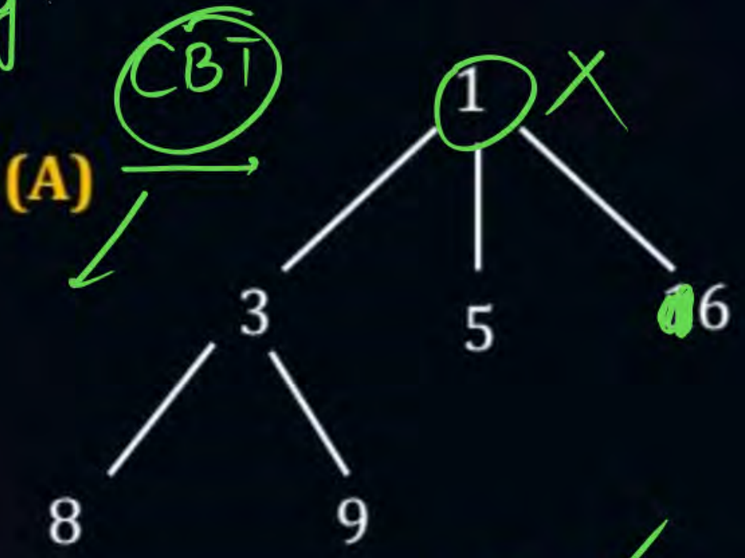
D <9, 5, 6, 8, 3, 1>

[MCQ]

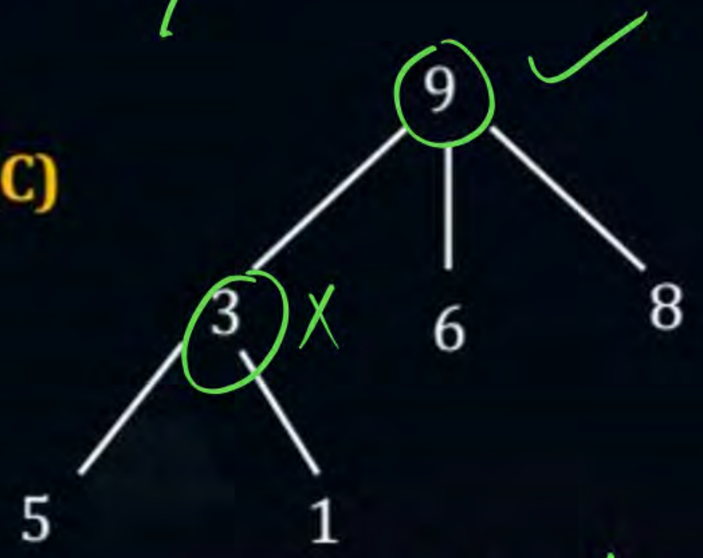
#Q. Which one is valid ~~3-ary~~ ternary Maximum Heap Array representation

Sol.

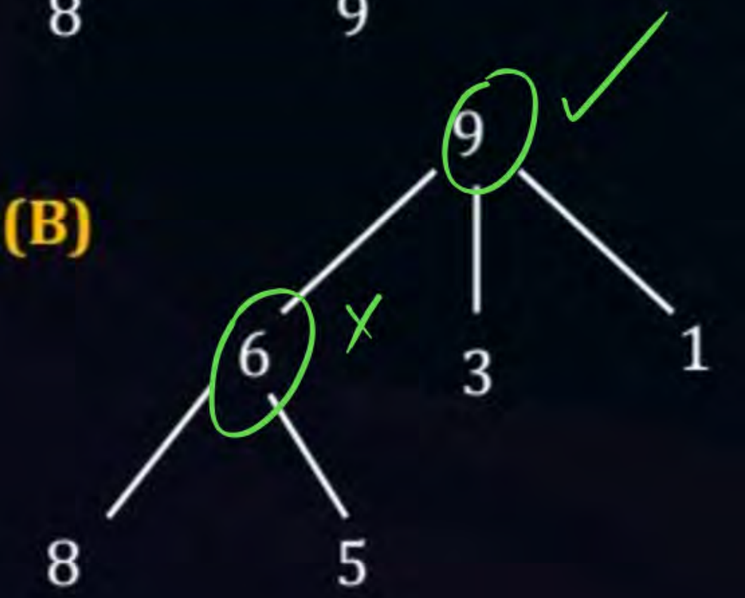
(A)



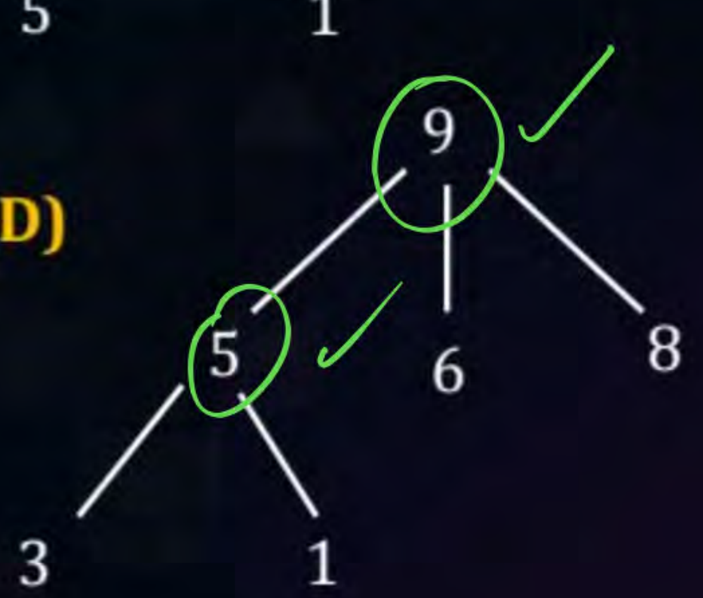

(C)




(B)




(D)




A <1, 3, 5, 6, 8, 9>

B <9, 6, 3, 1, 8, 5>

C <9, 3, 6, 8, 5, 1>

D <9, 5, 6, 8, 3, 1>

Valid is (D) <9, 5, 6, 8, 3, 1>

# Topic : Miscellaneous

**Sol.**

```
procedure HEAPSORT (A, n)
// A(1 : n) contains n elements to be sorted.//
1. call HEAPIFY (A, n)
2. for i ← n to 2 by – 1 do
        t ← A(i); A(i) ← A(1); A(i) ← t
        call ADJUST, (A, 1, i – 1)
        repeat
end HEAPSORT
```

2 → n

I/P A[1 . . . n]

↓

Heapify

↓

Heap

n → 2

Swap (a[i], a[1])

```
Swap (a, b)
{
   t = a
   a = b
   b = t
}
```

# Topic : Miscellaneous

**Explanation:**

CBT

Root → always the max element of max-Heap.

$2^{nd}$ max of original Heap

Max1, max2, max3

# Topic : Miscellaneous

**Logic:**

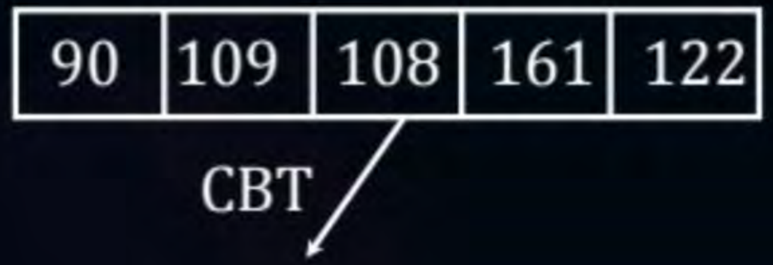
Every time a root node is deleted (max element), the delete operation will re-adjust the new root such that the new maximum (max among remaining elements) becomes the root node.

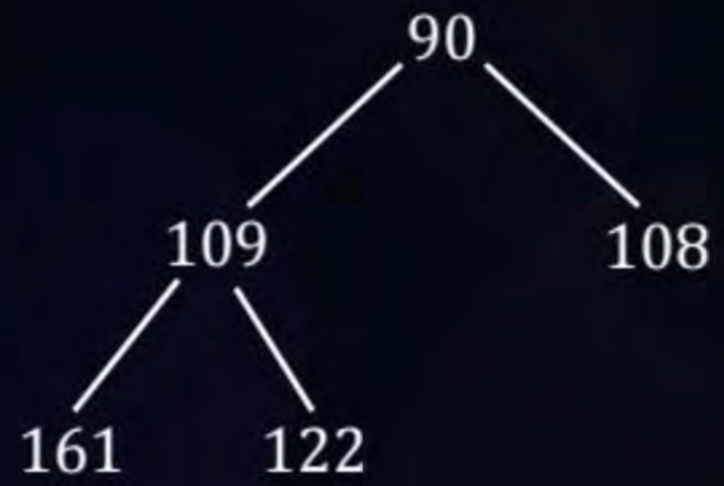
(Keep repeating this process).

# Topic : Miscellaneous

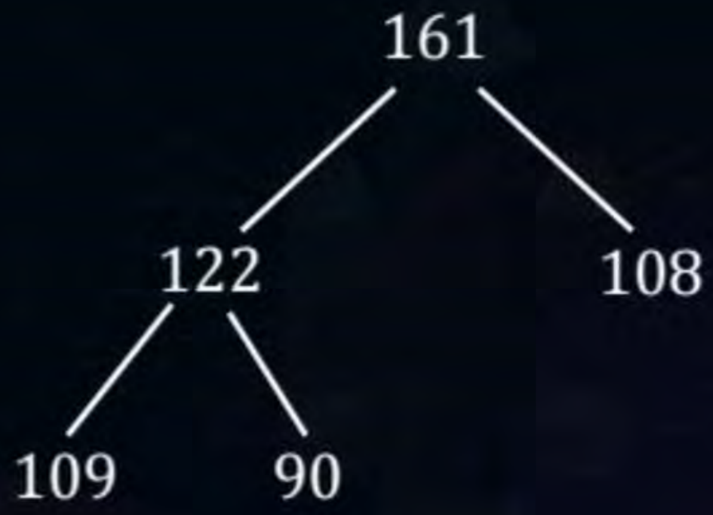
**Dry – Run:**

Sort given array in ascending order using Heap sort.




Heapify

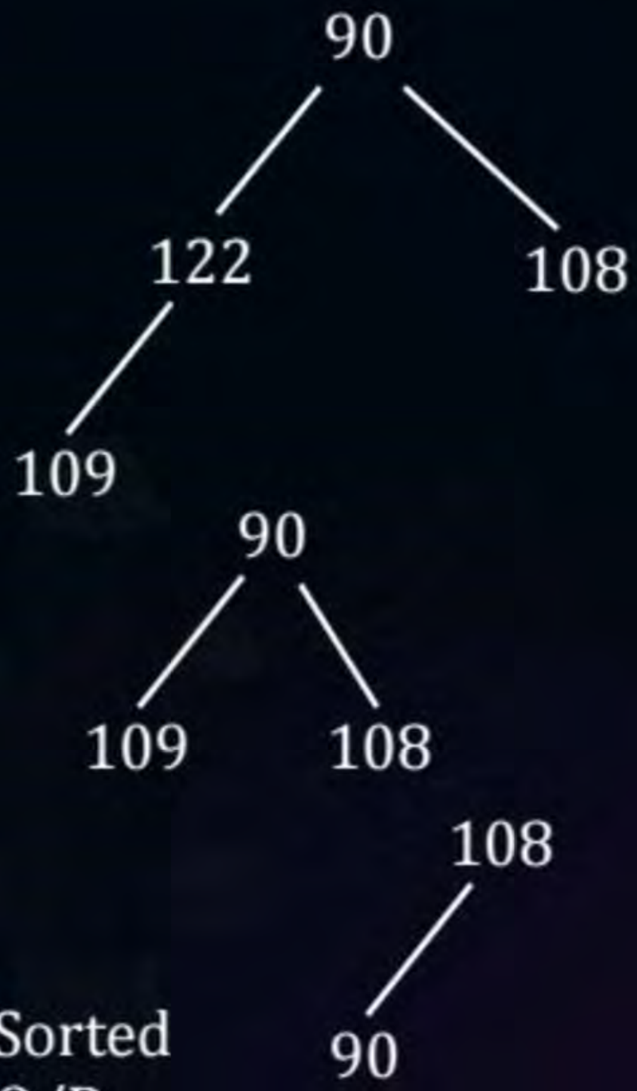


# Topic : Miscellaneous

**Step by step:**

| | | | | | |
|---|---|---|---|---|---|
| Given | 90 | 109 | 108 | 161 | 122 |
| Heapify | 161 | 122 | 108 | 109 | 90 |
| | 90 | 122 | 108 | 109 | 161 |
| Pass1 O/P: | 122 | 109 | 108 | 90 | 161 |
| | 90 | 109 | 108 | 122 | 161 |
| Pass2 O/P: | 109 | 90 | 108 | 122 | 161 |
| | 90 | 108 | 109 | 122 | 161 |
| Pass3 O/P: | 108 | 90 | 109 | 122 | 161 |
| Pass4 O/P: | 90 | 108 | 109 | 122 | 161 |

→ Sorted O/P



# Topic : Miscellaneous

**Time complexity analysis:**

TC $\Rightarrow$ $O(n + n \times \log_2 n)$

$= O(n \log_2 n) \rightarrow$ every case

$O(n \log_2 n)$ (every case)

# Topic : Miscellaneous

**Imp. Points:**

1. Space Complexity: O(1) → inplace
2. Not Stable

**[MCQ]**

#Q. Level order traversal of a binary max heap generates: <10, 8, 5, 3, 2>. To this Heap Insert: <1 and 7>; What is the resultant level order Traversal

A 10, 8, 7, 3, 5, 2, 1

B 10, 8, 7, 3, 2, 1, 5

C 10, 7, 8, 3, 5, 1, 2

D 10, 7, 8, 5, 3, 2, 1

[MCQ]

#Q. Level order traversal of a binary max heap generates: <10, 8, 5, 3, 2>. To this Heap Insert: <1 and 7>; What is the resultant level order Traversal

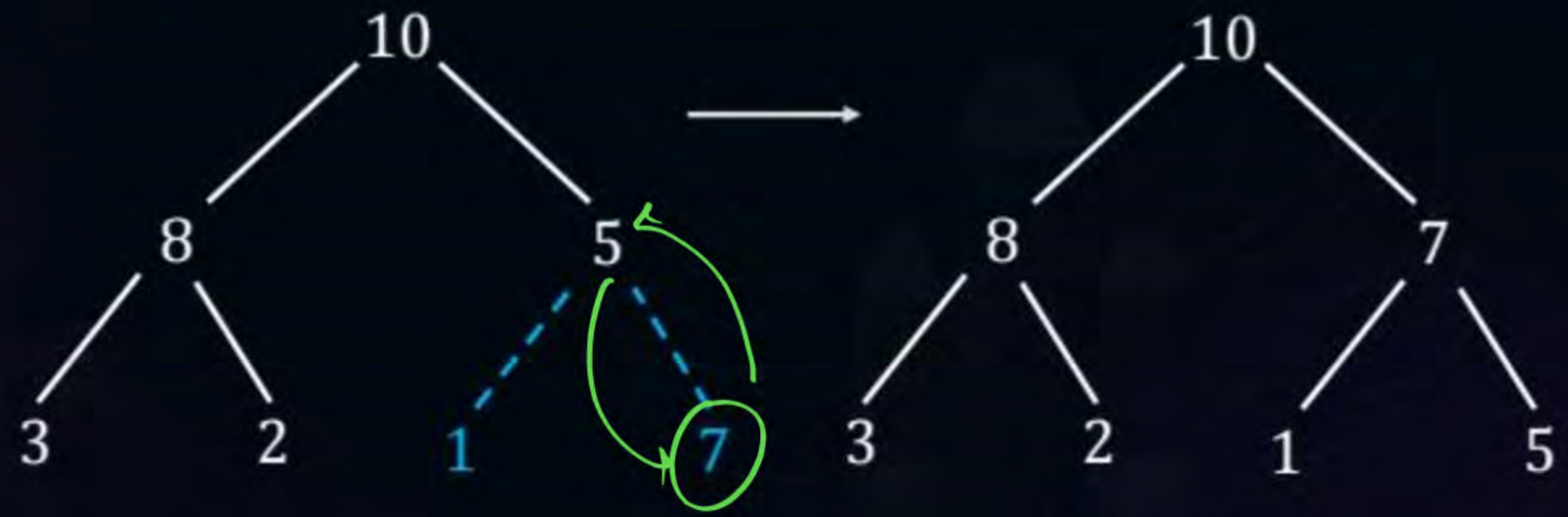
A 10, 8, 7, 3, 5, 2, 1

B 10, 8, 7, 3, 2, 1, 5

C 10, 7, 8, 3, 5, 1, 2

D 10, 7, 8, 5, 3, 2, 1

**Sol.** (10 8 5 3 2)




Max-Heap

Level order Traversal: 10 8 5 3 2

[MCQ]

V. imp

#Q. In a Binary Max-Heap with n elements, the smallest element can be found in time of_____.

6 3.1.

A $O(n \log n)$

B $O(n)$

C $O(n^2)$

D $O(1)$

[MCQ]

#Q. In a Binary Max-Heap with n elements, the smallest element can be found in time of_____.

A $O(n \log n)$

B $O(n)$

C $O(n^2)$

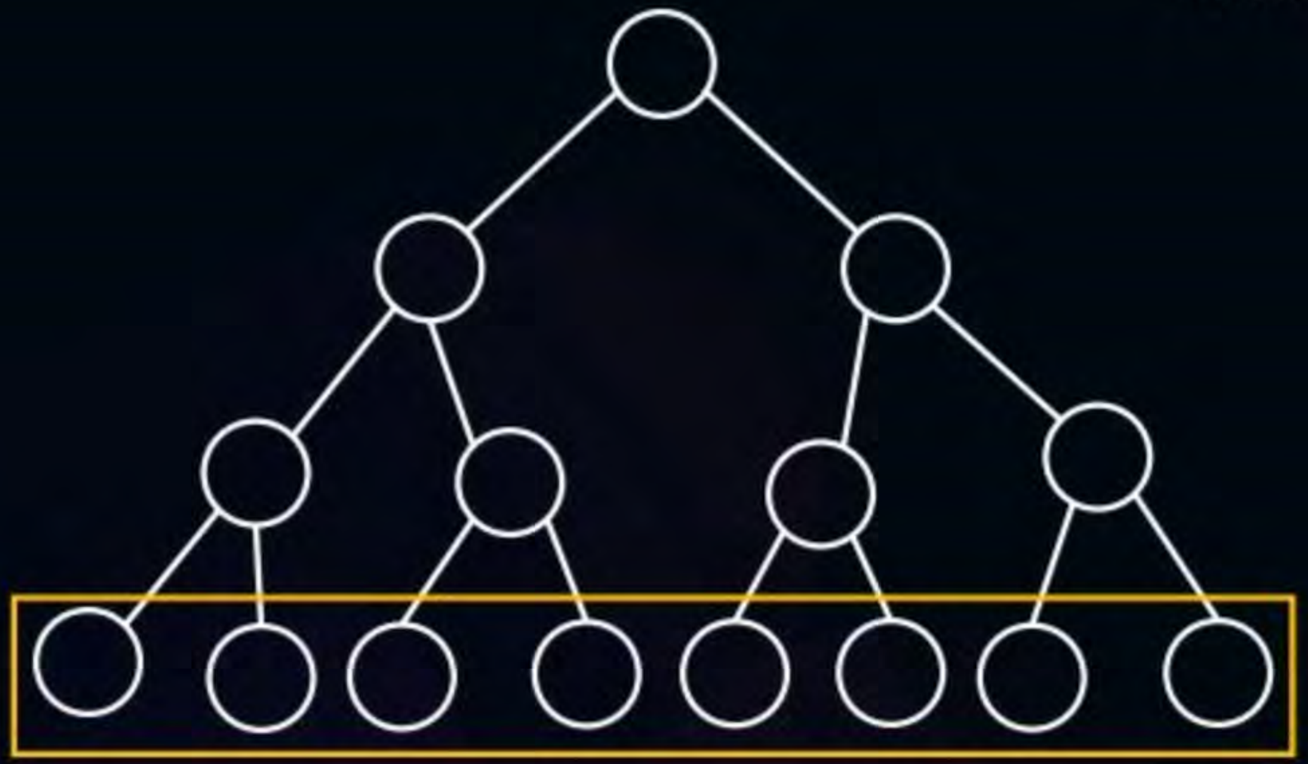
D $O(1)$

**Sol.** Appr1: Heap Sort : $O(n \log n)$


$O(n \log n)$

Not the best appr.

**Sol.** Appr2: Max-Heap



≈ n/2

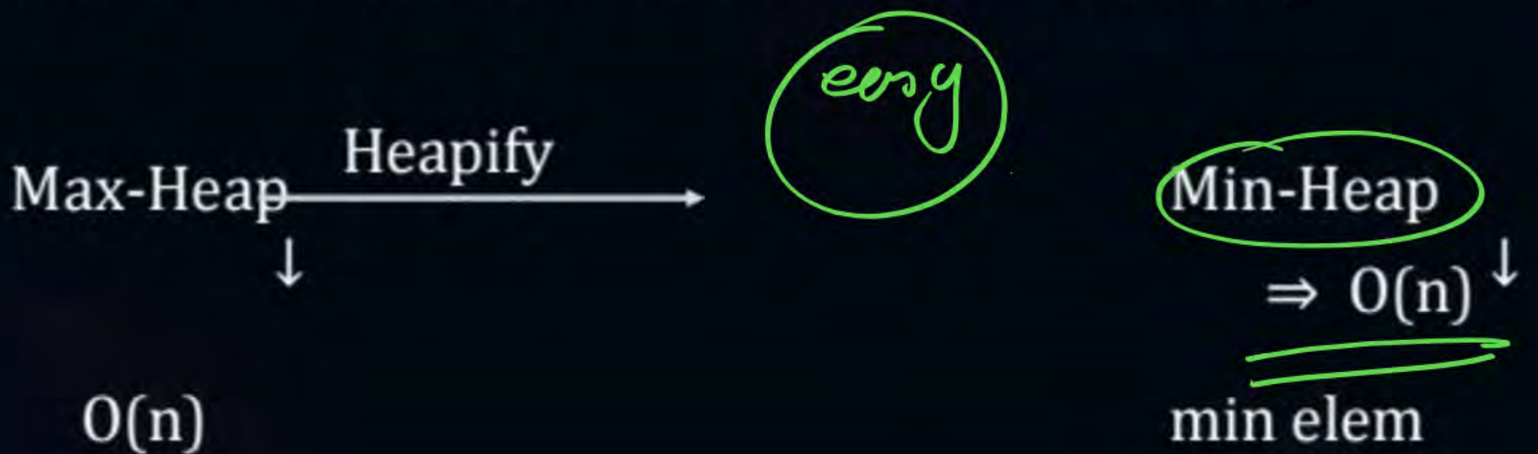
Total 15 elements → n
8 → leafs

15 → 8
n → ≈ n/2 elements at leaf

Logic: Linear search on those n/2 elements
→ O(n/2)
→ O(n)

In max Heap, Smallest element is at the leaf.

**Sol.** Appr3: Given max-heap, perform heapify to convert it to min-heap and then print the root (1st element / min element)




$O(n+1) = O(n)$

[MCQ]

#Q. Given binary Heap with 'n' elements & it is required to insert 'n' more elements not necessarily one after another into this Heap.
Total time required for this operation is:

A $O(n^2)$

B nlogn

C n

D $n^2$logn

[MCQ]

#Q. Given binary Heap with 'n' elements & it is required to insert 'n' more elements not necessarily one after another into this Heap.
Total time required for this operation is:

A $O(n^2)$

B nlogn

C n

D $n^2$logn

**Sol. (c)** WC complexity of Best Algorithm

min { $wc_1$, $wc_2$, $wc_3$ }

Given Heap inset

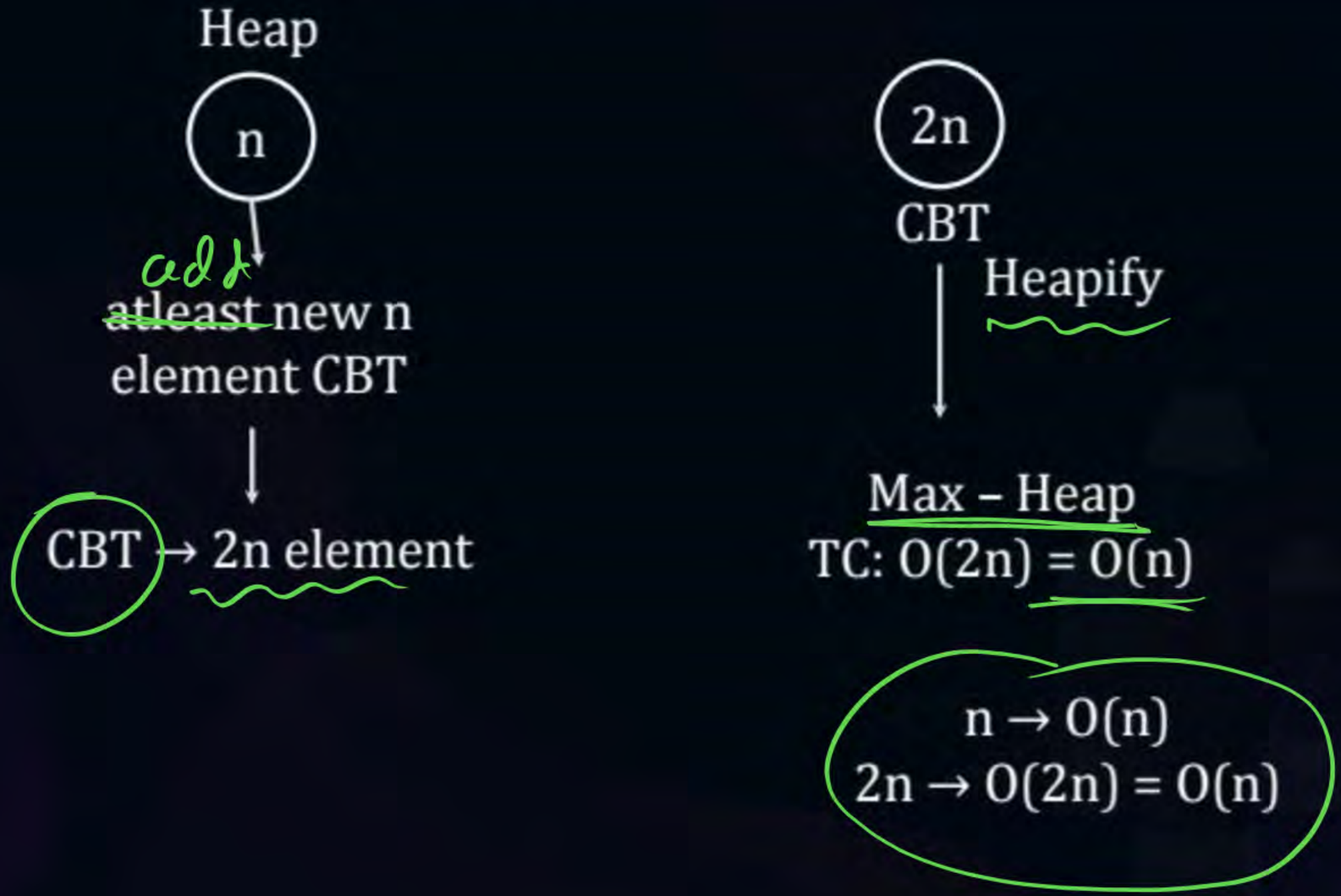
n ← n element

Appr1: Insertion method (one – by – one)

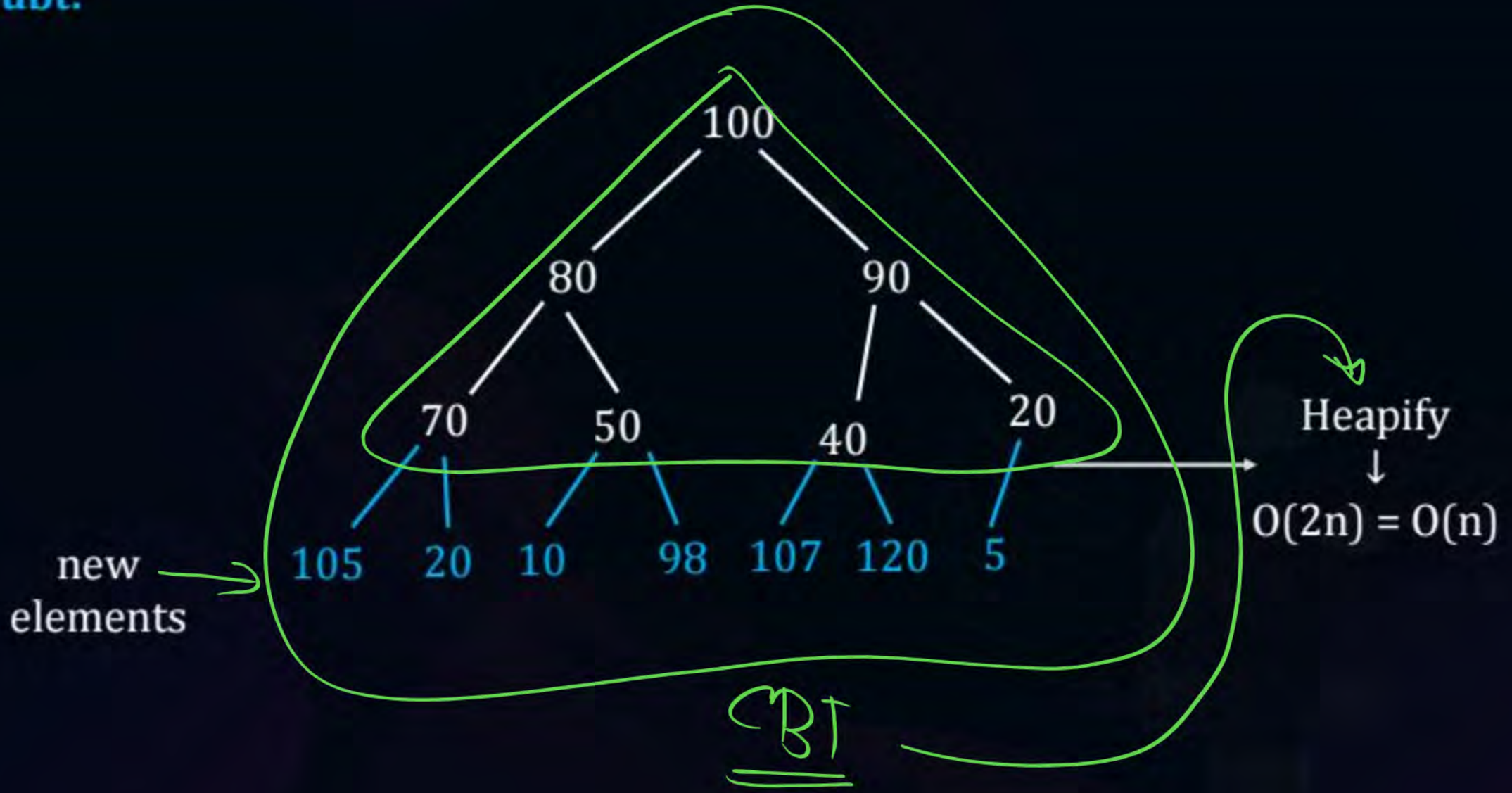
n ← 1 inset operation → O(logn)

To insert n elements → O(nlogn)

**Sol.** Appr2: Heapify / Build Heap:



**Doubt:**


100
80
90
70
50
40
20
105
20
10
98
107
120
5
new elements
Heapify
O(2n) = O(n)
CBT

[MCQ]

V. V. Good

#Q. Given Binary Heap in Array with the smallest at the root, the $7^{th}$ smallest element can be found in time complexity of _____.

A O(n)

B O(nlogn)

C O(logn)

D O(1)

[MCQ]

#Q. Given Binary Heap of size n in Array with the smallest at the root, the 7th smallest
element can be found in time complexity of ____.

A O(n)

B O(nlogn)

C O(logn)

D O(1)

Constant (independent of n)

**Sol:**

Appr1. Usual / Traditional Appr.


1st min element → 1 delete

2nd min element → 2 delete

⋮

7th min element → 7 delete

1 delete → O(logn)

7 delete → O(n × logn) = O(logn)

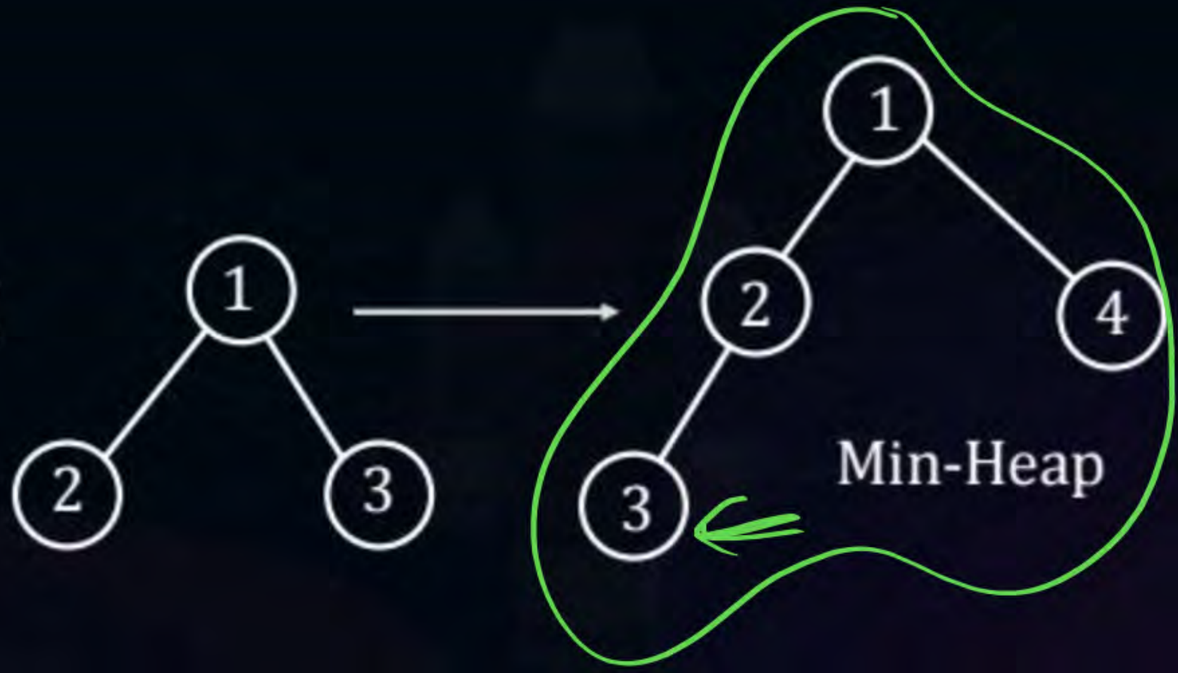
## Appr2. Better/non-traditional


Max depth

1st min element → 1

2nd min element → 2

3rd min element → ~~2~~ 3



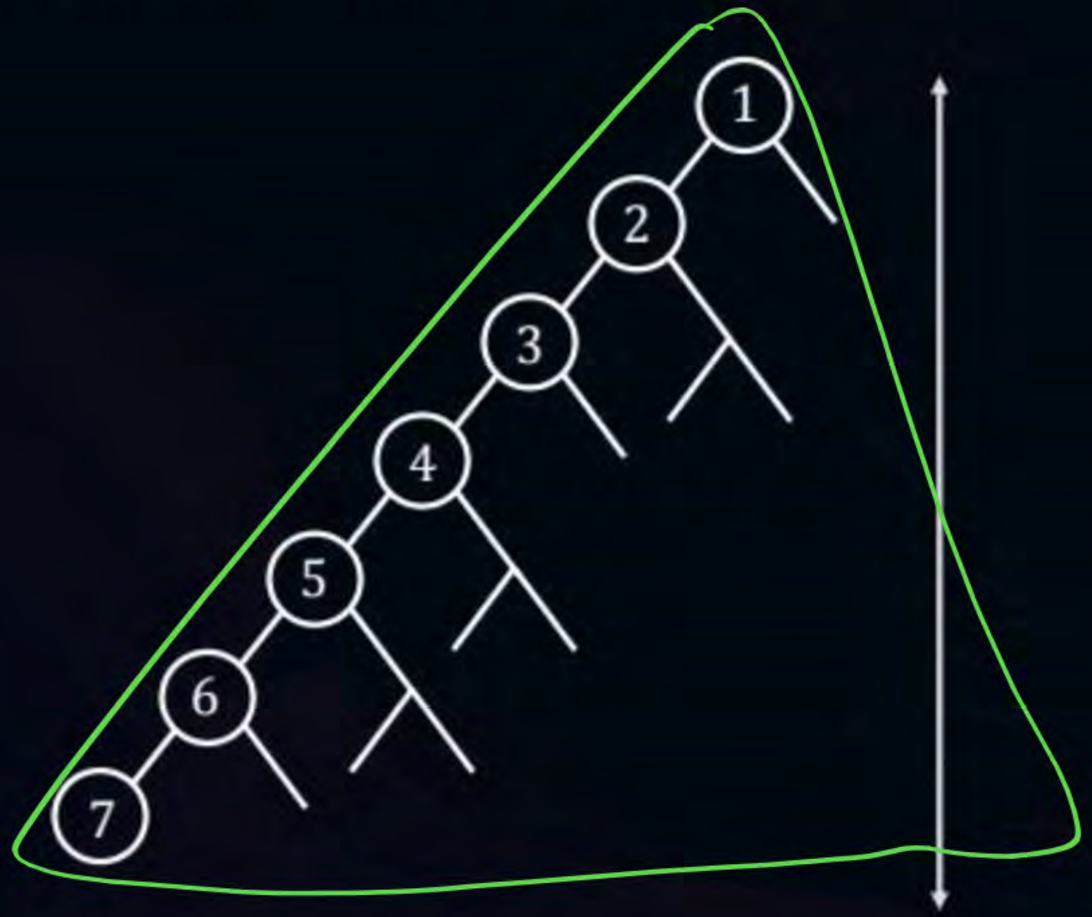
$4^{th}$ min element ⟶ max depth 4

$7^{th}$ min element ⟶ max depth 7.

$n = 10^6$

## CBT

In min-Heap $7^{th}$ min element within this always.




7th level (independent of n)

We can be sure that the $7^{th}$ minimum element is present within level 1 to level 7.

$1^{st}$ level ⟶ 1 elem→ $2^0$

$2^{nd}$ level ⟶ max 2 elem. → $2^1$

$3^{rd}$ level ⟶ $2^2$ elem.

⋮ ⋮

$7^{th}$ level ⟶ $2^6$ elem.

$i^{th}$ level ⟶ $2^{(i-1)}$ elem.

So, the $7^{th}$ min elem. in within level 1 to level 7 i.e. within

$(2^0 + 2^1 + 2^2 \ldots . 2^6)$elements

$$= \sum_{i=0}^{6} 2^i = \boxed{2^7 - 1}$$

Independent of n

[MCQ]

#Q. The approximate number of element that can be sorted in O(log n) time using Heap Sort is _____.

A $O(\log n)$ ✗

B $O(n)$ ✗

C $O\left(\frac{\log n}{\log\log n}\right)$ ✓

D $O(1)$ ✗

32 !

? ⟶ $O(\log n)$

**Sol:**

We know, n elements can be sorted in O(nlogn) time using Heap sort.

'n' elem. ⟶ O(n logn) time.

Check otp D:

n ⟶ nlogn.

logn ⟶ logn*(loglogn)

x elem. ⟶ xlogx

Check opt C.

$$x = 0\left(\frac{\log n}{\log\log n}\right)$$

$$\text{Time Read} = \frac{\log n}{\log(\log n)} * \log\left(\frac{\log n}{\log(\log n)}\right)$$

$$= \frac{\log n}{\log\log n} * (\log(\log n) - \log\log\log n)$$

$$= \frac{\log n}{\log\log n} * \log\log n - \frac{\log n}{\log\log n} * \log\log\log n$$

$$= \log - \left(\frac{\log n}{\log\log n}\right) * \log\log\log n = \log n * \left(\frac{\log n}{\log\log n}\right)$$

$$\neq O(\log n)$$

[MCQ]

#Q. Consider binary Heap in an Array with n elements. It is desired to insert an element into the Heap. If a binary search is performed along the path from newly inserted element to the root then the number of comparisons made is order of ____.

A $O(\log n)$

B $O(n)$

C $O(\log \log n)$

D $O(1)$

**Sol:**

max number of elements in the path from new node to root if Heap has elements?

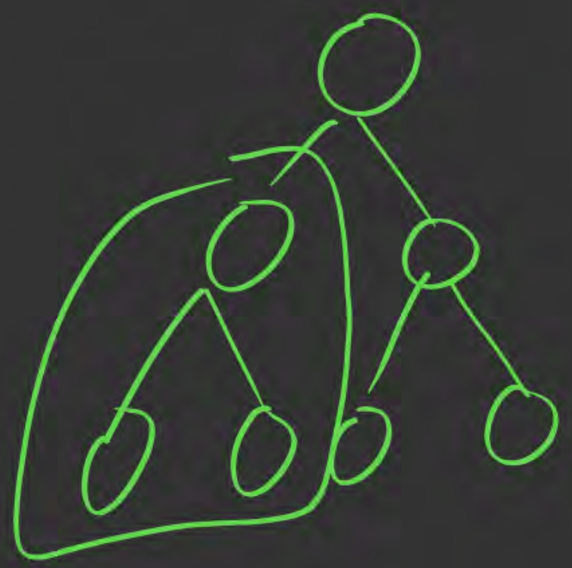

log (log n)

n elements ⟶ O(log n)

log n elements ⟶ O(log log n)

(Q) Set = {1, 2, 3, 4, 5, 6, 7} (unique)

How many max Heaps?



$$T(n) = {}^{n-1}C_k \, T(n-1-k) \cdot T(k)$$

$n \rightarrow$ no. of elem

$K \rightarrow$ no. of elem in left subtree.

$n = 7, K = 3$

$$T(7) = {}^{6}C_3 \, T(3) \, T(3)$$

$$= {}^{6}C_3 \times 2 \times 2 =$$

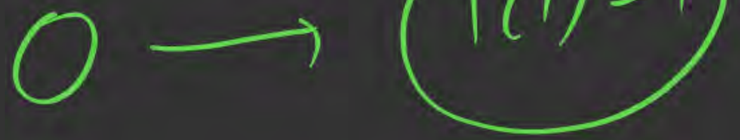
$$= \underline{\underline{80}}$$

$n = 3$
$K = 1$

$T(3) = {}^{2}C_{1}\, T(1)\, T(1)$

$T(3) = 2 \times 1 \times 1$

$= 2$



# Summary

Heaps

# THANK - YOU