Priority Queues

A priority queue, is a special kind of queue which access elements based on certain priority parameters defined by some mathematical definition.

In your normal queue, priority is defined with time of insertion.

a_{t_1} b_{t_2} $t_1 < t_2$ → FIFO

a has higher priority

Q Why do we need priority queue??

In OS, processes are executed based on priority.

Q → How to implement priority queue??

↳ array → queue → first (insertion sort)

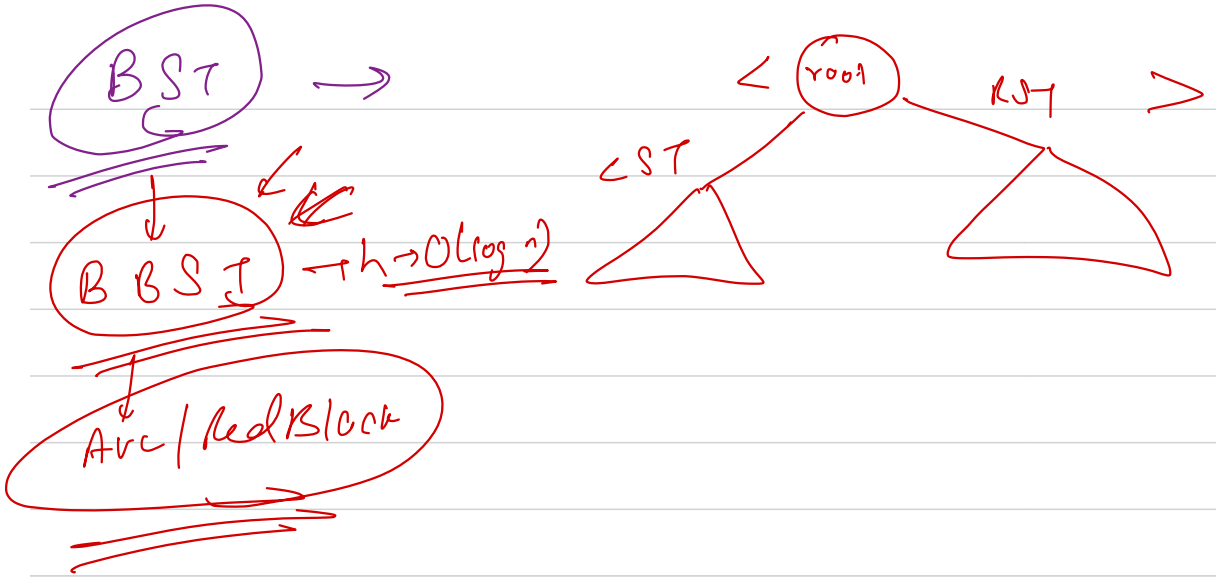
↳ sorted form

↳ insertion O(n)

get O(1)

↳ LL → no

↳ HM → No


BST
BBST
h → O(log n)
AVL / RedBlack
< root
RST
>
LST

Binary Heap → Priority Queue

- Binary Heap is a Binary Tree. (1)
- Binary Heap is a CBT.

- In a binary heap, every parent node has a higher priority than child node.

$$P_{parent} > P_{child}$$

Binary Heap → If priority of elements is defined as the larger value has higher priority then this type of heap is called Max-heap.

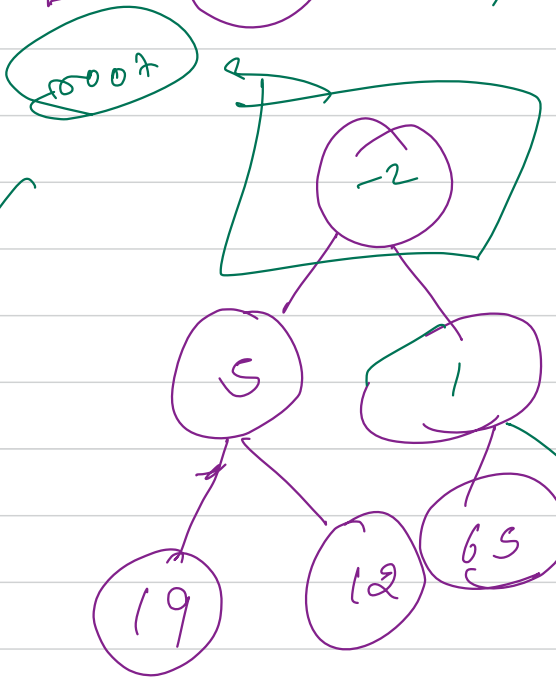
else if they are defined as the smaller value has higher priority then this type of heap is called Min-Heap.

Consider a min heap

12, 19, 33, 1, 5, 65, -2

CBT

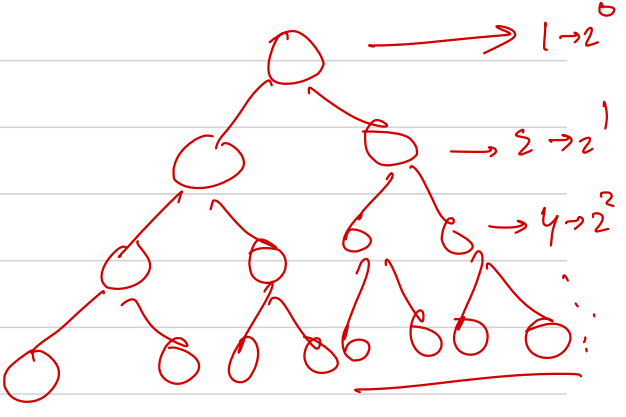
Binary heap



insert → O(log n)

get → O(1)

h



2^{h-1} ⇒ Nodes at last level of heap

$$\underbrace{2^{h-1} + 2^{h-2} + 2^{h-3} \cdots\cdots 2^1 + 2^0}_{h \text{ terms GP}} = n$$

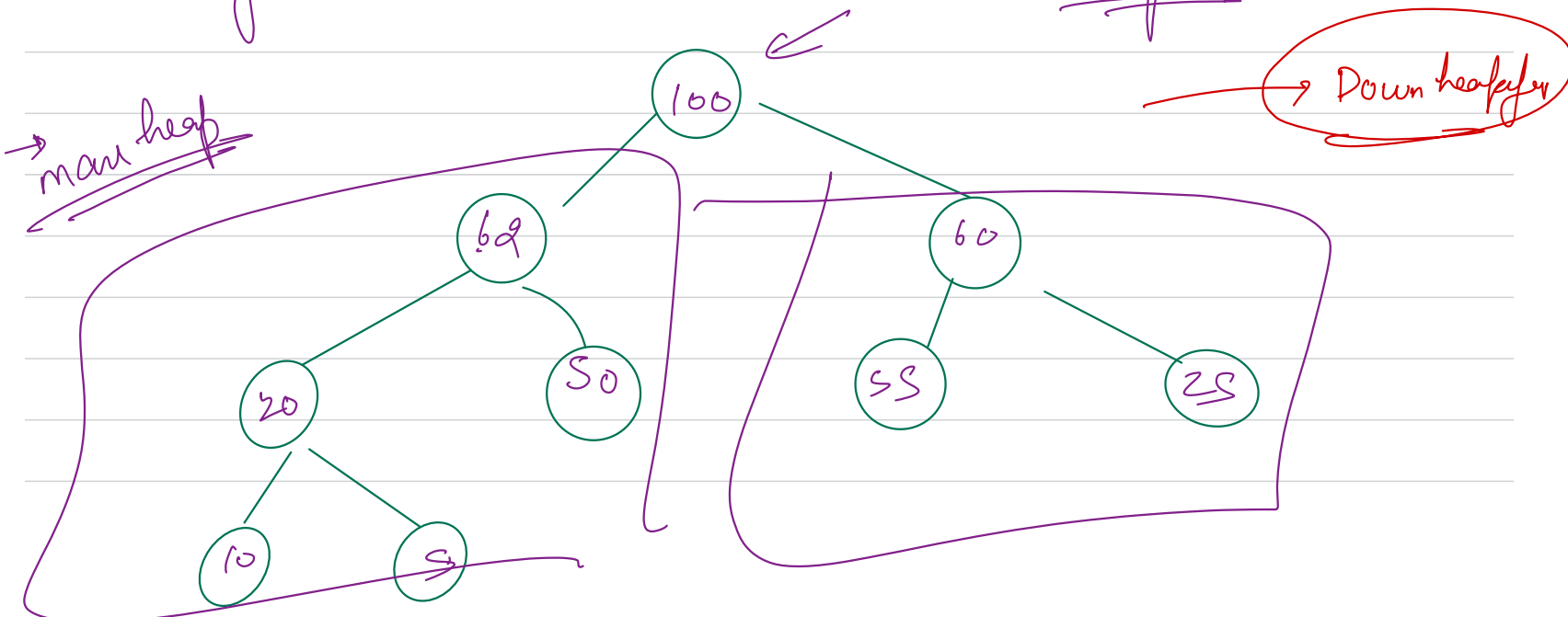
$$1 \times (2^h - 1) = n$$

$$2^h = n + 1$$

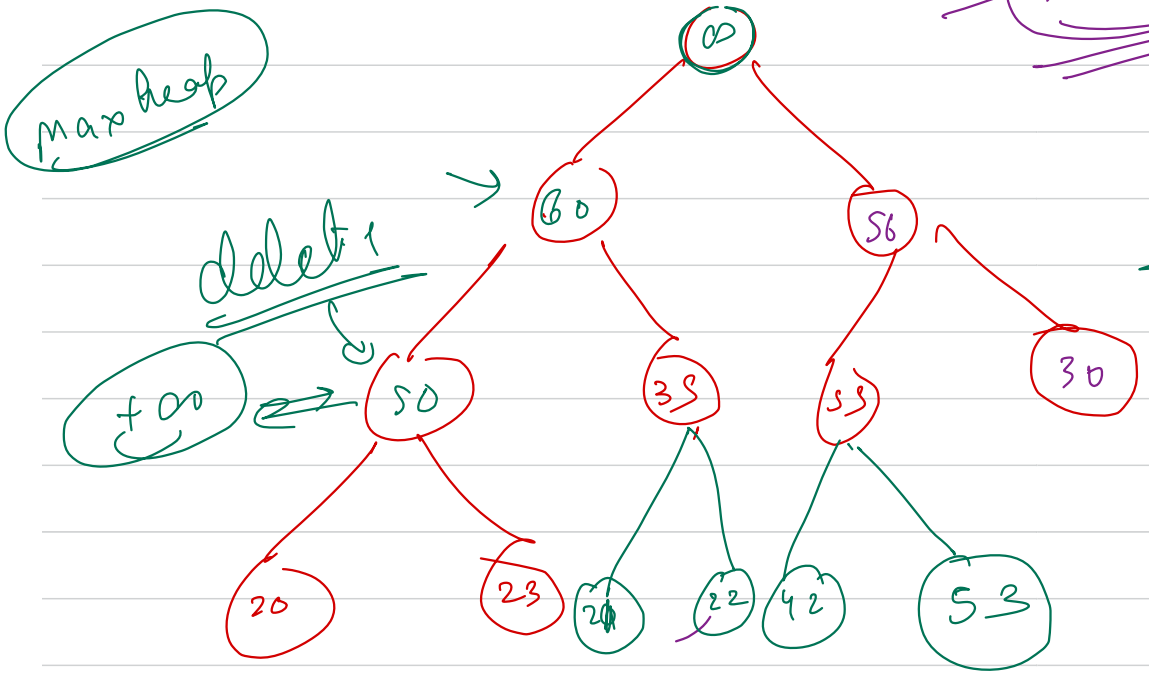
$$h = \log_2 (n+1)$$

$$\approx O(\log n)$$

Q. Given a BT, whose left subtree is a heap, RST is also a heap, but the tree is not a heap. Convert it into a heap.



max heap → root delete



removal → O(log n)

$T(n) = T(\frac{n}{2}) + 2$

$T(\frac{n}{2}) = T(\frac{n}{4}) + 2$

$T(\frac{n}{4}) = T(\frac{n}{8}) + 2$

$\vdots$

$T(2) = T(1) + 2$

$T(n) = T(n) + 2 \times K$

$O(\log n)$

K =

$T(n) = T(\frac{n}{3}) + 3$

$T(\frac{n}{3}) = T(\frac{n}{9}) + 3$

$T(\frac{n}{9}) = T(\frac{n}{27}) + 3$

$\vdots$

$T(3) = T(1) + 3$

$T(n) = T(1) + 3m$

$O(\log n)$

m =

$$n \to \frac{n}{2} \to \frac{n}{4} \to \frac{n}{8} \quad \cdots\cdots \quad \frac{n}{2^k}$$

$$\frac{n}{2^k} = 1$$

$$k = \log_2 n$$

$$m = \log_3 n$$

2^k

$2 \log_2 2^n$

$2 \log_2 2^n$

3^m

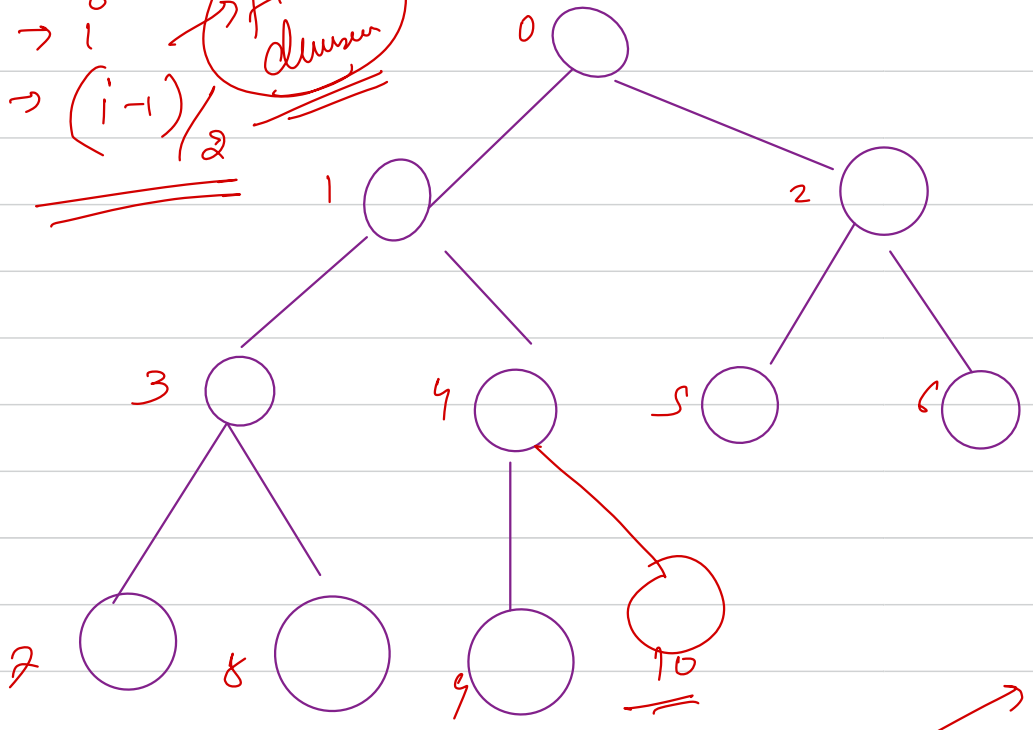
$3 \log_3 3^n$

$\frac{3}{\log 3} \log_2 2^n$

child → i

parent → $(i-1)/2$ → floor division

why heap is CBT?



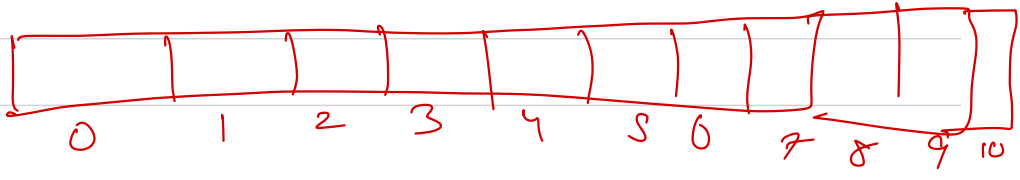
for any i^{th} node

LC → $2i+1$

RC → $2i+2$

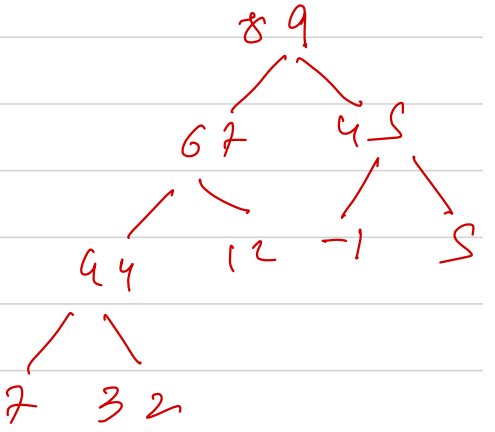
vector / list / array / ...

array →



heap → PQ for one operation

- insert → O(log n)
- remove → O(log n)
- get → O(1)



Q: If an array is sorted in asc order, will it represent a min heap ??

Q. Given an array, convert this array to a heap, in place.

Sorting → n → $O(n \log n)$ TC

Can you optimise ??

upheapify

i

random

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9

arr

0 — i-1

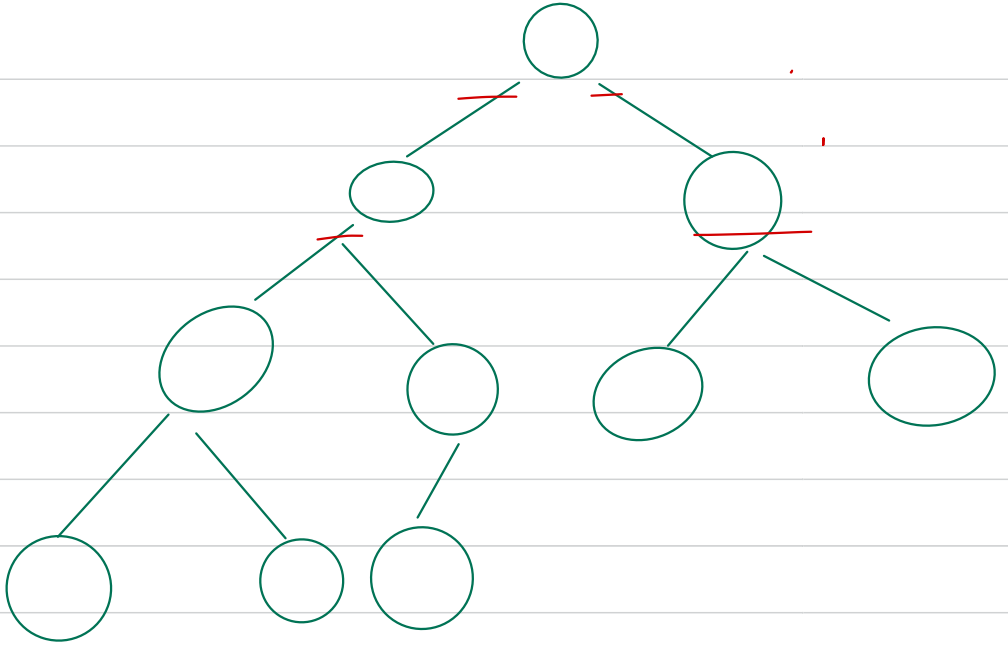
heap

$O(n \log n)$

2

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9

$O(n)$



AGP

$$S = 1\times 0 + 2\times 1 + 2^2\times 2 \cdots\cdots 2^{n-2}\times(n-2) + 2^{n-1}\times(n-1)$$

$$\rightarrow S = 2 + 2^2\times 2 + 2^3\times 3 + 2^4\times 4 \cdots 2^{n-2}\times(n-2) + 2^{n-1}(n-1)$$

$$2S = 2^2 + 2^3\times 2 + 2^4\times 3 \cdots\cdots 2^{n-1}\times(n-2) + 2^n(n-1)$$

$$2S - S = 2 + 2^2(1-2) + 2^3(2-3) + 2^4(3-4) \cdots + 2^{n-1}(n-2-n+1) + 2^n(n-1)$$

$$S = 2 - 2^2 - 2^3 - 2^4 \cdots\cdots - 2^{n-1} + 2^n(n-1)$$

$$= 2 + (-1)(2^2 + 2^3 + 2^4 \cdots\cdots 2^{n-1}) + 2^n(n-1)$$

$$\Rightarrow 2 + (-1)\frac{2^2(2^{n-2}-1)}{(2-1)} + 2^n(n-1)$$

$$= 2 + (-1)(2^n - 2^2) + 2^n(n-1)$$

$= 2 - 2^n + 2^2 + 2^4 (n-1)$

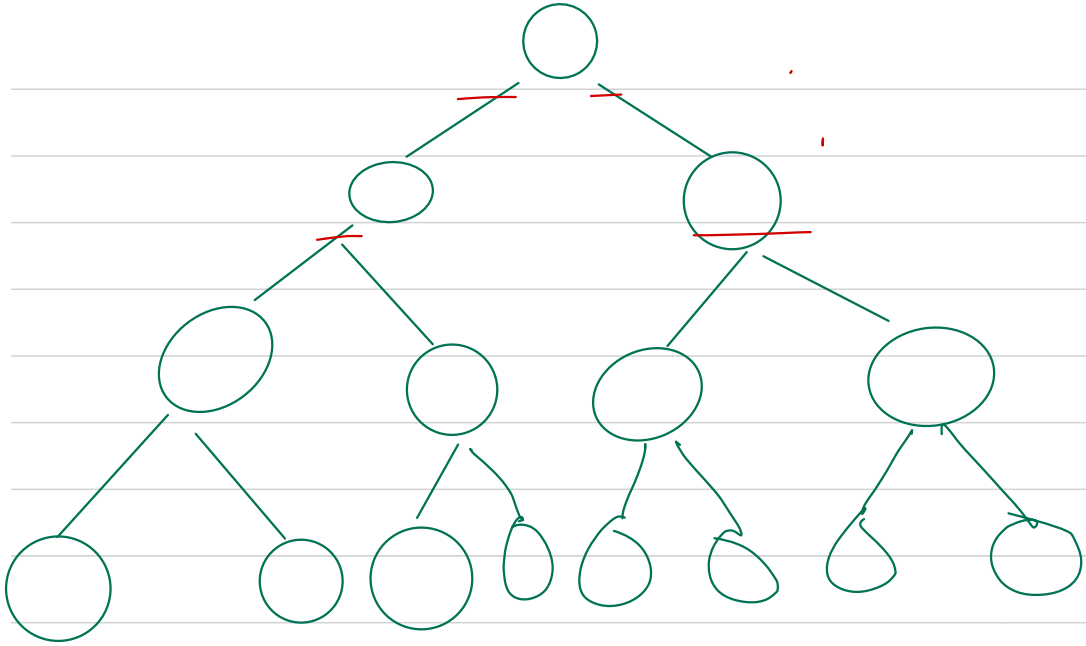
$h = \log n$

$2^{\log n} = n$

$\rightarrow 2 - 2^{\log n} + 2^2 + 2^{\log n}(\log n - 1)$

$2^{\log n}$: i runs 1 to $\log n$; $(\log n - 1)$: i runs 1 to $\log n$

$O(n \log n)$



$2^0 + 2^1 + 2^2 \text{ --- --- } 2^{h-2} + 2^{h-1} = n$

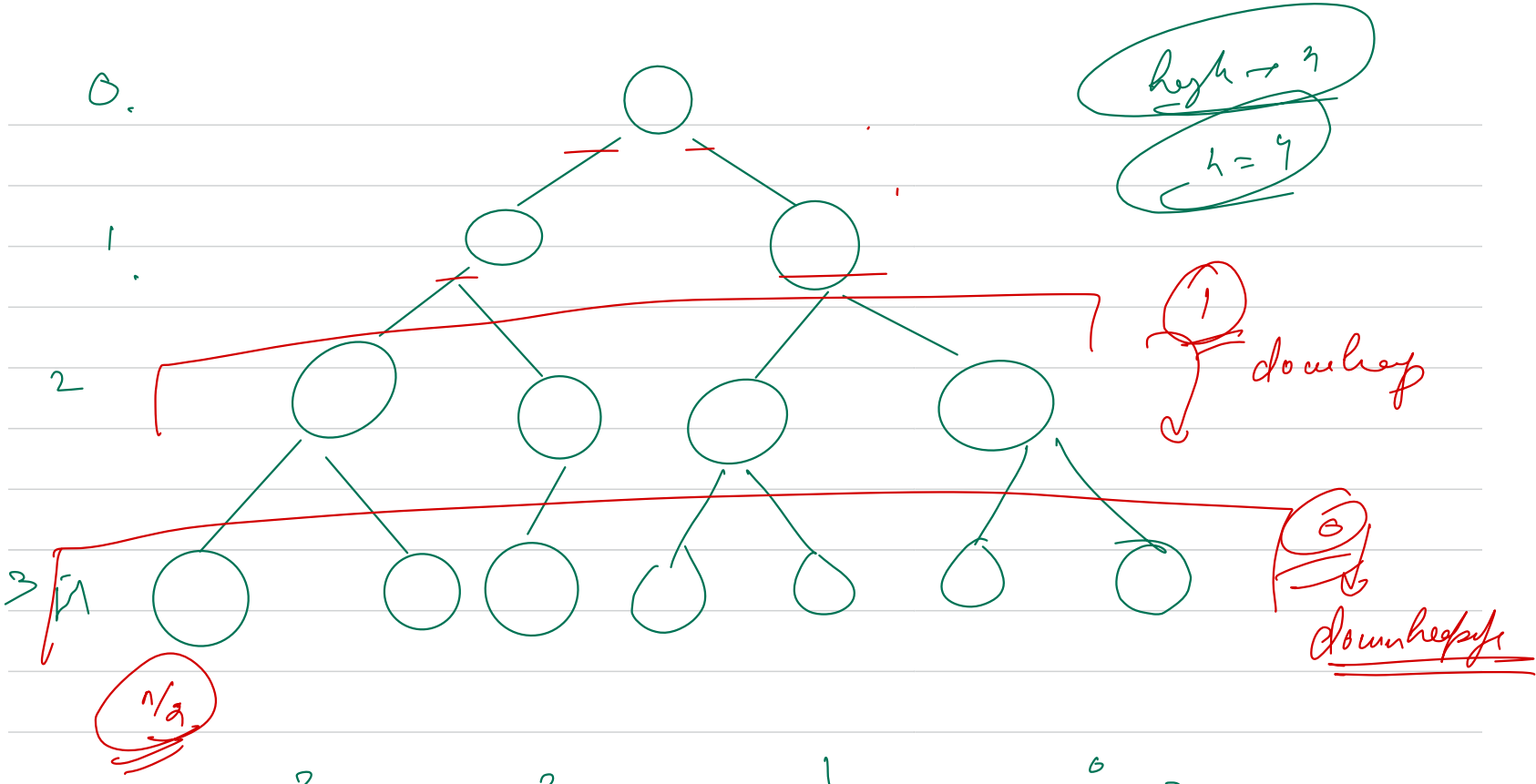
$1 \times (2^{h-1} - 1) + 2^{h-1} = n$

$2 \times 2^{h-1} = n + 1$

$\Rightarrow \boxed{2^{h-1}} = (n+1)/2$

# of nodes at last level

half of n nodes



$2^3 \times 0 + 2^2 \times 1 \rightarrow 2^1 \times 2 \quad 2^0 \times 3$

$$S = \underbrace{2^{h-1} \times 0}_{x} + 2^{h-2} \times 1 + 2^{h-3} \times 2 \cdots\cdots 2^{2} \times (h-3) + 2^{1} \times (h-2) + 2^{0} (h-1)$$

$$S = 2^{h-2} + 2^{h-3} \times 2 + 2^{h-4} \times 3 \cdots\cdots 2^{2}(h-3) + 2^{1}(h-2) + 2^{0}(h-1)$$

$$2S = 2^{h-2} \times 2 + 2^{h-3} \times 2 + 2^{h-3} \times 3 \cdots\cdots 2^{3}(h-3) + 2^{2}(h-2) + 2^{1}(h-1)$$

$$S = 2^{h-1} + 2^{h-2} + 2^{h-3} + 2^{h-4} \cdots\cdots 2^{3} + 2^{2} + 2^{1} - (h-1)$$

$$= 2^{h-1} + 2\left(2^{h-2} - 1\right) - (h-1)$$

$$\Rightarrow 2^{h-1} + 2^{h-1} - 2 - h + 1$$

$$\Rightarrow 2 \times 2^{h-1} - 1 - h$$

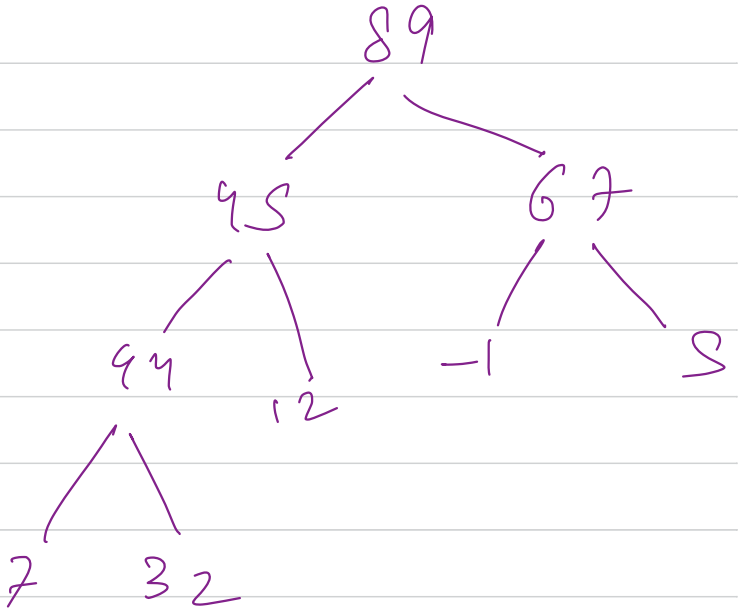
$$S = 2^h - h - 1$$

$$S = n - \log n - 1$$

$$\Rightarrow O(n)$$

2^h

$2^{\log n} = n$


89
45
67
44
12
-1
5
7
32

→ one by one input

↳ array ← O(n)

↓

build heap → O(n)