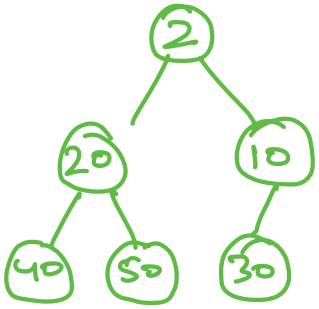
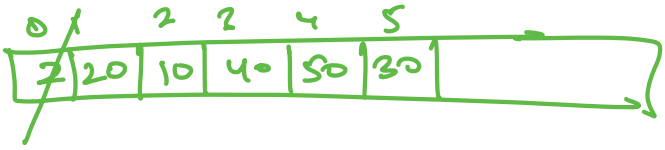
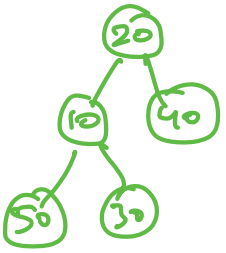
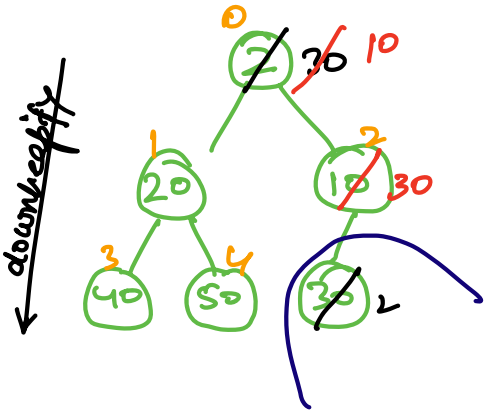
Removal → high Priority
least value

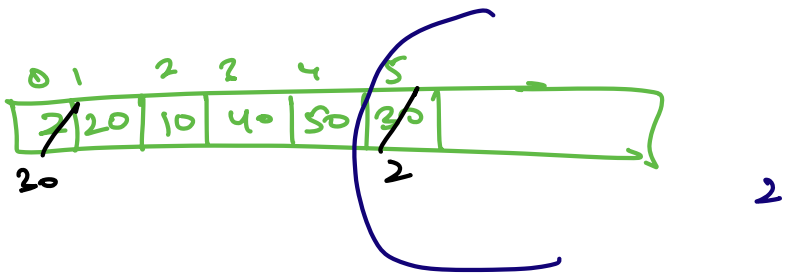




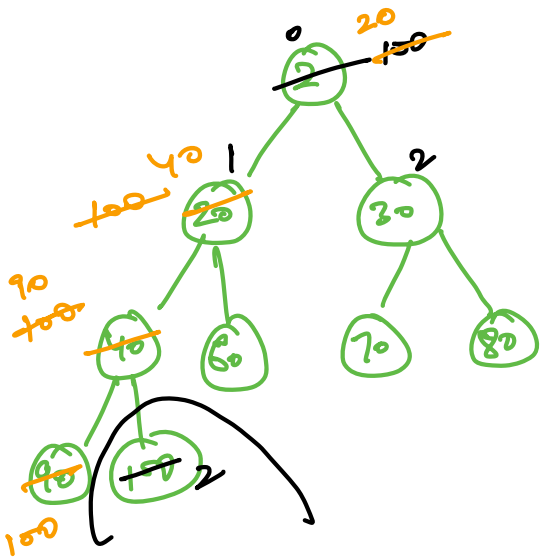


→ no longer heap



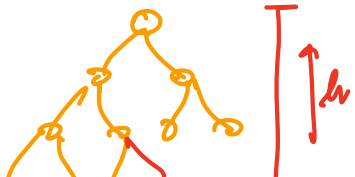


2



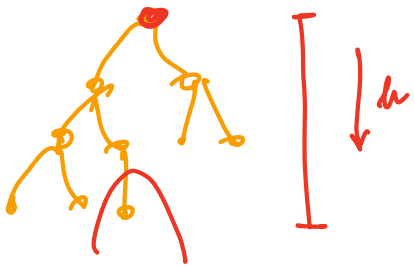
Time Complexity:

Add:



$1 + h : O(h) : O(\log n)$

Remove:

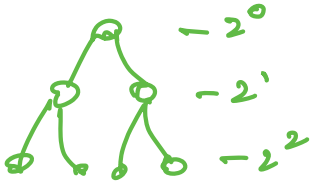


$1 + 1 + h = O(h) = O(\log n)$

Height ?

- Max elems to have ht h in a LBT.

$h = 2$



$2^0 + 2^1 + 2^2$

$2^0 + 2^1 + \dots + 2^h = n$

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

$$2^{h+1} = n+1$$

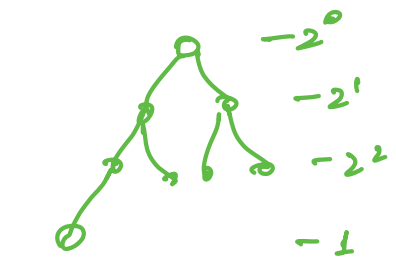
$$h+1 = \log(n+1)$$

$$h = \log(n+1) - 1$$

$$h = O(\log n)$$

- Min elems to have ht h

$h = 3$

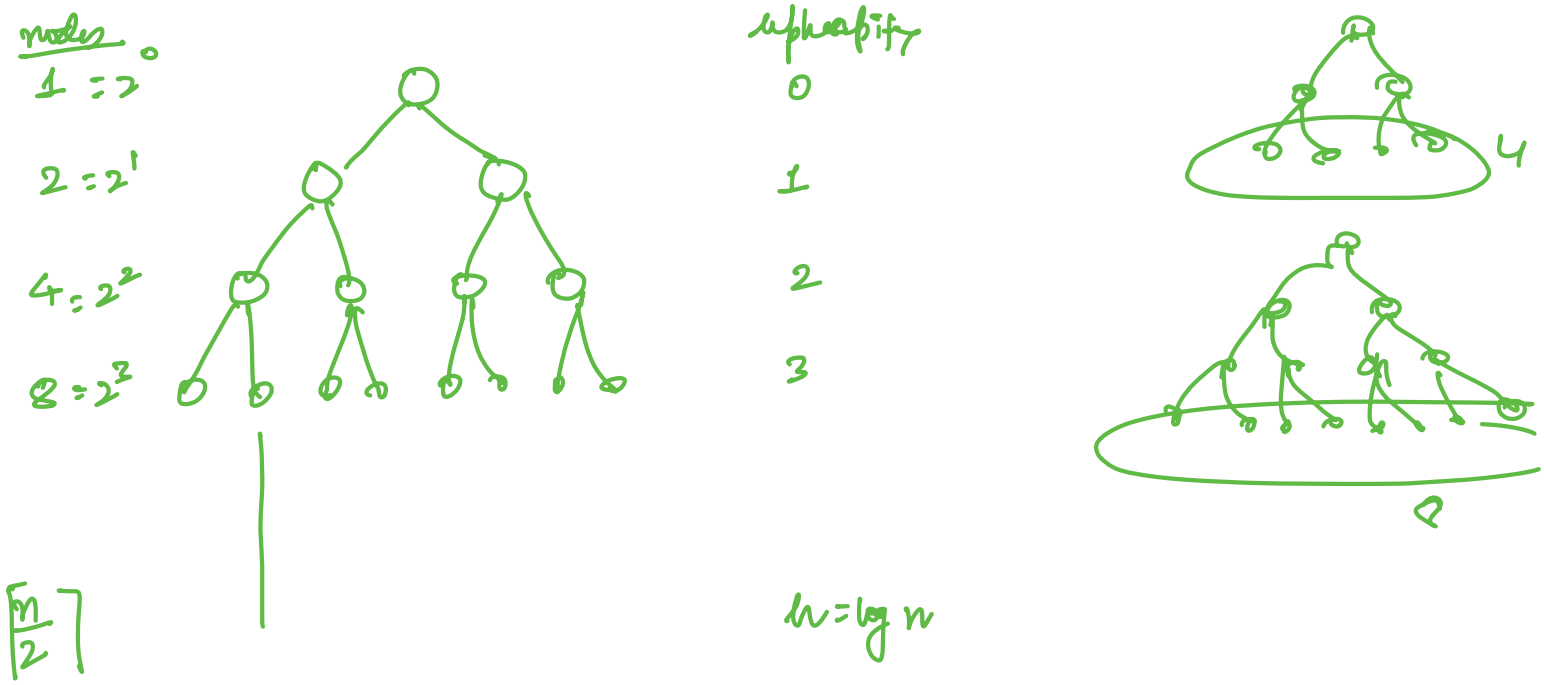


$2^0 + 2^1 + 2^2 + \dots + 2^{h-1} + 1 = n$

$$2^h - 1 + 1 = n$$

$$2^h = n$$

$$h = \log_2 n$$



$2^0 \times 0 + 2^1 \times 1 + 2^2 \times 2 + \longrightarrow \frac{n}{2} \log n$

$= O(n \log n)$

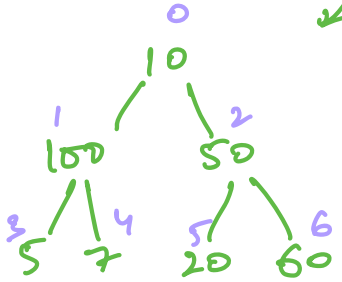
$1 \rightarrow \log n$

$n \rightarrow n \log n$

Setup 1: at a time only 1 element given

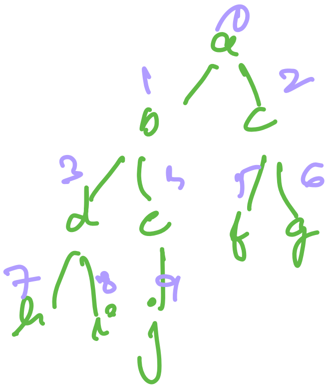
Setup 2: All n elements are given at once — $O(n \log n) \downarrow O(n)$

array: [10 (0) | 100 (1) | 50 (2) | 5 (3) | 7 (4) | 20 (5) | 60 (6)]



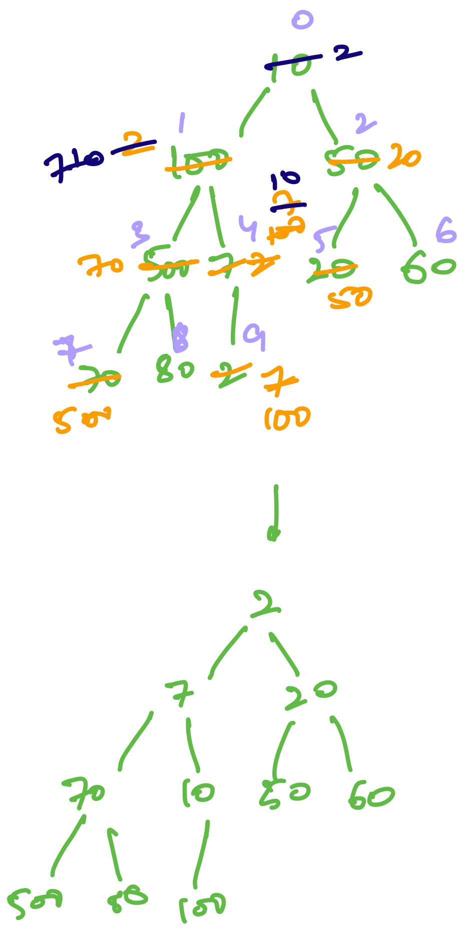
$n = 7$

$\frac{n}{2} \rightarrow n-1$ } leaf nodes



$n = 10$

$\frac{n}{2}$ — $n-1$ } $5 \to 9$



$n = 10$

$\frac{n}{2} \to n-1$ } $5 \to 9$ dont do anything

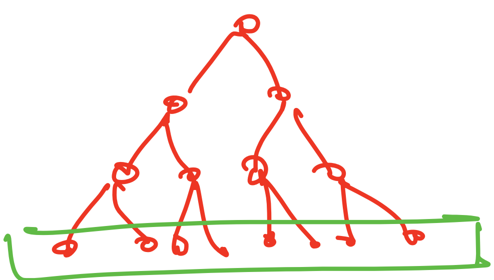
4 → downheapify

3 → dh

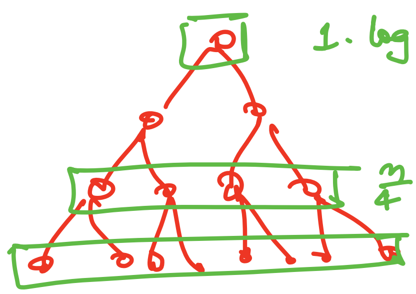
2 →

1 →

0 →



$n/2 \cdot \log n$



$1 \cdot \log n$

$\frac{n}{4} \cdot 1$

$\frac{n}{2} \cdot 0$