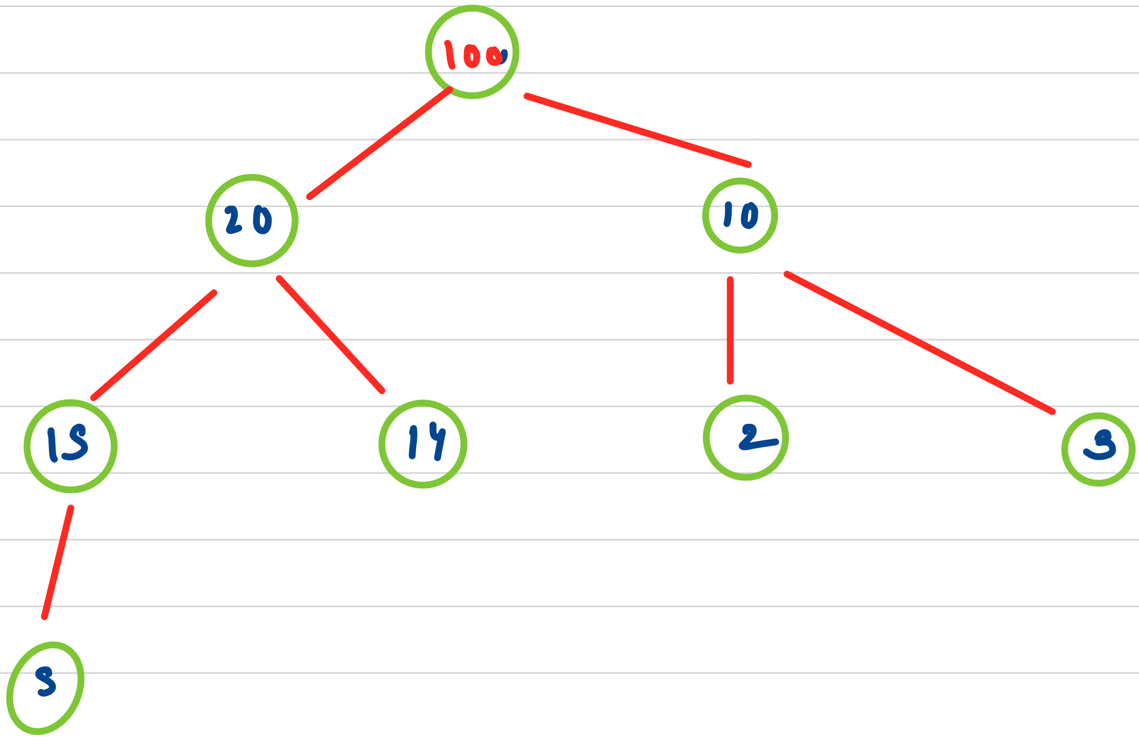
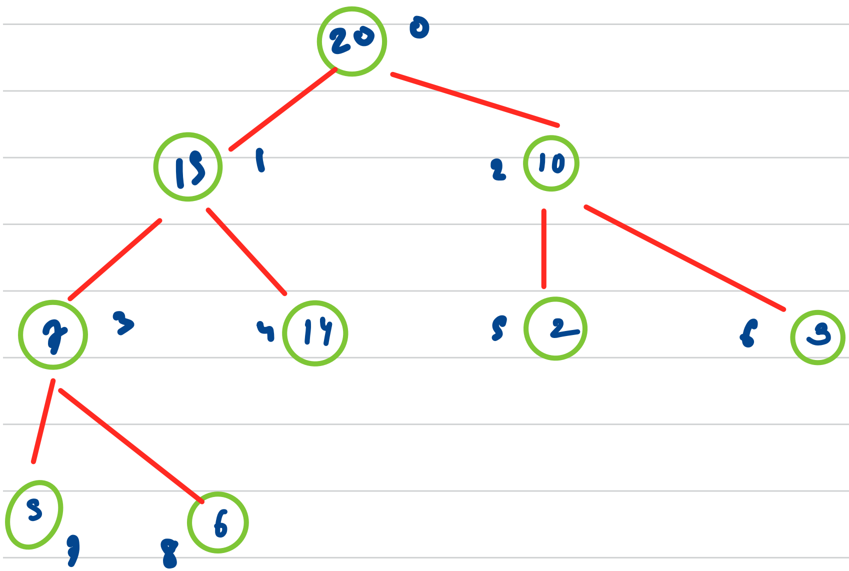
Max heap




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~100~~ 6 | 20 | 10 | 15 | 14 | 2 | 3 | 5 |

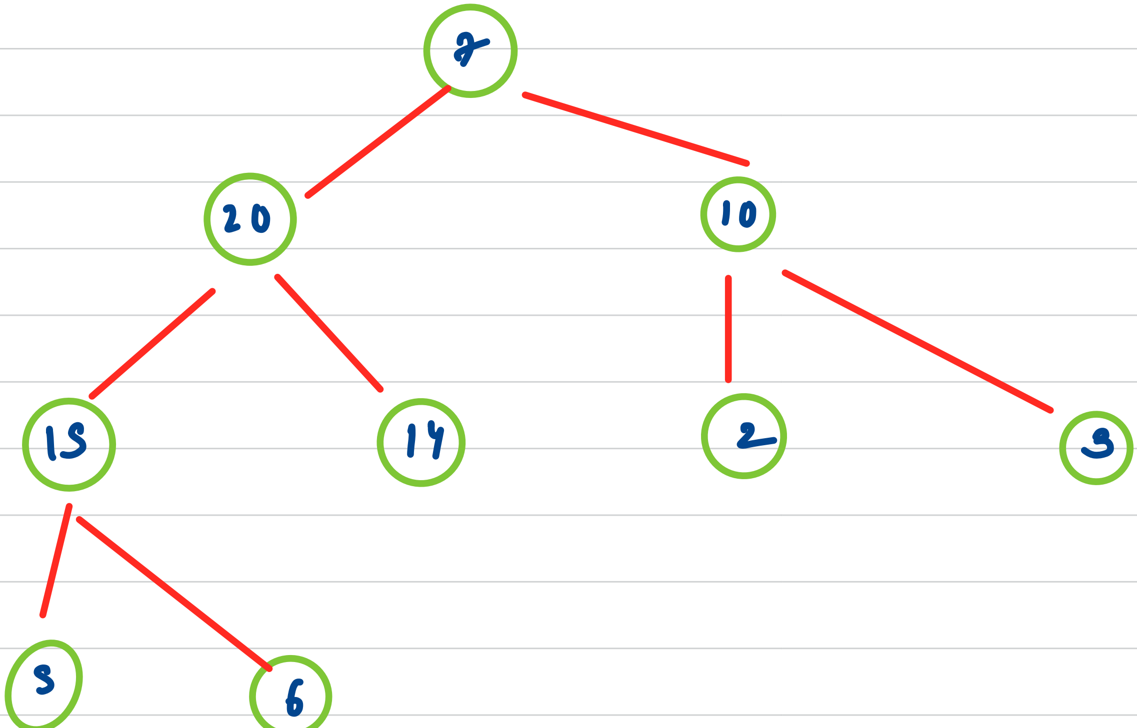


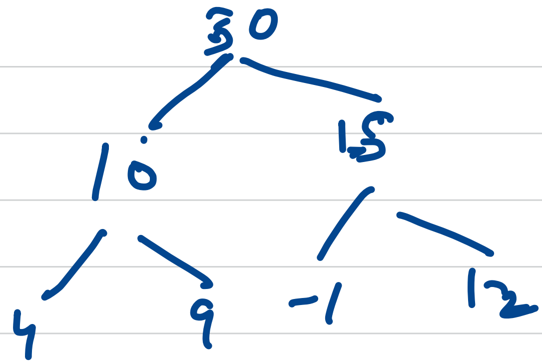

```
downHeapify(idx) {
    while(idx < this.arr.length) {
        let left = 2*idx + 1;
        let right = 2*idx + 2;
        let greatest = idx; // initially assume root is the greatest
        if(left < this.arr.length && this.arr[left] > this.arr[greatest]) {
            // if left child exist and it is greater than root, then greatest is
            greatest = left;
        }
        if(right < this.arr.length && this.arr[right] > this.arr[greatest]) {
            // if right child exist and right is greater than max(root, left) the
            greatest = right;
        }
        if(greatest == idx) {
            // we dont need to swap and we can stop
            break;
        }
        // swap
        let temp = this.arr[greatest];
        this.arr[greatest] = this.arr[idx];
        this.arr[idx] = temp;
        idx = greatest;
    }
}
```

idx = 3. left = 7 Right = 8

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 20 | 15 | 10 | 7 | 14 | 2 | 3 | 5 | 6 |

greatest = 3.


7
20
10
15
17
2
3
5
6

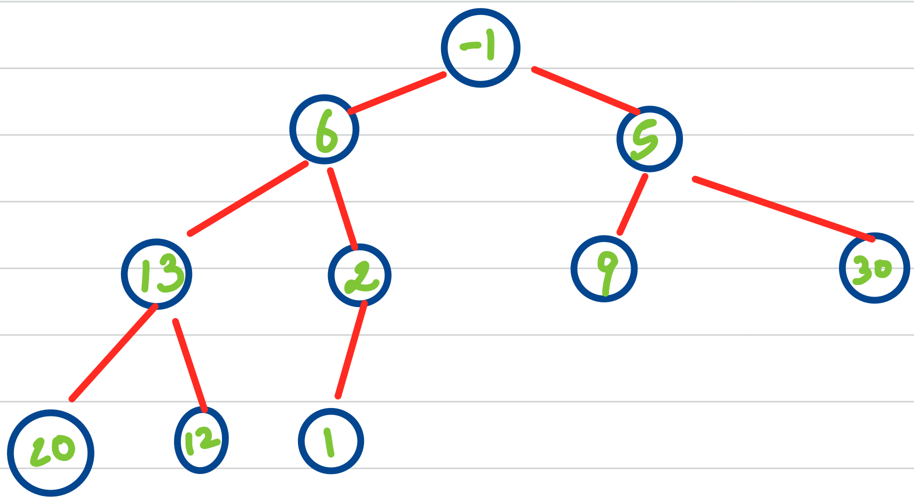

30
10
15
4
9
-1
12

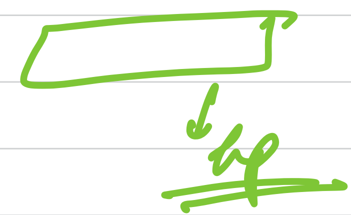
Green

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| -1 | 6 | 5 | 13 | 2 | 9 | 30 | 20 | 12 | 1 |

n

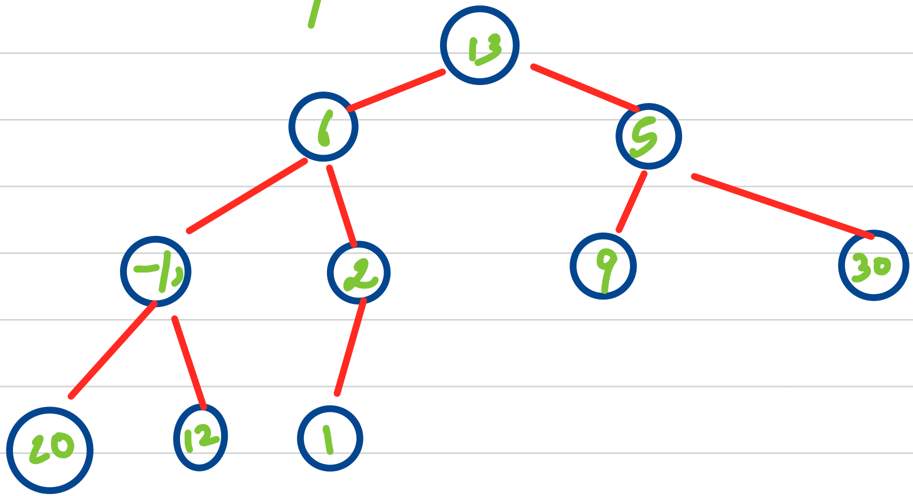
given array

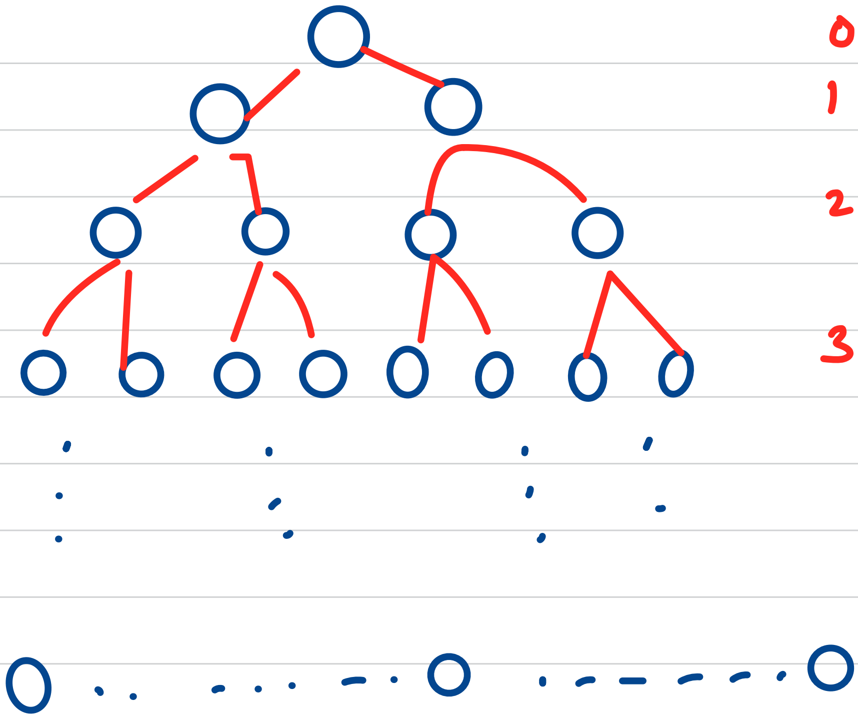


13, 6 , 5, -1, , 2 , 9, 30 , 20, 12, )

↑
i





| level | | Task done in units | |
|---|---|---|---|
| 0 | $2^0$ | 0 | $\rightarrow 2^0 \times 0$ |
| 1 | $2^1$ | 1 | $\rightarrow 2^1 \times 1$ |
| 2 | $2^2$ | 2 | $\rightarrow 2^2 \times 2$ |
| 3 | $\vdots$ | $\vdots$ | $\vdots$ |
| $h$ | $2^h$ | $h$ | $2^h \times h$ |

Total task done $\rightarrow (T)$

AGP $\rightarrow$ arithmetic geometric progression

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

multiply LHS and RHS by 2

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

Subtract (2) − (1)

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

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

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

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

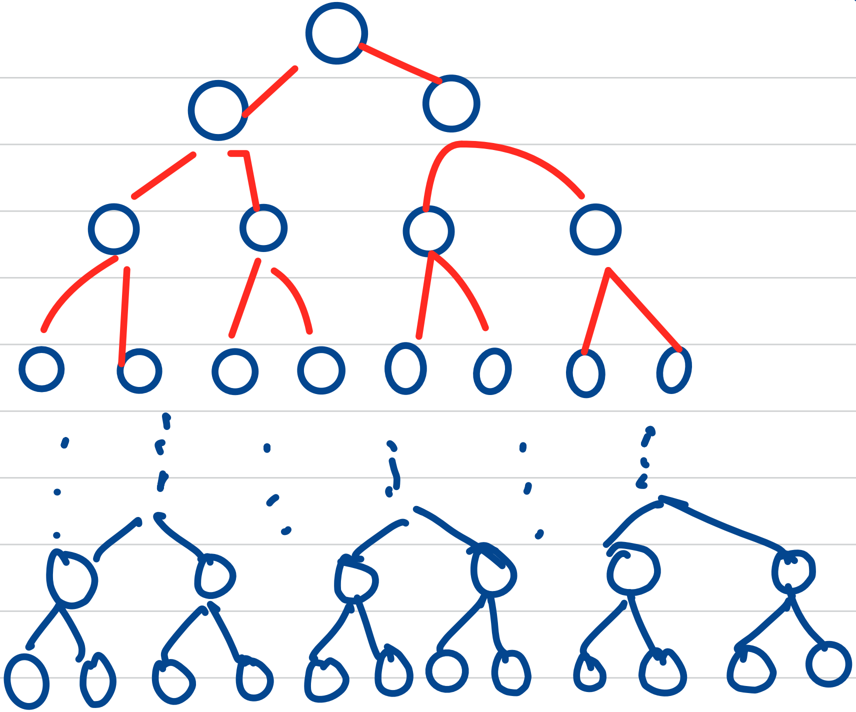
$$T = -1\left(\frac{2 \times (2^h - 1)}{2-1}\right) + 2^{h+1} \times h$$

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

$$h = \log_2 n \qquad (2^{\log_2 n} = n)$$

$$T = 2^{\log n + 1}(\log n - 1) + 2 \Rightarrow 2 \times 2^{\log n}(\log n - 1) + 2$$

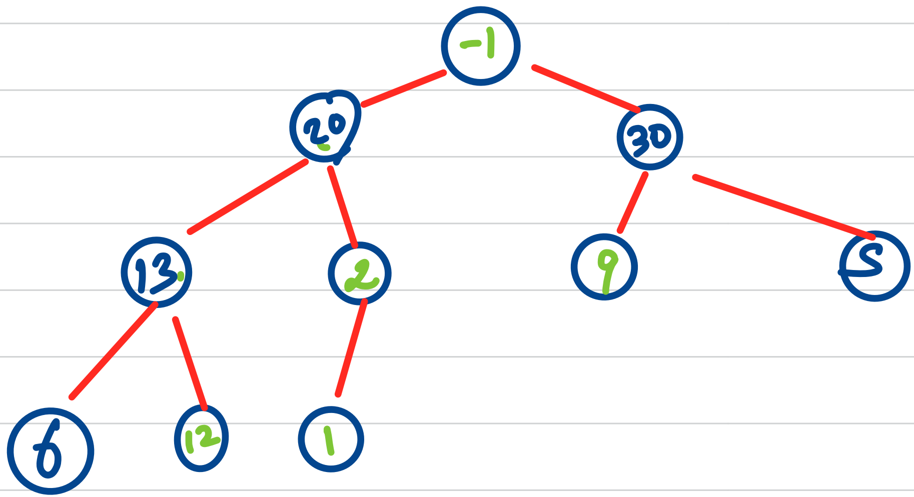
$$T = 2n(\log n - 1) + 2 \approx \underline{\underline{O(n \log n)}}$$



| level | Total nodes | Total work |
|---|---|---|
| 0 | $2^0$ | $2^0 \times h$ |
| 1 | $2^1$ | $2^1 \times (h-1)$ |
| 2 | $2^2$ | $2^2 (h-2)$ |
| 3 | $2^3$ | $\vdots$ |
| $h-1$ | $2^{h-1}$ | $2^{h-1} \times 1$ |
| $h$ | $2^h$ | $2^h \times 0$ |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| -1 | 20 | 30 | 13 | 2 | 9 | 5 | 6 | 12 | 1 |

n



$$T = 2^0 \times h + 2^1(h-1) + 2^2(h-2) \cdots\cdots 2^{h-1} \times 1 + \underbrace{2^h \times 0}_{\to 0} \quad \text{— (1)}$$

multiply both sides by 2

$$2T = 2 \times h + 2^2(h-1) + 2^3(h-2) \cdots\cdots 2^{h-1} \times 2 + 2^h \times 1 \quad \text{— (11)}$$

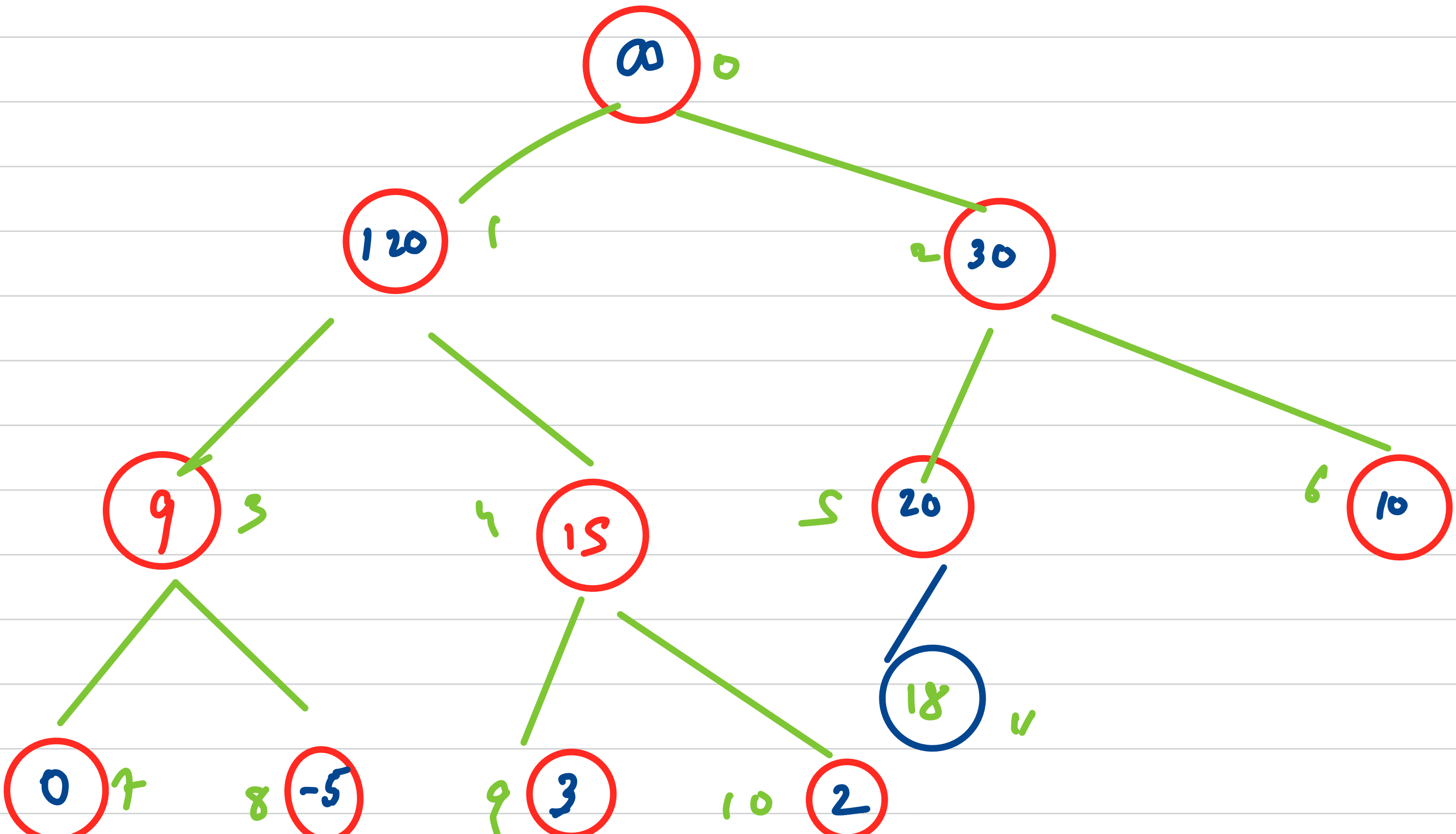
(11) − (1)

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

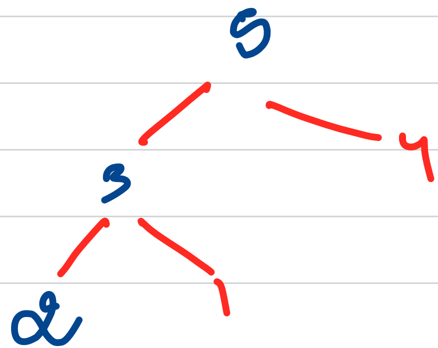
$$T = h + \underline{2(2^h - 1)} \qquad h = \log_2 n$$

$$T = \log_2 n + 2 \times 2^{\log_2 n} - 2 \Rightarrow \log_2 n + 2n - 2$$

$$\approx \underline{O(n)}$$


∞
0
120
1
30
2
9
3
15
4
20
5
10
6
18
11
0
7
-5
8
3
9
2
10

3, 2, 1, 5, 6, 4




k

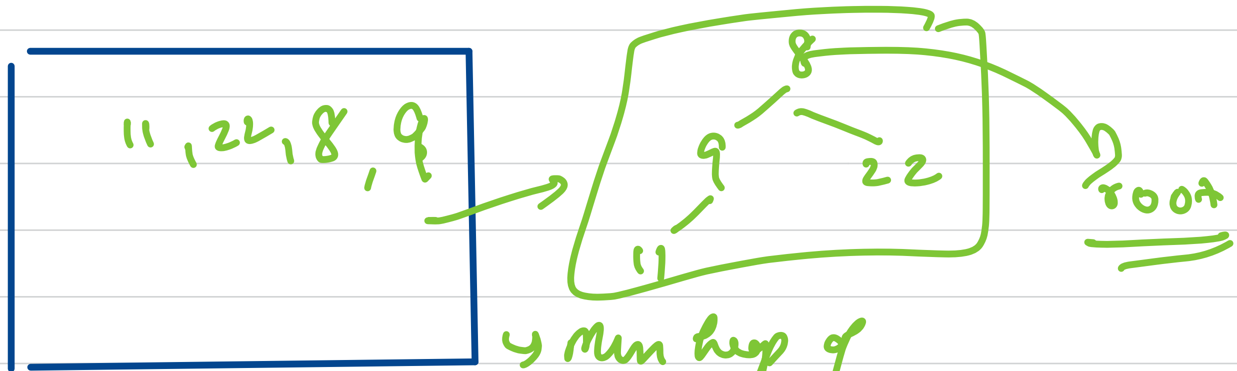
2 largest

k-1 elements from max heap

(k log n)

11, 2, 1, 3, 6, 5, 9, 22, 4, 8

K = 4th largest




→ min heap of size k

Space → O(k)

every time we remove an element & add ith index elements we get k largest elements uptill index i

$\boxed{11, 2, 1, 3}$, 6, 5, 9, 22, 4, 8

Space → O(k)

Time → O(n log k)

11, 22, 8, 9

→ Min heap  k