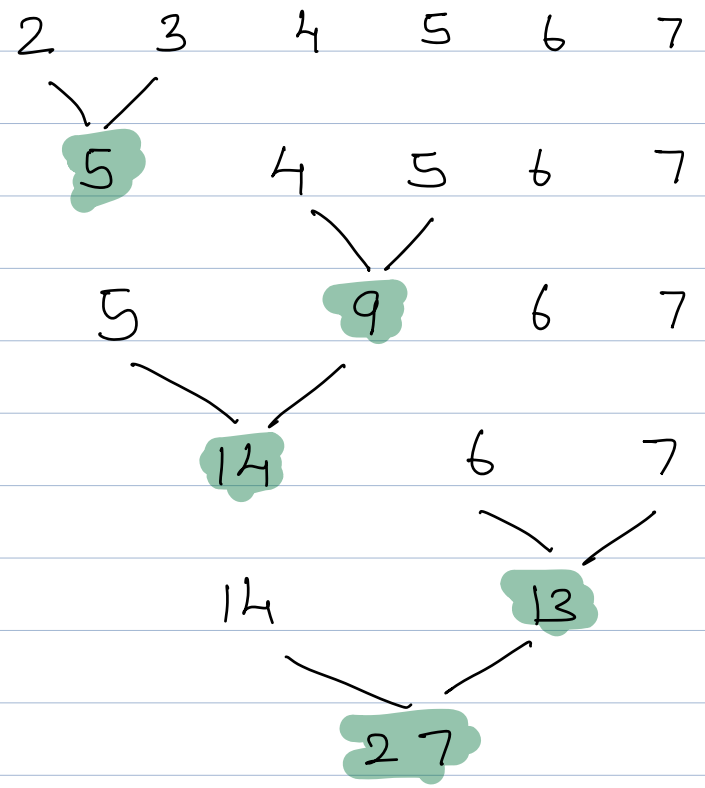


Cost = 5
+ 9
+ 14
+ 13
+ 27
⇒ 68

$x < y < z$

$(x+z), y$

$x+z+y$

⇒ $x+z$

$x+z+y$

$= 2x+y+2z$

$x < y < z$

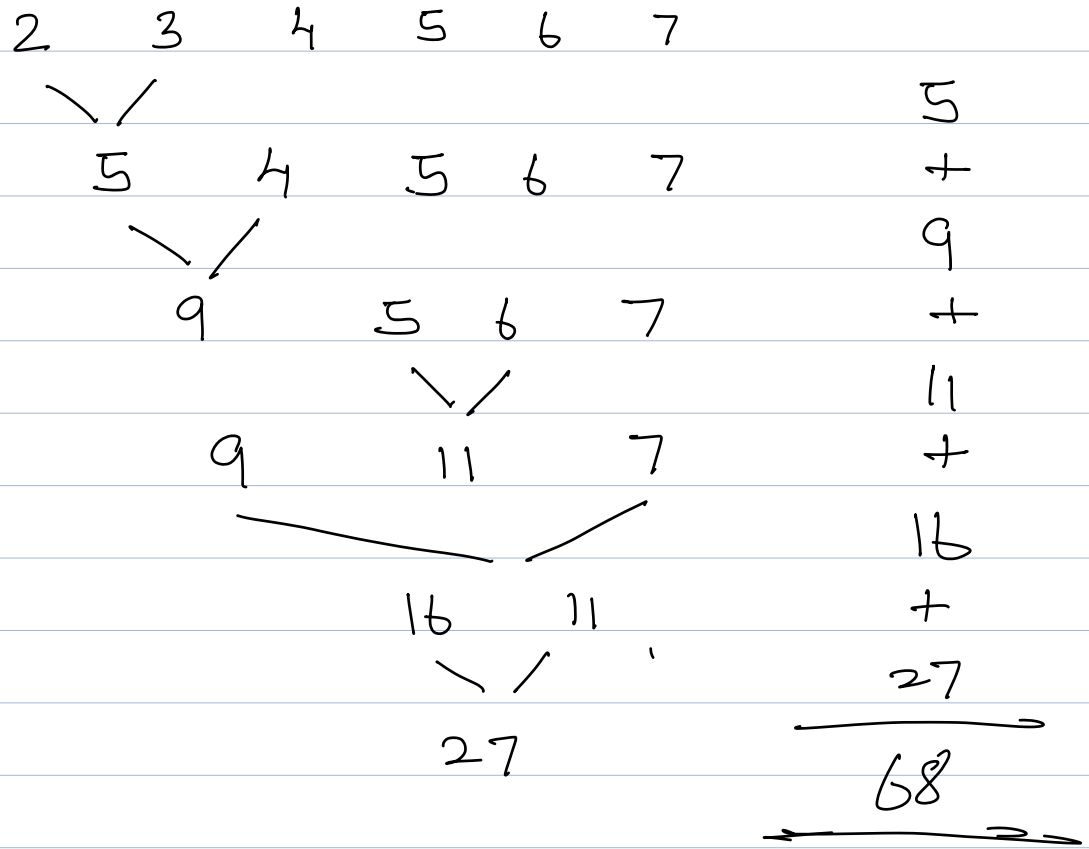
$x+y, z$

$x+y+z$

$x+y$.

$+ x+y+z$

$2x+2y+z$



1) Sort after every iteration

$n \log n + n-1 + n-2 + n-3 \ldots\ldots 1$

$\Rightarrow n^2 + n^2$

$\simeq n^2$

Sc: O(1)

2) Ordered map [Treeset , map]

↳ Keys are sorted
- insertion O(log n)
- deletion O(log n)
- Fetching element O(log n)

BBST

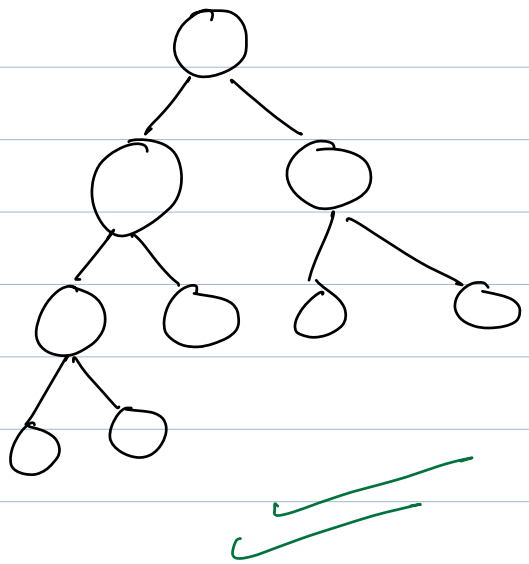
TC : $n \log n$. $+ n \log n$

$\approx n \log n$

SC : $O(n)$

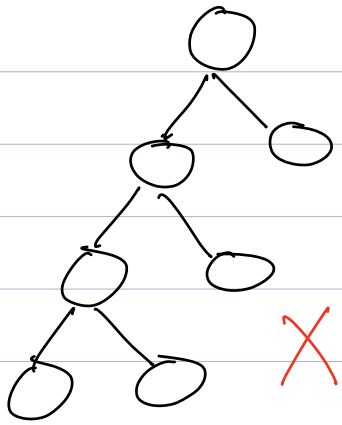
CBT

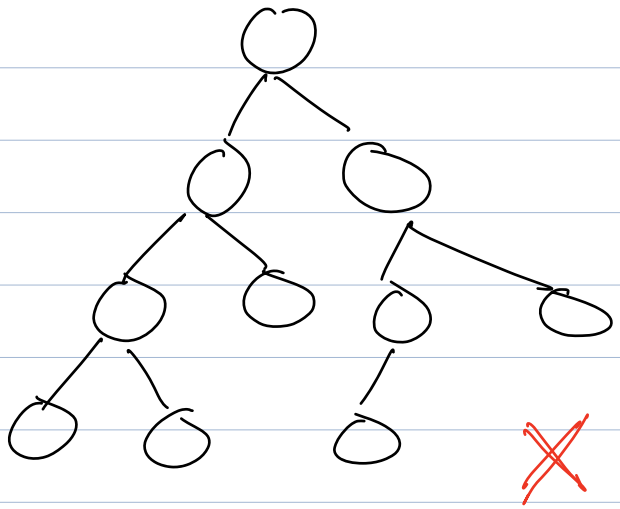
[Complete Binary Tree]



→ All levels are completely filled except the last level.

→ In the last level, insertion would happen from left to Right





Heaps are a Binary Tree

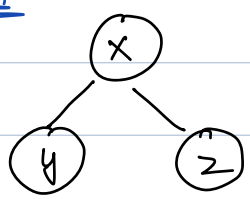
Structural.

↓

Complete Binary Tree

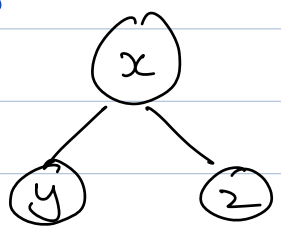
Order of Element

Max heap



$x \geq y \;\&\&\; x \geq z$

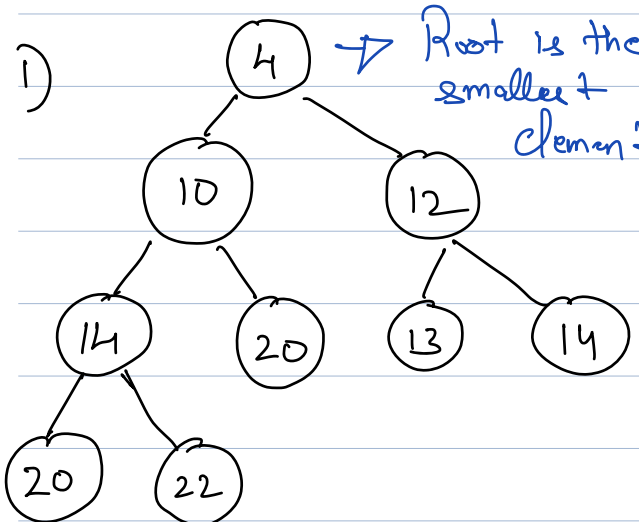
Min heap



$x \leq y \;\&\&\; x \leq z$

---

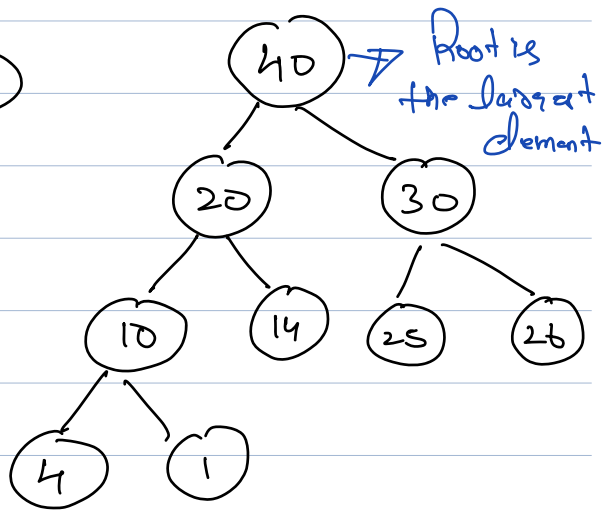
1)



Root is the smallest element

Min heap

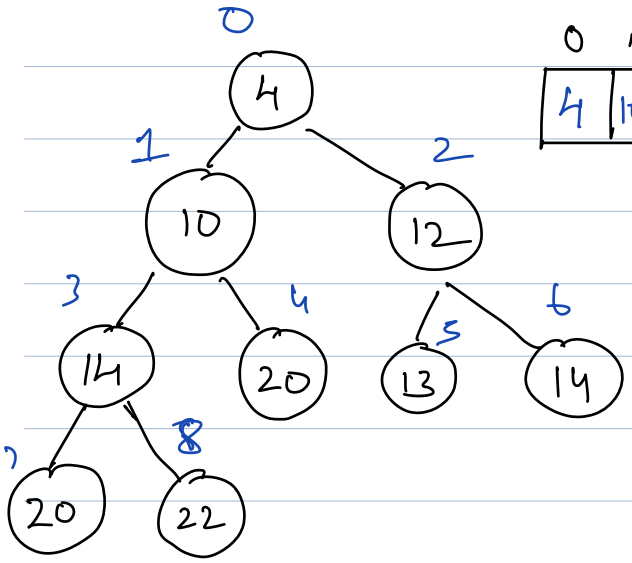
2)



Root is the largest element

Max heap

Array implementation of Heaps



0	1	2	3	4	5	6	7	8
4	10	12	14	20	13	14	20	22

Parent index	Children ind
0	1, 2
3	7, 8
2	5, 6
(i)	(2i+1, 2i+2)

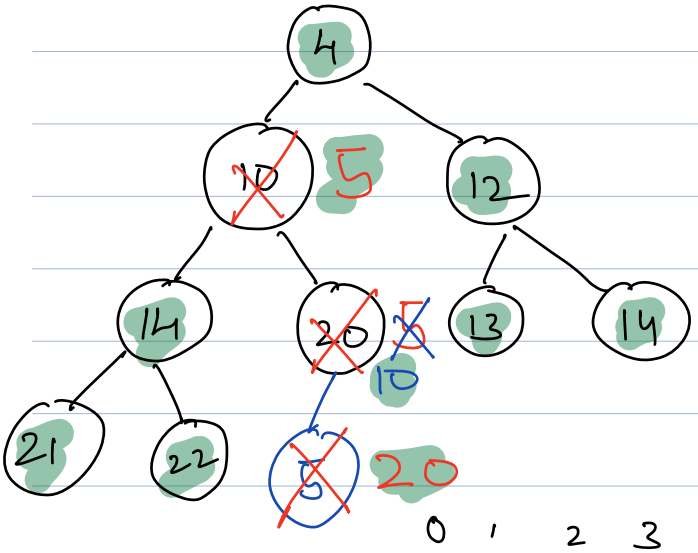
Child index	Parent ind
8	3
4	1
2	0
(i)	$(\frac{i-1}{2})$

$i \Rightarrow 2i+1$

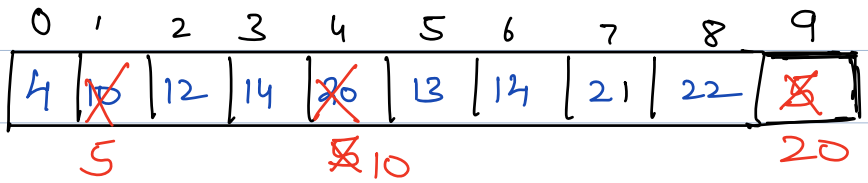
~~$\times$~~ $\Rightarrow 2i+1$

$\frac{x-1}{2} \Rightarrow i$

Insertion ⇒ You already have a heap & you want to insert a new element



insert (5)



index	Parent	
9	4	arr[4] ≤ arr[9] 20 ≤ 4
4	1	arr[1] ≤ arr[4] 10 ≤ 5
1	0	arr[0] ≤ arr[1] 4 ≤ 5

TC : O(log n)

```
int heap[];

int index = heap.length();
heap[index] => new_value;

int i => index

while (i != 0) {

    int parent => (i-1)/2;

    if (heap[parent] > heap[i]) {

        swap(parent, i);

        i => parent;

    } else

        break;

}
```

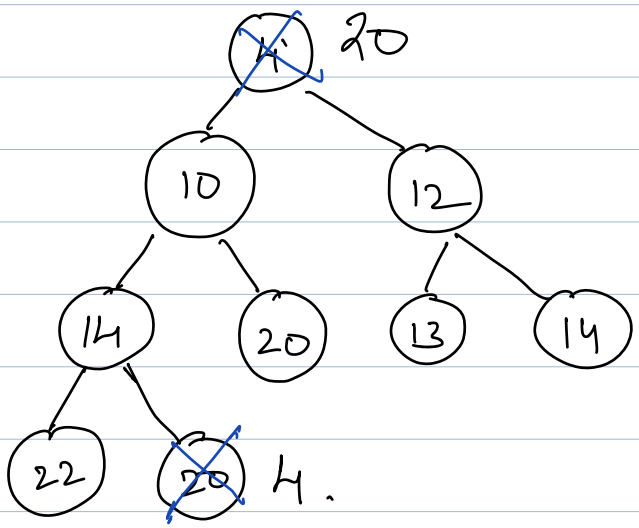
TC : O(log n)

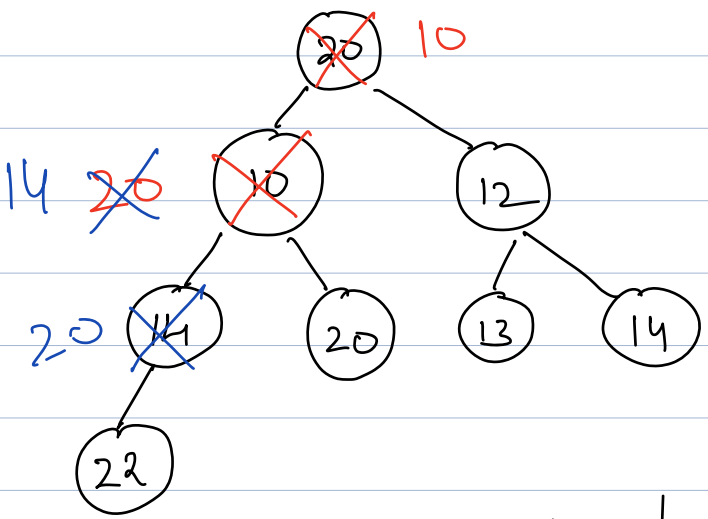
SC : O(1)

Extract min

↓

Return the min & remove it.





size ⇒ heap.length

swap (0, size-1)

size --

index	children	min (0, i, 2)
0	1, 2	⇒ arr[0]
1	3, 4	10 ≠ 20
3	7	

TC: O(log n)

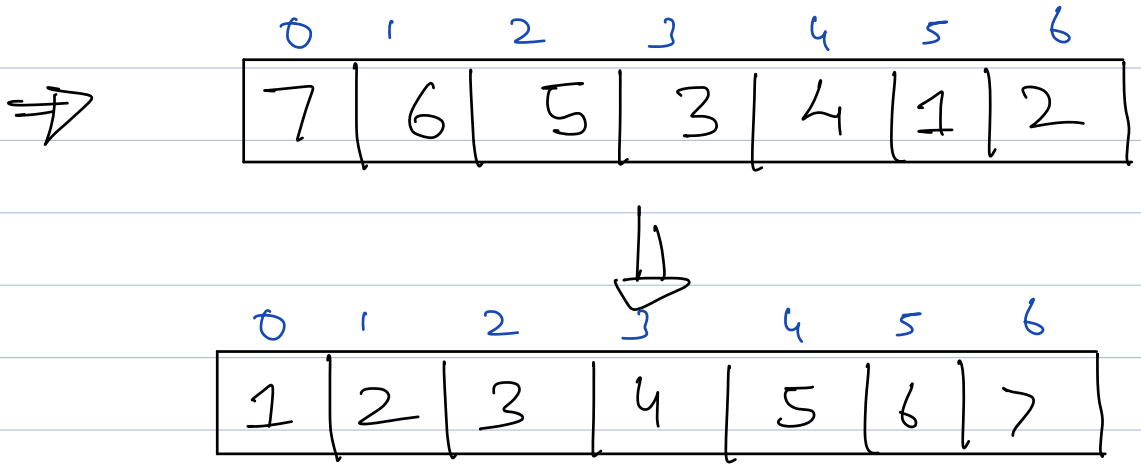
Code: H.W

⇒ swap (0, n-1)

heapify (heap[], 0);

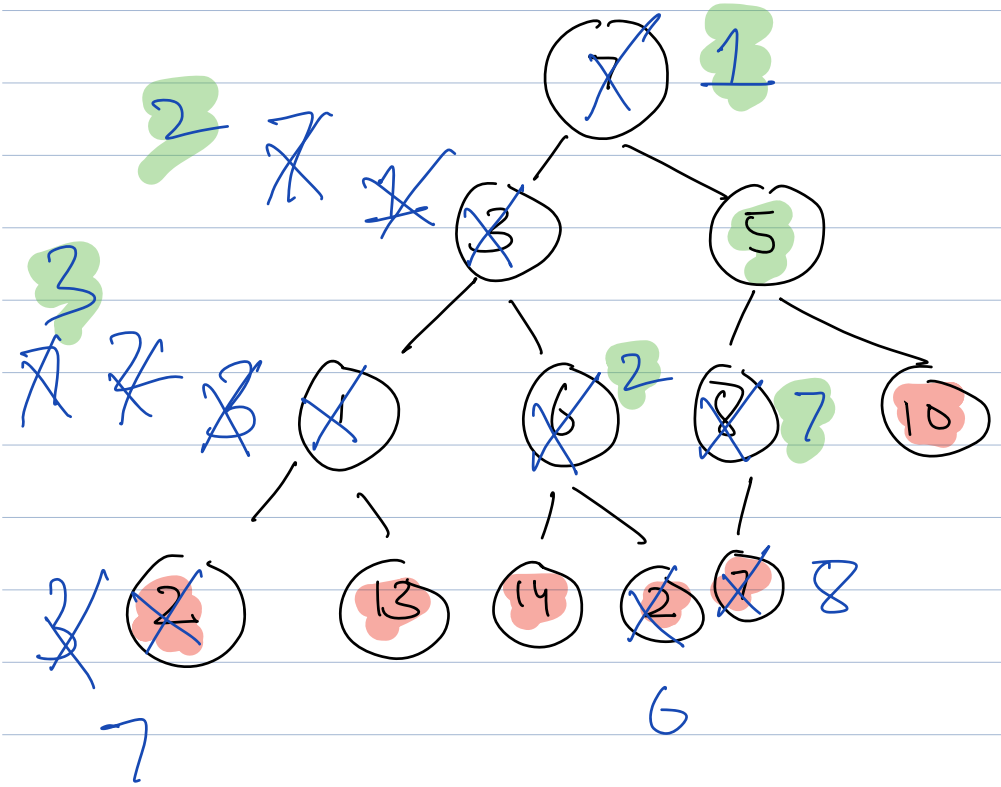
Construct a heap

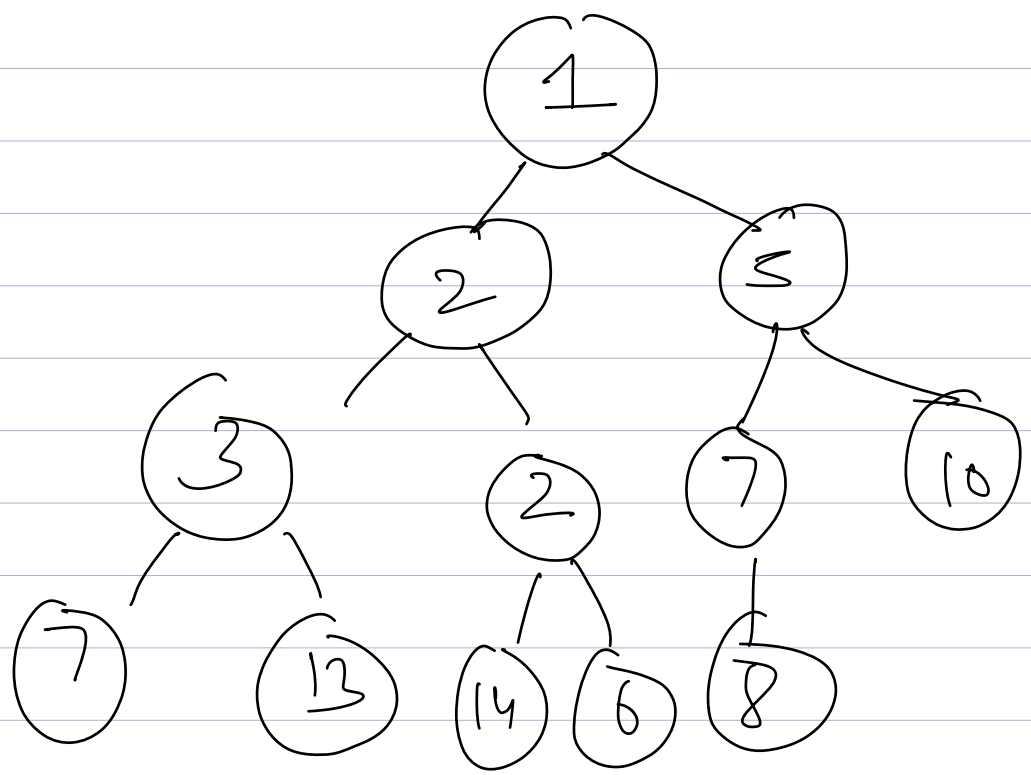
Min heap

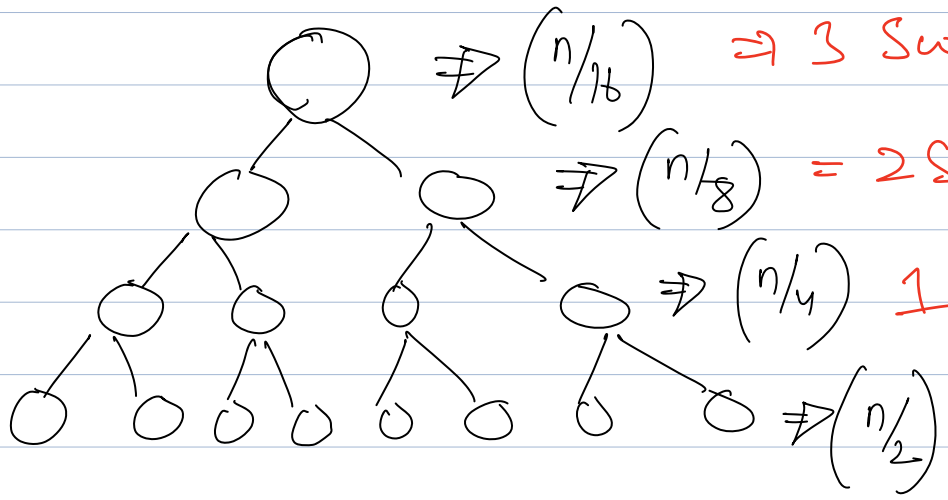


TC: O(n log n)

7 3 5 1 6 8 10 2 13 14 2 7.







Total Swaps

$$\Rightarrow \frac{n}{2} \times 0 + \frac{n}{4} \times 1 + n/8 \times 2 + n/16 \times 3 + n/32 \times 4 \ldots\ldots$$

$$\Rightarrow \frac{n}{2}\left[\frac{1}{2} + \frac{2}{4} + 3/8 + \frac{4}{16} \ldots\ldots\right]$$

$$S$$

$$\Rightarrow \left(\frac{n}{2} \times S\right)$$

⇒ AGP

APAGP

$$S \Rightarrow \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{5}{32} \cdots$$

$$\frac{S}{2} \Rightarrow \frac{1}{4} + \frac{2}{8} + \frac{3}{16} + \frac{4}{32} + \frac{5}{64} \cdots$$

$$S/2 \Rightarrow \frac{1}{2} + 1/4 + 1/8 + 1/16 + 1/32 \cdots$$

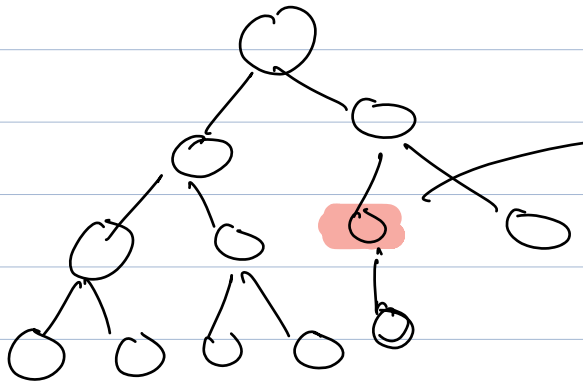
$$S/2 \Rightarrow 1 \qquad \Rightarrow \frac{a}{1-r}$$

$$\boxed{S \Rightarrow 2} \qquad \Rightarrow \frac{1/2}{1-1/2} \Rightarrow 1$$

$$\text{Total Swaps} \Rightarrow \left(\frac{n}{2} \times S\right)$$

$$\Rightarrow \frac{n}{2} \times 2 \simeq n$$

$$TC \Rightarrow O(n)$$



last non leaf node

index ⇒ last leaf node

last non leaf node ⇒ $\left(\frac{index-1}{2}\right)$

$n-1$ ⇒ $\frac{n-1-1}{2}$ ⇒ $\frac{n-2}{2}$

⇒ $\frac{n}{2}-1$

```
for (i = n/2 - 1 ; i ≥ 0 ; i--) {
    heapify (heap[], i);
}
```

```
void heapify (int heap[], int index, size ) {
    while ((2*index + 1) < size) {
        int min_val = min (heap[index],
                           heap[2*index+1],
                           heap[2*index+2])
        if (min_val == heap[index])
            return;
        else if (min_val == heap[2*index+1]) {
            swap (index, 2*index+1);
            index = 2*index+1;
        else {
            swap (index, 2*index+2);
            index = 2*index+2;
    }
}
```

Check if right exists

Priority - Queue