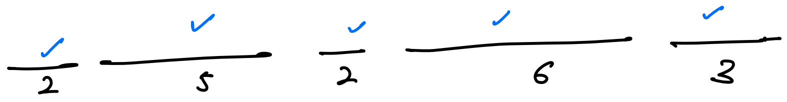
Connecting the ropes



You can connect two ropes together.
Cost of connecting two ropes → sum of length of ropes.
Find the minimum cost of connecting all the ropes.

cost.

2 6 = 8 8

2 8 = 10 10

5 10 = 15 15

3 15 = 18 18

51

↓

~~2~~, ~~2~~, ~~3~~, ~~5~~, ~~6~~

cost

2 + 2 = 4 4

3 + 4 = 7 7

5 + 6 = 11 11

7 + 11 = 18 18

40

idea: Always pick two of the smallest ropes.

			Cost.
2, 2, 3, 5, 6.			4
4, 3, 5, 6	$\xrightarrow{Sort}$	3, 4, 5, 6	7.
7, 5, 6	$\xrightarrow{Sort}$	5, 6, 7	11
11, 7	$\xrightarrow{Sort}$	7, 11	18
18.			

D.S → insert → $\log_2 N$

D.S → extractMin → $\log_2 N$

⇓

Heap

Heap Data Structure

↓

Binary Tree

↓ Structure

↓ Order of Elements

Structure.

Complete Binary Tree [C.B.T]

⇓

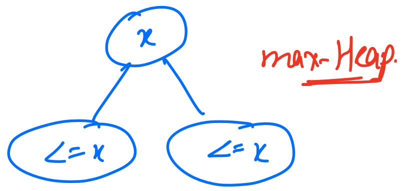
All levels should be completely filled. Last level can be exception.

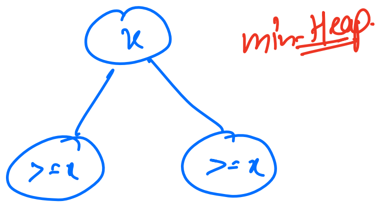
↓

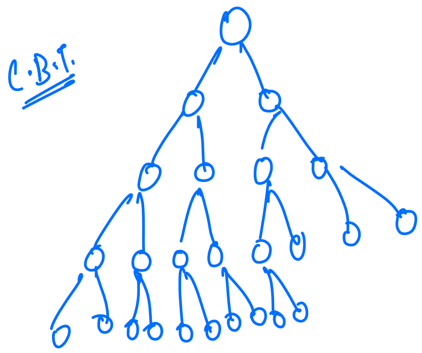
from left to right.

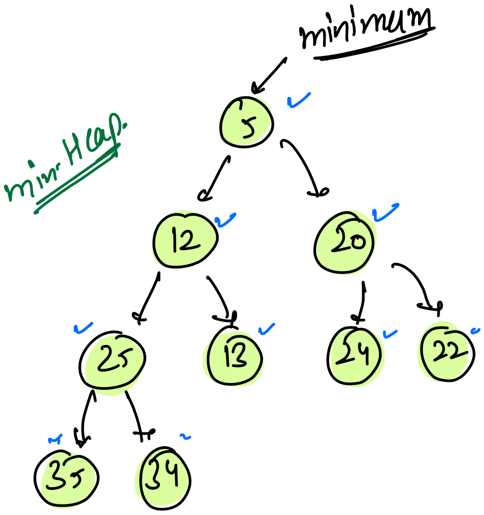
Order of Elements

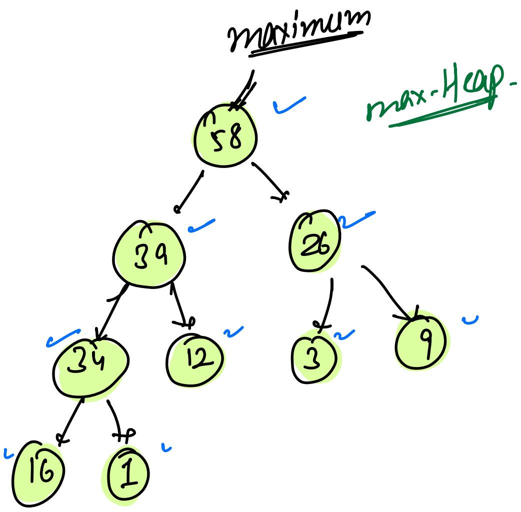
Heap Order Property [H.O.P]











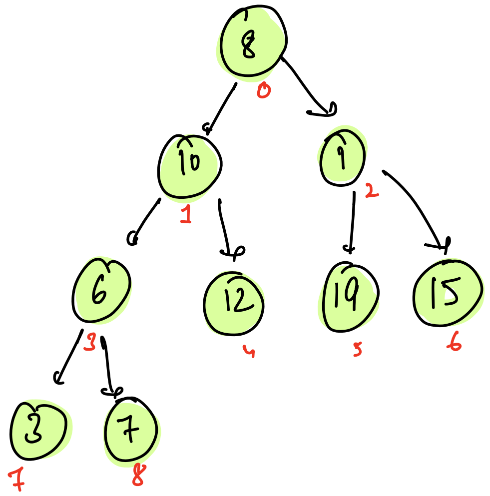
{ getMin() / getMax()

⇓

O(1) }

Array Implementation Of Trees

↓
C.B.T.



8	10	1	6	12	19	15	3	7
0	1	2	3	4	5	6	7	8

0 → 1 (2*0 + 1) = 1
0 → 2 (2*0 + 2) = 2

1 → 3 (2*1 + 1) = 3
1 → 4 (2*1 + 2) = 4

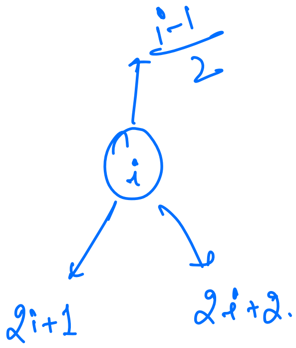
2 → 5 (2*2 + 1) = 5
2 → 6 (2*2 + 2) = 6

3 → 7 (2*3 + 1) = 7
3 → 8 (2*3 + 2) = 8

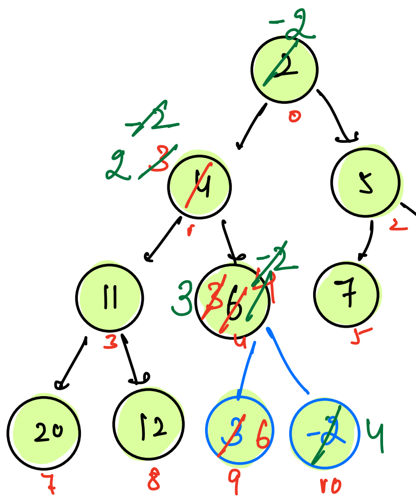
$i \to 2i+1$

$i \to 2i+2$

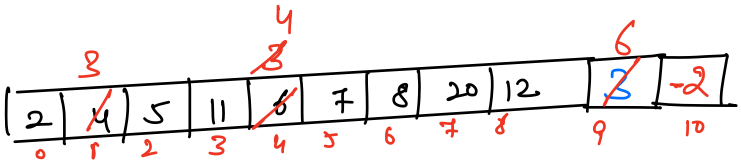
$C.i \longrightarrow \frac{(ci-1)}{2}$



min-Heap. (insertion)



insert(3)

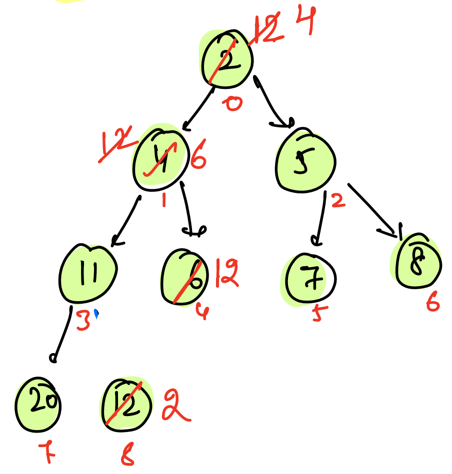


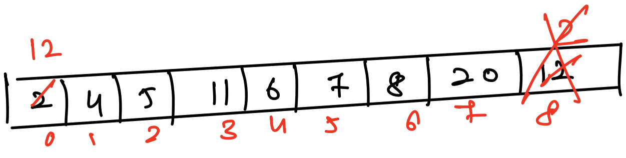
i	parent.	arr[parent] ≤ arr[child]
9	(9-1)/2 = 4.	No
4	(4-1)/2 = 1	No
1	(1-1)/2 = 0	Yes

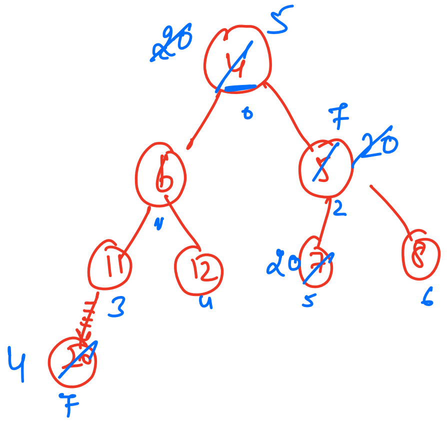
T.C - $O(H) \rightarrow O(\log_2 N)$

```
heap [ ] ;
heap. insert ( val ) ;   //insert at last.
 i = heap.size() - 1 ;
 while ( i != 0 ){
              pi = (i-1) / 2;
              if ( heap (pi] > heap (i] ){
                      swap ( heap[pi], heap [i])
                      i = pi
              }
              else{
                      break;
              }
 }
```

Extract - Min







```
swap ( heap [0], heap[heap.size() - 1] );
    size --;
    heapify ( heap [], 0);

void heapify ( heap [], i ) {
        while ( 2i+1 < N ){
            x = min( heap[i], heap[2i+1], heap[2i+2]);
            if (x == heap[i]) {return }
            else if (x == heap[2i+1]) {
                swap( heap[i], heap[2i+1])
                i = 2i+1
            }else{
                swap (heap[i], heap[2i+2]);
                i = 2i+2
            }
        }
}
```

T.C → O(H)

→ $O(\log_2 N)$

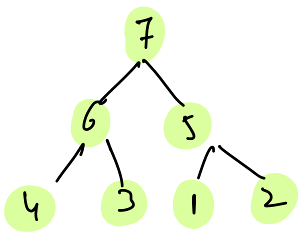
```
Priority Queue< Integer> pq = new PriorityQueue<>();
        pq.add(val);      // Insert
        pq.peek();        // get min
        pq.remove();  // extract Min
```

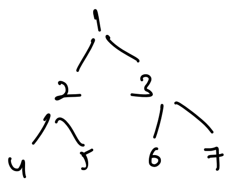
Build a heap.

7	6	5	4	3	1	2
0	1	2	3	4	5	6

Sort

1	2	3	4	5	6	7

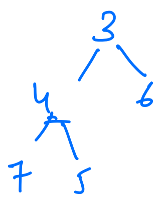




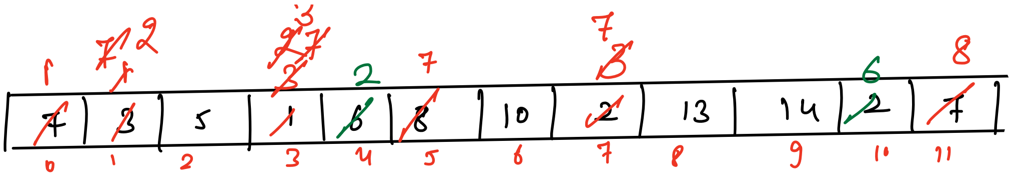
T.C → O(N logN)

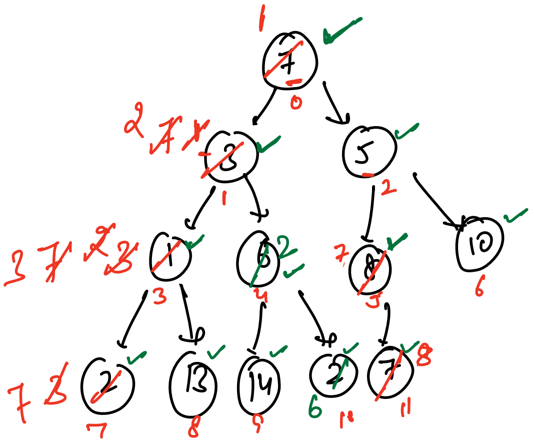
Insert() for all the elements

3	4	6	7	5



T.C → O(N logN)





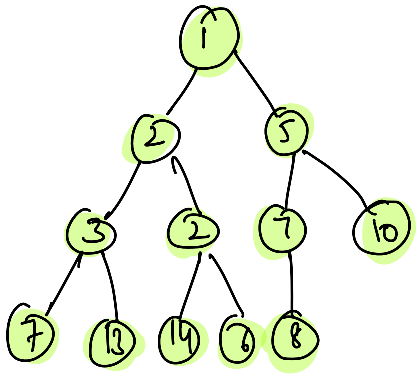
leaf node → H.O.P will always valid.

{ first non-leaf node → $\frac{N}{2} - 1$ }

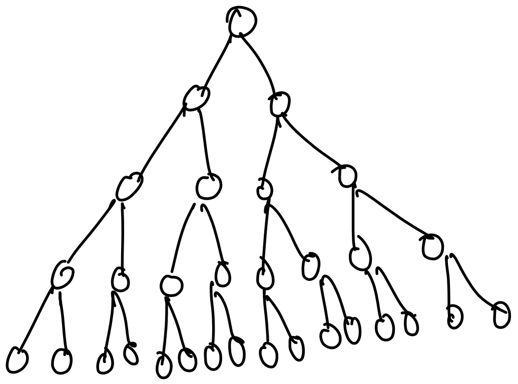
$i = N-1$

$pi = \frac{(i-1)}{2} = \frac{(N-1-1)}{2}$

$= \frac{N-2}{2} = \frac{N}{2} - 1$



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



swaps.

1

⋮

$N/16 \Rightarrow 3$

$N/8 \Rightarrow 2$

$N/4 \Rightarrow 1$

$N/2 \Rightarrow 0$

$$\rightarrow \frac{N}{2} * 0 + \frac{N}{4} * 1 + \frac{N}{8} * 2 + \frac{N}{16} * 3 + — —$$

$$= \frac{N}{2}\Big[\underbrace{0 + \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{5}{32} + —}_{S}\Big]$$

A.G.P.

$$S = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{5}{32} + — — —$$

$$\frac{S}{2} = \frac{1}{4} + \frac{2}{8} + \frac{3}{16} + \frac{4}{32} + — — —$$

$$- \quad - \quad - \quad - \quad -$$

$$\frac{S}{2} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + —$$

$$\frac{S}{2} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + — — \cdot\infty$$

$$S_\infty = \frac{a}{1-r}$$

$$a \to \frac{1}{2}, \quad r \to \frac{1}{2}$$

$$S = \frac{1/2}{1 - 1/2} = \frac{1/2}{1/2}$$

$$= 1.$$

$$\frac{S}{2} = 1$$

$$\boxed{S = 2.}$$

$$\boxed{T.C \rightarrow O(N)}$$

→ Building a Heap

{in-built Heap}

$A = [\underset{0}{5}, \underset{1}{2}, \underset{2}{7}]$

$[i < j \leq k]$

i j k

0, 1, 1 → $5 \neq 2$

0, 1, 2 → $5 = 5$

0, 2, 2 → $7 = 7$

1, 2, 2 → $2 \neq 7$

$A[i] \wedge A[i+1] \wedge \text{--}\ A[j-1]$

$= A[j] \wedge A[j+1] \wedge \text{--}\ A[k]$

[Xor of subarray from i, j-1 =
Xor of subarray from j to k.]

[count = 2]

$$\begin{array}{r} 0101 \\ 10 \\ \hline 111 \end{array}$$