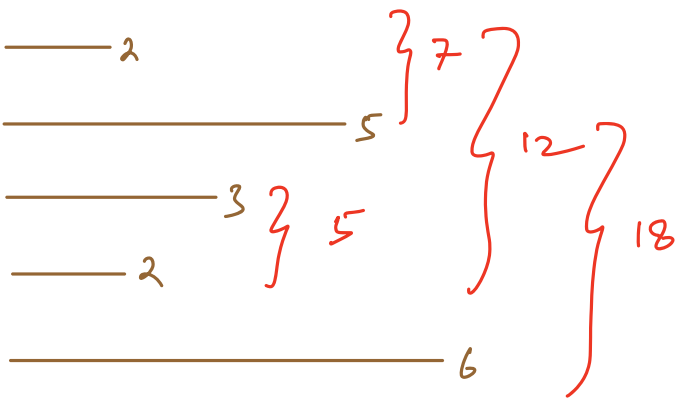
Heaps 1 : Introduction

Question

Given N ropes of different sizes.

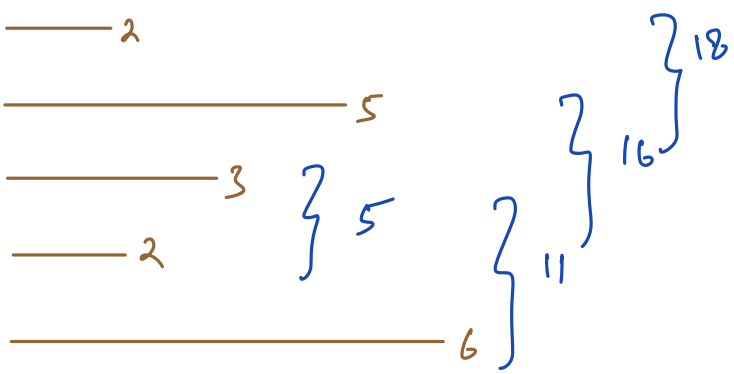
In one operation we can connect 2 ropes & the cost is the sum of length of both ropes.

find min. cost to connect all ropes.



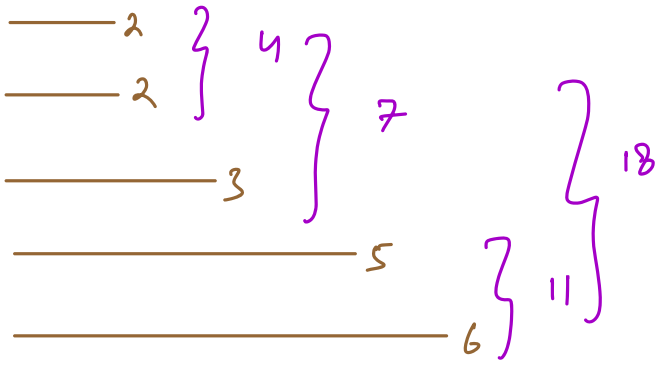
cost = 7 + 5 + 12 + 18

= 42



cost = 5 + 11 + 16 + 18

= 50



cost = 4 + 7 + 11 + 18

= 40

let say we have 3 ropes of length → $x < y < z$

Case 1		Case 2		Case 3
$x+y$ + $(x+y)+z$	<	$x+z$ + $(x+z)+y$	<	$y+z$ + $(y+z)+x$

⇒ connect smaller length ropes to get min. cost.

Soln : sort & connect ropes

2 2 3 5 6

4

7

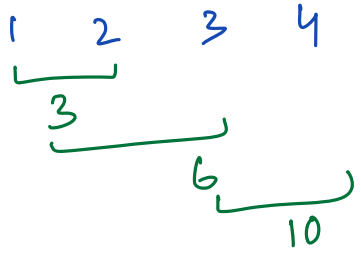
After every operation of connecting 2 ropes, insert new rope in its sorted position. ↖ Insertion Sort

total TC = $O(N^2)$

SC = $O(1)$

Quiz

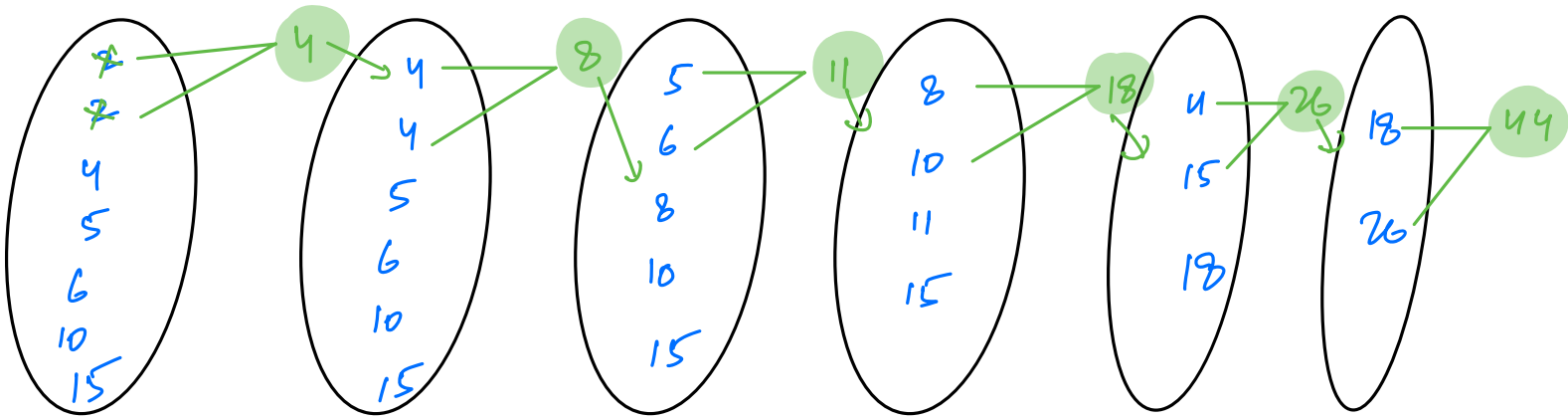
Min. cost for connecting [1 2 3 4]



cost = 3 + 6 + 10

= 19

Heaps → insert ()
getMin()
> TC = O(logN)



TC per operation = $O(3 \log N) = O(\log N)$

total TC = $O(N \log N)$

SC = $O(N)$

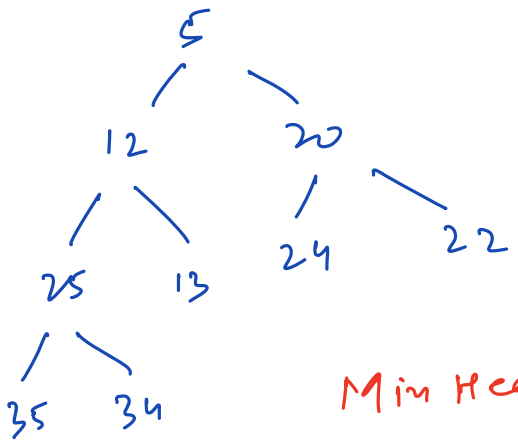
Heaps / Priority Queue

1. Structure → Complete Binary tree

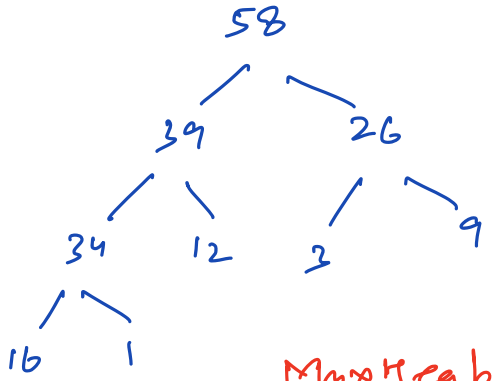
   All levels are completly filled except maybe the last level. Elements in last are from left to right.

2. Types
   - → MinHeap — # nodes, data <= children's data
   - → MaxHeap — # nodes, data >= children's data

3. No relation b/w left & right subtree.

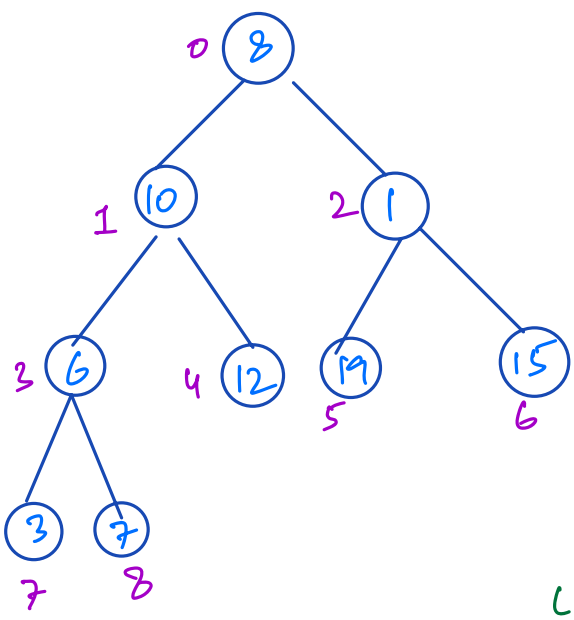


Min Heap



Max Heap

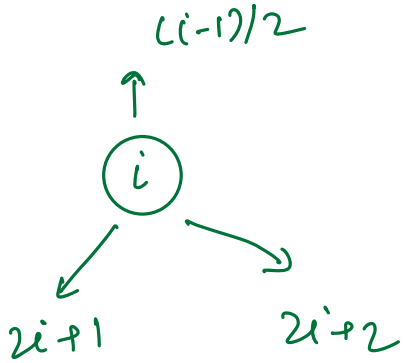
Array implementation of complete binary tree



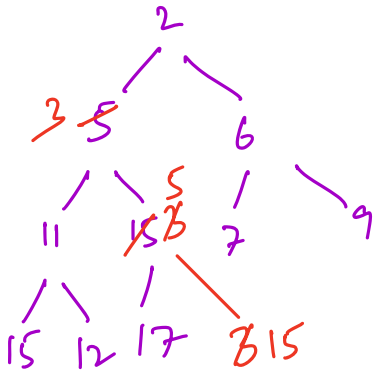
8	10	1	6	12	19	15	3	7
0	1	2	3	4	5	6	7	8

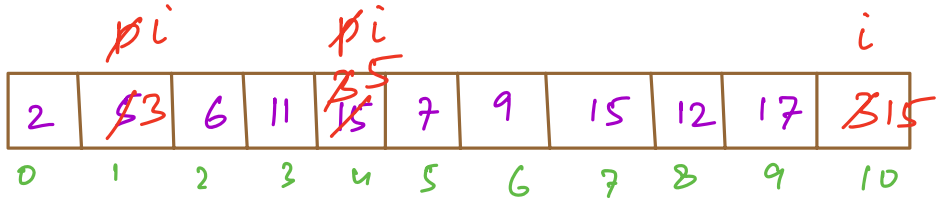
If

- left child = $2i+1$
- right child = $2i+2$
- parent = $(i-1)/2$



Insertion





insert(3)

MinHeap

i = 10 4 1

p = (i-1)/2 = 4 1 0

TC = O(H) = O(logN)

[H = logN for complete BT]

Code

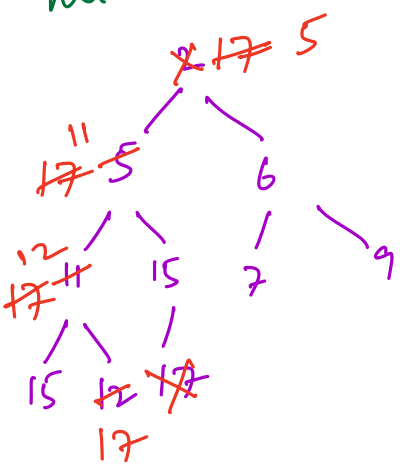
```
ArrayList<int> heap;  // minHeap

// insert x in heap

heap.append(x);

i = heap.size() - 1;

while ( i > 0) {
    p = (i-1)/2 ;
    if( heap[p] > heap[i] ) {
```

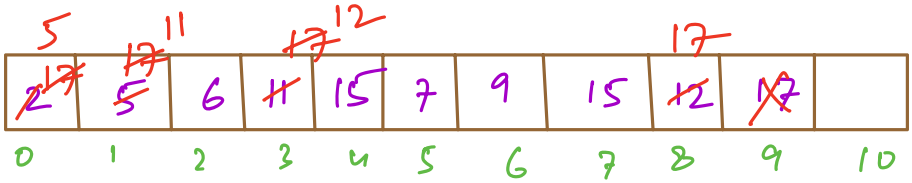
```
        swap( heap[p], heap[i])
        i = p;
    }
    else {
        break
    }
}
```

TC : O(logN)

getMin()

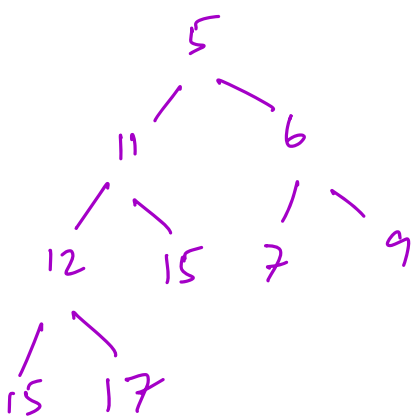
minHeap





min → root → heap[0]

i =	0	1	3	8
le = 2i+1 =	1	3	7	17
rc = 2i+2 =	2	4	8	18



TC = O(logN)

Code

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

```
void heapify ( heap [], int i) {
    while( 2i+1 < heap.size() ) { // need to handle edge case when right child is not present
        x = min ( heap[2i+1], heap[2i+2]);
        if ( heap[i] <= x ){
            break;
        }
        else if (heap[2i+1] < heap[2i+2] ) {
            swap( heap[i], heap[2i+1]);
            i = 2i+1
        }
        else {
            swap( heap[i], heap[2i+2]);
```

```
        i = 2i+2
    }
}
}
```

TC = O(log N)

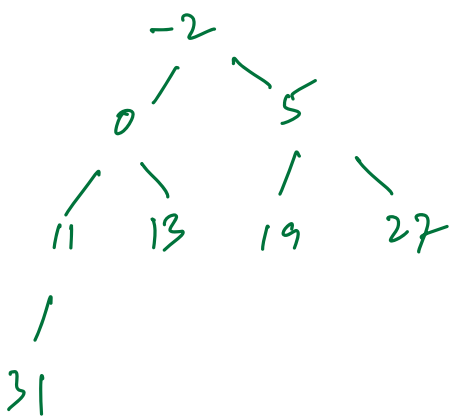
Build a Heap

Given an array, create a min heap out of it.

[5 13 -2 11 27 31 0 19]

Idea 1 : sort the array

[-2 0 5 11 13 19 27 31]



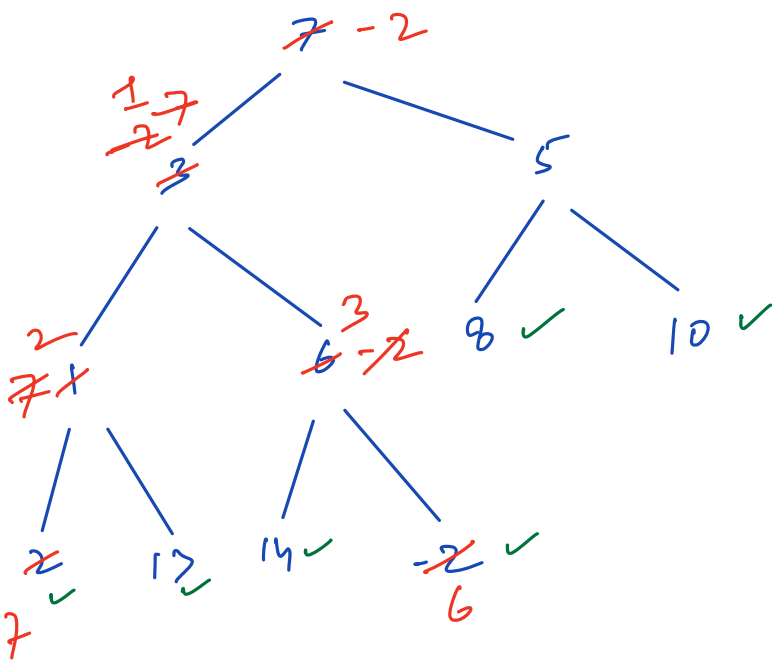
TC = O(N log N)

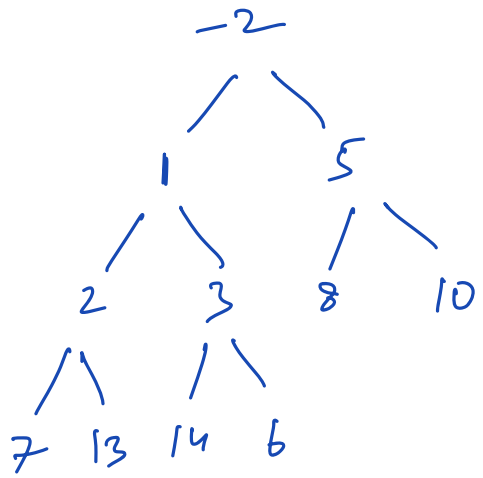
Idea 2 : insert elements in heap one by one.

$TC = O(N \log N)$

How to build in linear time ?

[7 3 5 1 6 8 10 2 13 14 -2]



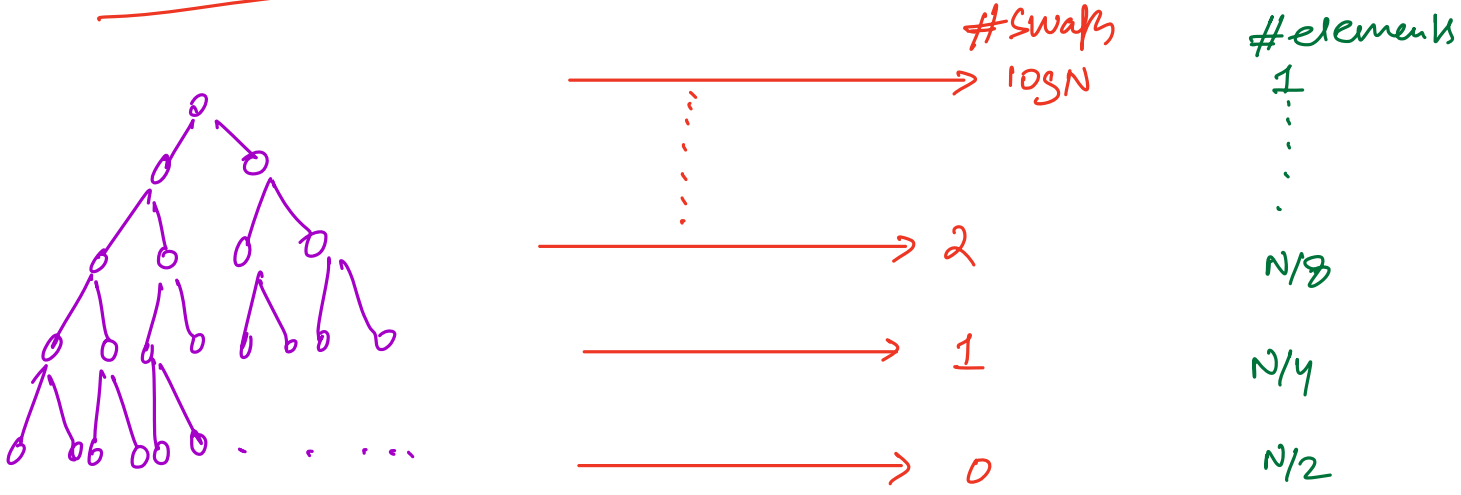


There are $\sim N/2$ leaves in a complete binary tree.

Code

```
for( i = n/2  to  0) {
    heapify( heap, i)
}
```

Calculate TC



total iterations $= 0 \times \frac{N}{2} + 1 \times \frac{N}{4} + 2 \times \frac{N}{8} + \cdots\cdots$

$$= \sum i \times \frac{N}{2^{i+1}}$$

$$= \frac{N}{2} \underbrace{\sum \frac{i}{2^i}}_{=S}$$

$$S = \frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{2^3} + \cdots\cdots$$

$$S * \frac{1}{2} = \frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \cdots\cdots$$

$$\frac{S}{2} = \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots\cdots$$

sum of infinite GP $= \frac{a}{1-r}$

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

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

total iterations $= \frac{N}{2} \times 2 = N$

$TC = O(N)$