Welcome :)

Agenda:

1 ques^n
Heap / Priority Queue <u>D.S</u>
Opera^{n}s
Build a heap
1 ques^n.

Q Given N ropes with their length
Cost of connecting 2 ropes = sum of length of both
Find min. cost to connect all ropes.

2
5 } 7
2
6 } 8
3 } 11

7 + 11 → 18

final cost = 7 + 8 + 11 + 18
= 44

2
5 } 7
2 } 9
6
3 } 9

9 + 9 → 18

final cost
⇒ 7 + 9 + 9 + 18
⇒ 43

2
2 } 4
3 } 7
5
6 } 11

7 + 11 → 18

total cost
⇒ 4 + 7 + 11 + 18
⇒ <u>40</u>

$x < y < z$

S1 $\quad x+y \quad x+z \quad y+z$

S2 $\quad (x+y)+z \quad (x+z)+y \quad (y+z)+x$

$\underline{2x+2y+z} \quad \underline{2x+2z+y} \quad \underline{2y+2z+x}$

Observations ⇒ Connect smaller length ropes first.

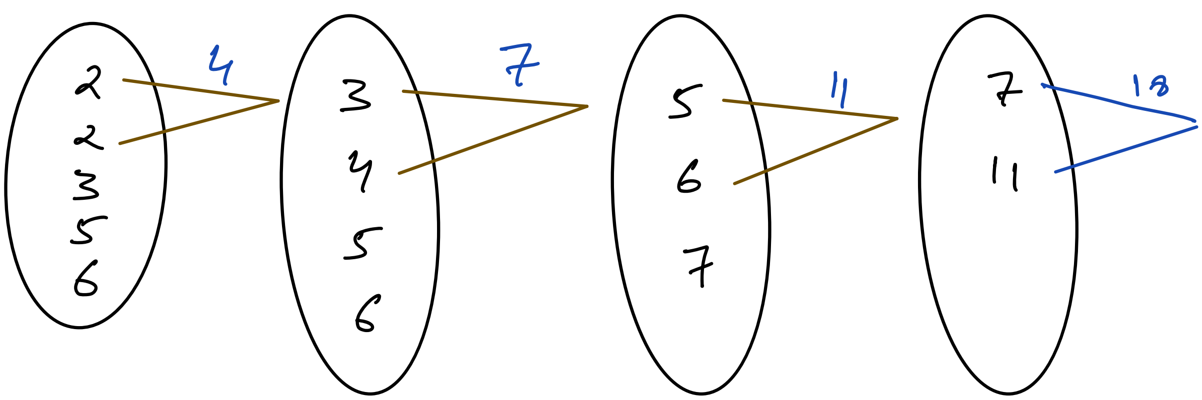
In order to get smallest

⇒ Sort the data

But we need to sort at every step

⇒ Can be done using insertion sort. ⇒ $O(N^2)$

D.S → inserting, getMin() > $O(\log N)$



T.C ⇒ $3 * O(\log N) * (N-1) \Rightarrow \simeq O(N \log N)$

S.C ⇒ $O(N)$

Heap / Priority Queue

1) It should be a complete binary tree.

2) Heap Order property ⇒ 1. Max Heap

⇒ ∀ nodes

node.data ≥ children.data

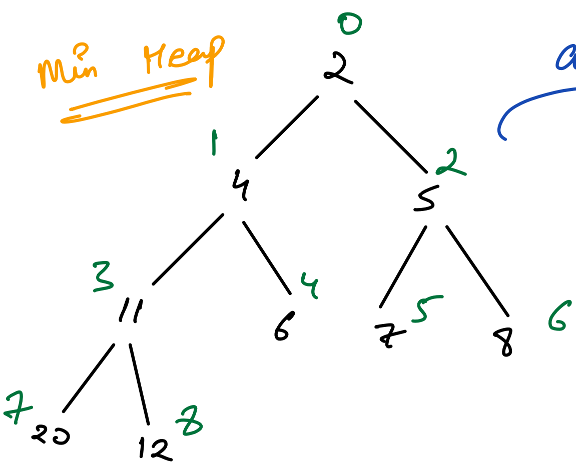
2. Min Heap

⇒ ∀ nodes

node.data ≤ children.data

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

Min Heap



array ← only possible b/c it is a complete binary tree.

0	1	2	3	4	5	6	7	8
2	4	5	11	6	7	8	20	12

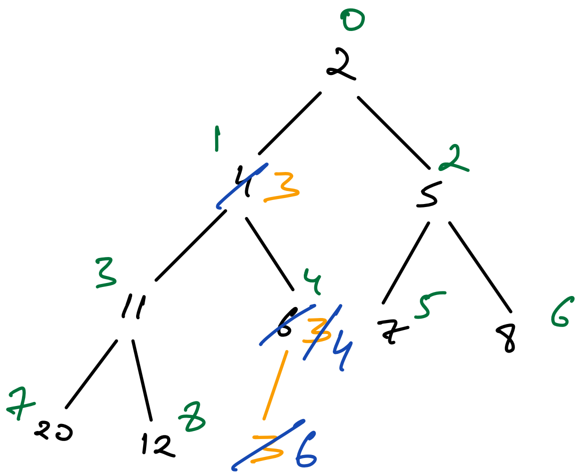
∀ nodes i

left child → $2*i + 1$

right child → $2*i + 2$

Parent → $(i-1)/2$

Insertion in Heap



insert (3)

Heapify $\Rightarrow$ Maintaining property of heap after any operaⁿ.

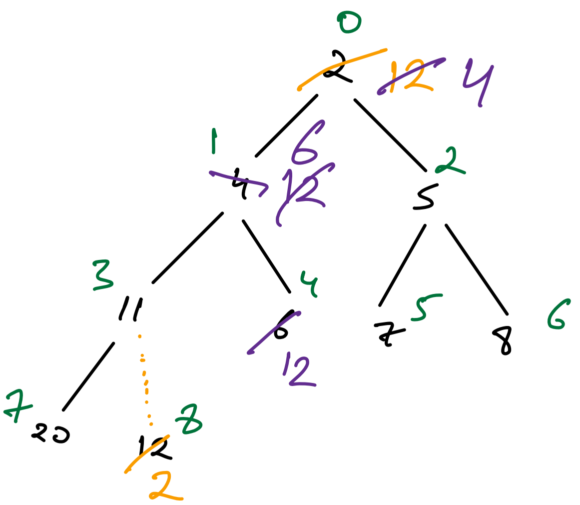
T.C $\Rightarrow$ O(log N)

Code

```
A[N] = x        i = N        N += 1

while ( i > 0 )
{
    p = (i-1)/2      // parent
    if ( A[p] > A[i] )
    {
        swap ( A[p], A[i] )
        i = p
    }
    else
        break ;
}
```

Get Min()



1. Swap first & last
2. Heapify.

T.C $\Rightarrow$ O(log N)

Code

```
ans = A[0]          A[0] = A[N-1]          N - = 1
i = 0
while ( i < N)
{
  lc = 2*i + 1
  rc = 2*i + 2
  if ( rc < N )  // both child exist
  {
      x = min ( A[i] , A[lc], A[rc])
      if ( x == A[i] )     break;
      else if ( x == A[lc])
      {
           swap ( A[i], A[lc])
           i = lc
      }
      else
      {
           swap ( A[i], A[rc] )
           i = rc
      }
  }
```

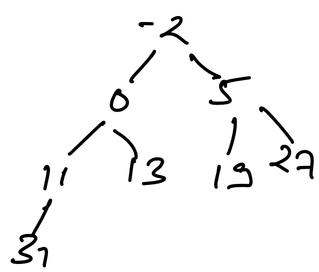
```
    elseif ( lc < N )   // left child exist.
    {
        x = min ( A[i], A[lc] )

        if ( x == A[lc] )
        {
            swap( A[i], A[lc] )
            i = lc
        }
        else
            break
    }
    else
        break;
}
return ans;
```

Build Heap

5 13 -2 11 27 31 0 19

Sol 1 Sort the array



-2 0 5 11 13 19 27 31

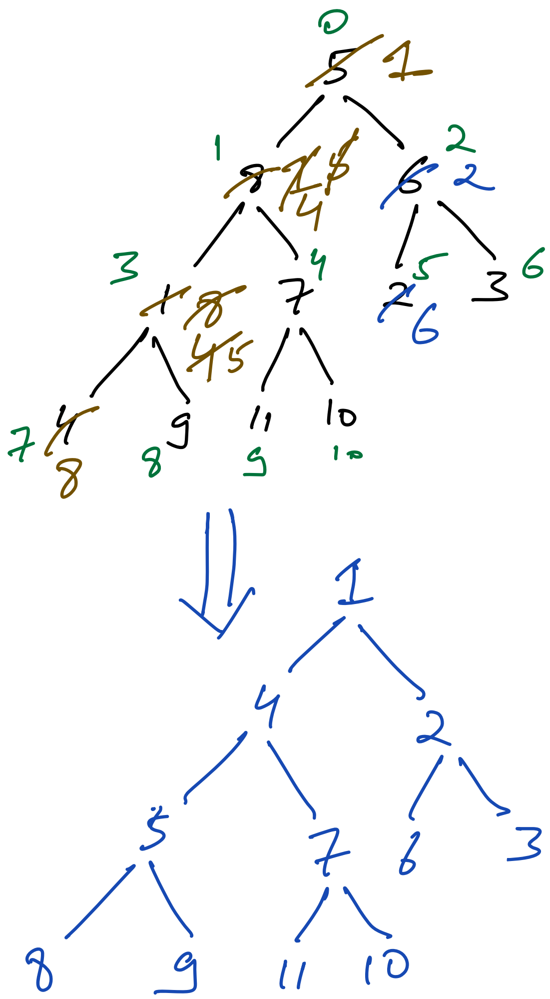
T.C ⇒ N log N

Sol 2 call insert (A[i]) for every element

T.C ⇒ N log N

Sol3

A : 5 8 6 1 7 2 3 4 9 11 10



index of last parent $= \frac{N}{2} - 1$

index $\xrightarrow{\text{heapify}}$ 4 → 0

Heapify from last parent to first parent

T.C ⇒ $O(N)$

⇒ last level of perfect b.T has N/2 nodes.

Max # swaps for

2nd last level ⇒ 1 ⇒ N/4

3rd last level ⇒ 2 ⇒ N/8

4th last level ⇒ 3 ⇒ N/16

⋮

Total swaps ⇒ $\frac{N}{2} * 0 + \frac{N}{4} * 1 + \frac{N}{8} * 2 + \frac{N}{16} * 3 + \dots$

⇒ $\sum_{0}^{H} \frac{N}{2^{i+1}} * i$ ⇒ $\frac{N}{2} * 2$ ⇒ N ⇒ $O(N)$

Q Merge N sorted arrays

⇒ Merge 2 sorted arrays. ⇒ 2 pointers.
" 3 " " ⇒ 3 pointers.
" N " " ⇒ N pointers.

Sol 1 ⇒

2	3	11	15	20	
1	5	7	9		
0	2	4			
3	4	5	6	7	8
-2	5	10	20		

⇒ Add 0^{th} element of every array in min heap

⇒ Get min ele.

⇒ Insert next element from same array from which you removed the smallest element

⇒ Continue this process.

T.C ⇒ $\underbrace{N}_{\text{build heap}} + X \log(N)$

X : total # of elements in all arrays.