Welcome ☺

Agenda: 1 ques^n
Heap / Priority Queue D.S
Operations
Build Heap
2 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 —— } ⇒ 8 } ⇒ 8+3 = 11 } 11+7 = 18
6 ————————
3 ———

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

2 ——
5 ————— } ⇒ 7 } 7+2=9 } 9+9 = 18
2 ——
6 ———————— } ⇒ 9
3 ———

final cost = 7 + 9 + 9 + 18
= 43

3 ———
2 —— } ⇒ 4 } 3+4 = 7 } 7+11 = 18
2 ——
6 ———————— } 11
5 ————

final cost = 4 + 7 + 11 + 18
= 40

Observation ⇒ Connect smaller length ropes first.

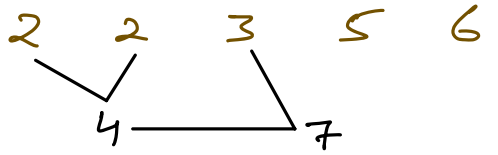
x (1) < y (2) < z (3)

S1 $x+y$ < $x+z$ < $y+z$

S2 $(x+y)+z$ (9) $(x+z)+y$ (10) $(y+z)+x$ (11)

In order to get smallest data.

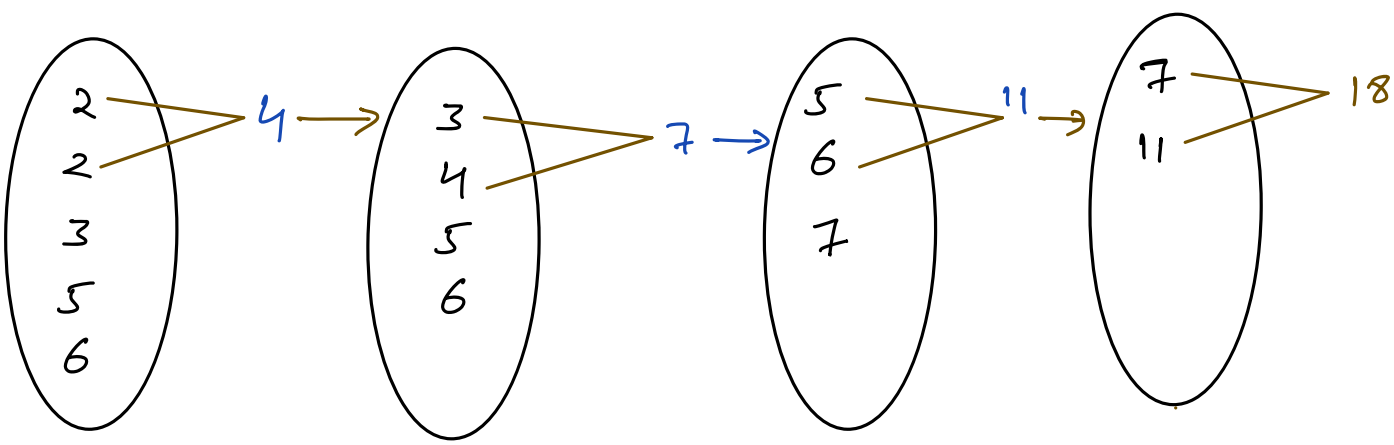
1) Sorting.



sort at every step to get the min. length each time.

can be done using Insertion sort ⇒ T.C = $O(N^2)$

DS → inserting → $O(\log(N))$
getMin() → $O(\log(N))$



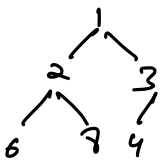
T.C ⇒ $3 * O(\log N) * N-1$ ⇒ $O(N \log N)$

S.C ⇒ $O(N)$

Heaps / Priority Queue

1) Structure ⇒ Complete binary tree

Complete binary tree → Every node has 2 children except maybe the last level. All nodes of last level are as left as possible.



2) Type of heaps

1) Min Heap → ∀ nodes

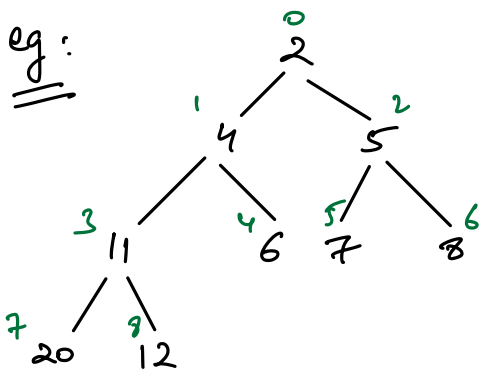
node.data ≤ children.data.

2) Max Heap → ∀ nodes

node.data ≥ children.data.

3) No relationship b/w left & right subtree.

eg:



Storing in array

2	4	5	11	6	7	8	20	12

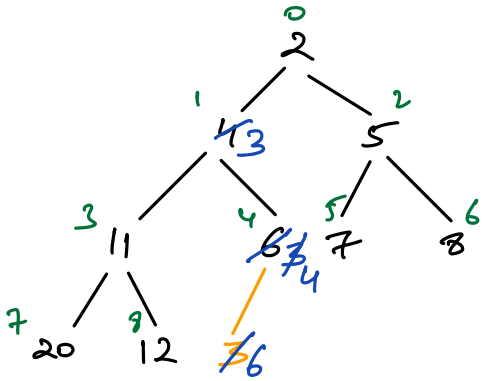
∀ nodes i

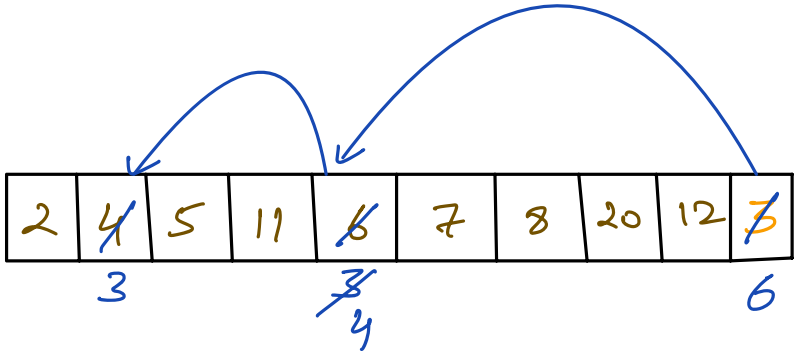
left child → $2i+1$

right child → $2i+2$

parent → $(i-1)/2$

Insertion in Heap





insert (3)

Heapify ⇒ Maintaining property of heap after any operation.

T.C → O(log N)

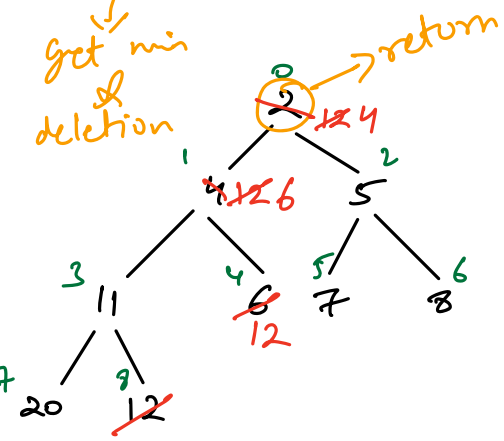
Code for Min Heap

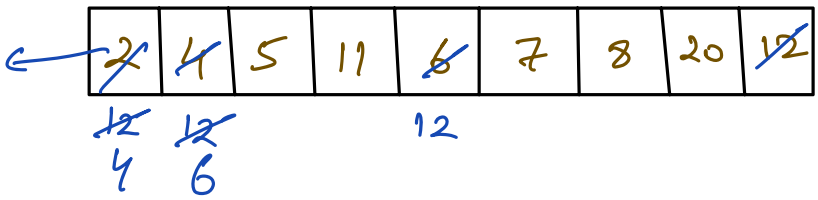
```
A[n] = x        i = N        N++ = 1

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

ExtractMin() [Min Heap]

Get min & deletion





T.C $O(\log N)$

Code

```
ans = A[0]    A[0] = A[N-1]    N--
i = 0
while ( i < N)
{
    lc = 2*i + 1
    rc = 2*i + 2
    if( rc < N) // both child exists
    {
        if ( A[lc] <= A[i] && A[lc] <= A[rc] )
        {
            swap ( A[lc], A[i] )
            i = lc
        }
        elseif ( A[rc] <= A[i] && A[rc] <= A[lc])
        {
            swap ( A[rc] , A[i])
            i = rc
        }
        else    break;
    }
    elseif ( lc < N) // only left child present
    {
        if ( A[lc] <= A[i] )
        {
            swap ( A[lc], A[i])
            i = lc
```

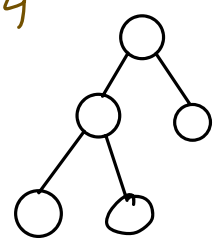
```
        }
    else        break
}
else        break
}
return ans
```

Build Heap

1) Insert all elements one by one → T.C ⇒ O(NlogN)

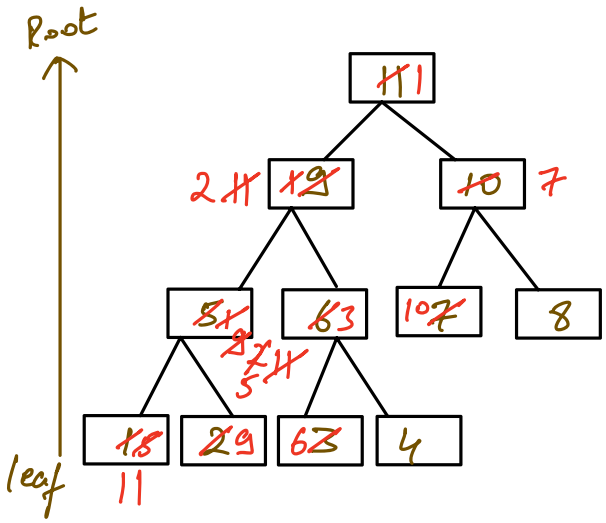
2) If all elements are known

N →	1	2	3	4	5	6	7
	1	1	2	2	3	3	4



leaves in CBT ⇒ $\frac{N+1}{2}$

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



Swaps per element
3
2
1
0

Total swaps = $\frac{N}{2} * 0 + \frac{N}{4} * 1 + \frac{N}{8} * 2 \cdots$

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

T.C ⇒ O(N)

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

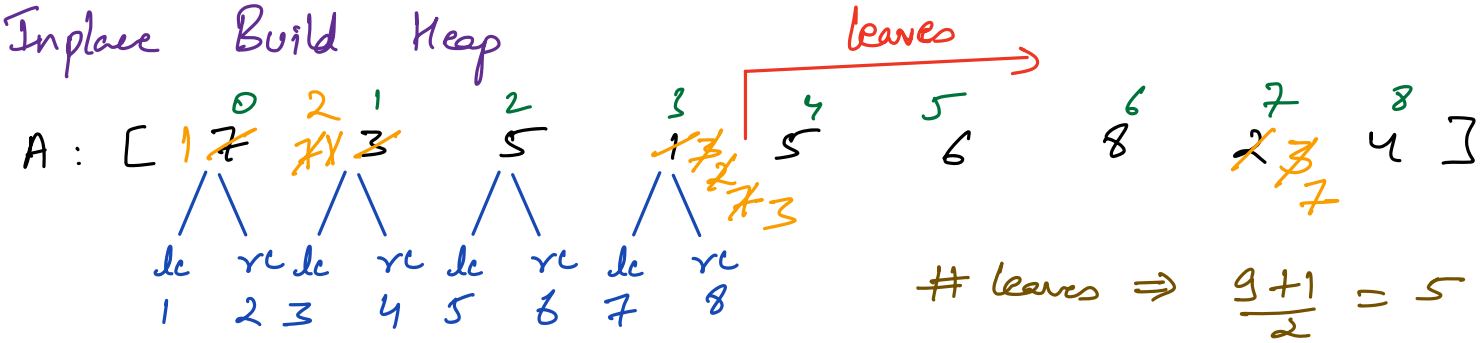
$$-S/2 = \frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \cdots\cdots$$

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

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

$$S = 2$$

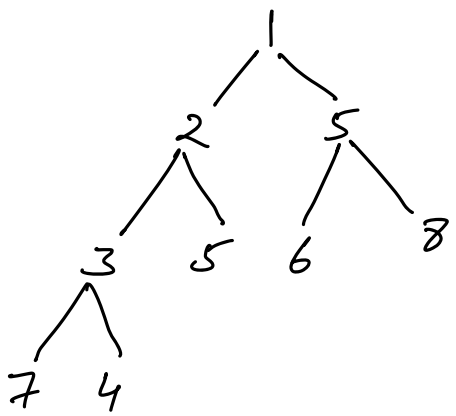
Inplace Build Heap



leaves ⇒ $\frac{9+1}{2} = 5$

T.C ⇒ O(N)

S.C ⇒ O(1)



Q Merge 2 sorted arrays → using 2 pointer.

Q Merge N sorted arrays

1) Add 0^{th} element of every array in a heap.
2) Get min element
3) Insert the next element from same array from which you just now removed smallest element