Heaps 1 : Introduction

Question 1 :- We are given an array that represents the size of different ropes. In a single operation, you can connect two ropes. Cost of connecting two ropes is sum of the length of ropes you are connecting. Find the minimum cost of connecting all the ropes.

rope length

Ex1 :- $A = [\ 2 \quad 5 \quad 3 \quad 2 \quad 6\ ]$

TRY1 :-

$2+5 = 7 \quad [\ 7 \quad 3 \quad 2 \quad 6\ ]$

$7+3 = 10 \quad [\ 10 \quad 2 \quad 6\ ]$

$10+2 = 12 \quad [\ 12 \quad 6\ ]$

$12+6 = 18 \quad [\ 18\ ]$

Total Cost = 47

TRY2 :-

$2+2 = 4 \quad [\ 4 \quad 5 \quad 3 \quad 6\ ]$

$4+3 = 7 \quad [\ 5 \quad 7 \quad 6\ ]$

$5+6 = 11 \quad [\ 7 \quad 11\ ]$

$7+11 = 18 \quad [\ 18\ ]$

Total cost 40

OBSERVATION

↳ we are connecting the ropes with min length.

$$A = [\, x + y + z \,]$$

Proof :- Lets say we are having 3 ropes x, y, z

$$x < y < z$$

Case	I	II	III
Step 1 :-	$(x+y)$	$(y+z)$	$(x+z)$
Step 2 :-	$(x+y)+z$	$(y+z)+x$	$(x+z)+y$
Total Cost	$2x+2y+z$	$2y+2z+x$	$2x+2z+y$

CONCLUSION :- Connecting smaller ropes is beneficial for minimum cost

APPROACH 1 :- Sort the array & start connecting from smaller values. After connecting two ropes & making a new array insert that rope in sorted position.

$A = [\, 2 \;\; 2 \;\; 3 \;\; 5 \;\; 6 \,]$

$A = [\, 4 \;\; 3 \;\; 5 \;\; 6 \,]$

$A = [\, 5 \;\; 6 \;\; 7 \,]$

$A = [\, 7 \;\; 11 \,]$

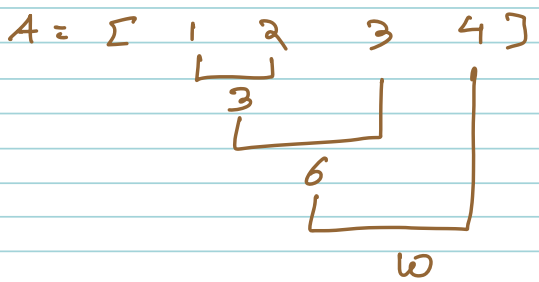
$A = [\, 18 \,]$

TC — O[N~~log~~N + N*N]

≈ O(N²)

SC — O(1)

Quiz :- what is the minimum cost of connecting all the ropes for the array [1, 2, 3, 4]



Total Cost = 3 + 6 + 10 = 19

↳ OPTIMIZATION

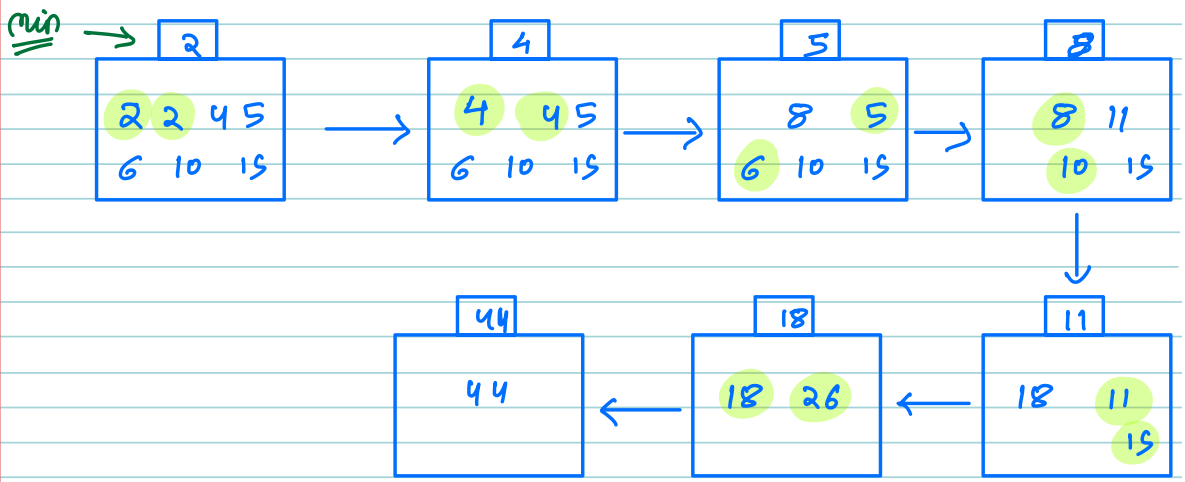
APPROACH 2 :— HEAP

⇒ Benefits of Heap

→ Insert : TC → O(log N)

→ Extract Minimum :- TC → O(log N)

Ex :- [2, 2, 4, 5, 6, 10, 15]



ans = 0 + (2+2) + (4+4) + (5+6) + (8+10) + (11+15) + (18+26)

→ 2 Extract + 1 Insert

TC → [3 log N * (N-1)]

↳ O(N log N)

SC → O(N)

Heap Data Structure

↳ This is a binary Tree with two Specid properties:-

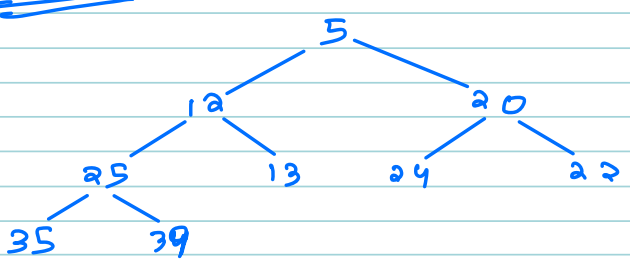
① Complete Binary Tree :- All levels are completely filled. The last level can be exception but it should be filled left to Right.

② Heap order property :-

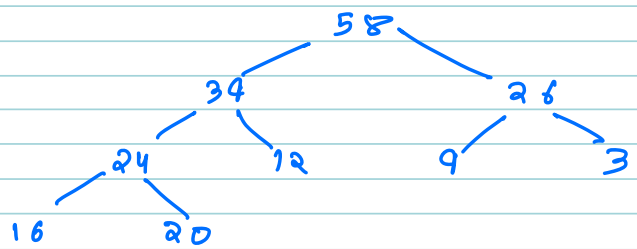
→ Min Heap :- Every Node Should be smaller then all of its descendants

→ Max Heap :- Every Node should be greater then all of its decendants

Ex :- Min Heap

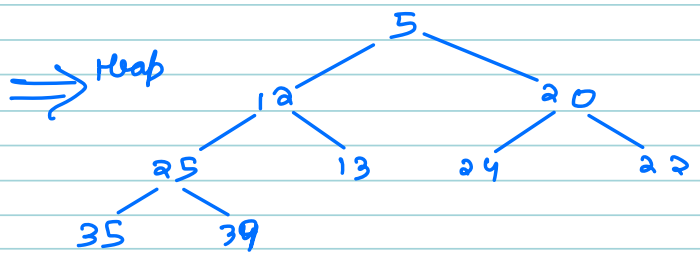


Ex Max Heap



* Since we have been given complete BT, so we can use arrays to implement

ARRAY IMPLEMENTATION OF HEAP



⇒ Relation of children given parent

Parent (i)	children
0	1, 2
1	3, 4
3	7, 8

Relation :- Parent: i → left: $2i+1$, right: $2i+2$

(left, right = children)

⇒ Array Implementation ⇒

0	1	2	3	4	5	6	7	8
5	12	20	25	13	24	22	35	39

⇒ Relation of parent given child

child ⟶ Parent

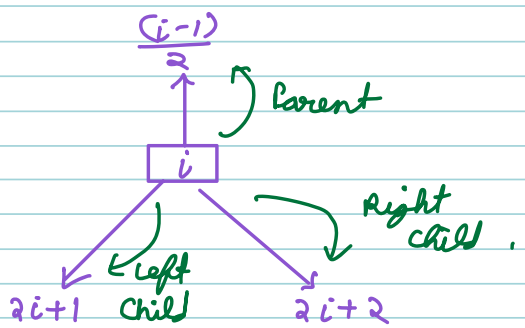
2 ⟶ 0

3 ⟶ 1

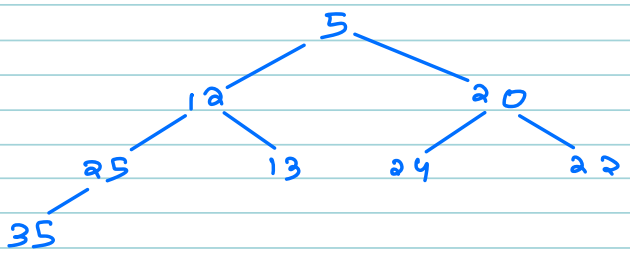
4 ⟶ 1

Relation :- i (child) ⟶ $\frac{(i-1)}{2}$ (Parent)

// Generalize



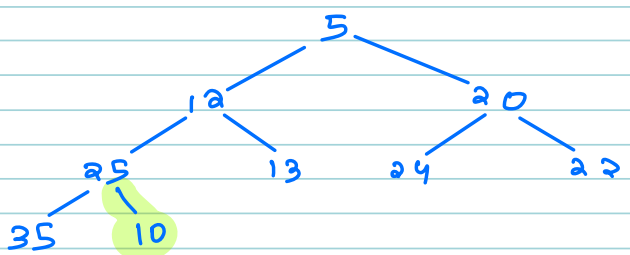
Insertion in Min Heap



0	1	2	3	4	5	6	7
5	12	20	25	13	24	22	35

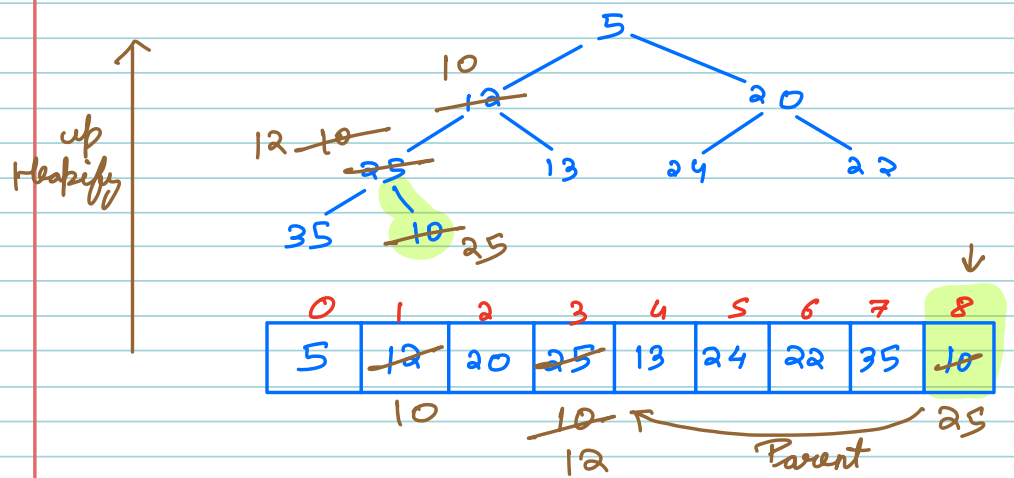
To Insert the element we need to maintain Structure & order

① For Structure :- Insert at the last of array.



0	1	2	3	4	5	6	7	8
5	12	20	25	13	24	22	35	10

② For order



Quiz 2 :- Time complexity of inserting an element in heap having n nodes

TC :- $N \rightarrow \left(\frac{N-1}{2}\right) \rightarrow \left(\frac{\frac{(N-1)-1}{2}}{2}\right) \Rightarrow \frac{N}{4} \rightarrow \frac{N}{8} \rightarrow \cdots 1$

$TC = O(\log N)$

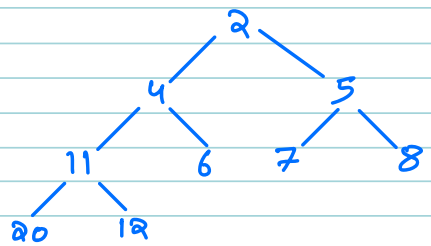
PSEUDO CODE

```
heap [ ];   -> array list

heap.add (value);
i = heap.size() - 1

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

Extract Minimum from Min-Heap.



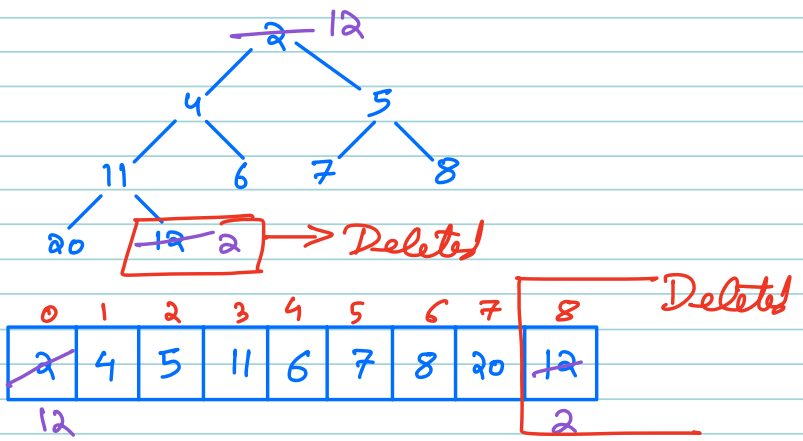
0	1	2	3	4	5	6	7	8
2	4	5	11	6	7	8	20	12

⇒ GOAL = we want to remove 2.

Q: what could be the easiest element to be removed ?

↳ last Element (12)

★ So as a first step lets Swap 2 & 12

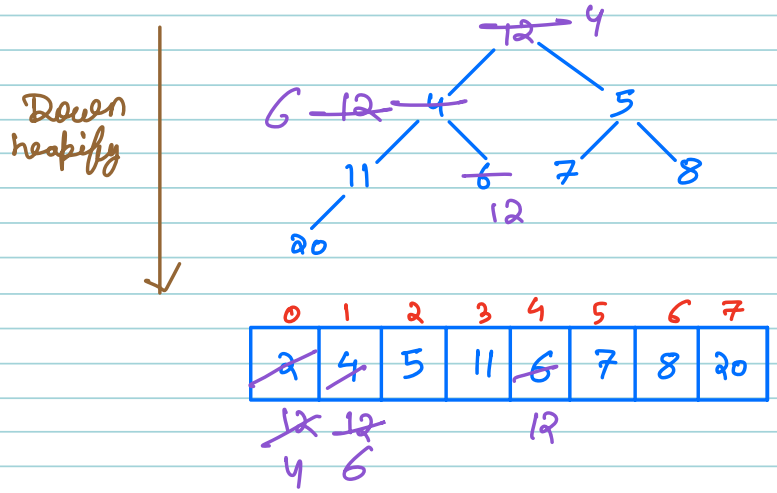


Q: Is it a heap now ?

↳ Not in terms of order.

Q: How we can make the tree to be Min Heap?

↳ Compare a node with its children & swap with smaller child if they are smaller



PSEUDO CODE

```
swap ( heap , 0 , heap.size()-1 );
heap.remove ( heap.size() - 1 ); // Removed Last element.
heapify ( heap , 0 );
```

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

```
        else {
            swap(heap, i, 2i+2);
            i = 2i+2;
        }
    }
}
```

TC : O(log N)

0 → 1 → 2 → 4 → 8 ..... N

BUILD A HEAP

Problem Statement :- we have an array of values, we want to make a heap of it.

[5, 12, -2, 11, 27, 31, 0, 19]

APPROACH1 :- sort the array.

[-2 0 5 11 12 19 27 31]

TC → O(N log N)
SC → O(N)

APPROACH2 :- Start with Empty array & call insert for all the elements

TC — O(N log N) → one Insert
Total elements

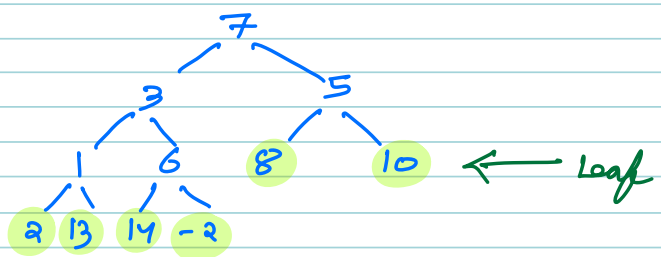
SC — O(N)

(MIN HEAP)

O(N)

APPROACH :- Try to Convert Binary Tree into Heap from Leaf to Root (bottom to Top)

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

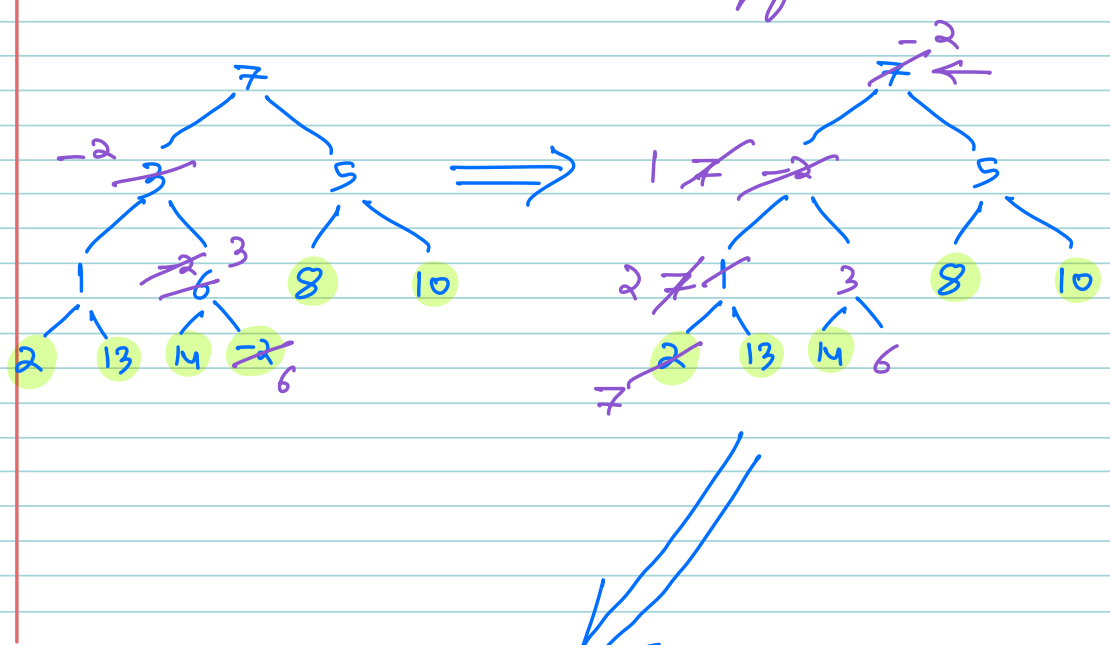


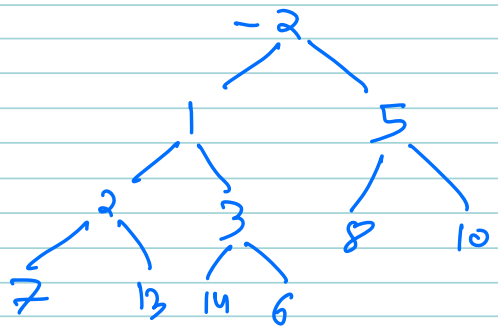
OBSERVATIONS

① Leafs are already in Correct order.

② Exist Non Leaf :- 6 in our Case

↳ If 6 is In order then move ahead otherwise heapify.





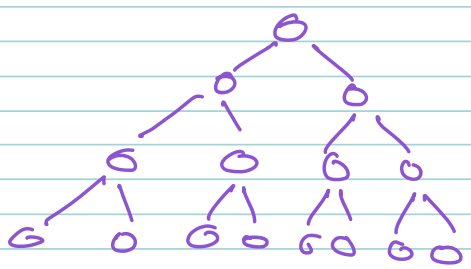
PSEUDO CODE

```
Exist_Non_Leaf = ((N-1) -1)/2 ;

for ( i = Exist_Non_Leaf ; i>=0; i--){
        downheapify ( heap, i);
}
```

Q How this is better ?

Time Complexity



In terms of N	No. of swaps for one node
1	H
⋮	⋮
N/8	2
N/4	1
N/2	0

$$TC :- \left(\frac{N}{2} * 0\right) + \left(\frac{N}{4} * 1\right) + \left(\frac{N}{8} * 2\right) + \ldots !\ldots$$

$$= \frac{N}{2}\left(\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \ldots\ldots\ldots\right)$$

Arithmetic Geometric progression.

Now, $S = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \cdots\cdots$

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

$$S - \frac{S}{2} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots\cdots$$

$$\frac{S}{2} = \frac{a}{1-r} = \frac{1/2}{1-1/2} \quad \text{[Sum of infinite GP} = \frac{a}{1-r}\text{]}$$

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

$$\therefore S = 2$$

Quiz 3 :- what is the time Complexity for building a heap with N nodes

TC :- $\frac{N}{2} * 2$

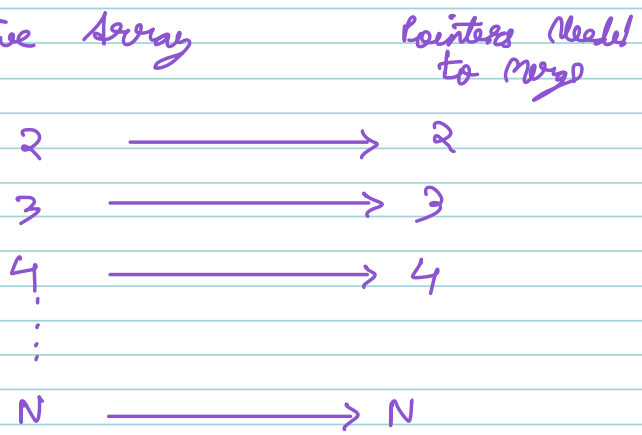
N

$\simeq O(N)$

Question:- Merge N Sorted Arrays

a → [2, 3, 11, 15, 20]
b → [1, 5, 7, 9]
c → [0, 2, 4]
d → [3, 4, 5, 6, 7, 8]
e → [-2, 5, 10, 20]

IDEA:- Use pointers & Merge



★ This is very complex & even nearly impossible.

Quiz 4:- For merging N sorted arrays, which data structure would be most efficient for this task?

↳ Min Heap.

APPROACH :- We can use min-Heap to store this N array pointed by our N pointers.

DRY RUN

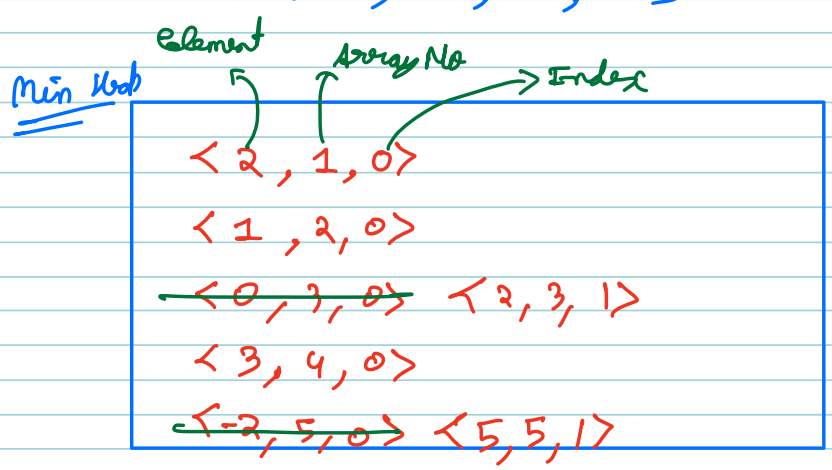
a → [2 , 3 , 11 , 15 , 20]

b → [1 , 5 , 7 , 9]

c → [0 , 2 , 4]

d → [3 , 4 , 5 , 6 , 7 , 8]

e → [-2 , 5 , 10 , 20]



* Find min based on element, increase index in that array & insert that in heap.

TC → O(X log N)

↓

Total element in all the arrays