Heap 1

Madhan Kumar M S
Abhishek Sharma
Akansh Nirmal
amit khandelwal
Balaji S K
Bhaveshkumar
Burhan
Gagan Kumar S
Gowtham
Hemant Kumar
Ishan
Khushi Raj
KULDEEP PATIDAR
Naval Oli
Nikhil Pandey
Pankaj Bhanu
Prajwal Khobragade
Purusharth A
Rajat Sharma
Rajendra
Sanket Giri
Saurabh Ruikar
Shani Jaiswal
sharath r
Shradha Srivastava
Shreya Gupta
Sneha L
Sridhar Hissaria
Subhashini
Subhranil Kundu
Sumit Adwani
Suyash Gupta
Vasanth
Vetrivel H M
Vimal Kumar
Yugesh v

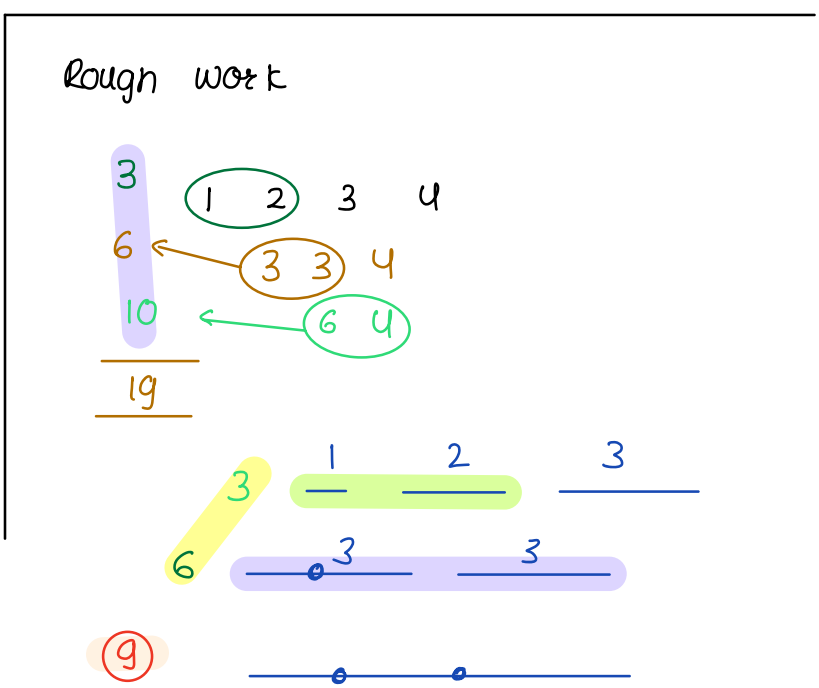
AGENDA:

- Connecting the ropes
- Array Implementation of Trees
- Heap basics
- Merge K sorted arrays.

Current PSP

62 % → 70%.

Rough work

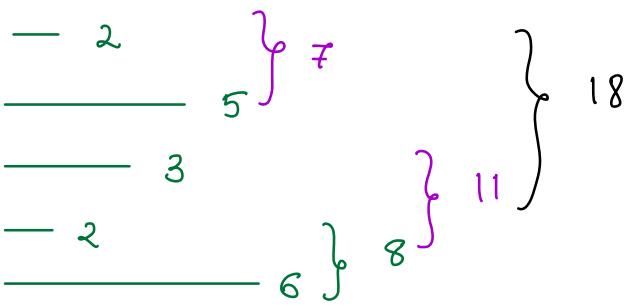


Connecting the ropes

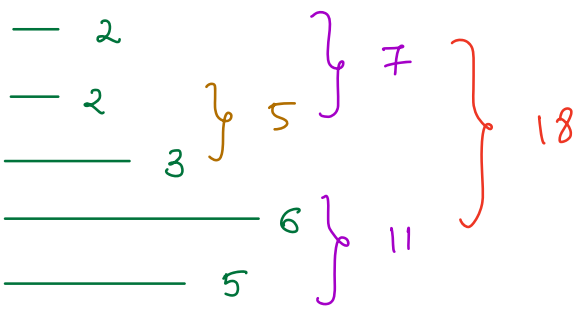
You can connect two ropes together

Cost of connecting two ropes = sum of length of ropes

Find the minimum cost of connecting all the ropes



cost = 7 + 8 + 11 + 18 = 44



cost = 11 + 5 + 7 + 18 = 41

Bruteforce

$x < y < z$

Cost ① < ② < ③

① $x + y$ → $(x+y)+z$

② $x + z$ → $(x+z)+y$

③ $y + z$ → $(y+z)+x$

Idea ⟶ always connect the shortest two ropes

2 5 3 2 6

step 1 ⟶ sort A

4 | 2 2 3 5 6

4 3 5 6

Again sort

7 | 3 4 5 6

7 5 6

11 | 5 6 7

11 7 ⟶ 7 11 → 18

ans = 40

TC: sort N times

$N * N\log N$ $O(N^2 \log N)$

1> Insert an element

2> Extract the minimum

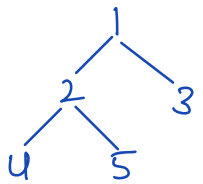
} BBST $O(\log N)$

→ Heap

NOTE: Do not confuse with BST

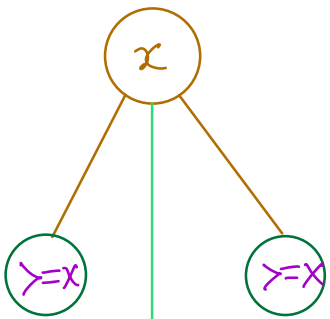
Heap → ① Heap is a complete BT.

All levels are filled except for last level which is filled from L → R



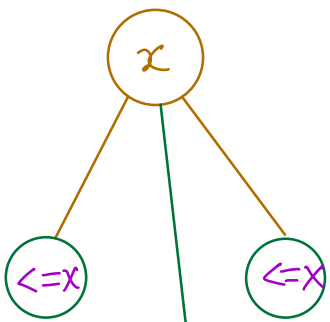
Heap Order property

Min Heap



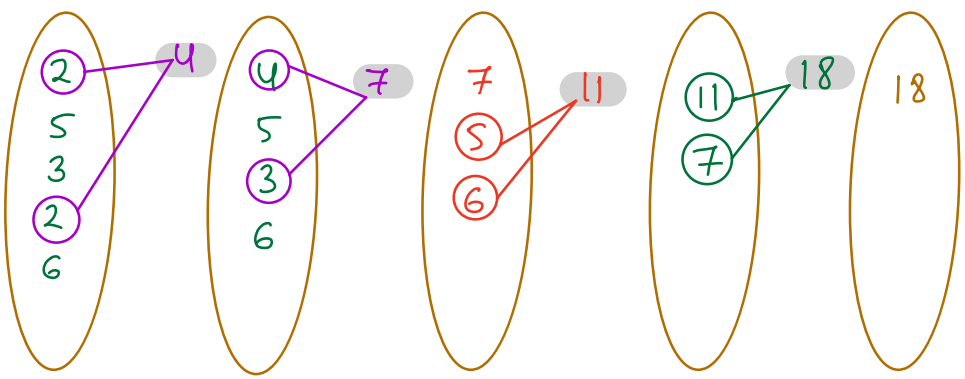
parent is min out of (p, lchild, rchild)

Max Heap



parent is max out of (p, lchild, rchild)

2 5 3 2 6

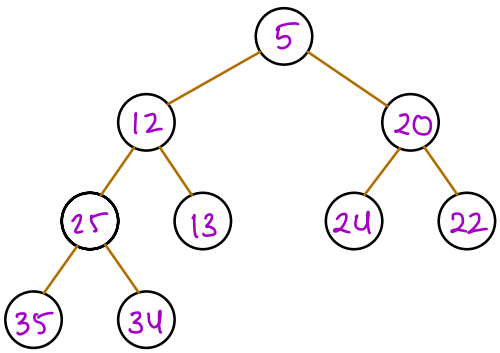


AN = 40

TC: O(NlogN)

SC: O(N)

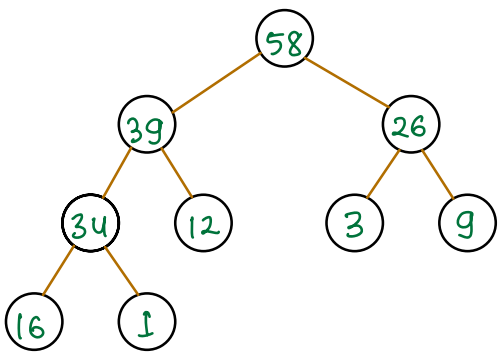
Valid Min Heap



① CBT

② HOP

Valid Max Heap



CBT

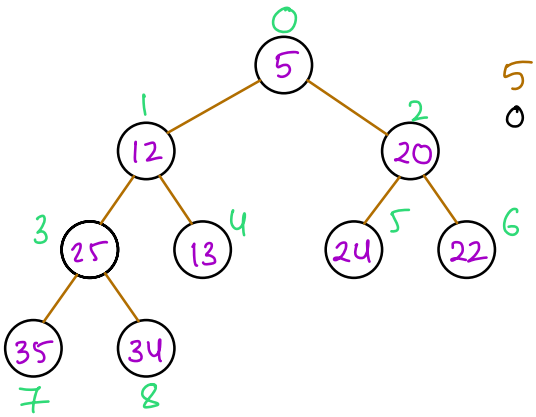
HOP

Java — Priority Queue
Python — heapq
C++ — priority queue

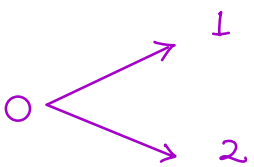
} Read the docs

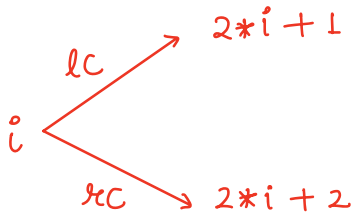
Array Implementation of Trees

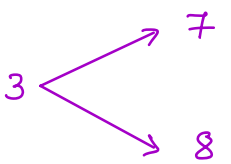
Only valid for CBT.

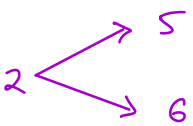


5	12	20	25	13	24	22	35	34
0	1	2	3	4	5	6	7	8



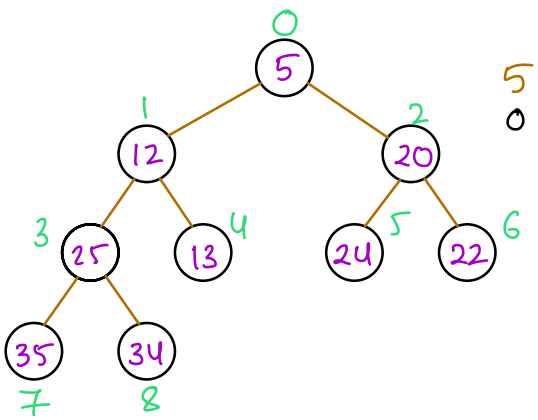






Given i index as root →

- left child = $2i + 1$
- right child = $2i + 2$



5	12	20	25	13	24	22	35	34
0	1	2	3	4	5	6	7	8

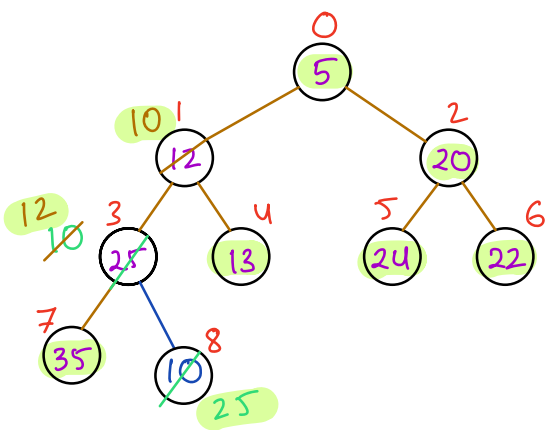
- 4 —parent→ 1
- 7 —parent→ 3
- 5 —parent→ 2
- 8 —parent→ 3

i —parent→ $\frac{(i-1)}{2}$

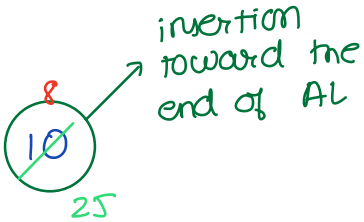
Min Heap Insertion {Heapify Up}

insert 10

TC : O(log N)



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



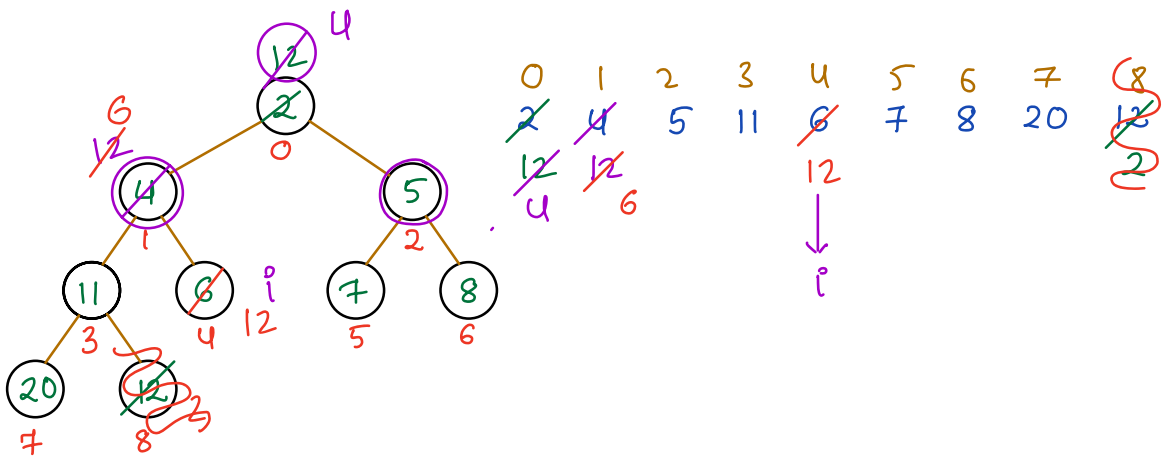
Pseudocode

```
insert ( heap , x ) {                    insert (x)
    heap.add( x )      // x may break HOP
    i = heap.size() -1
    while ( i > 0 ) {
        pi = (i-1)/2
        if ( heap[i] < heap[pi] ) {
            swap ( heap , i , pi )
        }
        else {
            break
        }
        i = pi
    }
}
```

Heapify Up

TC: O(log(N))

Extract Min Element { Heapify Down }



∵ Removing from front in an AL is O(N)
we need to think of a diff way to remove (2)

Remove from end in an AL is O(1)

---

step 1 ⟶ swap (0, n-1)
step 2 ⟶ remove the last index
step 3 ⟶ compare p.val, l.val, r.val

parent left right

TC: O(log N)

if parent is min of (p.val, l.val, r.val)
break

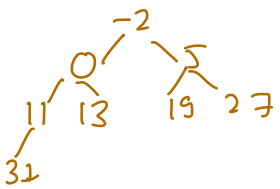
step4 ⟶ move in the direction of min child.

HW.

22:50

Build a heap from an array **** {Heapify}

5	13	-2	11	27	31	0	19
0	1	2	3	4	5	6	7



Bruteforce

sort given A[]

-2 0 5 11 13 19 27 31

TC: O(N logN)

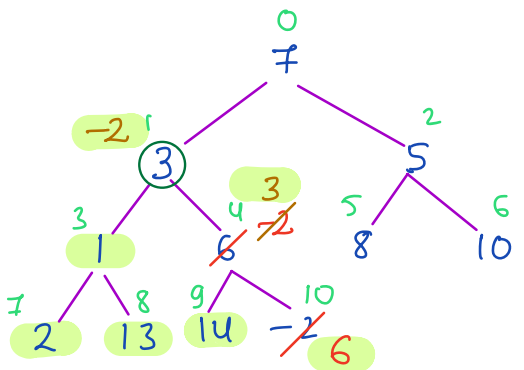
Idea 2

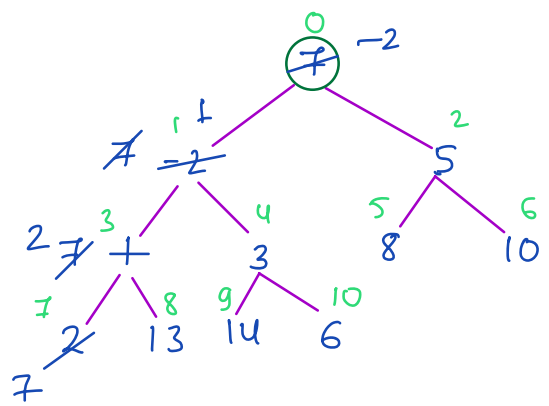
~~∀~~ values in A insert them one by one inside another DS {AL} using above insert fn"

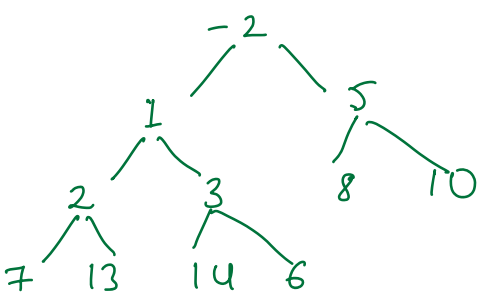
```
heap = []
for (val : A ) {
    |       insert ( heap, val)
    |_ }
```

TC: O(N logN)

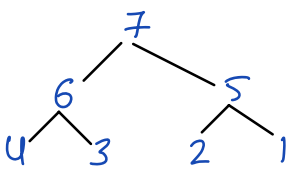
7	3	5	1	6	8	10	2	13	14	-2
0	1	2	3	4	5	6	7	8	9	10



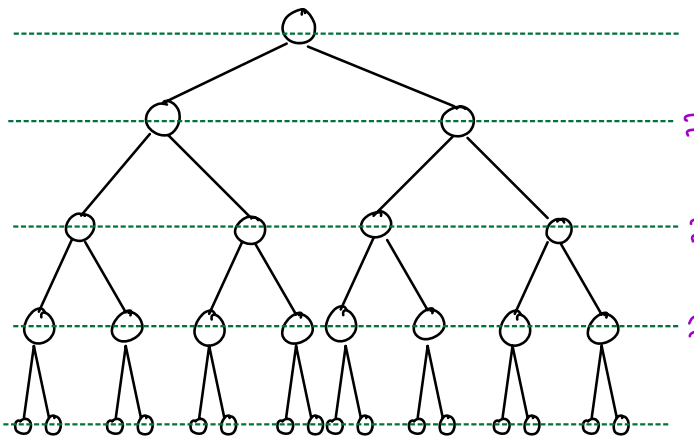




worst case




```
for (i ⟶ N-1 to 0) {
    heapifyDown(A, i)
}
```



swaps per node

$N/16$	3
$N/8$	2
$N/4$	1
$\frac{N}{2}$	0

TC : $\frac{N}{2} * 0 + \frac{N}{4} * 1 + \frac{N}{8} * 2 \cdots + \frac{N}{16} * 3$

$\Rightarrow \frac{N}{2} \left\{ 0 + \underbrace{\frac{1}{2} + \frac{2}{4} + \frac{3}{8}}_{\text{AGP}} \cdots \right\}$

$\frac{N}{2} * 2 \Rightarrow$ Refer doubt session

TC : O(N) ***

$\Rightarrow$ NOTE : Heapify or building a heap from an array takes O(N)

Microsoft

Merge N sorted arrays.

```
M[][] = [ [0   2   9   20]
          [24  31  110]
          [1   5   7   10   11]
          [3   20  21  22   23]
        ]
```

N

Final output should be a single 1D array with all the above values as sorted.

Bruteforce

A[]

→ Create an array with all values from M[][]

→ Sort A

Total elements are X

TC: O(X log X)

Idea 2

→ Put all elements in heap A[] O(X)

→ Heapify all elements O(X)

→ Extract them one by one → log(X) per removal

TC: O(X log X)

M[][] = [[0 2 9 20]

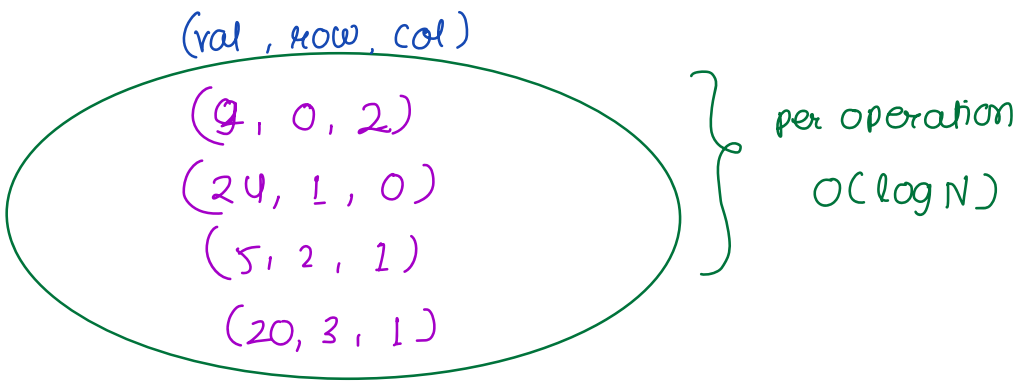
[24 31 110]

[1 5 7 10 11]

[3 20 21 22 23]

]

Idea put all column values in min heap



any

0 1 2 3 .....

Given total elements is X : TC: O(X log N)

Pseudocode

```
int[]  mergeKArrays ( M [][] ) {
    ans = []
   heap = []   // heap init
   R =  // no of rows                                  pair class
   for  r ⟶ 0 to R-1
          heap.insert ( ( M[r][0] , r , 0 ) )

    while ( heap is not empty ) {
        val , r , c   =  extractMin ( heap )
         ans.add ( val )
         if ( c+1 < M[r].length ) {
              heap.insert ( M[r][c+1] , r , c+1 )
         }
    }
     return  ans
}
```

$X = R * \{\max \text{ col}\}$

TC: $O(X \log N)$

Doubt session