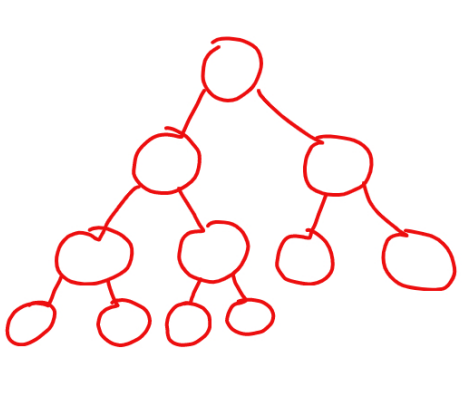
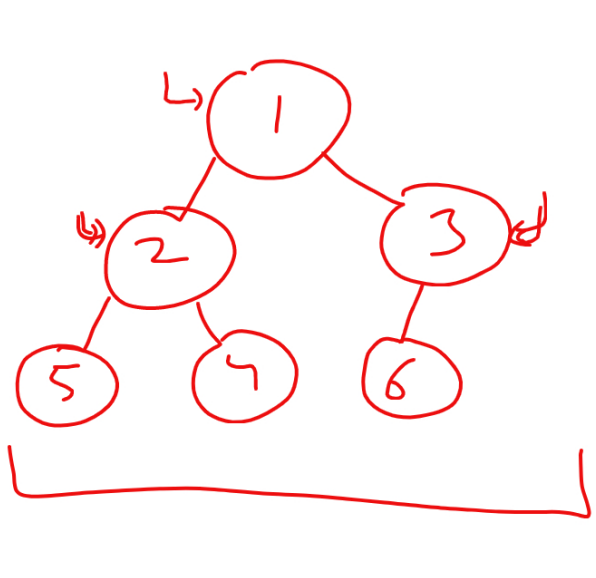
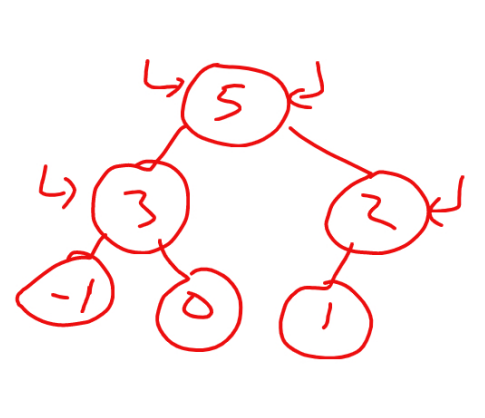


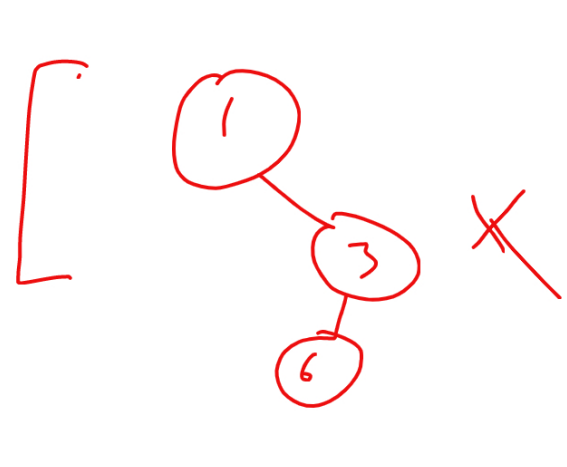
(2, 1, 3, 5, 7)
get_min: 1
pop-min: (2, 3, 5, 7)
get_min: 2

min-heap:



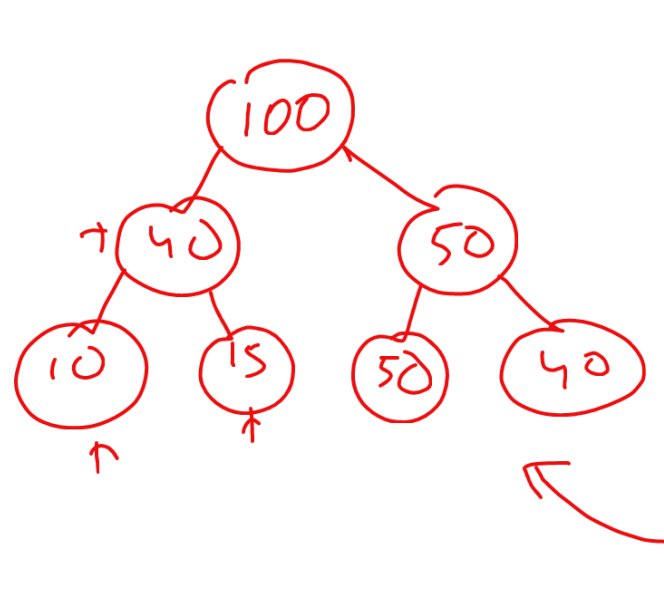
max_heap:





BINARY HEAPS USING ARRAYS

→ We are able to represent heaps using arrays because they are complete binary trees.


100
40
50
10
15
50
40

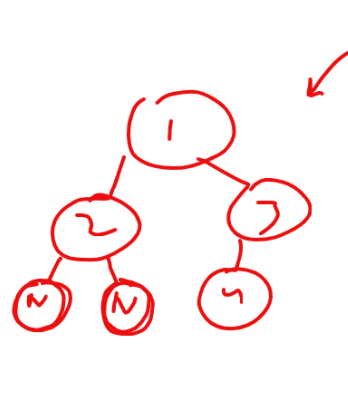
0 1 2 3 4 5 6
⇒ arr[100, 40, 50, 10, 15, 50, 40]

$2*1+1 = 3$
$2*1+2 = 4$

$(3-1)/2 = 1$

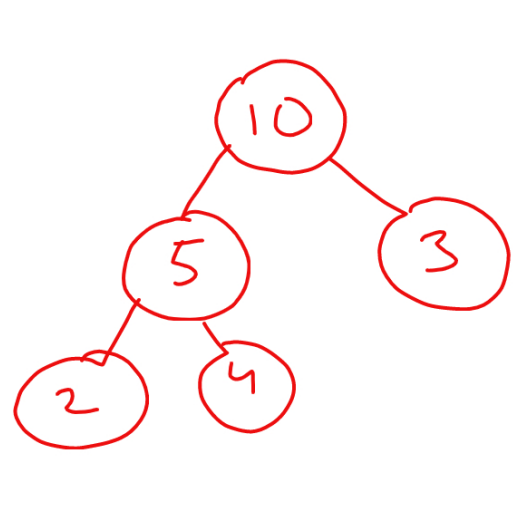
→ Root is at arr[0].
→ For the node at index i:
- → Parent is at index: $(i-1)/2$
- → Left child " " : $2*i+1$
- → Right child " " : $2*i+2$

"null"



[1, 2, 3, N, N, 4]

TOP-DOWN HEAPIFY: Used for DELETION

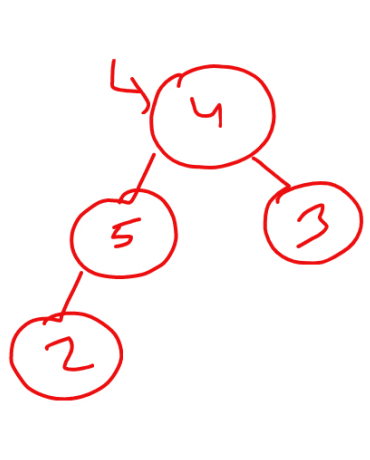


Deletion in a max-heap
⇓
Deleting the maximum value node.

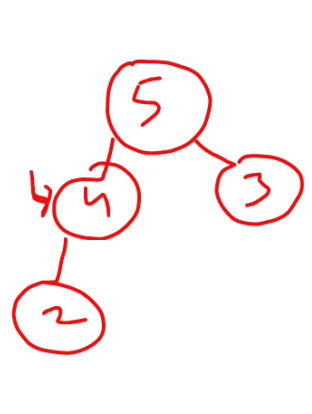
Steps:

1) Replace the root node with the last node and delete the last node.
2) Perform top-down heapify on root.

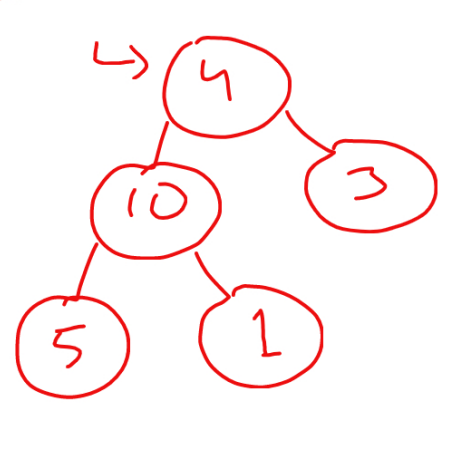
Step-1:



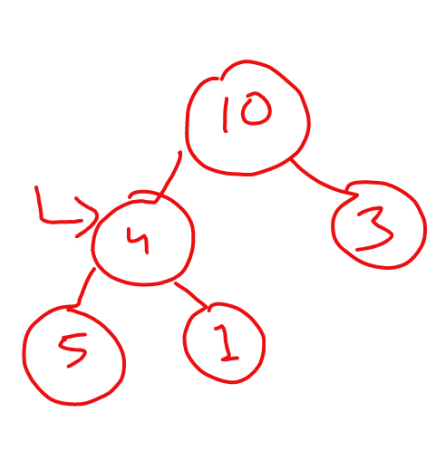
Step-2:



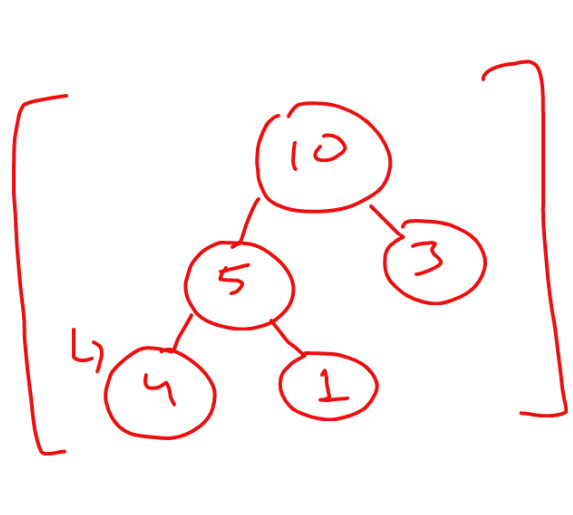
TOP-DOWN HEAPIFY:



↓
0 1 2 3 4
[4, 10, 3, 5, 1]
left child: $2*0+1 = 1$
right child: $2*0+2 = 2$

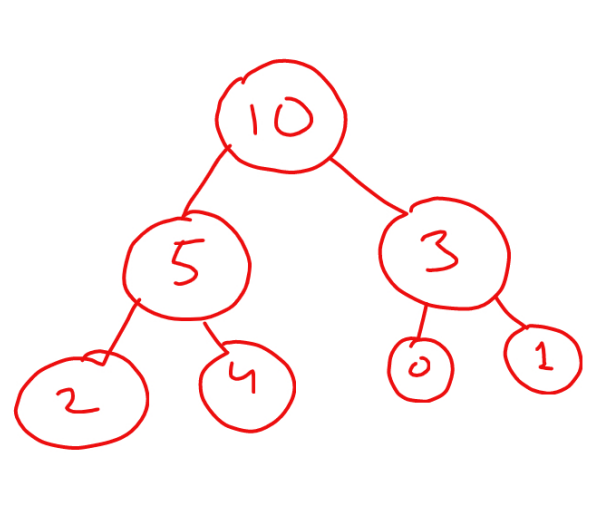


↓
0 1 2 3 4
[10, 4, 3, 5, 1]
left child: $2*1+1 = 3$
right child: $2*1+2 = 4$



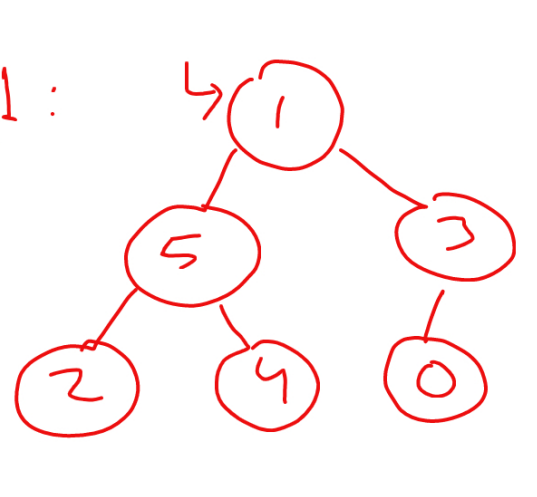
↓
0 1 2 3 4
[10, 5, 3, 4, 1]
left child $= 2*3+1 = 7$ ×
right child $= 2*3+2 = 8$ ×

Deletion Example:



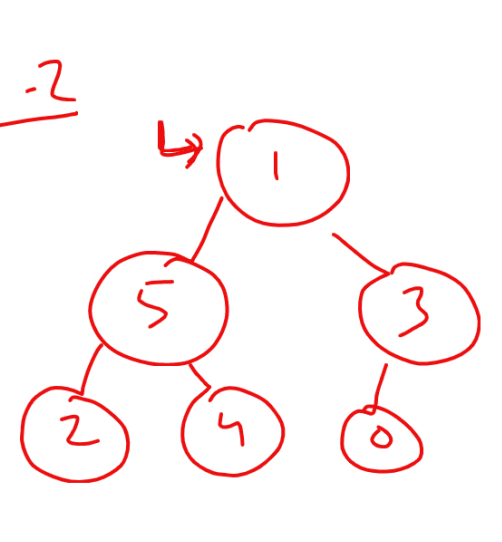
[10, 5, 3, 2, 4, 0, 1]

Step-1:

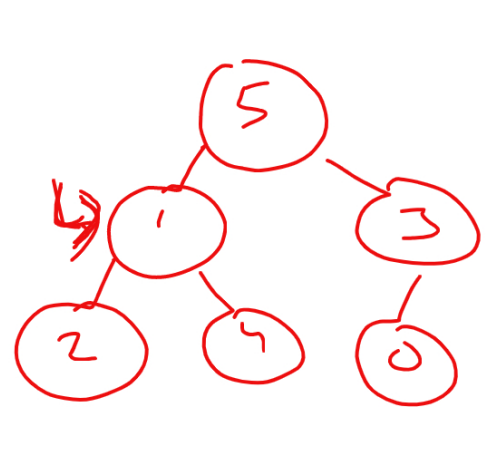


[1, 5, 3, 2, 4, 0]

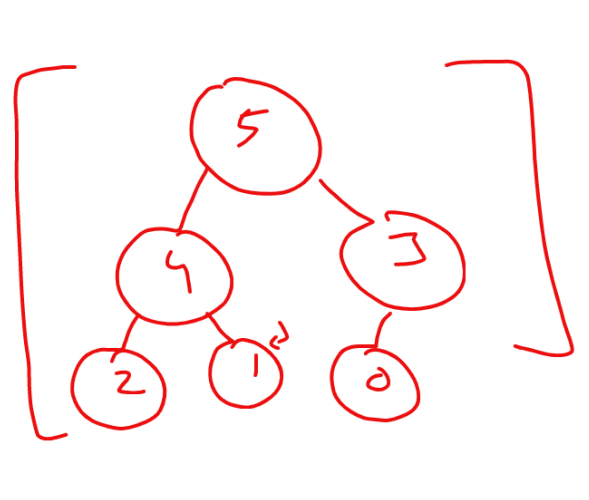
Step-2



↓
0 1 2 3 4 5
[1, 5, 3, 2, 4, 0]
index = 0
lvi = ~~0~~ ①
$lc = 2*0+1 = 1$
$rc = 2*0+2 = 2$



0 1 2 3 4 5
↓
[5, 1, 3, 2, 4, 0]
$lc = 2*1+1 = 3$
$rc = 2*1+2 = 4$

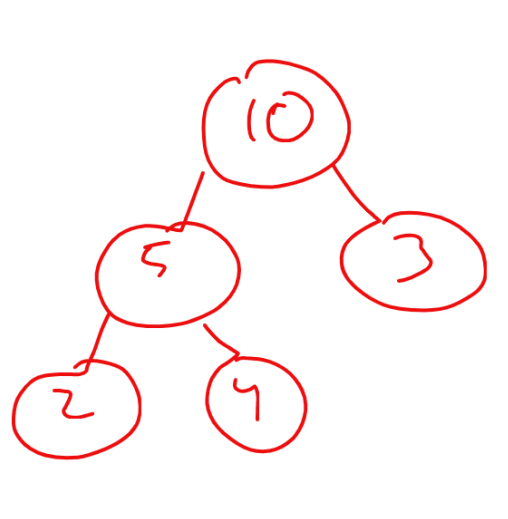


0 1 2 3 4 5
↓
[5, 4, 3, 2, 1, 0]
$lc = 2*4+1 = 9$ ×
$rc = 2*4+2 = 10$ ×

TC: O(height)
= O(log(n))
Aux. space = O(log(n))

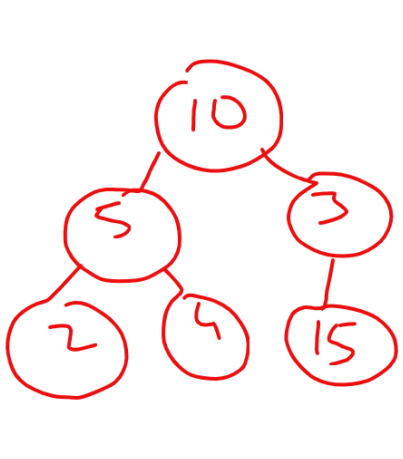
BOTTOM-UP HEAPIFY: USED FOR INSERTION

1) Inc. the heap size by 1
2) Insert the new element at the end
3) Bottom-up heapify on the last index.



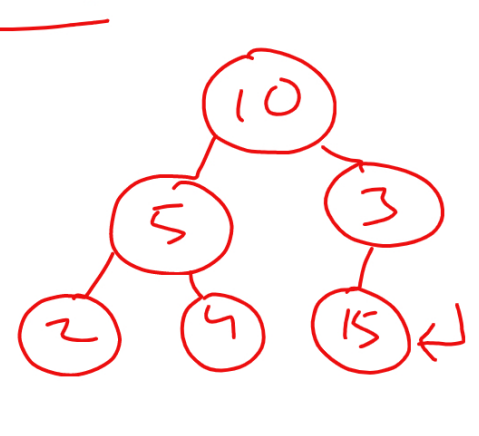
(Insert-15)
[10, 5, 3, 2, 4]

Step-1 & 2:

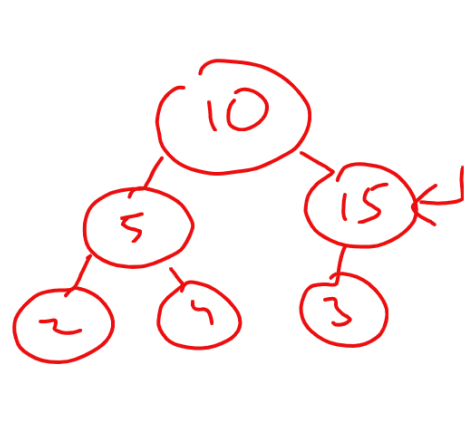


0 1 2 3 4 5
[10, 5, 3, 2, 4, 15]

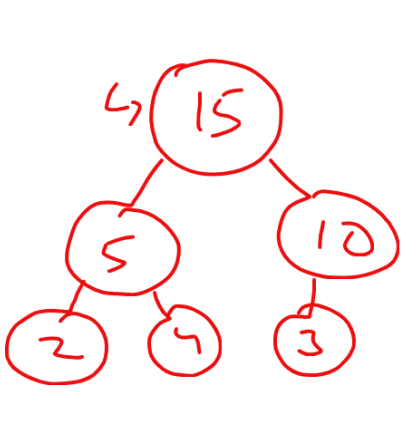
Step-3:



↓
0 1 2 3 4 5
[10, 5, 3, 2, 4, 15]
parent $= \frac{5-1}{2} = \frac{4}{2} = 2$

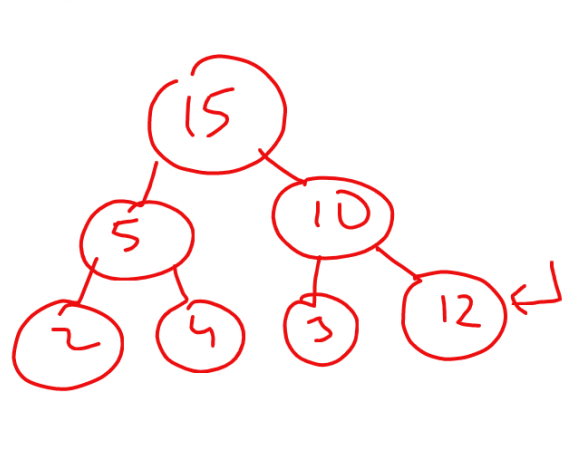


↓
0 1 2 3 4 5
[10, 5, 15, 2, 4, 3]
parent $= \frac{2-1}{2} = \frac{1}{2} = 0$

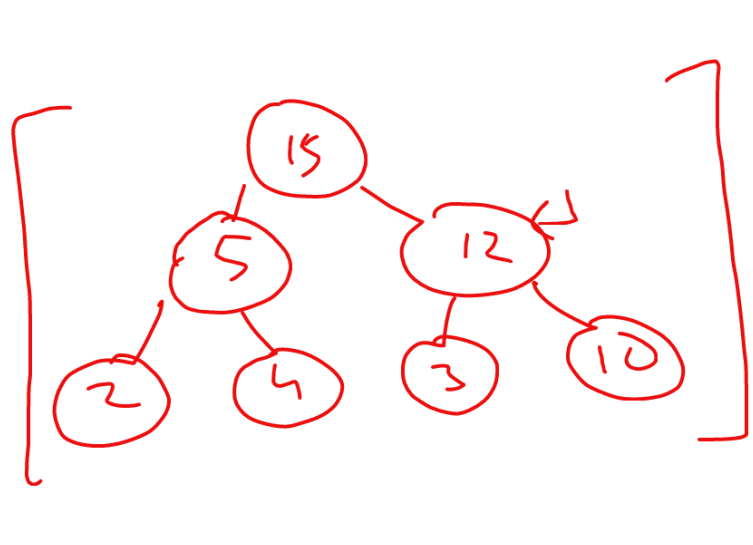


0 1 2 3 4 5
↓
[15, 5, 10, 2, 4, 3]
parent: $\frac{0-1}{2} = \frac{-1}{2}$

BOTTOM-UP HEAPIFY:

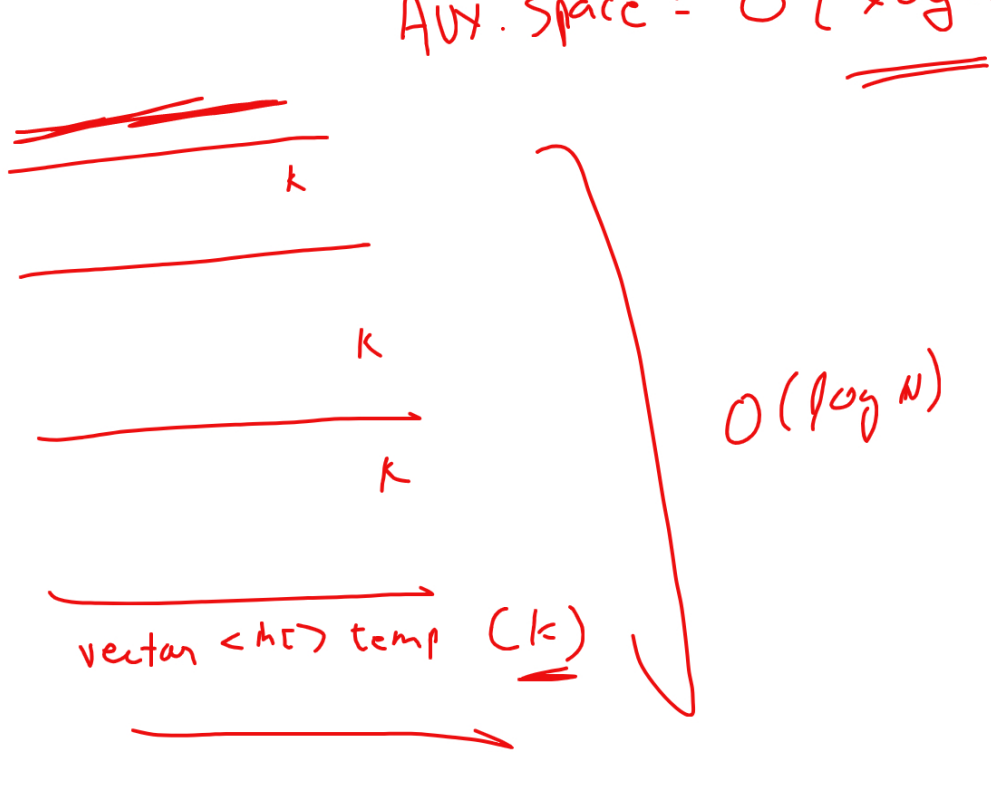

15
5
10
2
4
3
12

↓
0 1 2 3 4 5 6
[15, 5, 10, 2, 4, 3, 12]
parent $= \frac{6-1}{2} = \frac{5}{2} = 2$

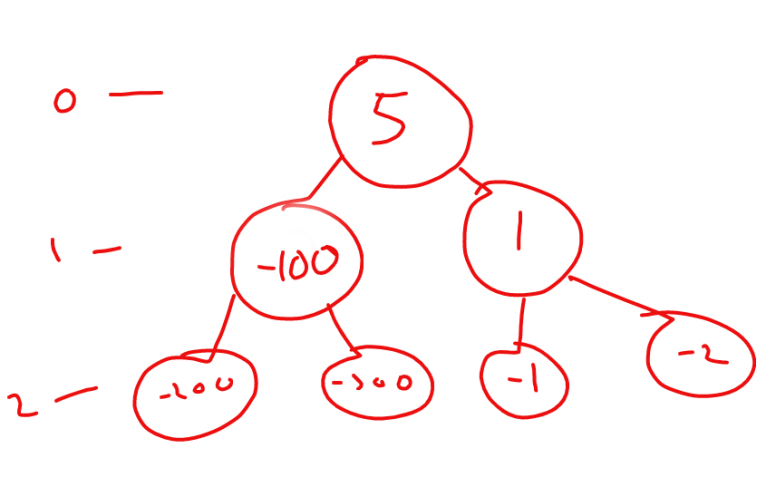

15
5
12
2
4
3
10

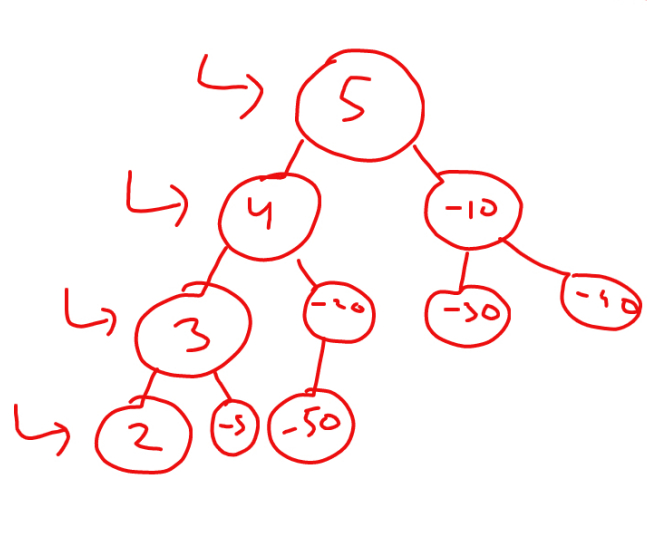
↓
0 1 2 3 4 5 6
[15, 5, 12, 2, 4, 3, 10]
parent: $\frac{2-1}{2} = \frac{1}{2} = 0$

TC: O(ht) = O(log N)
Aux. space = O(log N)


k
k
k
vector<int> temp (k)
O(log N)

Aux. space = O(k*log N)


0
1
2
5
-100
1
-200
-300
-1
-2


5
4
-10
3
-20
-30
-40
2
-3
-50

---

arr: [3, 5, 4, 2, 9]   K = 3   5, ④, 9

K-th largest element is the smallest of the K-largest elements in the array.

(3, 5, 4)

[3, 5, 4, 2] k=3
$(\overset{2}{\cancel{3}}, 5, 4)$

⇒ Being at any index i, we need to make sure that the heap contains the K-largest elements till index i

arr = [3, 5, 4, 2, 9], k = 3

→ $(\cancel{3}, 5, 4)$
→ (9, 5, 4) ⇒ get_min: ④

↓
[-3, -5, -4, -2, -9]   k = 3
↓
(-3, -5, -4)   arr[i] = 2

Merge K sorted arrays.   (1, 0, 4, 1)

0 [1, 3, 5, 7]
1 [0, 2, 6, 10]
2 [4, 5, 6, ⑦]
3 [1, 2, 3, 4]

⇒ [0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 10]

{arr[i][j], {i, j}}

[0,

[1, ~~(0,0)~~] [3, (0,1)]
[0, ~~(1,0)~~] [2, ~~(1,1)~~] [6, (1,2)]
[4, (2,0)]
[1, ~~(3,0)~~] [2, (3,1)]

[0, 1, 1, 2 . . .]