Priority Queue / Heap

FIFO → first in first out

10 | 20 | 30

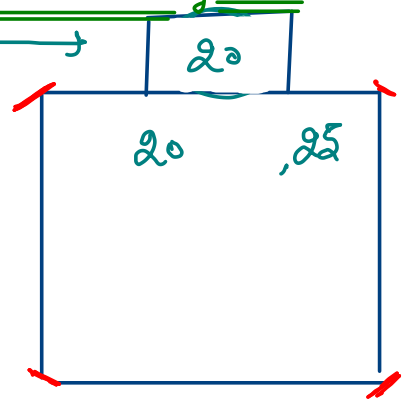
Max priority

- ☆ min → priority
- ☆ max → priority

Default behaviour of PQ

priority Element (min)

add - 10
add - 20
add - 9
add 25
add 7
peek() → 7



remove() → 7 removal form priority Element

peek() = 9

remove() → 9

remove() → 10

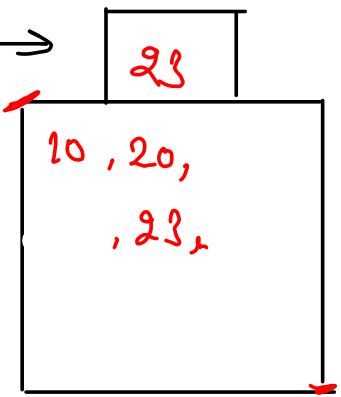
priority Queue → Null ✗

class - comparable ?

comparable

most priority Element (max)

add 10
add 20
add 30
add 25
peek() → 30
remove() → 30
add 23
add 27
remove() → 27
remove() → 20
peek() → 23



this
a.compareTo(b);
other

K Largest Element

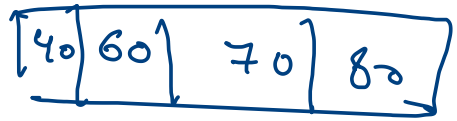
arr → _____

K Largest Elements

allowed space complexity - O(K)

allowed time complexity - O(n log(K))

10 70 9 4 30 40 60 80



Complexity of priority Queue.

add ⟶ O(log n)

remove ⟶ O(log n)

peek ⟶ O(1)

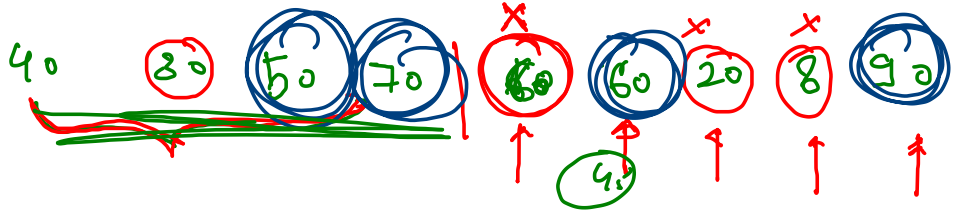
size ⟶ O(1)

where 'n' is no. of element present in priority Queue.

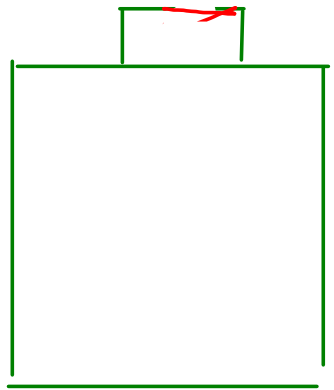
K- largest

k = 4

arr →



min



k-size

50
60
70
90

priority Queue → k

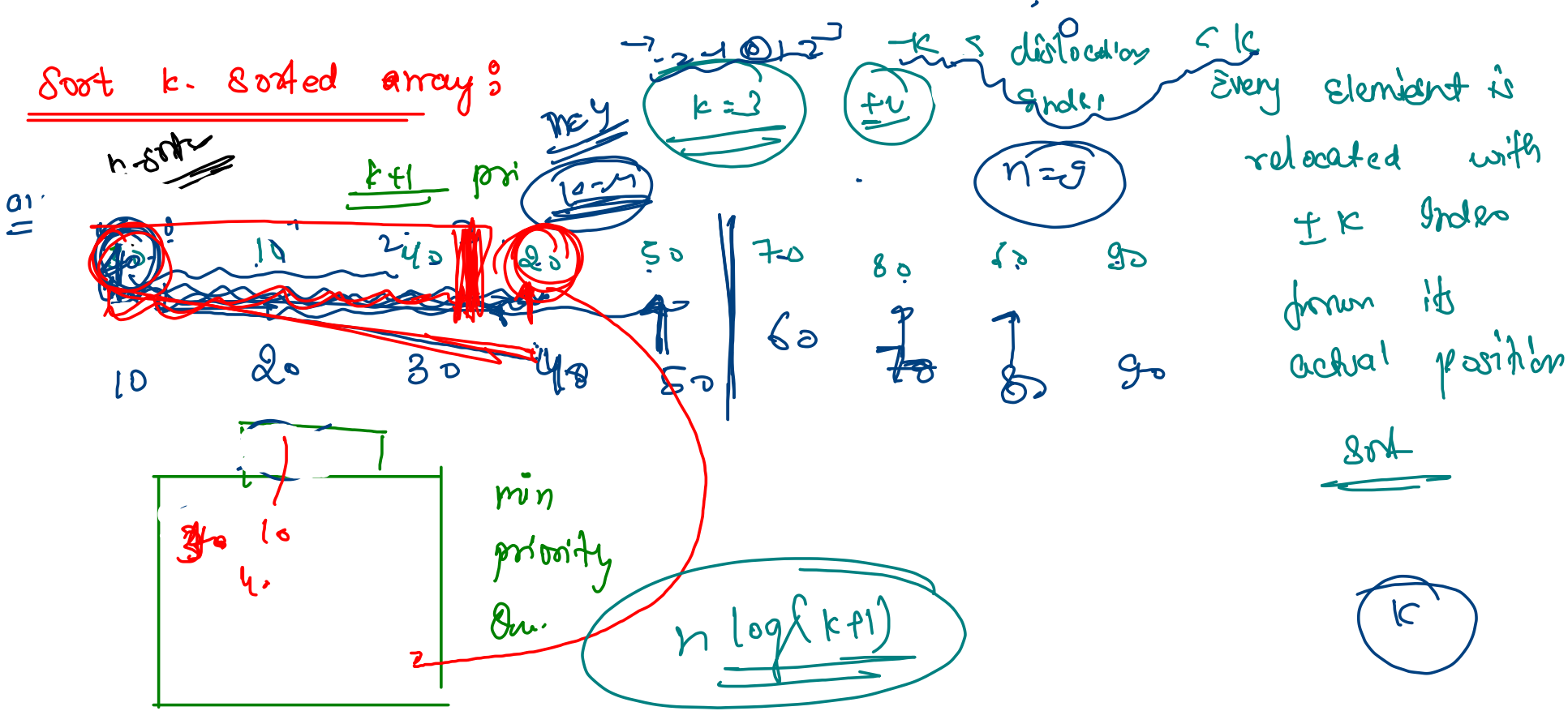
addition in priority Queue = logk

n time.

time complexity = nlogk.

space → O(k)

Sort k-sorted array :

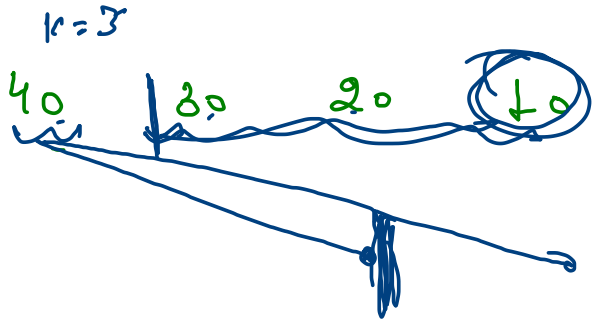


Array Sort → complexity = nlogn.

Array Sort → allowed complexity - nlogk

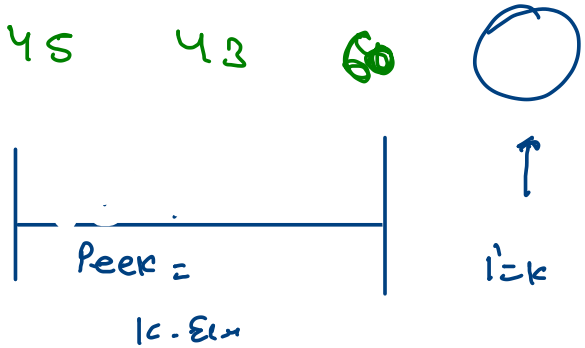
① fill pq. with k+1 Elements

k = 3



min

50 45 43 60

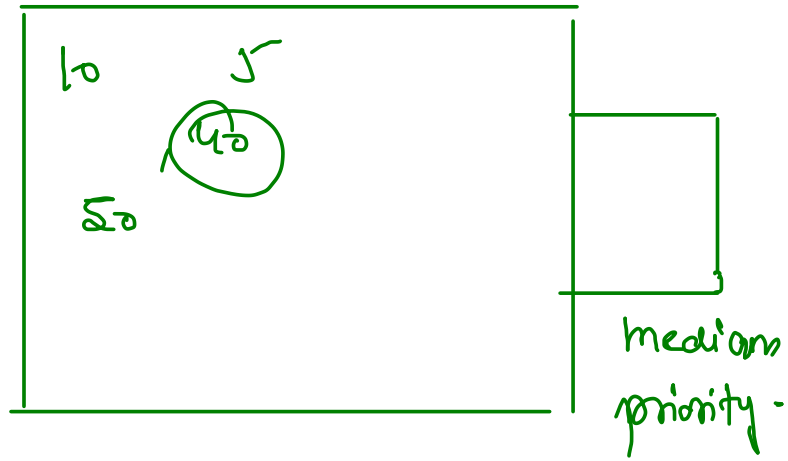


① add k Element in PQ.

② add i^{th} Element in pq & remove peek elmt and remove.

10 20 30 40 43 45 50 60

Median Priority Queue



pq.add. 10

" 20

" add. 5 — 30, 40 — peek

pq.peek() ⟵ median ⟵ 20

Even → End of first half (Sorted consideration)

pq.add (50)

pq. peek() → 30

pq. remove() → 30

pq. remove() → 20

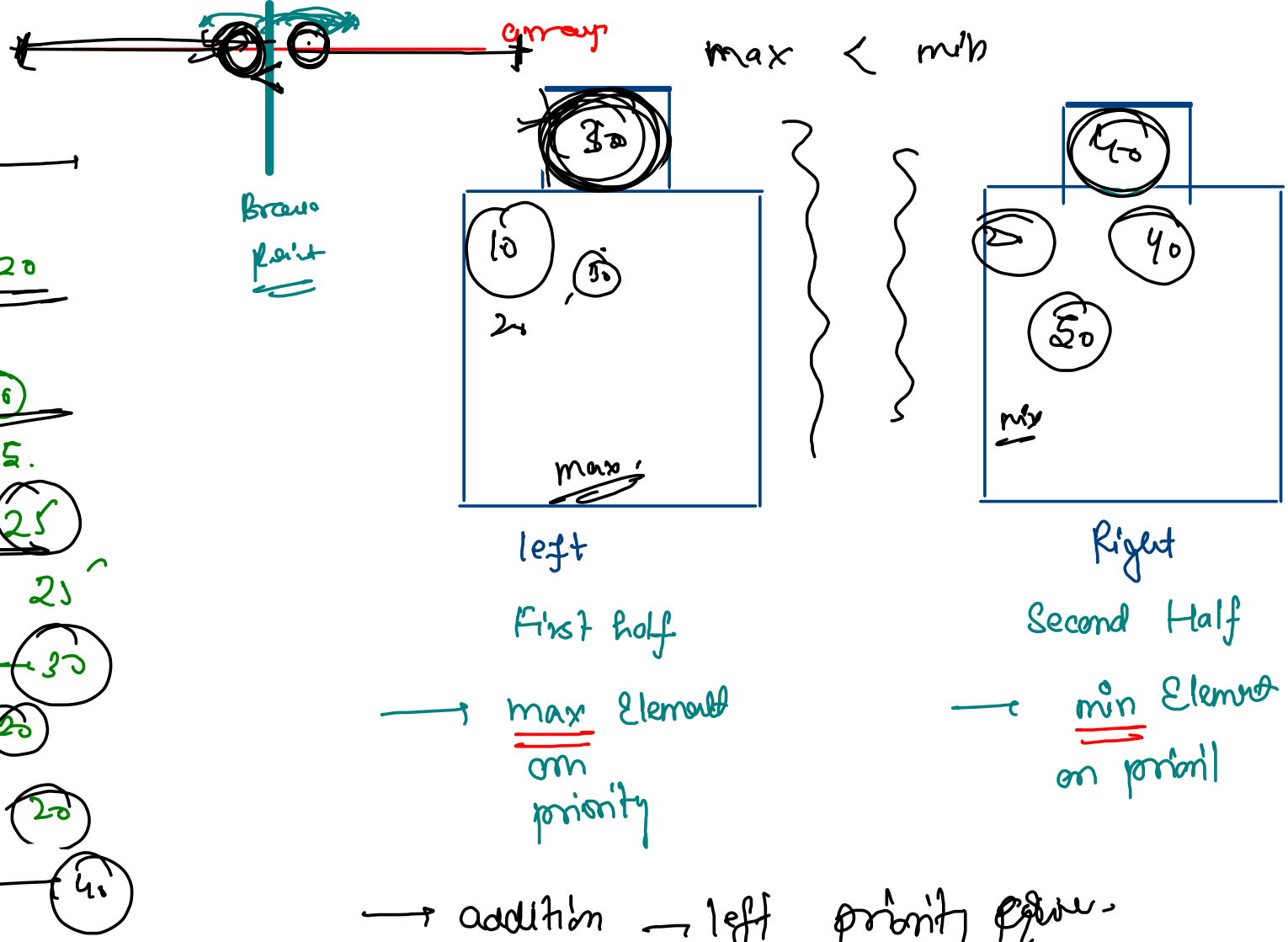
pq. peek() → 40

add → O(log n)

remove → O(log n)

peek → O(1)

size = O(1)


add 10
add 20
add 30
add 40
peek → 20
add - 50
peek → 30
add → 25.
peek → 25
remove - 25
remove → 30
peek → 20
remove → 20
peek → 40
Brain Point
array
max < min
30
10
50
20
max
40
20
50
min
left
First half
→ max Element on priority
Right
Second Half
→ min Element on priority
→ addition → left priority Queue.
→ size difference maintain.

push

```
if( right.size() > 0 && right.peek() < val) {
    right.add(val);
} else {
    left.add(val);
}
```

pop.

case two

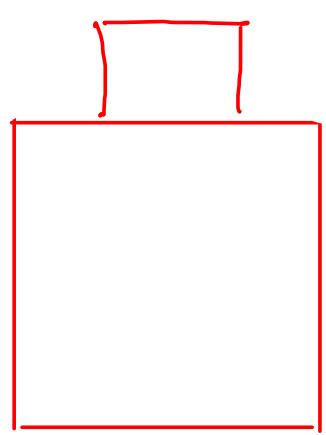
order

peek

```
if(this.size() >= 2){
    and also:
if( right.size() > left.size() ){
    right.peek();
} else {
    left.peek();
}
```

size.

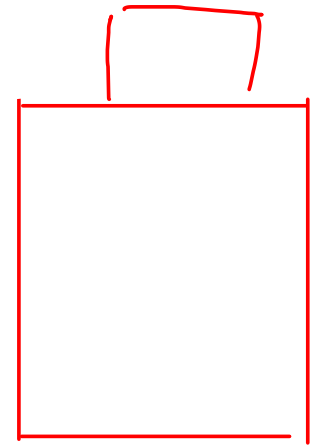
```
left.size() + right.size()
```



5 left

5

4



5.

4

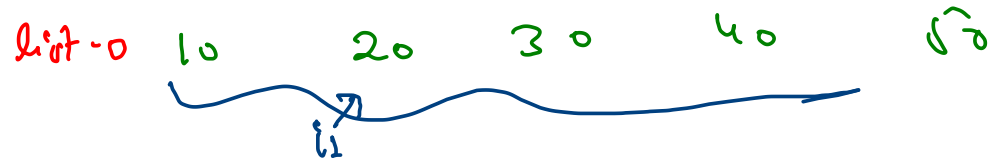
1

right

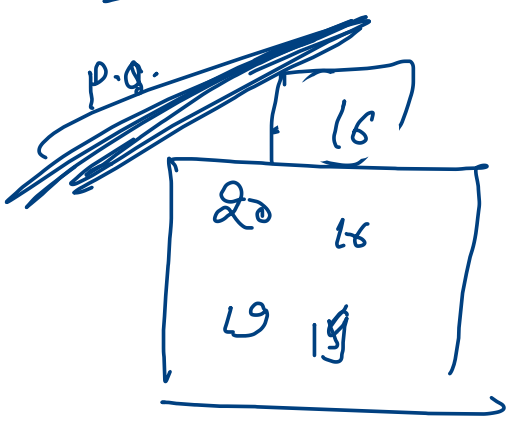
Merge k sorted list (Homecode)

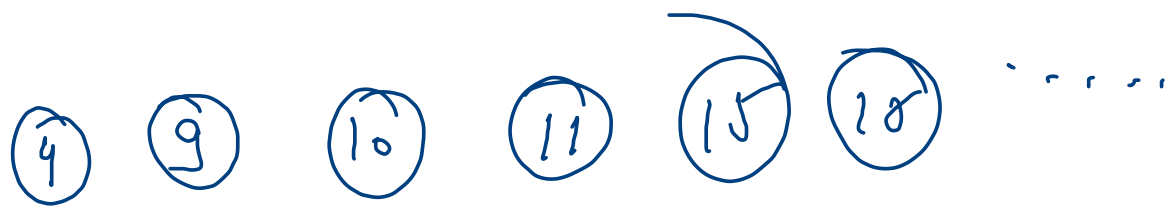
<Pair>

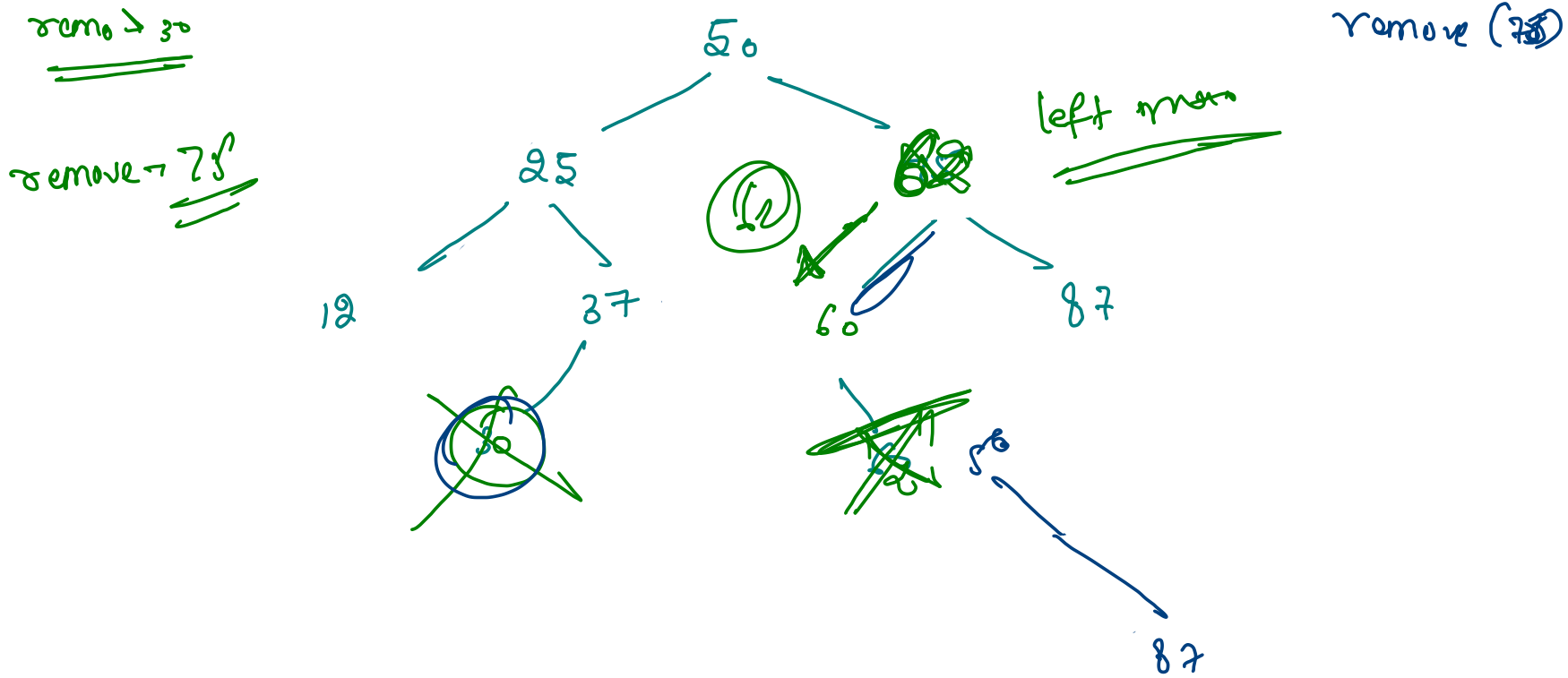
Comparable*



min





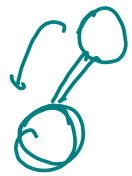

remove → 30
remove → 75
remove (75)
50
25
left max
12
37
60
87
30
87

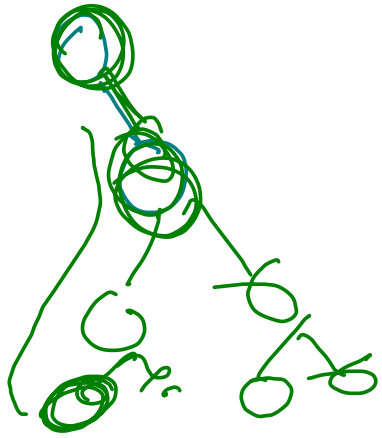
data found

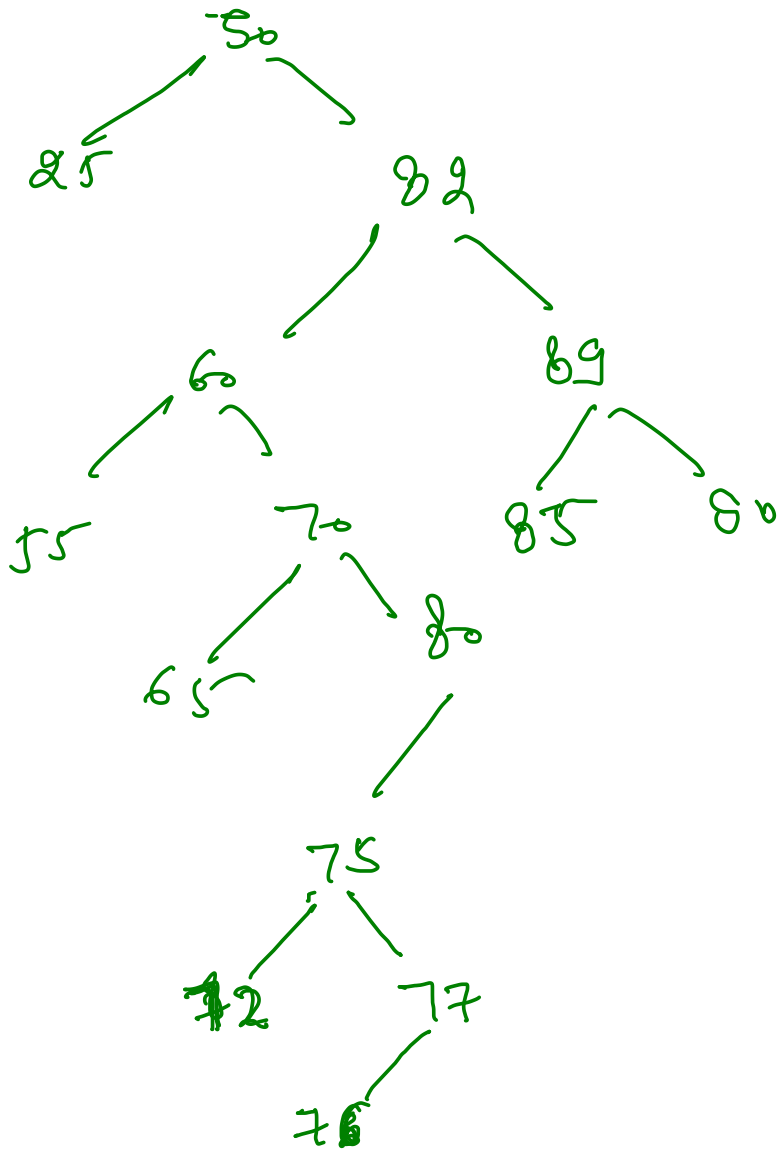
```
if( node.left == null && node.right == null ) {
    return null;
```

leaf node.

```
} else if( node.left != null ) {
    lmax = max(node.left)
    node.data = lmax;
node.left = remove ( node.left,lmax);
} else {
    return node.right;
}
```







Remove - 82

target

DRY RUN