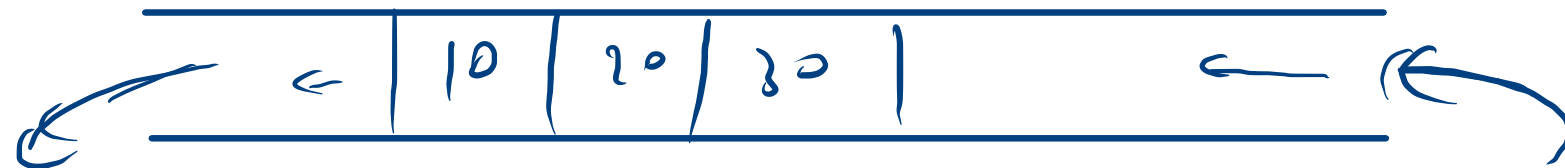


Priority Queue



add
remove

1

higher priority

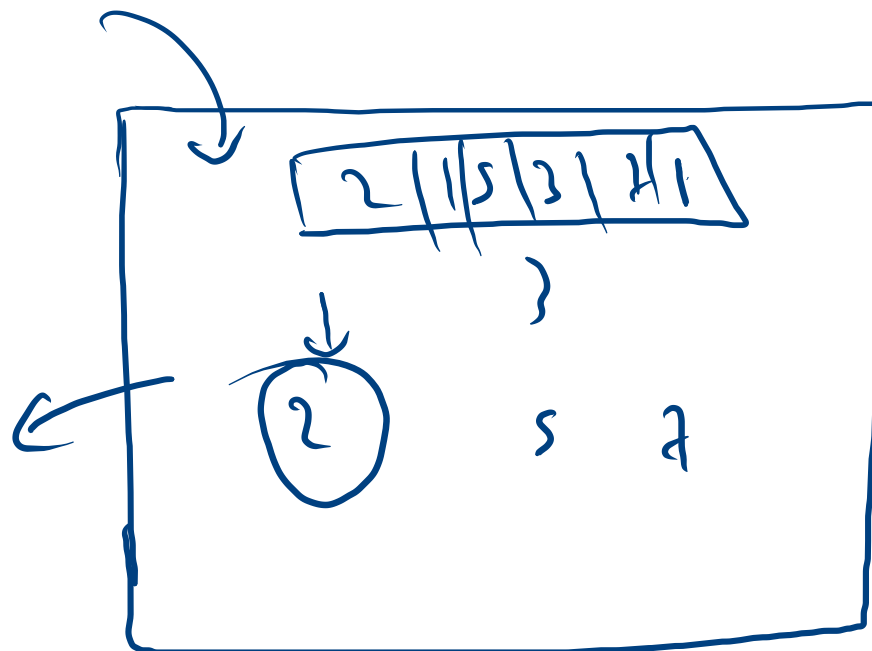
peek

1

Stream

max

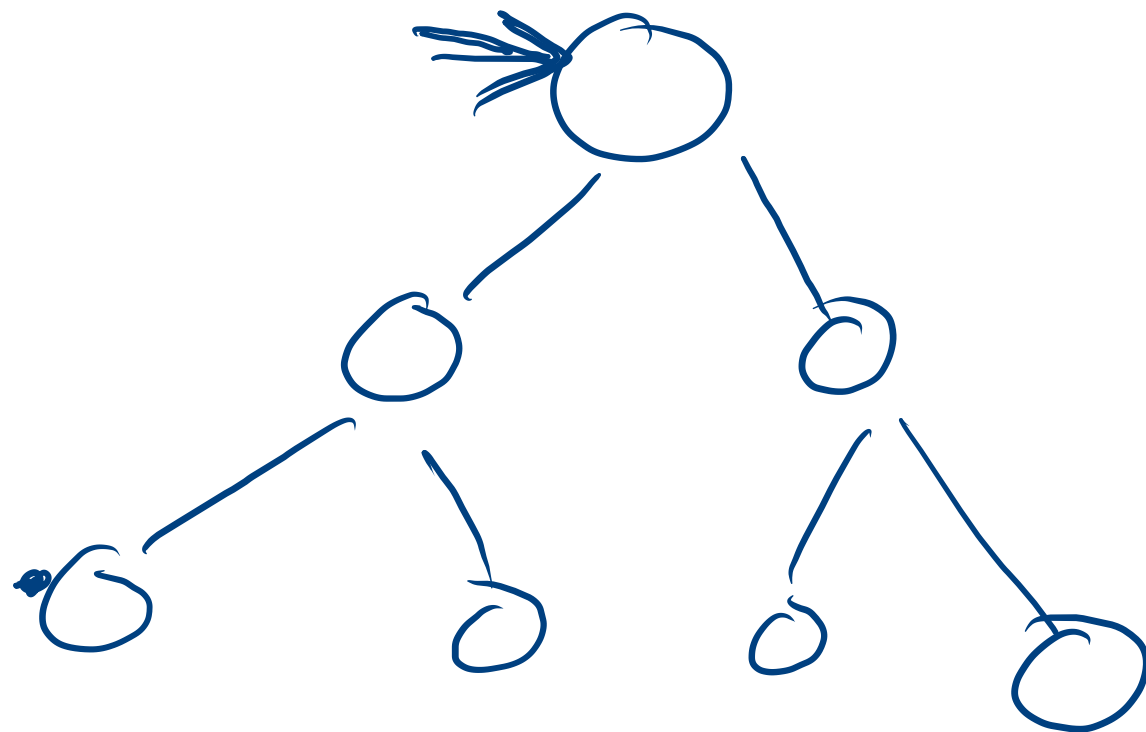
pq



5

ascend / descend

rank
score



Time
 $O(h)$

add

$O(h)$

remove

$O(h)$

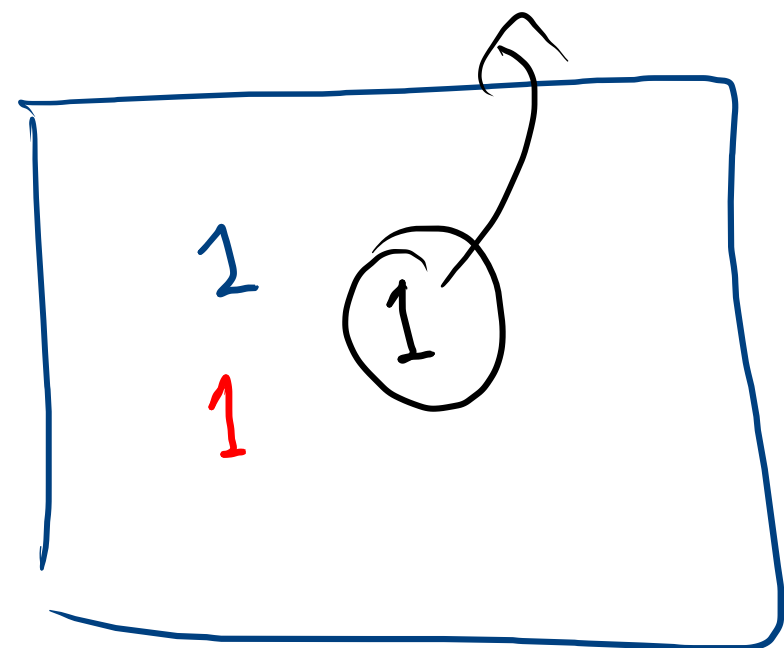
peek

$O(1)$

1

1

1



$k =$ 12 62 22 15 37 99 11 37 98 67 31 84 99

4

99
98
99
84
62

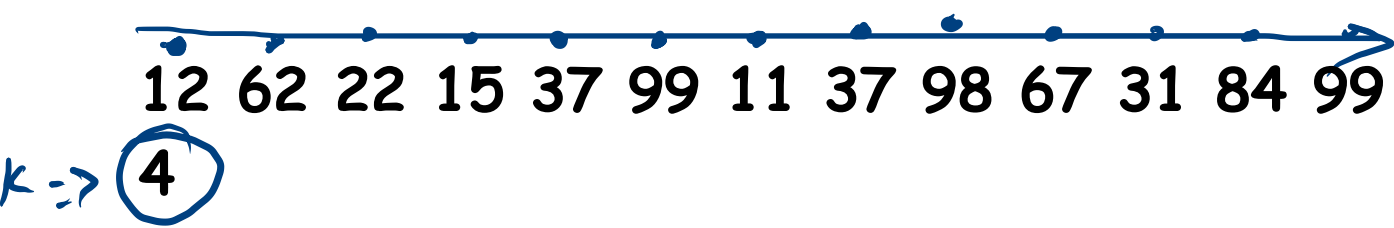
84
98
99
99

add
remove
print

$O(\log(n))$
 $O(\log(n))$
 $O(1)$

Input read only

$n \log(n)$

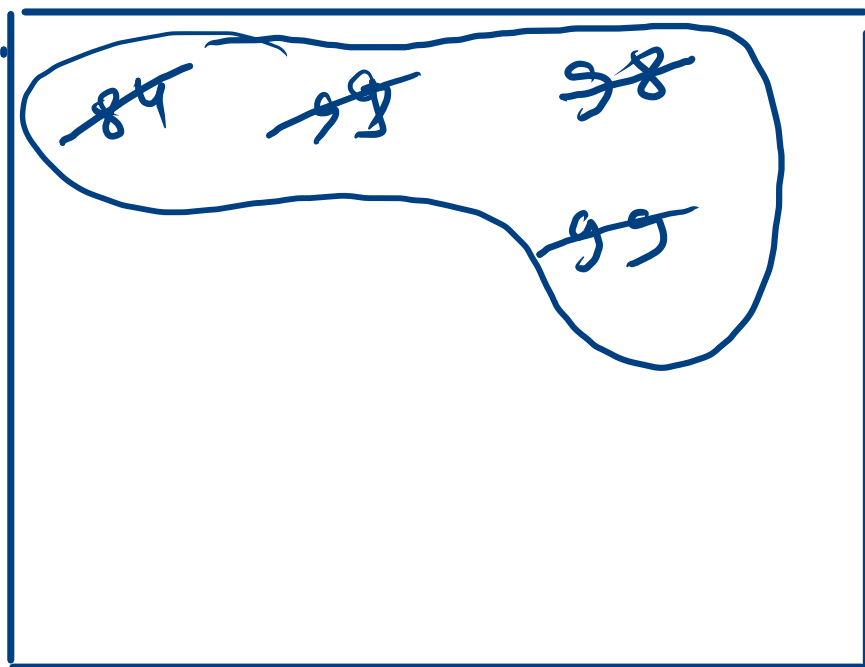


greedy

pd \rightarrow

low
priority

k



84
98
99
99

add(val[i])

if (pq.size > k)
pq.remove()

k

ans → 12 62 22 15 37 99 11 37 98 67 31 84 99

n

(K)

sort $O(n \log(n))$

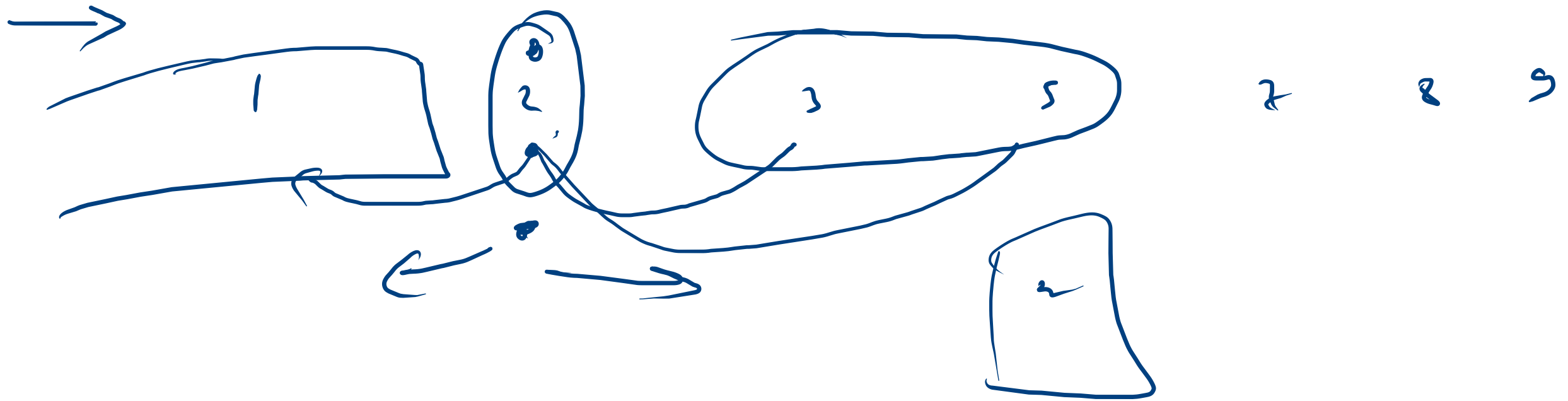
19

$n \left(\overset{\downarrow}{\log(k)} \right)$

•

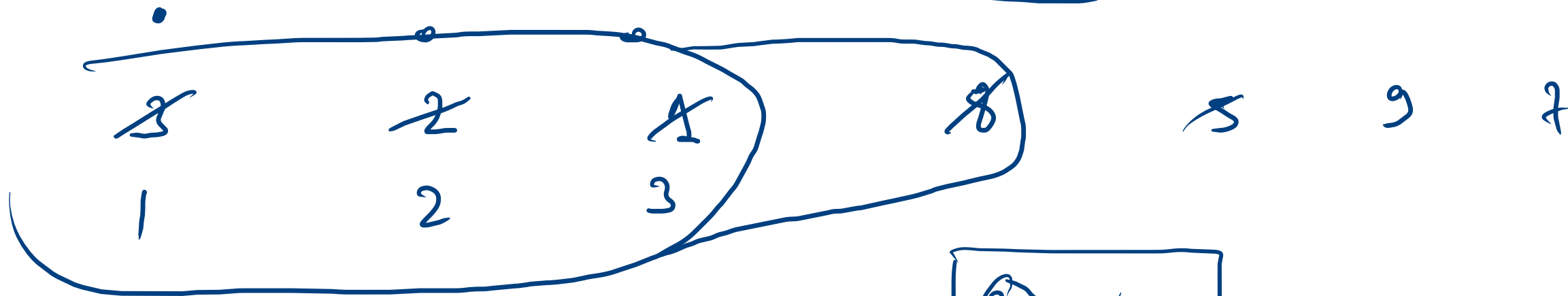
add	$\log(k)$
remove	$\log(k)$

Sorted



k-sorted

$k=2$



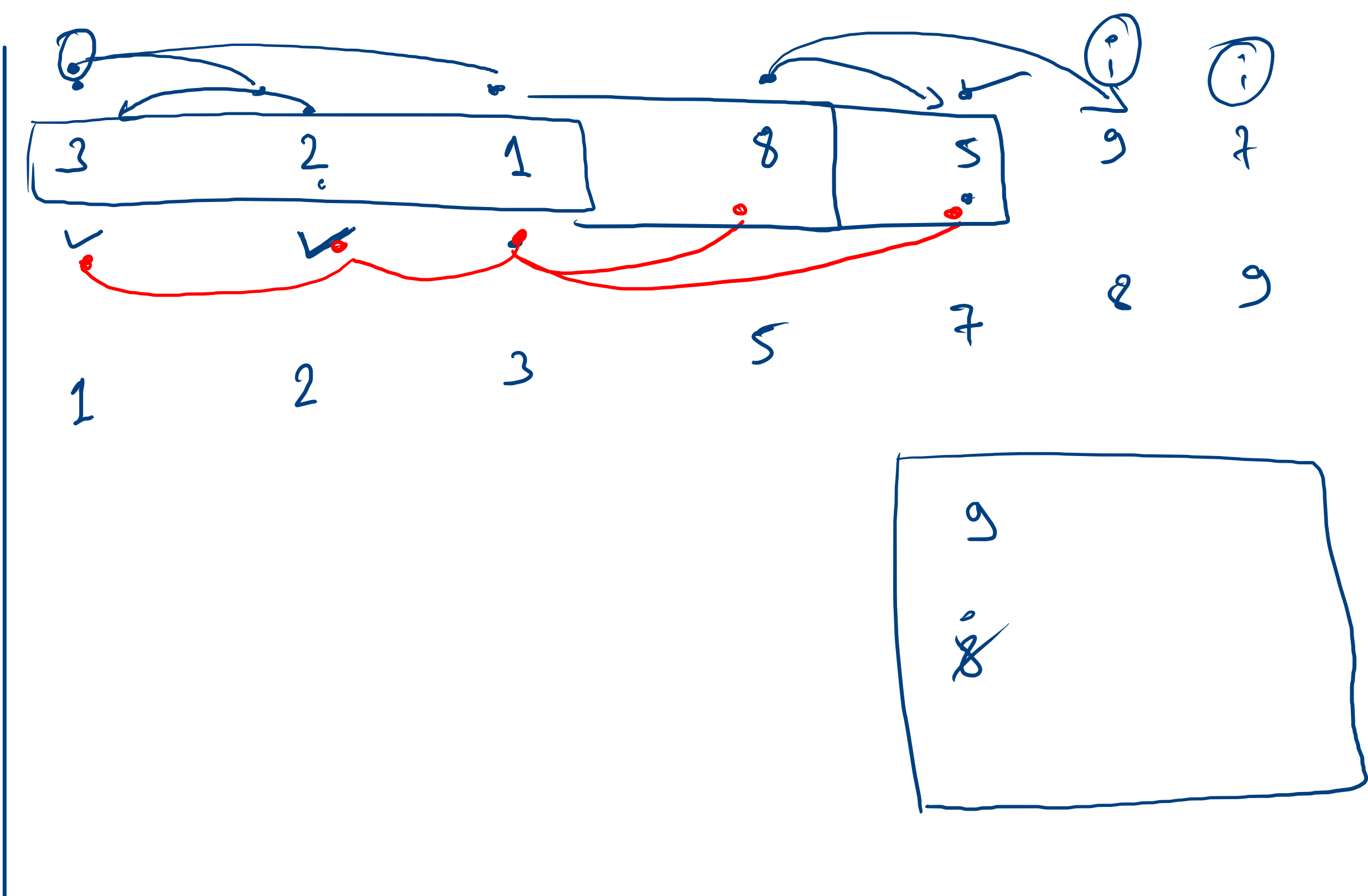
$n \log(n)$

✓ 1 2 3 5 7 8 9

3 2 5
7 8

array

k=2



$k = 3$



3 2 4 1 6 5 7 9 8

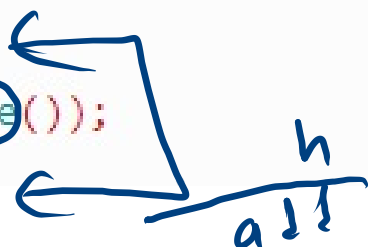
$k \neq 1 \approx k$

```
PriorityQueue<Integer> pq = new PriorityQueue<>();  $k = 3$ 
```

```
for(int i=0; i<=k; i++){  
    pq.add(arr[i]);  
}
```

```
for(int i=k+1; i<arr.length; i++){  
    System.out.println(pq.remove());  
    pq.add(arr[i]);  
}
```

```
while(pq.size() > 0){  
    System.out.println(pq.remove());  
}
```

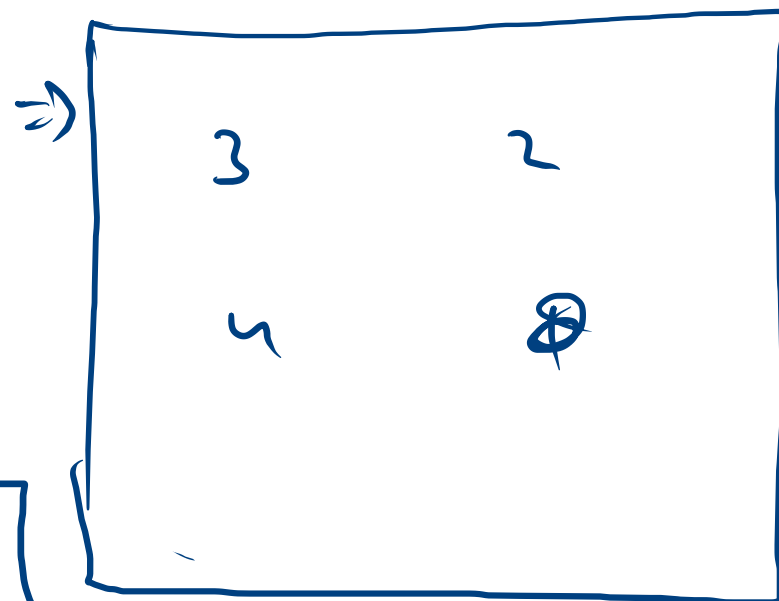


$n \log(k) + n \log(k)$

$O(n \log(k))$

remove $\rightarrow \log(k)$

add $\rightarrow \log(k)$



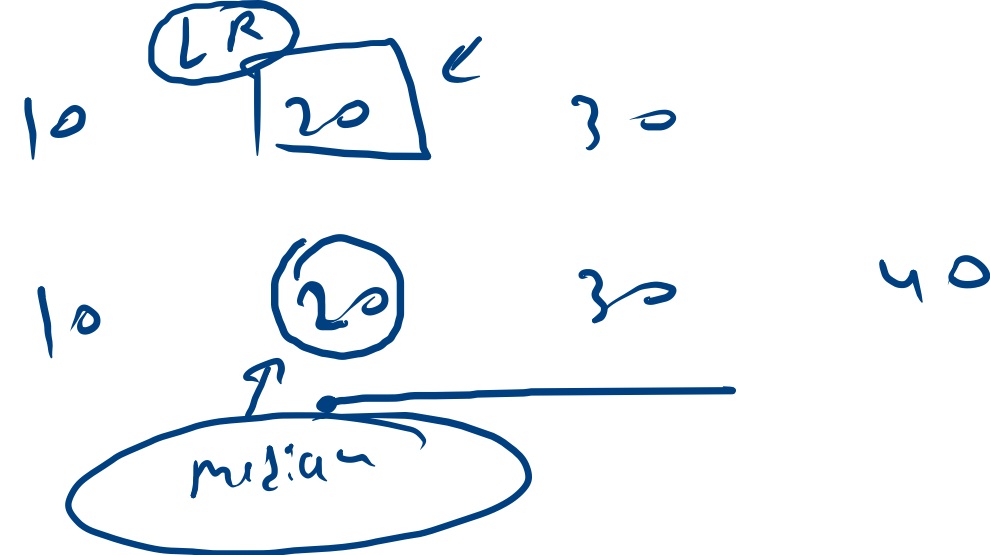
1

add 10 •
add 20 •
add 30 •
add 40 •
peek add
50 peek
~~remove~~
peek •
~~remove~~
~~peek~~
~~remove~~
peek ←
remove ←
peek quit

20
30
30
20
40
40
10
10
50

odd →

even →



like True

5

10

20

30

40

45

Left

PQ

right

PQ

5

35

45

Left peak → median

even

odd

4 →

2

2

5



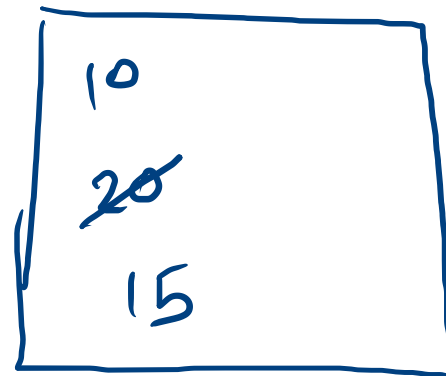
10 |

10 | 20

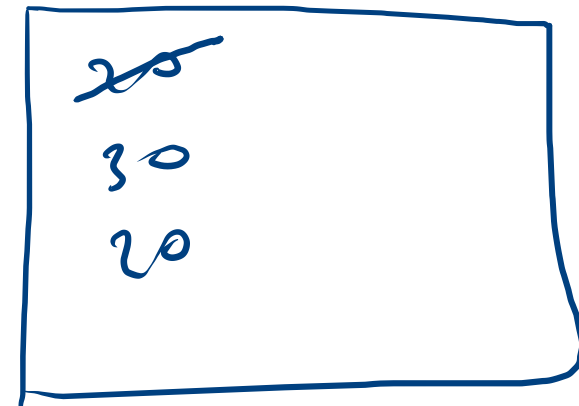
10 (20) | 30

10 15 | 20 30

20
30
15



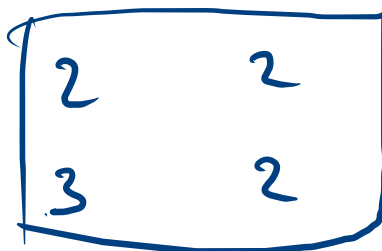
left



right

left insert

4 →



5 →

left-size < right-size
right → left

Left-size == right-size + 2

left → right

is smaller

add → check

{ left part
right part

balance

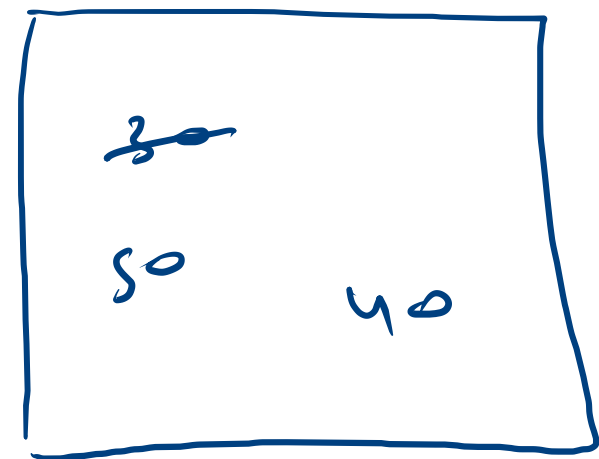
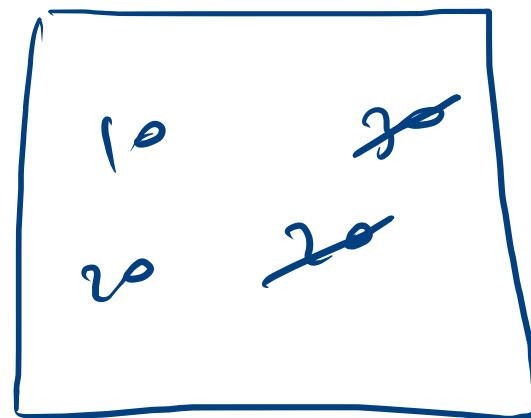
Left-peak == median

even

10 ~~20~~ 30 | 40

odd

10 20 ~~30~~ | 40 50



Left-size() < right-size
right → left

a	b
c	d

$idx < size$

Sox h z

rv

b 0 1 2 3 4 5 6 7
 5 7 9 11 19 55 ~~57~~

1 2 3

32 39


$$k > (\log_2 k)$$

```

class Pair {
    ArraayList<T>
    int idx;
}

```

A hand-drawn rectangular box containing the number '100' on the left and the number '7' on the right.

$$\begin{array}{r} 24 \\ 2 \overline{) 48} \\ \underline{48} \\ 0 \end{array}$$

[1 2 3 5 7 9 10 11 19 20 30 32
39 40 50 55 57

A hand-drawn red outline of a trapezoid. The shape has a shorter top horizontal edge, a longer bottom horizontal edge, and two slanted sides connecting them. The lines are slightly irregular, suggesting a hand-drawn sketch.

2

class ==

Lambda function

1 2 3 5 7 9 10 11 19 20 20

32 39

```

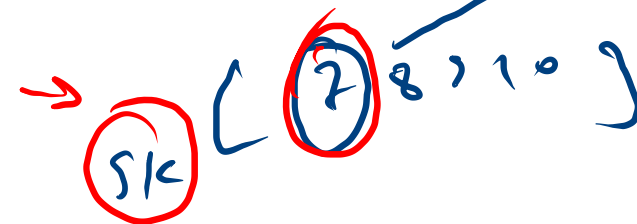
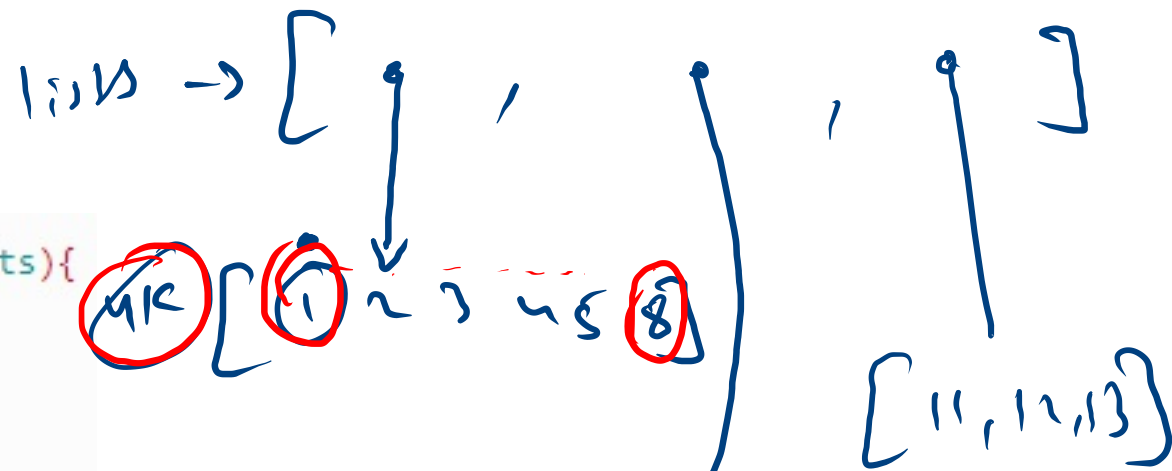
public static ArrayList<Integer> mergeKSortedLists(ArrayList<ArrayList<Integer>> lists){
    ArrayList<Integer> rv = new ArrayList<>();

    PriorityQueue<Pair> pq = new PriorityQueue<>();

    for(ArrayList<Integer> list : lists){
        Pair p = new Pair(list, 0);
        pq.add(p);
    }

    return rv;
}

```



this. curr = b(b)

a
a > b
+1
p

8 < 2

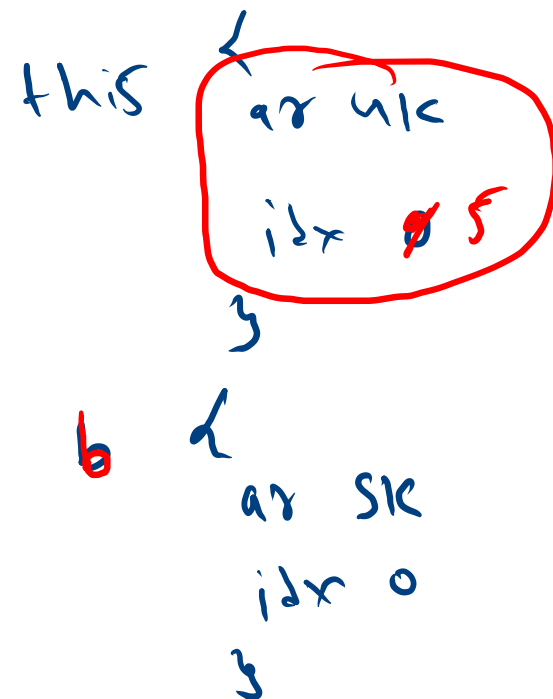
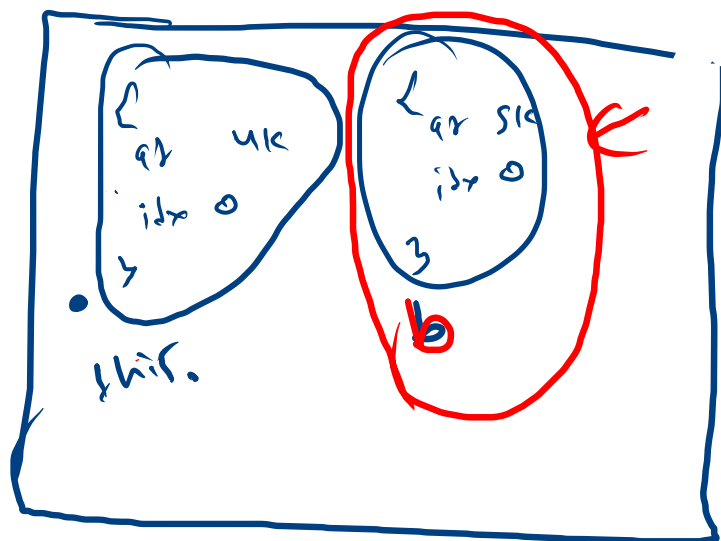
8 > 2

```

public int compareTo(Pair b){
    if(ar.get(idx) < b.ar.get(b.idx)){
        return -1;
    }
    if(ar.get(idx) > b.ar.get(b.idx)){
        return +1;
    }
    return 0;
}

```

-1

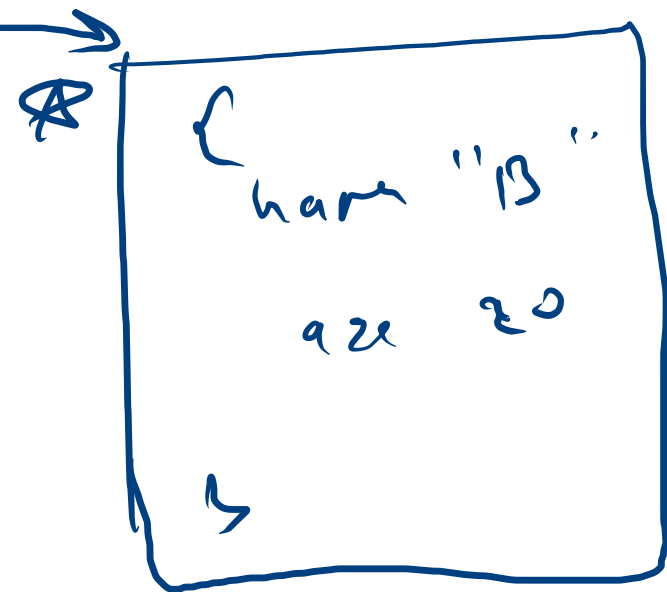
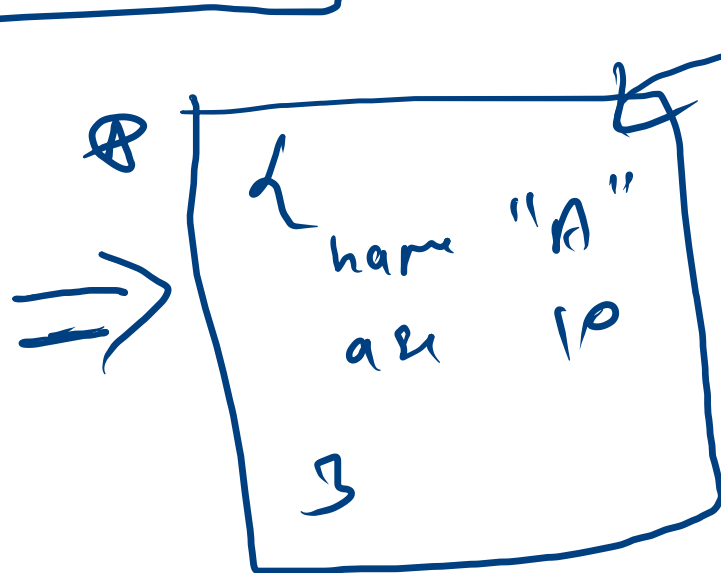
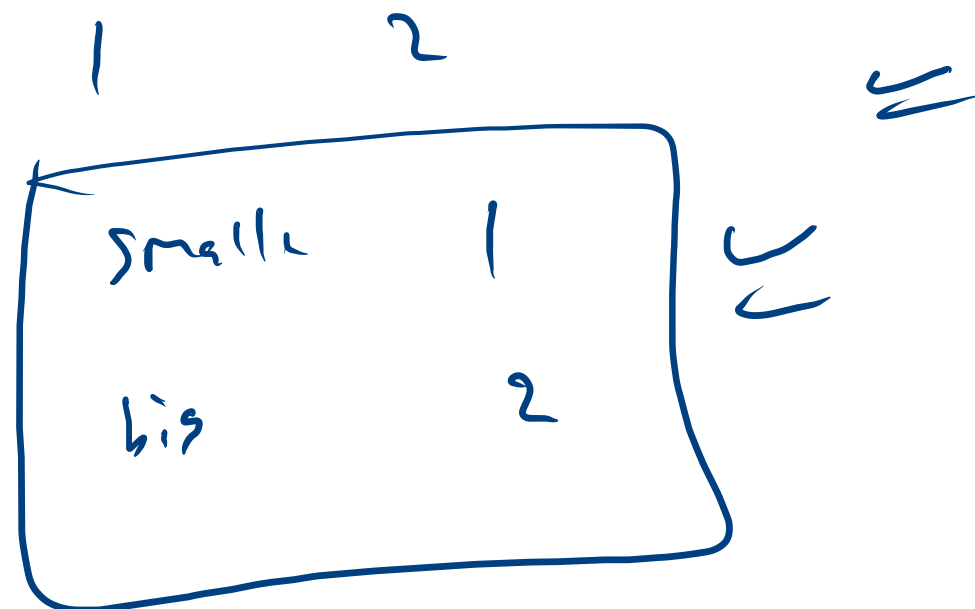
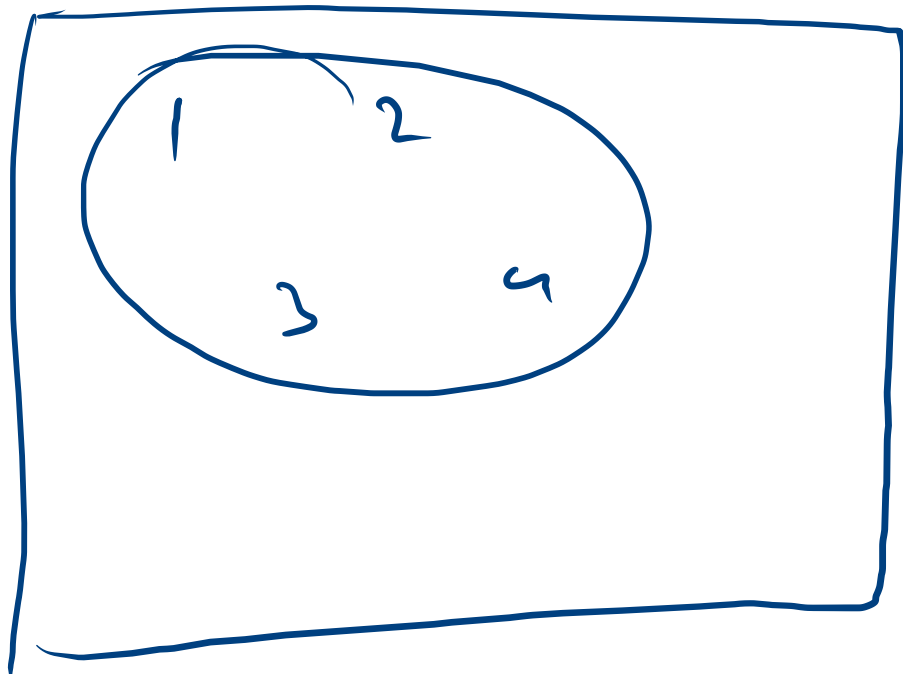


• 1 2 4
• 3 4 5 6

Los verify
Implement

[1 2 3 4 4 5 6]

ask



A hand-drawn diagram illustrating a hierarchical tree structure with four levels, labeled 1, 2, 3, and 4. The root node is 'c' at level 1. Level 2 contains nodes 'a' and 'b'. Level 3 contains nodes 'd' and 'f'. Level 4 contains nodes 'e' and 'g'. Arrows point to the levels from the right.

```

graph TD
    c((c)) --- a((a))
    c --- b((b))
    a --- d((d))
    a --- f((f))
    d --- e((e))
    f --- g((g))
  
```