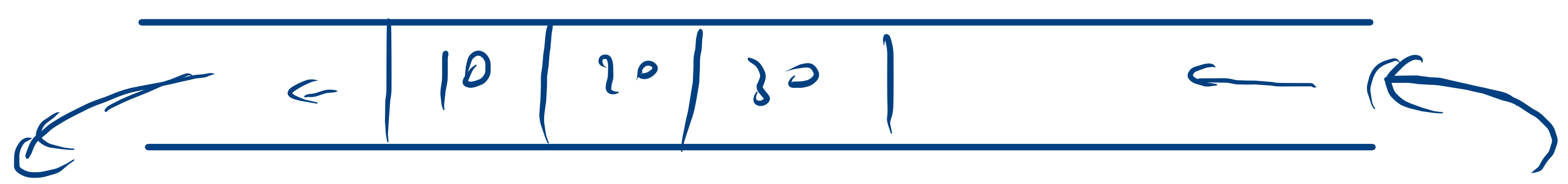
Priority Queue



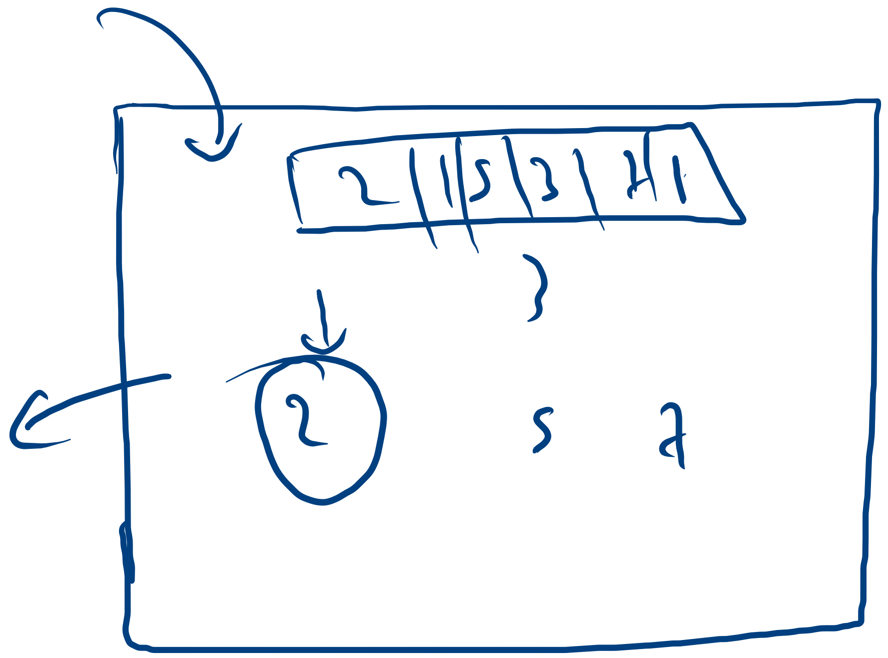
PQ

add

remove 1

peek 1

highest priority



5

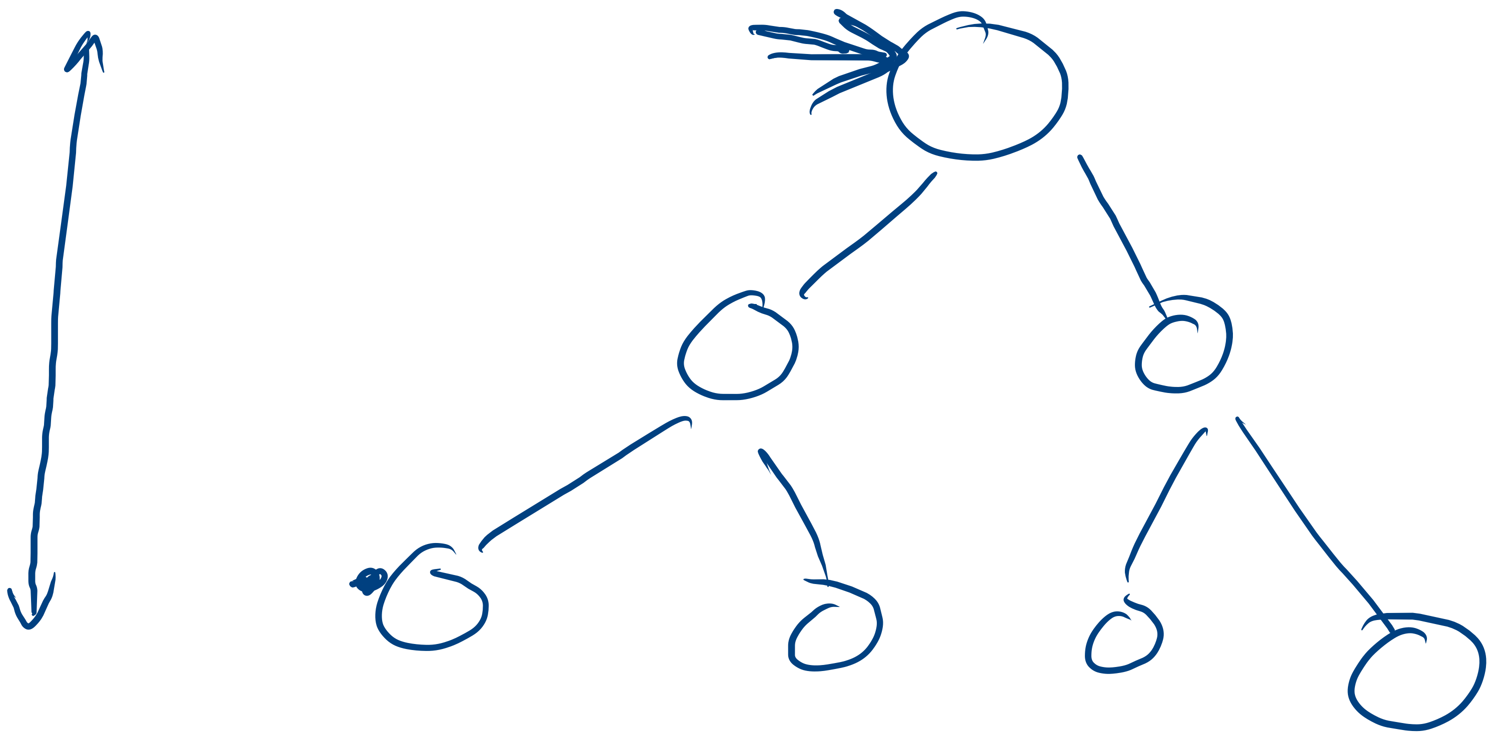
ascend / descend

rank

score

Stream

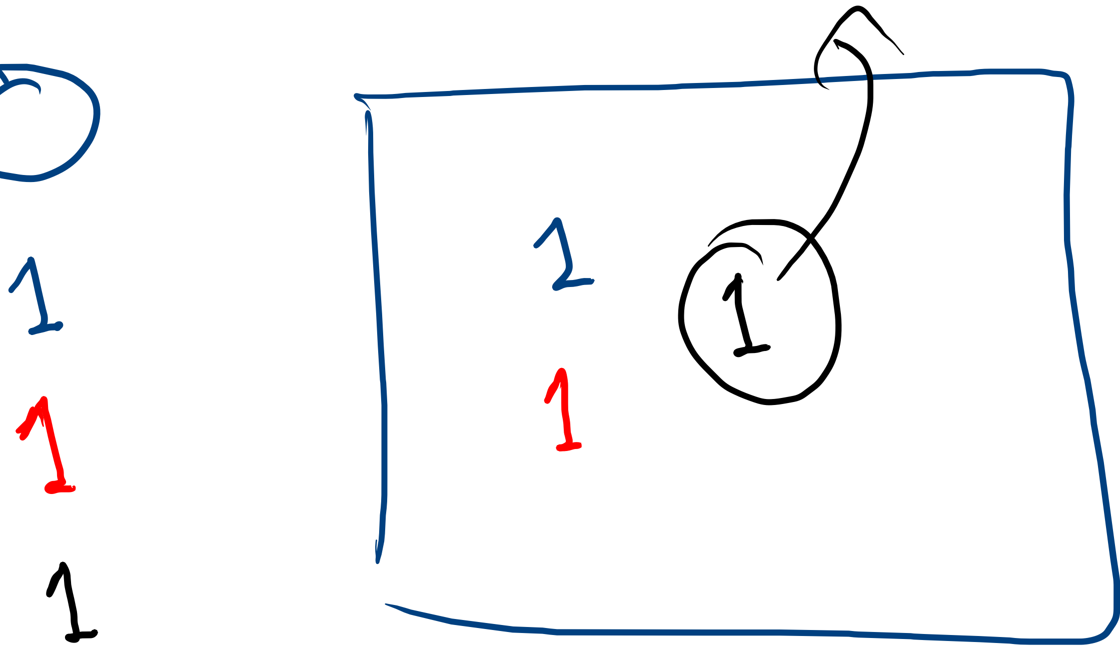
priority



add	$O(n)$
remove	$O(n)$
peek	$O(1)$

find

$O(n)$



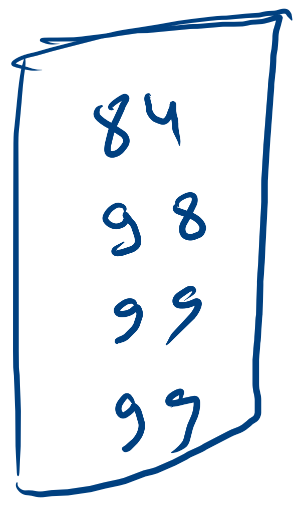
12 62 22 15 37 99 11 37 98 67 31 84 99

k = 4

4

add	O(log(n))
remove	O(log(n))
peek	O(1)

99
98
95
84
67



n log(n)

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

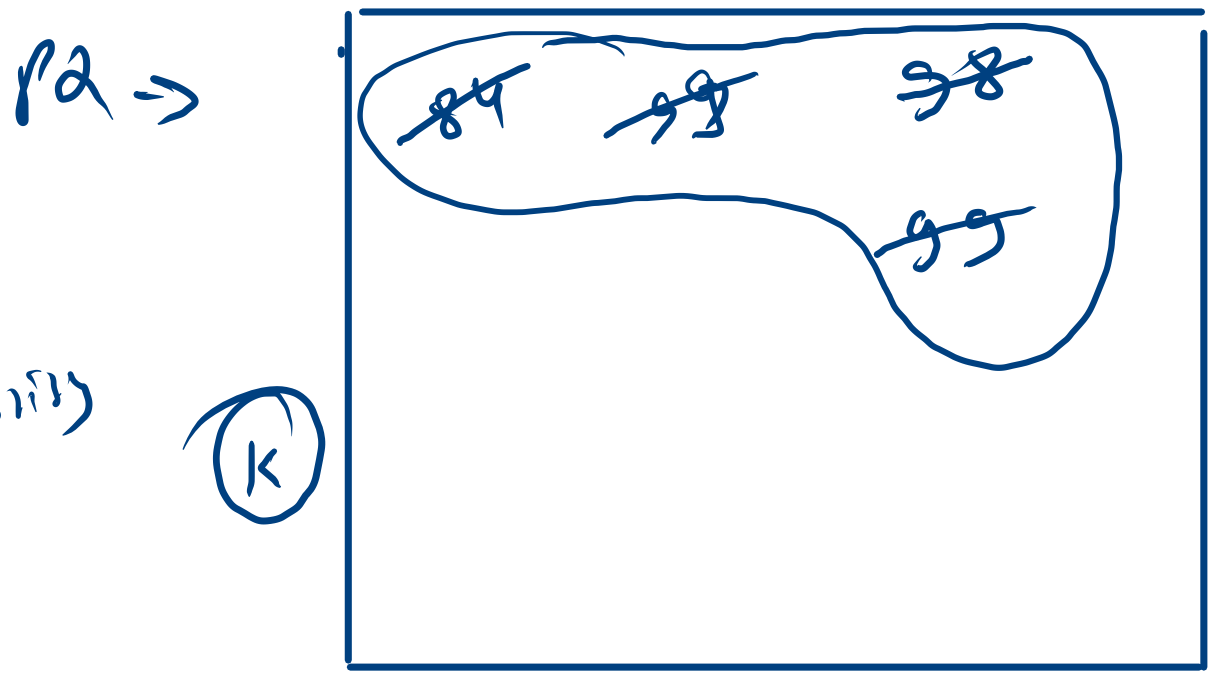
k => $\boxed{4}$

greatest

add(val[i])

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

k



84
98
99
99

array $\xrightarrow{\hspace{8cm}}$
12 62 22 15 37 99 11 37 98 67 31 84 99

n

$\textcircled{k}$

• add	$\log(k)$
remove	$\log(k)$

goal $O(n\log(n))$

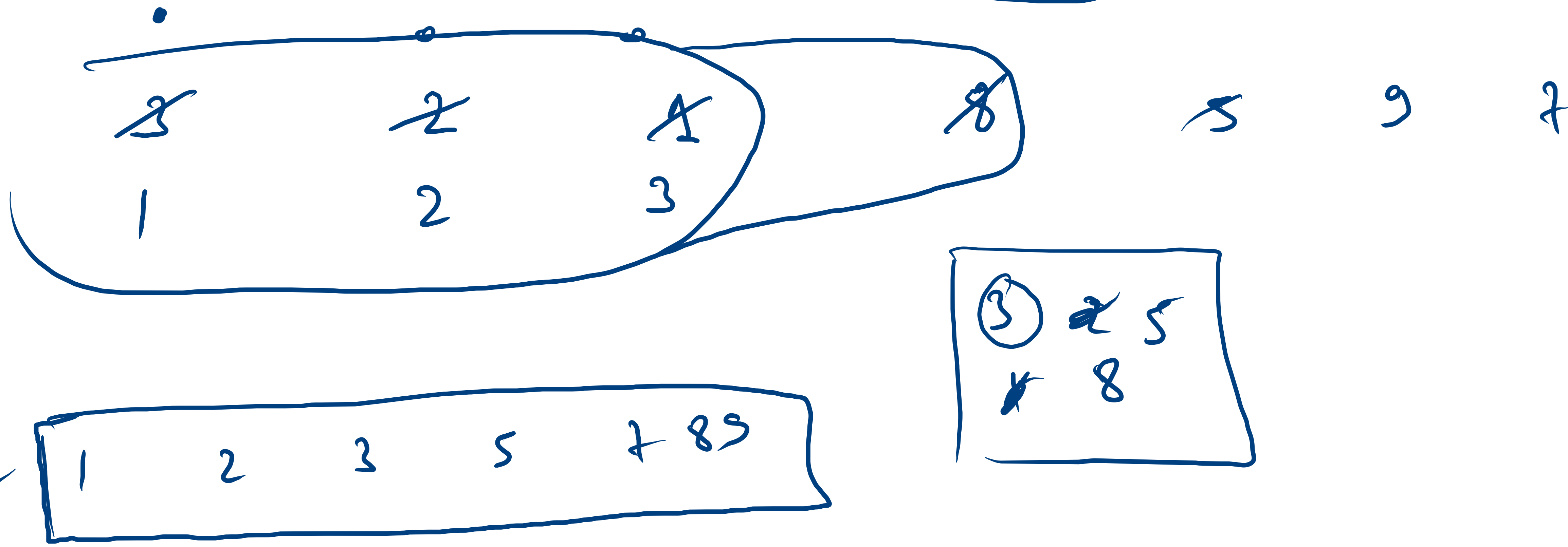
pq

$n\left(\boxed{\log(k)}\right)$

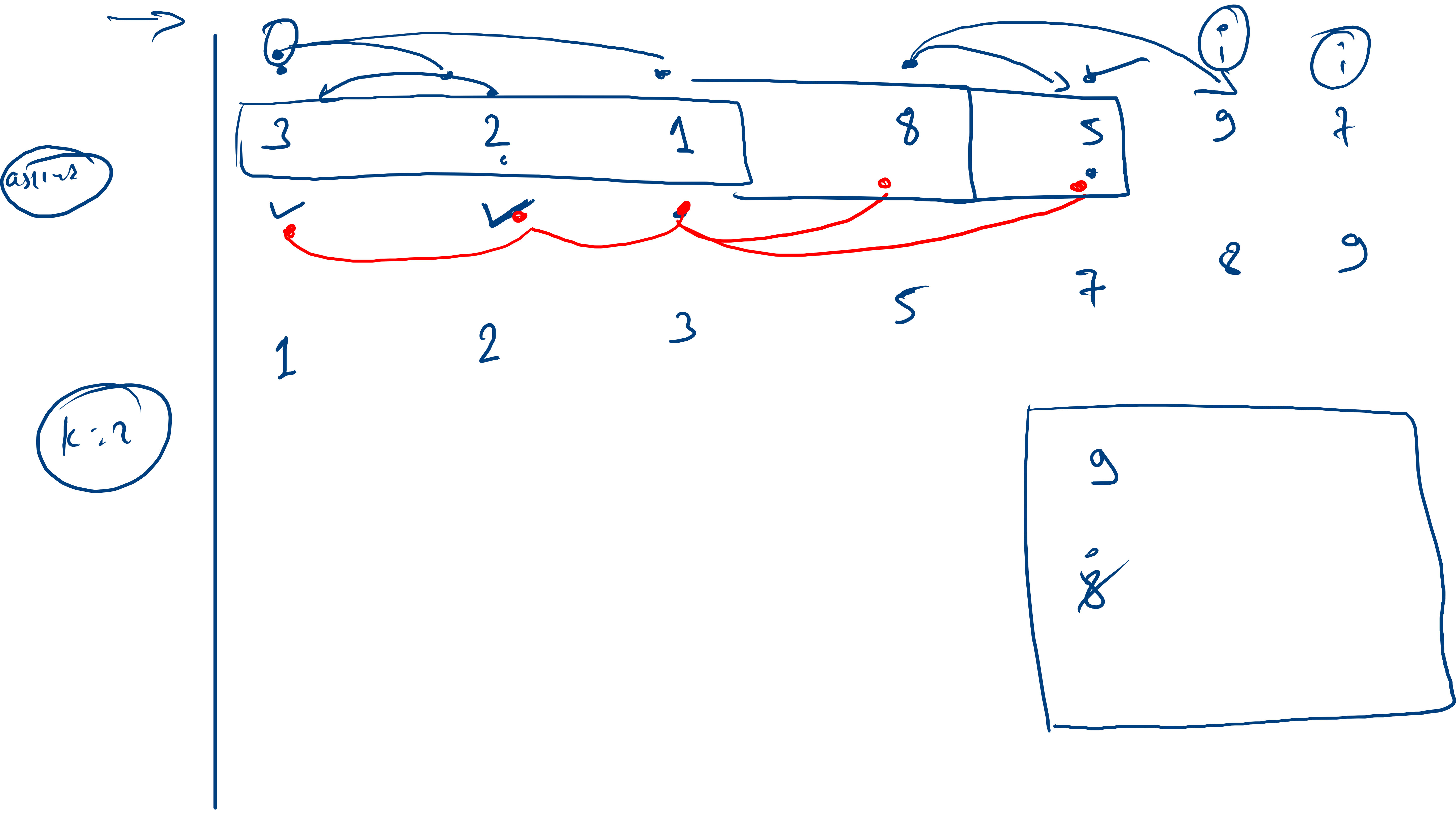
Sorted

k- sorted

$\boxed{k = \textcircled{2}}$



nlog(n)


k=2
3 2 1 8 5 9 7
8 9
5 7
1 2 3
9
~~8~~

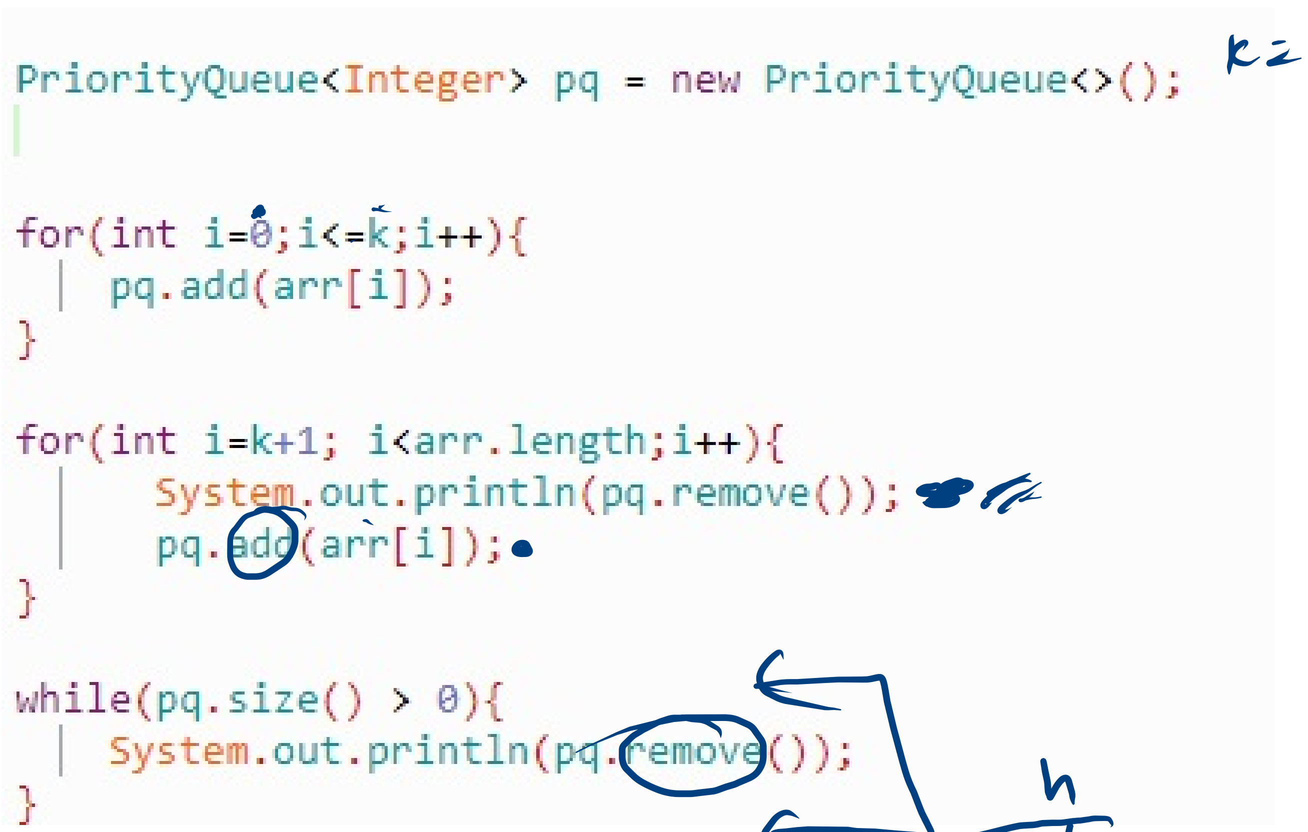
$k = 3$

3 2 4 1 6 5 7 9 8

$k = 3$

$\boxed{k+1} \approx \boxed{k}$

1

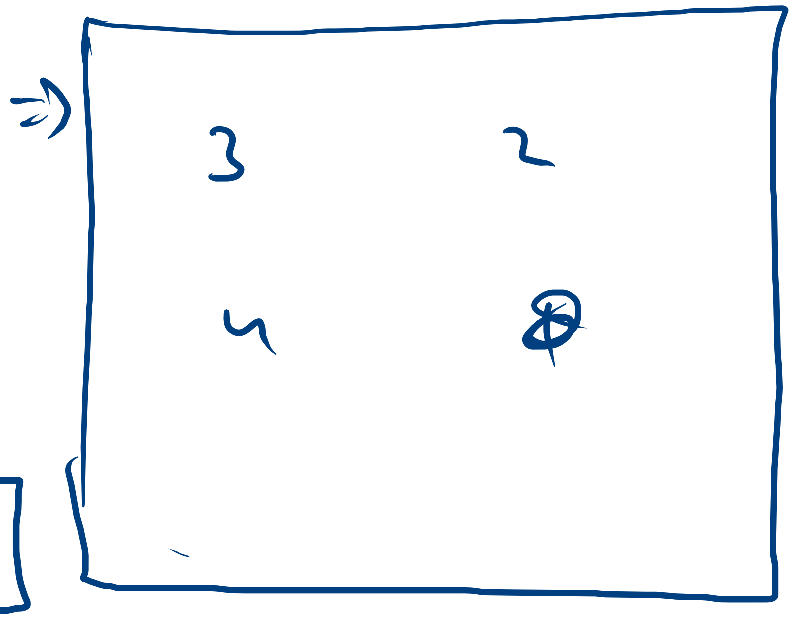


```
PriorityQueue<Integer> pq = new PriorityQueue<>();

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());
}
```



$\frac{n}{add}$

$\boxed{n \log(k) + n \log(k)}$

$O(n \log(k))$

remove $\rightarrow \log(k)$

add $\rightarrow \log(k)$

add 10 •

add 20 •

add 30 •

add 40 •

peek add

50 peek

~~remove~~

peek •

~~remove~~

~~peek~~

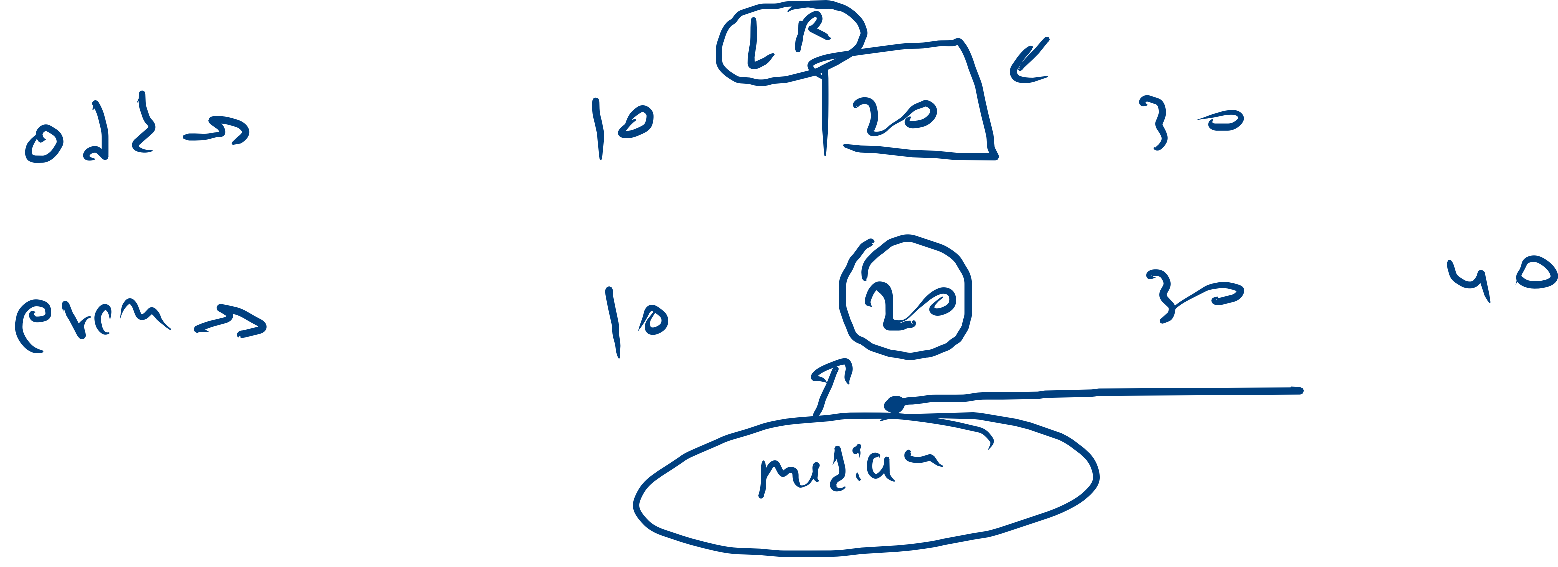
~~remove~~

peek ←

remove ←

peek quit

20
30
30
20
40
40
10
10
50



lide True

5 10 20 | 30 40 45

left

P2

right

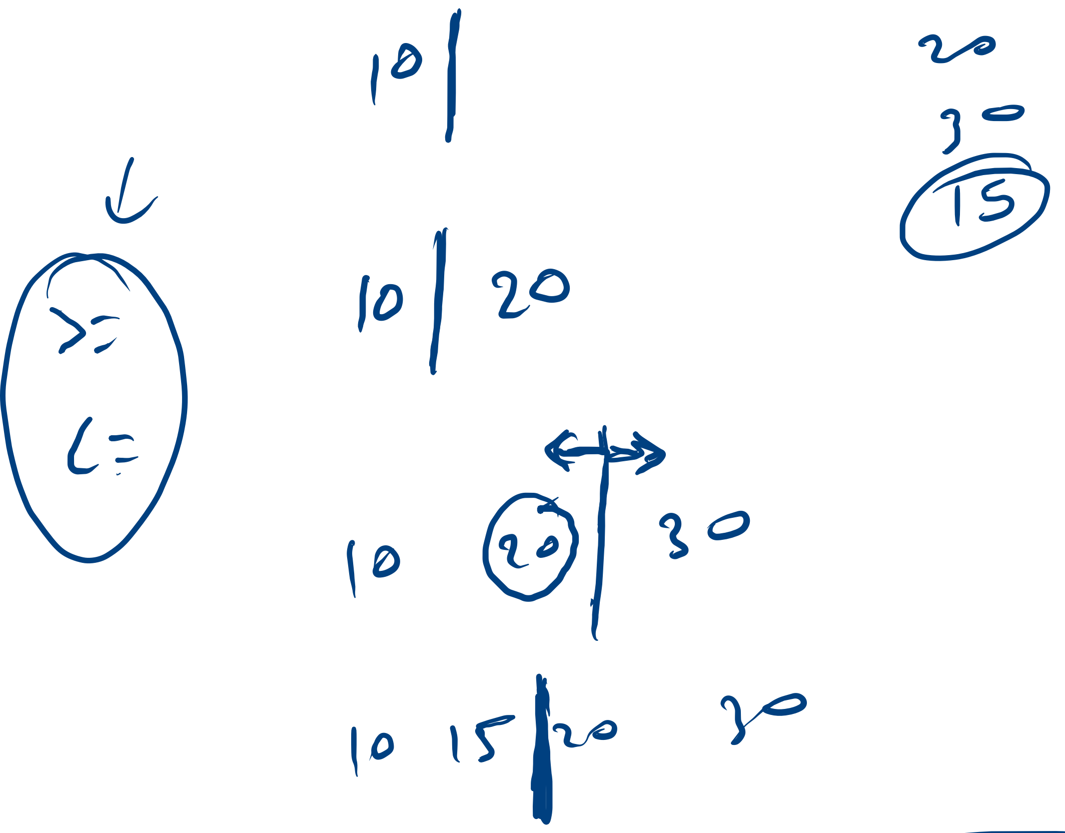
P2

5
35
45

Left peek → median

even 4 → 2
2

odd 5



4 →	2	2
5 →	3	2

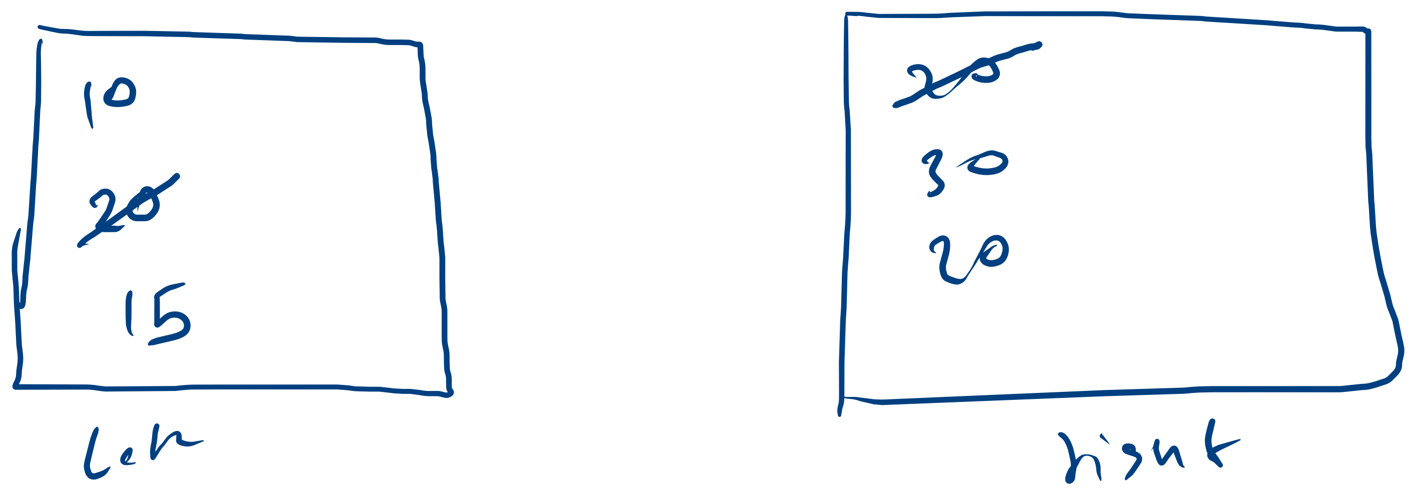
left larger

Right smaller

is

add → check { left part / right part

balance



left.size < right.size

right → left

left.size == right.size + 2

left → right

$\boxed{\text{left.peek == median}}$

even 10 ~~20~~ 30 | 40

odd 10 20 ~~30~~ | 40 50

10 ~~30~~
20 ~~20~~

~~30~~
50 40

left.size() < right.size

right $\rightarrow$ left

K

a b
c d

Sorted

idx < size

↘ sorted

TV

a 10 20 30 40 50

0 1 2 3 4 5 6 7

b 5 7 9 11 19 55 ~~57~~

7

d 1 2 3

c 32 39

k

k > log$_2$ k

[1 2 3 5 7 9 10 11 19 20 30 32
35 40 50 55 57]

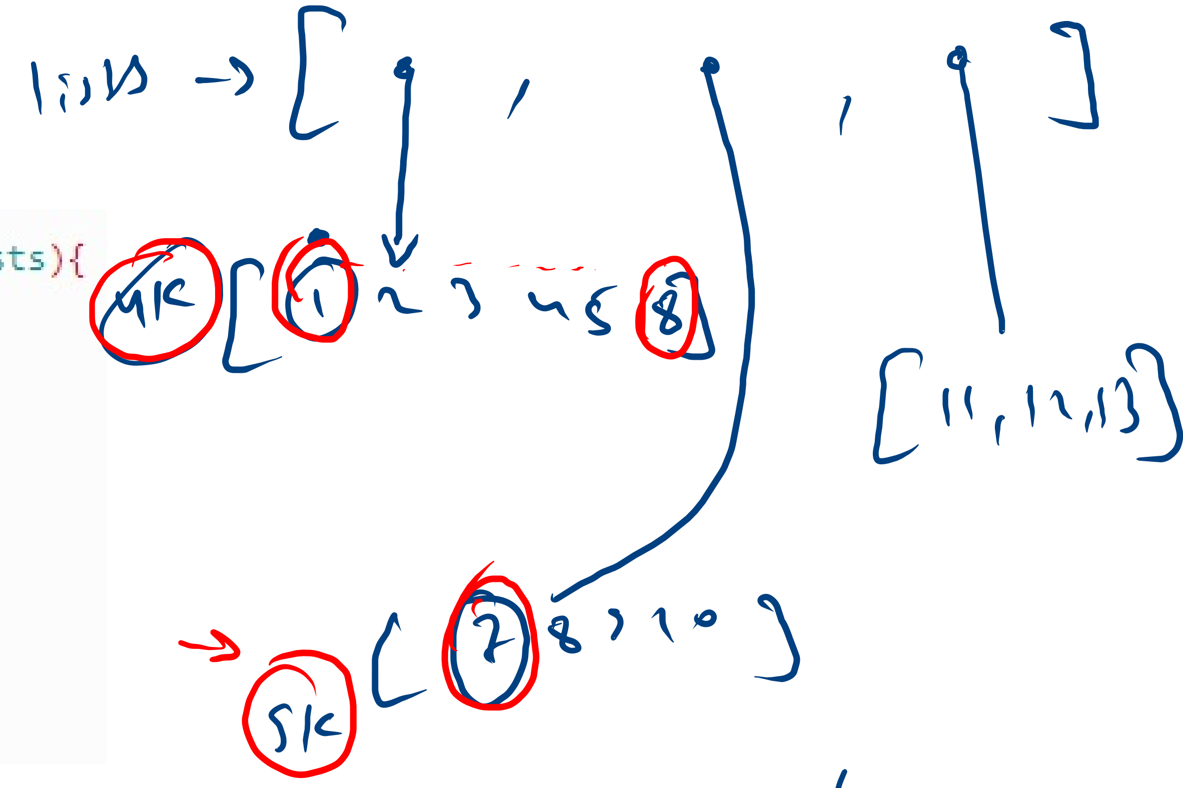
2 — class
— Lambda function

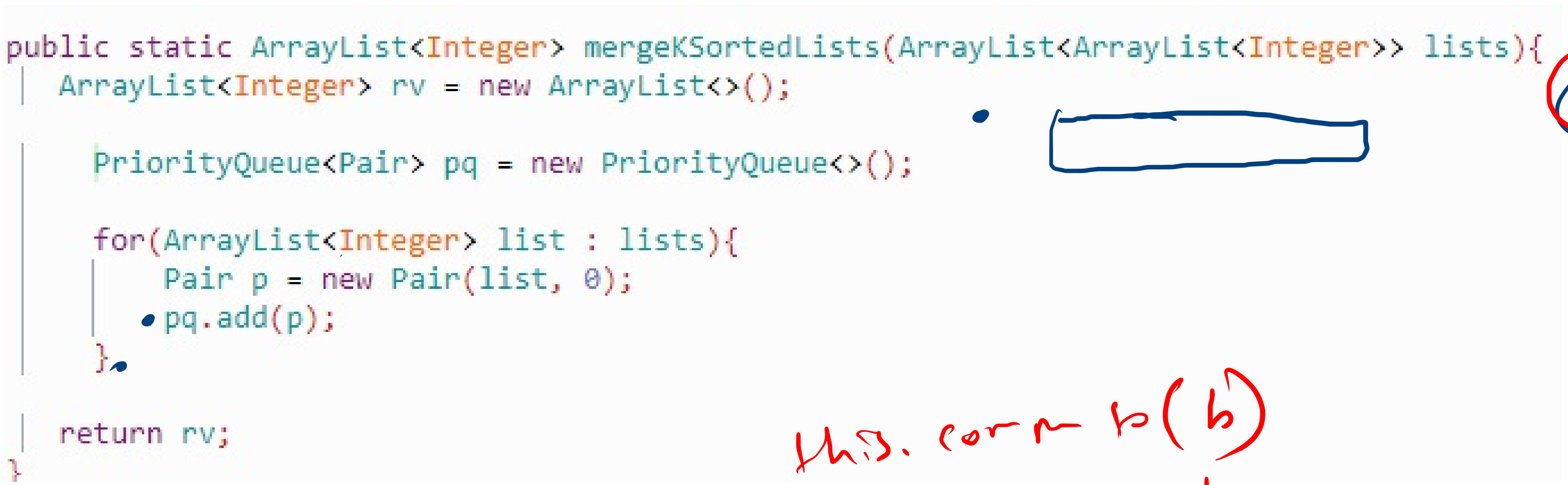
```
class Pair {
    ArrayList<I~>
    int idx;
}
```

100 7

[1 2 3 5 7 9 10 11 19 20 30
32 35

2 4
8
16
64 32





```java
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;
}
```

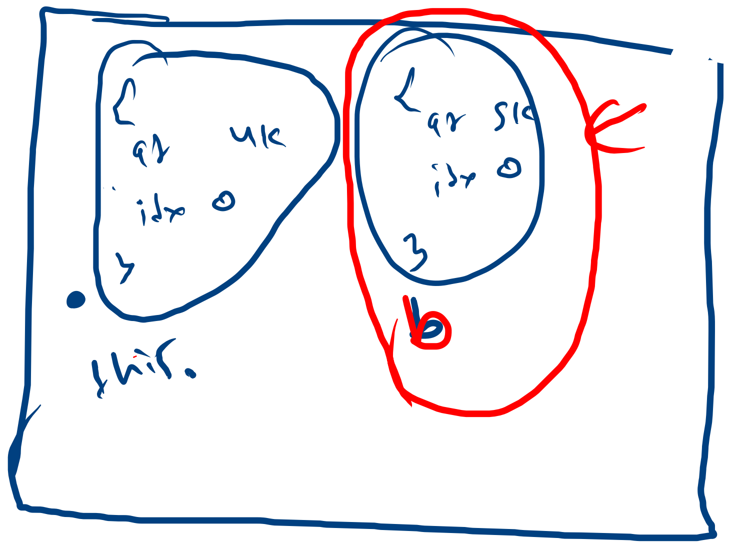
this. compareTo(b)

a b

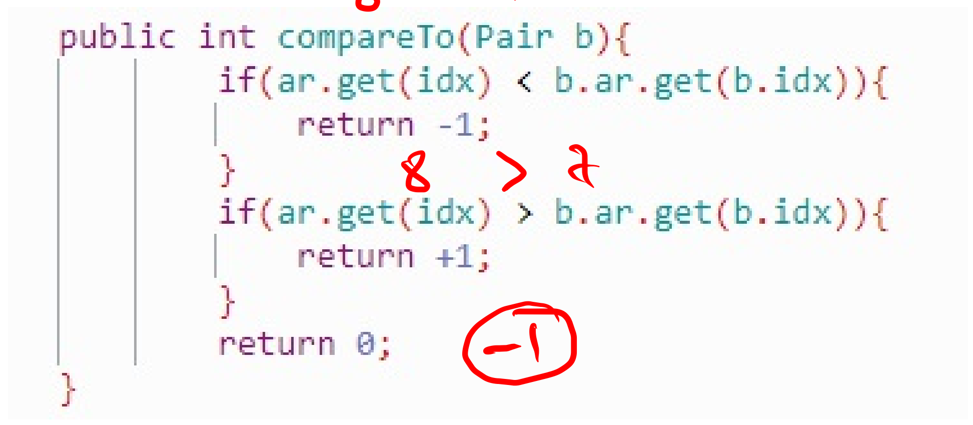
a > b

+1

p



8 < 7



```java
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;
}
```

8 > 7

-1

this {
ar 4k
idx ~~0~~ 5
}

b {
ar 5k
idx 0
}

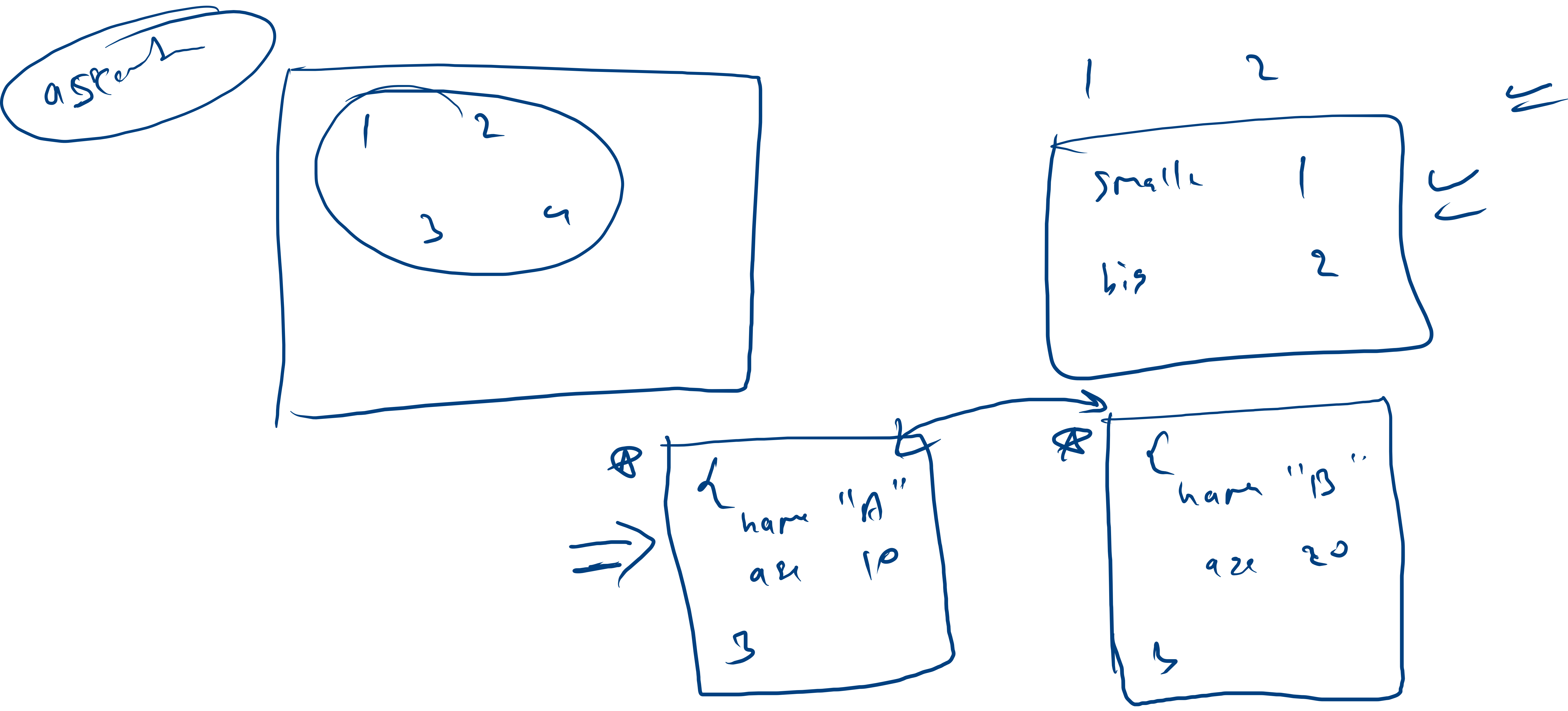
1 2 4

3 4 5 6

los unify

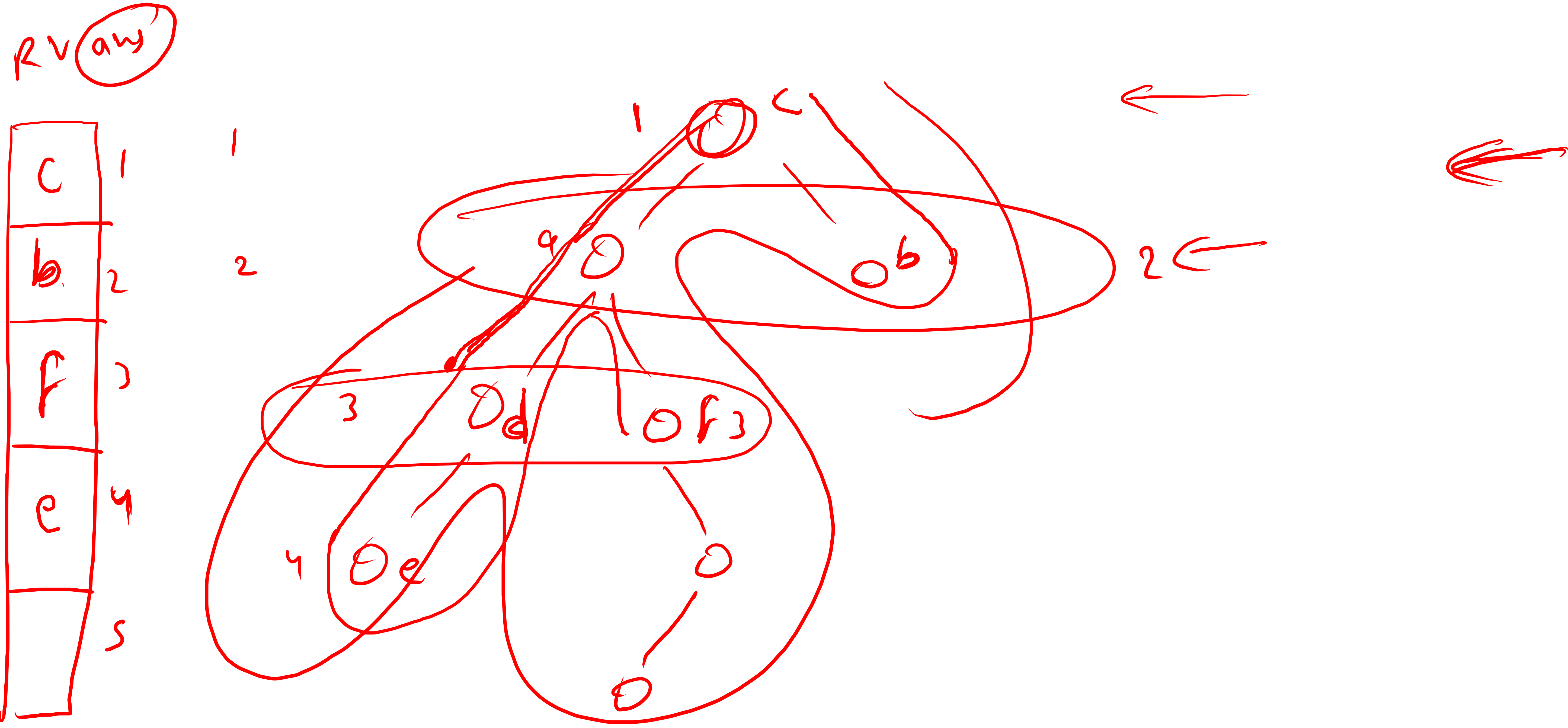
duplicados

$[\overset{0}{1}\ 2\ 3\ 4\ 4\ 5\ 6]$



aspect
1
2
3
4
1
2
smaller 1
big 2
{
name "A"
age 10
}
{
name "B"
age 20
}

RV ~~aw~~


c
b
f
e
1
2
3
4
5
1
2
1
c
a
b
2
3
d
f
3
4
e
2