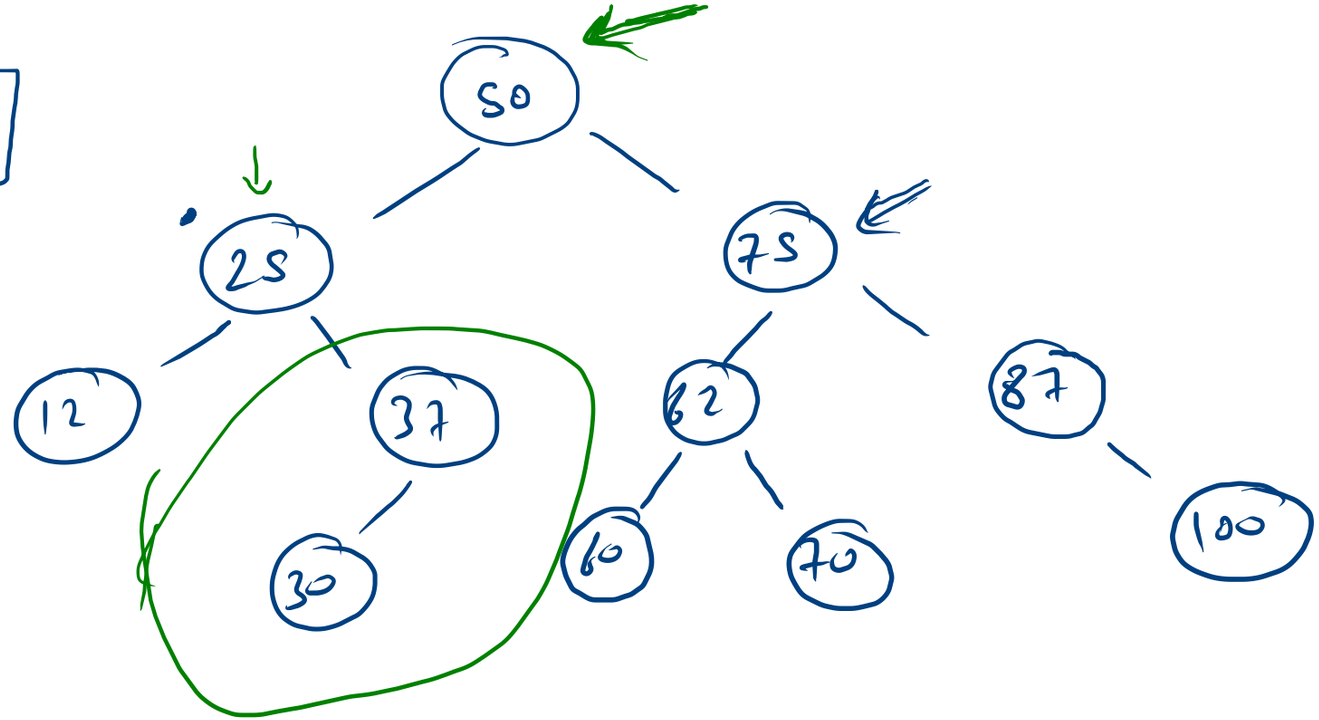
$\boxed{\text{tars} = 100}$

75



25 + 75

30 + 70

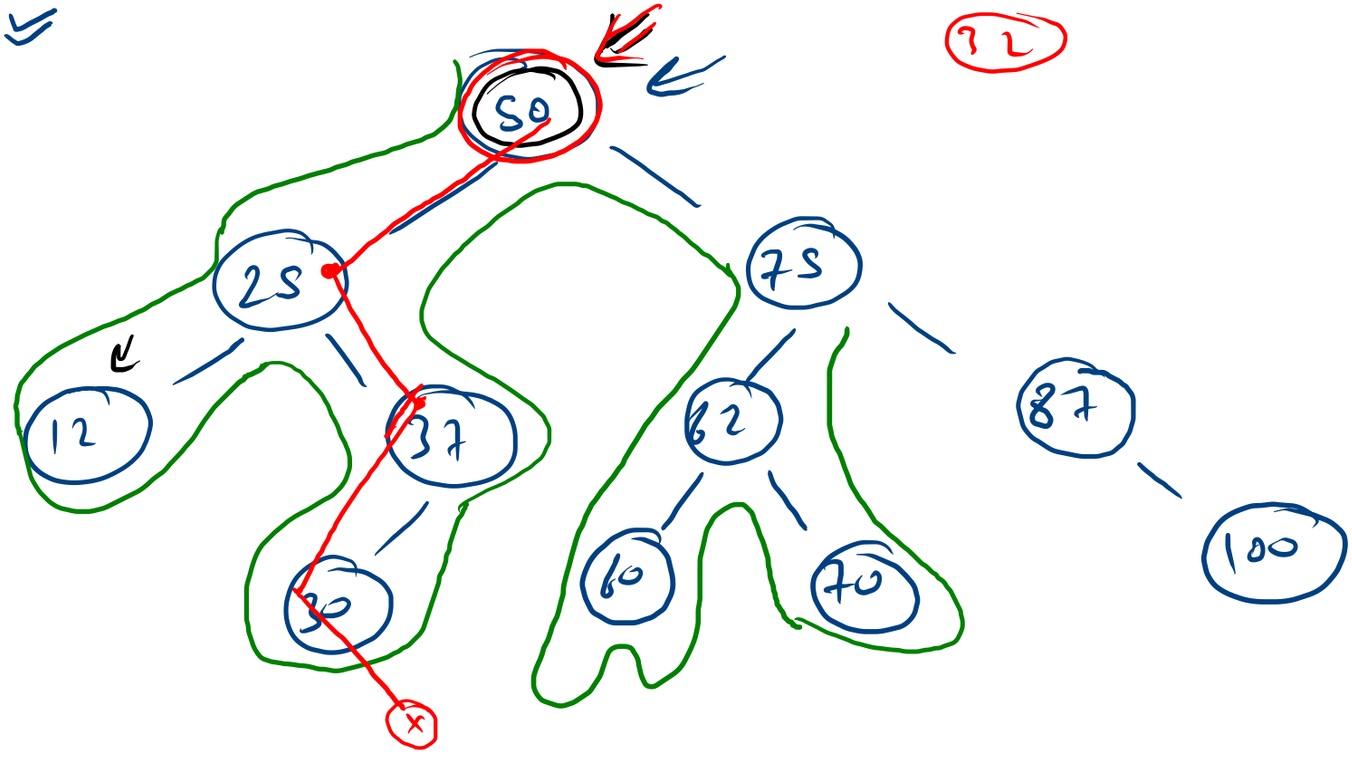
25 75

30 70

$\boxed{a + b} = 100$

log(n) → lin 2 ✓

n log(n)



72

node	comp	
25	< 75	✓
30	< 70	✓
70	> 30	✗
75	> 25	✗
50	= 50	✗

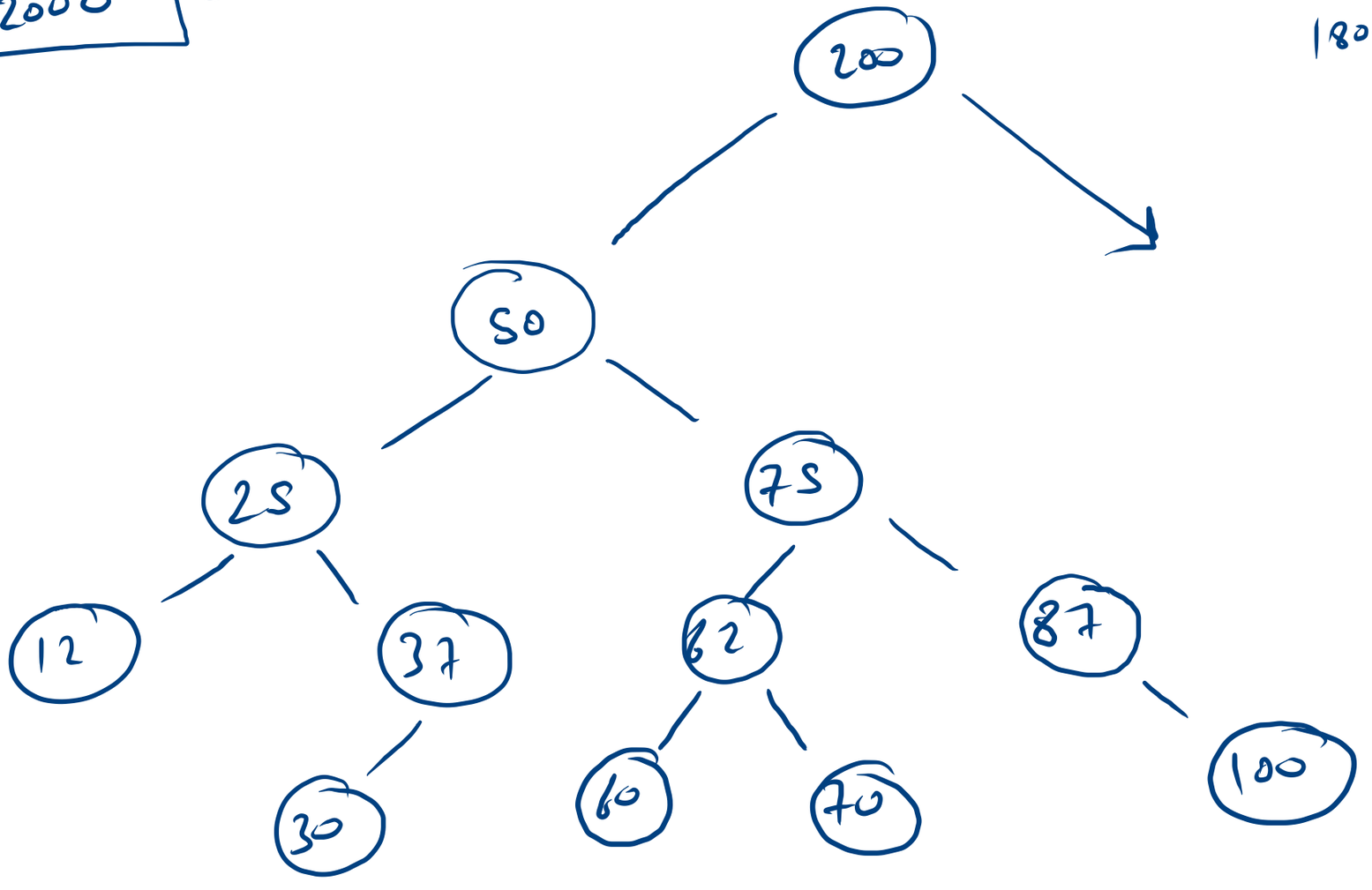
inorder

comp = tar - node.data

find (root, comp) ✓ ✗

Ans = 2000 ✓

1800



int → ?

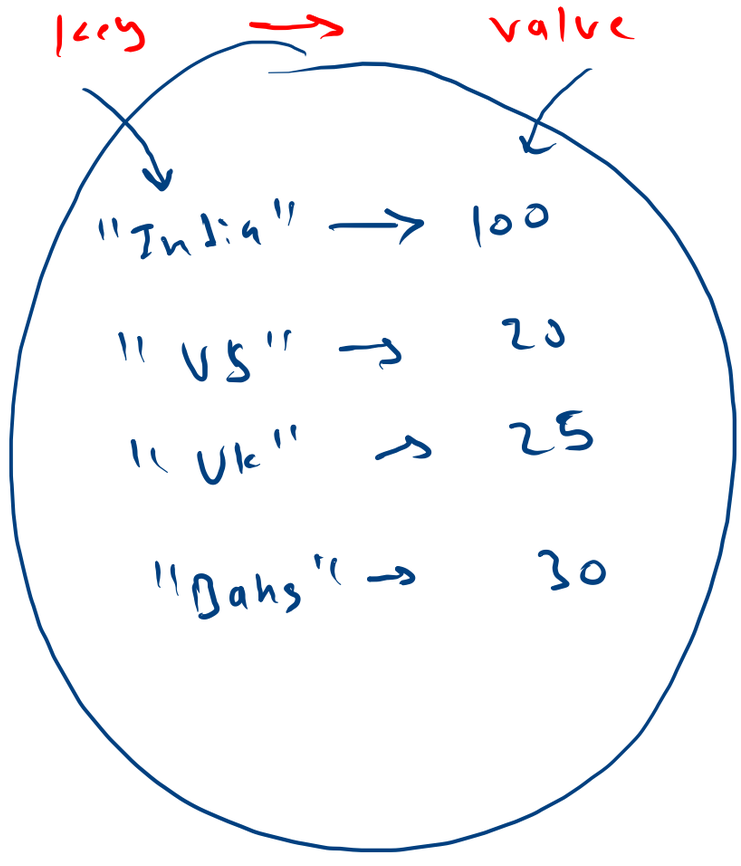
index → ?

? → ?

string → string

add / update → O(1)

remove O(1)



Scanner → string

put (key, value)

exist update
add new

value = get (key)

not exist null

True, false contains key (key)

remove (key)

zmszeqxllzvheq

z

z +x3

m 1

s 1

e +2

q +2

x 1

l → 2

v 1

h 1

char

<Character, Integer>

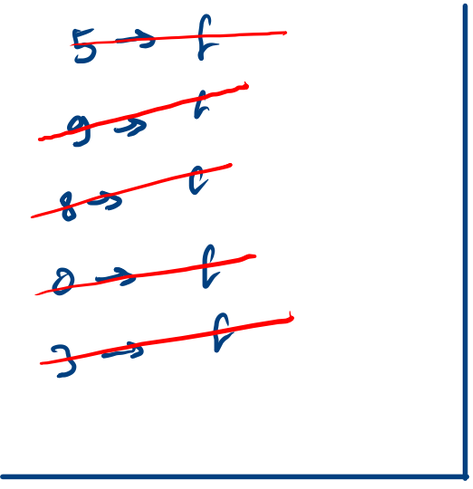
Integer(2)
cre++
put(2, cre)

z → x2 3	q → x 2
m → 1	x → 1
s → 1	l → x 2
e → x 2	v → 1
	h → 1

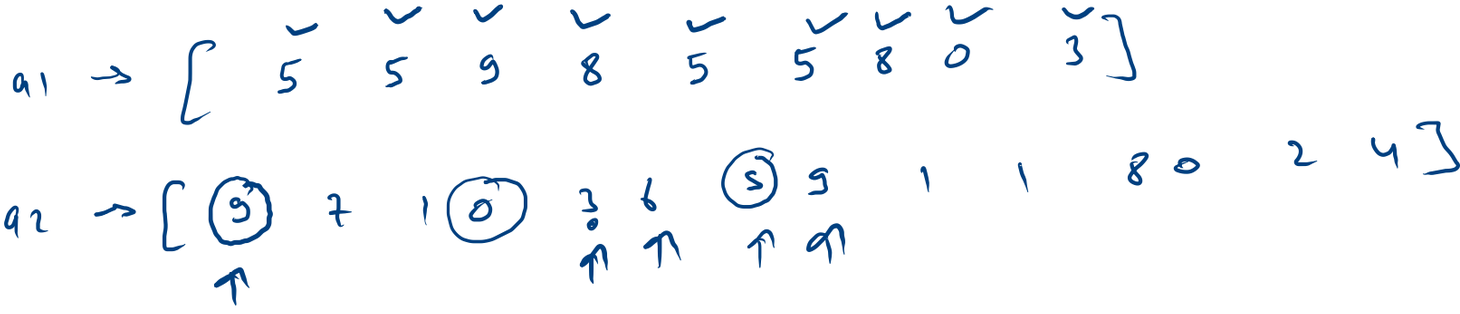
q1 → [5 5 9 8 5 5 8 0 3]

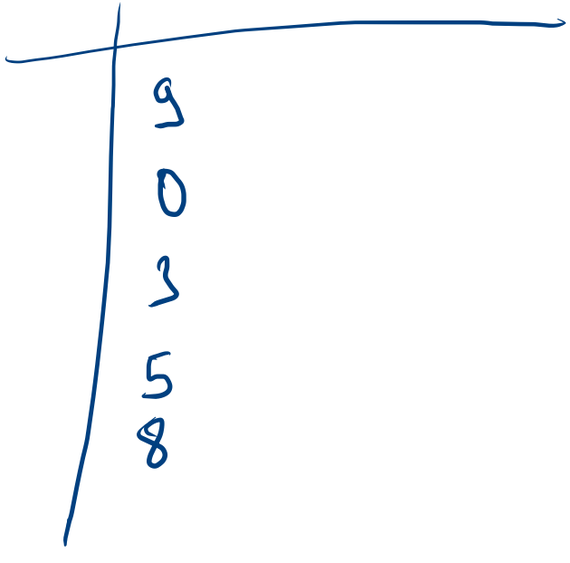
q2 → [9 7 1 0 3 6 5 9 1 1 8 0 2 4]

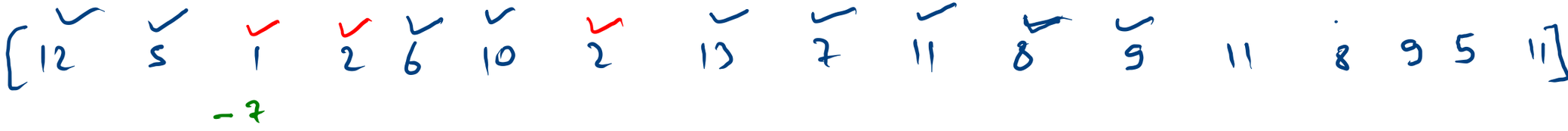
9
0
3
5
8

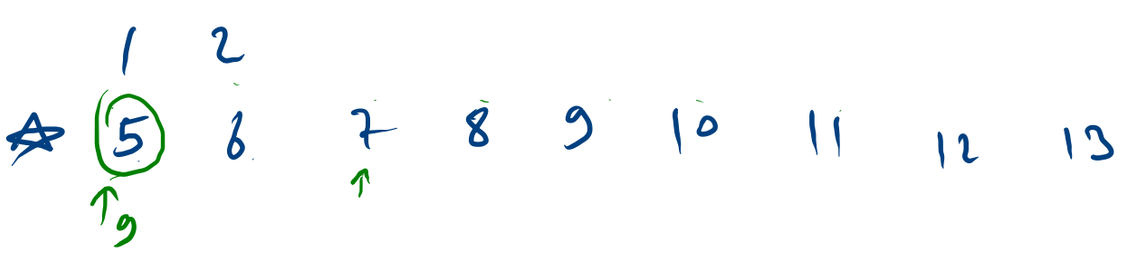


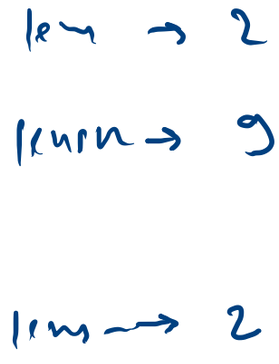
O(1)



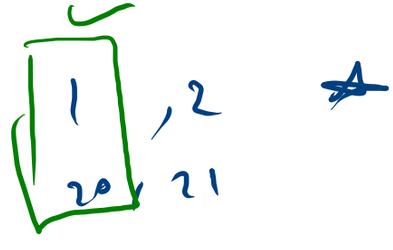








20 21



17 12 5 1 2 10 2
13 7 11 8 9 11 8
9 5 6 11

[12 5 1 2 6 10 2 13 7 11 8 9 11 8 9 5 11]

n	O(1)
12	13
5	7
1	11
2	8
6	9
10	

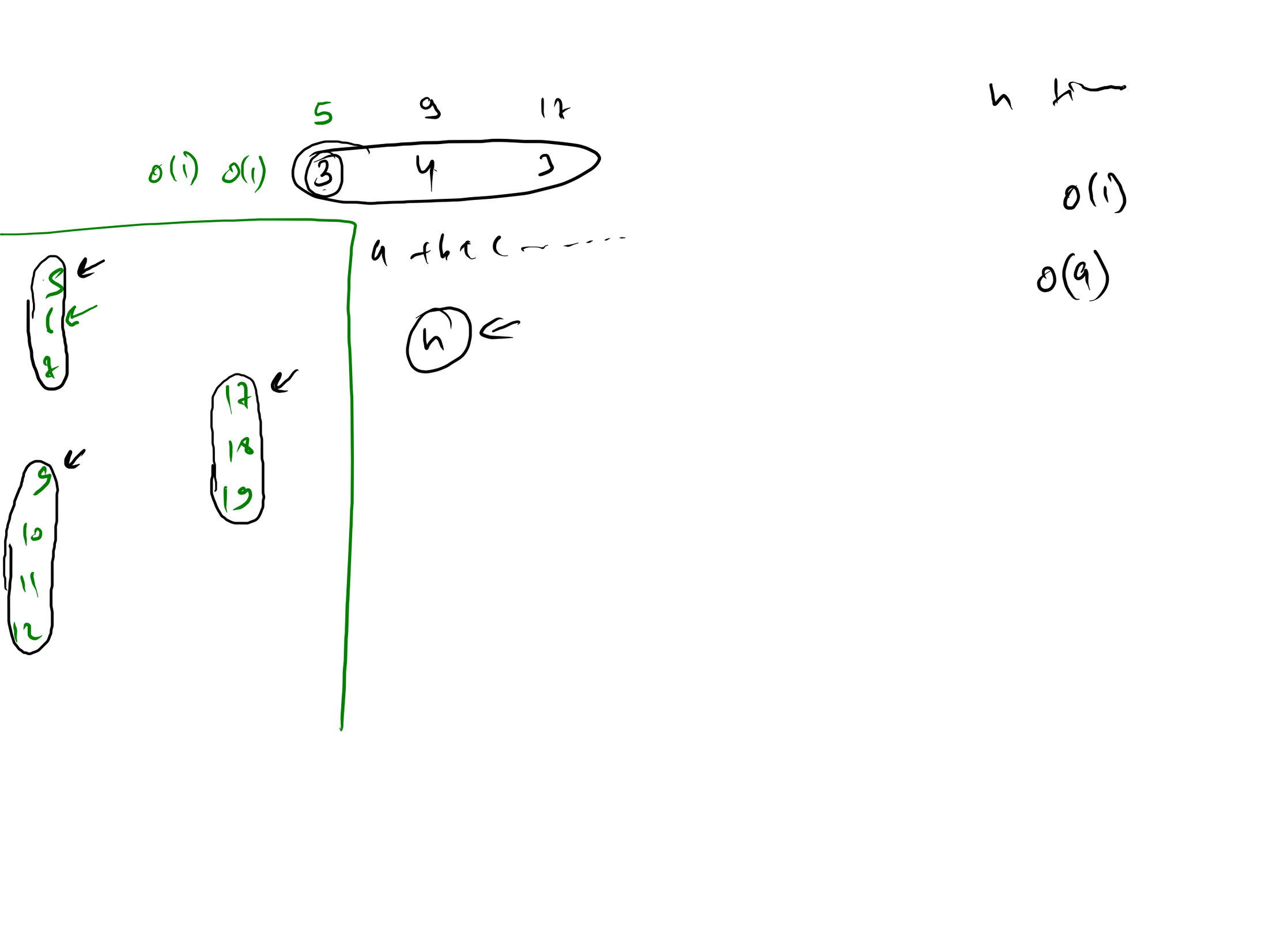
12 → 11 ✓

5 → 4

sum →	5	6	7	8	9	10	11	12	13	14
size	x	2	3	4	5	6	7	8	9	x

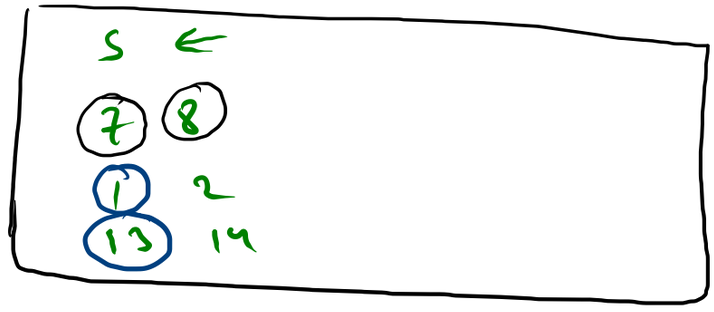
11	1	2	3
size	1	2	x

sum
size


5
9
17
O(1)
O(1)
3
4
3
O(1)
O(9)

```java
HashMap<Integer, Integer> hm = new HashMap<>();

for(int val: ar){
    hm.put(val, 1);
}

int size=0;
int start = 0;

for(Integer key: hm.keySet()){
    if(hm.containsKey(key-1))continue;
    int s = 1;
    int f = key+1;
    while(hm.containsKey(f)){
        s++;
        f++;
    }
    if(s > size || (size == s && key < start)){
        start = key;
        size = s;
    }
}

for(int i=0;i<size;i++){
    System.out.println(start);
    start++;
}
```

hm



size → ~~0~~ ~~1~~ ~~2~~ 2

start → ~~0~~ ~~5~~ ~~7~~ 1

7, 8

1, 2

key = 13

s = 2

f = 15

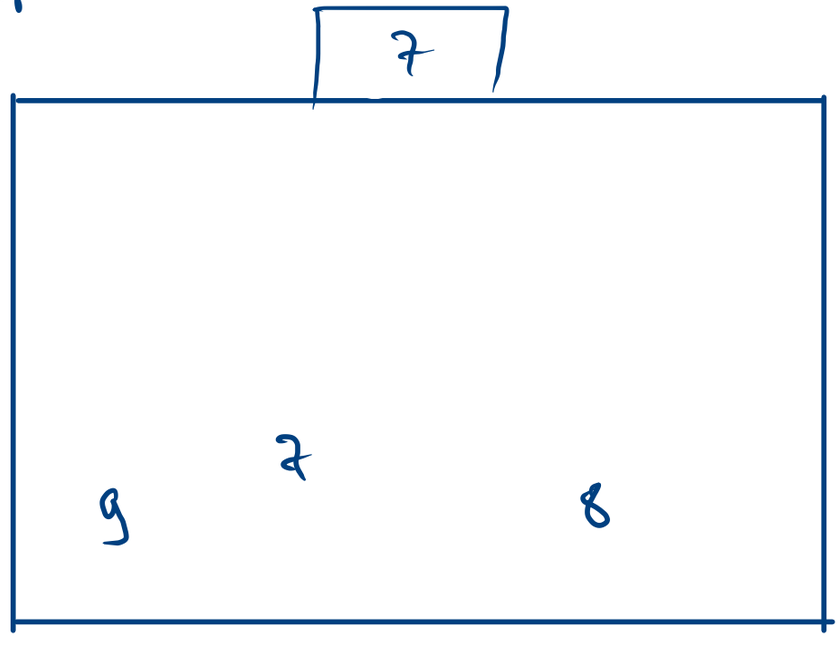
13, 14

[

1 > 0

PriorityQueue

Heap



add (val) → log(n)

remove → O(log n)

peek → O(1)

size

1 2 3 5

1 3 5

1 Rank

2

100 } score

90

< default

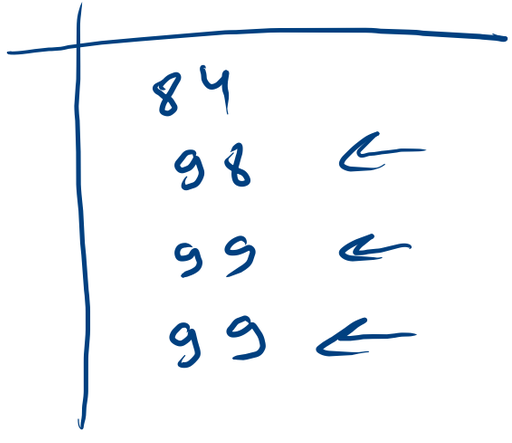
>

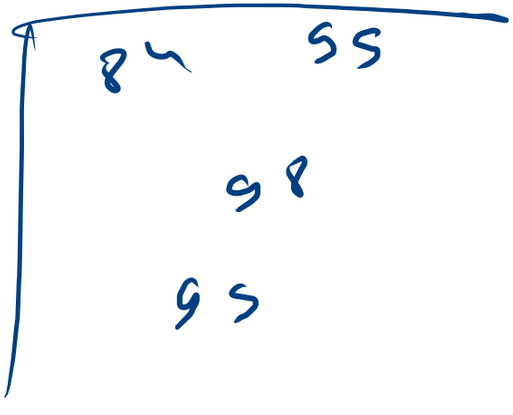
Collections.reverseOrder()

[12, 62, 22, 15, 37, 95, 11, 37, 98, 67, 31, 84]

95

k = 4

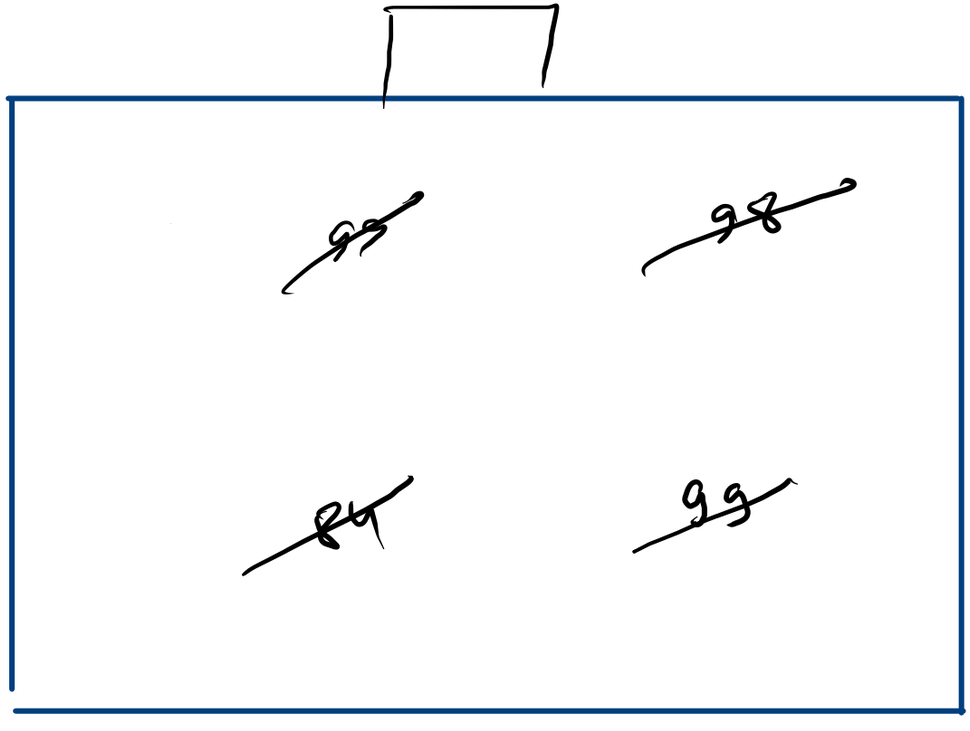




[12, 62, 22, 15, 37, 99, 11, 37, 98, 67, 31, 84]

99

k = 4



smaller

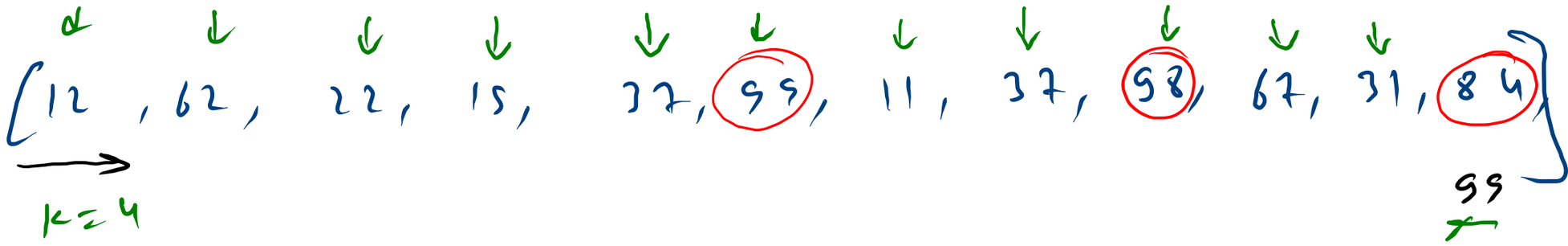
pq

84

98

99

99

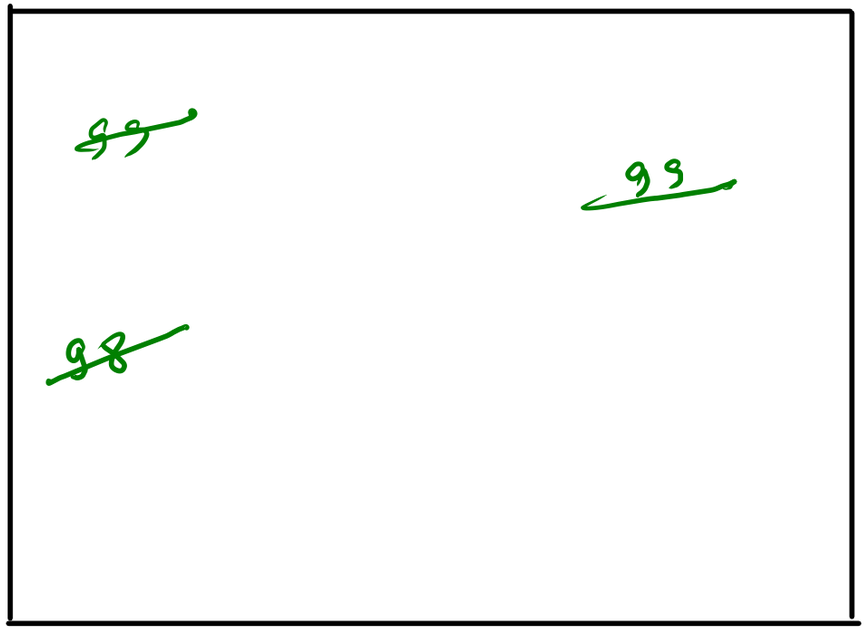



```java
PriorityQueue<Integer> pq = new PriorityQueue<>();

for(int val: arr){
    pq.add(val);
    if(pq.size() > k){
        pq.remove();
    }
}

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

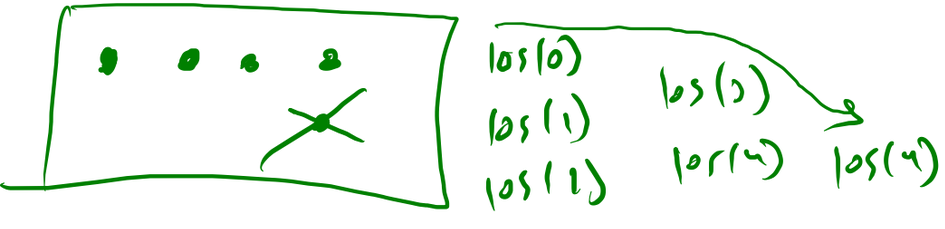
n log(k)

k log(k)

pq →

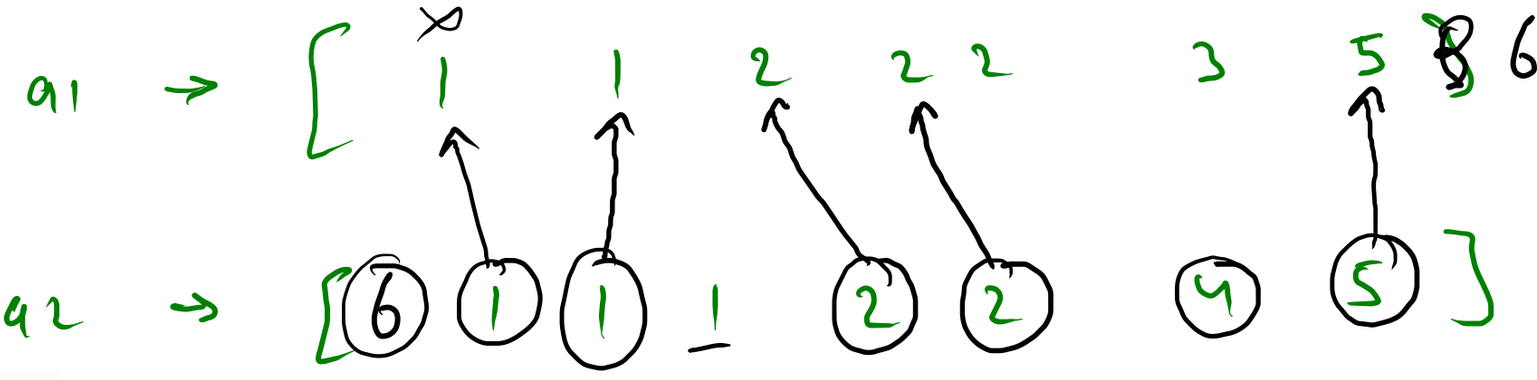


84
98
99
99



log(0)
log(1)
log(2)
log(3)
log(4)
log(4)

H.W



Frequency

```
if a1 -> 1 1 2 2 2 3 5
and a2 -> 1 1 1 2 2 4 5
intersection is -> 1 1 2 2 5
```

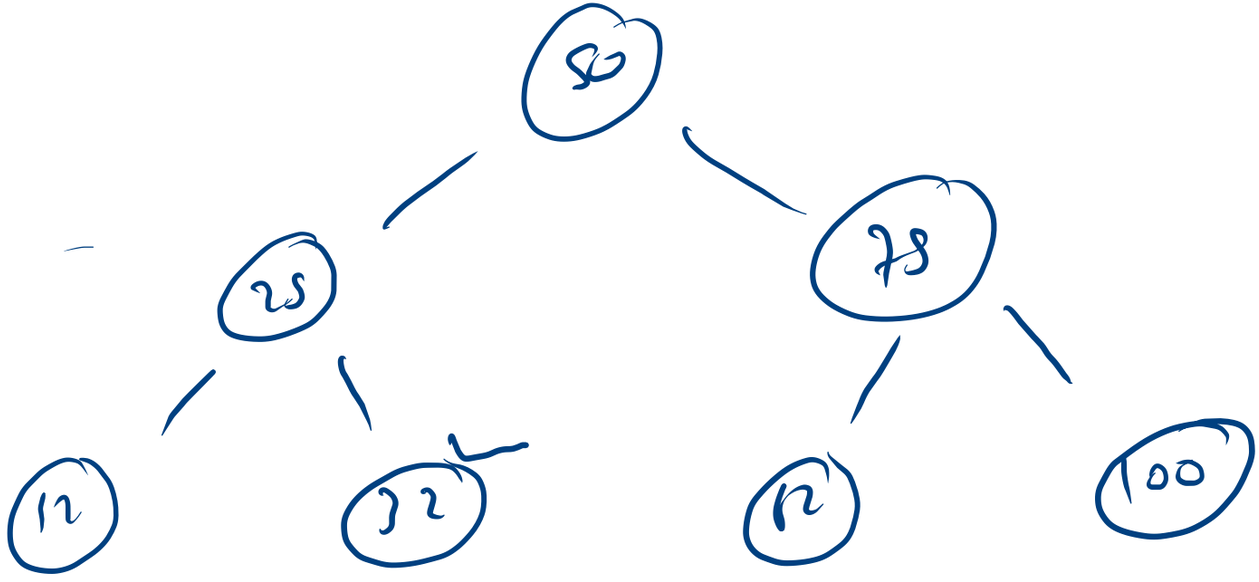
a2 → order

6
1
1
2
2
5

N.W.

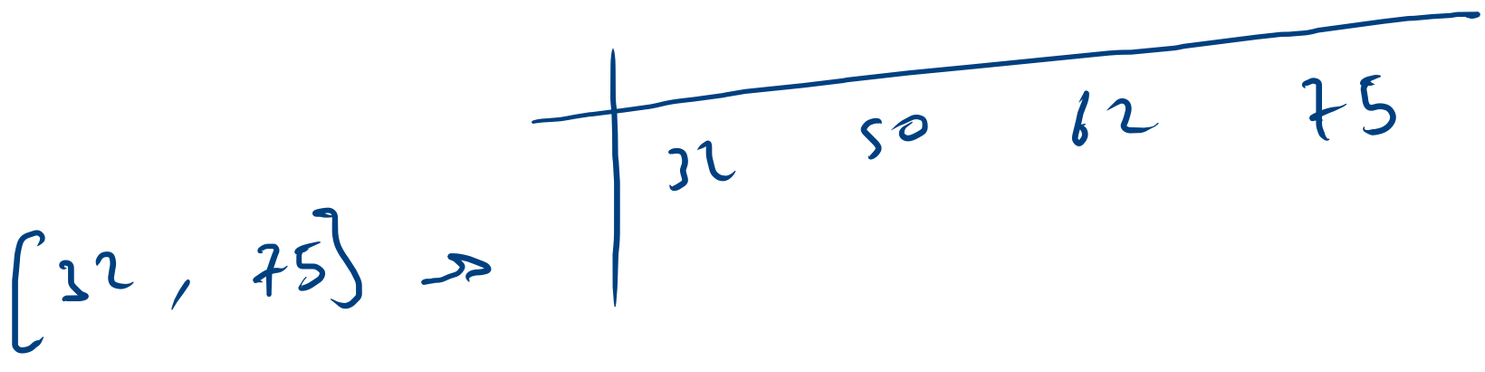
PIR

Ascending order

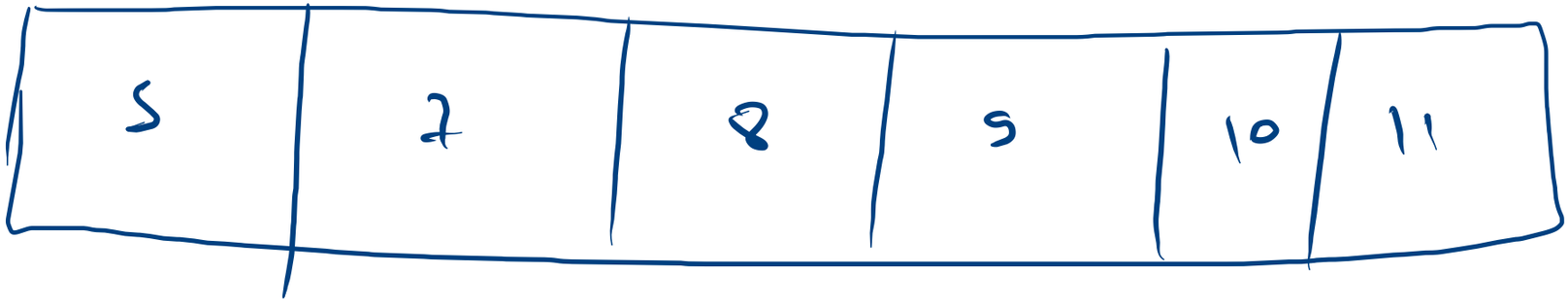


low => 32

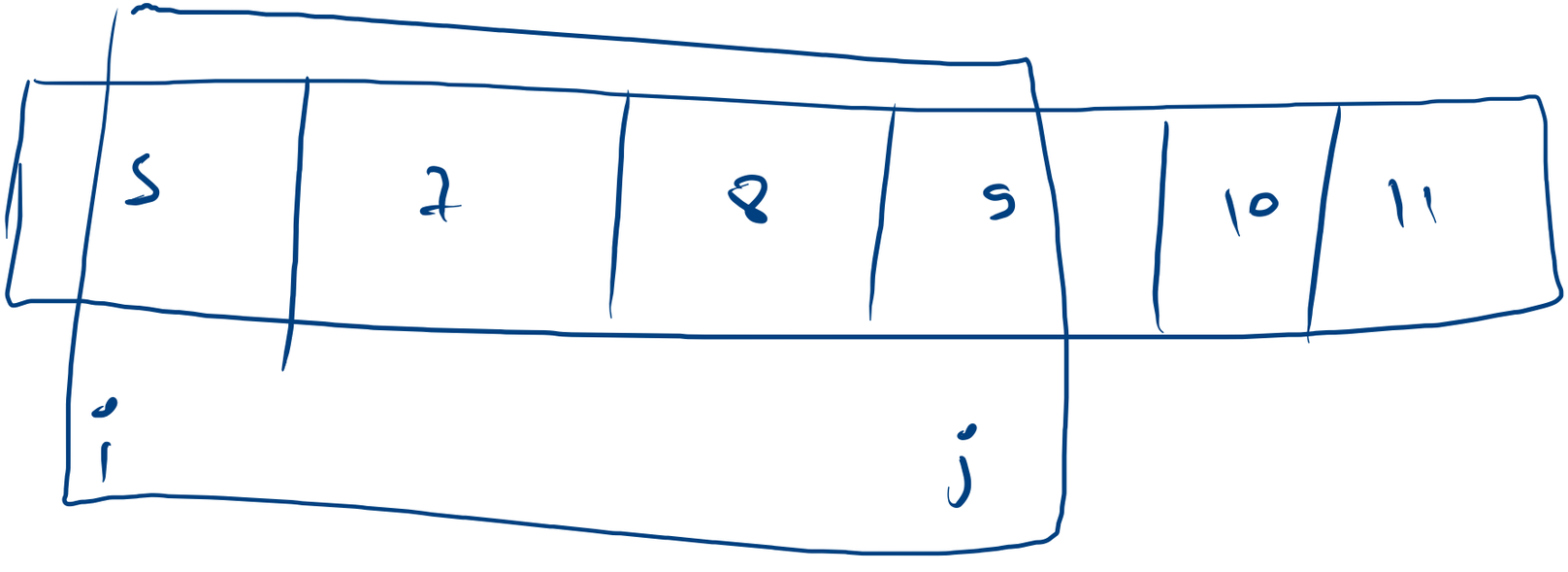
high -> 75



[32, 75] =>



h = 6



los(6)

los(4)

[a, b, c, d, e, f, g ...

n

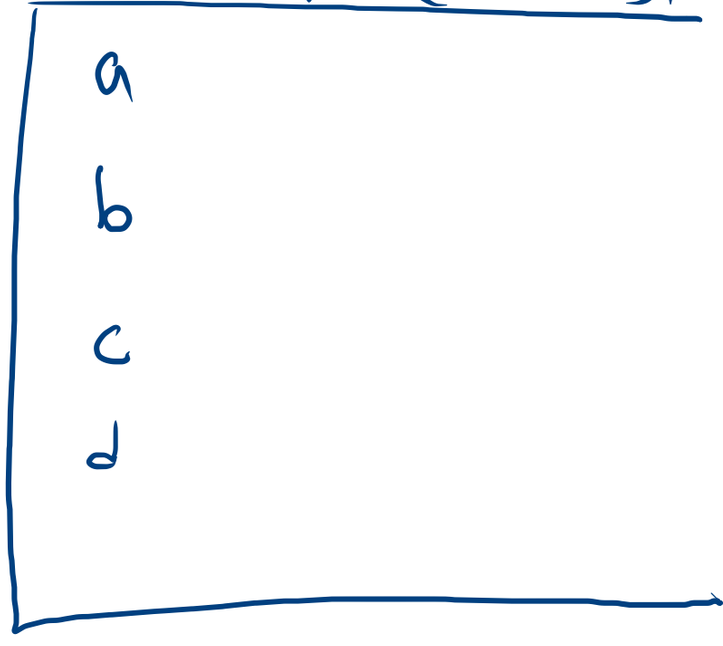
```
for( int val : arr) {
    pq.add(val);

    if( qq.size > k)
        pq.remove()
}
```

k = n

$\boxed{l(1) + l(2) \ldots\ldots} \; l(k) +$

$l(k) + l(k) + l(k) \ldots$

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

log(1)	a
log(2)	b
log(3)	c
log(4)	d
log(5)	
log(5)	
add log(5)	
rem log(5)	
log(5)	
log(5)	
⋮	

$\approx \log(k)$