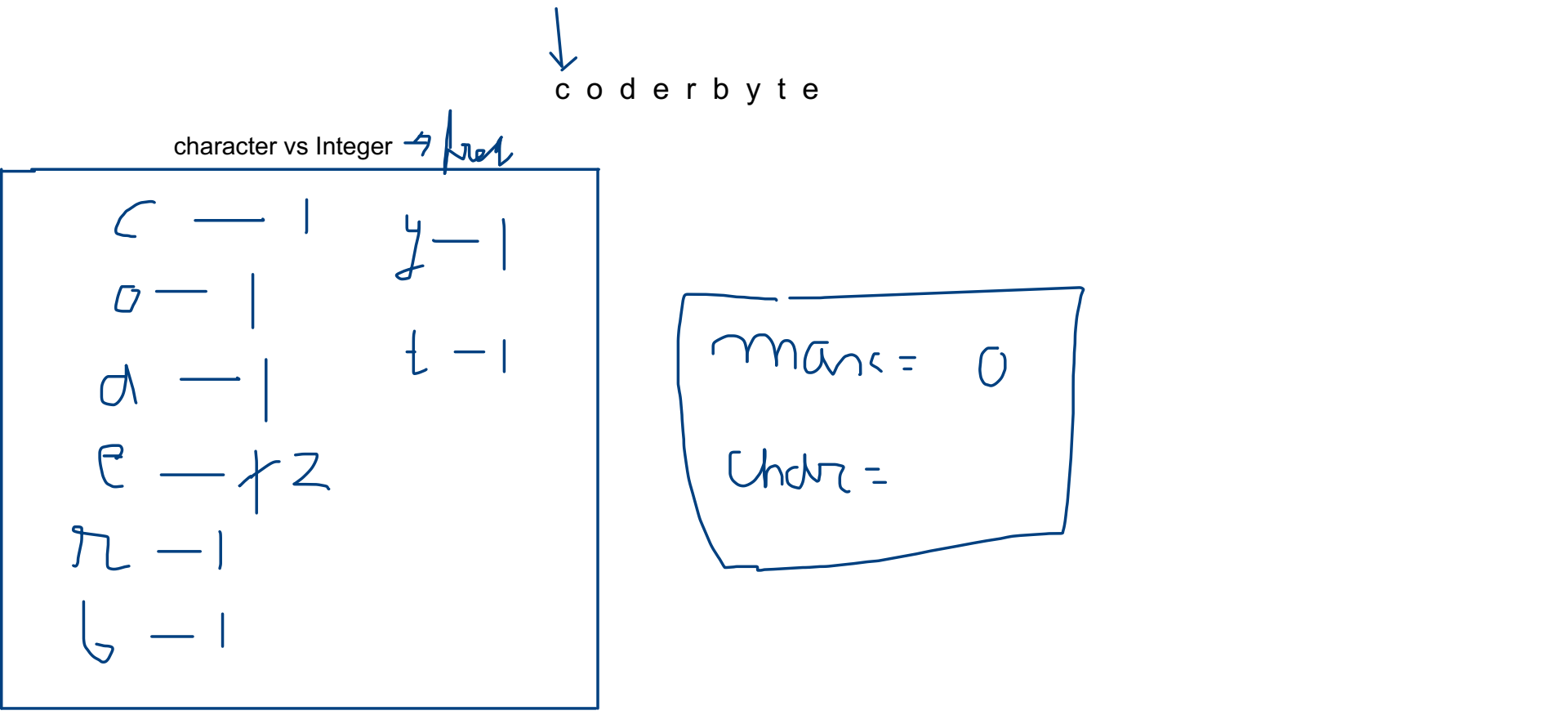
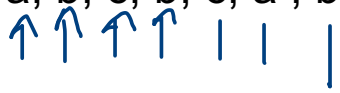
Highest Frequency Character



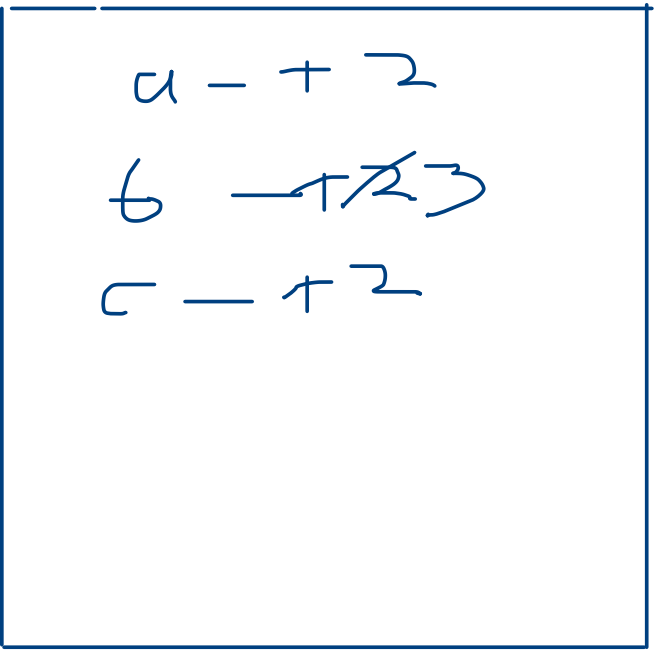
a, b, c, b, c, a , b



```
for (char ch: str.toCharArray()) {
    if (fmap.containsKey(ch)) {
        int oldFreq = fmap.get(ch) + 1;
        fmap.put(ch, oldFreq);
    } else {
        fmap.put(ch, 1);
    }

    if (maxFreq < fmap.get(ch)) {
        maxFreq = fmap.get(ch);
        maxFreqChar = ch;
    }
}
```

1
2
3



maxFreq = 0 → 3

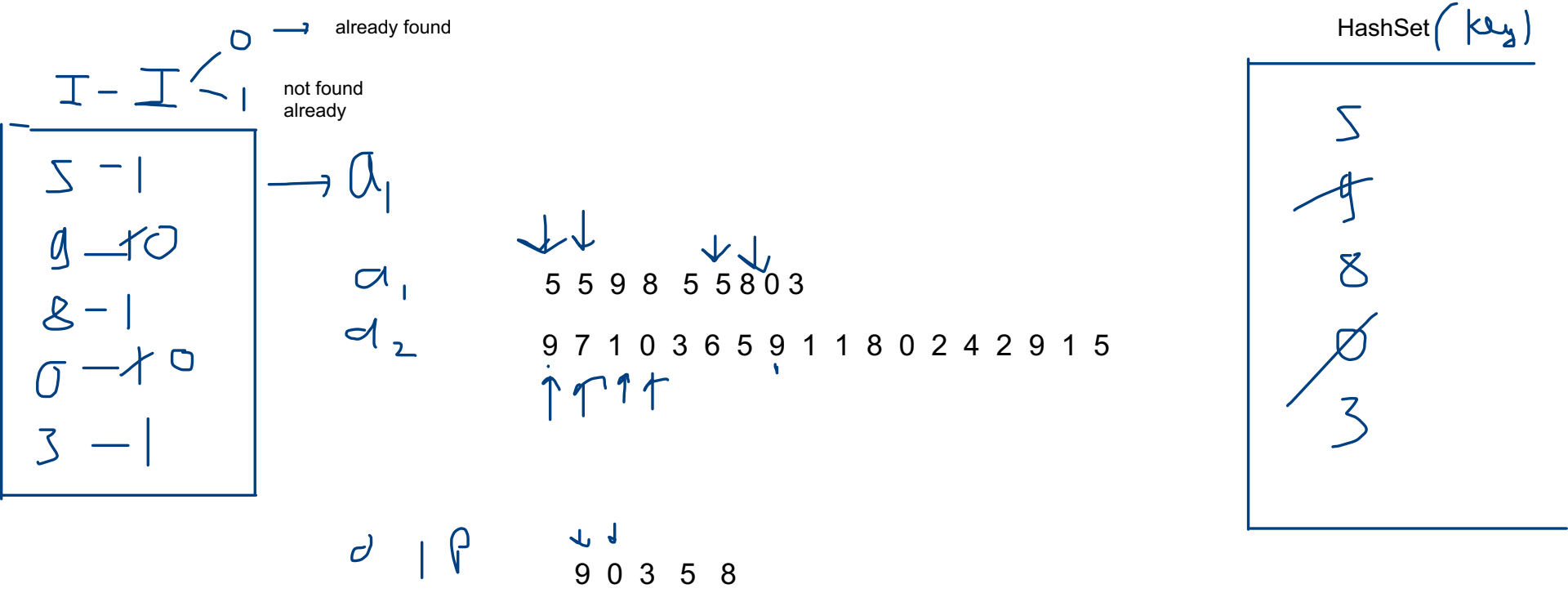
maxFreqChar = '\n' → a → b

Get Common Elements 1

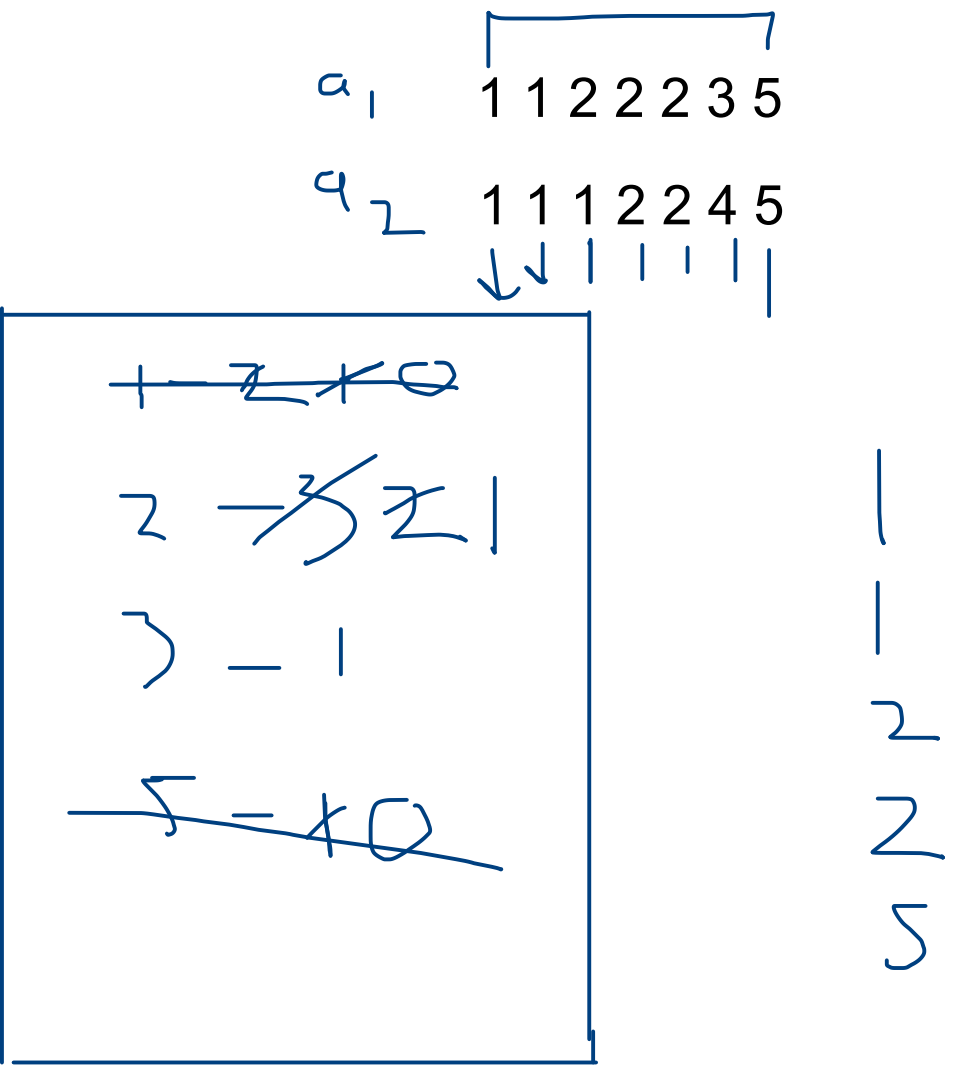
0 → already found

1 → not found already

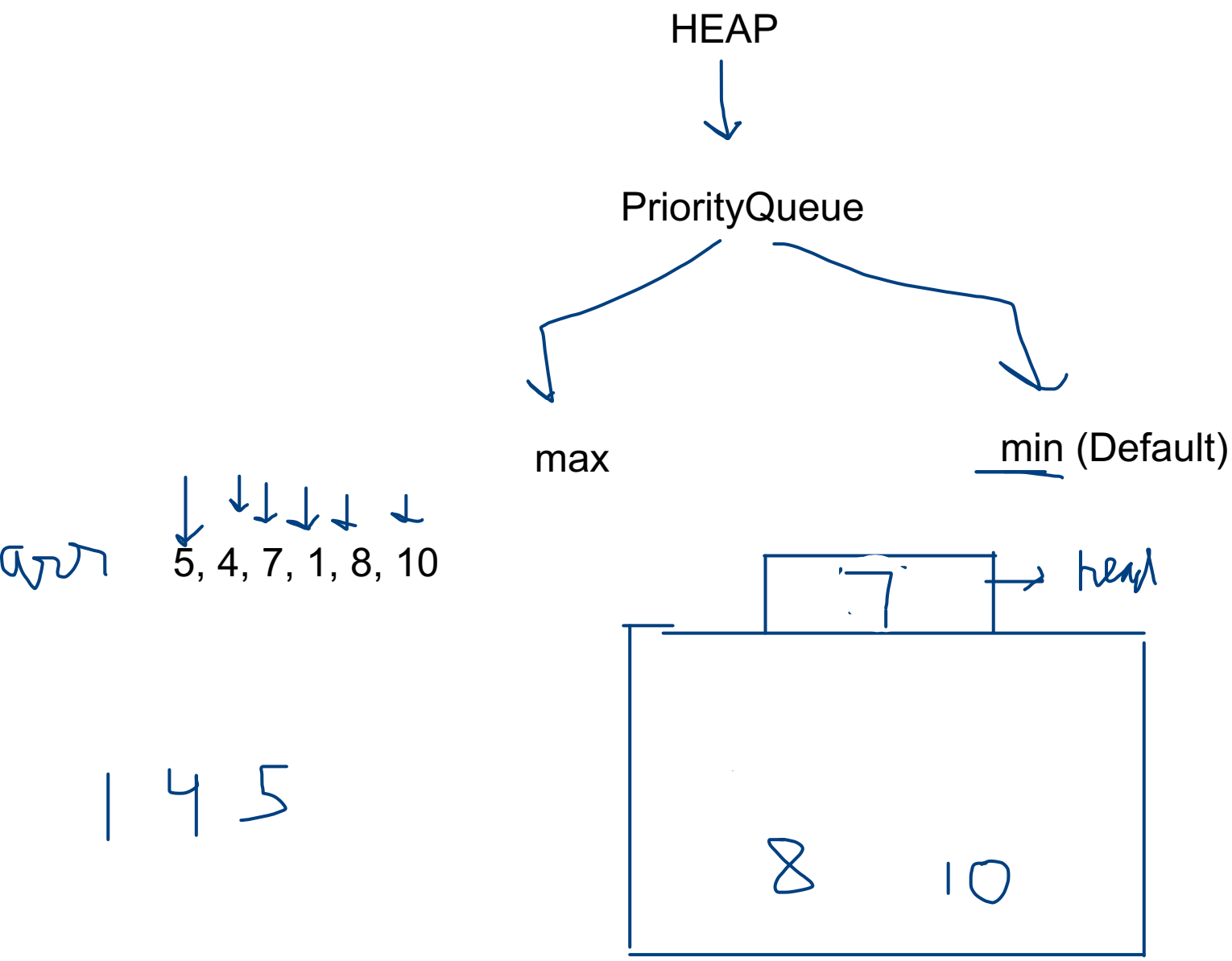
HashSet (key)



Get Common Elements - 2 (



1
1
2
2
5


HEAP
PriorityQueue
max
min (Default)
5, 4, 7, 1, 8, 10
arr
1 4 5
7
heap
8
10

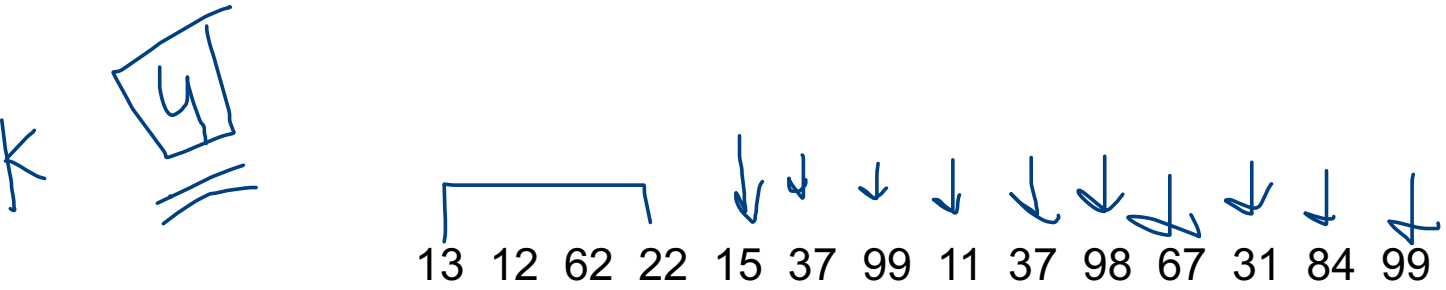
K Largest Elements

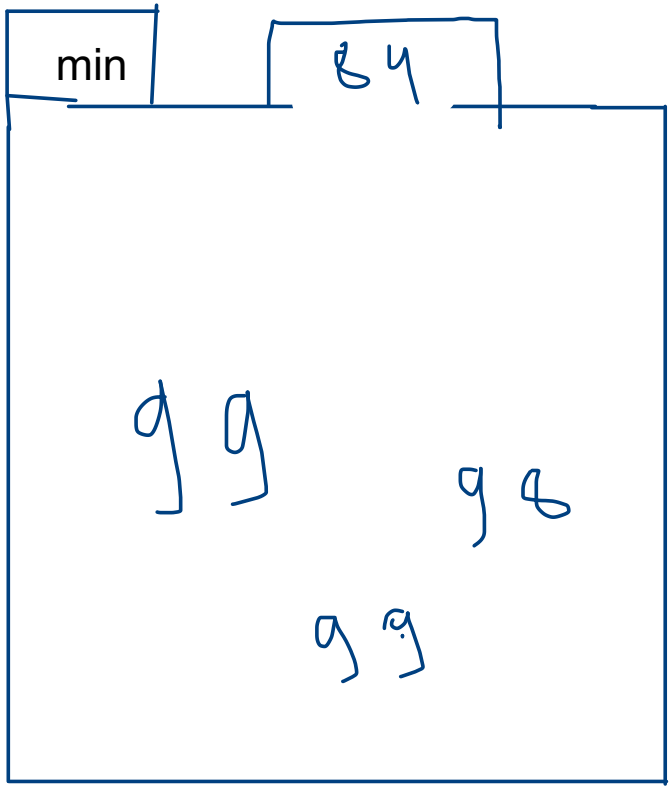
13 12 62 22 15 37 99 11 37 98 67 31 84 99

Sort

last K elements

Time = nlogn
Space= constant





37 > 13

99 > 15

11 < 22 → ignore

if (curr_val > pq.peek()) --> remove pq.peek and add curr_val

in single timeframe, we have only k elements in PQ

time == nlogk
space == k