

# Highest Frequency Character



c o d e r b y t e

character vs Integer

→ ~~for~~

c — 1	y — 1
o — 1	t — 1
d — 1	
e — 2	
r — 1	
b — 1	

max = 0

char =

a, b, c, b, c, a, b  
 ↑ ↑ ↑ ↑ | | |

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

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

maxFreq = 0 ~~1~~ ~~2~~ 3

maxFreqChar = '\n'

a  
~~b~~  
 b

a - 1 2  
 b - 1 2 3  
 c - 1 2

# Get Common Elements 1

$I - I < 0 \rightarrow$  already found  
 $I - I < 1$  not found already  
 $\rightarrow a_1$   
 $a_2$   
 $d \mid p$

$5 - 1$   
 $9 - 10$   
 $8 - 1$   
 $0 - 10$   
 $3 - 1$

$\downarrow \downarrow$   
 $5 \ 5 \ 9 \ 8 \ 5 \ 5 \ 8 \ 0 \ 3$   
 $\downarrow \downarrow$   
 $9 \ 7 \ 1 \ 0 \ 3 \ 6 \ 5 \ 9 \ 1 \ 1 \ 8 \ 0 \ 2 \ 4 \ 2 \ 9 \ 1 \ 5$   
 $\uparrow \uparrow \uparrow \uparrow$   
 $\downarrow \downarrow$   
 $9 \ 0 \ 3 \ 5 \ 8$

HashSet(key)

$5$   
 ~~$9$~~   
 $8$   
 ~~$0$~~   
 $3$

## Get Common Elements - 2 (

$a_1$       
1 1 2 2 2 3 5

$a_2$     1 1 1 2 2 4 5  
      ↓ ↓ | | | | |

~~1 - 2 - 3 - 5~~

2 - 3 - 2 - 1

3 - 1

~~5 - 4 - 0~~

1  
1  
2  
2  
5

HEAP



PriorityQueue

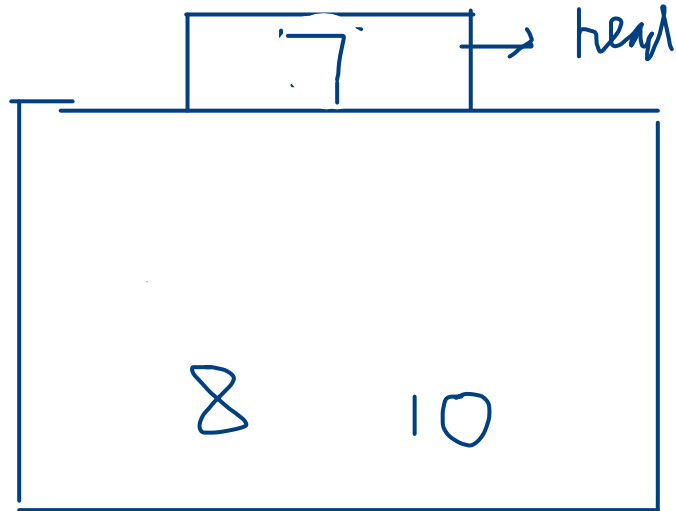


max

min (Default)

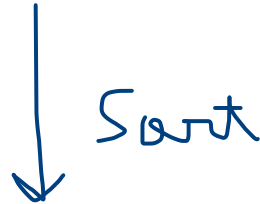
arr  
↓ ↓ ↓ ↓ ↓ ↓  
5, 4, 7, 1, 8, 10

1 4 5



## K Largest Elements

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

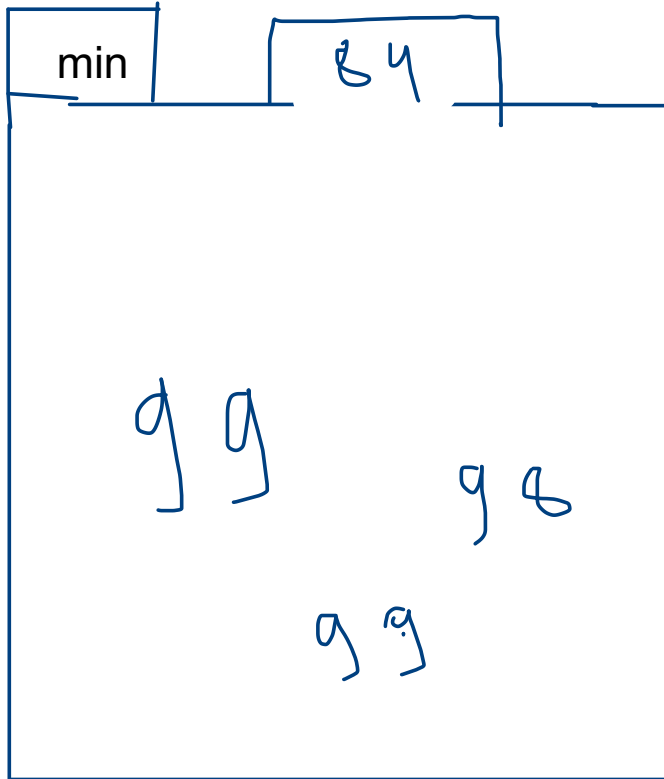
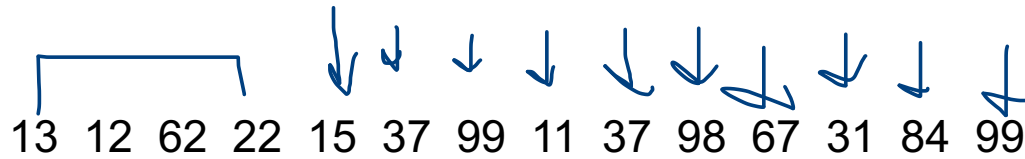
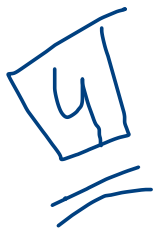


last  $K$  elements

Time =  $n \log n$

Space = constant

K



37 > 13

99 > 15

11 < 22 → ignore

if (curr\_val > pq.peak()) --> remove pq.peak and add curr\_val

in single timeframe, we have only k elements in PQ

time ==  $n \log k$

space == k