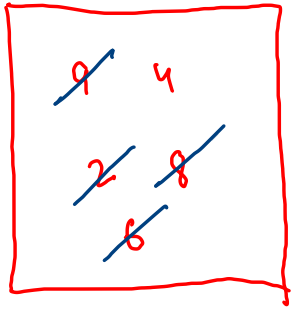
nCr-3

a1 : 9 4 2 8 4 2 9 6

a2 : 5 9 6 7 2 9 8

common : 9 6 2 8

elements



HM -> key vs value

HS -> keys

Hashset :

(i) unordered

(ii) key

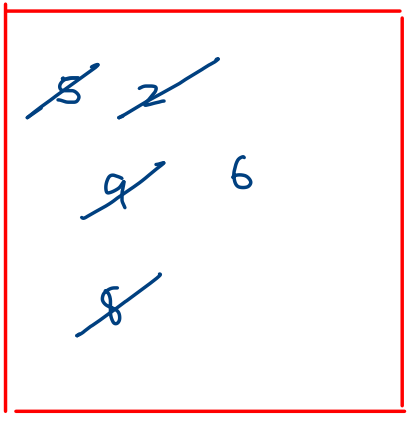
(iii) store unique elements

```java
public static void getCommonElements1(int[]a1,int[]a2) {
    HashSet<Integer>hs = new HashSet<>();

    for(int val : a1) {
        if(hs.contains(val) == false) {
            hs.add(val);
        }
    }

    for(int val : a2) {
        if(hs.contains(val) == true){
            System.out.println(val);
            hs.remove(val);
        }
    }
}
```

a1 : 5 2 9 5 9 2 6 8

a2 : 8 9 2 4 9 5

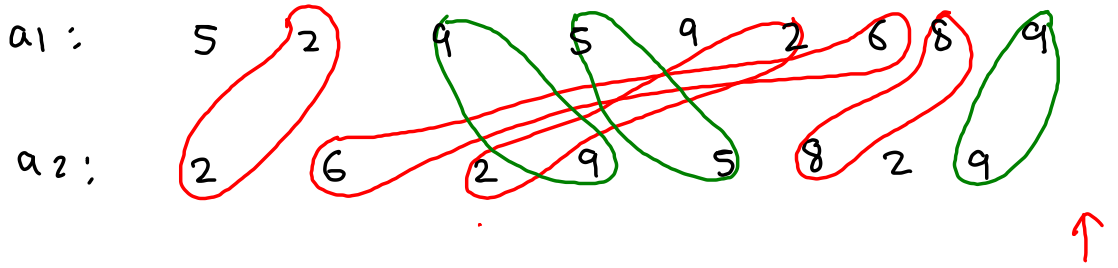
↑



hs (fill using a1)

8 9 2 5

gce 2



2 6 2 9 5 8 9

5 -> ~~2~~ 1

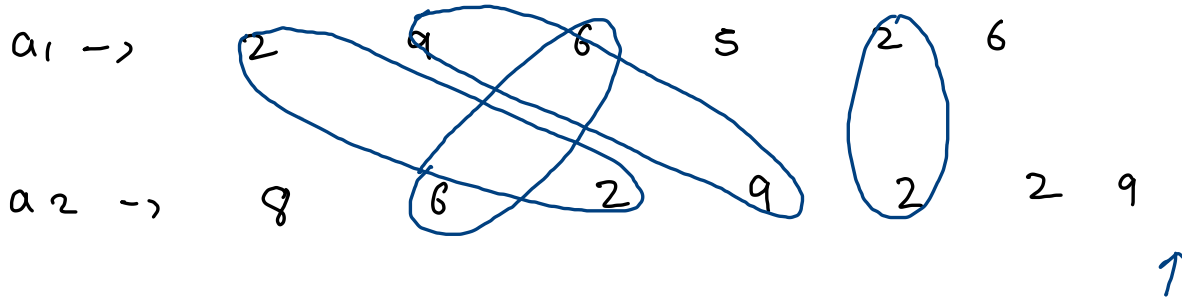
2 -> ~~2 2~~ 0

9 -> ~~3~~ 2

6 -> ~~1~~ 0

8 -> ~~1~~ 0

```java
public static void getCommonElements2(int[]a1,int[]a2) {
    HashMap<Integer,Integer>map = new HashMap<>();

    //fill this map using 1st array
    for(int ele : a1) {
        if(map.containsKey(ele) == false) {
            map.put(ele,1);
        }
        else {
            int nf = map.get(ele) + 1;
            map.put(ele,nf);
        }
    }


    //travel second and print your ans
    for(int ele : a2) {
        if(map.containsKey(ele) == true && map.get(ele) > 0) {
            System.out.println(ele);
            int nf = map.get(ele) - 1;
            map.put(ele,nf);
        }
    }
}
```



longest consecutive seq. of elements

19 14 5 2 4 15 16 13 1 7 20 9 3 8

1 2 3 4 5

7 8 9

13 14 15 16

19 20

19 14 5 2 4 15 16 13 1 7 20 9 3 8

19 -> T ✔
14 -> F
5 -> F
2 -> F
4 -> F
15 -> F

16 -> F
13 -> T ✔
1 -> T ✔
7 -> T ✔
20 -> F
9 -> F

3 -> F
8 -> F

Integer, boolean
↳ is this key a seq. start

19 14 5 2 4 15 16 13 1 7 20 9 3 8

19 -> T

14 -> F

5 -> F

2 -> F

4 -> F

15 -> F

16 -> F

13 -> T

1 -> T

7 -> T

20 -> F

9 -> F

3 -> F

8 -> F

len =

1 2 3 4 5

5

4

2

mlen = 0

st = 1

13

1

19 14 5 2 4 15 16 13 1 7 20 9 3 8 17 1

```java
//assume each element can be a seq start
for(int i=0; i < arr.length;i++) {
    if(map.containsKey(arr[i]) == false) {
        map.put(arr[i],true);
    }
}

//determine actual seq start
for(int key : map.keySet()) {
    int ele = key;

    if(map.containsKey(ele-1) == true) {
        map.put(ele,false);
    }
}
```

```java
//find the longest consecutive seq out of all consective seq
int st = -1;
int mlen = 0;

for(int key : map.keySet()) {
    int ele = key;
    int len = 1;

    if(map.get(ele) == true) {
        while(map.containsKey(ele + len) == true) {
            len++;
        }

        if(len > mlen) {
            mlen = len;
            st = ele;
        }
    }
}


for(int k = st; k < st + mlen;k++) {
    System.out.println(k);
}
```

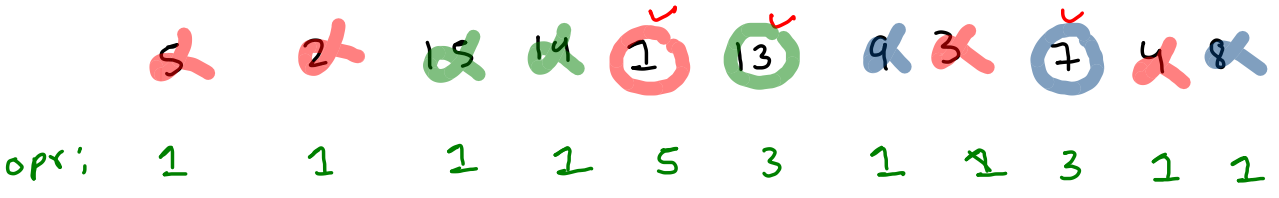
Arr

19 -> T F
14 -> ~~T~~ F
5 -> ~~T~~ F
2 -> ~~T~~ F
4 -> ~~T~~ F
15 -> ~~T~~ F
16 -> ~~T~~ F

13 -> T
4 -> T
7 -> T
20 -> ~~T~~ F
9 -> ~~T~~ F
3 -> ~~T~~ F
8 -> ~~T~~ F
17 -> ~~T~~ F

st = ~~-1~~ ~~19~~ ~~13~~ 1
mlen = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4
5

```java
for(int i=0; i < arr.length;i++) {
    int ele = arr[i];
    int len = 1;

    if(map.get(ele) == true) {
        while(map.containsKey(ele + len) == true) {

            len++;
        }

        if(len > mlen) {
            mlen = len;
            st = ele;
        }

    }

}
```

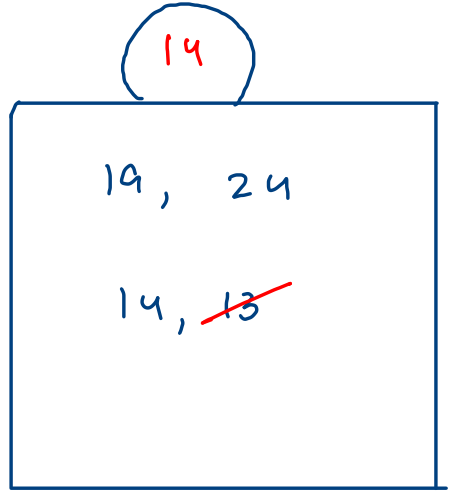


Priority queue

min (PQ) —> smaller the value higher the priority

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

```
pq.add(19);
pq.add(24);
pq.add(14);
pq.add(13);
pq.peek() —> 13
pq.remove()
pq.peek() -> 14
```



add, remove

logn

peek -> O(1)

arr : 9 8 7 13 6 12 18 25 3 16

```java
public static void solve(int[]arr,int k) {
    PriorityQueue<Integer>pq = new PriorityQueue<>(Collections.reverseOrder());

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

    int[]ans = new int[k];
    int idx = k-1;

    while(idx >= 0) {
        ans[idx] = pq.remove();
        idx--;
    }


    for(int i=0; i < ans.length;i++) {
        System.out.println(ans[i]);
    }
}
```

nlogn

n >> k

klogn

k

S: O(n)

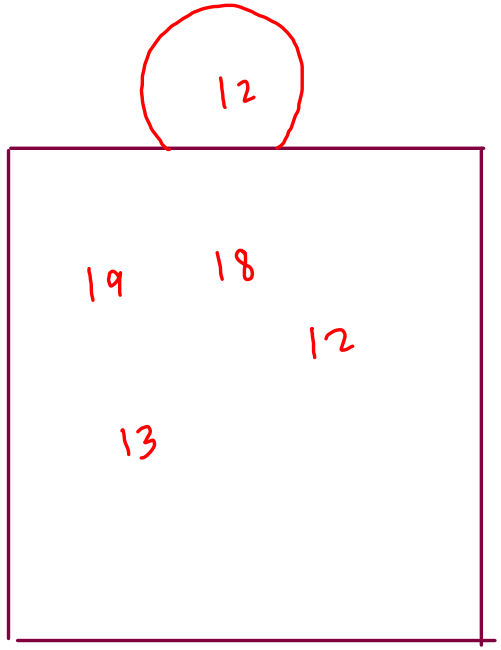
T: nlogn

add, remove

log n

peek -> O(1)

arr: 9 8 7 13 6 12 18 3 5 19



k = 4

k - dagest elments

12 13 18 19

min pq

```java
public static void solve(int[]arr,int k) {

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

    for(int i=0; i < arr.length;i++) {
        if(pq.size() < k) {
            pq.add(arr[i]);
        }
        else if(pq.peek() < arr[i]){
            pq.remove();
            pq.add(arr[i]);
        }
    }

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

$n * \log k$

$k * \log k$

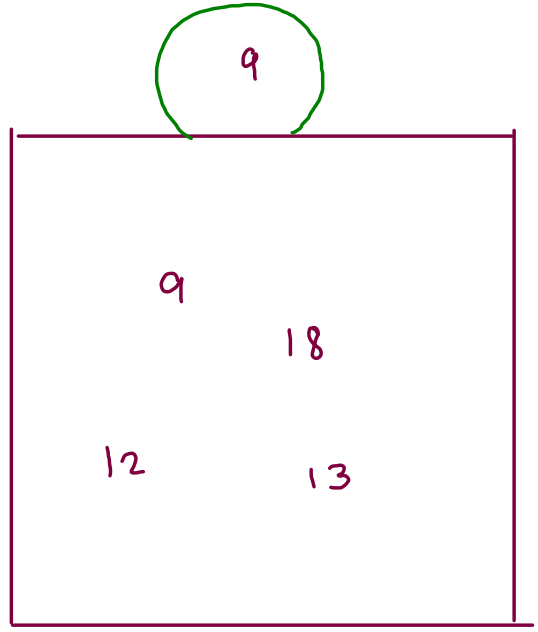
$\log k$

T : $n \log k$

S : k

arr: 9 8 7 13 6 12 18 3 5

```
public static void solve(int[]arr,int k) {

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

    for(int i=0; i < arr.length;i++) {
        if(pq.size() < k) {
            pq.add(arr[i]);
        }
        else if(pq.peek() < arr[i]){
            pq.remove();
            pq.add(arr[i]);
        }
    }

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



k = 4

minPQ

9 12 13 18