

Map =>
└─> interface

① HashMap
② TreeMap
③ LinkedHashMap

Set
└─> interface

key-value
├─> map
└─> map

index ~~index~~

hashmap X

o.c.)

Name	marks
Raj	87
Ankit	89
Ankita	77
Kunal	65
Kamlesh	87
Annu	55

Raj, 87	Ankit, 89	Ankita, 77	100, 65
---------	-----------	------------	---------

Set -> interface
└─> HashSet
└─> TreeSet
└─> LinkedHashSet

Given two integer arrays `nums1` and `nums2`, return an array of their intersection. Each element in the result must appear as many times as it shows in both arrays and you may return the result in **any order**.

Example 1:

Input: `nums1 = [1,2,2,1]`, `nums2 = [2,2]`
Output: `[2,2]`

[2, 1, 3 3, 4 > 2]

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
[2, 3, 1, 4, 5, 3, 7, 1, 2, 4, 5]
[2, 1, 3, 3, 4, 3, 7, 2, 2, 7, 1, 1, 5, 3]
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

key	value
2	2
3	2
1	1
4	1
5	2
7	1

14 → 30
20 → 102
7
7002 × 20
14000
8000L
2000L
6000

```
List<Integer> ll = new ArrayList<>();  
for (int i = 0; i < nums2.length; i++) {  
    if (map.containsKey(nums2[i]) && map.get(nums2[i]) > 0) {  
        ll.add(nums2[i]);  
        map.put(nums2[i], map.get(nums2[i]) - 1);  
    }  
}
```

```
public static int[] Intersection(int[] nums1, int[] nums2) {  
    HashMap<Integer, Integer> map = new HashMap<>();  
    for (int i = 0; i < nums1.length; i++) {  
        if (!map.containsKey(nums1[i])) {  
            map.put(nums1[i], 1);  
        } else {  
            map.put(nums1[i], map.get(nums1[i]) + 1);  
        }  
    }  
    int[] res = new int[nums2.length];  
    int j = 0;  
    for (int i = 0; i < nums2.length; i++) {  
        if (map.containsKey(nums2[i]) && map.get(nums2[i]) > 0) {  
            res[j++] = nums2[i];  
            map.put(nums2[i], map.get(nums2[i]) - 1);  
        }  
    }  
    return res;  
}
```

[2, 3, 2, 3, 1]
↑ ↑ ↑ ↑ ↑

2	2
3	2
1	1