*Rich people have small TVs and big libraries, and poor people have small libraries and big TVs.*

# Searching Technique

# HASHING

HASHING

PROBLEMS

NEED OF HASH

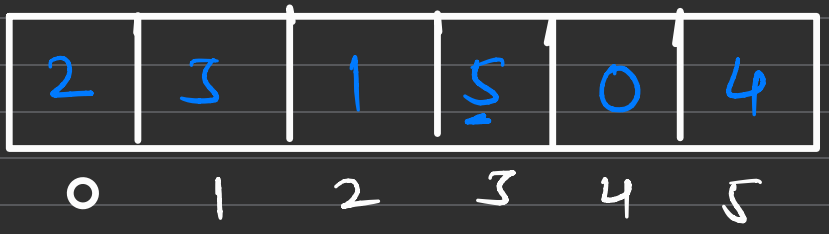
TRICKS

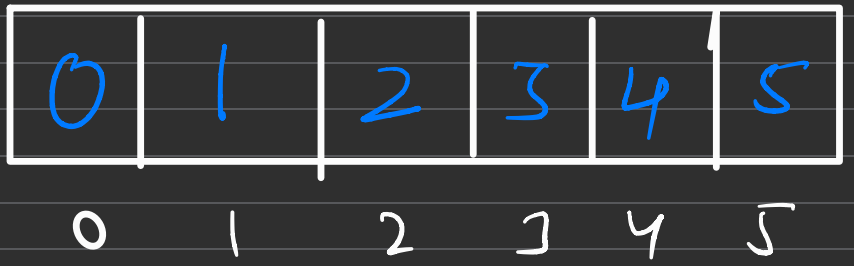
# Why we need Hashing

O(N)

O(1)

insert
delete
update




① x=0

② x=3

③ delete 5

$10014$ $10018$ $100000$ $1000111_{23}$

# What is Hashing

- Hashing is a technique that efficiently stores and retrieves data in a way that allows for quick access.
- We can perform the operations like, Insert, Delete, Update in O(1) Time
- In hashing, large number is converted into smaller number, and that smaller number is used as index to store the large number.
- This can be done, using the hash function

$$h(x) = x \bmod N$$

X : Key which need to inserted into the table.

N: Size of the table.

Keys: 3, 9, 14, 25, 36, 2, 1, 47, 18, 20

$h(x) = x \% 10$

$3 \% 10 = 3$

$9 \% 10 = 9$

$1A \% 10 = A$

$25 \% 10 = 5$

$3C \% 10 = 6$

$2 \% 10 = 2$

$1 \% 10 = 1$

$47 \% 10 = 7$

$18 \% 10 = 8$

$20 \% 10 = 0$

Keys: 3, 9, 14, 25, 36, 2, 74, 46, 18, 20

$h(x) = x \% 10$

$3 \% 10 = 3$

$9 \% 10 = 9$

$14 \% 10 = 4$

$25 \% 10 = 5$

$36 \% 10 = 6$

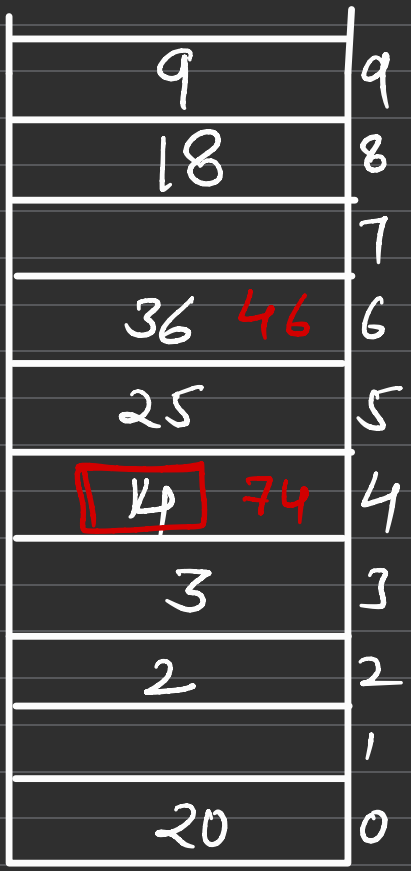
$2 \% 10 = 2$

$74 \% 10 = 4$

$46 \% 10 = 6$

$18 \% 10 = 8$

$20 \% 10 = 0$



## Collisions Techniques

When multiple no. shares same index

1. Separate chaining
2. Open Addressing
   1. linear probing
   2. Quadratic prob
   3. **Double Hashing**

# ( Separate Chaining )

Keys: 3, 9, 14, 25, 36, 2, 74, 46, 18, 20

$h(x) = x \% 10$

$3 \% 10 = 3$

$9 \% 10 = 9$

$14 \% 10 = 4$

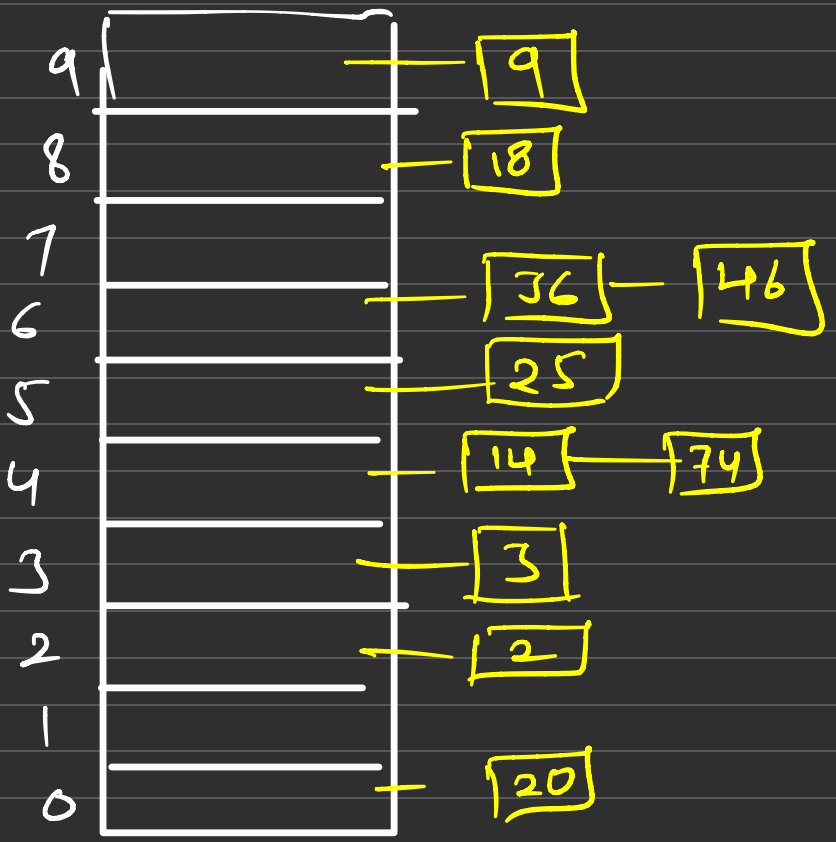
$25 \% 10 = 5$

$36 \% 10 = 6$

$2 \% 10 = 2$

$74 \% 10 = 4$

$46 \% 11 = 6$



# Linear probing

**Keys: 3, 9, 14, 25, 36, 2, 74, 46, 18, 20**

$3 \% 10 = 3$

$9 \% 10 = 9$

$25 \% 10 = 5$

$36 \% 10 = 6$

$2 \% 10 = 2$

$74 \% 10 = 4$

$46 \% 10 = 6$

$18 \% 10 = 8$

$20 \% 10 = 0$

$$h(x) = (x+i) \% N$$

$i = 0, 1, \ldots\ldots N$

| | |
|---|---|
| 9 | 9 |
| 18 | 8 |
| 20 | 7 |
| 36 | 6 |
| 25 | 5 |
| 14 | 4 |
| 3 | 3 |
| 2 | 2 |
| 74 | 1 |
| 46 | 0 |

# Quadratic probing

**Keys: 3, 9, 14, 25, 36, 2, 74, 46, 18, 20**

$3 \% 10 = 3$

$9 \% 10 = 9$

$14 \% 10 = 4$

$25 \% 10 = 5$

$36 \% 10 = 6$

$2 \% 10 = 2$

$74 \% 10 = 4$

$46 \% 10 = 6$

$(46 + 0^2) \% 10 = 6$

$(46 + 1^2) \% 10 = 7$

$(18 + 0^2) \% 10 = 8$

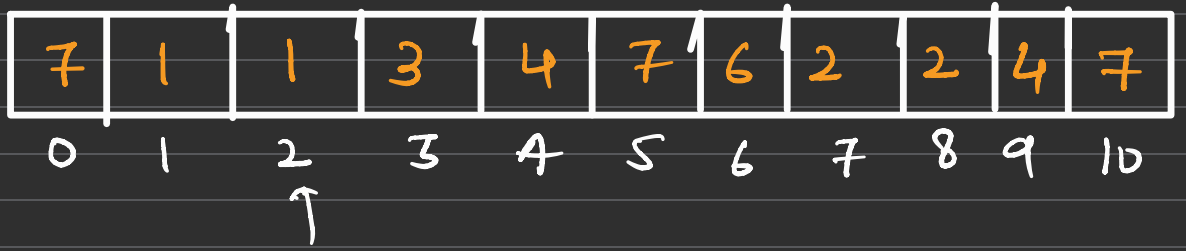
$(20 + 0^2) \% 10 = 0$

$(20 + 1^2) \% 10 = 1$

$$h(x) = (x + i^2) \% N$$

# Count Occurrence of elements

7 → 3
1 → 2
3 → 1
4 → 2
6 → 1




( No., Occurance)

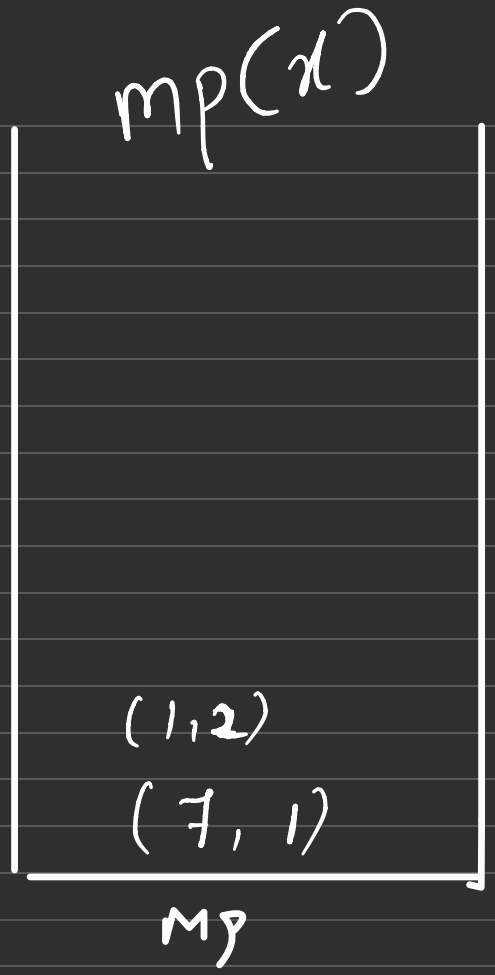
( Key, Value)

No.

Occurance

(2,2)
(6,1)
(4,2)
(3,1)
(1,2)
(7,3)

```
map <int, int> mp
for ( i=0; i<n ; i++)
{
    mp[a[i]]++
}
for (auto it:mp)
{
    cout<< it.first
    cout<< it.second
}
```



# 3005. Count Elements With Maximum Frequency

Solved

Easy | Topics | Companies | Hint | Re-do

You are given an array `nums` consisting of **positive** integers.

Return *the **total frequencies** of elements in* `nums` *such that those elements all have the **maximum** frequency*.

The **frequency** of an element is the number of occurrences of that element in the array.

**Example 1:**

```
Input: nums = [1,2,2,3,1,4]
Output: 4
Explanation: The elements 1 and 2 have a frequency of 2 which is the maximum frequency in the array.
So the number of elements in the array with maximum frequency is 4.
```

**Example 2:**

```
Input: nums = [1,2,3,4,5]
Output: 5
Explanation: All elements of the array have a frequency of 1 which is the maximum.
So the number of elements in the array with maximum frequency is 5.
```

Given an array **arr[]** of positive integers. Find the number of pairs of integers whose absolute difference equals to a given number **k**.
**Note:** (a, b) and (b, a) are considered the same. Also, the same numbers at different indices are considered different.

The answer is guaranteed to fit in a 32-bit integer.

$a - b = k$

**Examples:**

**Input:** arr[] = [1, 4, 1, 4, 5], k = 3
**Output:** 4
**Explanation:** There are 4 pairs with absolute difference 3, the pairs are {1, 4}, {1, 4}, {4, 1} and {1, 4}.

**Input:** arr[] = [8, 16, 12, 16, 4, 0], k = 4
**Output:** 5
**Explanation:** There are 5 pairs with absolute difference 4, the pairs are {8, 12}, {8, 4}, {16, 12}, {12, 16}, {4, 0}.

1 4 1 4 5 K=3

(1 4) (4 1)

(1 4) (4 1)

$$|a-b| = K$$

```
int cnt=0
for(i=0;i<n;i++)
{   for(j=i+1;j<n;j++)
    {   diff=abs(a[i]-a[j])
        if(diff==K)
            cnt++
    }
}
return cnt
```

$$a - b = k$$

$$\boxed{a = k + b}$$

// step 1: find occurrence

```
for (auto it: arr)
    mp[it]++
```

```
int ans = 0
for(i=0; i<n; i++)
{   a = k + a[i]
    if(mp.find(a) != mp.end())
        ans = ans + mp[a]

return ans
```

## Union of Arrays with Duplicates

Difficulty: **Easy** Accuracy: **42.22%** Submissions: **440K+** Points: **2** Average Time: **10m**

Given two arrays **a[]** and **b[]**, the task is to find the number of elements in the union between these two arrays.

The Union of the two arrays can be defined as the set containing distinct elements from both arrays. If there are repetitions, then only one element occurrence should be there in the union.

Note: Elements of **a**[] and **b**[] are not necessarily distinct.

**Examples**

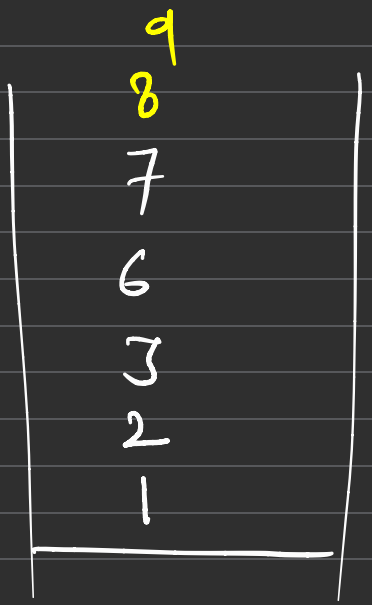
**Input:** a[] = [1, 2, 3, 4, 5], b[] = [1, 2, 3]
**Output:** 5
**Explanation:** Union set of both the arrays will be 1, 2, 3, 4 and 5. So count is 5.

**Input:** a[] = [85, 25, 1, 32, 54, 6], b[] = [85, 2]
**Output:** 7
**Explanation:** Union set of both the arrays will be 85, 25, 1, 32, 54, 6, and 2. So count is 7.

arr1 = { 1, 2, 1, 3, 6, 7 }

arr2 = { 7, 6, 8, 9 }

Set :




Set Size = 7

## Check Equal Arrays

Difficulty: **Easy** Accuracy: **42.18%** Submissions: **392K+** Points: **2** Average Time: **30m**

Given two arrays **a[]** and **b[]** of equal size, the task is to find whether the elements in the arrays are equal.

> Two arrays are said to be equal if both contain the same set of elements, arrangements (or permutations) of elements may be different though.

Note: If there are repetitions, then counts of repeated elements must also be the same for two arrays to be equal.

**Examples:**

**Input:** a[] = [1, 2, 5, 4, 0], b[] = [2, 4, 5, 0, 1]
**Output:** true
**Explanation:** Both the array can be rearranged to [0,1,2,4,5]

**Input:** a[] = [1, 2, 5], b[] = [2, 4, 15]
**Output:** false
**Explanation:** a[] and b[] have only one common value.

# 169. Majority Element

Solved

Easy | Topics | Companies | Re-do

Given an array `nums` of size `n`, return *the majority element*.

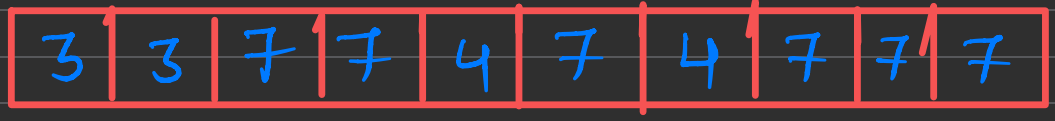
The majority element is the element that appears more than `⌊n / 2⌋` times. You may assume that the majority element always exists in the array.

**Example 1:**

```
Input: nums = [3,2,3]
Output: 3
```

**Example 2:**

```
Input: nums = [2,2,1,1,1,2,2]
Output: 2
```

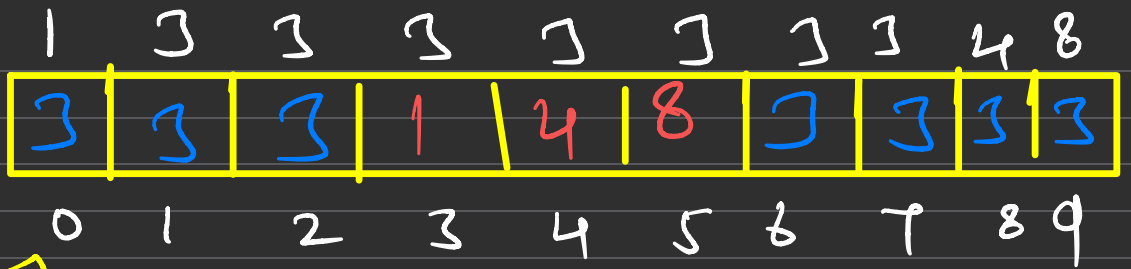



N=10

Brute force

```
for (i=0 ; i<n ;i++)
{    int cnt=1
     for (j=i+1;j<n;j++)
     {   if (a[i] == a[j])
             cnt++
     }
     if (cnt > n/2)
         return a[i]
}
```

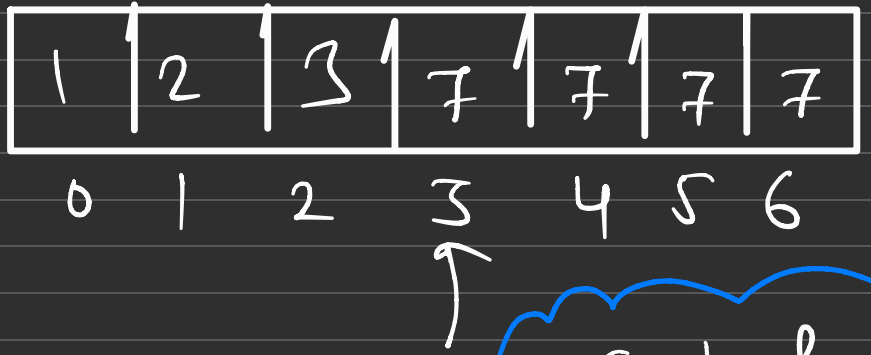
$O(N^2)$




Even Size

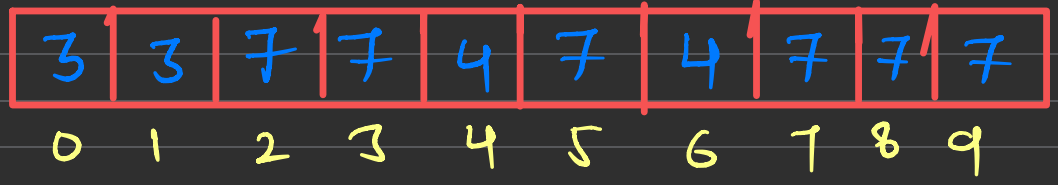
N.logN

$\frac{10}{2} = 5$




Odd Size

Sort & return mid Value

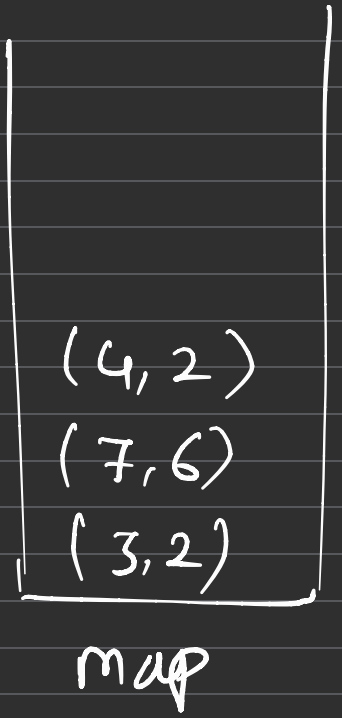



① Count Occurance

```
map <int, int> mp
for (auto it: arr)
    mp[it]++
```

O(N)

② Traverse the map, check which no. has occured more than n/2



O(N)

```
for (auto it: mp)
{
    int num = it.first
    int occ = it.second

    if (occ > n/2)
        return num
}
```

{ Key, Value }

{ Num, Occurrance }

O(N) + O(N) : TC

O(N) : SC

## Intersection of Two arrays with Duplicate Elements

Difficulty: **Easy** Accuracy: **61.4%** Submissions: **33K+** Points: **2** Average Time: **20m**

Given two integer arrays **a[]** and **b[]**, you have to find the **intersection** of the two arrays. Intersection of two arrays is said to be elements that are common in both arrays. The intersection should not have duplicate elements and the result should contain items in any order.

Note: The driver code will sort the resulting array in increasing order before printing.

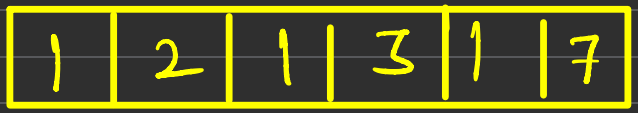
**Examples:**

**Input:** a[] = [1, 2, 1, 3, 1], b[] = [3, 1, 3, 4, 1]
**Output:** [1, 3]
**Explanation:** 1 and 3 are the only common elements and we need to print only one occurrence of common elements.

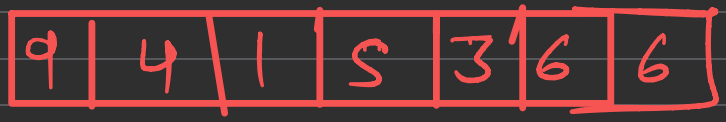
**Input:** a[] = [1, 1, 1], b[] = [1, 1, 1, 1, 1]
**Output:** [1]
**Explanation:** 1 is the only common element present in both the arrays.

**Input:** a[] = [1, 2, 3], b[] = [4, 5, 6]
**Output:** []
**Explanation:** No common element in both the arrays.

a1

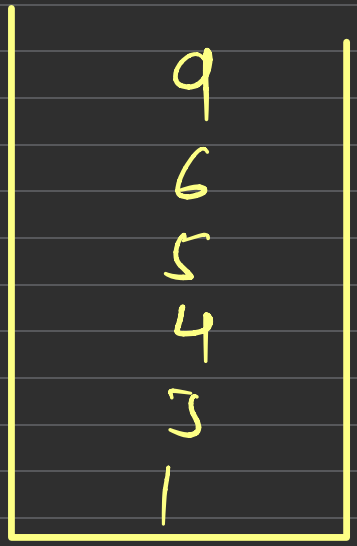



a2





ans



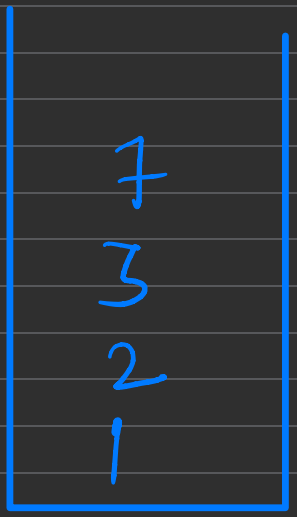

set

$a_1$

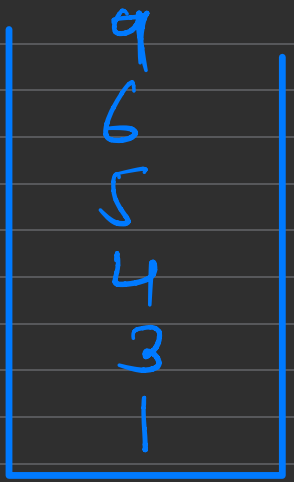
| 1 | 2 | 1 | 3 | 1 | 7 |
|---|---|---|---|---|---|

$a_2$

| 9 | 4 | 1 | 5 | 3 | 6 | 6 |
|---|---|---|---|---|---|---|

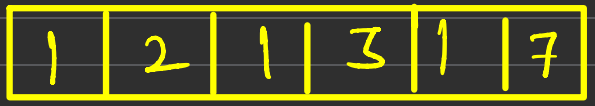



$Set_1$




$Set_2$

a1

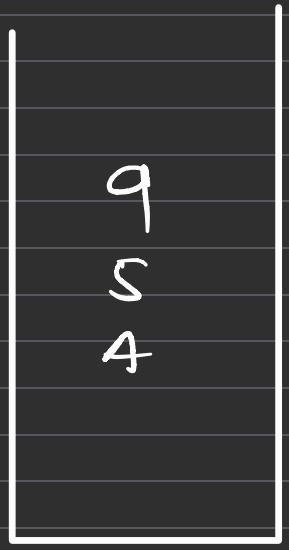


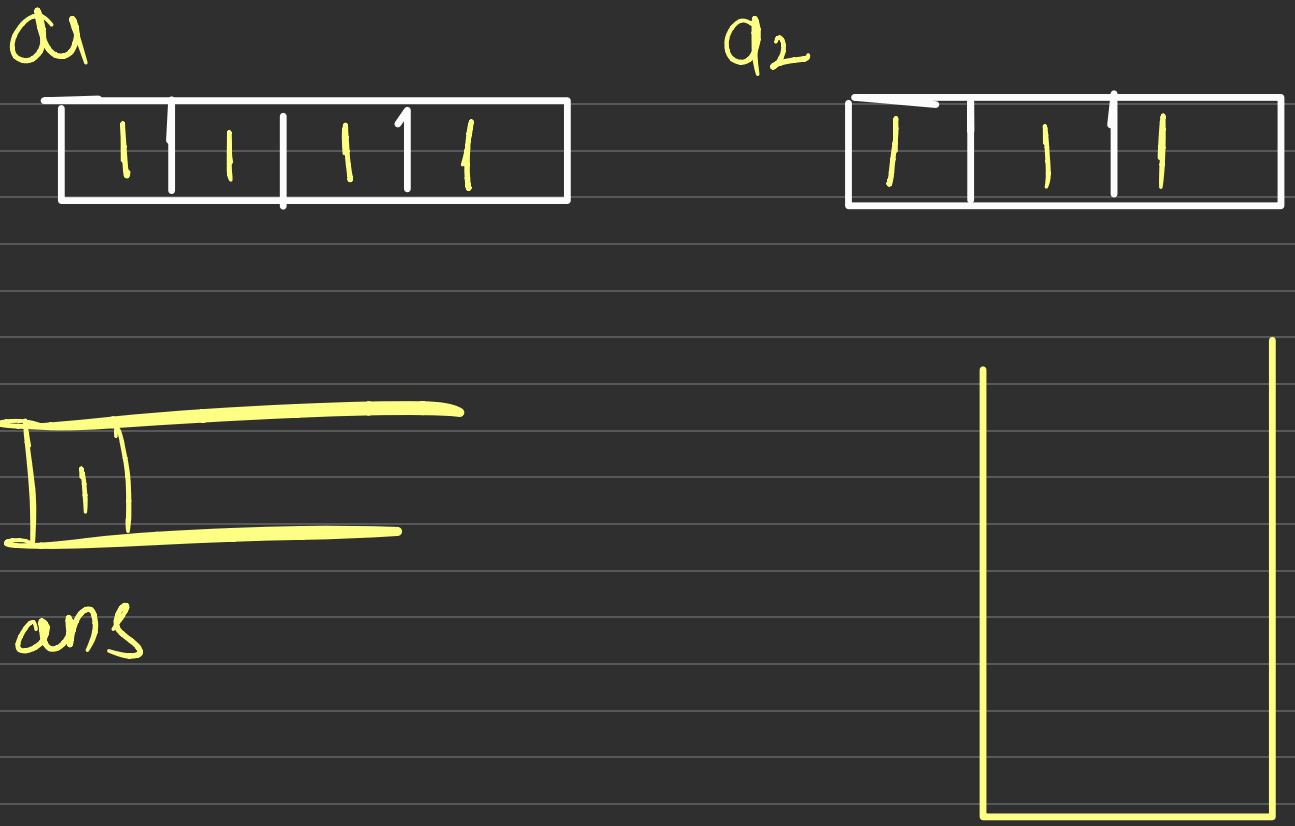

a2



ans

if Element is present in set, take it & remove from set





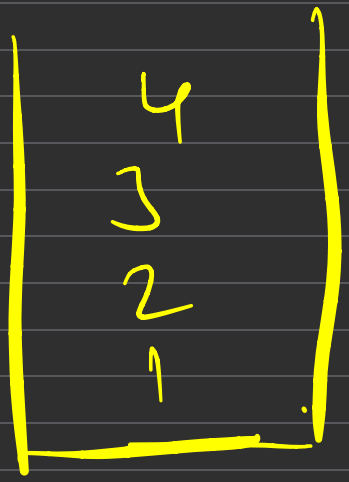
```
int[] intersect (a1, a2, n, m)
{
    set <int> st
    for (auto it : a2)
        st.insert (it)

    for (auto it : a1)
    {   if (st.find (it) != st.end())
        {   ans.push (it)
            st.erase (it)
    }   }
```

return ans

set<Integer> st = HS

st.contains(3)



## Find Only Repetitive Element from 1 to n-1

Difficulty: **Easy** Accuracy: **59.22%** Submissions: **20K+** Points: **2**

Given an array **arr[]** of size **n**, filled with numbers from **1** to **n-1** in random order. The array has **only** one repetitive element. Your task is to find the **repetitive element**.

**Note:** It is guaranteed that there is a repeating element present in the array.

**Examples**:

Exactly ① ⇓ appears more than 1

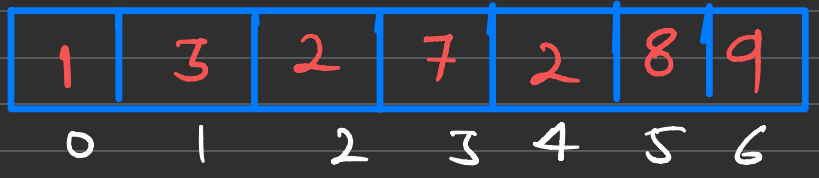
**Input**: arr[] = [1, 3, 2, 3, 4]
**Output**: 3
**Explanation**: The number 3 is the only repeating element.

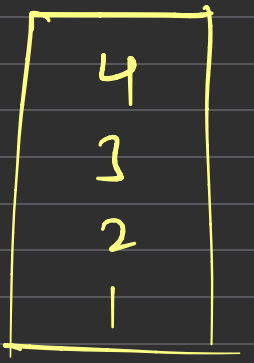
**Input**: arr[] = [1, 5, 1, 2, 3, 4]
**Output**: 1
**Explanation**: The number 1 is the only repeating element.

**Input**: arr[] = [1, 1]
**Output**: 1
**Explanation**: The array is of size 2 with both elements being 1, making 1 the repeating element.

```
int duplicate ( int [] arr , n)
{
    for (i=0 ; i<n ; i++)
    {   for(j=i+1 ; j<n ; j++)
        {   if (a(i) == a(j))
                return a(i)
        }
    }
}
```

O(N) — O(N-1) — O(N²)




```
set <int> st
for (i=0; i<n; i++)
{       int x = a[i]
        if ( st.find (x) != st.end ())
        {
            return x
        }
        st.insert (x)
}
```

x=3
x=7