One good offer can change your life,

WHY WE NEED HASHING

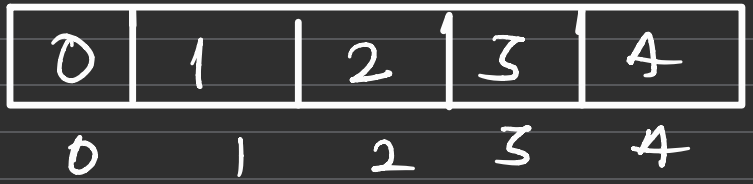
arr:

$x = 3$

O(1)

O(N)

arr



$x = 3$

$x = 1$

1008768908 1008768909 1008768910

WHAT IS HASHING

Hashing is a technique that efficiently stores and retrieve the data in a way that allow for the quick access.

We can perform the operation like insert, delete & update in O(1) time.

In hashing, we mainly convert large number into smaller number & we use the smaller number as a index to store the large number.

This can be done using the hash function

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

X = key which need to insert in the table
N = size of the hash table.

Keys = 3 9 14 25 36 2 1 47 18 20

N = 10

$h(x) = x \% N$

$3 \% 10 = 3$

$9 \% 10 = 9$

$14 \% 10 = 4$

$25 \% 10 = 5$

$36 \% 10 = 6$

$2 \% 10 = 2$

$1 \% 10 = 1$

$47 \% 10 = 7$

$18 \% 10 = 8$

$20 \% 10 = 0$

$x = 36$

Key: 3 9 14 25 36 2 74 46 18 20

N = 10

$h(x) = x \% N$

$3 \% 10 = 3$

$9 \% 10 = 9$

$14 \% 10 = 4$

$25 \% 10 = 5$

$36 \% 10 = 6$

$2 \% 10 = 2$

$74 \% 10 = 4$

$46 \% 10 = 6$

$18 \% 10 = 8$

$20 \% 10 = 0$

Collision

COLLISION TECHNIQUES

When two or more numbers get the same hash value, it called as collision.

This can be solved using two methods

- Separate chaining
- Open Addressing
  - linear probing
  - Quadratic probig
  - Double Hashing

Separate Chaining

Key = 3 9 14 25 74 2 36 18 46 20

N = 10

$h(x) = x \% N$

$3 \% 10 = 3$

$9 \% 10 = 9$

$14 \% 10 = 4$

$25 \% 10 = 5$

$74 \% 10 = 4$

$2 \% 10 = 2$

$36 \% 10 = 6$

$18 \% 10 = 8$

$46 \% 10 = 6$

$20 \% 10 = 0$

Linear probing

Key = 3 9 14 25 36 2 74 46 24 35

$$h(x) = (x + i) \% N \quad (i = 0 \ldots\ldots N-1)$$

$(3+0) \% 10 = 3$

$(9+0) \% 10 = 9$

$(14+0) \% 10 = 4$

$(25+0) \% 10 = 5$

$(36+0) \% 10 = 6$

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

$(74+0) \% 11 = 4$

$(74+1) \% 10 = 5$

$(74+2) \% 10 = 6$

$(74+3) \% 10 = 7$ ✗

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

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

$(46+2) \% 10 = 8$

9	9
46	8
74	7
36	6
25	5
14	4
3	3
2	2
35	1
24	0

$(24+0)\ \%10 = 4$

$(24+1)\ \%10 = 5$

$(24+2)\ \%10 = 6$

$(24+3)\ \%10 = 7$

$(24+4)\ \%10 = 8$

$(24+5)\ \%10 = 9$

$(24+6)\ \%10 = 0$

$(35+0)\ \%10 = 5$

$(36+6)\ \%10 = 1$

Quadratic probing

N = 10

Key: 3 9 14 25 36 2 74 46 18 20

$$h(x) = (x + i^2) \% N \quad (i = 0 \ldots (N-1))$$

$(3+0^2) \% 10 = 3$

$(9+0^2) \% 10 = 9$

$(14+0^2) \% 10 = 4$

$(25+0^2) \% 10 = 5$

$(36+0^2) \% 10 = 6$

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

$(74+0^2) \% 10 = 4$

$(74+1^2) \% 10 = 5$

$(74+2^2) \% 10 = 8$

$(46+0^2) = 6$

$(46+1^2) = 7$

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

$(18+1^2) \% 10 = 9$

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

$(18+3^2) \% 10 = 7$

$(18+4^2) \% 10 = 4$

$(18+5^2) \% 10 = 3$

$(18+6^2) \% 10 = 4$

$(18+7^2) \% 10 = 7$

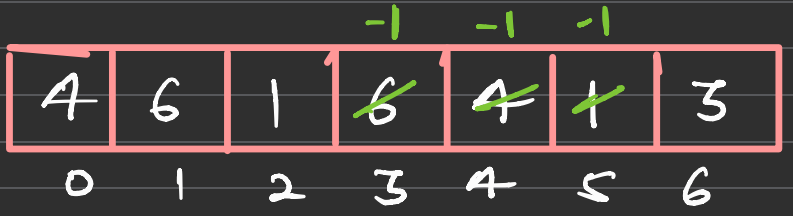
$(18+8^2) \% 10 = 2$

$(18+9^2) \% 10 = 9$

$(18+10^2) \% 12$

$(18+11^2) \% 10$

Count occurrence of each element in array



4 → 2
6 → 2
1 → 2
3 → 1

$O(N^2)$

```
(N)   for (i=0 ; i<n ; i++)
      {   if (a[i] == -1) continue
(N-1)     for (j=i+1 ; j<n ; j++)
          {   if (a[i] == a[j])
              {   a[j] = -1
                  cnt++
              }
          }
          print (a[i], cnt)
      }
```

$N \times (N-1)$

$N^2 \times N$

arr: 4 6 1 6 4 1 3

sorted arr: 1 1 3 4 4 6 6

↑ ↑

cnt = 1 ~~2 2 2 2~~

~~x2~~

6 → 2

1 → 2

3 → 1

4 → 2

if (a[i] == a[i-1])

cnt++

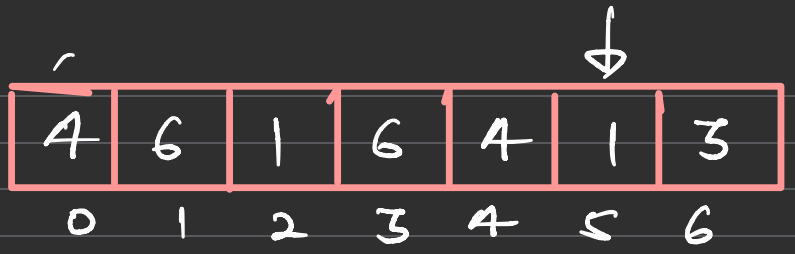
else

print (a[i-1], cnt)

cnt = 1

$O(N \cdot \log N) + O(N)$

(Key, value)



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

```
for (i=0 ; i<n ; i++)
{
    mp[a(i)]++      mp(A) = 1++
}
```

```
for( i=0 ; i<n ; i++)
{
mp.put(a(i), mp.getOrDefault(a(i), 0) +1 )
```

CPP

map <int, int> mp

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    vector <int> arr {6, 1, 4, 2, 1, 6, 10, 4, 4, 2, 2, 3};
    int n = arr.size();

    map <int, int> mp;

    for (auto i: arr){
        mp[i]++;
    }

    for (auto i: mp){
        cout << i.first << " --> " << i.second << "\n";
    }


}
```

Java

Map <Integer, Integer> mp = new HashMap<>()

```java
public static void main(String[] args) {

    int [] arr = {6, 1, 4, 2, 1, 6, 10, 4, 4, 2, 2, 3};
    int n = arr.length;

    Map <Integer, Integer> mp = new HashMap<>();

    for (int i: arr){
        mp.put(i, mp.getOrDefault(i, defaultValue: 0) + 1);
    }

    // map traverse
    for (Map.Entry<Integer, Integer> i: mp.entrySet()){
        System.out.println(i.getKey() + " ---> " + i.getValue());
    }
```

3005. Count Elements With Maximum Frequency

Solved

Easy | Topics | Companies | Hint | Re-do

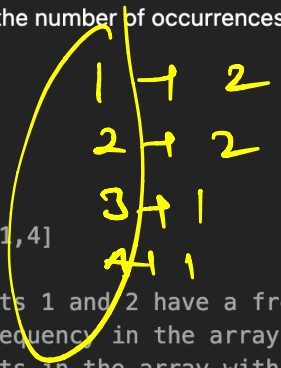
You are given an array `nums` consisting of **positive** integers.

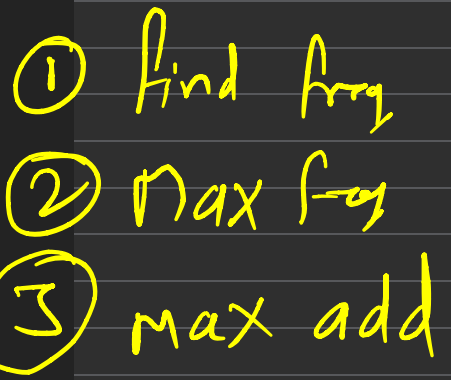
Return *the **total frequencies** of elements in* `nums` *such that those elements all have the **maximum** frequency*.

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.
```





```java
public int maxFrequencyElements(int[] nums) {

    Map <Integer, Integer> mp = new HashMap<>();

    // step 1: find the freq
    for (int i: nums){
        mp.put(i, mp.getOrDefault(i, 0) + 1);
    }

    // step 2: get the max freq among all
    int maxFreq = 0;
    for (Map.Entry<Integer, Integer> i: mp.entrySet()){
        if (i.getValue() > maxFreq){
            maxFreq = i.getValue();
        }
    }

    // step 3: add all the freq which is equal to maxFreq
    int ans = 0;

    for (Map.Entry<Integer, Integer> i: mp.entrySet()){
        if (i.getValue() == maxFreq){
            ans += i.getValue();
        }
    }

    return ans;
```

```cpp
int maxFrequencyElements(vector<int>& nums) {
    int n = nums.size();

    map <int, int> mp;

    // step 1: find the freq of all the numbers from the arr
    for (auto i: nums){
        mp[i]++;
    }

    // step 2: find the max freq among all the freq
    int maxFreq = 0;
    for (auto i: mp){
        if (i.second > maxFreq){
            maxFreq = i.second;
        }
    }

    // step 3: add all the freq which is equal to maxFreq
    int ans = 0;
    for (auto i: mp){
        if (i.second == maxFreq){
            ans += i.second;
        }
    }

    return ans;
```

Pairs with difference k

Difficulty: **Easy** Accuracy: **22.41%** Submissions: **59K+** Points: **2**

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.

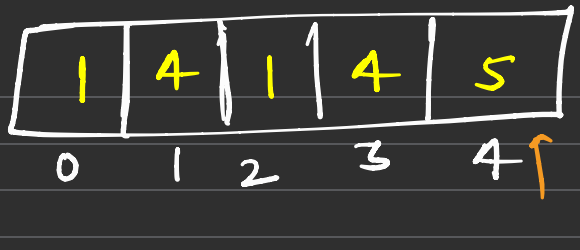
$|a-b| = k$

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}.

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



$|a-b| = k$

$a = k+b$

$a = k+b$

$= 3+1 = 4$

$= 3+4 = 7$

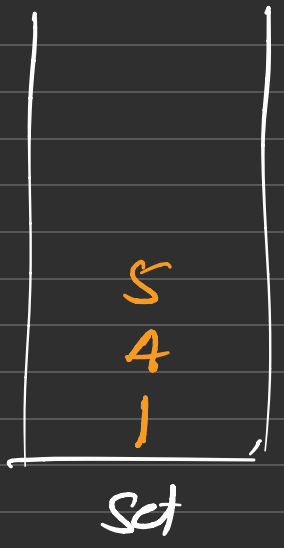
$= 3+1 = 4$

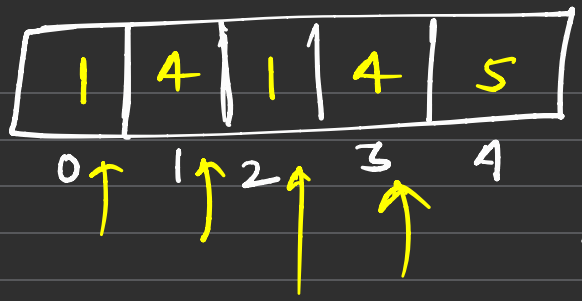
$= 3+4 = 7$

$= 3+5 = 8$

cnt = 1

Set





$|a-b| = K$

$|a| = K+b$

$= 3+1 = 4$

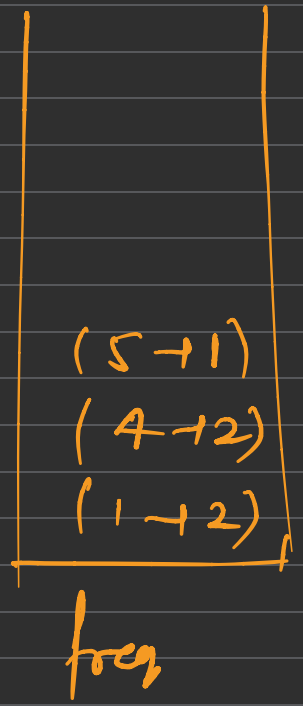
$= 3+4 = 7$

$= (3+1) = 4$

$= (3+4) = 7$

$= (3+5) = 8$

cnt = ~~2~~ 4



```java
int countPairs(int[] arr, int k) {
    Map <Integer, Integer> mp = new HashMap<>();

    for (int i: arr){
        mp.put(i, mp.getOrDefault(i, 0) + 1);
    }

    /*
        |a - b| = k
        |a| = k + b
        check if a is present in the map or not
    */
    int cnt = 0;
    for (int i: arr){
        int a = k + i;

        if (mp.containsKey(a) == true){
            cnt += mp.get(a);
        }
    }

    return cnt;
}
```

```cpp
int countPairs(vector<int>& arr, int k) {
    map <int, int> mp;

    for (auto i: arr){
        mp[i]++;
    }

    int cnt = 0;
    for (auto i: arr){
        int a = k + i;

        // check if a is present in map or not
        if (mp.find(a) != mp.end()){
            cnt += mp[a];
        }

    }

    return cnt;
}
```

Union of Arrays with Duplicates

(4, 3, 2)

Difficulty: **Easy** Accuracy: **42.22%** Submissions: **465K+** Points: **2** Average Time: **10m**

(2, 3, 6)

You are given two arrays **a[]** and **b[]**, return the **Union** of both the arrays in any order.

The **Union** of two arrays is a collection of all **distinct elements** present in either of the arrays. If an element appears more than once in one or both arrays, it should be included **only once** in the result.

(2, 3, 4, 6)

Note: Elements of **a[]** and **b[]** are not necessarily distinct.
Note that, You can return the Union in any order but the driver code will print the result in **sorted order** only.

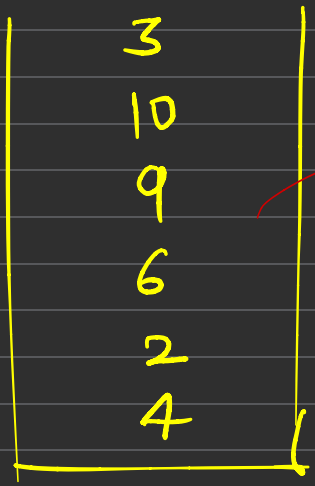
Examples:

Input: a[] = [1, 2, 3, 2, 1], b[] = [3, 2, 2, 3, 3, 2]
Output: [1, 2, 3]
Explanation: Union set of both the arrays will be 1, 2 and 3.

Input: a[] = [1, 2, 3], b[] = [4, 5, 6]
Output: [1, 2, 3, 4, 5, 6]
Explanation: Union set of both the arrays will be 1, 2, 3, 4, 5 and 6.

arr1 = { 4, 2, 6, 9, 4 } arr2 = { 9, 10, 6, 3, 4 }

ans = { 4, 2, 6, 9, 10, 3 }



Union

Set → Sorted

Unordered ✗

```cpp
vector<int> findUnion(vector<int>& a, vector<int>& b) {

    set <int> st;

    // step 1: put all the elements from arr1
    for (auto i: a){
        st.insert(i);
    }

    // step 2: put all the elements from arr2
    for (auto i: b){
        st.insert(i);
    }

    vector <int> ans(st.begin(), st.end());

    return ans;

}
```

```java
    public static ArrayList<Integer> findUnion(int[] a, int[] b) {

        Set <Integer> st = new HashSet<>();

        for (int i: a){
            st.add(i);
        }

        for (int i: b){
            st.add(i);
        }

        ArrayList<Integer> ans = new ArrayList<Integer>();
        ans.addAll(st);

        return ans;
    }
```

349. Intersection of Two Arrays

Solved

Easy | Topics | Companies | Re-do

Given two integer arrays `nums1` and `nums2`, return *an array of their* intersection. Each element in the result must be **unique** and you may return the result in **any order**.

Example 1:

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

Example 2:

```
Input: nums1 = [4,9,5], nums2 = [9,4,9,8,4]
Output: [9,4]
Explanation: [4,9] is also accepted.
```

Find Only Repetitive Element from 1 to n-1

Difficulty: **Easy** Accuracy: **59.22%** Submissions: **22K+** 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:

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.