Hashing ****

0	1	2	3	4
1	2	1	3	2

frequency of elements

1 - 2

2 - 2

3 - 1

4 - 0

10 - 0 ⟶ Q

Given an array, asked a lot of times. अब हर बार तो iterate करेंगे नहीं ।

Brute force Approach -

```
int funct (int n, int arr[]) {
    int cnt = 0;
    for (int i = 0; i < arr.size(); i++) {
        if (arr[i] == n) cnt++;
    }
    return cnt;
}
```

⟶ O(N)

for Q no's

O(Q x N)

If array size $N = 10^5$
Operations $Q = 10^5$
So, TC $O(10^5 \times 10^5)$
$= O(10^{10})$

$O(10^8) \cong 1s$, $O(10^{10}) \cong 100$ sec

cannot wait for so much time

Hashing

↙ Pre-Storing

↘ Fetching

watch code

At max अगर 12
↓
hash[13] → 0 1 2 3 4 5 6 7 8 9
10 11 12

what if array had max elements
till 10^9 can we declare

hash array of hash[10^9+1]
The ans is X NO

The maximum size array is (10^6)

Inside main

arr[10^7] inside main will through segmentation fault error.

watch the 1.6 folder

storing & Fetching in map in all case -> takes O(log N) where N is the number of elements in N.

In Case of Unordered Map →

Average cases → O(1)

Best cases →

Worst cases → O(N)

In maps anything can be key, values
In UM pair <, > cant be
keys only single datatypes not pair

UM usually O(1) so first go with that, occassionally due to underlying hashing algo implementation & collision it may take linear time of O(N)