Basic Hashing

very important for dsa

1	2	1	3	2

```
int f(number, arr[]){
  cnt=0;
  for(i=0; i<n; i++){
    if(arr[i] = number){
      cnt+=1;
    }
  }
  return cnt;
};
```

when asked many times:

1 appears 2
3 → 1
4 → 0
2 → 2

check if every number 1 by 1 is equal to num.
asked for: O(N) x (num of ans / # of values asked to check)

however, this can take very long if ↗ is a huge number, so we use hashing: prestoring / fetching

{value max given}

```
hashArr[] =
```

0	0+2	0+2	0+1	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10

10 = maxValue

prestores in another array

```
hashArr[1]=2 // instead of asking then looping
hashArr[2]=2
hashArr[3]=1
hashArr[4]=0
```

solved all queries at once

// if arr has large size

```
arr[number] <= arr[10^6] // max arr size
arr[number] <= arr[10^7] // declared outside of main (global)
```

Map / Hash Map

stl (c) | collection (java) | arr[] =

map / unordered_map → hashMap | 1, 2, 3, 1, 3, 2

map<int key, int value>

key → number; value → frequency

*
mpp{arr[i]}++

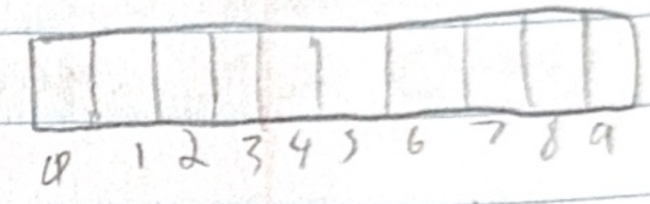
storing / fetching } (logN)

Hashing Methods

Mid Square	Folding	Division

{2, 5, 16, 28, 39}

*cannot create array ≤ 10



mod o/s array to trim

28%10 = 8

Collision = when multiple numbers go to same hash space ex) 8 holds 28, 38, 58, etc.