Scenario1 :

→ Hotel with 10 rooms

→ Maintain a register

Room No	Occupied	AC/Non AC				
1						
2						
⋮						

Scenario2 : Hotel with 1000 rooms

bool arr [1000] ;

Scenario3: Covid has hit and rooms were re-numbered

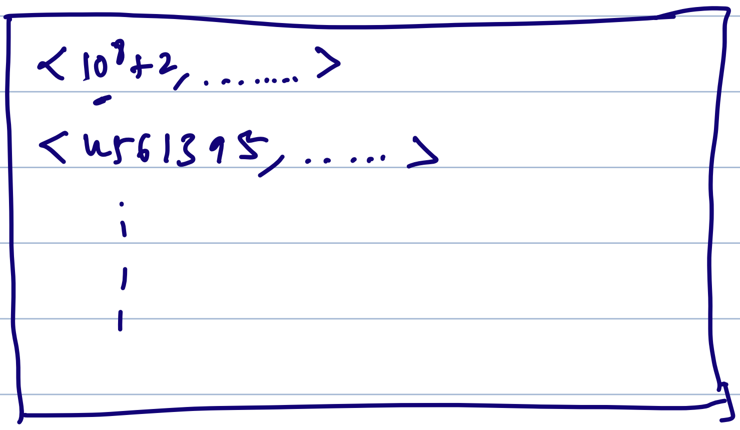
[0 - 999] ⇒ Choose any 1000 random numbers in [1 – 10^9]

[10^9+3, 4516797 . . . .]

arr[10^9] ?

Array is not a feasible option.

⇒ HashMap has < key, value > pairs



Hashmaps

Keys in a hashmap should be unique

Quiz : Population of every country

Hashmap < String, Long >
country, population

Quiz : Hstate of every country

Hashmap < String , Integer >
country , Hstates

Quiz: Names of all states

Hashmap < String, List<String> >
country

Quiz: In every country, store population of each state

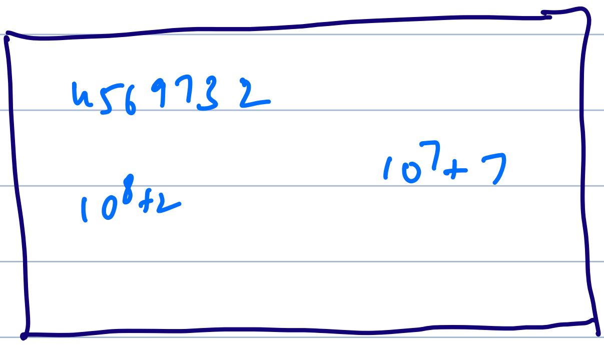
HashMap < String, HashMap < String, Long > >
country state population

↑ Key ↑ Value

→ value can be anything
→ key can be primitive datatype
(Integer, Long, Double, Boolean, Float, String)

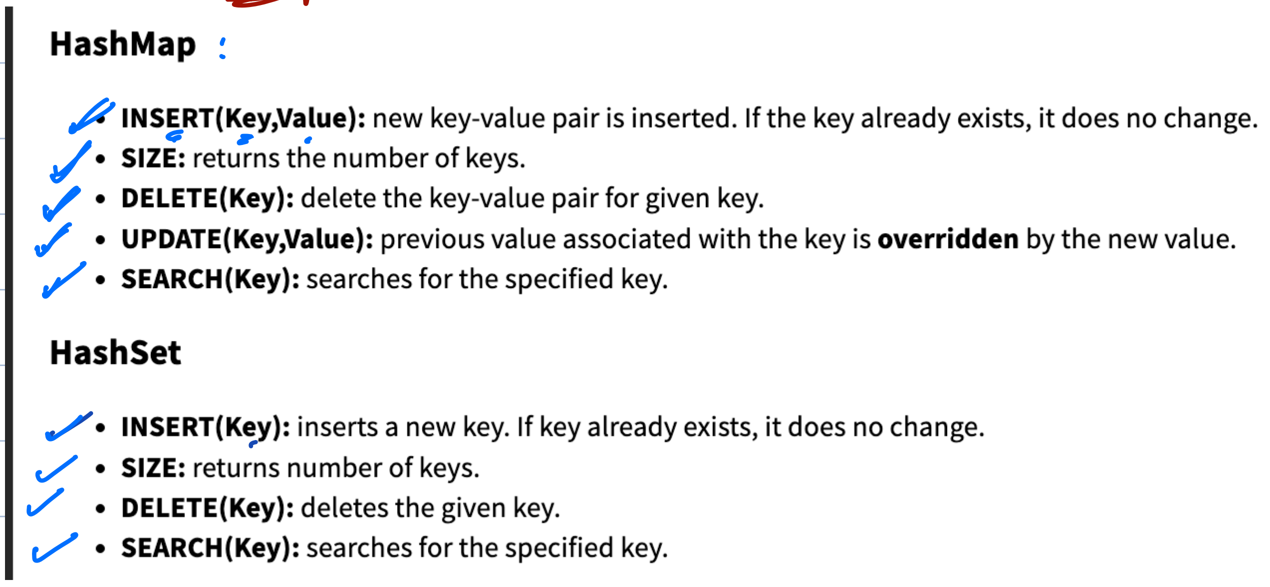
Hashset

A list of just the keys



Key has to be unique

HashMap :



HashMap :

- ✓ **INSERT(Key,Value):** new key-value pair is inserted. If the key already exists, it does no change.
- ✓ **SIZE:** returns the number of keys.
- ✓ **DELETE(Key):** delete the key-value pair for given key.
- ✓ **UPDATE(Key,Value):** previous value associated with the key is **overridden** by the new value.
- ✓ **SEARCH(Key):** searches for the specified key.

HashSet

- ✓ **INSERT(Key):** inserts a new key. If key already exists, it does no change.
- ✓ **SIZE:** returns number of keys.
- ✓ **DELETE(Key):** deletes the given key.
- ✓ **SEARCH(Key):** searches for the specified key.

T.C: O(1)

Hashing Library Names in Different Languages

	Java	C++	Python	Js	C#
Hashmap	Hashmap	unordered_map	dictionary	map	dictionary
Hashset	Hashset	unordered_set	set	set	Hashset

Question: Given an array and Q queries, find frequency of each query

A: 2 3 3 1 3 4 1 4 9 8

Queries:

3 : 3

2 : 1

5 : 0

$N \le 10^5$

$Q \le 10^5$

Approach 1: Brute Force

For every query, iterate the array

T.C: $O(Q \cdot N)$ S.C: $O(1)$

Approach: Freq Hashmap

A : 2 3 3 1 3 4 1 4 9 8

Hashmap < integer, Integer >

Query:

2 ⇒ 1

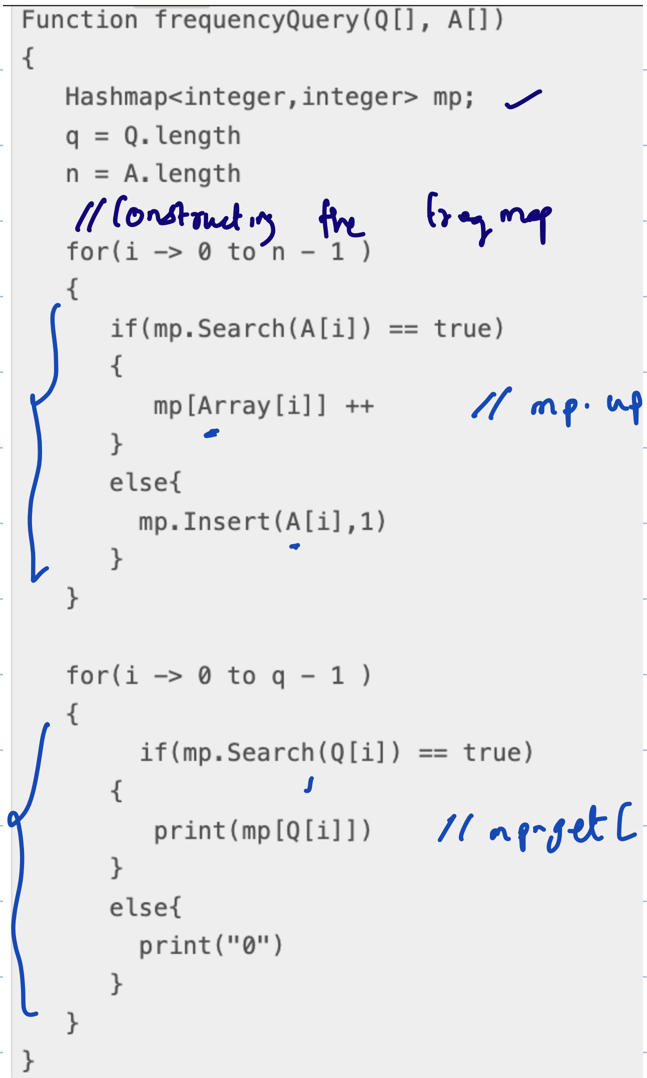
3 ⇒ 3

5 ⇒ 0

<2, 1>	<4, 2>
<3, 3>	<9, 1>
<1, 2>	<8, 1>

⇒

Hashmap



```
Function frequencyQuery(Q[], A[])
{
   Hashmap<integer,integer> mp;
   q = Q.length
   n = A.length
   //Constructing the freq map
   for(i -> 0 to n - 1 )
   {
      if(mp.Search(A[i]) == true)
      {
         mp[Array[i]] ++        // mp.update[ A[i], map.get[A[i]]+1 ];
      }
      else{
        mp.Insert(A[i],1)
      }
   }

   for(i -> 0 to q - 1 )
   {
        if(mp.Search(Q[i]) == true)
      {
         print(mp[Q[i]])        // mp.get[Q[i]];
      }
      else{
        print("0")
      }
   }
}
```

O(N)

O(Q)

T.C: $O(N+Q)$
S.C: $O(N)$

Question: Count # of distinct/unique elements

A : 3 5 5 4 3 5 4 3

3, 5, 4 $\Rightarrow$ 3

→ Use a Hashset and insert all elements to it.
→ Return size of Hashset

```
Function distinctCount(Array[])
{
  hashset<integer>set; ✓
  for(i -> 0 to Array.length - 1 )
  {
    set.insert(Array[i])
  }
  return set.size();
}
```

T.C: $O(N)$
S.C: $O(N)$
↳ Hashset

Questn:

Question: Pair Sum = k
In array of integers, check if there exists a pair a[i], a[j], i != j such that
a[i] + a[j] = k;

A :	7	4	10	2	5	16	3
	0	1	2	3	4	5	6

K : 9 $\Rightarrow$ A[0] + A[3] $\Rightarrow$ True
A[1] + A[4]

K = 11 A[0] + A[1]

K = 3 False

K = 4 A[3] + A[3] False

Approach1: Brute Force

A :	↓ 7	4	10	2	5	18	3
	0	1	2	3	4	5	6

K = 7

$$N \begin{cases} a = 7, & 7 - 7 = 0 & \Rightarrow & & O(N) \\ a = 4, & 7 - 4 = 3 & \Rightarrow & \text{Check if 3 exists} & O(N) \\ \vdots \end{cases}$$

T.C : $O(N^2)$

S.C : $O(1)$

Approach2 : Hashmap

For every A[i], we are spending O(N) T.C to check if (K - A[i]) exists or not

Eg 1: Set = { $\overset{\downarrow}{1}$, $\overset{\downarrow}{4}$, $\overset{\downarrow}{-2}$, $\overset{\downarrow}{8}$, -9, $\underbrace{14}$, 25, 22, 17, 13 }

a+b = 22

a = 1, b = 21

a = 4, b = 18

a = -2, b = 24

a = 8, b = 14 True

a + b = 28

a = 1, b = 27

a = 4, b = 24

a = -2, b = 30

a = 8, b = 20

a = -9, b = 37

a = 14, 28-14 = 14 True if freq (14) > 1

else False

Hence, construct a hashmap

T.C: O(N)

S.C: O(N)

Approach 3 : :

$A[i] + A[j] == K \quad (i < j)$

A[j]

A : 1   4   -2   8   -9   14   25   14   17   13

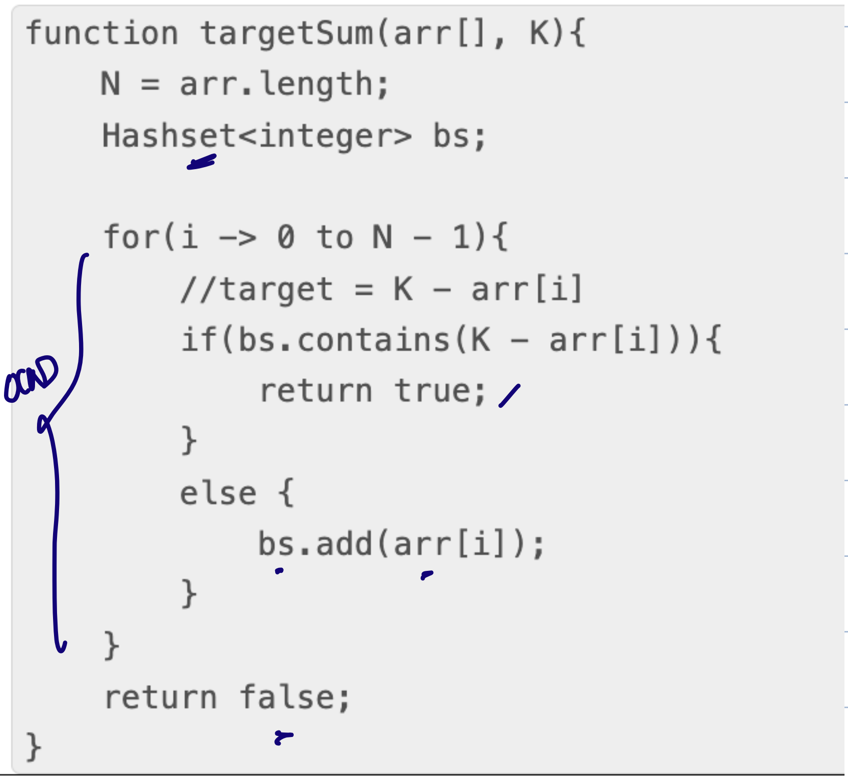
K=28

a : 1 ,	b : 27
a = 4,	b : 24
a = -2,	b = 30
a = 8	b : 20
a = -9 ,	b = 37
a = 14 ,	b : 14
a : 25	b = 3
a = 14	b : 14

$\boxed{\text{True}}$

1   4   14
-2
-9   8
25

Hashset



```
function targetSum(arr[], K){
    N = arr.length;
    Hashset<integer> bs;

    for(i -> 0 to N - 1){
        //target = K - arr[i]
        if(bs.contains(K - arr[i])){
            return true;
        }
        else {
            bs.add(arr[i]);
        }
    }
    return false;
}
```

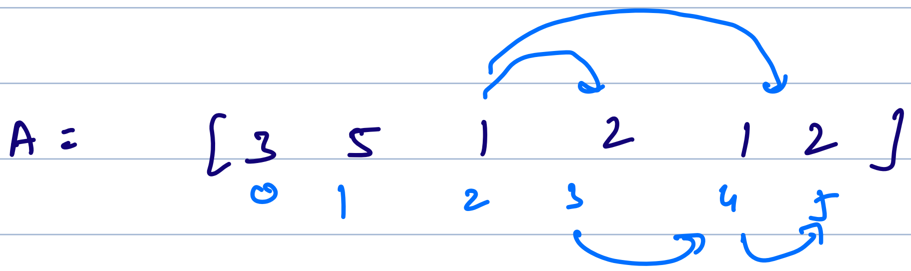
O(N)

(A[i], K - A[i])

T.C : O(N)

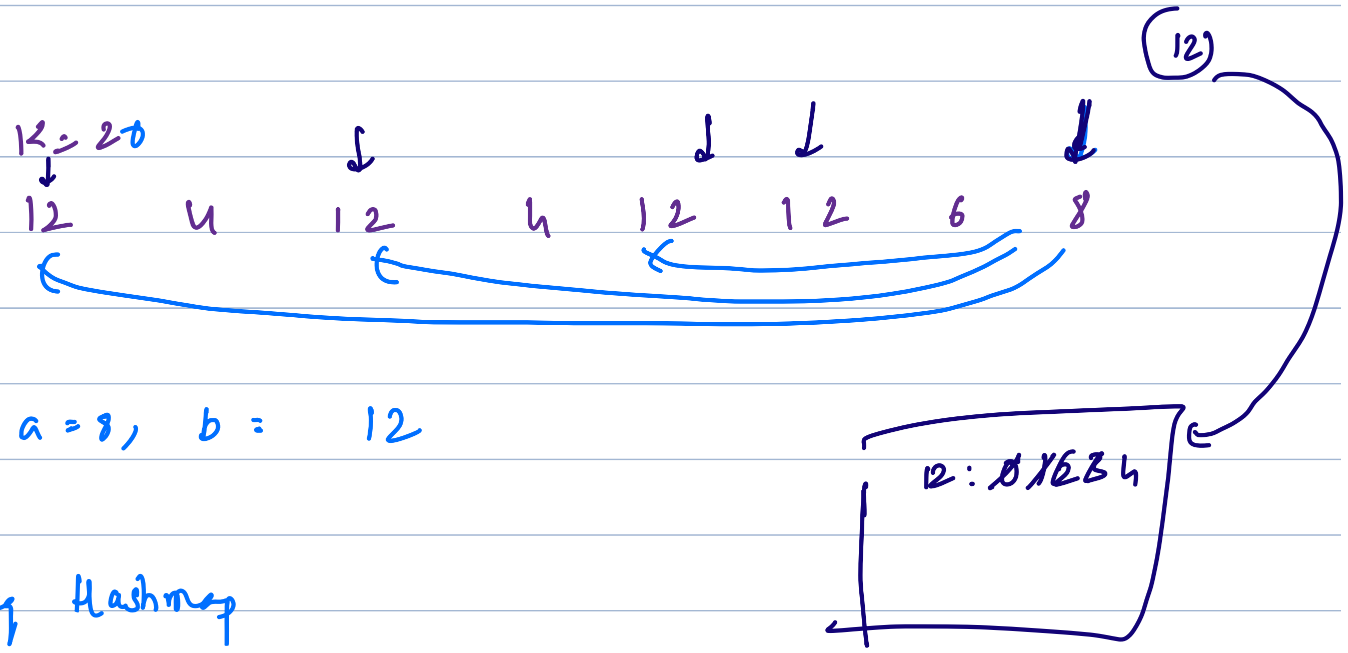
S.C : O(N)
↳ Hashmap

Question: Count #pairs whose sum = k



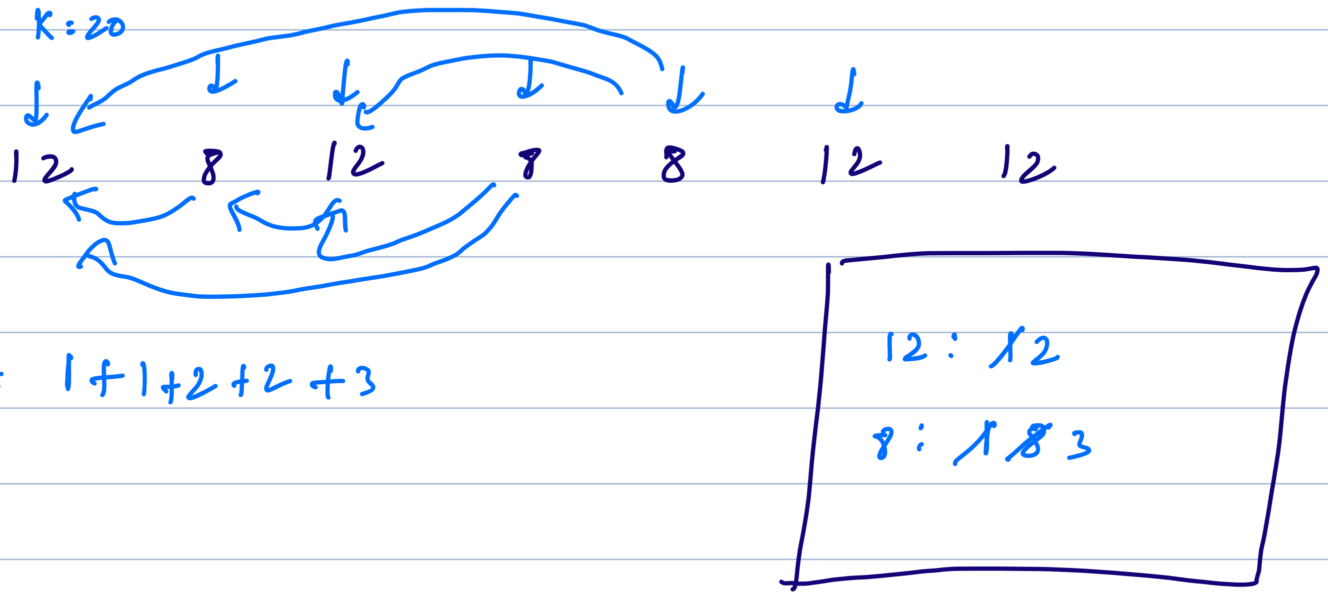
k = 3

(2, 3)
(2, 5)
(3, 4)
(4, 5)

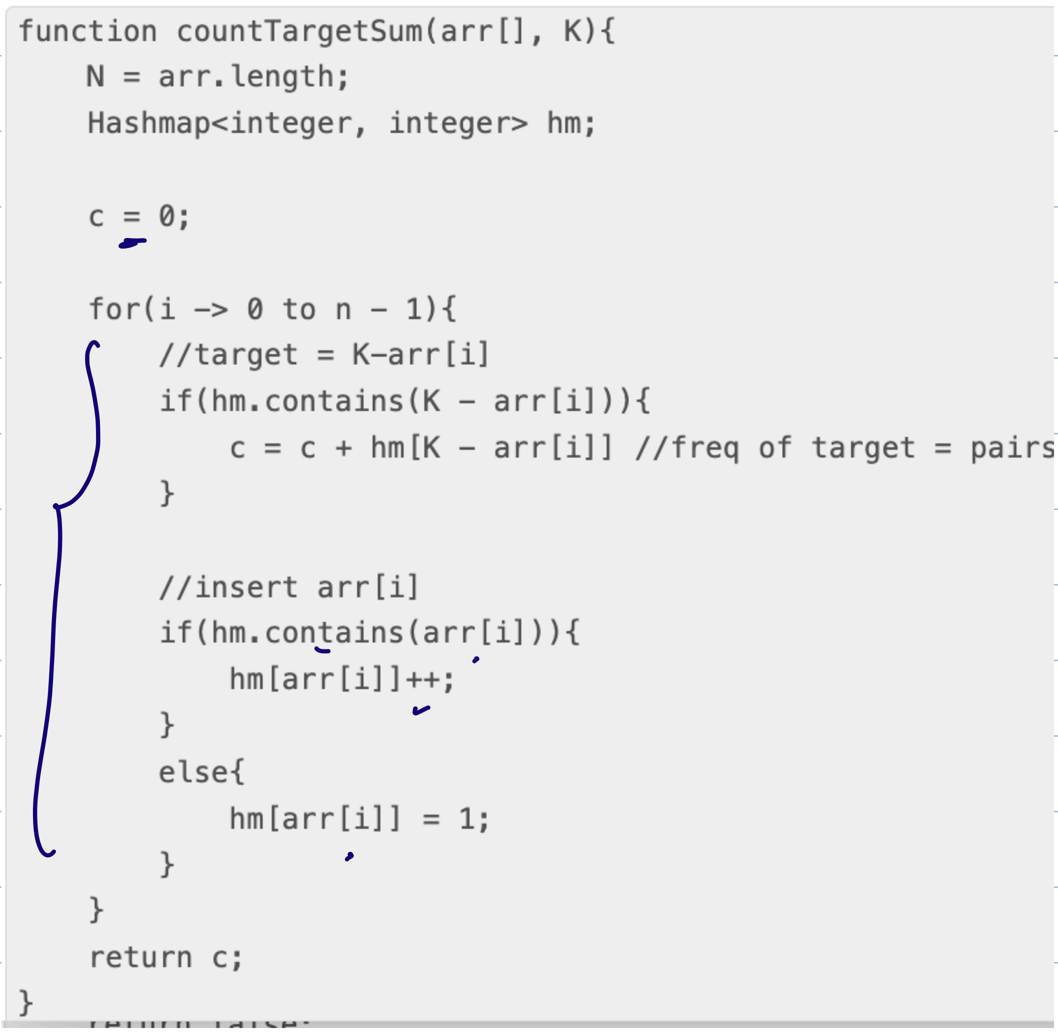


a = 8, b = 12

Freq Hashmap



count : 1 + 1 + 2 + 2 + 3



```
function countTargetSum(arr[], K){
    N = arr.length;
    Hashmap<integer, integer> hm;

    c = 0;

    for(i -> 0 to n - 1){
        //target = K-arr[i]
        if(hm.contains(K - arr[i])){
            c = c + hm[K - arr[i]] //freq of target = pairs
        }

        //insert arr[i]
        if(hm.contains(arr[i])){
            hm[arr[i]]++;
        }
        else{
            hm[arr[i]] = 1;
        }
    }
    return c;
}
```

T-C : O(N)

S.C : O(N)
↳ Hashmap