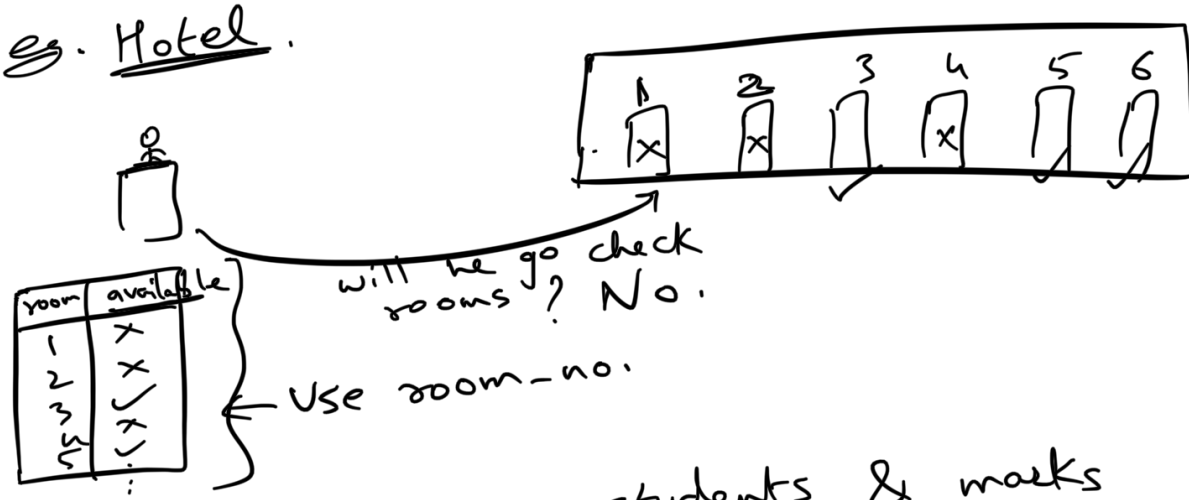
Introduction to Hashing

Eg. Hotel.



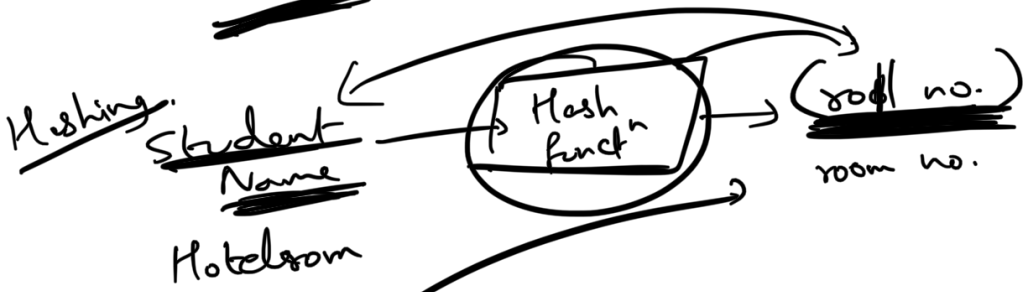
School → 200 students & marks

storing names & finding names is to

↳ Roll_no. → sorted order

Map → String (Name) → int (Roll no.)

Hashing!



HashMap (Key, value)

↳ room_no

↳ marks.

↳ available?

times a num in

eg Q. Find how many [illegible] range [1, 500] occcrred before.

1, 19, 2, 5, 19, 2, 19, 8 etc.

0 0 0 0 1 1 2 0

count[500]

count[i] → how many i has occurre

count[i]++

int a[500]

a[1]++, a[19]++, a[2]++, a[5]++, a[19]++

Now, nos → [1, 100] → 10 numbers.

memory → 10

a[]

58, 31, 27, 50, 9, 27

HF → n % 10

58 % 10 → 8

31 % 10 → 1

27 → 7

50 → 0

9 → 9

2 % 10 → 2

58

38 → 8

28 → 8

58 → 8

7, 27, 97

→ depends on

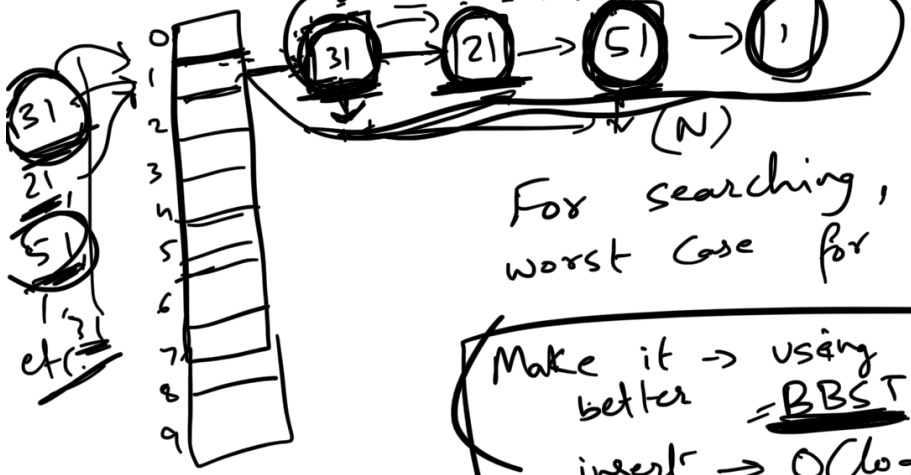
K

Collision! → hash fn & memory.

n%K

m%K

To avoid collision, use → chaining.



- separate ch
- Linear pro
etc.

For searching, worst case for N elent → O(

Make it → using better → BBST etc.
insert → O(log N)
searching → O(log N)

eg Q. Given random strings, for every string check if it has already occurr or not

Right now, array is the only knowledge we have

f(string) → int [0 →

a[10^6]

① hashfunctions.

↳① Add ascii values of string.

$\left(\sum_{i=0}^{m} s[i]\right) \%10^{6}$ collision → abc (1+2+3)

↓

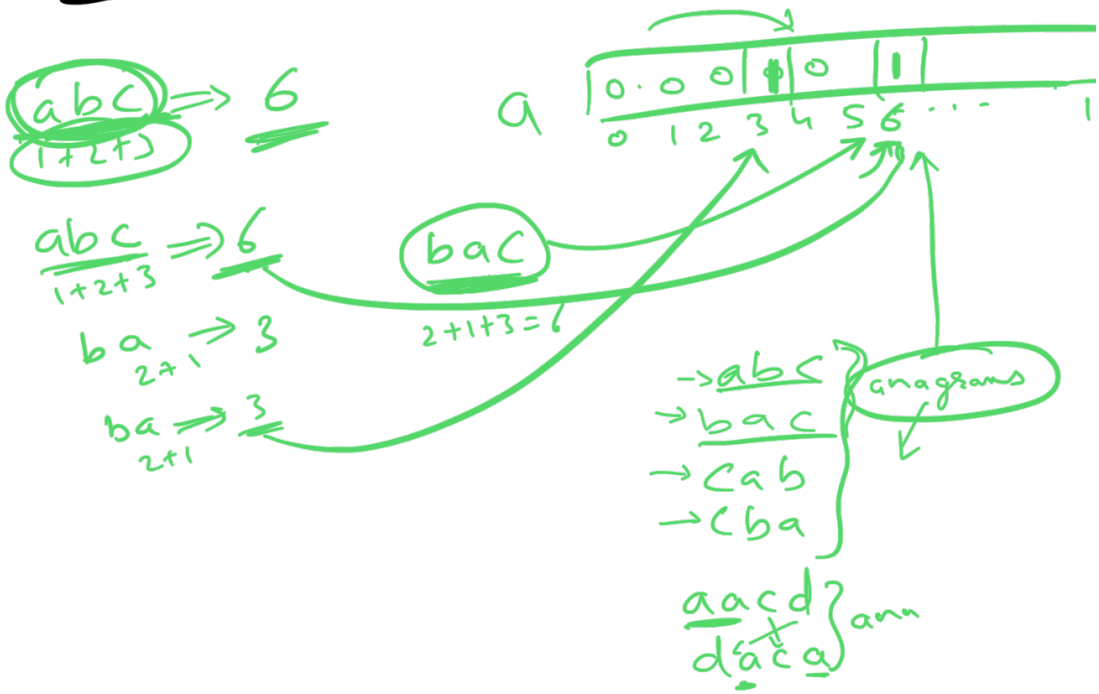
lots of collisions

z → 26

ay → 1+25 → 26

cba → 3+2+1

cab → 3+1+2



② $1 \quad 10^{1} \quad 10^{2} \quad 10^{3}$

abc ...

↳ (a×1 + b×10 + c×100 etc.) %10^6

bac → (b×1 + a×10 + c×100 etc.

multiply by power of 10^index.

abc x y p q a z n

10^{0} ... 10^{10}

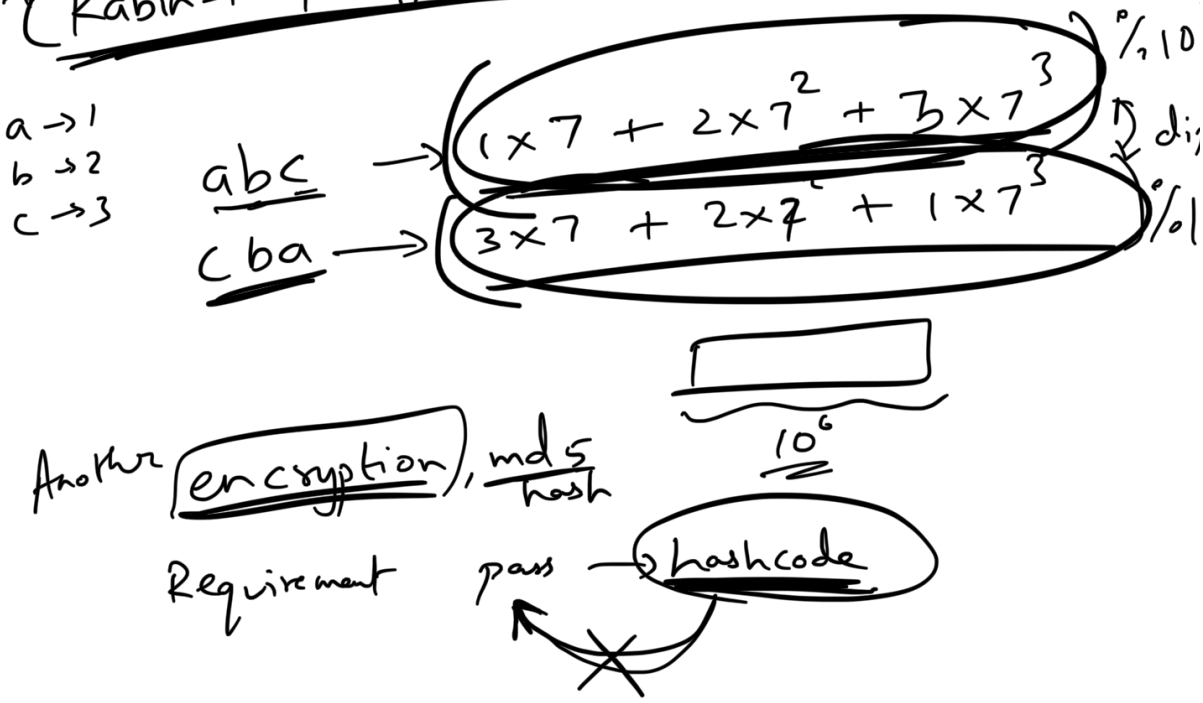
$a \times 1 + b \times 2 + c \times 3$

and hash f^n, → $(a \times p^{0}) + (b \times p^{1})$

For good research says → use prime no. to ensure uniform distribut^n & less collision

$+ (c \times p^2)$ etc.

* (Rabin-Karp algorithm)

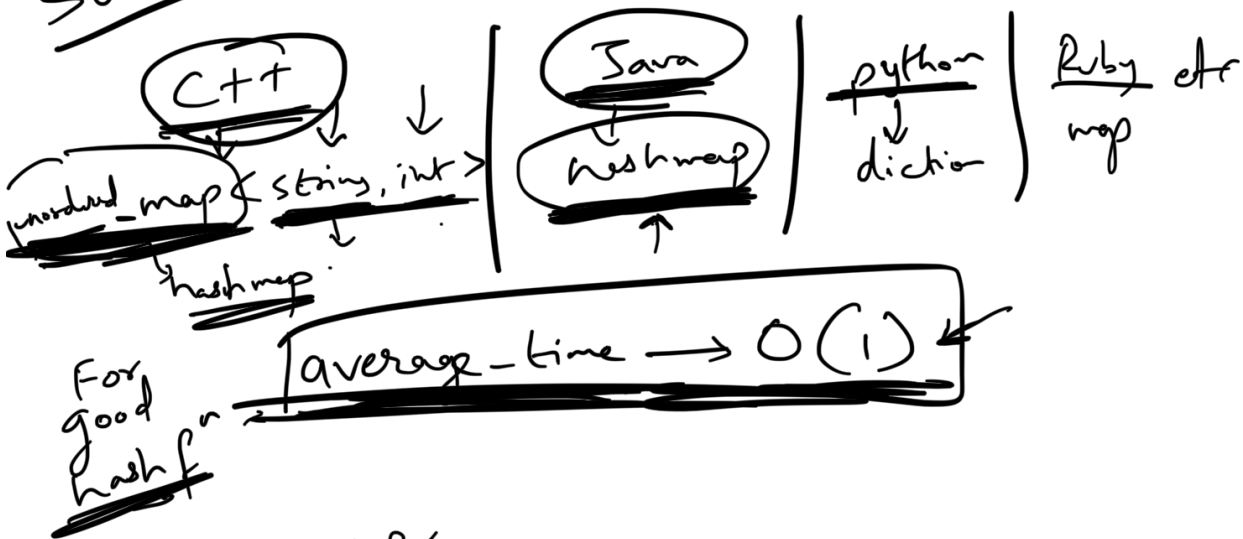


10^6

Another encryption, md5 hash

Requirement pass → hashcode

Summary



map<string, int> a

value

a[abc] = 5

~~abc → 5~~

a[10] → 7 → O(1).

Q.1 Given array, find freq. of every

a = [2, 7, 5, 2, 3, 5, 7, 2] ← N

use hashmap<a[i], freq>.

un_map<int, int>

2 → 3
7 → 2
3 → 1
5 → 2

[2, 2, 2, 7, 7]

2 → 5.

T.C ⇒ O(N)
S.C ⇒ O(N)

```
for(i=0; i<n; i++)
{
  if(HM.contains a[i]) → O(1)
    HM[a[i]]++;
  } else
    HM[a[i]] = 1 → O(1)
```

add freq

```
for(Array)
print a[i] → HM[a[i]]
```

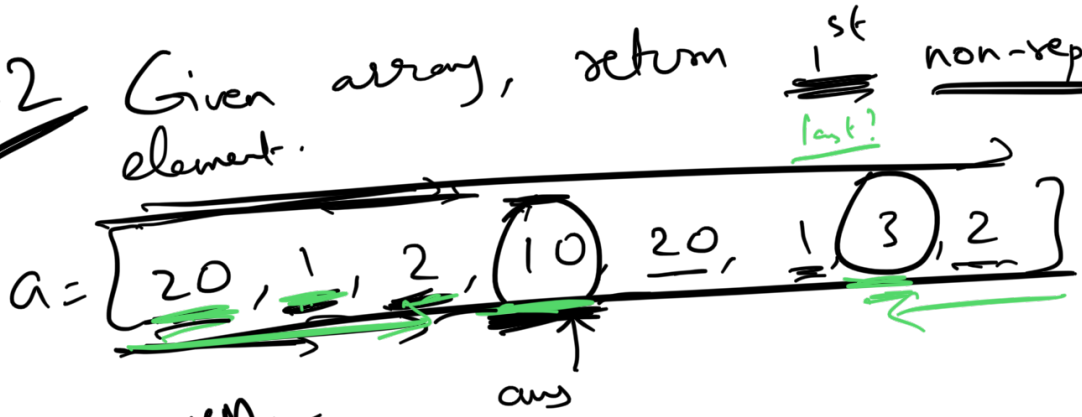
print freq

[1, 2, 7, 5, 13, 100, 200, 201, ... N]

HM[ ] → freq.

Has( ) → value

Q.2 Given array, return 1st non-repeating element.



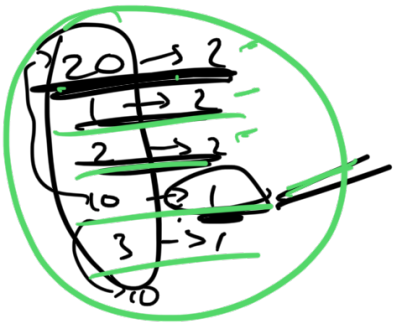
insert into HM.

for(- freq.)

check 1st

for(i=0; i<n; i++ check HM → 1)

return a[i].



HM[value]

insert

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

⇒ O(N)

check

```
for( i=0; i<n; i++ )
{
    if( HM[a[i]] == 1 )
    { return a[i]; }
}
```

i=n-1; i>=0; i--

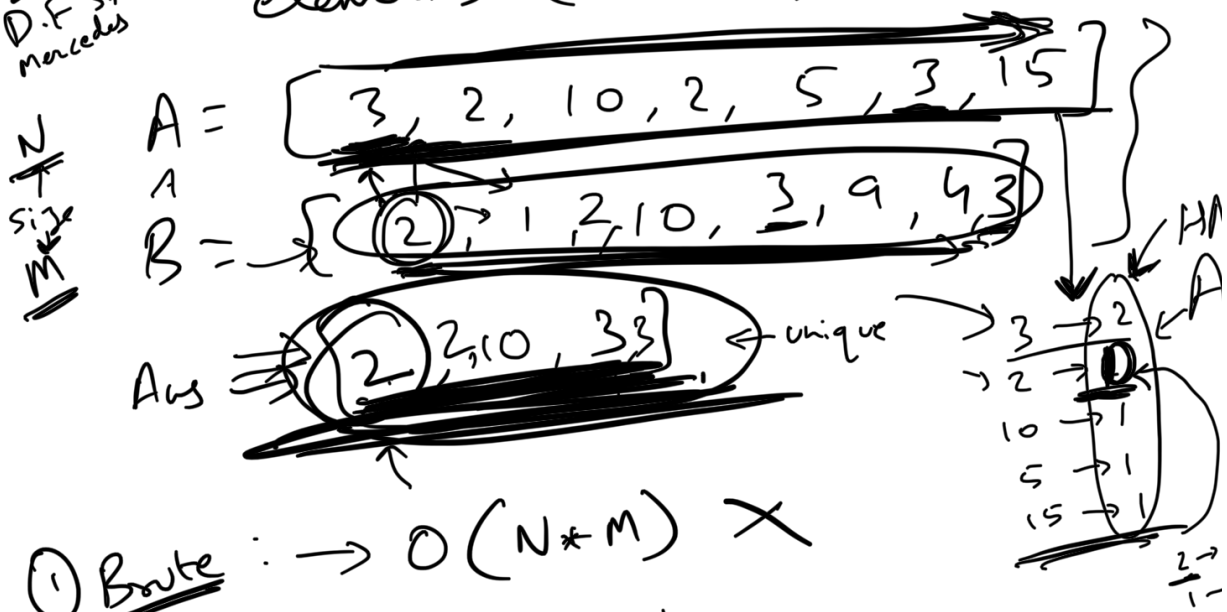
⇒ O(N)

T.C ⇒ O(N)

S.C ⇒ O(N)

Q.3 Given 2 arrays, find the common elements (intersection)

D.E Shaw Mercedes



① Brute :→ $O(N*M)$ ✗

② Use (sort A + Binary search for B in A) ✗

$\Rightarrow O(N \log N + M \log N)$

$\Rightarrow O((N+M)\log N)$

③ Use hashmap for A.

Search elements from 2nd array in HM $\Rightarrow O(N+M)$

T.C $\Rightarrow O(N)$

S.C $\Rightarrow O(N)$

Q.4 Given an array, check if there

FB. etc.

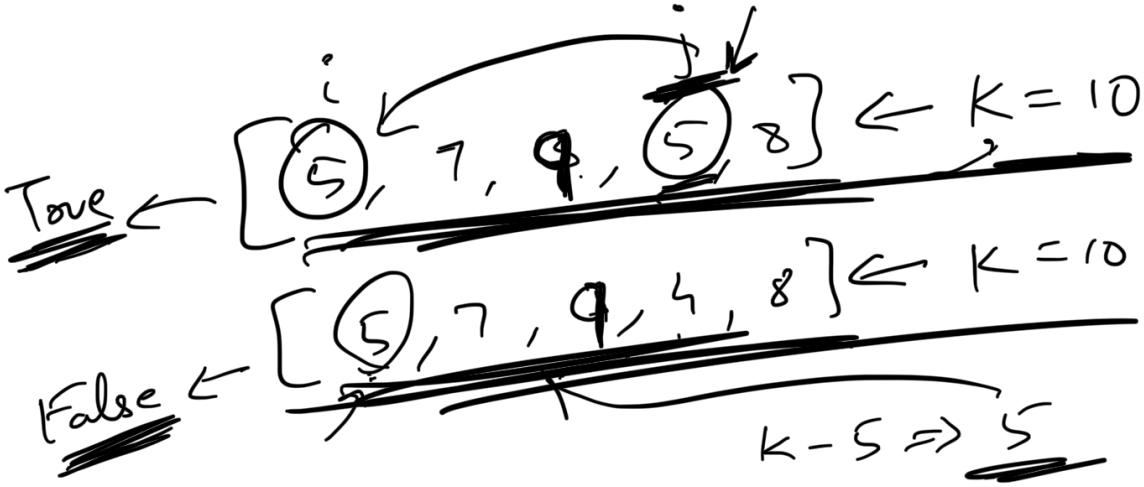
exist a pair a[i] & a[j] s.t

such that $a[i] + a[j] = K$ (K is g

eg $a = [7, 4, 10, 2, 5, 16, 3]$ & $K = 9$

7 + 2 = 9 → return. True

4, 5



① Brute → 2 loop

i = 0 → N-1

j = i+1 → N

→ T.C ⇒ $O(N^2)$

S.C ⇒ $O(1)$

5, 7, 9, 4, 8

② Sort & use B.S. ⇒ $O(N \log N)$ ✗

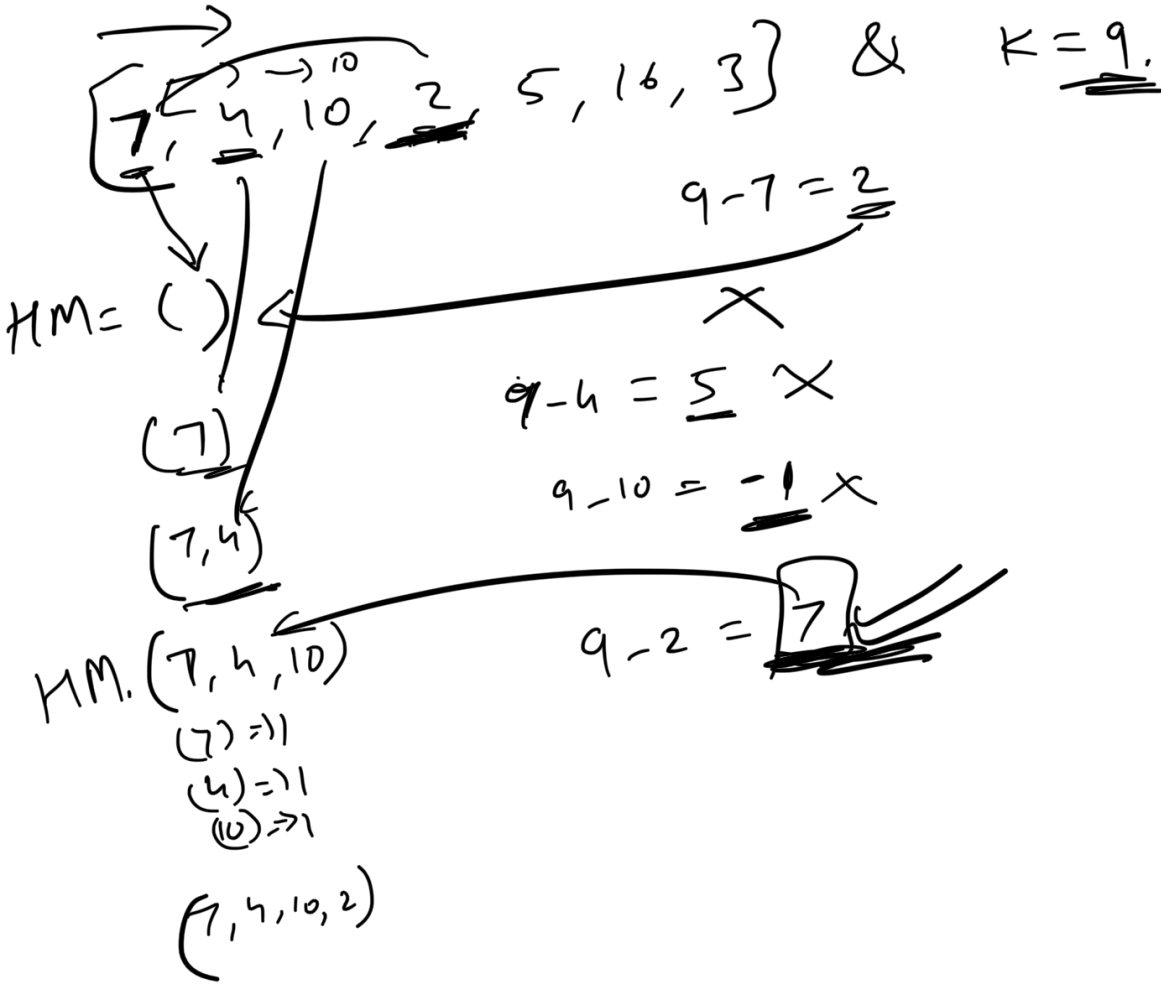
③ a[i] a[j] == K - a[i]

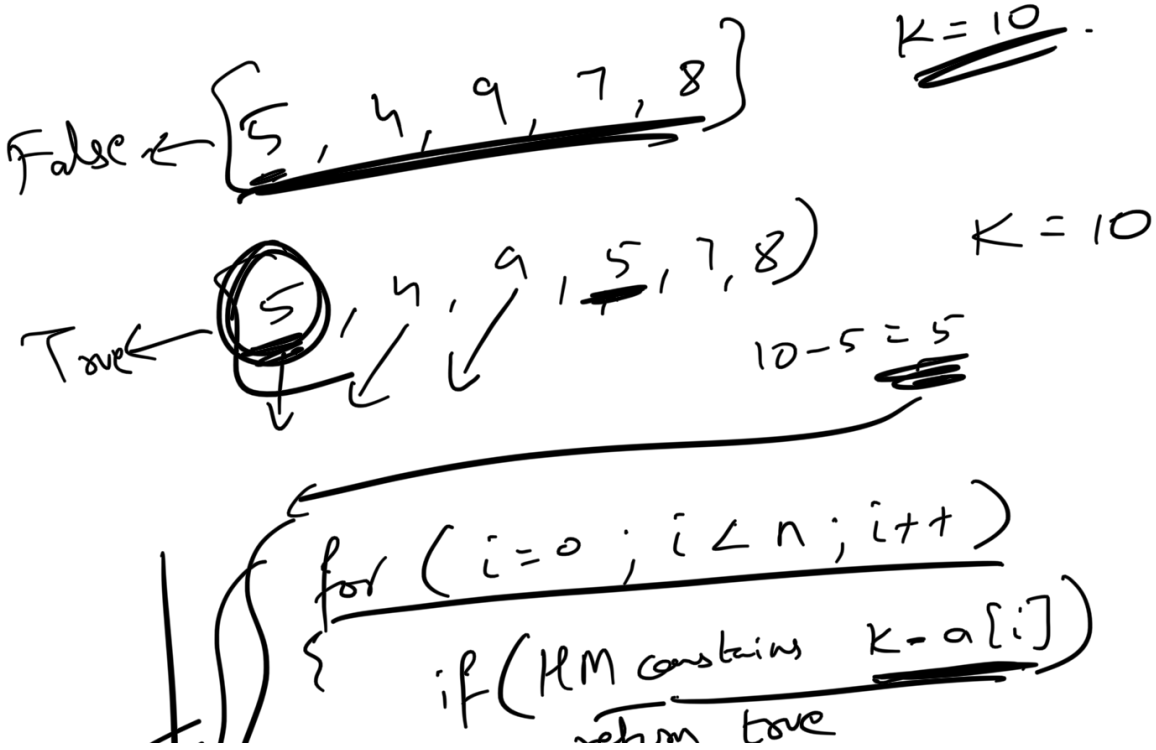
K

a[i] + a[i] = K

check if a[i] == K - a[i]

Use H.M, to check if is present & then add a[j]





else HM[a[i]]++;

}

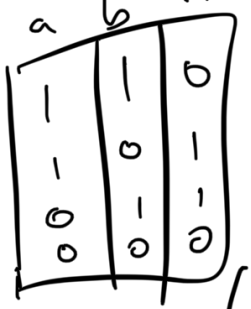
T.C ⇒ O(N)

S.C ⇒ O(N)

Q.5 Find if pair $A[i] \wedge A[j] = K$, $i \neq j$ exists.

[3, 1, 5, 9] K = 2

3 ∧ 1 = 2 → True



A[i] A[j] = ? ⇒ K ∧ A[

(3) 1 ∧ 2 = 3

3 ∧ 3 ⇒ 0

00 → 0 HM( )

N

10

11

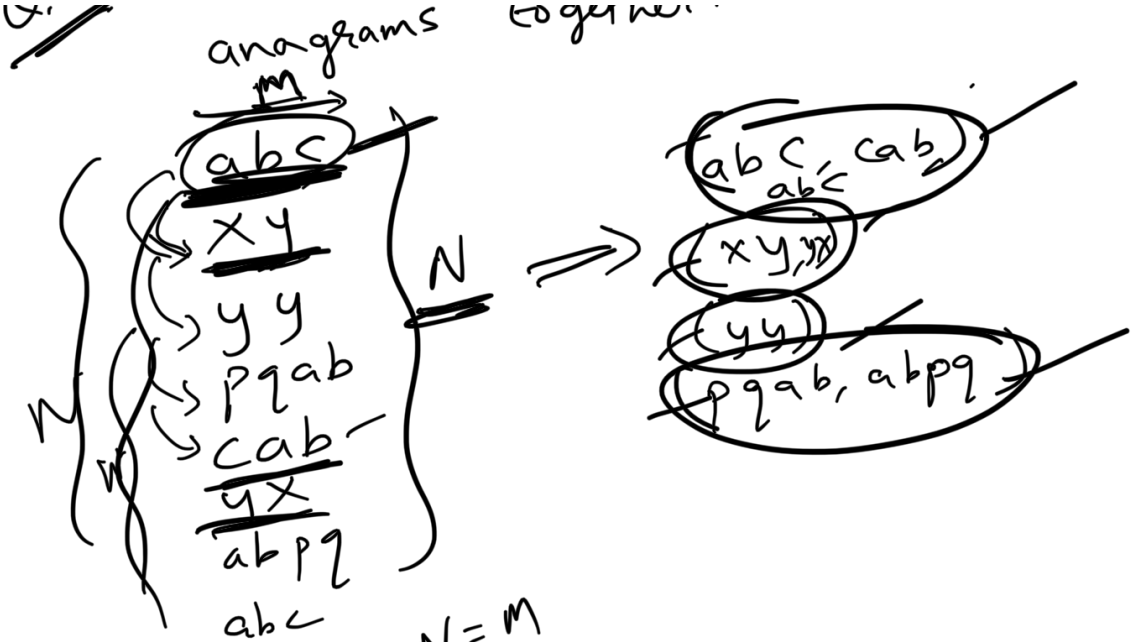
N

O(N)

5 → 10
3 → 0
5 ∧ 3 = 6 11

H.W.

Q.6 Given N strings, group all the



N = M

N * M

hashcode()

N = 10^5

length → 10^5 } 10^{10}