Hashing →

$$\begin{tikzcd}
 & & & \boxed{\text{Db}} & \\
\text{Day 1} \arrow[r] & \text{App} \arrow[r] & \text{Address/Phone} \arrow[r] \arrow[ur] & \text{Restaurants} \arrow[r] & \text{Mode of Pay} \arrow[ul] \arrow[d] \\
\text{Day 2} \arrow[r] & \text{App} \arrow[r] \arrow[d, Rightarrow] & \text{Address/Phone} \arrow[r] & \text{Restaurants} \arrow[r] & \text{Place Order} \\
 & \text{Internal Memory} & & & \text{ICICI Card} \arrow[d] \\
 & & & & \underline{\text{Place Order}}
\end{tikzcd}$$

Caching → Internally it uses Hashing

Hashing → Prestore the data somewhere so that we can use it later.

* <u>DSA</u> → Hashing

arr[] = [5, 6, 4, 4, 6, 5, 5]

num1 = 5
num2 = 4
num3 = 6
num4 = 7
} Queries

count = ~~0~~ ~~1~~ ~~2~~ 3

```
for (0 → n-1) {
    if (arr[i] == num1) {
        count++
    }
}
```

7 ⇒ N size of array

TC → O(N * Q) ⇒ (7 × 4) = 28

N → Size of Array ⇒ 10^5 }

Q → Queries ⇒ 10^5

TC → ~~C~~ $O(N \times Q) = 10^5 \times 10^5 = \boxed{10^{10}} = 10^{10}$

Leetcode ⇒ 1 sec = 10^8 operations

TLE → Time Limit Exceeded

Hashing → Precompute data

arr[] = [5, 6, ~~4~~, ~~4~~, ~~6~~, 5, 5, 5]

hash[] =

0	1	2	3	4	5	6	7
0	0	0	0	~~0~~ ~~1~~ 2	~~0~~ ~~1~~ ~~2~~ ~~3~~ 4	~~0~~ ~~1~~ 2	0

Max + 1

```
max = [illegible]
Operation → for ( 0 → n-1 )
                max ⇒ Math.max(max, arr[i])
            3                                   O(N)
        for ( 0 → n-1 ) {
```

3

	0	1	2	3	4	5	6	7
Hash[] →	0	0	0	0	2	4	2	0

```
Query → num1 ⇒ 5      ⇒ 4      O(1)
        num2 ⇒ 6      ⇒ 2
        num3 ⇒ 4      ⇒ 2
        num4 ⇒ 7      ⇒ 0
```

```
max = MIN_VALUE
for(0 -> n-1) {
    max = Math.max(max, arr[i])
}
hash[] = [      ]   // max size
for (0 -> n-1) {
    hash[arr[i]] = hash[arr[i]]+1
}
let Queries = 4
while (Querys--) {
    res[] = hash [query]
```

TC -> O(N) + O(Q) + O(N)

SC -> O (maxmum of my array)

Queries $\rightarrow 10^5$ SC $\rightarrow 10^5$

N $\rightarrow 10^5$

TC $\rightarrow 10^5 \times 3 \Rightarrow 10^5$

Number Hashing

Array $\rightarrow \boxed{10^9} \Rightarrow$ Hash $\rightarrow 10^9$

Max value

Javascript

$2^{32}-1$

$\boxed{10^9}$

$\downarrow$

10[illegible]

Java $\rightarrow 10^6$ Inside Main Method

10^7 Outside Main Method

10^8 Boolean $\rightarrow$ 108

Character Hashing →

hashing

ASCII Value → 256

Only Smallcase

a → 97

b → 98

c → 99

⋮

z → $\boxed{122}$

$\boxed{123}$

hash[ ] = [0 — . -- 123]

hash[123

String

abaca

Queries

a

b

c

d

TC → O(N × Q)

0 1 2 3 -- ----- 97 98 ------ 122

```
for (0 → n-1) {
    hash[arr[i]]++
```

'a'

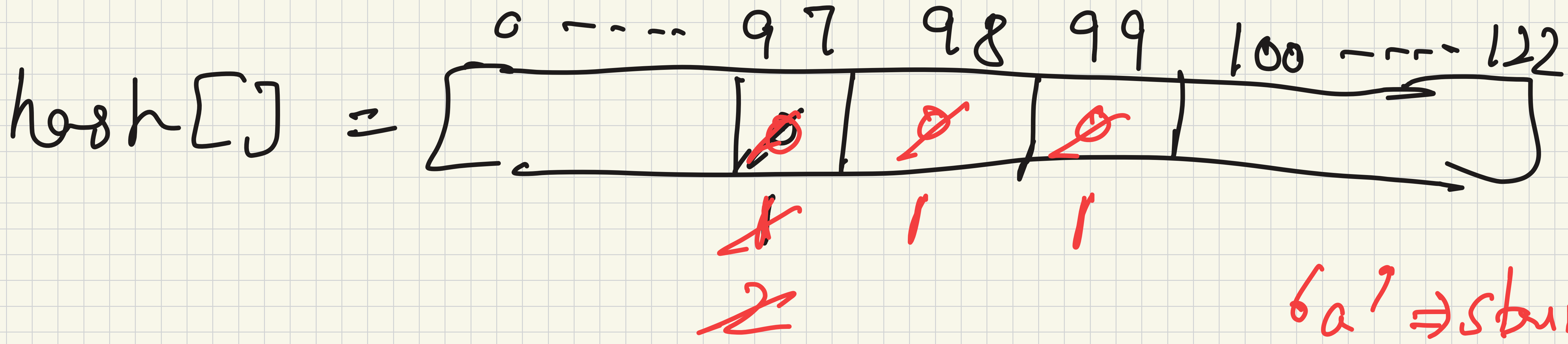
3

hash[arr[i].charCodeAt(0)]

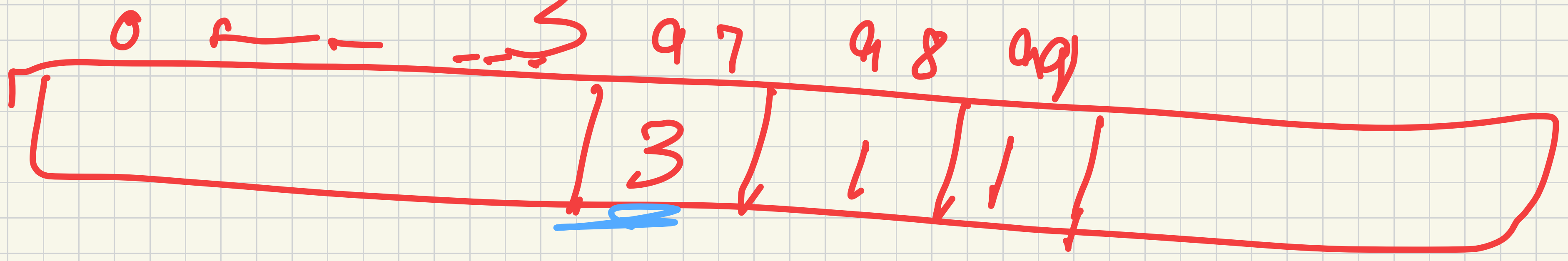
hash['a']

hash[97]

abaca



'a' ⇒ string



Queries → 4

'a' → 97

'e' → 6a ?

```
while (queries--){
    return hash[query]
}
```

$\boxed{3}$

$\boxed{97}$ 6a ?

3

TC → O(N) + O(Q)

SC → O(123) // Small lowercase

A → Z

65 $\boxed{86}$

256 → O(256) SC → O(256)

$a \rightarrow z$

hash[123]

$a \rightarrow 97$	$a \rightarrow 0$
$b \rightarrow 98$	$b \rightarrow 1$
$c \rightarrow 99$	$c \rightarrow 2$
$d \rightarrow 100$	$d = 3$

hash['a' - 97
- 'a']

98 - 'a'

99 - 'a'

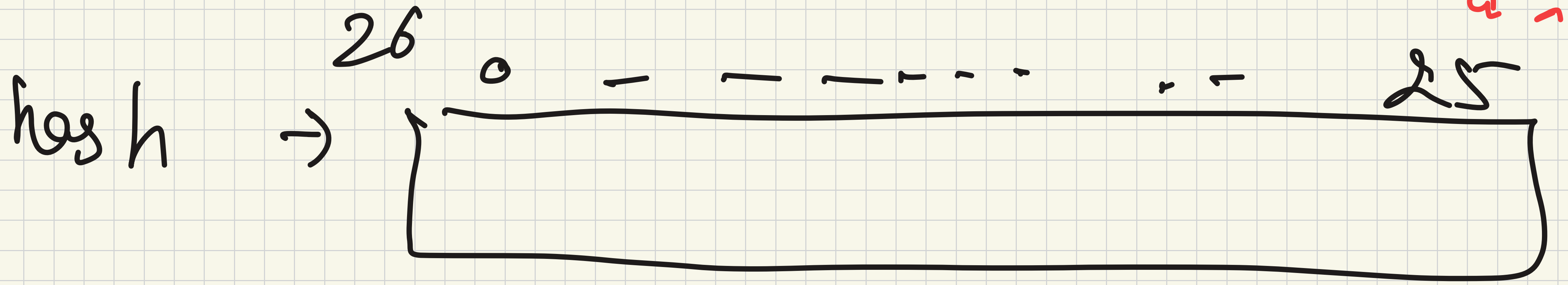
$d = 3$ — - - - $z \rightarrow 25$

```
for( 0 -> n-1 ){
    hash[arr[i] - 'a']++
}
```

Javascript →

hash[arr[i].charCodeAt − 97]

'a'.charCodeAt(0)



*

$> 10^6$

C++	Java	Javascript
Map	HashMap	Map
STL & Collections		
Unordered Map	TreeMap	

Instead of hash[ ], we will use hashmap.

new Map() → map.get( )

Map → O(1) → Best Average

O(N) → Worst

very very Rare

N is no of elements in Map

Internally → Division Method

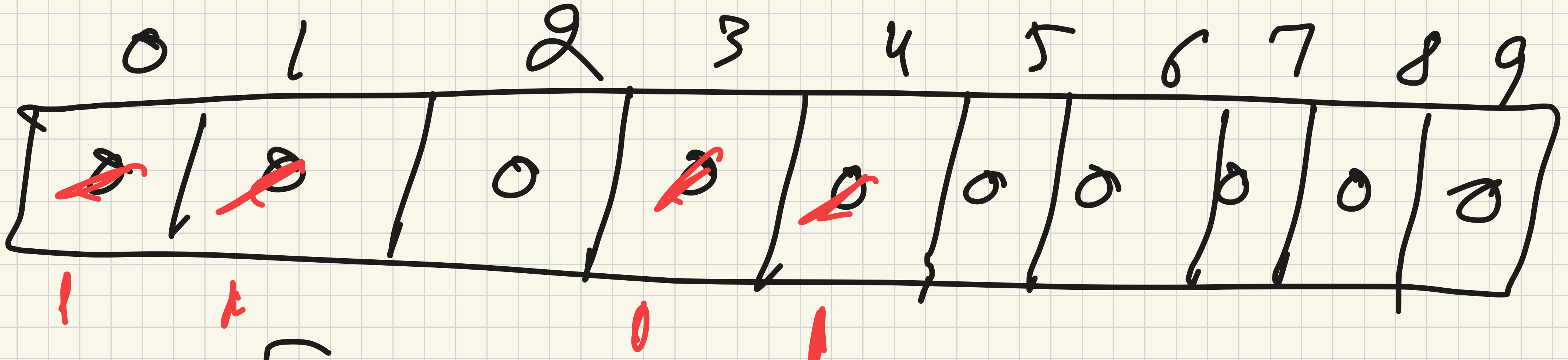
Division Method

↓

10^9 hash[10^9]

indexes → 0 → 10^9

hash[10] ⇒



arr = [21, 25, 36, 52]

Modulo → Largest Prime Number → 7

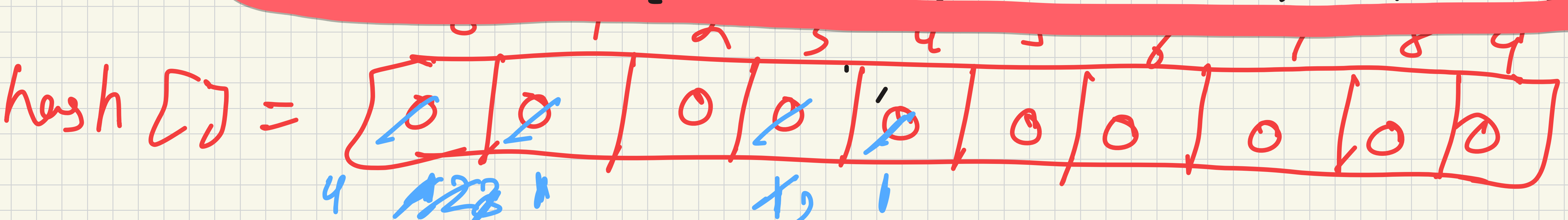
21 % 7 ⇒ 0

25 % 7 ⇒ 4

36 % 7 = 1

52 % 7 ⇒ 3

arr = [21, 25, 36, 52, 14, 35, 7, 38]



21 % 7 = 0

25 % 7 = 4

36 % 7 = 1

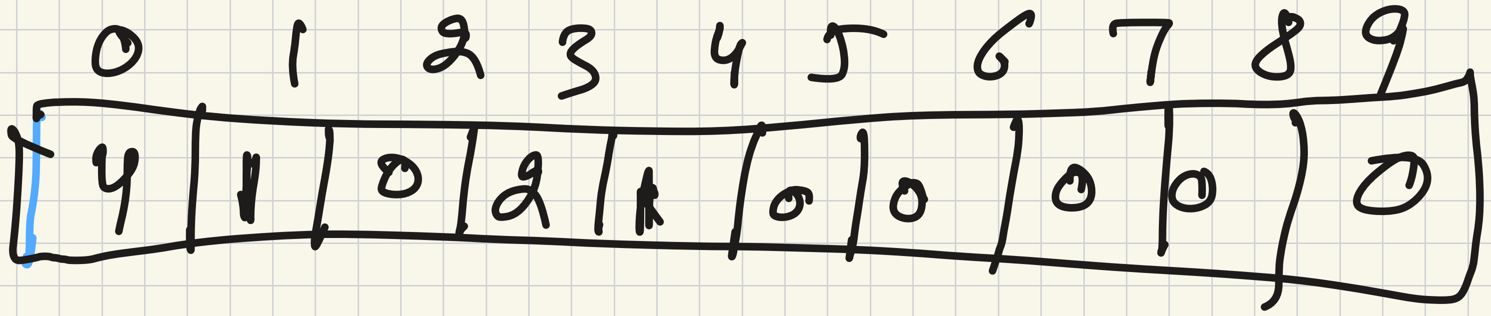
52 % 7 = 3

14 % 7 = 0

35 % 7 = 0

7 % 7 = 0

38 % 7 = 3



Queries = 21 % 7 → 0 → 4

~~Count~~ 4 ←

[21, 25, 36, 52, 14, 35, 7, 38]

$21 \% 7 = 0$

$25 \% 7 = \boxed{4}$

$36 \% 7 = 1$

$52 \% 7 = 3$

$14 \% 7 = 0$

$35 \% 7 = 0$

$7 \% 7 = 0$

$88 \% 7 = 3$

$7 \rightarrow 14$

$0 \rightarrow 21 \rightarrow 35 \rightarrow \boxed{7 \rightarrow 14 \rightarrow 21 \rightarrow 35}$

$1 \rightarrow 36$

$2 \quad 38$

$3 \rightarrow 52 \qquad \rightarrow 38 \rightarrow 52$

$4 \rightarrow 25$

5

6

7

8

9

Queries $\rightarrow 21 \% 7 \rightarrow 0$