Hashmaps → D.S

→ what → ?

↓

D.S

↓

why → ?

→ Dev

↳ T.C → =

Ins/Del/Search → O(1)

↳ operation

Quest 1:

lower case 'a' – 'z'

↓

i/p

string str = "abcbdeacf"

maximum occurring character → ?

o/p → answer = 'a'

a → 3

b → 2

c → 1

d → 1

c -> 1

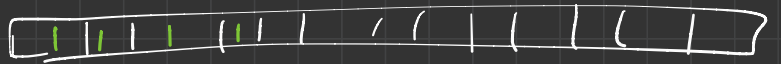
f -> 2

algo:- a → count

For(a->2) → answ → a, 3

max →

algo:-



size = 26

count

'a' 'b' 'z'

str = "abcd"

```
For( int i=0 → str.size() )
{   char ch = str[i];
    count [ch - 'a'] ++;
```

'a' -

'a' - 'a' → 0

'b' - 'a' → 1

}

'z' - 'a' = 25

for (int i=0 ; 25)

↳ max ⟶

'a' → 0

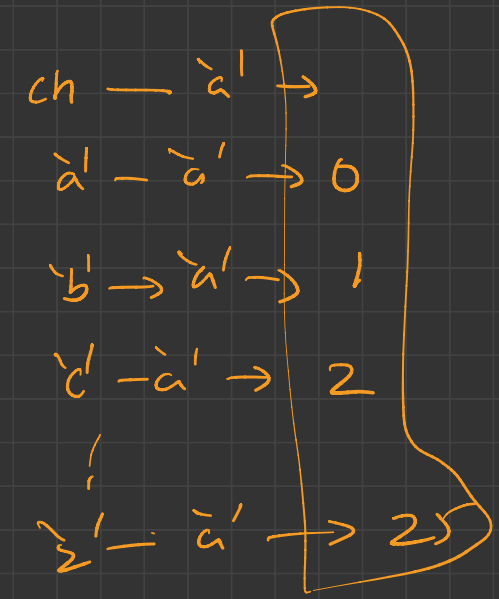
'b' → 1

'c' → 2

'd' → 3

⋮

'z' → 25



str → " mera naam hai babbar mera babbar babbar ";

o/p → maximum occurring word

mera → 2

naam → 1

hai → 1

babbar → 3

answer

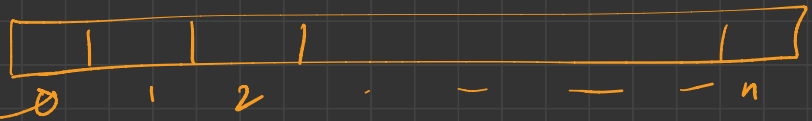
mera → 0

naam → 1

hai → 2

babbar → 3

→ naively possible



<mera, 2>

<naam, 1>

<babbar, 3>

class node

{

}

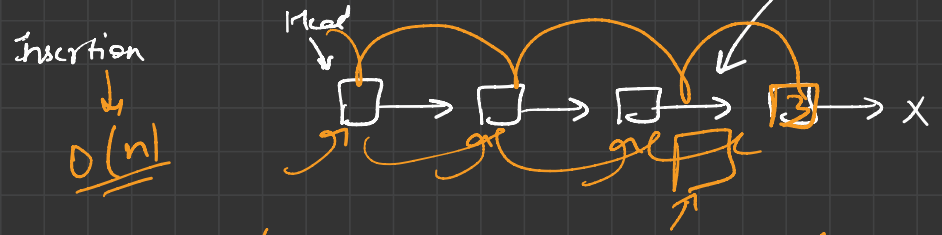
map <node, int> m;

STL

map

Implementation:

↳ Using Linked List



Deletion → O(n) , Searching → O(n)

↳ Using BST → Ordered_map

Insertion → O(log n)
Search → ~~
Delete → O

→ O(log n)

O(1) α

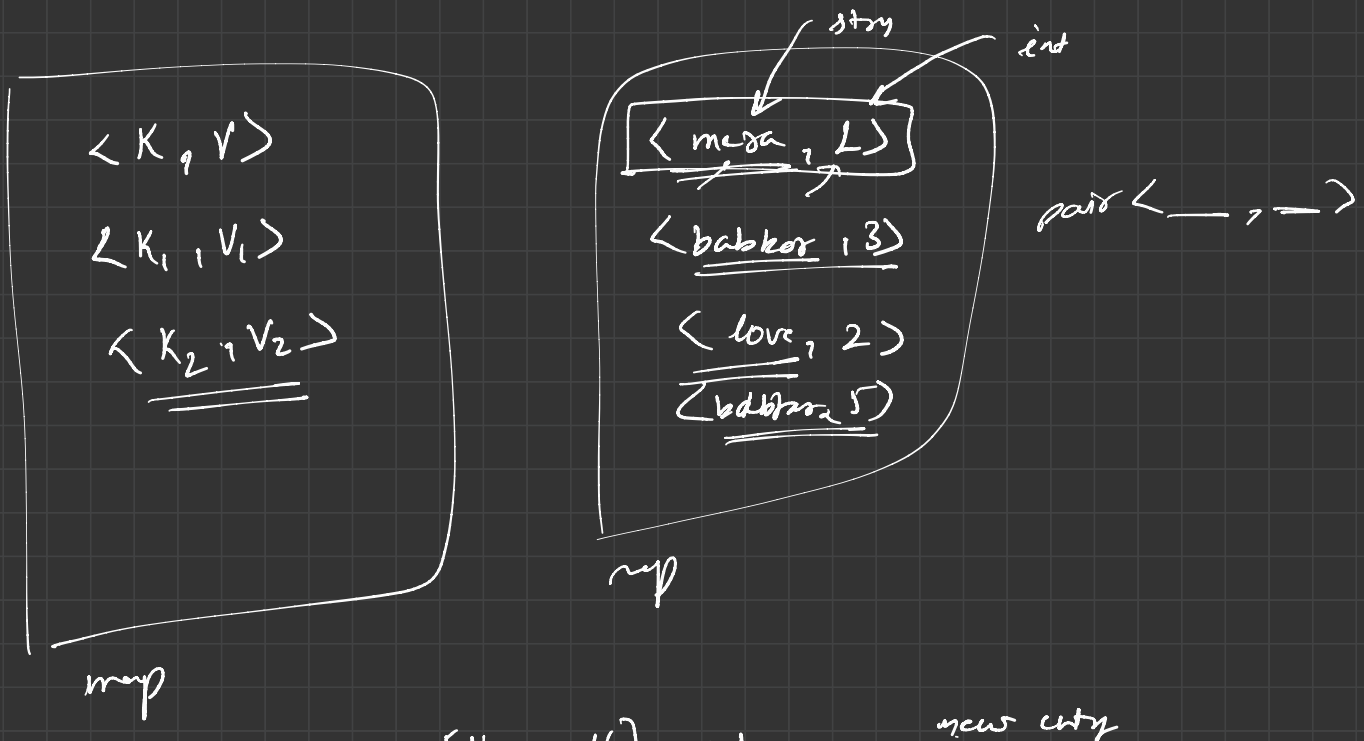
Hashtable

I/O/S → O(1)

Inbuilt stuff

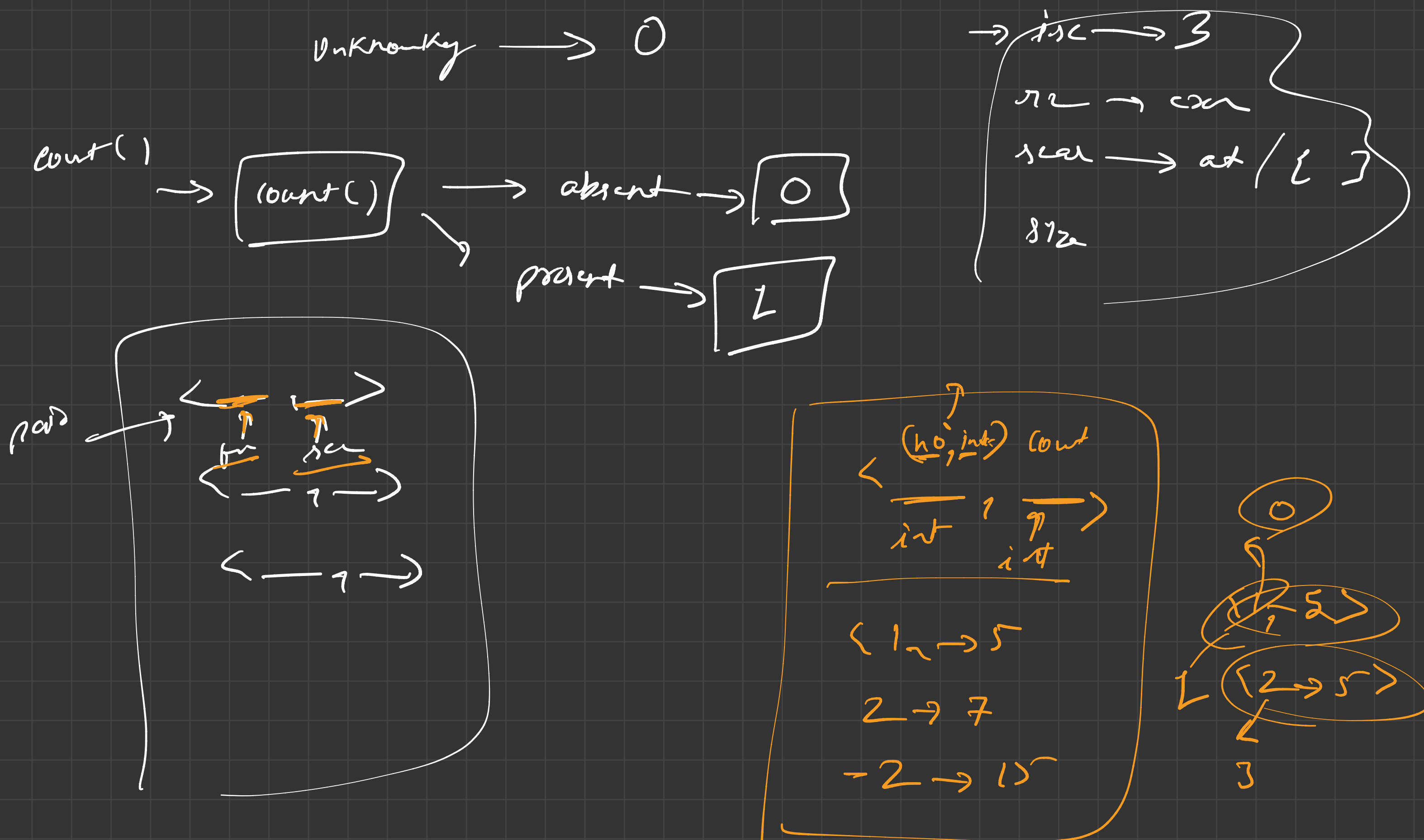
→ map → O(log n)

→ unorder_map → O(1)

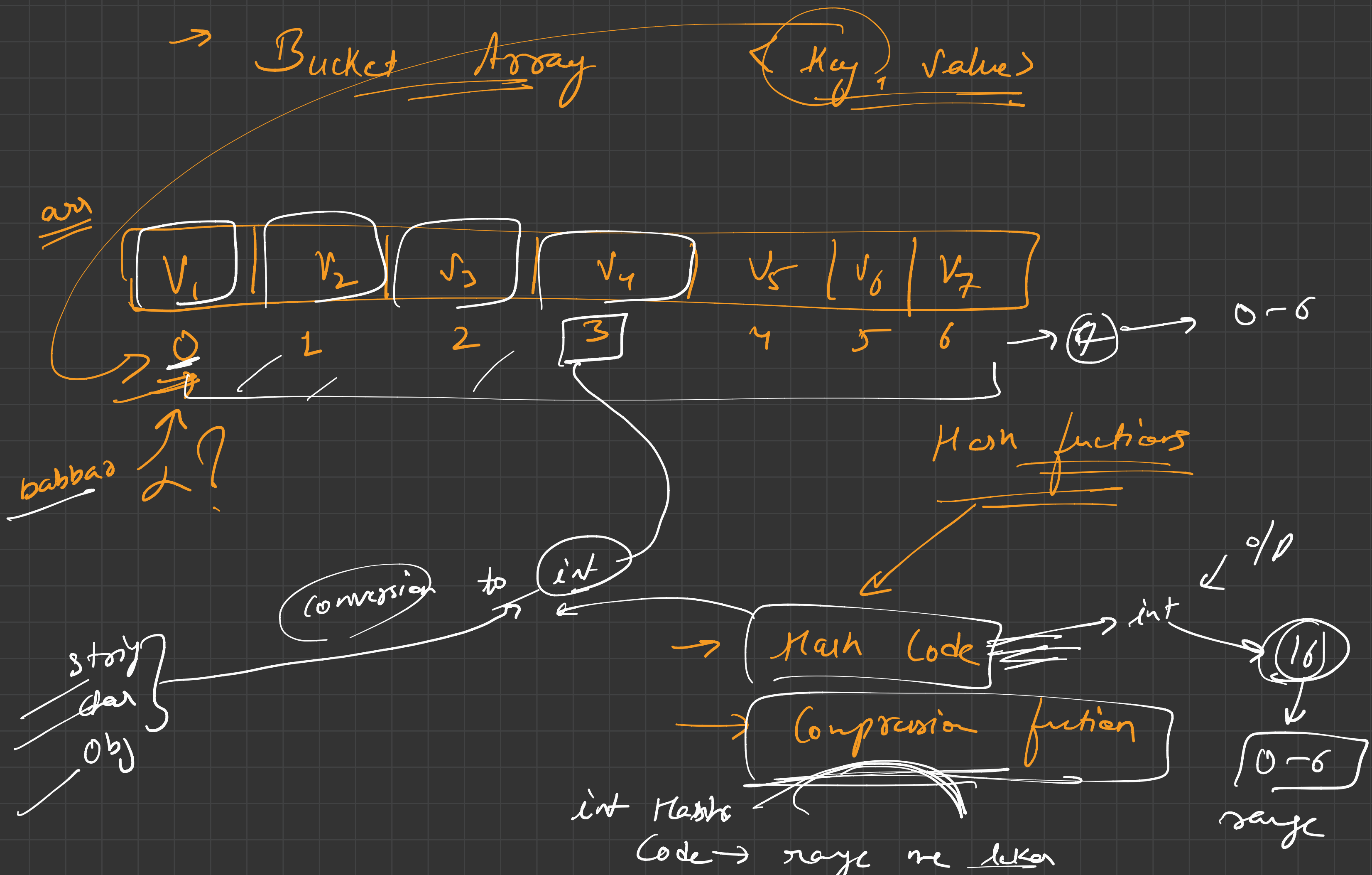


m["mera"] = 1 → new entry <"mera", ~~1~~ 2>

m["mera"] = 2



UnknownKey → 0
count()
count()
absent → 0
present → 1
size
int
int
count
1 → 5
2 → 7
-2 → 15
3


Bucket Array
<Key, Value>
arr
V1 V2 V3 V4 V5 V6 V7
0 1 2 3 4 5 6
7
0-6
babbar
Hash functions
conversion to int
string
char
obj
Hash Code
int
O/P
16
Compression function
0-6
range
int Hashcode
Code → range me lekar

25

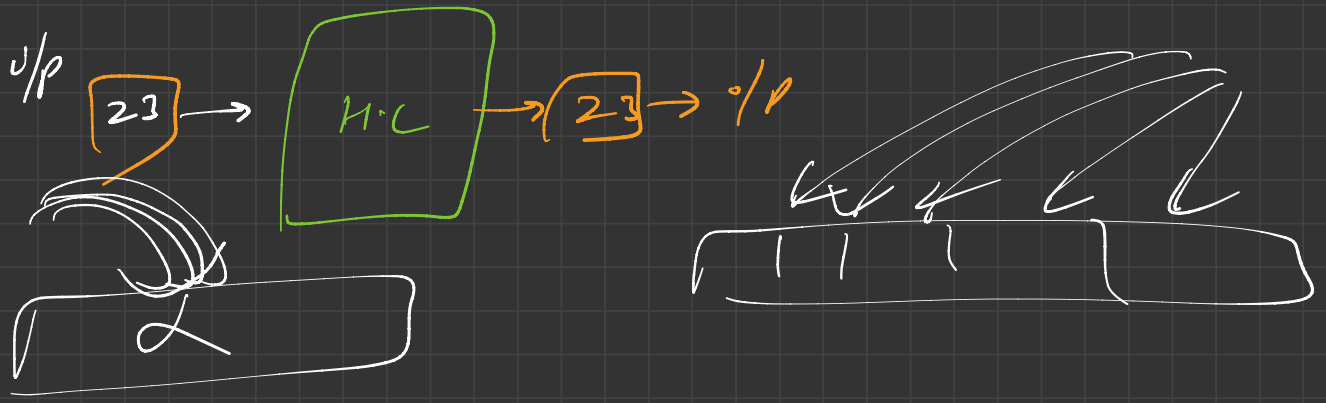
% ≠

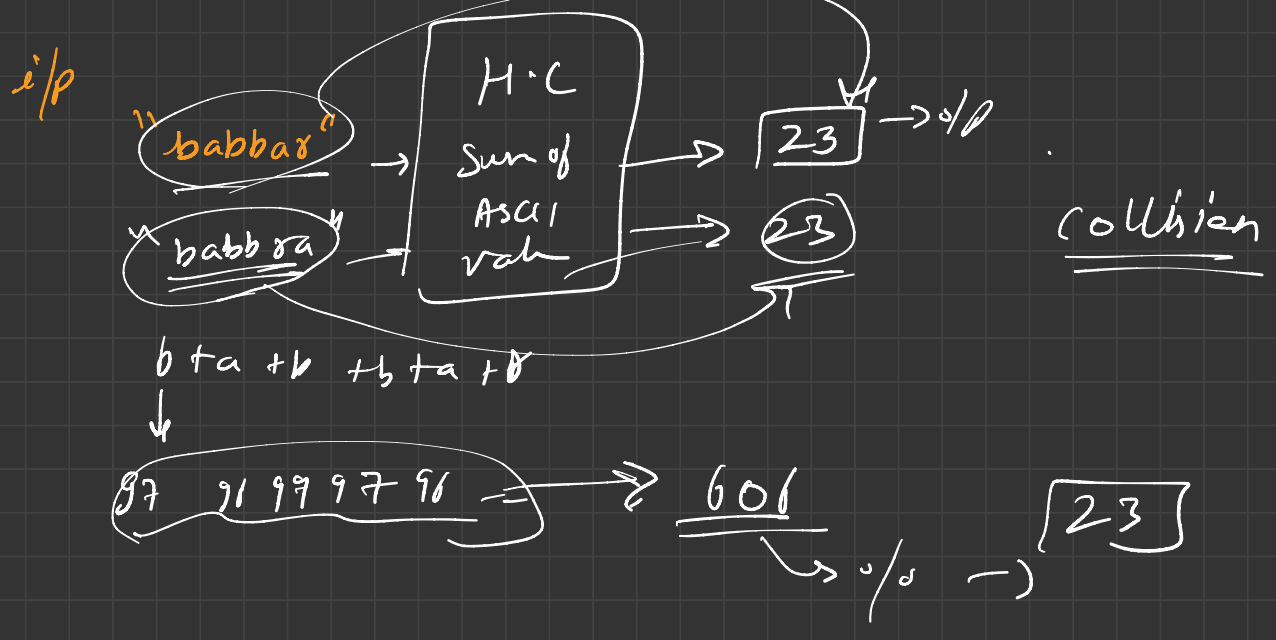
% cross size → 0 – 6 → Yes

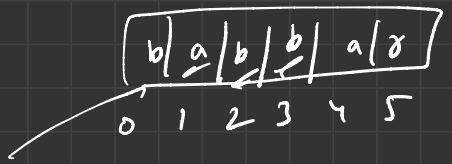
→ Hash Code → why → Conversion to int

identity fn

uniform distribution



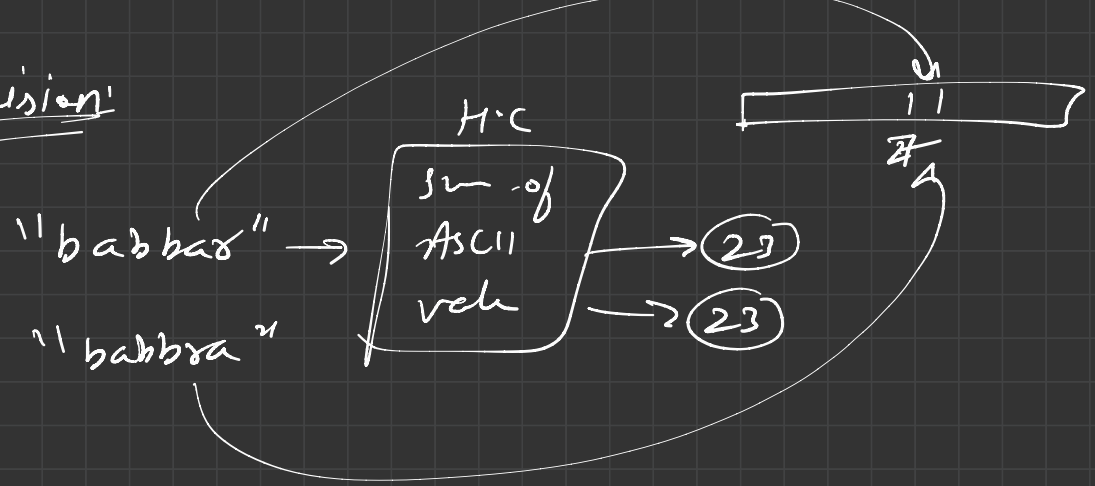




$(b \times 0) + (a \times 1) + (b \times 2) + (b \times 3) + (a \times 4) + (r \times 5)$

$= 1283 \rightarrow \%\ N \rightarrow 22$

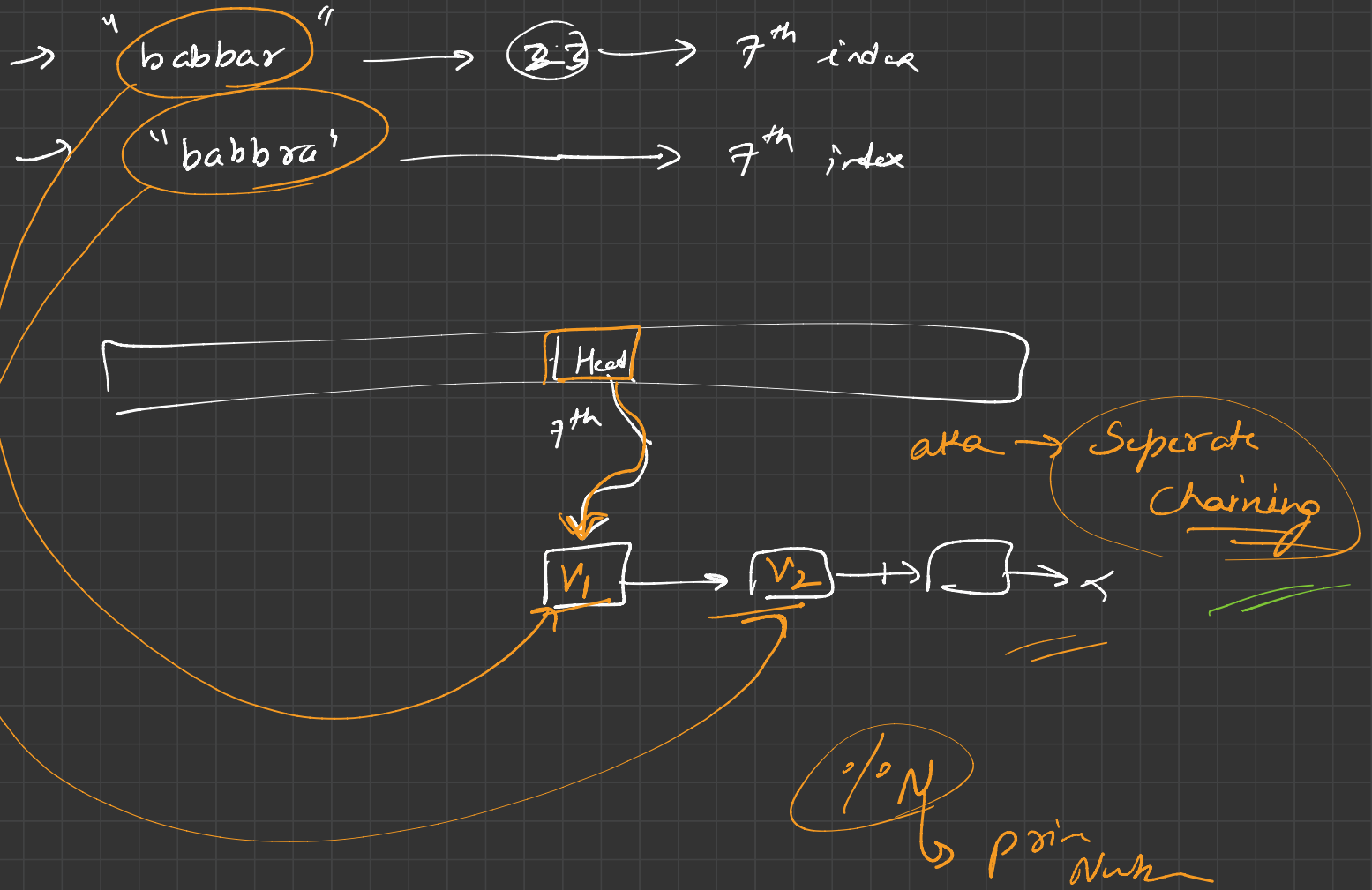
→ Collision:



Collision Handling :-

→ Open Hashing → same place pr hi jaa

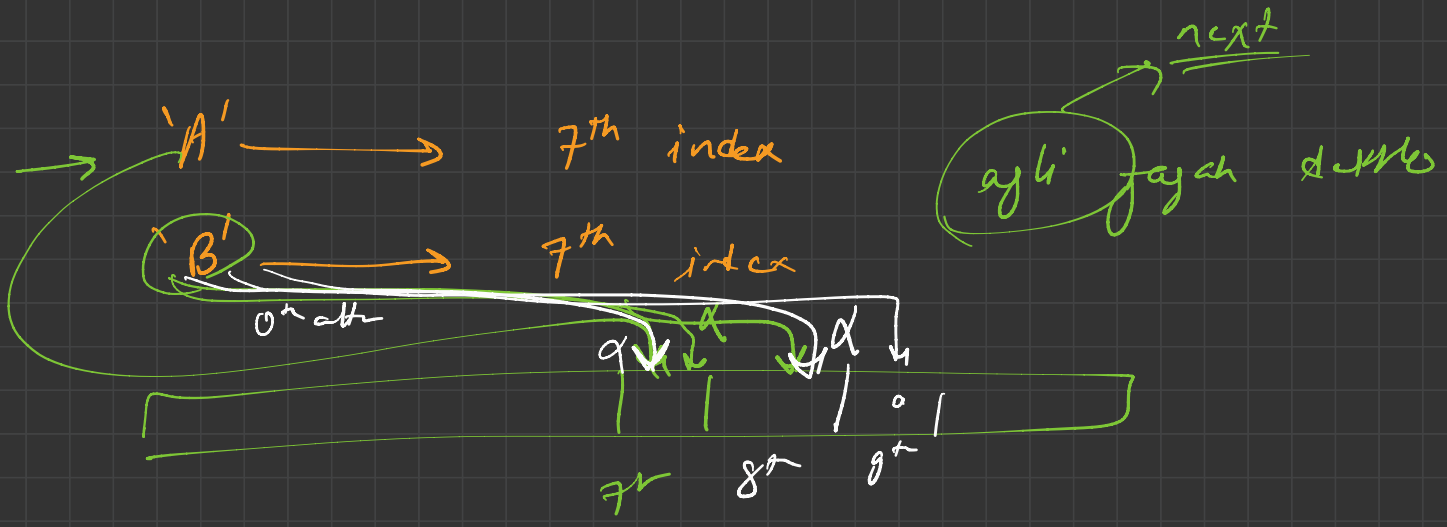
→ Closed Addressing


"babbar"
23
7th index
"babbra"
7th index
Head
7th
aka → Seperate Chaining
V1
V2
%N
Prime Num

100 word → index

→ 100 LL α

→ Open Addressing



7^{th} $+1$

→ $H_i(a) = h(a) + f_i(a)$

i^{th} attempt me kaha pr jaye

→ Linear probing

$f(i) \rightarrow i$

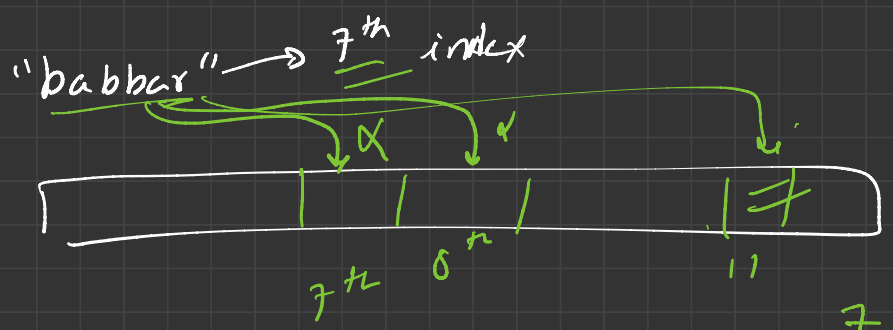
$F(1) \rightarrow 1$

$F(2) \rightarrow 2$

→ Quadratic Probing

$F(i) \rightarrow i^2$

$F(0) \rightarrow 0$



$F(1) \rightarrow (1)^2 \rightarrow 1$

$F(2) = 2^2 \rightarrow 4$

$h_i(a) = \boxed{h(a)} + \boxed{f_i(a)}$ (7 + 1) $= 8$

$= 7 + 4 = 11$

S.C

$i^3 + 2$

$i^7 + 13$

$i^2 + 9$

=> Complexity analysis

" mirza bhai love babbar bbb kya had "

Function

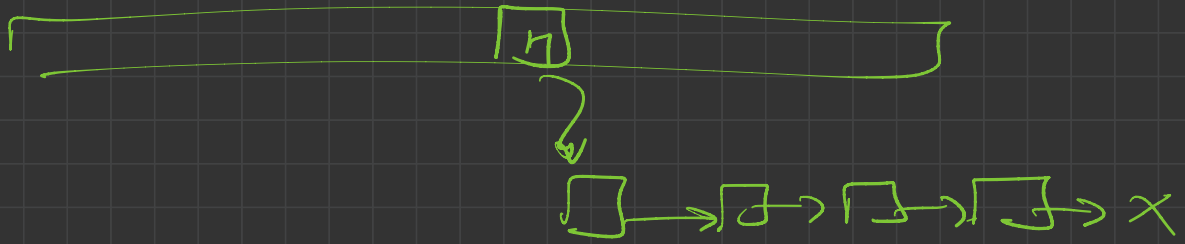
n → total no of word

k → word length

if n >> k

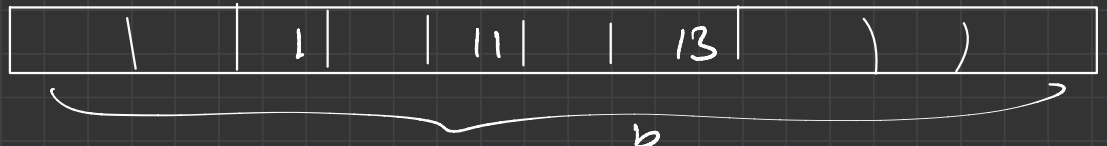
H.C → O(k) → ≈ O(1)

→ traverse → O(n)



n → no of entries in map

b → no of boxes available n = 3



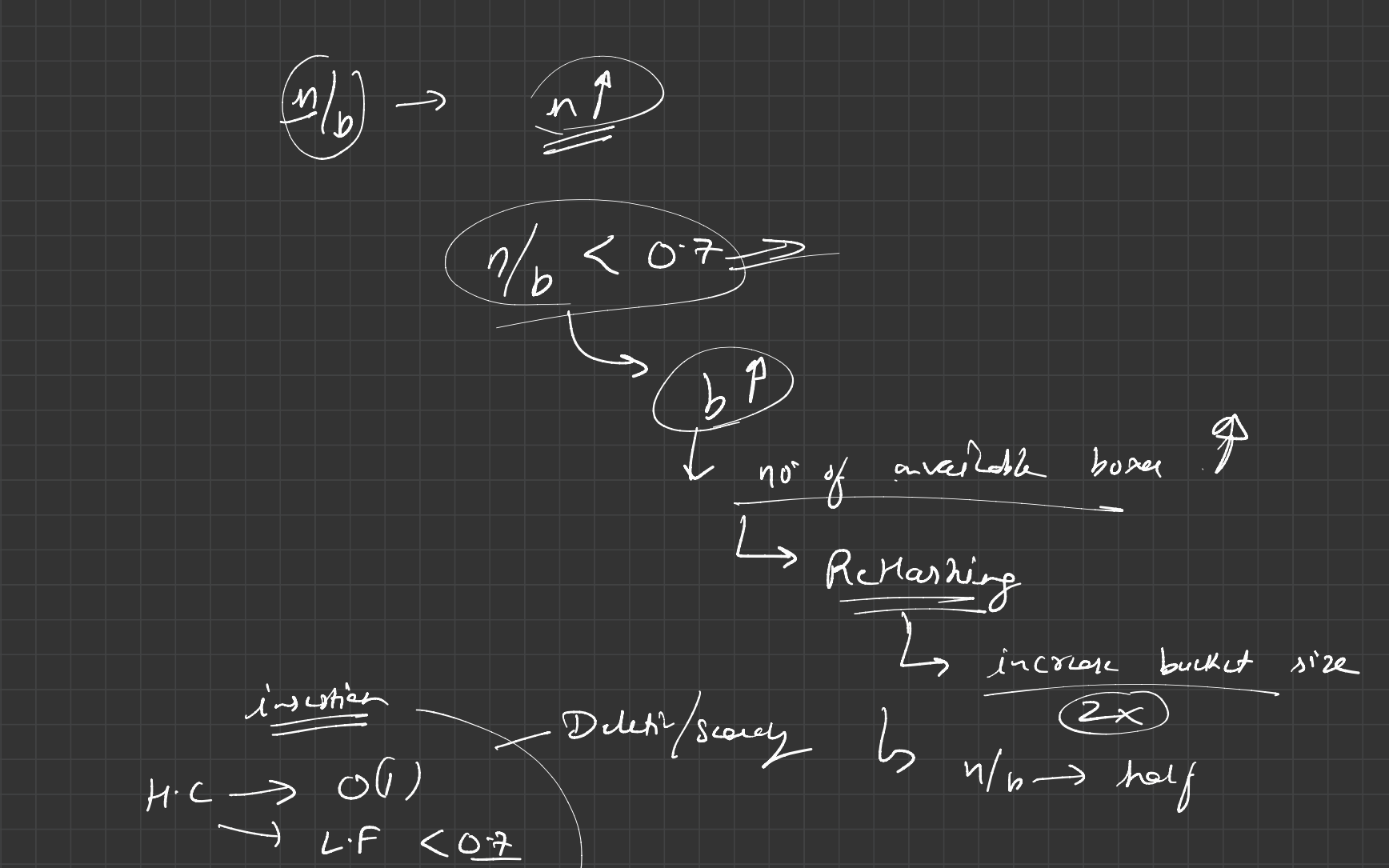
no of entry in a box → n/b → Load factor

we always ensure n/b < 0.7

→ H.C → acha → O(1)

L.F ≥ n/b < 0.7

it is very safe to use → O(1)


n/b → n↑
n/b < 0.7
b↑
no of available boxes ↑
Rehashing
increase bucket size
(2x)
n/b → half
insertion
Delete/search
H.C → O(1)
L.F < 0.7

↳ Rehash

↳ saari entries dubara hashing karo.

Maps → what?

Maps → why?

STL

O(1) → I, D, S

H/W

Impl → LL → O(n)

Impl → BST → O(log n) → map

Impl → Bucket array → } → O(1) → I/D/S

Hash f → H.C

Hash f → C.F

H.C → OH, CA } → n/b → L.F < 0.7