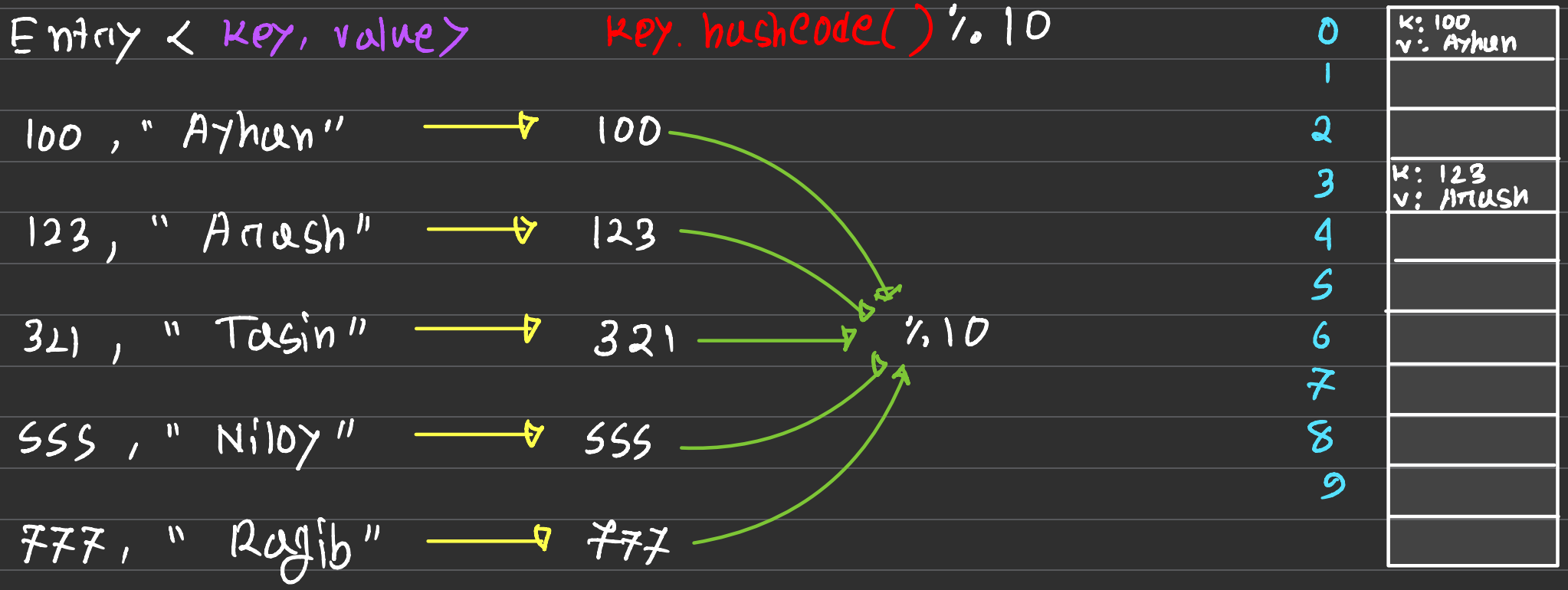

Entry < key, value >
key.hashCode() % 10
100 , " Ayhan"
123 , " Anash"
321 , " Tasin"
555 , " Niloy"
777 , " Rajib"
100
123
321
555
777
% 10
0
1
2
3
4
5
6
7
8
9
K: 100
V: Ayhan
K: 123
V: Anash

Syntax:

```
import java.util.*;

public class main{

    Public static void main(string args[]){

        Hashtable<Integer, String> table = new Hashtable<>();

        table.Put(100, "Ayhan");

        table.Put(123, "Anash");

        table.Put(321, "Tasin");
```

Paint the value syntax:

```
table.get(100);
```
⟶ Ayhan

Display all key value Pairs:

```
for (Integer key : table.keySet())
    System.out.Println( key + "\t" + table.get(key));
```

to see the hashcode

```
key.hashCode()
```

to see the index value

```
key.hashCode() %10
```

remove:

```
table.remove(key);
```

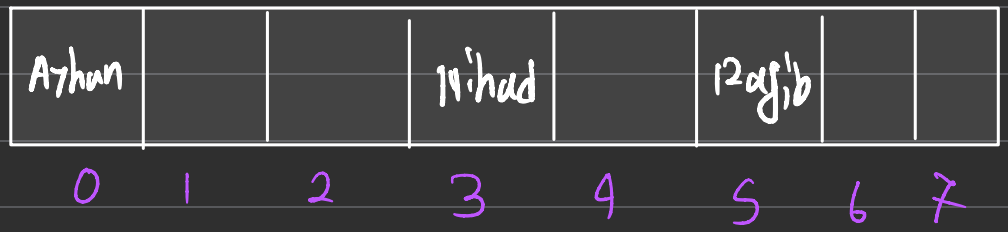
Benefits:

(i) A data structure stored unique keys to values ex. Integers, strings

(ii) Each insertion, look up, deletion of key/value pairs

(iii) No ideal for small data sets, great with large data sets

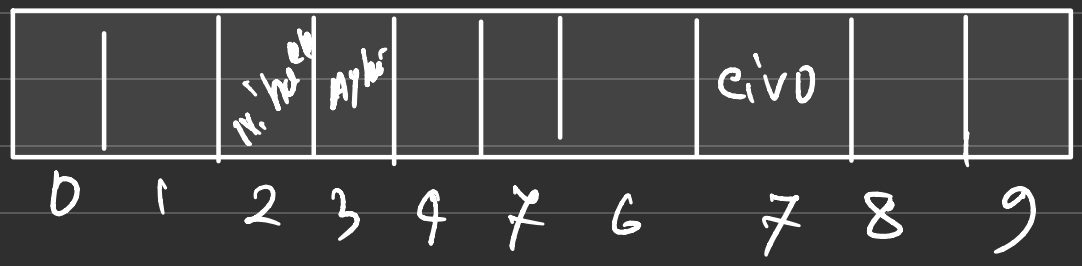
Hash Table

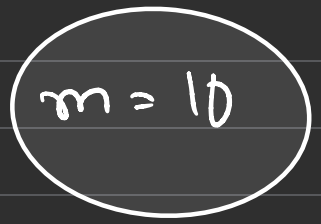


if Nihad exist ? ⟶ Yes

How? Hashcode problem ① we need all elements as positive numbers
we use hashcode function

② Hash code is very large ⟶ we need to reduce it.
↓
this is known as hashing





"civo" ⟶ hash("civo") = 7

hash("Ayhan") = 3

hash("Ninad") = 2

but ⟶ hash("civo") = 7349855 % 10 = 5

hash("Ayhan") = 3813497 % 10 = 7

hash("Ninad") = 2348737 % 10 = 3

let's Jump into new Problem:

For Example:

if hash("Farchan") = F9321 % 10 = 3 ← index

if hash("Rugib") = 62923 % 10 = 3 ← index

is it collision?

Yes!

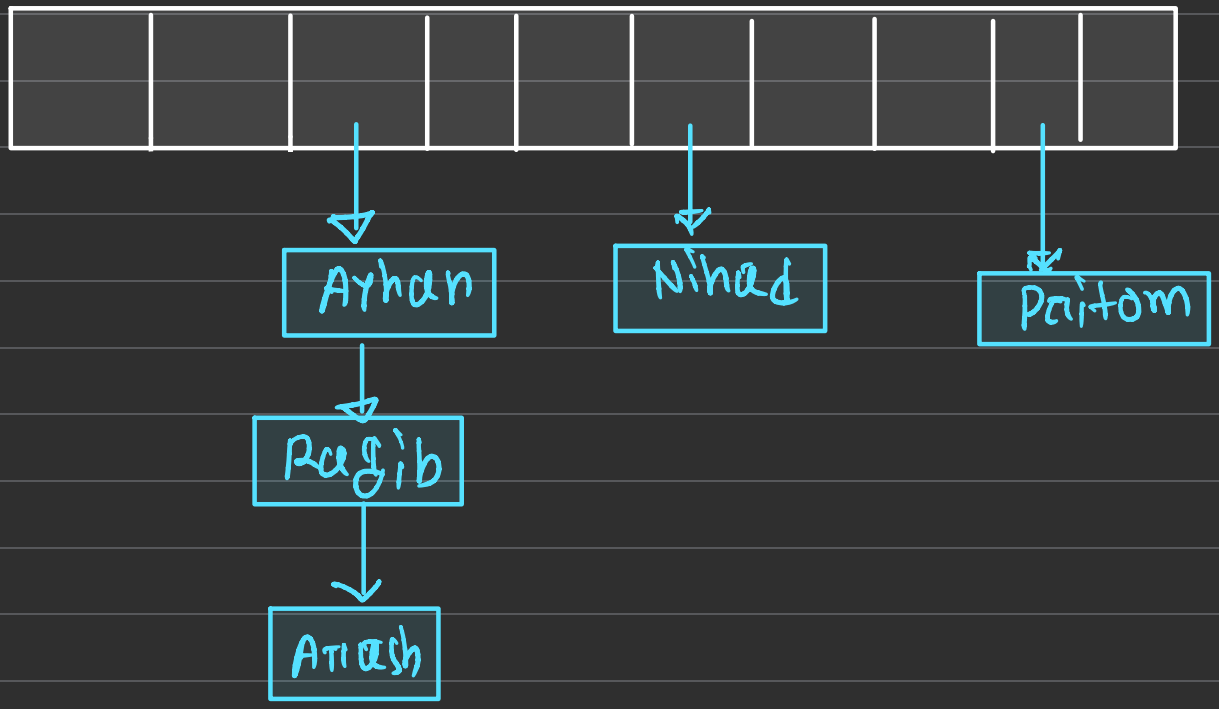
How to get rid from this collision?

Two way —

① chaining

② open addressing

Chaining



simple uniform hashing

Assumption:

n = Number of keys in table

m = size of table

load factor = $\alpha = \frac{n}{m}$ = expected key per slot.

time complexity $O(1+\alpha)$

/

for linked list