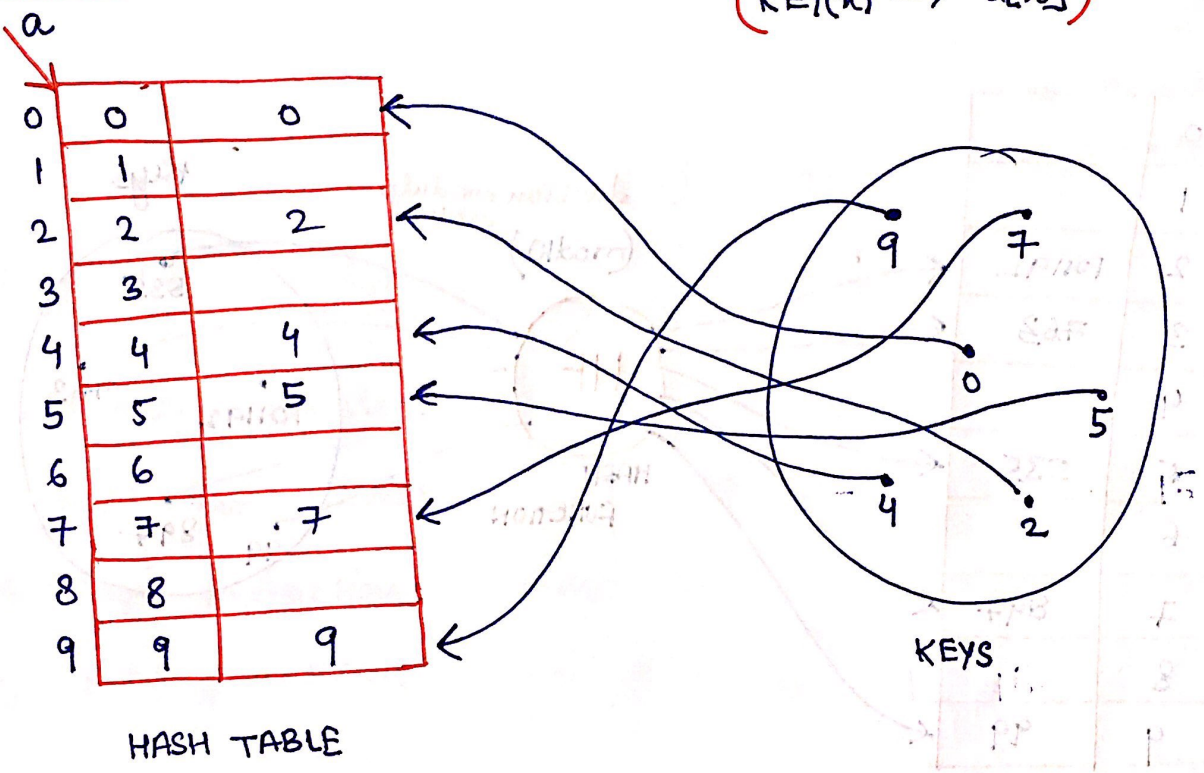
⇉ Hashing :-

It is one of the searching techniqne, where $T(n) = O(1)$ i.e const. in every case.

Direct address table :-

(KEY(x) → a[x])



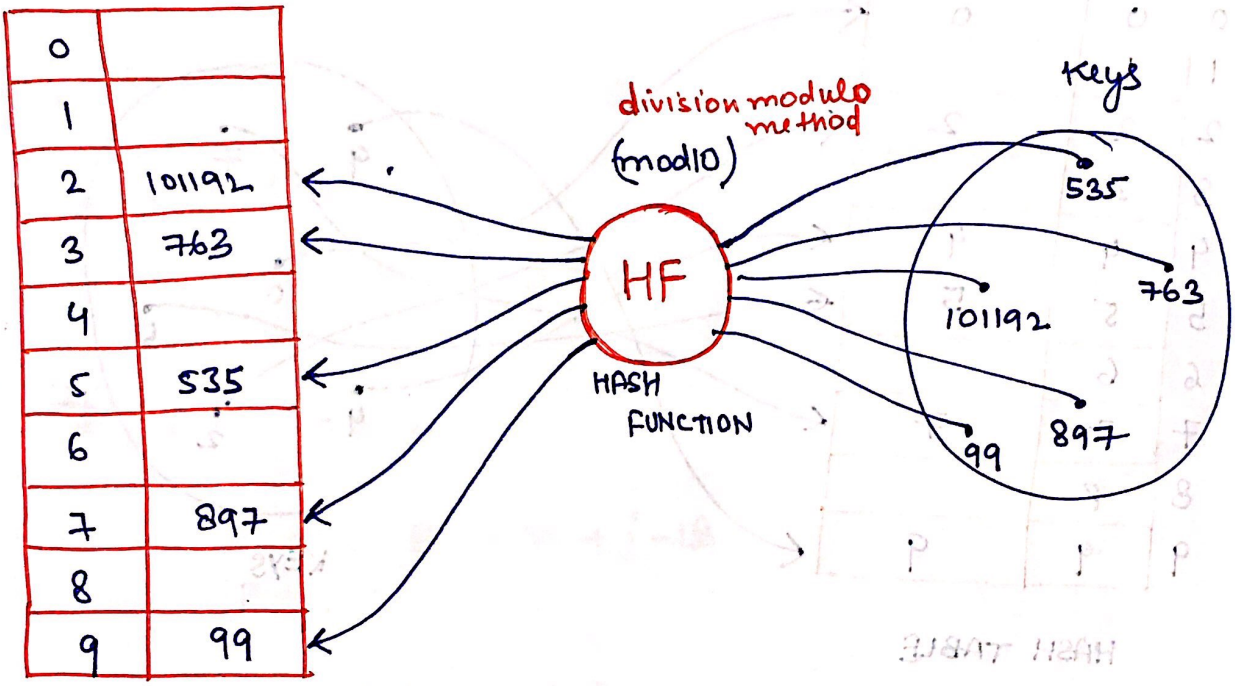
- According to direct address table, key itself is the address.
- Searching time $T(n) = O(1)$ (every case).
- Drawback :- Large space is required if value of key is large eg:

one of Key = 1,00,000 then Hash Table size ≥ 1,00,000

To eliminate this problem, we use Hash function

Hash function :- It is mapping techniqne

- I/p : Key value
- O/p : Hash Table Index value



Key = 897

then HF = 897 mod 10

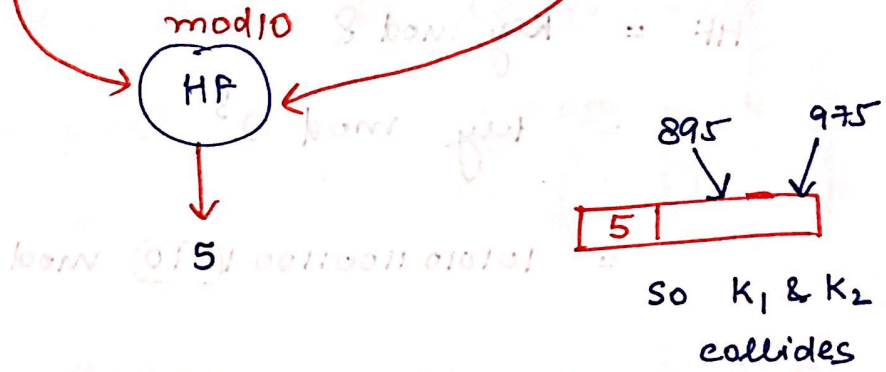
= 7

(Index = 7)

- Space problem is solved, as it compress large value of key to a small index value.

Drawback :- It suffers collision (more than one key fights for same position)

eg. $K_1 = 895$ and $K_2 = 975$



So K_1 & K_2 collides

⇒ Types of Hash function :-

(i) Division modulo Method

(ii) Mid Square method

(iii) Digit Extraction method

(iv) Folding method

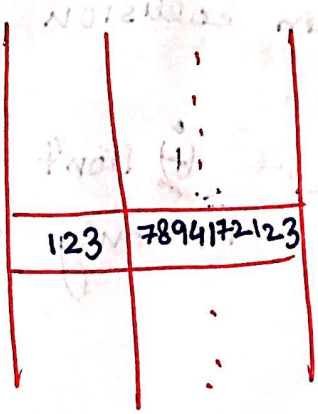
(i) Division modulo method :- [OK]

eg:- $M = 1000\ (0 - 999)$ size of table

Let Key = 7894172123

HF(Key) = Key mod 1000

= 123



eg:- $M = 8$ (0–7) table size

Key = 101010 11001100 1010 (Binary format)

HF = Key mod 8

= key mod 2^3

= 101010 11001100 1010 mod 2^3

= 010

≅ 2

* As we can see, only 3 LSB are used for calculating Hash table index.

eg:- $M = 2^K$ (table size)

Key = 101010 11001100 1100 101011
K-bits (LSB)

HF = Key mod 2^K

- So, if LSB k-bits of two or more keys are same then collision will occurs.

NOTE:- (1) Don't pick M value as 2^K, otherwise HF only checks K-bits LSB of key always.

(iii) Choose M to a prime number which is also not close to 2^K.

eg:-

Bad choice	Good choice
16 — (M=31) — 32	64 — (M=97) — 128
32 — (M=63) — 64	32 — (M=59) — 64

(ii) Mid-square method :- [GOOD]

M = 1000 (0-999)

key = 84792

HF(key) = mid [(84792)^2] = 7189683264

mid

Let choose 6 as mid value and take 3 middle elements

7189683264

7189683264 OR 7189683264

we take 3 digits because M=1000 (0-999) i.e 3 digit address or index

968	84792

683	84792

(iii) Digit Extraction method :- [WORST]

M = 1000 (0 - 999)

1 2 3 4 5 6 7 8 9

Key = 7 5 9 4 5 6 8 2 3

HF(Key) = 963

since size of M is of 3 digits then we extract digits at (3x) position

Since digits are taken directly from consecutive 3rd positions, then this will generate more collisions.

(iv) Folding method :- [GOOD]

Fold Boundary method

M = 1000 (0 - 999)

Key = 789 456 123

HF = 789 + 123

= 912

Fold Shifting Method

M = 1000 (0 - 999)

Key = 789 456 123

= 789 + 456 + 123

= 1368

= 136 + 8

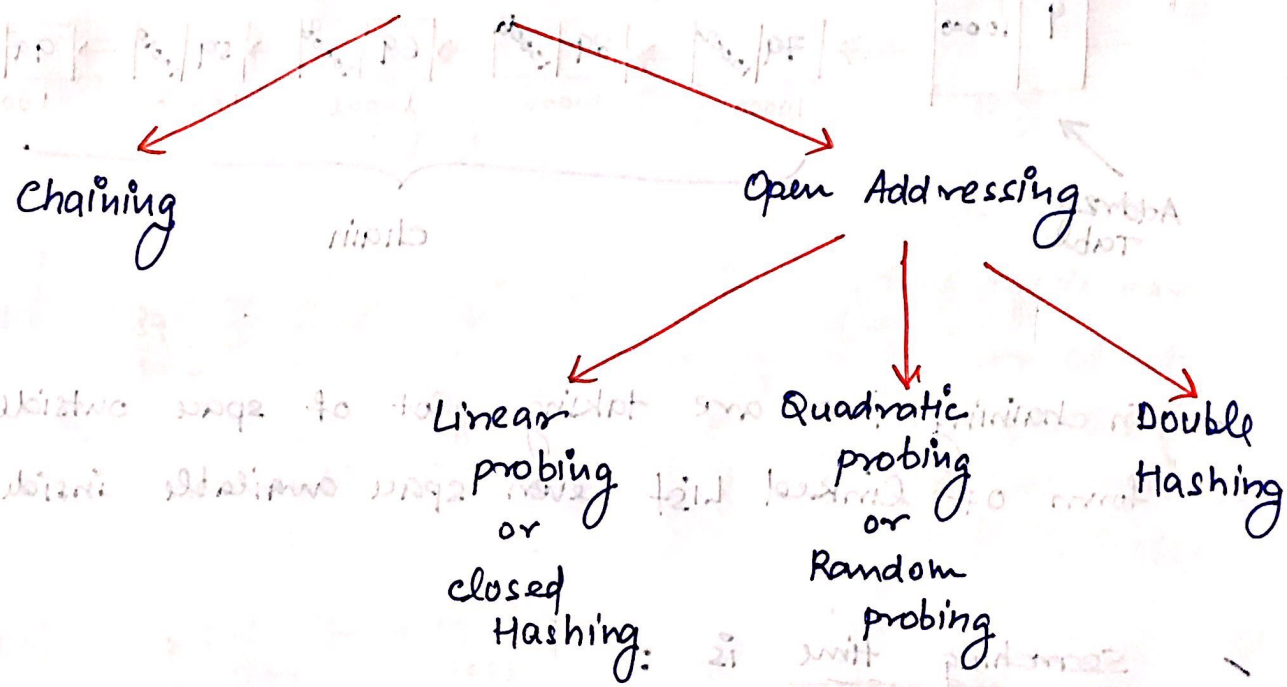
= 144

{ Fold until we get index key of atmost length 3 }

NOTE:- Good Hashing Functions are those which consider all digits to calculate index value so that collision is minimized. But by using these Hashing Function collision occurence is not 0.

To make collision to be 0 we use collision resolution techniques.

Collision Resolution Techniques:-

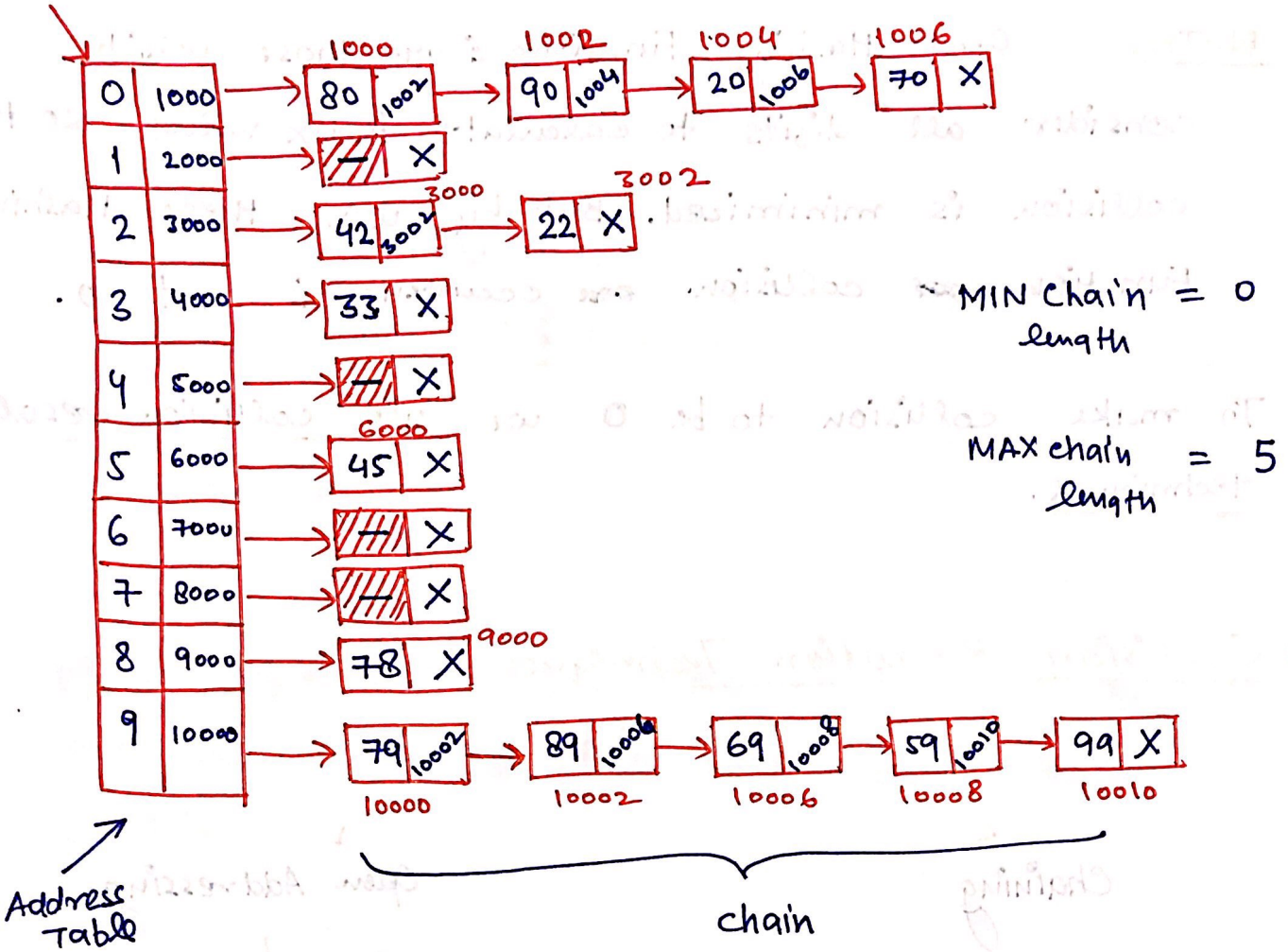


1. Chaining:-

M = 10 (0-9)

Keys = 79, 80, 45, 89, 90, 69, 20, 33, 42, 59, 65, 78, 99, 70, 22

HF(Key) = Key mod m



MIN Chain length = 0

MAX chain length = 5

- In chaining, we are taking lot of space outside in the form of linked list even space available inside table.

- Searching time is :

	BEST	Avg	WORST
T(n)	O(1)	O(1)	O(n)

↓

uniform distribution due to good HASH func^n

- As we can see we have table size 10 and we are storing 15 elements i.e chaining can handle ∞ collisions.

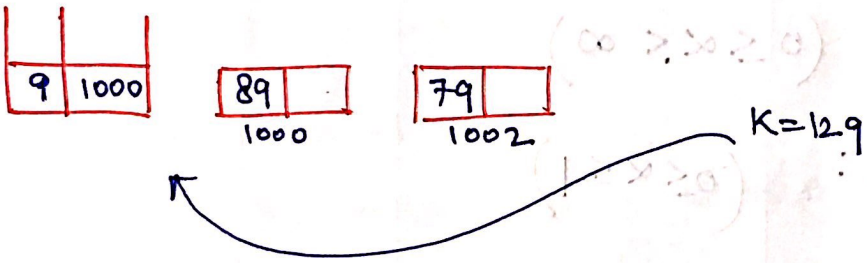
So, if we have no idea of amount of elements to be store in table then we use chaining techniques.

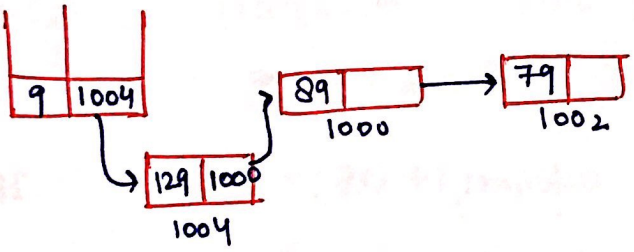
- Insertion time is

	BEST	AVG	WORST
T(n)	O(1)	O(1)	O(1)

↓

: new node can be inserted at first position (no need to traverse until its tail)





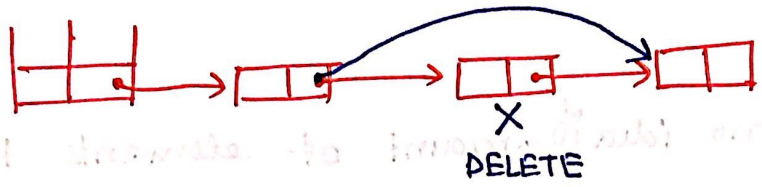
- Deletion Time is

	BEST	AVG	WORST
T(n)	O(1)	O(1)	O(n)

↓

node can be present at n^{th} position

- Deletion is easier in chaining but difficult in open addressing.



2. Open Addressing :-

N = No. of Keys

M = No. of slots

No. of keys / slot = $\frac{N}{M}$ K/s

'α'

(Load factor)

- chaining : $(0 \leq \alpha \leq \infty)$
- open Addressing : $(0 \leq \alpha \leq 1)$

(i) Linear probing :-

M = 10 (0-9)

Keys = 29, 35, 70, 95, 76, 99, 85, 88, 80

HF(Key) = key mod m

$$LP(Key, i) = (HF(Key) + i) \bmod m$$

(i ← attempt number)

$\forall\ i \in \{0, 1, 2, \ldots, m-1\}$

0	70
1	99
2	88
3	80
4	
5	35
6	95
7	76
8	85
9	29

$LP(29,0) = (HF(29)+0) \bmod 10$

$= (9+0) \bmod 10 = 9$

$LP(35,0) = (HF(35)+0) \bmod 10$

$= 5$

$LP(70,0) = (HF(70)+0) \bmod 10$

$= 0$

$LP(95,0) = (HF(95)+0) \bmod 10$

$= 5$ FAILED

$LP(95,1) = (HF(95)+1) \bmod 10$

$= 5+1 = 6$

$LP(76,0) = (HF(76)+0) \bmod 10$

$= 6$ FAILED

$LP(76,1) = (HF(76)+1) \bmod 10$

$= 7$

$LP(85,0) = (HF(85)+0) \bmod 10$

$= 5$ FAILED

$LP(85,1) = (HF(85)+1) \bmod 10$

$= 6$ FAILED

$LP(85,2) = (HF(85)+2) \bmod 10$

$= 7$ FAILED.

$LP(85,2) = (HF(85)+3) \bmod 10$

$= 8$

$LP(99,0) = (HF(99)+0) \bmod 10$

$= 9$ FAILED

$LP(99,1) = (HF(99)+1) \bmod 10$

$= 0$ FAILED

$LP(99,2) = (HF(99)+2) \bmod 10$

$= 1$

(Total Collisions = 14)

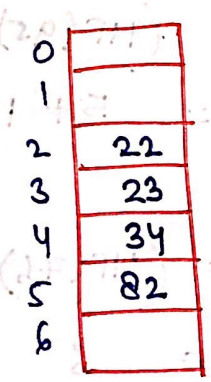
- We are searching one-by-one slot

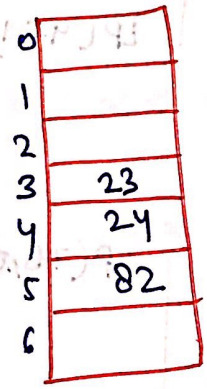
- Searching time / Deletion time / insertion time :-

	BEST	AVG	WORST
T(n)	O(1)	$O(\frac{m+1}{2})$ $\approx O(m)$	O(m) ← size of table

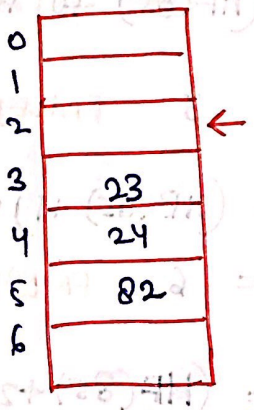
- Deletion can produce problem if we leave deleted space empty.

eg:-



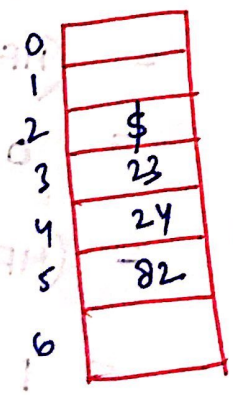


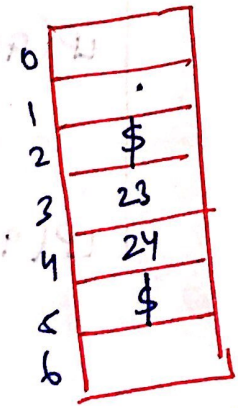
DELETE 22



Now 82 can't be deleted as 2 index is empty and deletion procss stops.

So, we replace deleted element with '$'





DELETE 82

- when number of $ sign increases inside table then we rehash the table and remove $ sign.

Q. $m = 10\ (0-9)$, $HF(Key) = K \bmod m$

After inserting 6 values into an empty HASH Table shown below.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

(i) Correct sequence to get above table:

(a) 46 42 34 52 23 33

(b) 34 42 23 52 33 46

(c) ✓ 46 34 42 23 52 33

(d) 42 46 33 23 34 52

(ii) No. of possible correct sequence to get above HASH Table

(a) 20

(b) ✓ 30

(c) 40

(d) 50

(ii) Quadratic Probing :-

$m = 10 (0-9)$; $HF(Key) = Key \bmod m$

Keys = { 75, 39, 46, 55, 139, 230, 85, 99 }

0	230
1	139
2	
3	
4	
5	75
6	46
7	55
8	
9	39

$$QP(Key, i) = \left(HF(Key) + \underbrace{c_1 i + c_2 i^2}_{\text{Quadratic}} \right) \bmod m$$

(i : attempt no.)

$$\forall i \in \{0, 1, 2, \ldots, m-1\}$$

Let $c_1 = 1$; $c_2 = 1$

$QP(75,0) = (5 + 1(0) + 1(0)^2) \bmod 10$

$= 5$

$QP(39,0) = (9 + 1(0) + 1(0)^2) \bmod 10 = 9$

$QP(46,0) = (6 + 1(0) + 1(0)^2) \bmod 10 = 6$

$QP(55,0) = (5 + 1(0) + 1(0)^2) \bmod 10 = 5$ Collision

$QP(55,1) = (5 + 1(1) + 1(1)^2) \bmod 10 = 7$

$QP(139,0) = 9$

$QP(139,1) = (9+2) \bmod 10 = 1$

- QP (85,0) and QP (99,0) will fails, they keep on colliding.

[Total no. of collision = 22

No. of elements we can't insert = 2 (85, 99)]

NOTE:- By default $c_1 = 0$ and $c_2 = 1$

QP is fast but it can't gurantee to place (insert) elements.

- Searching / Delelion / Insertion time:-

	BEST	AVG	WORST
T(n)	O(1)	O(m)	O(m)

- Delelion produce the same problem as LP, so we insert ∮ at deleted place.

(ii) Double Hashing:-

$m = 10 (0-9)$ $HF_1(key) = key \bmod m$

$HF_2(key) = 1 + (key \bmod (m-2))$

(keys = 75, 39, 46, 55, 139, 230, 85, 99)

0	230
1	85
2	
3	55
4	
5	75
6	46
7	139
8	
9	39

$$DH(Key, i) = \left(HF_1(Key) + i \cdot HF_2(Key) \right) \bmod m$$

$$\forall\ i \in \{0, 1, 2, \ldots, m-1\}$$

$DH(75,0) = (HF_1(75) + 0) \bmod 10 = 5$

$DH(55,0) = (HF_1(55) + 0) \bmod 10 = 5$ collision

$DH(55,1) = (HF_1(55) + HF_2(55)) \bmod 10 = (5+8) \bmod 10 = 3$

$DH(139,0) = 9$ collision

$DH(139,1) = (9 + 4) \bmod 10 = 3$ collision

$DH(139,2) = (9 + 8) \bmod 10 = 7$

$DH(85,0) = 5$ collision

$DH(85,1) = (5+6) \bmod 10 = 1$

$DH(99,0) = 9$ collision

$DH(99,1) = (9+4) \bmod 10 = 3$ collision

$DH(99,2) = (9+16) \bmod 10 = 7$ collision

$DH(99,3) = (9+36) \bmod 10 = 5$ collision

So, 99 can't be place.

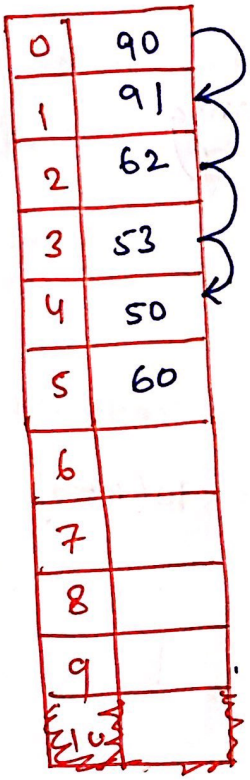
Total collision = 1 + 2 + 1 + 10 = 14

Q. Apply Linear probing :-

m = 10 (0-9)

HF(key) = key mod m

keys = 53, 90, 62, 91, 50, 60



LP(50, 0) = 0
LP(50, 1) = 1
LP(50, 2) = 2
LP(50, 3) = 3
LP(50, 4) = 4

} 4 collisions

LP(60, 0) = 0
LP(60, 1) = 1
⋮ 2
⋮ 3
LP(60, 4) = 4
LP(60, 5) = 5

} 5 collisions

As we can see 60 ~~ods~~.
do same comparison as 50
, this is known as primary clustering

- **Primary clustering :-** If 2 keys contain same starting HASH address, then they both follow same path unnecessarily in linear manner, because of this region ~~avg~~ $T(n)_{avg}$ for searching increases.

$$\frac{1+2+3+\cdots+m}{m} = \frac{\frac{m(m+1)}{2}}{m} = O(m)$$

- LP has the problem of Primary clustering
- LP searching time :- Avg: $[O(m)]$

Q. Apply Quadratic probing :- $m = 10 (0-9)$

keys = 53, 90, 62, 91, 50, 60

0	90
1	91
2	62
3	53
4	
5	
6	50
7	
8	
9	

$QP(50,0) = 0$ C

$QP(50,1) = (0 + 1(1) + 1(1)^2) \bmod 10$

$= 2$ C

$QP(50,2) = (0 + 1(2) + 1(4)) \bmod 10$

$= 6$

$QP(60,0) = 0$

$QP(60,1) = (0+1+1^2) \bmod 10 = 2$

$QP(60,2) = (0+2+4) \bmod 10 = 6$

$QP(60,3) = (0+3+9) \bmod 10 = 2$

$QP(60,4) = (0+4+16) \bmod 10 = 0$

⋮ ⋮

- **Secondary Clustering:-** If 2 keys contain same starting HASH address, then they both follow same path in quadratic manner unnecessarily, because of this reason T_{avg} for searching increases.

- QP has problem of ~~Quad~~ secondary clustering

- **NOTE:-** Primary clustering $\leftrightarrow$ secondary clustering

(Asymptotically equal)

Primary clustering $>$ secondary clustering

(Mathmatically)

Q. Apply Double Hashing :- $HF_1(Key) = K \bmod m$

$HF_2(K) = 1 + K \bmod m-2$

Keys = { 53, 90, 62, 91, 50, 60, 80 }

0	90
1	91
2	62
3	53
4	80
5	60
6	50
7	
8	
9	

DH(50,0) = 0 C

DH(50,1) = (0 + 3) mod 10 = 3 C

DH(50,2) = (0 + 2(3)) mod 10 = 6

DH(60,0) = 0 C

DH(60,1) = (0+5) mod 10 = 5

DH(80,0) = 0 C

DH(80,1) = 0+1 C

DH(80,2) = 0+2 C

⋮

DH(80,4) = 0+4 = 4

All keys follow different paths ; so Tavg of searching time will not increase.

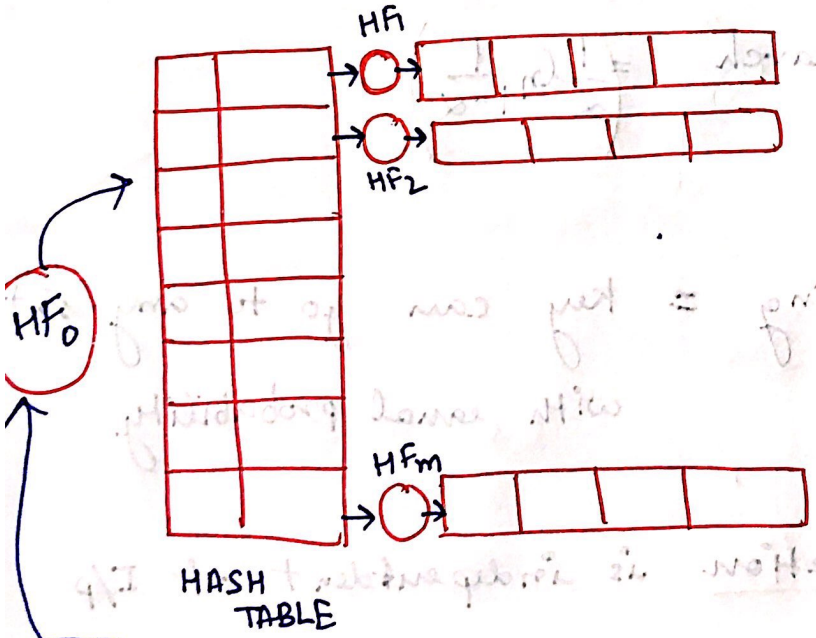
$\sum$ different path = m = O(m) So, T(n) = O(1)

Searching Time	BEST	AVG	WORST
Linear probing	O(1)	O(m) Primary clustering	O(m)
Quadratic probing	O(1)	O(m) Secondary clustering	O(m)
Double Hashing	O(1)	O(1) No clustering	O(m)

NOTE :-

Using "perfect hashing" we will get searching time

$T(n) = O(1)$ (every case)



we have (m+1)
HF (HF_0, HF_1, -- -- HF_m)

(HF_0 → keys to HASH TABLE)

HF_1 → slot 1
HF_2 → slot 2
. . .
HF_m → slot m

- In this technique we have a table of m slots & each slot has there own space.

- HF_0 is used to select index of slot, and then the corresponding HF_i of that slot will choose (select) the sub-slot.

- Space complixity ↑ & worst time complixity ↓ O(1)

NOTE:-

- The expected no. of probes (comparision) in an unsuccessful search of open addressing technic $= \left(\frac{1}{1-\alpha}\right)$.

  and for successful search $= \left(\frac{1}{\alpha} \ln \frac{1}{1-\alpha}\right)$.

- simple uniform hashing = key can go to any slot with equal probability

- Universal hash function is independent of I/p type.