Hashing Technique

1. why hasing
2. Ideal hasing
3. modulus hasing function
4. Drawbacks
5. solutions

we know
1. linear - n
2. Binary - log n

☐ Hasing is usefull for searching

we want more faster than logn
O(1) How?

key : 8, 3, 6, 10, 15, 18, 4

A → Linear

8	3	6	10	15	18	4
0	1	2	3	4	5	6

sort

A → Binary search

3	4	6	8	10	15	18
0	1	2	3	4	5	6

position

key is store in the same index.

H

-	-	-	3	4	-	6	-	8	-	10	-	-	-	-	15	-	-	18	-	-
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

How much time to Find out 10?
it constant time.

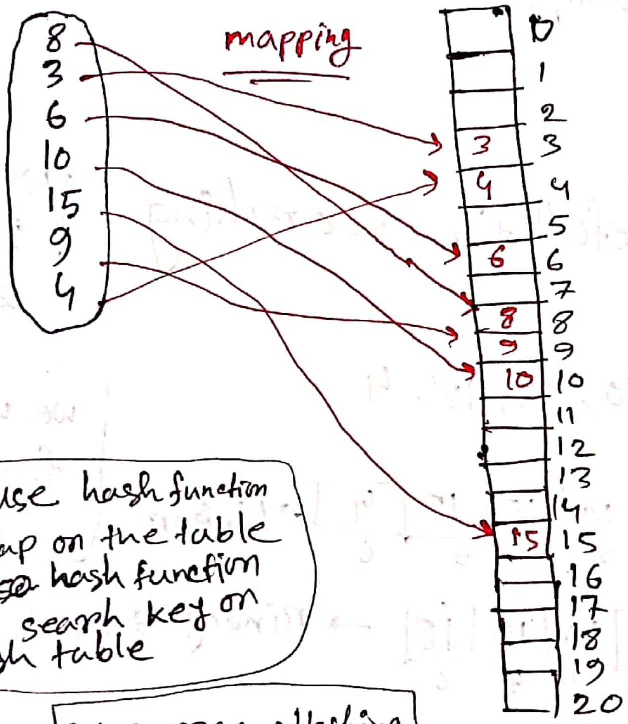
Drawbacks : lots of space is wostes here:
⇓
for solving the problem we need some mathametical model.

key: 8, 3, 6, 10, 15, 9, 4

key space $\quad h(n)=n \quad$ Hash table

mapping(4)

- $h(n)=n$ function: one – one
- one – many
- $h(n)=n\%10$: many – one
- many – many



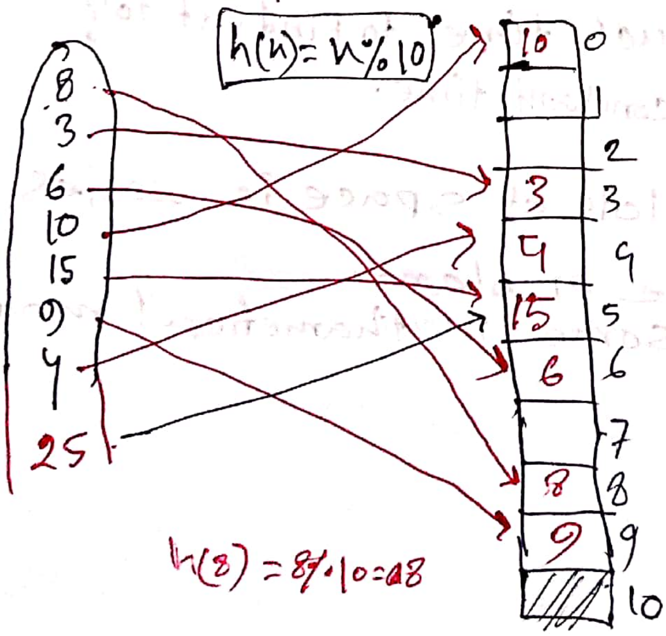
We use hash function to map on the table and ~~so~~ hash function also search key on hash table

One-one → Hashing take lots space

*** change hash function to reduce the space** $\%$

$h(n) = n\%10$

$h(25) = 25\%10 = 5$



when tow key map on the same location we call it collection

15, 25

$h(8) = 8\%10 = 8$

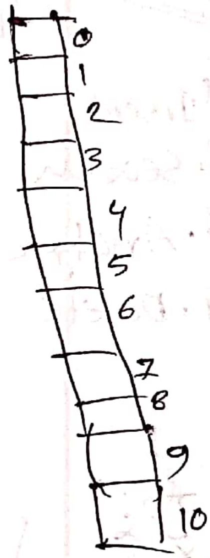
What are the methord Resolving collision

<u>Open Hashing</u> ⇒ we assuming extra speace

- chaining

<u>Closed Hashing</u>

- open · Addressing
  1. Linear Probing
  2. Quadratic probing.
  3. Double Hashing.



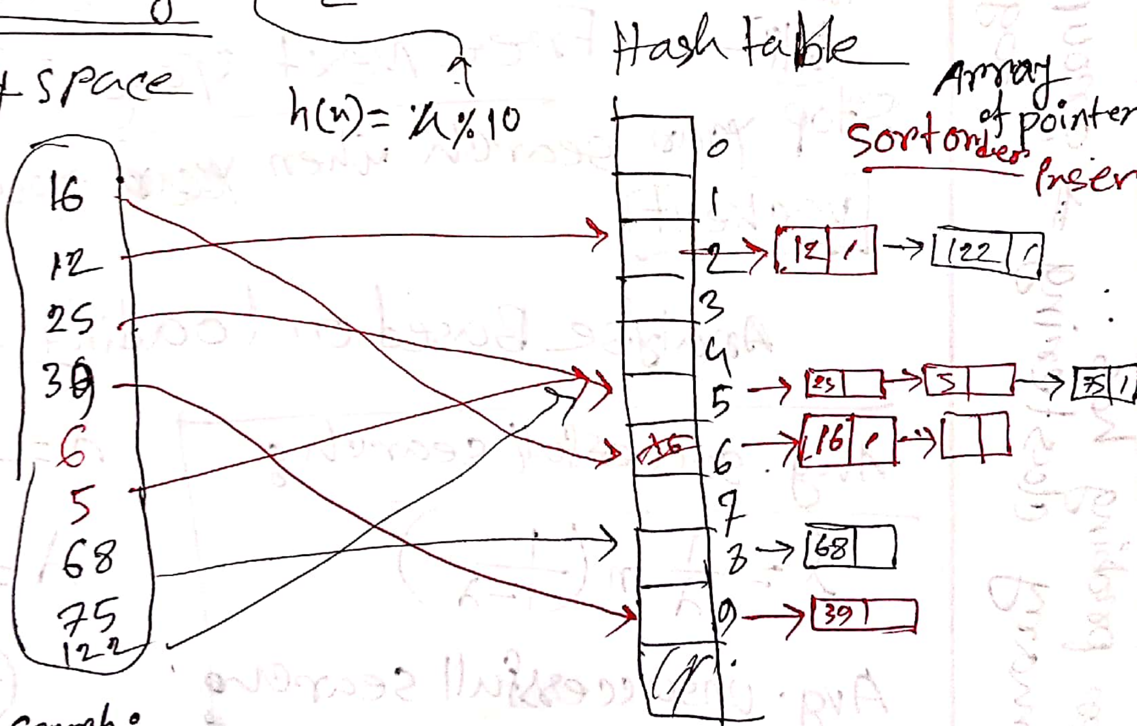
* Chaining

[we are working in the last digit. But 2nd last digit ⇒ $h(n) = (n/10)\%10$]

Key space

$h(n) = n\%10$

1. Insert
2. search
3. Analysis
4. Delete



<u>successfull search:</u>

$t = 1 + \lambda/2$ assuming Average time

<u>unsuccessfull search:</u>

$t = 1 + \lambda$

key → $n = 100$

table → size $= 10$

Loading factor $\lambda = \frac{n}{size}$

We have to select the hash function so that key is uniformly distributed.

Lecture 9

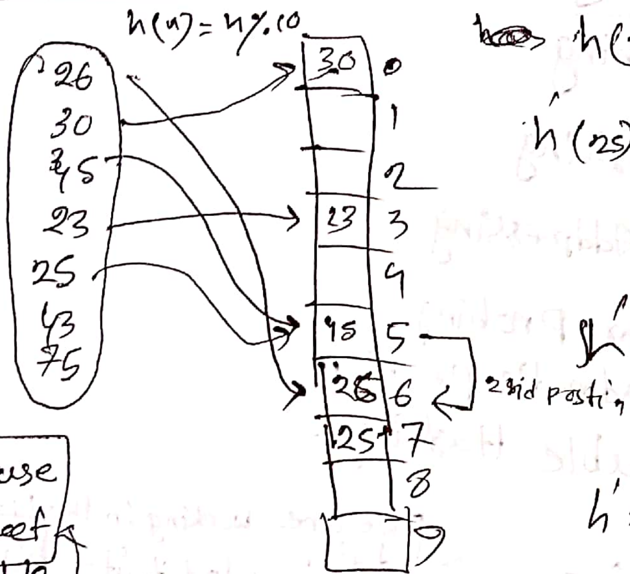
Linier probing

we not consume extra space

$f(i) = i$

1. Insert
2. Search
3. Analysis
4. Delete

$h(n) = n \% 10$



$h'(n) = (h(n) + f(i)) \% 10$

$h'(25) = (h(25) + f(0)) \% 10$
$= (5 + 0) \% 10$
$= 5$

$h' = (h(25) + f(1)) \% 10$
$= (5 + 1) \% 10$
$= 6$

$h' = (h(25) + f(2)) \% 10$
$= (5 + 2) \% 10$
$= 7$

Stag use for delect two table

Free Next space

Stop your search when your space is $\boxed{\text{vackent}}$

Linier probing has: primary clustering ⇒ group of key together.

Analyise Based on loading factor

Aveg. successful search:

$$l = \frac{1}{\lambda} \ln\left(\frac{1}{1-\lambda}\right)$$

Avg. unsuccessfull search:

$$l = \frac{1}{1-\lambda}$$

$$\lambda = \frac{n}{size}$$

$$\lambda = \frac{9}{10} = 0.9$$

$$\boxed{\lambda \leq 0.5}$$

Whe delete key from Hash table

ReHashing ⇒ again all the key Insert

Scanned with CamScanner

suggest that not delete elemem from the linier broping table. usin flag rather than.

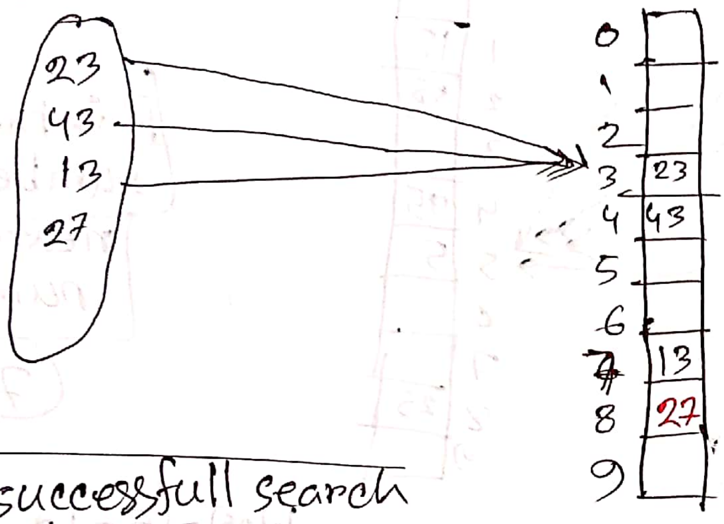
Lecture 6

Qudratic Probing

key space

$h(n) = n \% 10$

Hash table



modified for qudratic equation

$h'(n) = (h(n) + f(i)) \% 10$

$f(i) = i^2$

$h'(43) = (h(43) + f(0)) \% 10$

$= (3 + 0) \% 10$

$= 4$

$h'(13) = (h(13) + f(0)) \% 10$

$= (3 + 0) \% = 3$

$i=1 \quad = (3 + 1) \% 10 = 4$

$i=2 \quad = (3 + 4) \% 10 = 7$

i
0
1
4
9

Avg successfull search

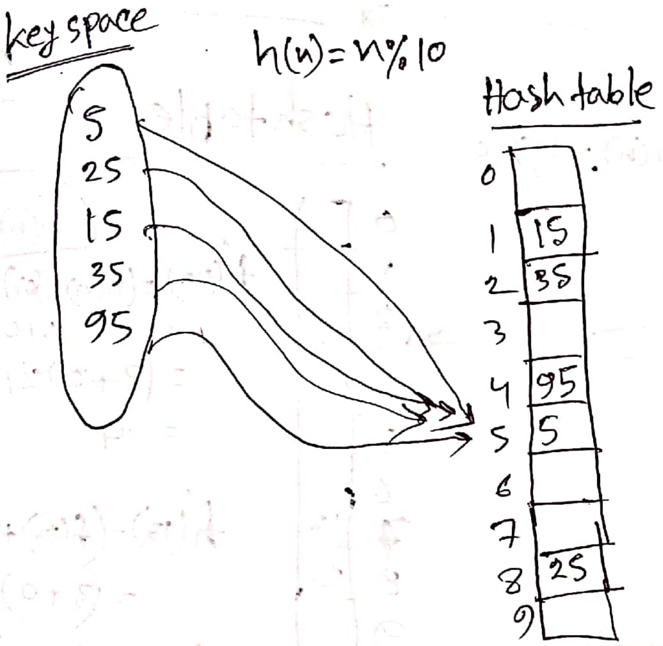
$$t = -\log_e (1-\lambda)$$

Averge unsuccessfull search

$$t = \frac{1}{1-\lambda}$$

Lecture 7

Double hasing



$h_1(n) = n\%10$

$h_2(n) = R - (n\%R)$

$h'(n) = (h_1(n) + i * h_2(n))\%10$

$i = 0, 1, 2 ----$

R is the primary number

near prime numbers of size

$7 = R$

$h'(25) = (5 + 1 * 3)\%10 = 8$

$h_2 = 7 - (25\%7)$
$= 7 - 4 = 3$

$h'(15) = (5 + 1 * 6)\%10 = 1$

$h_2 = 7 - (15\%7)$
$= 7 - 1 = 6$

$h'(35) = (5 + 1 * 7)\%10 = 2$

$h_2 = 7 - (35\%7)$
$= 7 - 0 = 7$

$h'(95) = (5 + 1 * 3)\%10 = 8$

2nd colution

$h_2 = 7 - (95\%7)$
$= 7 - 4 = 3$

$= (5 + 2 * 3)\%10 = 1$

3rd $= (4 + 3 * 3)\%10 = 4$

Lecture 8

Different hasing function

evoid 0 ⇒ h(n) = (n%size) +1

↑ prime num collution is Reduce

1. mod
2. midsquare
3. Folding

Mid suquare

key = $(11)^2$
= 121
↑
digit of that

key = $(13)^2$
= 169
↑
_ _ _
↑
_ _ _ _
→ % of middil of 2 digit

key = "ABC"
A, B C
65 66 67
656667

  65
+66
+67
198

like

12 33 47

⇒ 12
+33
+47
92 ⇒ 9+2=11 → Hash table

This numbe is larger then mood %