* Hash Tables :-

* what is Hashing ?

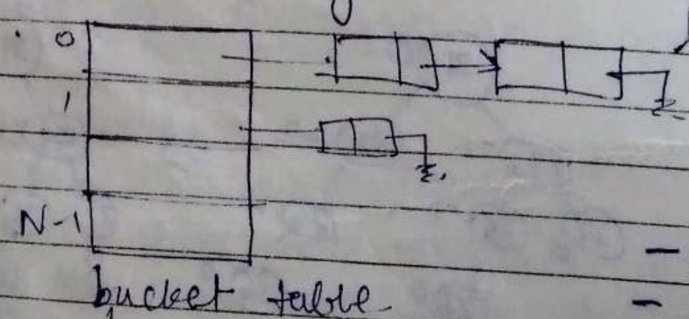
→ Table used for storing of records is known as hash table.
→ Function h(key) is known as hash function

eg: K = { ~~ab~~ "aaa", "bbb", "ccc", "ddd" }

N	h(x)
aaa	0
bbb	1
ccc	2
ddd	3

* Hash table data structure .-

The diff ~~Hash table~~ forms of hashing :-
1) external hashing, or open hashing.
2) close hashing.



bucket table header

0	a
1	~~d~~
2	b
3	c
4	
6	d

Topics :-
- Hashing
- Hash table (key, value)
- hash function.
  ↳ chara of H. F.
- bucket
- collision
- overflow / full table
- probe / synonyms

probe ↑ allen to distinct location

synonyms ↑ colliding two keys are called syn.

* Hashing functions :-

→ A hash function maps a key into a bucket in the hash table.
→ The position of a key in the array is given by a function h(·), called a hash function, which determines the position of a given key directly from that key.

→ There will exist many pairs of distinct keys x & y such that $x \neq y$ for which $h(x) = h(y)$. This is called collision.

* chara. of Good Hash function :-

1) Avoids collisions.
2) Easy to compute.

There are following hash functions :-

1) Division - method :-

$$L = (K \bmod m)$$

L = location of key in the table.
K = key, m = table size.
e.g. :- K = 23 & m = 10 then L = 3.

2) Midsquare methods :-

→ Square the value of key & take the no. of digits required to form the address from the middle position of squared value.
e.g. $16^2 = 256$ then $h(k) = 56$
$3111^2 = 9678321$ if addr is 3 bit then
= 783. will be the address.

3) **Folding method :-**

1) fold-shifting :- Here actual values of each parts of key are added.

e.g. 12345678.

$= 12 + 34 + 56 + 78$

$= \boxed{1}80$ → Ignore

addr. = 80.

2) Fold-boundary :- Here the reversed values of outer parts of key are added.

$= 21 + 34 + 56 + 87.$

$= \boxed{1}98$ → Ignore.

addr. = 98.

(4) **Digit analysis :-**

e.g :- 9861234 so 3rd & 8th position digits are occur quite frequently then we choose these digits & reverse them 62 so addr is 26.

(5) **Length dependent method :**

K = 324. then length of K = 34 last digit so 34 is the address.

⑥ Multiplicative Hashing :-

→ This method is based on obtaining an address of a key, based on the multiplication value i.e. If K is the non-negative value, & a constant c, ($0<c<1$) compute KC mod 1, which is a fractional part of KC. mul. this part with m & take floor value to get addr.

$$h(K) = \lfloor m \cdot (KC \bmod 1) \rfloor$$

$$0 \le h(K) < m$$

* Collision Resolution strategies :-

① Separate chaining :- (Open hashing).

→ In this a separate list of all elements mapped to the same value is maintained.

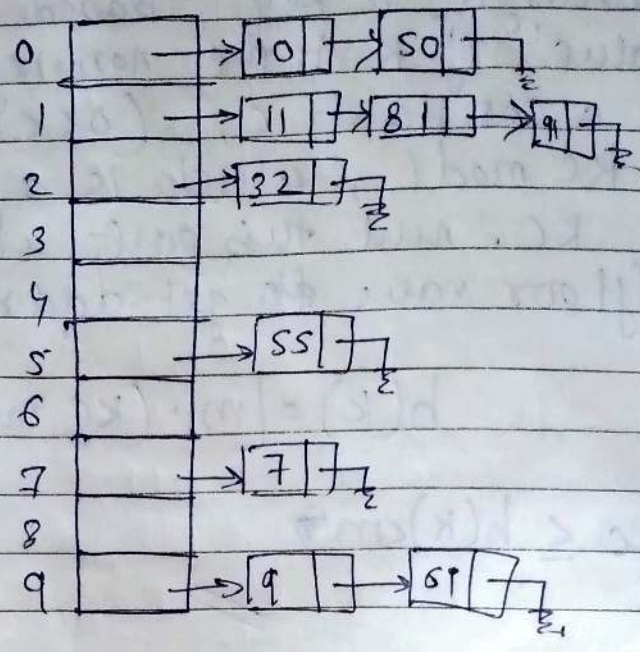
→ Separate chaining is used for collision avoidance.

→ If memory space is tight, this method is not feasible.

→ Additional memory space is wasted in storing address of linked elements.

e.g.:- a simple hash function i.e.

$h(x) = x \bmod 10$, Bucket size = 10.



(2) Open addressing :- (closed hashing)

→ Seperate chaining requires additional memory space for pointers.

→ This is alternate method used for collision handling, in which if a collision occurs, alternate cells are tried until an empty cell is found.

→ all data elements are stored inside the table a larger memory space is needed for Open addr.

→ There are following commonly used collision resolution strategies for open addr:-

1) linear probing
2) Quadratic probing.
3) Double hashing

① Linear probing :- (without chaining)

→ when collision occurs i.e when two different records demand for the same location in the hash table, then the collision can be solved by placing second record linearly down wherever the empty location is found.

for eg :- {~~131~~, ~~21~~, 5, 18, 55, 78, 35, 15} size of table is 10 so, m = 10.

$$h(x) = x \bmod m$$
$$= x \bmod 10$$

0	15
1	
2	
3	
4	
5	5
6	55
7	35
8	18
9	78

② Linear probing with chaining :- (without replacement)

→ Excessive collision in linear probing could become a major problem.

→ one way of dealing with collision is by means of chaining.

→ all records mapped to same location are stored in a chain. i.e the extra field is added with data.

e.g.: $m = 10$, $\{0, 1, 4, 71, 64, 89, 11, 33\}$

$h(x) = x \bmod m$.

	key	chain
0	0	-1
1	1	-1
2		-1
3		-1
4	4	-1
5		-1
6		-1
7		-1
8		-1
9		-1

0	0	-1
1	1	2
2	71	3
3	11	-1
4	4	5
5	64	-1
6	33	-1
7		-1
8		-1
9	89	-1

1 → 71 → 11

1 2
71 6
33 -1

11

③ Linear probing with Chaining (With Replacement)

→ problem of misplaced storing location of the chain can be handled through chaining with replacement method.

if we have like

add 22

0	0	-1
1	1	2
2	71	3
3	11	-1
4	4	5
5	64	-1
6		-1
7		-1
8		-1
9	89	-1

Q. 12, ~~0~~1, 4, 3, 7, 8, 10, 2, 5, 14, 6, 28.
insert into hash table. m = 10.
(0-9)

then,

1 → 71 → 11

→ for addition of 22 we have to delete 71 from the chain & added at the end of the chain so new chain is 1 → 11 → 71

→ New element 22 can be added to slot 2

0	0	
1	1	3
2	22	-1
3	11	6
4	4	5
5	64	-1
6	71	-1
7		
8		
9	89	-1

10, 51, 22, 63, 61,

e.g.: 3 {11, 33, 20, 88, 79, 98, 44, 68, 66, 22}

10, 51, 63, 61

e.g.:- 2 {10, 51, 63, 61, 73, 53}

0	10	-1
1	51	~~2~~ 4
2	61	-1
3	63	8
4	~~61~~ 73	~~9~~ 5
5	53	-1
6	~~73~~	-1
7	.	-1
8	53	6
9		-1

add 55.

0	10	-1
1	51	4
2		-1
3	63	8
4	61	-1
5	55	-1
6	73	-1
7		-1
8	53	6
9		-1

add 58.

63 → 53 → 73

Delete 53 & add at end

63 → 73 → 53 → 13

0	10	-1
1	51	4
2	.	-1
3	63	6
4	61	-1
5	55	-1
6	73	9
7		-1
8	58	-1
9	53	-1

63 → 53 73

58

63 → 73 → 53

0	10	-1
1	51	4
2		-1
3	63	6
4	61	-1
5	55	-1
6	73	9
7	13	-1
8	58	-1
9	55	7

add 13.

* Quadratic probing :-

⇒ one way of reducing "primary clustering" is to used quadratic probing to solve collision.

⇒ In quadratic probing :-

1) we start from the original hash location i.

2) If the location is occupied, we check the locations $i+1^2$, $i+2^2$, $i+3^2$, $i+4^2$ ------

3) we wrap around from the last table location to the first table location if necessary.

e.g.: Table size $m = 11$ (0 -- 10)

$h(x) = x \bmod m$

Keys = {20, 30, 2, 13, 25, 24, 10, 9}

20 mod 11 = 9

30 mod 11 = 8

2 mod 11 = 2

13 mod 11 = 2 = $(2+1^2) = 3$

25 mod 11 = 3 = $(3+1^2) = 4$

24 mod 11 = 2 = $(2+1^2) = 3$

= $(2+2^2) = 6$

10 mod 11 = 10

9 mod 11 = 9 = $(9+1^2) = 10$

= $(9+2^2)$ mod 11 = 3

= $(9+3^2)$ mod 11 =

18

= $\cancel{(9+4^2)}$

9+16

0	
1	
2	2
3	13
4	25
5	
6	24
7	9
8	30
9	20
10	10

2

= 2

* Double Hashing :-

→ Double Hashing reduces clustering in a better way.
→ This method requires two hashing functions $h_1(key)$ & $h_2(key)$.
→ $h_1(key)$ is known as primary hash function.
→ In case the address obtained by $h_1(key)$ is already occupied by a key, the function $h_2(key)$ is evaluated.
→ The second function $h_2(key)$ is used to compute the increment to be added to the address obtained by the first hash function $h_1(key)$ in case of collision.
→ The search for an empty location is made successively at the address.

$h_1(key) + h_2(key)$, $h_1(key) + 2h_2(key)$,
$h_1(key) + 3h_2(key)$, ------

<u>example</u> :

– $m = 11\ (0 \cdots 10)$, $h_1(x) = x \bmod 11$
$h_2(x) = 7 - (x \bmod 7)$

keys = { 58, 14, 91, 25 }

$= 58 \bmod 11 = 3$
$= 14 \bmod 11 = 3 \rightarrow 7-(0) = 7 = 7+3 = 10$
$= 91 \bmod 11 = 3 \rightarrow 7-(0) = 3+14 = 17 \bmod 11 = 6$
$= 25 \bmod 11 = 3+3\times7 = 24 \bmod 11 = 2$
$= 3 \rightarrow 7-(4) = 3 = 3+3 = 6$
$= 3+2\times3 = 9$

0	
1	
2	
3	58
4	
5	
6	91
7	
8	
9	25
10	14

example :-

Q. Explain linear probing with & without replacement using the following data:
12, 01, 04, 03, 07, 08, 10, 02, 05, 14, 06, 28
m = 10. calculate no. of comparison for both.

⇒ without replacement :-

12 - 10 = 1 comparison.
02 = 4 comp.
05 = 2 ".
14 = 6 ".

Total = 7×1 + 4 + 2 + 6 = 19

with replacement :-

12 - 10 = 1 comp.
02 = 4 "
05 = 3 "
14 = 6 "

Total = 1×7 + 4 + 3 + 6 = 20

* Rehashing :- load factor $\lambda = \frac{n}{N}$ → entries, → size of table
$\lambda < 1$ $\lambda \geq 1$ & > 1 desirable

* Extendible Hashing :- (or bucket hashing)

→ Hashing tech. is discuss so far, work effectively when the data size is small.

→ When data becomes bulky, there may be two many disk access.

→ In order to reduce disk accesses while retriving the data, we make use of extendible hashing.

→ which handle large amt of data.

In this method :-

1) we try to keep the information in the form of directory structure.

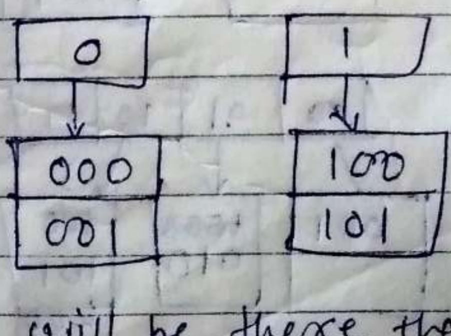
2) If K bit is used for directory then the no. of entries in its bucket are 2^K.
ie if $K = 2$ then $2^2 = 4$ max element of director 2 digit.

3) while adding no. in dictionary convert no. in 0's & 1's form ie binary no.

for example :-

| 0 | 1 | ← directory

$K = 1$ then $2^K = 2^1 = 2$ elements can be placed in each bucket.



0

if 010 will be there then ~~split~~ we can add to 0th directory because the 2 entries are already filled, then double the size of directory. now $K = 2$ so 4 maximum

| 00 | 01 | 10 | 11 |

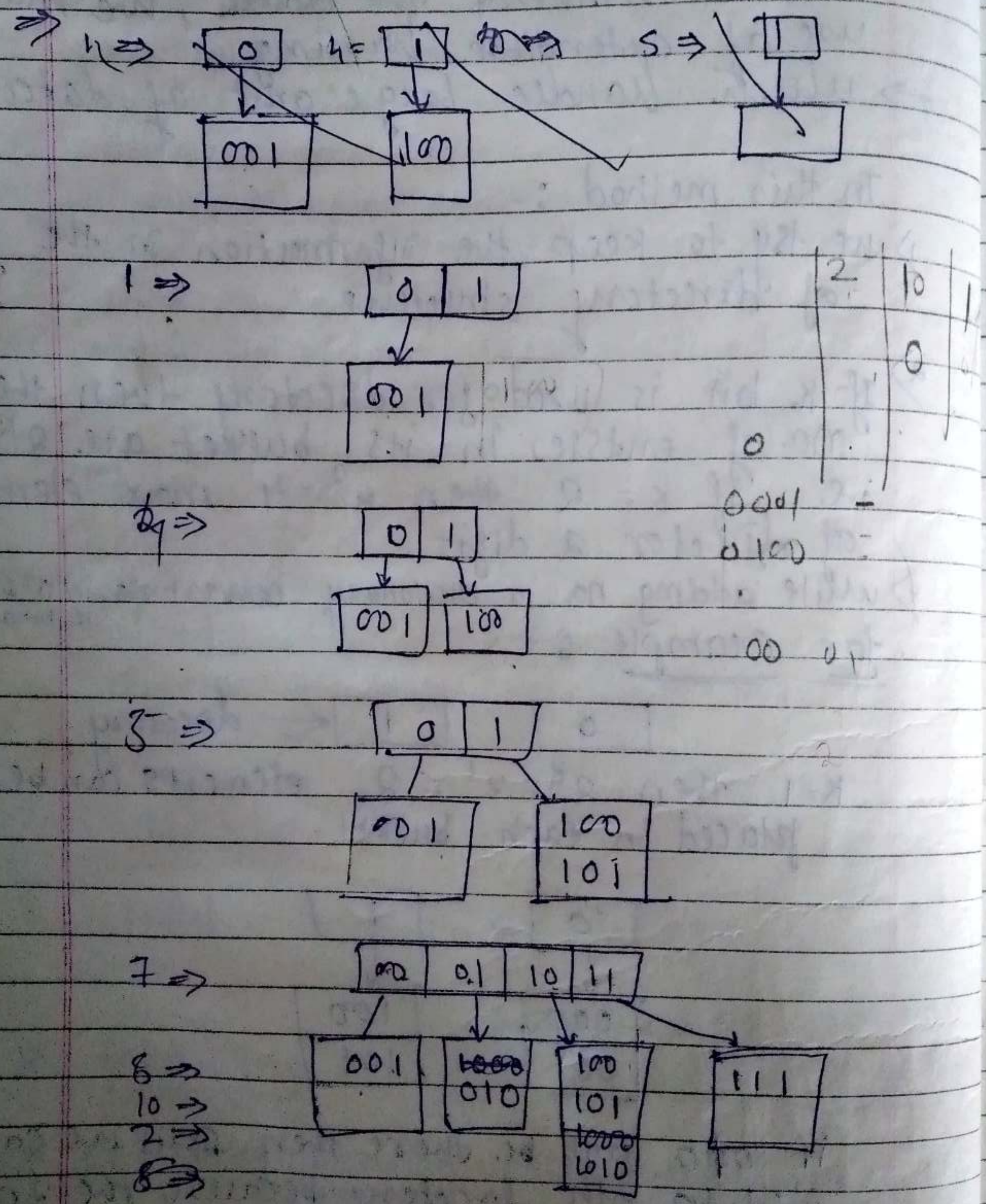
8421 0011   8421 0001

Example:-

0001 0100 0101 111 1000 1010

① Keys = {1, 4, 5, 7, 8, 10, 2}

Assume each page can hold 2 data entries



Adv:-

1) It is useful when the data size is large enough.
2) This method results in quick access time (both for finding & inserting)
3) The directory size can be increased.

Q. Given the input {4371, 1323, 6173, 4199, 4344, 9679, 1989} & hash function $h(X) = X \bmod 10$.
1) open addressing hash table using LP.
2) " " " " quadratic prob
3) " " " " using second hash function
$h_2(X) = 7 - (X \bmod 7)$. (12)

Q. {12, 01, 04, 03, 07, 08, 10, 02, 05, 14, 06, 28}

m = 10.

h 4371 % 10 = 1