Hashing

- **Direct Address Table =**

(DAT)

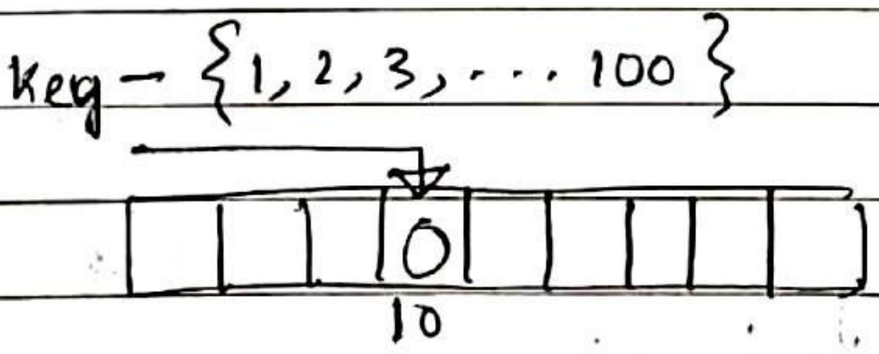
Searching time of some DS -

(i) unsorted array — O(n)
(ii) sorted array — O(log n)
(iii) linked list — O(n).
(iv) Binary tree(BT) — O(n)
(v) Binary search tree — O(n)
(vi) Balanced BST — O(log n).
(vii) priority queue(min & max heap) — O(n).

→ If we used hashing technique then on average time taken to search an element is — O(1).
Hashing is used to minimize the search time.

Before using hashing technique people used one more technique called as DAT (Direct address table).

DAT: It is similar to Array.



→ The first DS which has supported the searching in - O(1) is called DAT (Direct address table). In this table we are have a array and we are going ~~have~~ to place an element according to key value.

→ This method is usefull only when you know the no. - of keys which are going to in small range. ~~and~~

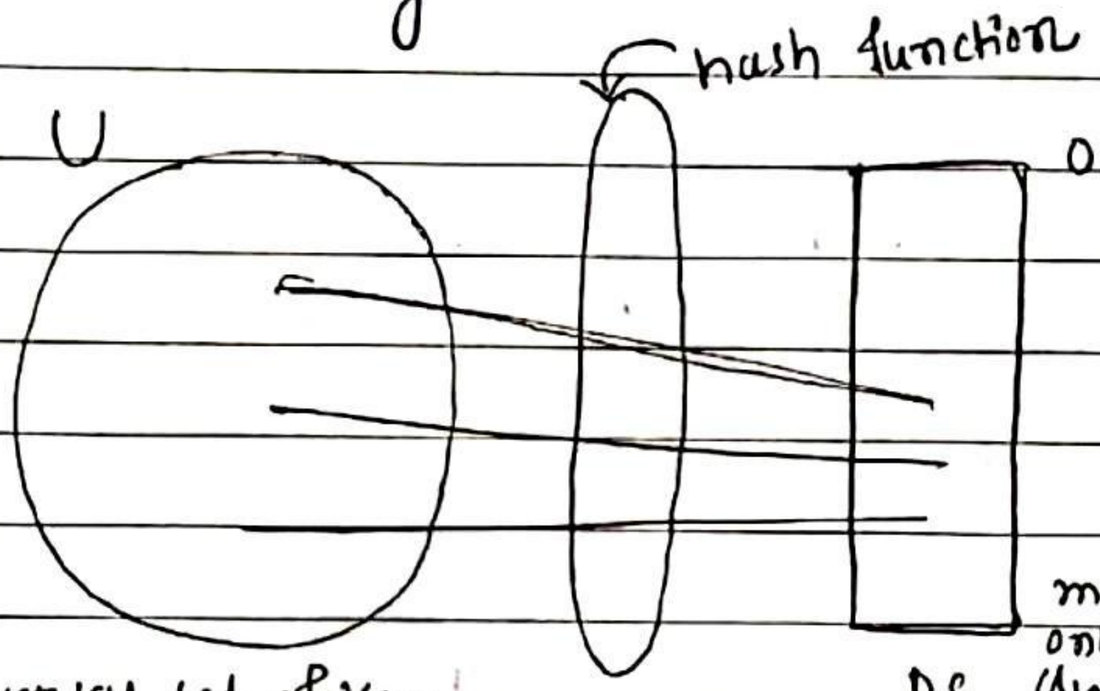
In case if the Key's used are very larger number, but the total no. of Key's are less in number.

Key - {100000, 100001, 100002, ..}

	100000	100001	...

so, that the array size will be 1M, just to store 10 value. In such case DAT fail. (Waste of space)
→ solve this problem we use hashing.

- Introduction to hashing =



hash function :
$h(U) \rightarrow (0-(m-1))$
$h(0-999) \rightarrow (0-9)$
hash function is any function which will take a key from U and it will map it to one of the values from 0 to m-1.

example : (0-999)
set of key - (121, 145, 132, 999)
hash function - (mod 10)

0	1	2	3	4	5	6	7	8	9
NULL	121	132	NULL	N	145	N	N	N	999

(size-10)

When searching for an element - 150, 145

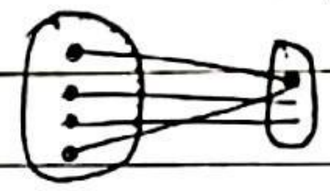
150 % 10
= 0
element is not present.

145 % 10
= 5 element is present.

→ Insertion, deletion and searching is going to take constant time by using the hashing technique.

→ ~~Collision problem, there is a problem called~~

→ Collision problem =

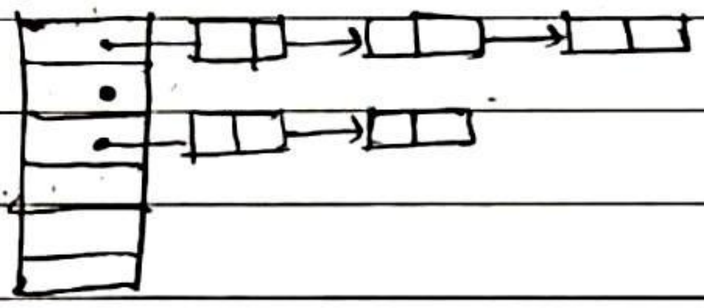


↳ When two key or two element being hash into same cell that condition is called collision.

→ Solving collision problem is —

(i) Better hash function.

(ii) Chaining. (I, D, S)



(For deletion chaining is better)

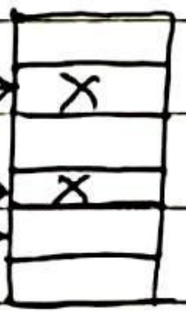
(iii) Open addressing. (I, S)

(probing — searching for a empty space to insert an element)

(i) linear probing

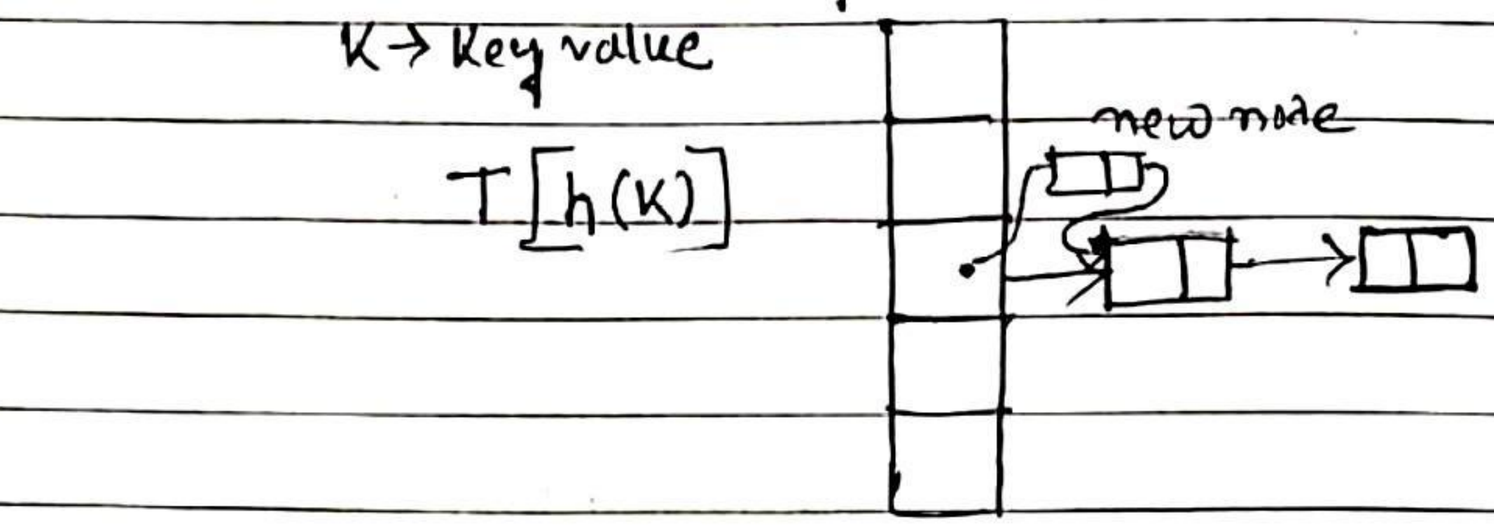
(ii) quad probing

(iii) Double hashing



(For deletion open addressing is not better)

- Collision solving technique : [chaining].



hash table

→ Insertion In case of chaining time taken = O(1).

→ In worst case time for searching = O(n).

(n element inserted in the table, all the element might map @ (in worst case) to a single entry (cell) and the list of entry might contain as many as n)

→ In worst case Deletion time = O(n).

$\alpha = \frac{n}{m}$

α → load factor.
n → total element present in table.
m → total size of table

In this case average search time = $\theta(1+\frac{n}{m})$

$n = Km$ (constant)

$\frac{n}{m} = K(\alpha)$

$= \theta(1+\alpha)$

$= \theta(1)$

average deletion time = $\theta(1)$.

~~Adv~~: Adv:

→ deletion is easy in chaining.

disadv:

→ pointers (space wasted).

1996
q-2014

Question-(1)

An advantage of chain hash table over open addressing scheme is -

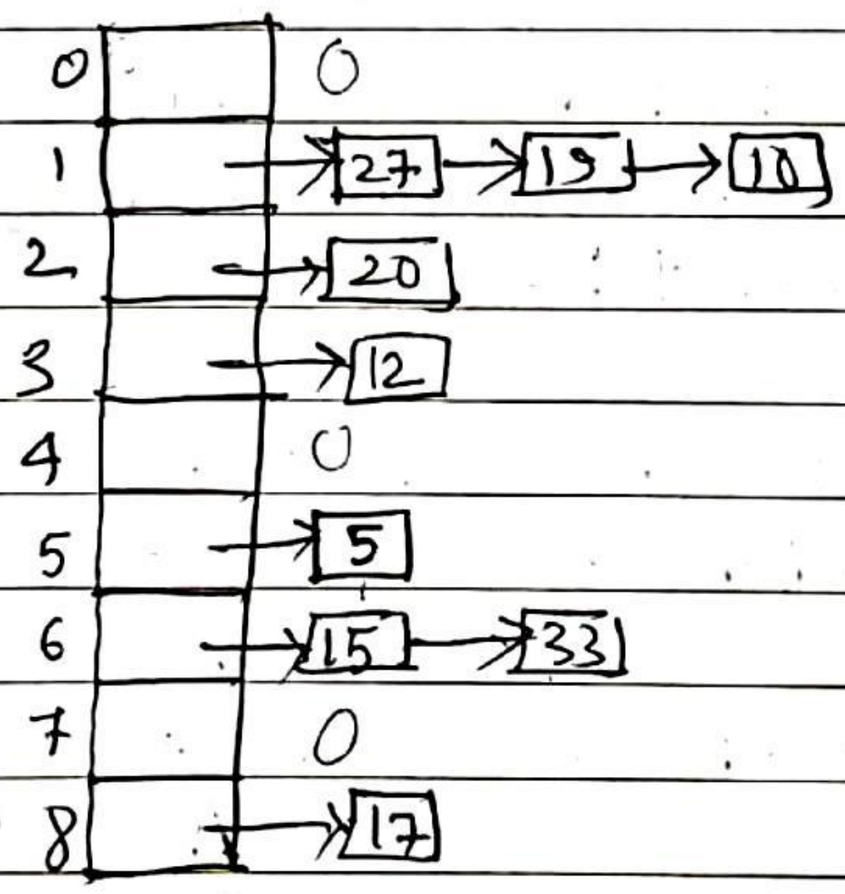
a) Worst case complexity of search is less.
b) space used is less.
✓ deletion is easier.
d) none of the above.

q-2014

Question-(2)

$h(K) = K \bmod 9$, hash table has 9 slots, chaining is used Keys: 5, 28, 19, 15, 20, 33, 12, 17, and 10. Then, max and min and average chain length in hash table.

→

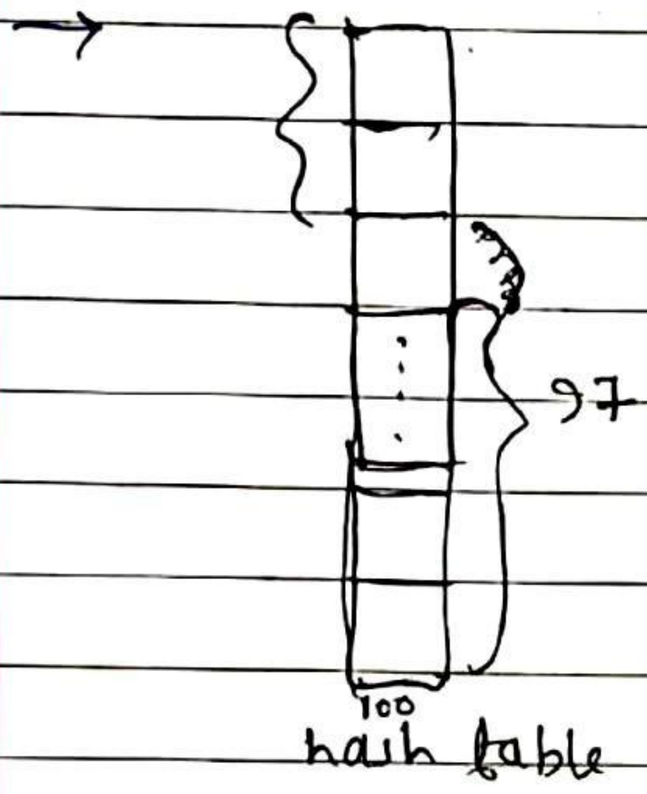


✓ max chain length → 3
✓ min " " → 0
✓ avg " " → $\frac{0+3+1+1+0+1+2+0+1}{9}$ → 1.

gate-2014

Question - (3)

Consider a hash table with 100 slots. collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 3 slots are unfilled after the first 3 insertions?



$$\left(\frac{97}{100}\right)\left(\frac{97}{100}\right)\left(\frac{97}{100}\right)$$

gate-1997

Question - (4)

Consider a hash table with 'n' ~~buckets~~ buckets, where chaining is used to resolve collisions. The hash function is such that the probability that a key value is hashed to a particular bucket is '1/n'. The hash table is initially empty and 'K' distinct values are inserted in the table.

a) What is the probability that bucket no. 1 is empty after K insertions.

b) What is the probability that no collision has occured in any of the 'K' insertions?

c) What is the prov that first collision occurs at the K^{th} insertion.

(a) → $1/n$

$(1 - 1/n)$

$= \left(\frac{n-1}{n}\right)$

$\left(\frac{n-1}{n}\right)\left(\frac{n-1}{n}\right) \cdots \left(\frac{n-1}{n}\right) = \left(\frac{n-1}{n}\right)^k$.

(b) $\left(\frac{n}{n}\right)\left(\frac{n-1}{n}\right)\left(\frac{n-2}{n}\right) \cdots \left(\frac{n-(k-1)}{n}\right)$

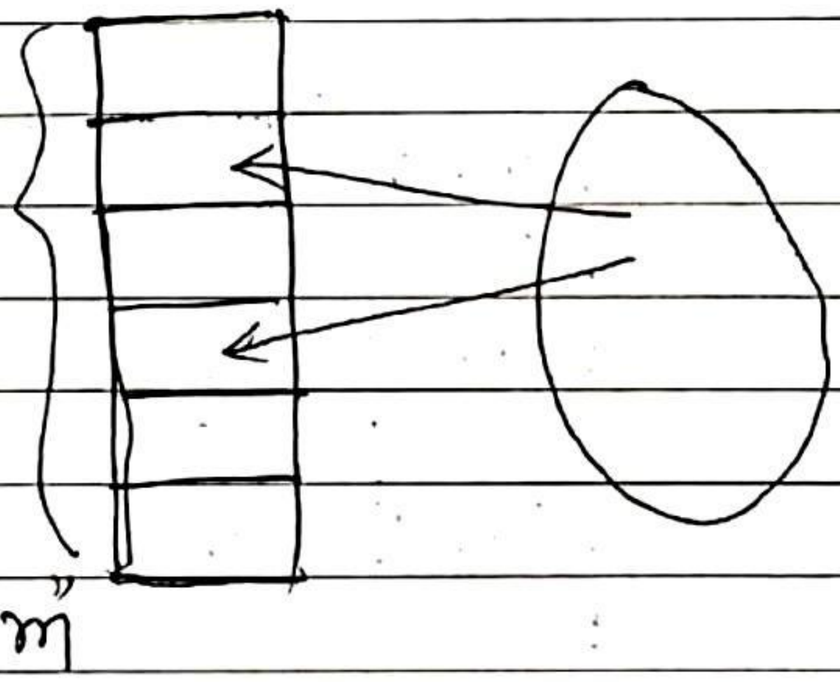
1^{st} ele 2^{nd} ele 3^{rd} ele ... k^{th} element.

(c) $\frac{(k-1)}{n}$.

*. Collision Solving Technique: Open addressing
or (closed hashing)

load factor, $\alpha = \frac{n}{m}$

$0 \leq \alpha \leq 1$



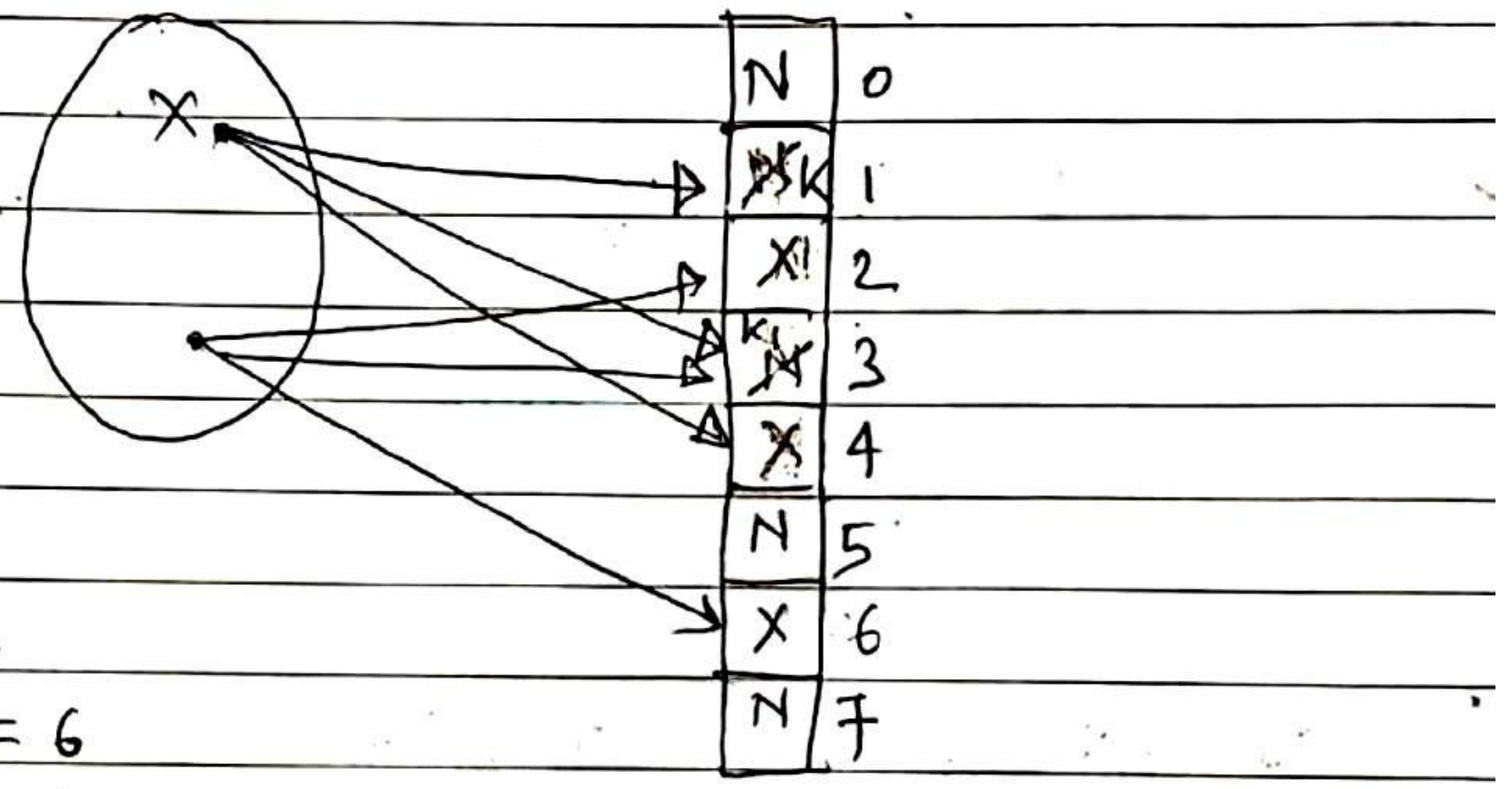
$h: U \to \{0, 1, \cdots, m-1\}$

$h: (U \times \{0, 1, 2, \cdots m-1\}) \to \{0, 1, \cdots, m-1\}$.

probe Sequence: try to
Whenever I Insert a key then follow a sequence of examinations, In case if you find any empty cell In this sequence you are going to Insert the element there, if you don't find any ~~sequence~~ empty cell In that sequence then declared

that entire table is full, you are not able to Inserted.

Worst time take to searching - O(n)



$h(K) = 1$

$h(K_1) = 2$

$h(K_{1,1}) = 6$

$h(K_{1,2}) = 3$ ← no. of collision

Searching for K_1

$h(K_1, 0) = 2$ ✓

$h(K_1, 1) = 6$ ✓

$h(K_1, 2) = 3$ ✓

$h(K_1, 3) = 4$

$h(K_1, 4) = 5$

$h(K_1, 5) = 1$

$h(K_1, 6) = 7$

$h(K_1, 7) = 0$

→ In case of full hash table and you are going to unsuccessfull search the time complexity for a search is - O(n).

In Average case time complexity - O(1).

Insert an new element K_1,

$h(K_1, 0) = 2$ ✓

$h(K_1, 1) = 6$ ✓

$h(K_1, 2) = 3$ ✓

$h(K_1, 3) = 4$ ✓

$h(K_1, 4) = 5$ ✓

$h(K_1, 5) = 1$

$h(K_1, 6) = 7$

$h(K_1, 7) = 0$

0	N
1	N
2	a
3	c
4	d
5	K_1
6	b
7	N

→ We are going to stop either when we find the searching element or empty cell.

Then Delete an element 'c' from table.

after deleting c search for K_1.

0	N	
1	N	
2	a	
x3	N	→ Stop searching here.
4	d	and declare
5	K_1	that K_1 is not
6	b	present even
7	N	When K_1 is present.

→ In case of deletion open addressing creat such a problem.

To solve this problem =

we are going to mark deleted place in such a way that it will indicate that some element are earllyer there but now it is deleted.

0	N
1	N
2	9
3	D → D indicate that, earlier there was
4	d an element but
5	(K1) now it deleted.
6	b
7	N

- Various technique which used in Open addressing =

(i) Linear probing :

hash function =

$h : U \to \{0, \cdots, m-1\}$

$h(K) = a$

$\boxed{h'(K,i) = (h(K)+i) \bmod m}$

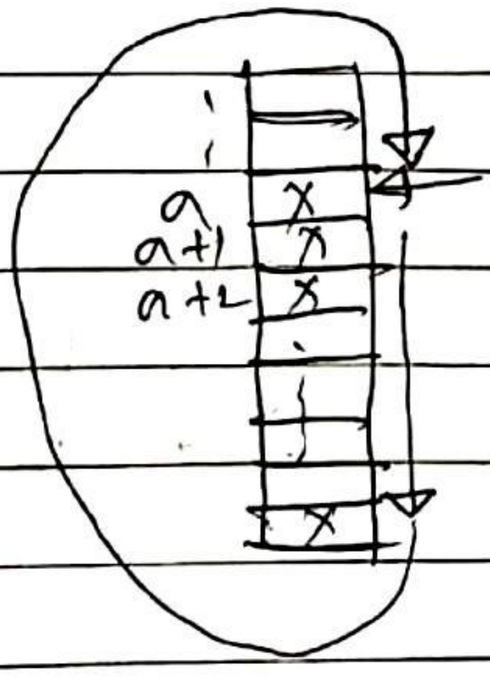
$h'(K,i) = (h(K)+1) \bmod m$

$= (a+1) \bmod m.$

$h'(K,2) = (h(K)+2) \bmod m$

$= (a+2) \bmod m.$

$h'(K,3) = (h(K)+3) \bmod m = (a+3) \bmod m$



Probe sequence :

$(0, 1, 2, 3, \cdots, m-1)$ always.

$(1, 2, 3, \cdots, m-1, 0)$ search for

$(2, 3, 4, \cdots, m-1, 0, 1)$ 'm' element

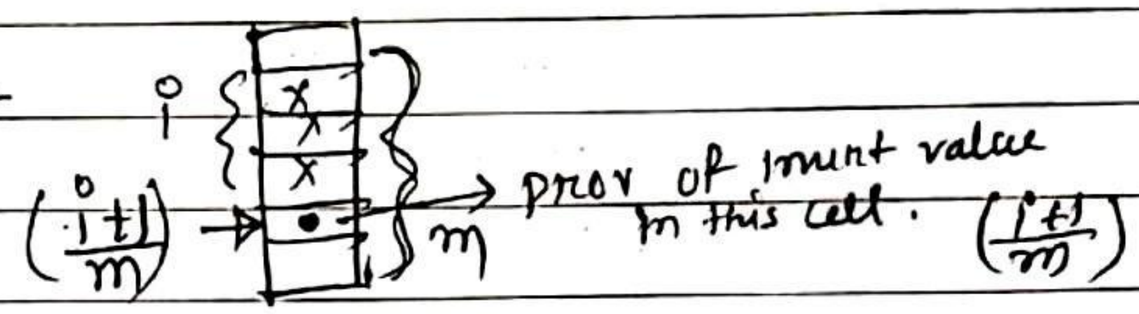
$(3, 4, 5, \cdots, m-1, 0, 1, 2)$ to insert element.

so, probe sequence is possible is "m".

Secondary Clustering =

In case ~~or~~ two elements first probe is found in same value then both of them ~~follow~~ going to follow same probe sequence, this call secondary clustering.

Primary clustering =



~~Average~~
dis
adv: Linear probing suffer two things (Secondary and primary clustering)

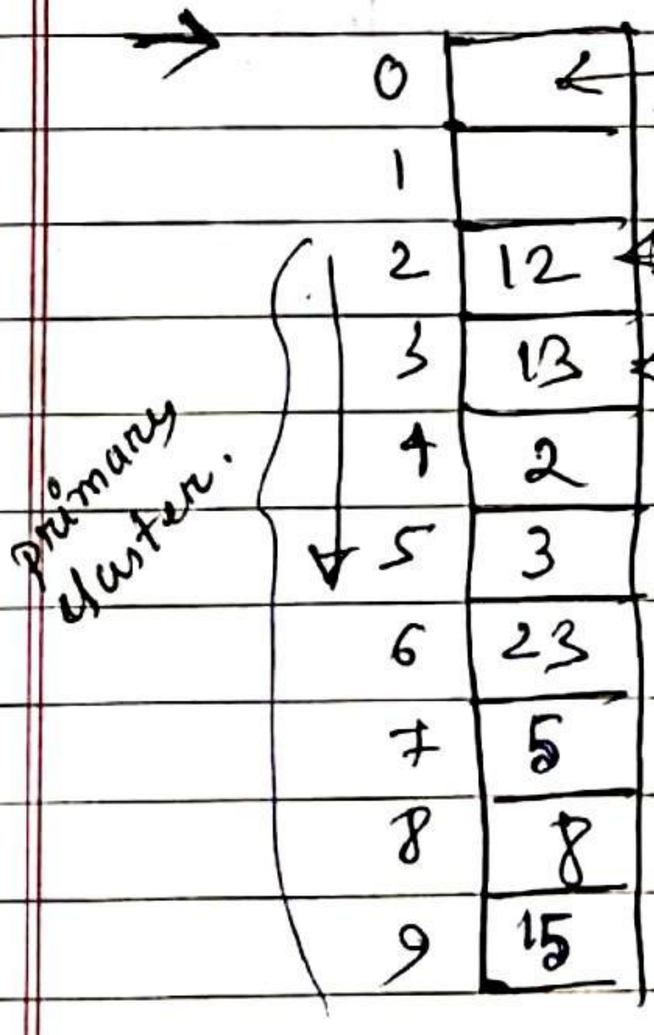
[Average ~~seek~~ time taken to search an element = O(2.5).
Worst case — O(n).

gate 2008 **Question — (1)**

Key: 12, 18, 13, 2, 3, 23, 5 and 15.

HST = 10 (Hash table size)

h(K) = K mod 10 and linear probing. What is resultant hash table?



$h'(k,i)$ ~~$h'(1,0)$~~ $= (h(k) + i) \bmod 10$

$h'(2,0) = (h(2) + 0) \bmod 10$
$= (2 + 0) \bmod 10$
$= 2$

$h'(2,1) = (2+1) \bmod 10 = 3$
$h'(2,2) = (2+2) \bmod 10 = 4$

gate-2018

Question-(2)

HTS = 11 (0-10) using

$h(K) = K \bmod 11$ & linear probing method.

Keys : 43, 36, 92, 87, 11, 4, 71, 13, 14.

What is the Index into the last record is inserted?

0	87	← 11
1	11	
2	13	
3	36	← 14
9	92	← 9
5	4	← 71
6	71	
7	14	
8	4	
9		
10	43	← 87

→ in $+^{th}$ Index last record is inserted.

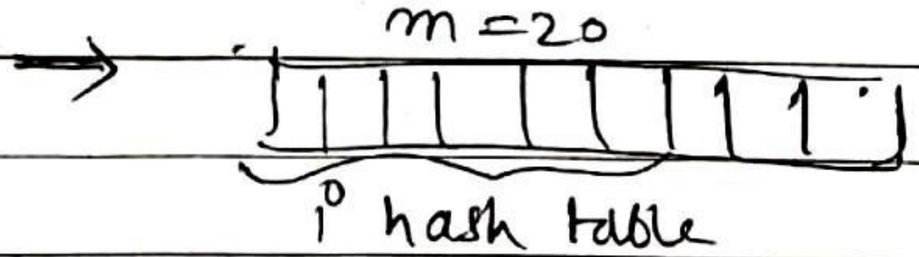
probability that this cell get nex insertion $\frac{2}{11}$.

gate-2007

Question-(3)

Consider a hash function that distributes keys uniformly. The hash table size is 20. After hashing of how many key, will the probability that any new key hashed collides with an existing one exceed 0.5.

a) 5 b) 6 c) 7 d) 10.



$\frac{i^0}{m} > 0.5$

$\frac{i^0}{20} > 0.5 \Rightarrow i^0 > 10$

gate 2010

Question - (1)

$h(K) = K \bmod 10$, linear probing.

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

A) After insertion, which will be look like give sequence —

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.

B) How many insertion sequences are possible, to get the give table sequence —

→ (42 – 23 – 34) – 52 – 33

$3!$

46 can take place any of the blank space.

$3! \times 5 = 30$ (insertion) sequence possible.

(ii) Quadratic probing =

$$h'(K,i) = (h(K) + c_1 i + c_2 i^2) \bmod m.$$

ex:

$m = 10$ (size of hash Table)
(0 --- 9)

$c_1 = 1, c_2 = 1$

$h(K) = K \bmod 10$.

$h(K_1) = 1$

$h'(K_1, 1) = 1 + 1 \cdot 1 + 1 \cdot 1^2$
$= 3$

$h(K_1, 2) = 1 + 2 + 4$
$= 7$

$h(K_1, 3) = 1 + 3 + 9$
$= (13) \bmod 10 = 3$

$h(K_1, 4) = 1 + 4 + 16 = 21 \bmod 10$
$= 1$

0		
1	X	←
2		
3	X	←
4		
5		
6		
7	X	
8		
9		

"m"

~~After 10 probs~~

In this case, you are not able to examin all the location ~~by using 10 probes~~ of hash table. This is the problem with quadratic probing.

You should ~~from~~ choose c_1, c_2 and m value ~~very well~~ Carefully. Other wise it ~~will~~ is going to lead problems.

probing sequence -

K_1 → (1, 3, 7, 3, ...)

K_2 → (1, 3, 7, 3, ...)

→ secondary Clustering (problem).

→ If two keys happen to have the same initial probe location then they are going have similar probe sequence.

two keys

→ ∧ probe sequence will same when Initial probe is same.

different

→ The no. of ∧ probe sequence possible is - "m" -
(every probe sequence depends on initial probe sequence)
↳ "m" probe sequences possible.

best probing technique.

(iii) **Double hashing**

In Double hashing no. of probe sequence is possible is $= m^2$
(where "m" → size of hash table) $= O(m^2)$

$$h'(K) = (h_1(K) + i\,h_2(K)) \bmod m.$$

→ there are no secondary and primary clustering problems.

$(i)(h_2(K), m)$ — $< m$, $\neq$ time.

$h_2(K)$ → Odd, m → 2^K

→ hashing is ~~more~~ better compare to other DS. when the most populere operation is only searching. and (Data is static).