HASHING

→ Search time of an algo depends on the no of elements in the arrary.

→ Hashing add^{n} is a search technique which is independent of the no of elements. i.e Search time is independent of no of ele

→ Mainly it is used for file management.

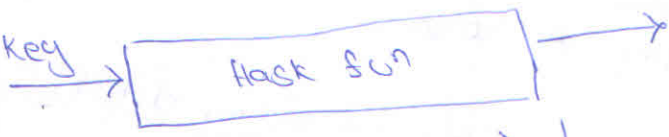
Consider

F :→ File of n records with a set of K of keys which uniquely determines the records in F.

T :→ A table which maintains record of F. T is having m memory loc (hash Table)

L :→ Set of memory add^{n} of location in T.

Hashing is H: K → L



Hashing is H: K → L

H is a fun which converts each key value of K into one add^{n} ∈ L. H is known as hash fun.

H(K) = a

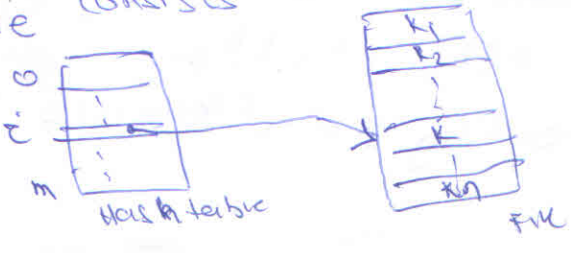
Where H is hash fun a → hash value.

K → key

The key K can be stored at arrray index a.

The process of generating add from keys is called hashing.

→ (Each entry of hash table consists of key and associated ~~record~~ index).



→ Criteria for hash fn

→ H should be easy to compute / quick to compute
⇒
→ H should uniformly distribute the hash addr throughout the set L such that there should be less / minimum no of collision.

Types of hash fn

(1) Division Method :→ Choose a no m larger than the no n of keys in K. usually m is prime no or a no without small divisors.

$H(K) = K \pmod m$ or

$H(K) = K \pmod m + 1$.

(2) Mid square method :→ The key K is squared and then middle 2 digits are chosen.

$H(K) = l$.

where l is obtained by deleting digits from both ends of K^2.

Disadv - Takes more time.

Adv :- Gives good result for uniform distribution of keys.

(3) Folding Method :- The key K is partitioned into no of parts $K_1, K_2 \cdots K_r$ where each part except last has same no of digits.

$H(K) = K_1 + K_2 \cdots + K_r$

Leading digits carries are ignored if any.

Types :- Pure Folding (All parts are added as it is)

Fold shifting (Even no parts are reversed before add^n)

Fold boundary (First and last parts are reversed before add^n)

Ex - In company with 68 emps each emp assigned a unique 4 digit no. L consists of 100 2 digit addr 00, ..., 99. Apply hash fn to each of following emp no.

3105, 6448, 3345

(a) Division Method :- Choose a prime no m close to 99. If memory addr start with 1 then

Let m = 97

H(3105) = 1

H(6448) = 46

H(3345) = 47.

H(3105) = 2

H(6448) = 47

H(3345) = 48

(b) mid Square Method

k: 3105

k^2: 9641025

H(k): 41

6448

41576704

76

3345

11189025

89

(c) Folding Method

H(3105) = 31+05 = 36

H(6448) = 64+48 = 12

H(3345) = 33+45 = 78

Foldshifing

H(3105) = 31+50 = 81

H(6448) = 64+84 = 55

H(3345) = 33+54 = 77

ex

k:	152756	5499025	11943936
Purge folding	01+52+27+56 = 136 = 36	05+49+90+25 = 169 = 69	11+94+39+36 = 180 = 80
fold shifting	10+52+72+56 = 190 = 90	50+49+09+25 = 133 = 33	11+94+93+36 = 234 = 34
fold boundary	10+52+27+65 = 154 = 54	50+49+90+52 = 241 = 41	11+94+39+63 = 207 = 07

Collision

if for 2 diff keys K_1 and K_2 $H(K_1) = H(K_2)$ then it is known as collision.

ex Suppose hash table is of size 10. Suppose set of keys are 10, 19, 35, 43, 62, 59, 31, 49, 77, 33.

H → ① Add digit ② Take digit at unit place.

K	I
10	1
19	0
35	8
43	7
62	8
31	4
49	3
77	4
33	6

H: K → I

0	19
1	10
2	
3	49
4	59, 31, 77
5	
6	33
7	43
8	35, 62

Hash table

59, 31 and 77 mapped to same index of hash table.

Collision Resolution

Load factor = $\frac{\text{No of keys in K}}{\text{No of add}^{n} \text{ in L}}$.

Efficiency of hash fn with collision resolution procedure is measured by avg no of key comparisions (probe) needed to find the location of record with a given key K.

There are 2 types of collision resolution technique.

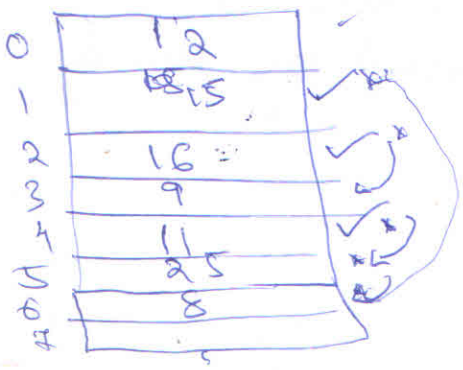
1- open hashing (chaining)
2- closed " (Linear probing)

a) Linear Probing

if a new record R with key K needs to be added to hash table but H(K) = h is already filled.

→ The natural way to resolve that collision is to search hash Table T linearly for location T[h], T[h+1] ... Untill finding R or empty location. The above resolution tech is known as linear probing.

ex H(K) = K%7 Key value 15, 11, 25, 16, 9, 8, 12



ex- A hash Table has 11 memory loc. The file F consists of 8 records A, B, C, D, E, X, Y, Z with following hash add^{n}

Record:	A	B	C	D	E	X	Y	Z
H(K):	4	8	2	11	4	11	5	1

8 records ~~en~~ are entered into T. Then F appear in memory as follows

T	1	2	3	4	5	6	7	8	9	10	11
add^{n}	X	C	Z	A	E	Y	→	B	—	—	D

Although y is only record with H(K)=5, the record is not assigned to T[5] since T[5] has already filled by E because of previous collision at T[4]. Similarly z does not appear in T[1].

Avg no of probe for a successful search

$$S = \frac{1+1+1+1+2+2+2+3}{8} = 13/8 \simeq 1.6$$

Avg no probes for an unsuccessful search

$$= \frac{7+6+5+4+3+2+1+2+1+1+8}{11} = \frac{40}{11} = 3.6$$

Sum the no of probes to find an empty location for each of 11 location.

→ one disadvantage of linear Probing is that records tend to form cluster (ie next to one another). Such a clustering increases avg search time for a record.

b) Quadratic Probing

It is a collision resolution method. Suppose a new record R with key K needs to added but $H(K) = h$ is already filled then search hashTable in location $T[h]$, $T[h+1], T[h+4], T[h+9] \ldots T[h+i^2]$

c) Double hashing choose a second hash fun H' for solving a collision.

$H(K) = h$ & $H'(K) = h' \neq m$ (no of addr in T)

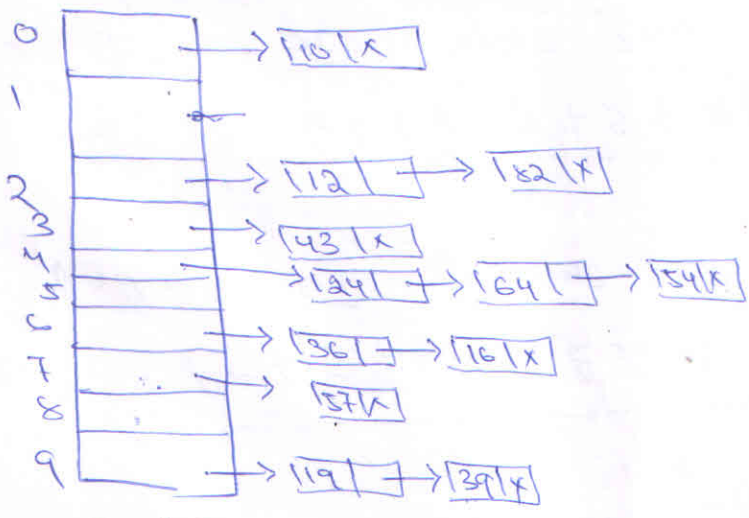
Then search loc

$T[h], T[h+h'], T[h+2h'], T[h+3h'], \ldots$

2. Chaining (Open Hashing)

Chaining method require 2 tables. (1) first table T contain the records in F and a link to the records of same hash address. Second table list which will contain pointers to linked list in T.

The chaining method maintains the chain of elements which have same hash addresses. Here hash table can be considered as an array of pointers. Each pointer points to a single linked list and elements which have same hash addresses will be maintained in the linked list.



Hash Table (T) (Open Hashing)

Generally this technique involves 2 operation.
(1) creation of a good hash fn for getting hash key value in the hash table.
(2) Maintain the elements in the linked list which is pointed by pointer available in a hash table.

→ For a given key value, hash add^{n} is calculated, it then searches the linked list pointed by the pointers at that location. if the element is found it returns the pointer to node containing that key value else insert at the end of that list.

Disadv - Require extra storage space for link field.

Ex - A table T has 11 memory loc. The file F consists of 8 records A, B, C, D, E, X, Y, Z. With following hash addr

Record:	A	B	C	D	E	X	Y	Z
H(K):	4	8	2	11	4	11	5	1

8 records are entered into T. Then F appear in memory as follows

addr	1	2	3	4	5	6	7	8	9	10	11
T	X	C	Z	A	E	Y	-	B	-	-	D
					*	*					*

Although Y is only record with H(K) = 5, the record is not assigned to T[5] since T[5] has already filled by E because of previous collision at T[4]. Similarly Z does not appear in T[1].

Avg no of probes for a successful search.

$$S = \frac{1+1+1+2+2+2+2+3}{8} = \frac{13}{8} \simeq 1.6$$

Avg no of probes for an unsuccessful search.

$$= \frac{7+6+5+4+3+2+1+2+1+1+8}{8} = \frac{44}{11} = 3.6$$

Sum the no of probes to find an empty location for each of 11 location.

→ One disadvantage of linear probing is that records tend to form cluster (i.e appear next to one another) When load factor is greater than 50%. Such a clustering increas the avg search time for a record.

-> $(H(K)+i^2)\%m$
for $i=1,2\ldots$

b) Quadratic Probing

It is a collision resolution method. Suppose a record R with key K has hash addr $H(K)=h$. Then if there is collision at loc h instead of searching addr h, h+1, ... linearly search the addr

h, h+1, h+4, h+9 ..., $h+i^2$ + ...

Suppose a new record R with key K needs to added but $H(K)=h$ is already filled we will search the table in the locations T[h], T[h+1], T[h+4] T[h+9], T[h+16] ... $T[h+i^2]$

c) Double hashing :- choose a second hash fn H' for solving a collision.

$H(K)=h$ & $H'(K)=h' \neq m$ (no of address in T)

Then Search the locations

T[h], T[h+h'], T[h+2h'], T[h+3h'] ....

Disadv of Linear Probing :->

if we will delete an element from the hashtable from the location T[r] and for a key K, $H(K)=r$ while Searching we find T[r] is empty. But that does not indicate that searching element not found.