Hashing → is useful for searching purpose. ①

Linear Search – O(n)

	0	1	2	3	4	5	6	7	8	9
A	8	5	12	6	15	9	4	3	7	10

Binary Search – O(logn)

A	3	4	5	6	7	8	9	10	12	15
	0	1	2	3	4	5	6	7	8	9

elements must be Sorted.

we want algorithm which takes O(1)

Hashing.

Keys – 8, 3, 13, 6, 4, 10

A				3	4		6		8		10			13	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

if a list of ele are given and array is given.

The value is also considered as indexed.

EX: 8 is stored in 8

3 is stored in 3.

Now search is directly go to index and search.

it have 50

array must be 50

lot of space is wasted.

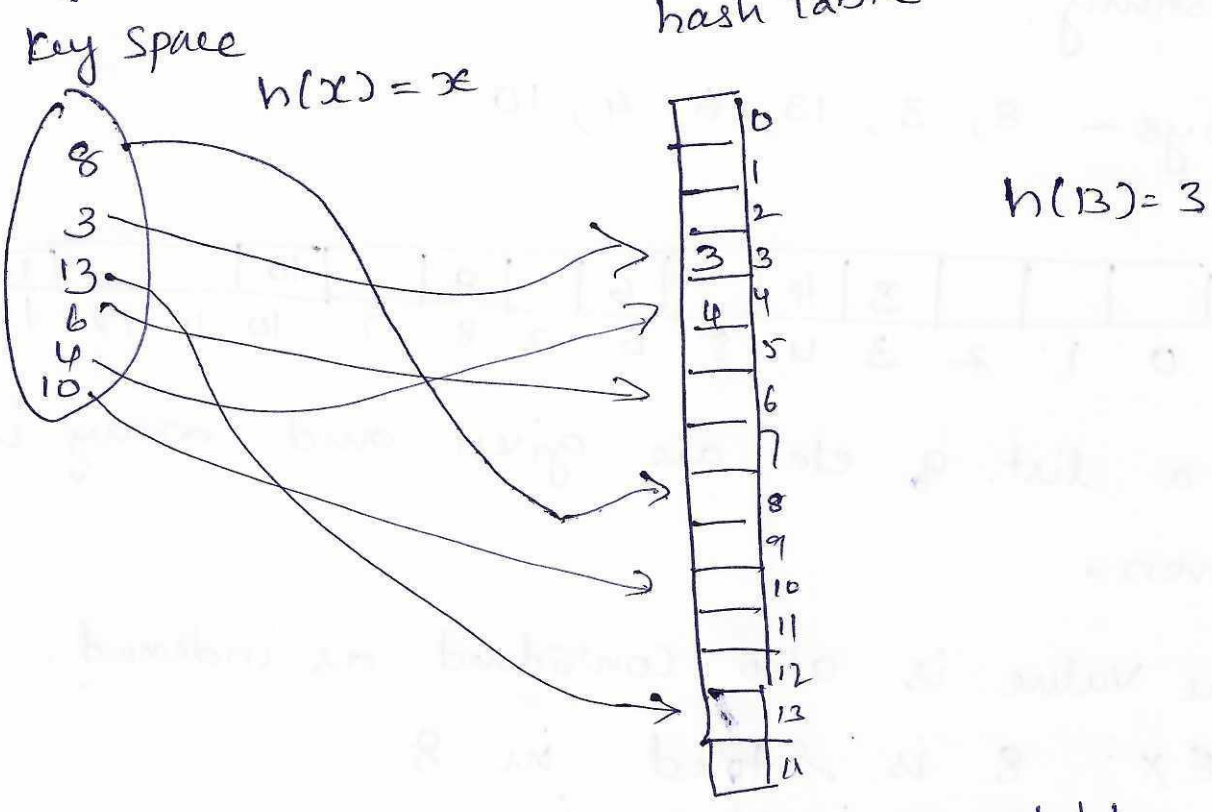
To improve we use mathematical model.

for hashing based on functions.

so let us see next how it is model

The list of elements what we have we call them as key space and the place we have to store them is called hash table.

and we say that there is some function say ~~H(x)=x~~ $H(x) = x$ which mapping



elements from key space to hash table.

$H(x) = x$ is hash function which is

- mapping key elements to hash table and this ideal hash function.

if I have key = 50 index must be 50 2
hash table must be lengthy waste of space.

$H(x) = x$ — one to one fnction.

many-one — any other.

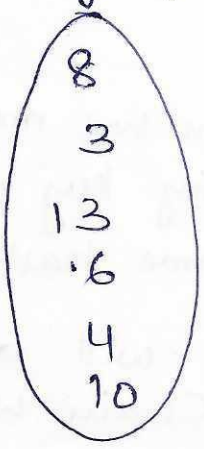
$h(x)$ is responsible for mapping to table.
modify the fnction to efficiently use.

we will use.

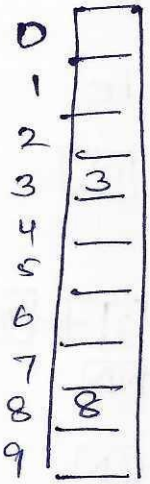
fnc

$H(x) = x \,\%\, 10$ Size & hash table

Key space.



hash table size 10 (ex)



$h(x) = x \,\%\, \text{Size}$

$x \,\%\, 10$

$8 = 8 \,\%\, 10 = 8.$

$3 = 3 \,\%\, 10 = 3$

$13 = 13 \,\%\, 10 = 3$ — collision.

Two or more ele are mapped to same index

We need to resolve collisions

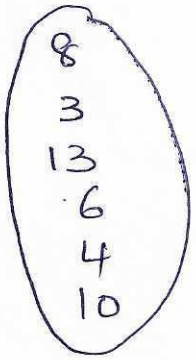
Collision Resolution Methods

1. Open Hashing
   Chaining

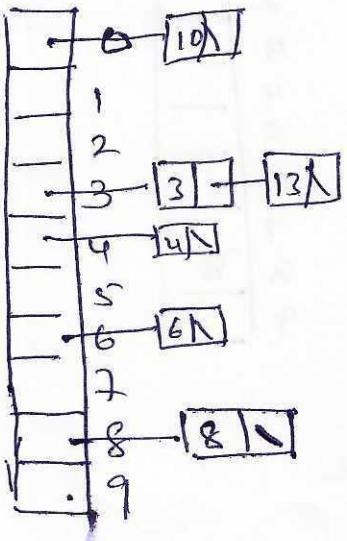
2. closed hashing
   Open addressing
   1. Linear Probing
   2. Quadratic Probing

1. Chaining.

Key Space



hash table



- when we map any key using same hash function we will add a chain which is linked list.

$h(x) = x \,\%\, Size.$

$8 \,\%\, 10 = 8.$

when there is a collision we insert a key element at the same index in a chain

When we want to search an element let us say 13 then we should use hash function H

$H(13) = 13 \% 10 = 3$

So go to index 3 and search for key in the chain.

They may be more than one element. It should search entire chain

time complexity will be more than 1.

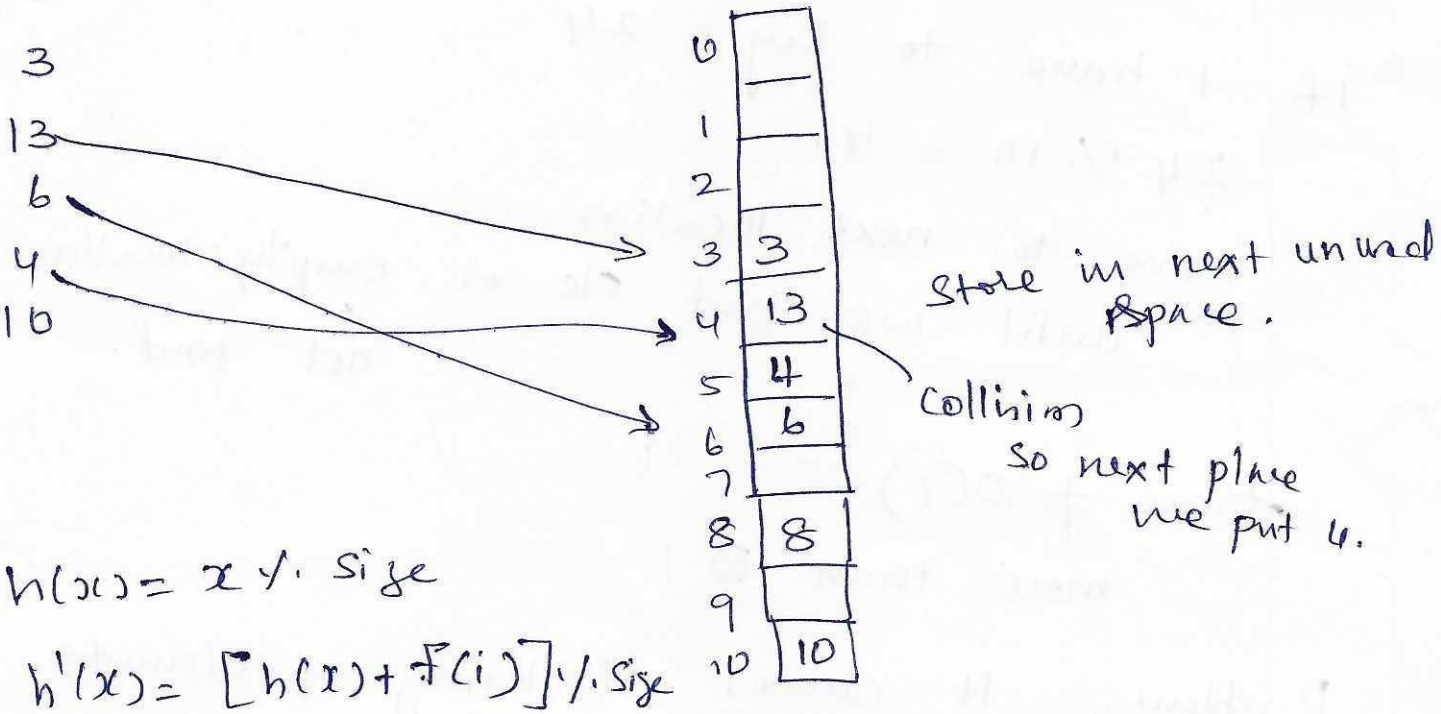
Closed Hashing.

Linear probing

8. $h(x) = x \% size$.

3
13
6
4
10

hash table



$h(x) = x \% size$

$h'(x) = [h(x) + f(i)] \% Size$

$f(i) = i$ where i takes value

$i = 0, 1, 2 \ldots$

When we use 13 $h'(13) = [h(13) + f(0)] \% 10$

3 - collision

$$h'(13) = [h(13) + f(1)] \% 10$$

$$= \underline{\underline{4}}$$

So 13 is stored in 4.

if there is a collision again we take next value of i

When we want to search for element

we use $h(x) = x \% size$

4 not present

next look at next location

it is present in 5th location.

if I have to key = 24

$24 \% 10 = 4$.

move to next location.

until we find ele or empty location

not found.

~~time~~ $\neq O(1)$

more than ⊗ 1

Problem: It causes clustering of elements at consecutive locations

— lot of time.

To avoid clustering of elements

Linear probing.

12, 44, 67, 22, 58, 64, 52, 89 (4)

$h(x) = (h(x) + i) \bmod 10$

$i = 0, 9$.

0	
1	
2	12
3	22
4	44
5	64
6	52
7	67
8	58
9	89

$h(12) = (h(12) + 0) \% 10$

$2 + 0 \% 10$

2

$h(44) = 4 + 0 \% 10$

4.

$h(67) = 7 + 0 \% 10$

$= 7$

$h(22) = 2 + 0 \% 10$

$= 2$ collision.

$(2 + 1) \% 10$

$3 \% 10$

$= 3$

$52 = 2 + 0 \% 10$

2

$2 + 1 \% 10$

$3 \% 10$

$= 3$

$2 + 2$

4

$2 + 3 = 5$

$2 + 4 = 6 \% 10$

$= 6$.

$h(58) = (8 + 0) \% 10$

8

$89 = (9 + 0) \% 10$

9

$64 = (4 + 0) \% 10$

$= 4$.

$4 + 1 \% 10$ $5 \% 10 = 5$

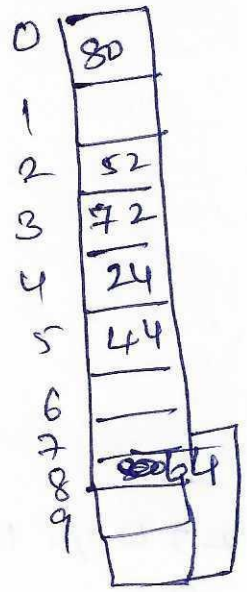
quadratic probing

$h(x) = x \bmod 10$

keys = {52, 24, 72, 80, 44, 64}

$h(x) = (h(x) + i^2) \bmod 10$

$i = 0$ to 9



$h(52) = (2 + 0) \bmod 10$

2.

$h(24) = (4 + 0) \% 10$

4.

$h(72) = (2 + 0) \% 10$

$= 2$. - collision.

$(2 + 1^2) \% 10$.

$3 \% 10$.

$= \underline{3}$.

$80 = (8 + 0) \% 10$

8.

$44 = (4 + 0) \% 10$

4 - collision

$(4 + 1) \% 10$.

$\underline{5}$

$64 = (5 + 0)$ $(4 + 0) \% 10$

4 - collision

$(4 + 1^2) \% 10$ - collision

$(4 + 2^2) \% 10$

$4 + 4 \% 10$

$8 \% 10$

$= 8$ - collision

We use quadratic probing.

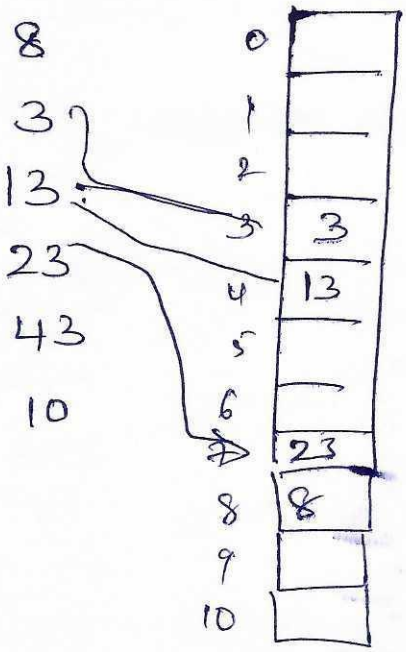
it is same as linear probe

$h(x) = x \,\%\, size$

$h'(x) = [h(x) + f(i)] \,\%\, size$

$f(i) = i^2$

$i = 0, 1, 2, 3, \ldots$



$h'(23) = [h'(23) + f(0)] \,\%\, 10$

$= 3$ collision.

$[h'(23) + f(1)] \,\%\, 10$

$3 + 1 \,\%\, 10$

4

Collision.

$[h[23) + f(2^2)] \,\%\, 10$

~~4 of~~.

$3 + 4$

$7 \,\%\, 10 = 7$

So place 23 in 7.

it is not close, it place away.

This will avoid clustering.