Hashing

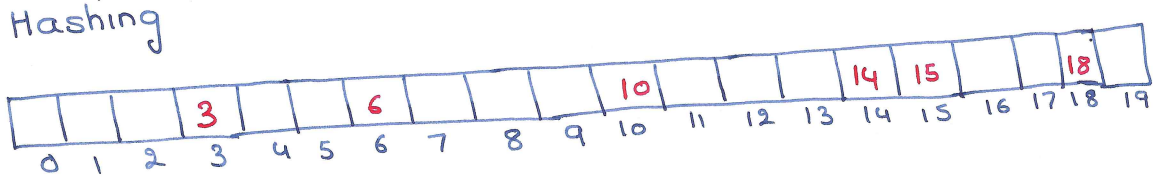
Hashing technique

Keys : 8, 3, 6, 10, 15, 18, 4.

1) Linear Search — O(n)
   ↳ No sorting is required

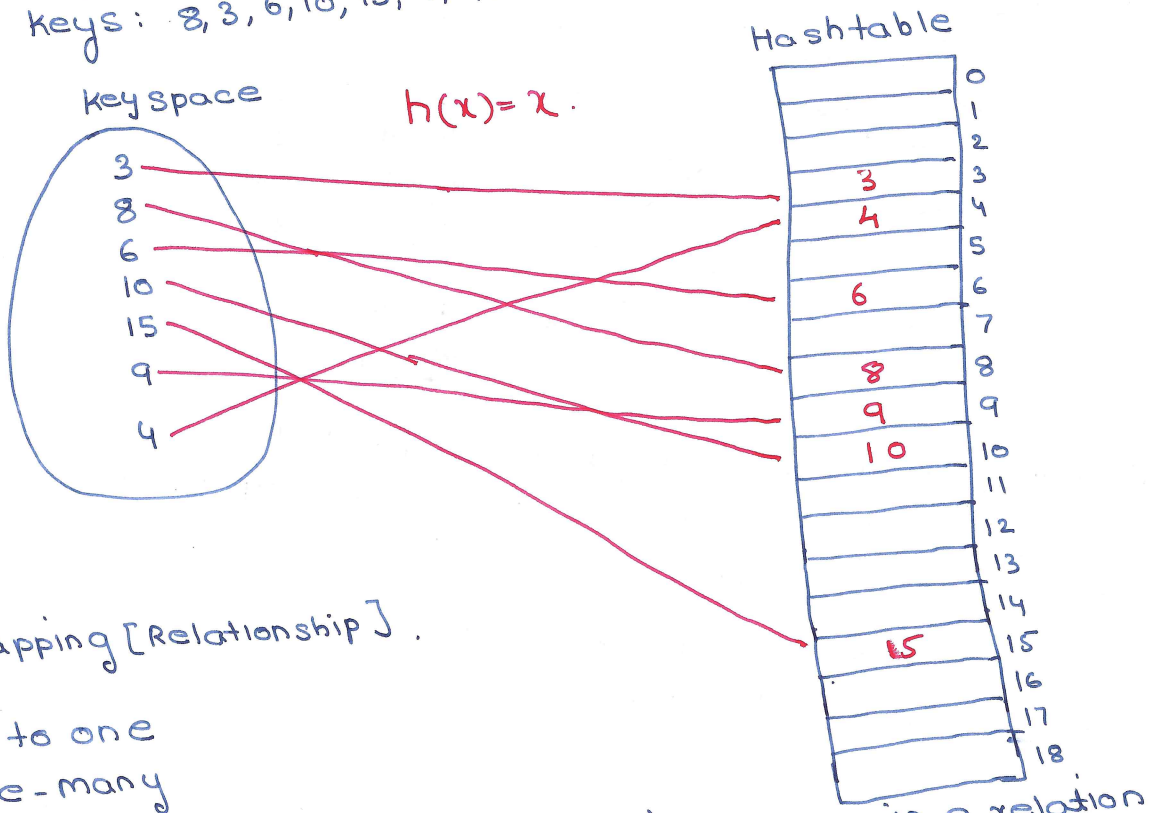
2) Binary Search — O(log n)
   ↳ Key should be sorted order.

3) Hashing



Hashing

Keys : 8, 3, 6, 10, 15, 9, 4.



mapping [Relationship].

✓ 1) one to one
2) one - many
✓ 3) many - one
4) many - many.

mapping domain to range is a relation

1 2 3 are function.

the searching is or insertion, we use hash function

* a function support is one-one
  many-many.

~~Hashing~~

~~Keys: 8, 3, 6, 10, 15, 9, 4.~~

~~Keyspace~~

Drawback of ideal hashing.

1) space.

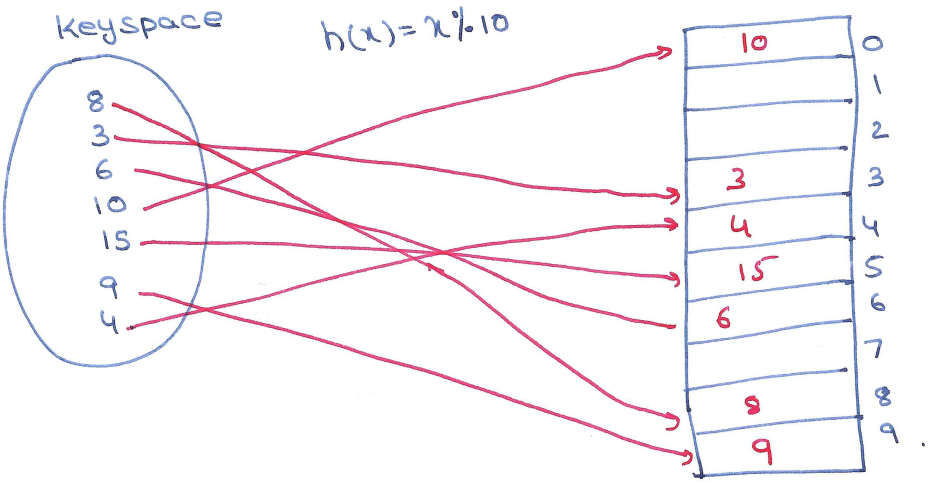
and the reason for this drawback is hash function as hash function is giving the mapping

↳ (i.e ideal hashing).

So to over come, let modify the hash function.

So that we can save space.

Keys: 8, 3, 6, 10, 15, 9, 4.



Drawback of modulan or (mod) hash function.

Now try to insert 25 in the key, which will map to 5,

So, 25 is mapped at index - 5,

Now 15 & 25 ⟶ are mapped at 5.

this is called, collision.

Collision:

open hashing

chaining

~~closed~~ closed hashing

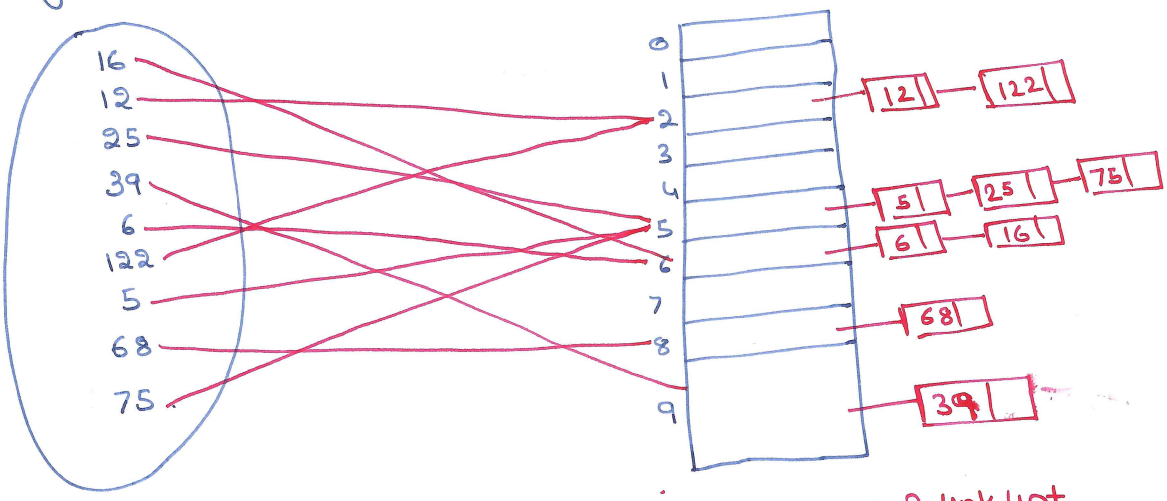
open addressing

1) Linear probing
2) Quadratic probing
3) Double hashing

chaining

Keys: 16, 12, 25, 39, 6, 122, 5, 68, 75 .

Key space .



This implementation is done, using array of linklist .

Search is
Key = 12,
use the hash function.

Key = 75 ✓
Key = 65 ✗
Key = 15 ✗

Analysis for search:
n = 100 [element].
size of hashtable = 10

$\lambda = n/size \rightarrow 100/10 = 10$.
↓
load factor (lamda) .

*Analysis in hash is done by loading factor.

Avg. Successful search

$t = 1 + \lambda/2$ {time taken for successful search, 2 it's average time}.

Avg unsuccessful search

$t = 1 + \lambda$ {time taken for unsuccessful search}.

deletion of key.

deletion is a node in linklist

another example:

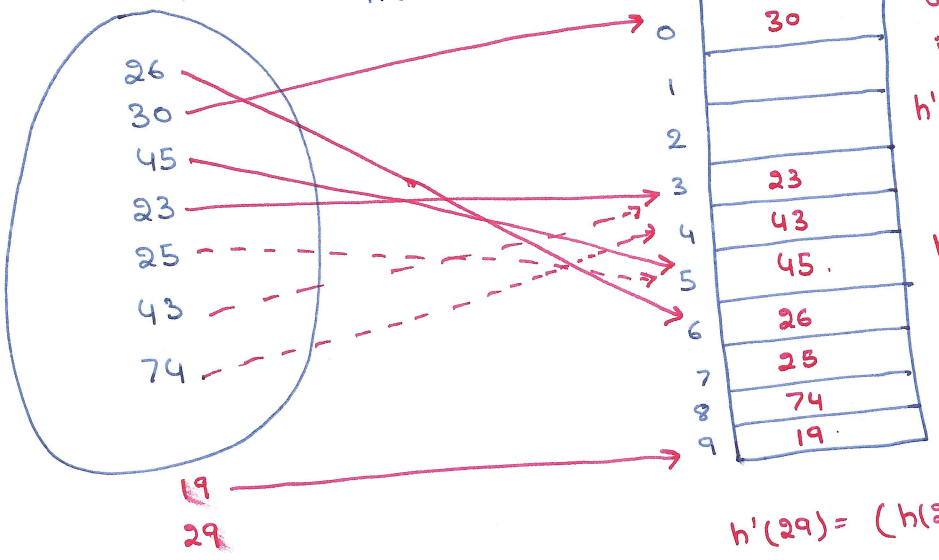
5, 35, 95, 145, 175, 265, 845, 105, 55, 25.

How to solve this problem.

Linear probing [open addressing].

Key space.

$h(x) = x\%.10$



$h'(x) = (h(x)+f(i))\%.10$
where $f(1) = i$
$i = 0, 1, 2, 3, \ldots$

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

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

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

$h'(29) = (h(29)+f(0))\%.10$
$= (9+0)\%.10$
$= 9$

$h'(29) = (h(29)+f(1))\%.10$
$= (9+1)\%.10$
$= 0$

$h'(29) = (h(29)+f(2))\%.10$
$= (9+2)\%.10$
$= 1$

* for searching, we use the same hash function.
* it take more time than constant

Search should stop when you find Empty space.

Key = 45 ✓, 74 ✓, 40 ✗

Anaylsis

loading factor $\lambda = n/size$

$\lambda = 9/10 = 0.9$

loading factor should be.

$\lambda \leq 0.5$

Avg successful search.

$t = 1/\lambda \ln(1/1-\lambda)$

Avg unsuccessful search.

$t = 1/1-\lambda$

disadvantage.

- space waste because of loading factor
- clustring

deletion of key in linear probing

key = 25

key = 74.

deletion should be rehashing

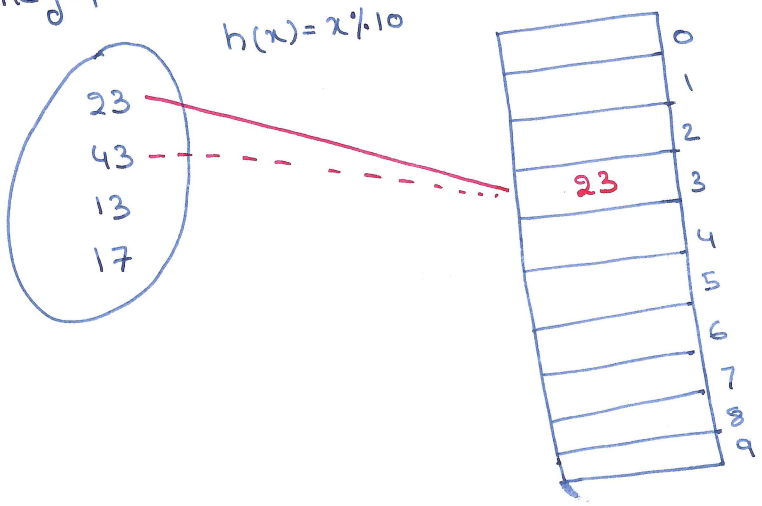
In linear probing

- deletion is not suggested.

Quadratic Probing

Key space

$h(x) = x\%10$



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

where

$f(i) = i^2$

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

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

$= (3+0)\%10$

$= 3$

$h'(43) = (h(43)+1)\%10$

$= (3+1)\%10$

$= 4.$

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

$= (3+0)\%10$

$= 3.$

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

$= (3+1)\%10$

$= 4$

$h'(13) = (h(13)+4)\%10$

$= (3+4)\%10$

$= 7$

Avg successful search:

$$\frac{-\log_e(1-\lambda)}{\lambda}$$

Avg unsuccessful search

$= 1/1-\lambda$

Anaylsis

loading factor λ = n/size

$\lambda = 9/10 = 0.9.$

loading factor should be $\lambda \leq 0.5$.

Avg successful search.

$t = 1/\lambda \ln(1/1-\lambda)$

Avg unsuccessful search.

$t = 1/1-\lambda$

disadvantage

↳ space waste because of loading factor

↳ clustring

deletion of key in linear probing

key = 25

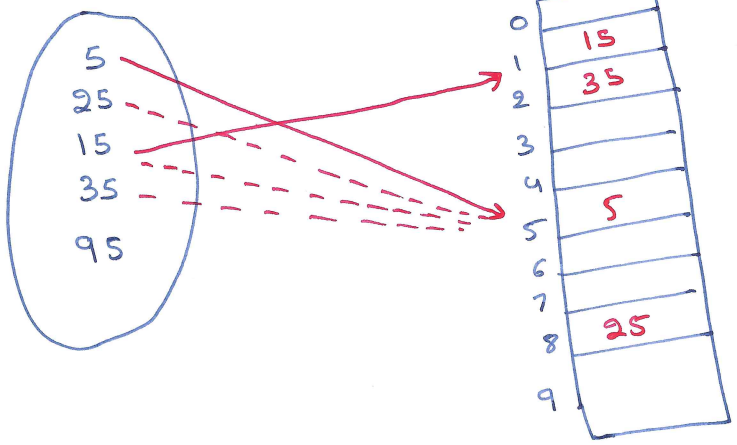
key = 74.

deletion should be rehashing

Double Hashing

$h_1(x) = x \% 10$

Keyspace



H.T.

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

$h_1(x) = x \% 10$

$h_2(x) = 7 - (x \% 7)$

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

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

* second hash function has two desired properties
  - It should not give index - zero
  - It should cover all the value in table.

So use a hash function is with prime number.

$h_1(25) = 25 \% 10 = 5$.

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

$h'(x) = (h_1(x) + i * h_2(x)) \% 10$
$= (5 + (1)3) \% 10$
$= (8) \% 10$
$= 8$

$h'(15) = (5 + 1(6)) \% 10$
$= (11) \% 10 = 1$

$h'(35) = (5 + 1(7)) \% 10$
$= 12 \% 10$
$= 2$.

Try 95, by yourself.