Programme : F. Y. B. Sc.
Subject : Computer Science
Semester : II
Course Code : CSC 102
Course Title : Data Structures
Unit 7 : Hashing
Module Name : Hash Table Reordering,Resolving collision by Open addressing

Module Number : 38

What is Rehashing?

- Rehashing is a collision resolution technique.
- In Rehashing technique the table is resized, i.e., the size of table is doubled by creating a new table.
- There are situations in which the rehashing is required.
  - When table is completely full
  - When insertions fail due to overflow.

Hash Table Reordering

There are several techniques of hash table reordering

- Ordered hash table method
  - The set of items that hash into the same location are maintained in descending order of the key.
- Brent's method
  - It involves rehashing the search argument until an empty slot is found.
  - Used to improve average search time for successful retrievals when double hashing is used.

Resolving collision by Open Addressing

- In hashing,hash function is used to compute the hash value for a key.
- When the hash value of a key maps to an already occupied bucket of the hash table,Collision occurs.

What is Open addressing(Closed hashing)?

- Open addressing,is a method in which hash collision is resolved by probing, or searching through alternative locations in the array until either the target record is found, or an unused array slot is found.
- In Open Addressing, all elements are stored in the hash table itself.

There are three methods of Open Addressing (Closed Hashing)

1. Linear Probing
2. Quadratic Probing
3. Double Hashing

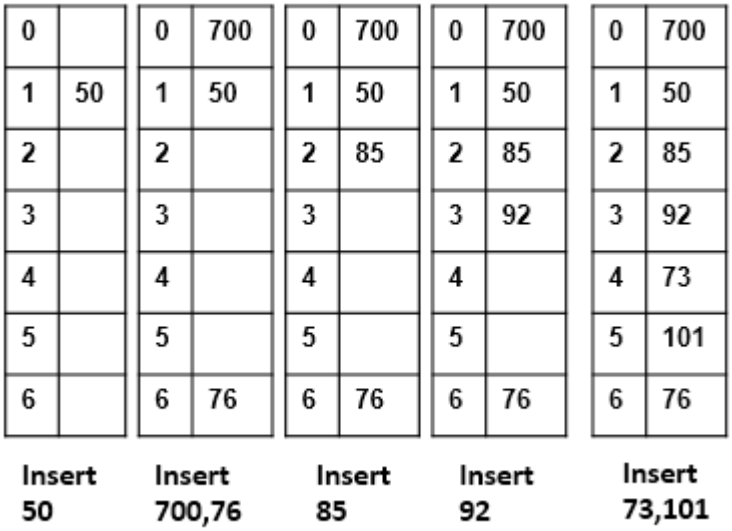
1. Linear Probing

- In linear probing, we linearly probe for next slot.
- The gap between two probes is 1.

- Let hash(x) be the slot index computed using a hash function and n be the table size
    - If slot hash(x) % n is full, then we try (hash(x) + 1) % n
    - If (hash(x) + 1) % n is also full, then we try (hash(x) + 2) % n
    - If (hash(x) + 2) % n is also full, then we try (hash(x) + 3) % n

  This process is repeated for all the values of i until an empty slot is found.

Example: let us take a table of size 7 and insert some keys in it taking a hash function h(key)=key% 7



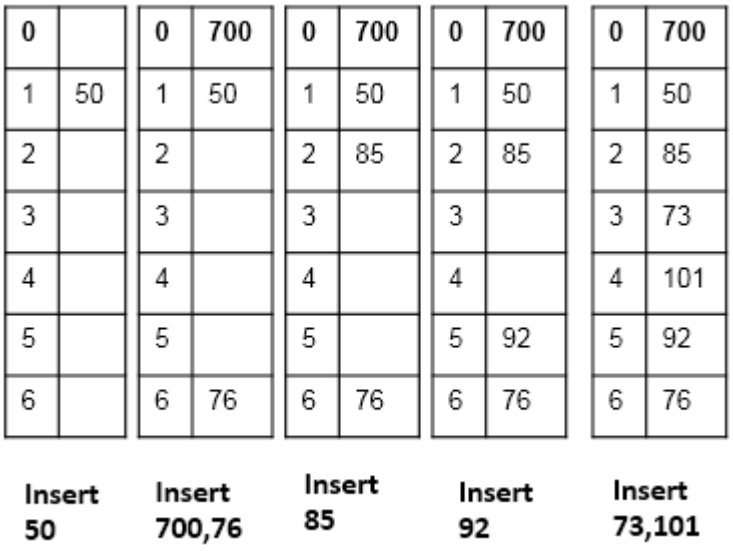
h(50)=50% 7=1
h(700)=700% 7=0
h(76)=76% 7=6
h(85)=85% 7=1
1+1%7=2
h(92)=92% 7=1
1+1%7=2
1+2%7=3
h(73)=73% 7=3
3+1%7=4
h(101)=101% 7=3
3+1%7=4
3+2%7=5

2. Quadratic probing

- We look for i^2'th slot in i'th iteration.
- Let hash(x) be the slot index computed using the hash function and n be the table size.
    - If the slot hash(x) % n is full, then we try (hash(x) + 1*1) % n.
    - If (hash(x) + 1*1) % n is also full, then we try (hash(x) + 2*2) % n.
    - If (hash(x) + 2*2) % n is also full, then we try (hash(x) + 3*3) % n.

This process is repeated for all the values of i until an empty slot is found.

Example: let us take a table of size 7 and insert some keys in it taking a hash function h(key)=key% 7



h(50)=50% 7=1
h(700)=700% 7=0
h(76)=76% 7=6
h(85)=85% 7=1
=1+1*1%7=2
h(92)=92% 7=1
=1+1*1%7=2
=1+2*2%7=5
h(73)=73% 7=3
h(101)=101% 7=3
=3+1*1%7=4

3. Double hashing

- In double hashing the increment factor is not constant as in linear and quadratic probing,but it depends on the key.
- The increment factor is another hash function hence the name double hashing.

The formula for double hashing can be written as:
H(k,i)=(h(k)+i h'(k)) mod Tablesize
The value of i varies from 0 to Tablesize-1 and h is the hash function,h' is the secondary hash function.
The search for empty locactions will be in the sequence:
h(k),h(K)+h'(K), h(K)+2h'(K), h(K)+3h'(K)….all mod Tablesize
keys:46,28,21,35,57,39,19,50
h(k)=key %11
h'(k)=7-(key%7)
h(46)=46%11=2
h(28)=28%11=6
h(21)=21%11=10
h(35)=35%11=2
h(57)=57%11=2

h(39)=39%11=6
h(19)=19%11=8
h(50)=50%11=6
H(k,i)=(h(k)+i h'(k)) mod Tsize
=2+1*(7-(35%7)%11
=2+1*(7)%11
=9

0						
1				39	39	39
2	46	46	46	46	46	46
3					19	19
4						
5						
6	28	28	28	28	28	28
7						50
8			57	57	57	57
9		35	35	35	35	35
10	21	21	21	21	21	21

Comparison Chart:

Parameter	Linear Probing	Quadratic Probing	Double Hashing
Primary Clustering	Yes	No	No
Secondary Clustering	Yes	Yes	No
Number of Probe Sequence (n = size of table)	n	n	n^2
Cache performance	Best	Lies between the two	Poor