HASHING NOTES

A hash function is a mathematical formula which, when applied to a key, produces an integer which can be used as an index for the key in the hash table.

TYPES -

- Division method
- Multiplication method
- Mid - square method
- Folding method

DIVISION METHOD -

This method divides x by M and then uses the remainder obtained.

h(x) = x mod M

It is always wise to choose M as an even number

Drawback: Consecutive keys map to consecutive hash values

Example:

Calculate the hash values of keys 1324 and 5642 (Use M as 97)

MULTIPLICATION METHOD -

It works in steps :

1. Choose A as 0 < A < 1
2. Multiply key k by A
3. Extract fractional part of kA
4. Multiply result of kA by size of hash table (m)

Knuth has suggested the best value to be used for **A as 0.618033**

Example:
Given a hash table of size 1000, map key 12345 to an appropriate location in the hash table

MID - SQUARE METHOD -

It works in steps:

1. Square the value of the key
2. Extract the middle r digits of the result obtained in 1.

Example:
Given a hash table of size 100 memory locations, map key 1234 and 5642 to an appropriate location in the hash table.

FOLDING METHOD -

It works in steps:

1. Divide key k value into number of parts, where each part has equal digits except the last part can have lesser digits.
2. Add all parts. The hash value is the one after ignoring the carry if any.

Example:
Given a hash table of size 100 memory locations, map key 5678, 321 and 34567 to an appropriate location in the hash table

COLLISION NOTES

COLLISION RESOLUTION TECHNIQUES -

1. Open Addressing
    1. Linear Probing
    2. Quadratic Probing
    3. Double Hashing
    4. Rehashing
2. Chaining

OPEN ADDRESSING -

1. LINEAR PROBING -

   Function : **h(k, I) = [h'(k) + i] mod m**

   Where m is the size of the hash table, h'(k) = k mod m and I is the probe number

   Drawback: Primary Clustering

   Example:

   Consider a hash table of size 10. Using Linear probing, insert the keys 72, 27, 36, 24, 63, 81, 92, 101 into the table.

1. QUADRATIC PROBING -

   Function : **h(k, I) = [h'(k) + c1i + c2i] mod m**

   Where m is the size of the hash table, h'(k) = k mod m and I is the probe number which varies from 0 to m-1, c1 and c2 are constants other than 0.

   Drawback: Successive probing explore only fraction of the table

Application: Widely used in Berkeley Fast File System to allocate free blocks

Example:
Consider a hash table of size 10. Using Linear probing, insert the keys 72, 27, 36, 24, 63, 81, 92, 101 into the table. Take c1 = 1 and c2 = 3

1. DOUBLE HASHING -

Function: **h(k, i) = [h1(k) + i*h2(k)] mod m**

Where m is the size of the hash table, h1(k) = k mod m, h2(k) = k mod m' and I is the probe number which varies from 0 to m-1, m' is a value less than m.

Advantage: Minimizes repeated collisions and the effects of clustering

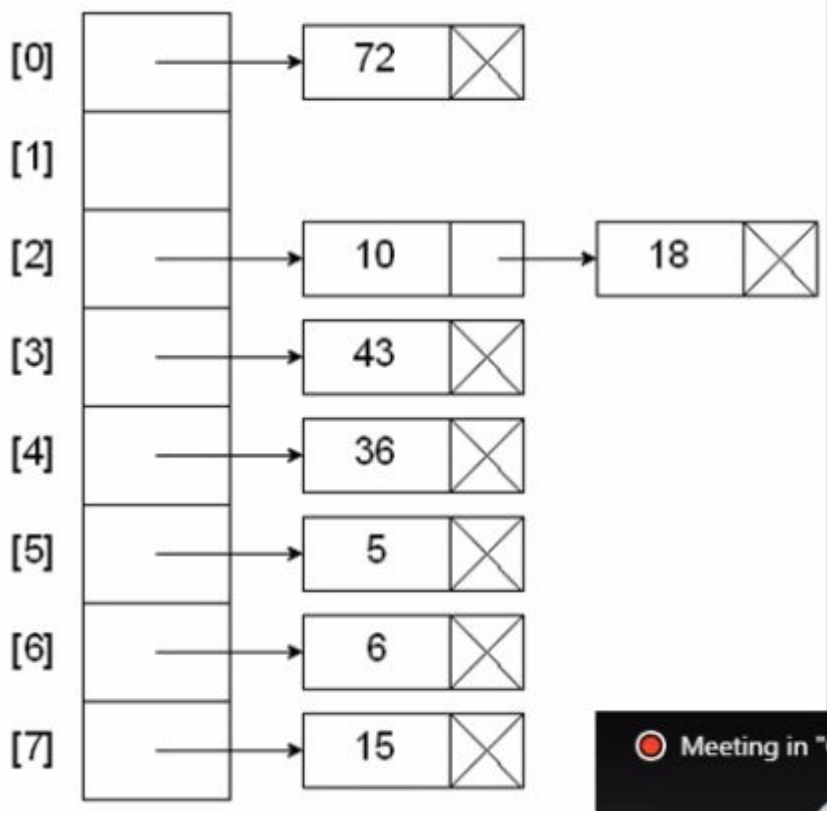
Example:
Consider a hash table of size 10. Using Linear probing, insert the keys 72, 27, 36, 24, 63, 81, 92, 101 into the table. Take h1 = k mod 10 and h2 = k mod 8

1. REHASHING -

Hash table nearly full; collision increases; thereby degrading performance
Create new hash table with double value
Move all values from old hash table to new one
Cons: Too expensive

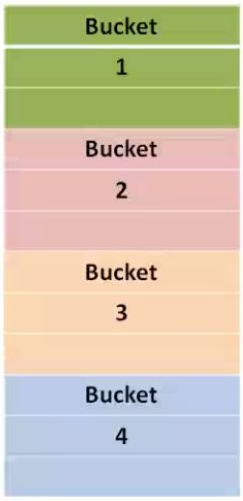
CHAINING -

ADVANTAGES -
Number of key values won't affect the number of locations in the hash table
DISADVANTAGES -
Overhead of storing pointers
Poor cache performance since traversing the linked list.



BUCKET HASHING -

- Divide M slots of hash table into B buckets
- So, each bucket has M / B slots
- Calculate position for key using hash function
- Slot free; allocate
- Else put the key in "Overflow bucket"
- Cons: If key not found in bucket; searching key in Overflow bucket is expensive



APPLICATIONS -

- Database Indexing
- Compiler for Symbol table
- Driver's Licence
- Sparse Matrix