HASHING

HASH TABLES

Hash Table - A list of records/structures with distinct keys.

Operations -

- Insert
- Search
- Delete
- Modify [fields other than the key field]

Goal of Hashing

- Search or insertion in array, list or tree is based on key comparison.

- In hash table, searching/insertion is guided by address of the record having a key value k, where the address is computed by applying the so-called Hash function on the key value.

- Objective is to perform search/insertion operation in constant bounded time, i.e., O(1) time.

A simple example

n elements are to be stored in a hash table of size M, M >> n. An element with key k, will be put in slot i of the hash table, where $i = h(k)$, h being the hash function.

$h(k) = k \bmod M$, such that $0 <= h(k) <= M-1$ for all key k.

Properties of Hashing

- Two keys k_1 and k_2 are said to be **synonyms** with respect to a hash function h if $h(k_1) = h(k_2)$.
- Thus if two synonyms k_1 and k_2 are likely to be inserted into a hash table, they will hash to the same slot in the table and a **collision** is said to occur.
- In the event of a collision, a **Collision Resolution Strategy** has to be called for, to find out a free slot in the table.
- A *good* hashing function would **minimize collisions**.
- A *good* Collision Resolution Strategy would locate a free slot **quickly**.

Hash Functions

- For any key value k, h(k), the hashed value of k is an integer between 0 & M-1, M being the size of the hash table.

Objectives:

- Computation of h should be **efficient**.
- **No. of collisions** should be relatively **small**.

Hash Function Examples

1. **Division:-** Assume k is some integer representing the key; h(k)=k mod M. A good choice of M is a prime number. (Why?)

2. **Mid-square:-** Square the key value, take appropriate no. of middle digits.

   $k = 637,\ M = 100,\ k^2 = 40\underline{57}69,\ h(k) = 57$

Hash Function Examples …

3. **Folding:-** Assume k is a big number. You can represent k in the form of a big number. Partition k into several parts, add the parts modulo M.

   Example: key – “INDIA”, M = 1000.

   Taking the ASCII codes, key = 7378687365.

   Partitioning with 3 digits, the key becomes
   7/378/687/365.

   h(k) = (7+378+687+365) mod 1000 = 437

Hash Function Examples …

4. **Radix Transformation:-** Let k be given in decimal system. Consider that the number has been given in a different radix and convert it into decimal equivalent and finally take modulus M.

   Suppose k = 123456, M = 1000

   $h(k) = (123456)_{11} \bmod 1000 = 195231 \bmod 1000 = 231$

5. **Multiplication:-** $h(k) = \lfloor M*((c*k) \bmod 1)) \rfloor$ where c is a constant fraction.

Collision Resolution Strategies

- Open Addressing
  - Linear Probing
  - Quadrate Probing
  - Double hashing
- Chaining
  - Separate
  - Coalesced
- Bucket hashing

Collision Resolution Strategies

Open Addressing:- The hash table is organized as an array of records and the hash function is used to find out the index in the array corresponding to the given key. Collision occurs when the position where a new key is to be inserted is already filled up.

Collision Resolution Strategies …

Linear Probing:- Probe sequence -- $(h(k)+i) \bmod M$

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

Example: $M = 11$, $h(k) = k \bmod M$

Key sequence- 8, 20, 98, 17, 25, 66, 45, 67

$\alpha = n/M$ is known as the load factor.

Considering equal probability of arrival of keys, expected no. of probes for searching an existing key is
$C = (1+1/(1-\alpha))/2$, $0< \alpha < 1$ for large n & M.

Expected no. of probes for insertion is

$C' = (1+1/(1-\alpha)^2)/2$.

C and C' gives two figures of merit for hashing.

Collision Resolution Strategies …

Problem of linear probing:- Once a cluster is formed, there are more chances that subsequent insertions will land up in the same cluster and thereby increase the length of cluster.

This phenomenon is known as **Primary Clustering**. It degrades the performance of hash.

Collision Resolution Strategies …

Quadratic Probing:-

Probe sequence -- $(h(k) + c_1 i + c_2 i^2) \bmod M$,

c_1 and c_2 are constants.

Small clusters will be formed instead of a very large cluster - **secondary clustering**.

Collision Resolution Strategies …

Double hashing:-

Probe sequence $(h(k) + i*h'(k)) \bmod M$,

where h and h' are two hashing functions.

If h and h' are chosen properly, it is very unlikely that k_1 & k_2 will be synonyms for both h and h'. Therefore no. of collisions can be greatly reduced. Cluster formation can also be avoided.

Example: $k = 123456$, $M = 701$, $h(k)=k \bmod M = 80$

$$h'(k) = 1 + k \bmod (M-1) = 257$$

Probe sequence = $80 + 257*i$, different for different k

Collision Resolution Strategies …

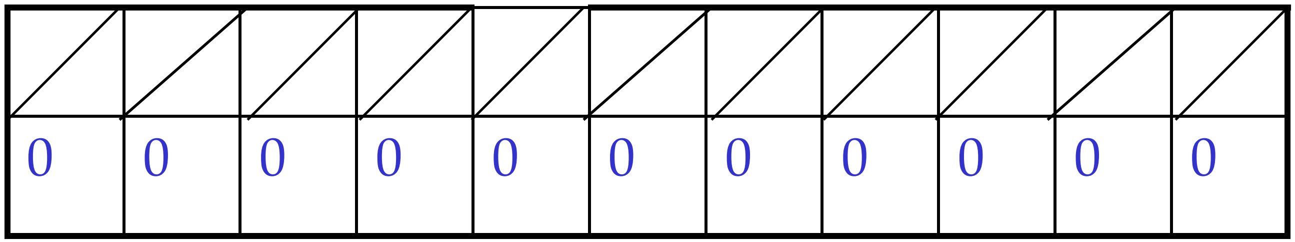
Chaining

a) **Separate Chaining:-** The hash table is a collection of M linear lists. Each key k hashes to one of these M lists. Storage requirement – n records and M+n pointers.

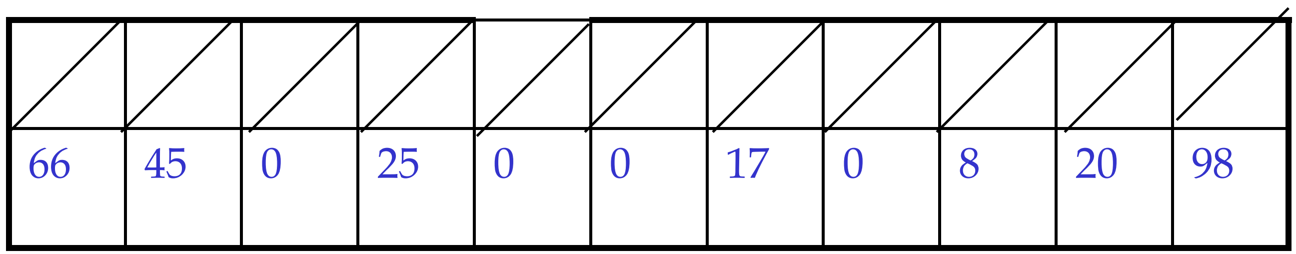
Collision Resolution Strategies …

b) **<u>Coalesced chaining:-</u>** The linked lists are maintained in the same array. Different chains may be coalesced (merged) -- chain headers are not maintained separately.

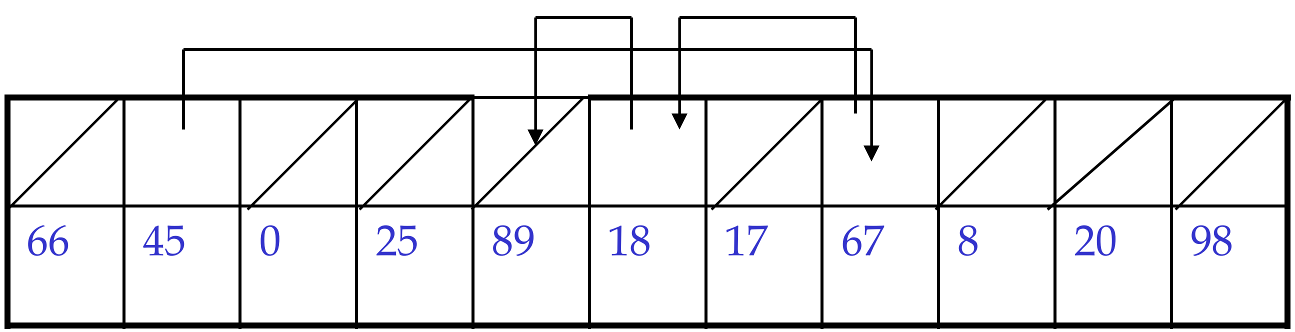
Example: After initialization



After entering 8, 20, 98, 17, 25, 66, 45



After entering 67, 18, 89



Bucket hashing

- Useful for external searching
- Each bucket can hold b records
- A bucket is stored in a disk block
- When a bucket overflows, another bucket is allocated and linked to the former.
- There is a bucket header which has M slots, each slot heading a bucket list.

Bucket hashing …

Example: $M = 7$, $b = 3$, $n = 20$.

insertion sequence – 10, 5, 20, 8, 4, 3, 19, 45, 71, 29, 84, 49, 54, 42, 73, 24, 18, 94, 84, 59.

Expected number of buckets for random inputs

$n>M$, $N = M * \lceil \lceil n/M \rceil /b \rceil$

Expected number of buckets in each chain is

$$\lceil \lceil n/M \rceil /b \rceil$$

Expected no. of probes [disk reads] = $1/2 \lceil \lceil n/M \rceil /b \rceil$