Unit-III
Hashing

08 hours
Pg No - 5-27
Book - V.H. patil DS using C++

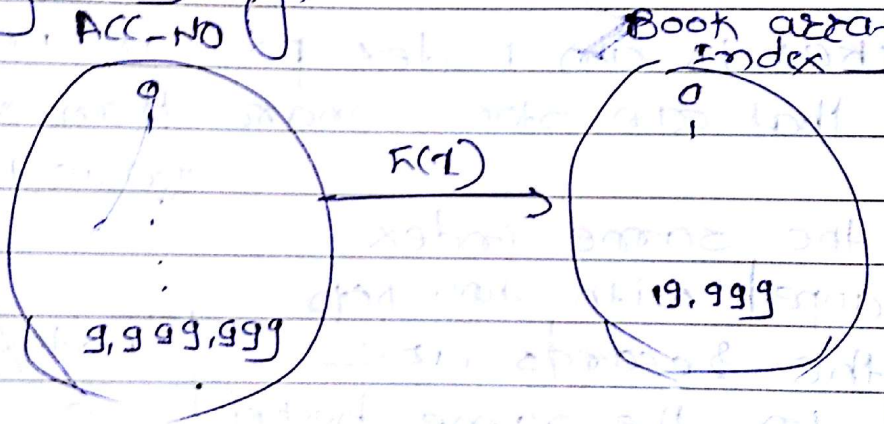
Hash table-

- Hashing is a method of directly computing the address of the record with the help of a key by using suitable mathematical function called the hash function.

- A hash table is an array-based structure used to store <key, Information> pairs.

- Hashing is finding an address where the data is to be stored as well as located using a key with the help of an algorithmic function.

Hashing is f



- The function f(x) will take Acc_no & return the indices where the book record is to be stored in the array & is called the hash function.
- Hash functions transform a key into an addr. Hashing is a technique used for storing & retrieving information associated with it that makes use of individual characters or digit in the key itself.

Hash table - Hash table is an array [0 to max-1] of size max

Hash function -

- Hash function is one that maps a key in the range [0 to max-1] the result of which is used as an index to the hash table for storing & retrieving records.

- hash function is as the function that transform a key into an address. The address generated by a hash function is called the home address

- All home addresses refer to a particular area of the memory called the prime area.

Bucket.

- A bucket is an index position in a hash table that can store more than one record

- when the same index is mapped with two keys. both the records are stored in the same bucket

Table 1.1 Table with bucket size 1

Index	Bucket size 1
0	Alka
1	Bindy
2	
:	
25	Zinat

Table 1.2 Bucket size 2

Index	Bucket size 2	
0	Alka	Abhay
1	Bindu	Babbli
:		
25	Zinat	Zija

Probe- Each action of address calculation & check for success is called as probe

Collision- The result of two keys hashing into the same address is called collision
- when this situation arises, the colliding records must be stored and accessed as determined by collision resolution technique.
- there are two such techniques
  1) open addressing
  2) chaining

Synonym: keys that hash to the same address are called synonyms

open or external hashing.
- when we allow records to be stored in potentially unlimited space, it is called as open or external hashing.

- closed or internal hashing
- when we use fixed space for storage eventu limiting the number of records to be stored it is called as closed or internal hashing

Has perfect hash function:
The hash function that transforms different keys into different addresses is caued a perfect hash function.

Load density
- The maximum storage capacity, that is the maximum number of records that can be accommodated is caued as loading density

Full table -
A full table is one in which all locations are occupied. owing to the characteristics of hash functions, there are always empty locations, Rather a hash function should not allow the table to get filled in more than 75%.

Load factor
- load factor is the number of records stored in a table divided by the maximum capacity of the table, expressed in terms of storage.

Rehashing
- Rehashing is with respect to closed hashing when we try to store the record with key 1 at the bucket position Hash(key 1) & find that it already holds a record.
It is collision situation.
- To handle collision we use a strategy to

choose a sequence of alternative locations Hash1(key1), Hash2(key1) & so on within the bucket table so as to place the record with key1. This is known as Rehashing

Issues - In case of collision

1) we need a good hashing function that minimizes the number of collisions.

2) we want an efficient collision resolution strategy so as to store or locate synonyms

P.g No - 531 - V.H. Pai
- 607 - Sorting

* Hash function -

- To store a record in a hash table, a hash function is applied to the key of the record being stored, returning an index within the range of the hash table
- The record is stored at that index position if it is empty.

Good Hash function -

- The avg performance of hashing depends on how the hash function distributes the set of keys among the slots
- An assumption is that any given record is equally likely to hash into any of the slots, independent of whether any other record has been already hashed to it or not
- This assumption is known as 'simple-uniform hashing'
- A good hash function is one which satisfies this assumption.

There are many methods of implementing hash function

1) Division method -

The resultant index must be within the table index range.

Hash (key) = key % m

- Key is divided by some no m & the remainder is used as the hash address.
- This function gives the bucket address in the range of 0 through (m-1), so the hash table should at least of size m.
- A good choice of m should be prime no greater than 20

2) multiplication method -

The multiplication method works as follows

1) multiply the key 'key' by a constant A in the range $0 < A < 1$ and extract the fractional part of key XA

2) Then multiply this value by m and take the floor of the result

$$\text{Hash (key)} = \lfloor m(\text{key} \times A \bmod 1) \rfloor$$

where key XA mod 1 is the fractional part of key XA.

3) Extraction method -

- when a portion of the key is used for address calculation, the technique is called as the extraction method.
- In digit extraction, a few digits are selected, extracted from the key and are used as the address.

key	Hashed address
345678	357
234137	243

4) mid- square Hashing

- Square of the key and extract the middle digits of the squared key as the address.
- mid - square is used when the key size is less than or equal to 4 digits.

Key	square	Hashed address
2341	5480281	802
1671	2792241	922

5) Folding Technique-

- In this technique, the key is subdivided into subparts that are combined or folded and then combined to form the address.
- There are two types

1) Fold shift -

Key value is divided into several parts of the size of the address. Left, Right, middle parts are added.

2) Fold boundary -

Key value is divided into parts of the size of the address.

Left and Right parts are folded on the fixed boundary between them & the centre part.

EX - 987654321

Left - 987 centre - 654 Right - 321

for fold shift the sum is 987 + 654 + 321 = 1962

Now discard digit 1 & the address is 962.

For Fold boundary

789 + 456 + 123 = ~~1962~~ 1368

discard 1 & the address is 368

c) Rotation -

When the keys are serial, they vary only in the last digit and this leads to creation of synonyms.

Rotating the key would minimize this problem. This method is used along with other method

ex - key 12060 it is rotated we get 512060, then the address is calculated using any other hash function.

d) Universal Hashing -

- To choose the hash function randomly in a way that is independent of the keys that are actually going to be stored.
- This approach is called universal hashing
- The main idea behind universal hashing is to select the hash function at random at run-time from a carefully designed set of functions.
- Bcs of randomization, the algo can behave differently on each execution, even for the same input.
- This approach guarantees good avg case perfor no matter what keys are provided as input.

* Collision Resolution strategies -

- No hash function is perfect. If Hash(key1) = Hash(key2) then key1 & key2 are synonyms & if the bucket size is 1, we say that collision has occured.

- As a consequence we have to store the record key2 at some other location. A search is made for a bucket in which a record is stored containing key2 using one of the several collision-resolution strategies

1) open addressing
   a) Linear probing
   b) Quadratic probing
   c) Double hashing
   d) Key offset
2) Separate chaining
3) Bucket hashing

1) open addressing -
- In open addressing, when collision occurs, it is resolved by finding an available empty location other than the home address.
If Hash(key) is not empty, the positions are probed in the following sequence until an empty location is found.
- when we reach the end of table, the search is wrapped around to start & the search continues till the current collision location

N(Hash(key) + c(1)), N(Hash(key) + c(2)), --- + c(i).

Here N is the normalization function,
Hash(key) is the is the hashing function
c(i) - collision resolution function

- The normalization function is required when the resulting index is out of range. A commonly used normalization function is mod.

- closed hash table uses open addressing. In open addressing, all records are stored in the hash table itself also said to be residing in the prime area which contains all home addresses.

a) linear probing
-A hash table in which a collision is resolved by placing the item in the next empty place following the occupied place is called linear probing.

(Hash(x) + P(i)) MOD max

As P(i) = i for linear probing, the function becomes

(Hash(x) + i) MOD max

initially i = 1 if the location is not empty then it becomes 2, 3, 4 ---

b) Quadratic probing
In quadratic probing, we add the offset as the square of the collision probe number.
(Hash(key) + i2) MOD max where i lies between 1 and (max-1)/2

- quadratic probing works much better than linear probing, but to make full use of the hash table, there are constraints on the values of i & max so that the address lies within the table boundaries

c) Double Hashing

Double hashing uses two hash functions one for accessing the home address of a key & the other for resolving the conflict.

(Hash1(key), (Hash1(key) + i * Hash2(key)) ...

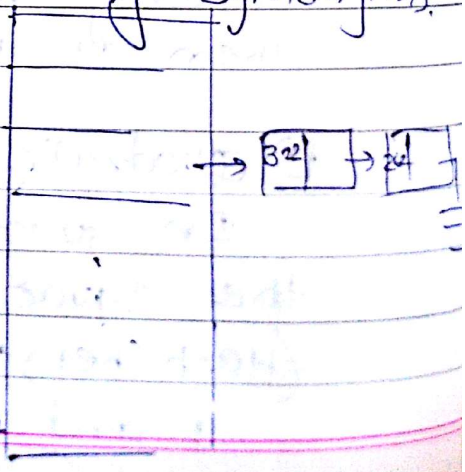
i = 1, 2, 3, 4 ....

The resultant address is divided by modulo max

2) chaining

- we can store the linked lists inside the hash table, in the unused hash table slots
- The technique used to handle synonyms is chaining, it chains together all the records that hash to the same address
- Instead of relocating synonyms, a linked list of synonyms is created whose head is the home address of synonyms.

- a hash table max = 10 both keys 322 & 262 probe to address 2.
- A chain a linked list, stores all items at a particular home address



Hash table overflow

- An overflow is said to occur when a new identifier is mapped or hashed into a full bucket.
- When the bucket size is one, a collision & an overflow occur simultaneously. Therefore any hashing program must incorporate some method for dealing with records that cannot fit into their home addresses.

- There are a number of techniques for handling overflow of records.

1) Open Addressing for overflow Handling
we shall study two ways to handle overflow
a) open addressing
b) chaining

a) open addressing
- we assume that the hash table is an array.
- When a new identifier is hashed into a full bucket, we need to find another bucket for this identifier.
- The simplest solution is to find the closest unfilled bucket through linear probing or linear open addressing

b) chaining -
- In each slot, additional space is required for a link.
- Each chain has a head node. The head node however, usually is much smaller than the other nodes.

* Extendible Hashing

- For fast searching and less disk access, extendible hashing is used.
- It is a type of hash system, which treats a hash as a bit string and uses a trie for bucket lookup.
- Assume that the hash function Hash(key) returns a binary number.
- The first i bits of each string will be used as indices to figure out where they will go in the hash table

h(key1) = 100101
h(key2) = 011110
h(key3) = 110110

Directory

0 → Bucket A for key 2

1 → Bucket B for key 1

Directory

00 → Bucket A for key 2

01

10 → Bucket B for key 1

11 → Bucket C for key 3

Case- study

* Dictionary

- A dictionary is a data structure for efficiently implementing searching, inserting & deleting operations.

- The simplest way to implement a dictionary is through the use of Arrays.

- Arrays are efficient for searching an element whereas insertion & deletion cannot be easily performed.

- Other sophisticated ways to implement a dictionary is using hashing & balanced search trees.

A typical dictionary include the following operations

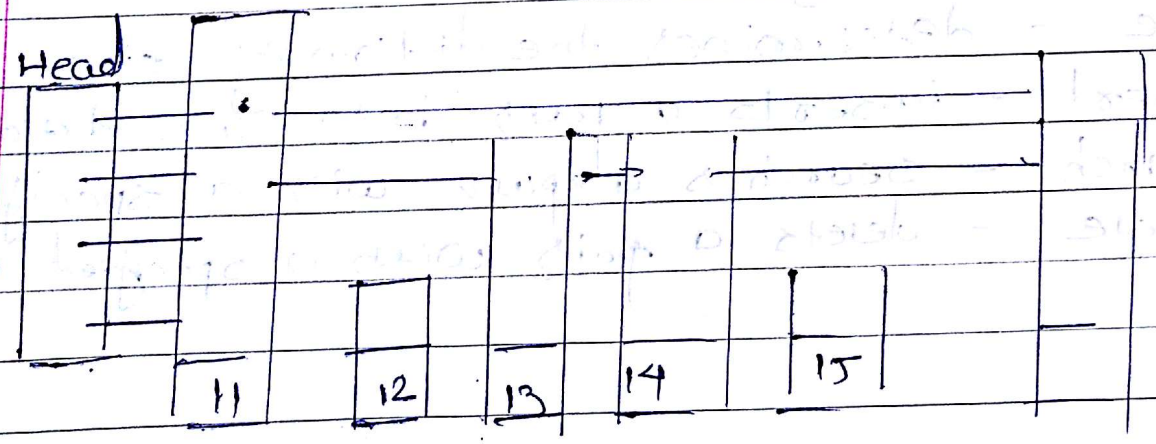
1) Empty - checks whether the dictionary is empty or not
2) Size - determines the dictionary size
3) Insert - inserts a pair into the dictionary
4) Search - searches a pair with a specified key.
5) Delete - deletes a pair with a specified key.

* Skip list.

- A bauonce tree is one of the most popular data structure used for seatching
- one of the vatiants of bauonced tree is the skip list.

- A skip list stores the sorted data in the form of linked list. These items are stored as a hieoratchy of linked lists where each list links increasingly sparse subsequences of the items.

- A skip list allows the process of item look up in efficient mannet. The skip list data structure skips over many of the items of the full list in one step, that's why it is known as skip list



skip list

Date

comparision of Hashing & skip list -

1) The hash table is a simple array of items. A hash table is an alternative method for representing a dictionary

- skip list is a linked list augmented with layers of pointers for quicky jumping over a large number of elements & then descending to the next layer

2) skip lists are one way of implementing a dictionary abstract data type, which stores a set of items & allows us to add, remove & search for items
Though hash tables are more popular

3) search & delete operation on skip list is $O(\log n)$, the worst-case performance $\Theta(n)$
The hash table is used in many applications takes $\Theta(1)$