Unit-6

Hashing

Page No.
Date

Imp
Que- Explain Hashing with its techniques.

⇒

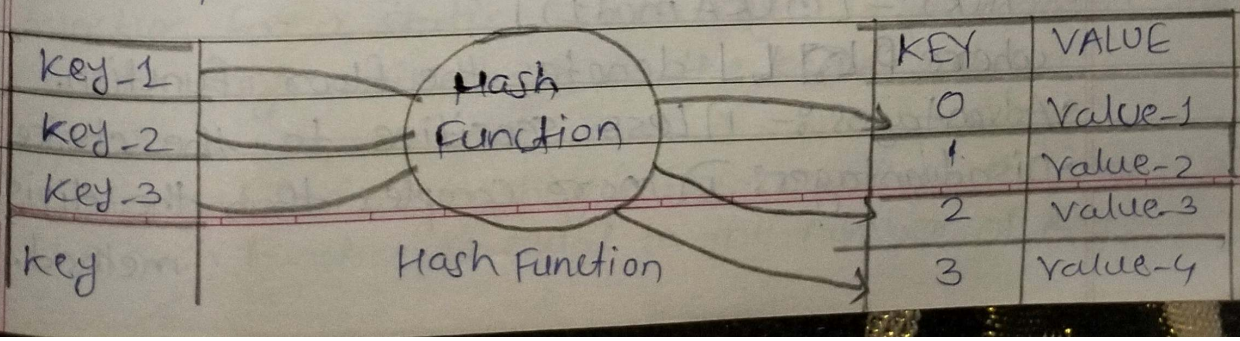
Hashing :-

- Hashing is Data ~~Root~~ structure refers to the Process of transforming a given key to another value.
- It involves mapping data to a specific index in a hash table using a hash function that enables fast retrieval of information based on its key.
- The transformation of a key to the corresponding value is done using a Hash Function and the value obtained from the hash function is called Hash code.

• Components of Hashing :-

There are three components of hashing :-

1) Key :- A key can be anything string or integer which is fed as input in the hash function the technique that determines an index or location for storage of an item in a data structure.
2) Hash Function :- The hash function receives the input key and returns the index of an element in an array called hash table. The index is known as the hash index.
3) Hash Table :- Hash table is a data structure that maps keys to values using a special function called hash function. Hash stores the data in an associative manner in an array where each data value has its own unique index.



What is a Hash function?

- The hash function creates a mapping between key and value, this is done through the use of mathematical Formulas known as hash functions.
- The result of the hash function is referred to as a hash value or hash.
- The hash value is a representation of the original string of characters but usually smaller than the original.

• Types of Hash functions:-

1) Division method:-

- The division method involves dividing the key by a prime number and using the remainder as the hash value.

$$h(k) = k \bmod m$$

where, k is the key.

Advantages:- 1) simple to implement
2) Works well when m is a prime number.

Disadvantages:- 1) Poor distribution if m is not chosen wisely.

2) Multiplication method:-

- In the multiplication method, a constant A ($0 < A < 1$) is used to multiply the key. The fractional part of the product is then multiplied by m to get the hash value.

$$h(k) = \lfloor m(kA \bmod 1) \rfloor$$

where $\lfloor \rfloor$ denotes the floor function.

• Advantages:- 1) Less sensitive to the choice of m.

• Disadvantages:- 1) More complex than the division method.

3] Mid-square method :-
-In the mid-square method, the key is squared, and the middle digits of the result are taken as the hash value.
steps :-
1] Square the key
2] Extract the middle digits of the squared value.
• Advantages :-
1] Produces a good distribution of hash values.
• Disadvantages :-
1] May require more computational effort.

4] Folding method :-
-The folding method involves dividing the key into equal parts, summing the parts and then taking the modulo with respect to m.
steps :-
1] Divide the key into parts
2] sum the parts
3] Take the modulo m of the sum.
• Advantages :-
1] Simple and easy to implement

• Disadvantages :-
1] Depends on the choice of partitioning scheme.

5] Perfect Hashing :-
-Perfect hashing aims to create a collision-free hash function for a static set of keys. It guarantees that no two keys will hash to the same value.
Types :-
• Minimal Perfect Hashing :- Ensures that the range of the hash function is equal to the number of keys.

• NON-minimal Perfect Hashing :- The range may be larger than the number of keys.

Advantages :- 1) No collisions
Disadvantages :- complex to construct

6) Universal Hashing :-
- universal hashing uses a family of hash functions to minimize the chance of collision for any given set of inputs.
$h(k) = ((a \cdot k + b) \bmod P) \bmod m$
where a and b are randomly chosen constants
P is a prime number greater than m and k is key
• Advantages :-
1) Reduce the Probability of collisions
Disadvantages :-
1) Requires more computation and storage.

7) Cryptographic Hash Functions :-
- Cryptographic hash functions are designed to be secure and are used in cryptography.
- Examples include MD5, SHA-1 and SHA-256.
• characteristics :-
- Pre-image resistance
- second pre-image resistance
- collision resistance

• Advantages :- 1) High security

• Disadvantages :- 1) computationally intensive.

Imp

Que- Explain open Addressing collision Handling technique.

⇒

Open Addressing :-

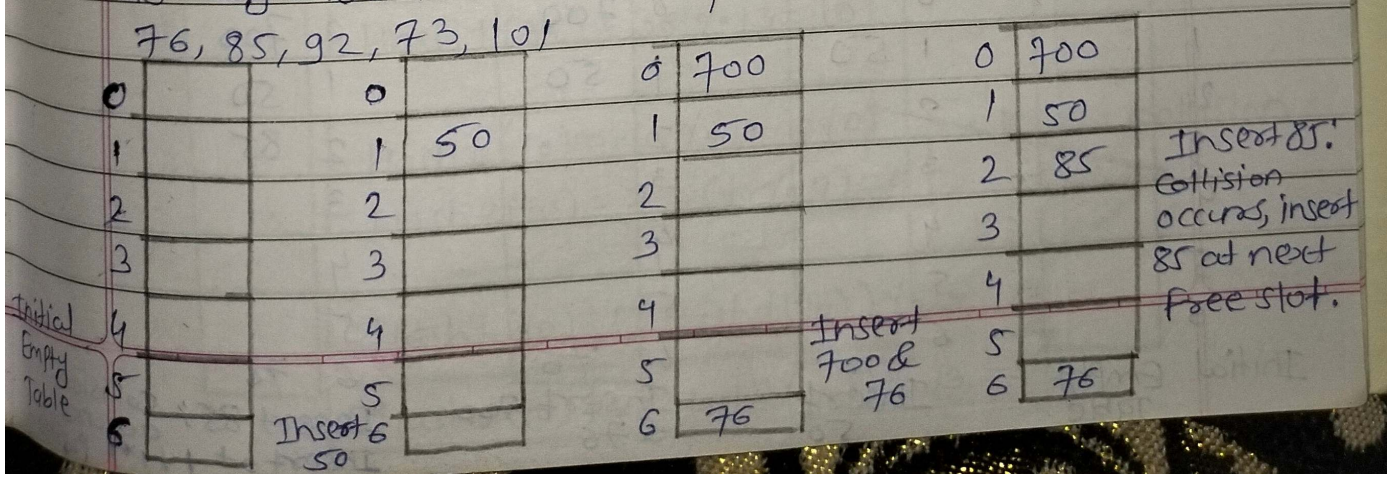
- In open Addressing, all elements are stored in the hash table itself.
- So at any point, the size of the table must be greater than or equal to the total number of keys.
- This approach is also known as closed hashing.

- **Insert(k) :-** keep Probing until an empty slot is found. Once an empty slot is found, insert k.
- **search(k) :-** keep Probing until the slot's key doesn't become equal to k or an empty slot is reached.

Different ways of open Addressing :-

1) **Linear Probing :-**

- In linear probing, the hash table is searched sequentially that starts from the original location of the hash.
- If in case the location that we get is already occupied, then we check for the next location.
- The function used for rehashing is as follows :-
  rehash(key) = (n+1) % table-size.
- For example :- Let us consider a simple hash function as "key mod 7" and a sequence of keys as 50, 700, 76, 85, 92, 73, 101



0	700
1	50
2	85
3	92
4	
5	
6	76

Insert 92, collision occurs as 50 is there at index 1. Insert at next free slot.

0	700
1	50
2	85
3	92
4	73
5	101
6	76

Insert 73 and 101

Application of Linear probing :-
1) Symbol tables
2) caching
3) Databases
3) compiler design
4) spell checking.

2) Quadratic Probing :-
- Quadratic Probing is a method with the help of which we can solve the problem of clustering.
- This method is also known as the mid-square method.
- In this method, we look for the i^{2}th slot in the i^{th} iteration.
- We always start from the original hash location.
- If only the location is occupied then we check the other slots.
- For example :- Let us consider simple hash function as "key mod 7" and sequence of keys 50, 700, 76, 85, 92, 73, 101

	Initial Empty Table	Insert 50	Insert 700 & 76	Insert 85; collision occurs, Insert at $1+1^{*}1$ position
0			700	700
1		50	50	50
2				85
3				
4				
5				
6			76	76

Time complexity :- O(n)

0	700
1	50
2	85
3	
4	
5	92
6	76

Insert 92: collision occurs at
1 :- insert 1 + 2*2 Position

0	700
1	50
2	85
3	73
4	101
5	92
6	76

Insert 73 and 101

3) Double Hashing :-
- The intervals that lie between probes are computed by another hash function.
- Double hashing is a technique that reduces clustering in an optimized way.
- In this technique, the increments for the probing sequence are computed by using another hash function.
- We use another hash function hash 2(x) and look for the i*hash 2(x) slot in the ith rotation.
- Example :- Hash1(key) = key % 13
Hash2(key) = 7 - (key % 7).

Hash1(19) = 19 % 13 = 6

Hash1(27) = 27 % 13 = 1

Hash1(36) = 36 % 13 = 10

Hash1(10) = 10 % 13 = 10
Hash2(10) = 7 - (10 % 7) = 4

collision

Hash1(10) + 1 * Hash2(10)) % 13 = 1

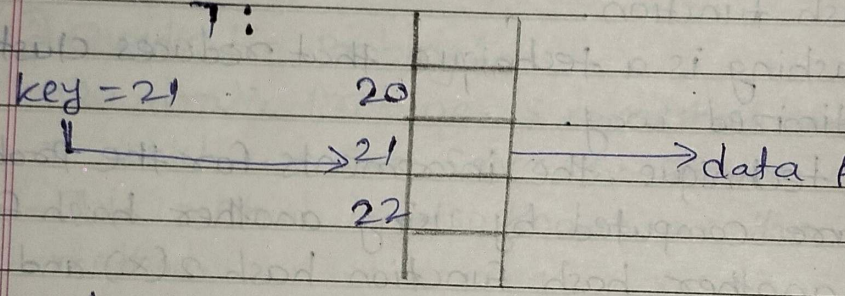
Hash1(10) + 2 * Hash2(10)) % 13 = 5

Page No.
Date

que- Explain Index mapping with example.

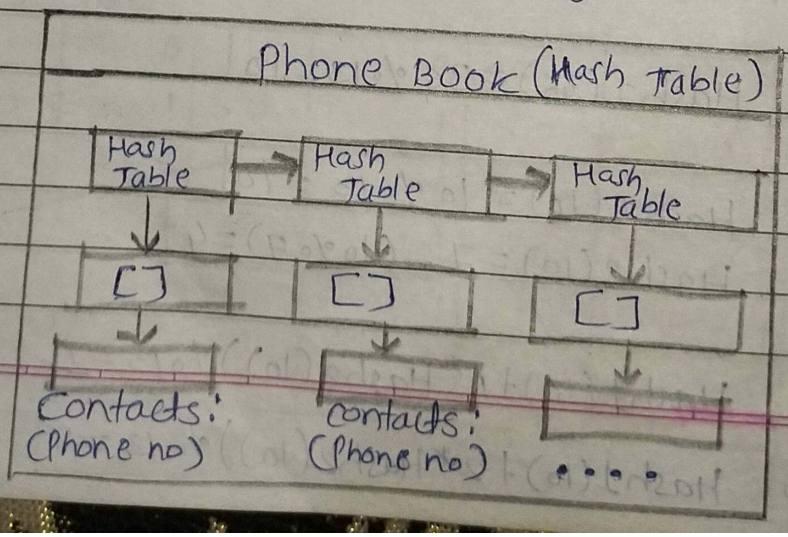
⇒ Index Mapping :-

- Index Mapping also known as Trivial Hashing.
- Index mapping is a simple form of hashing where the data is directly mapped to an index in a hash table.
- The hash function used in this method is typically the identity function, which maps the input data to itself.
- In this case, the key of the data is used as the index in the hash table and the value is stored at that index.



- Index :- An index is a unique identifier or key associated with each value in the data structure.
- mapping :- The process of associating an index / key with a specific value.

Example :- Phone Book using Index mapping.



Page No.
Date

Explanation :-

- Phone Book :- Represents the overall structure where the hash table stores the contacts Phone number.
- Hash Tables :- Each hash table reporesents a bucket where contacts are stored. Each bucket can store multiple contacts
- Hashing Function :- maps contact names to indices in the hash table.
- contacts (Phone Numbers) :- Actual data stored in the hash table, associated with contact names.

- Advantages :-
1) Fast Retrieval
2) Efficient storage
3) Flexibility
4) Collision Handling
5) Scalability
6) Reduced memory Requirements

- Disadvantages :-
1) Collision Resolution overhead
2) Space overhead
3) Hash Function sensitivity
4) Not suitable for ordered operations

- Applications :-
1) Databases
2) caching systems
3) Compiler symbol Tables
4) Web servers

Page No.
Date

Que- Explain collision resolution techniques:-

⇒

Collision resolution techniques:-
There are two types of collision resolution techniques
1) separate chaining (open hashing)
2) open addressing (closed hashing)

1) Separate chaining:-
- The idea behind separate chaining is to implement the array as a linked list called a chain.
- Separate chaining is one of the most popular and commonly used techniques in order to handle collisions.
- This method involves making a linked list out of the slot where the collision happened, then adding the new key to the list.
- Separate chaining is the term used to describe how this connected list of slots resembles a chain.
- It is more frequently utilized when we are unsure of the number of keys to add or remove.

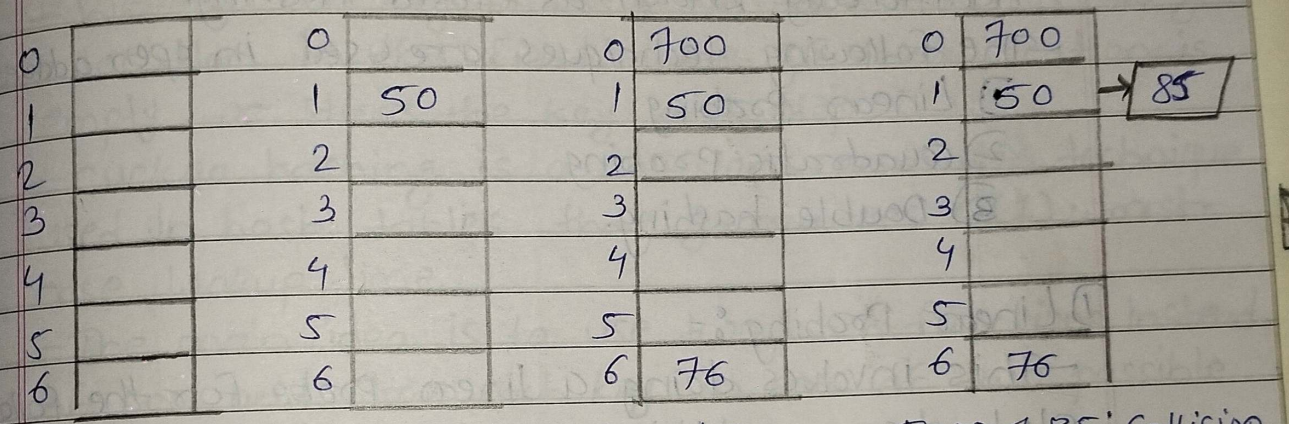
• Time complexity:-
- Its worst-case complexity for searching is O(n)
- Its worst-case complexity for deletion is O(n).

• Advantages:-
1) It is easy to implement
2) The hash table never fills full, so we can add more elements to the chain
3) It is less sensitive to the functions of the hashing

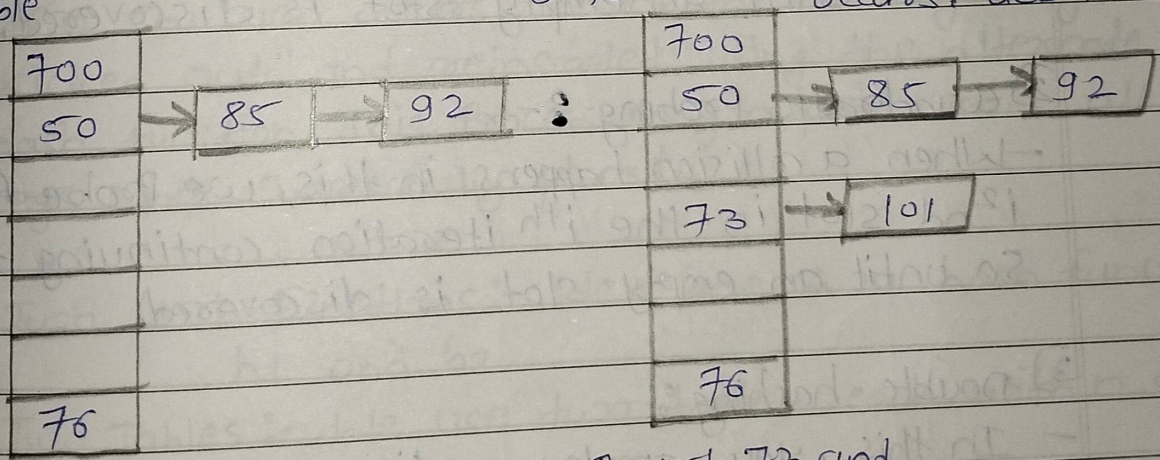
Page No.
Date

Disadvantages:-
1) the cache performance of chaining is not good.
2) memory wastage is too much in this method
3) It requires more space for element links.

Example:- Let us consider a simple hash function as "key mod 7" and sequence of keys as 50, 700, 76, 85, 92, 73, 101.



Initial Empty Table | Insert 50 | Insert 700 & 76 | Insert 85: Collision occurs: add to chain



Insert 92: Collision occurs, add to chain

Insert 73 and 101

2) open addressing :-
- To Prevent collisions in the hashing table, open addressing is employed as a collision-resolution technique.
- No key is kept anywhere else besides the hash table.
- As a result, the hash table's size is never equal to or less than the number of keys.
- Additionally known as closed hashing.
- The following techniques are used in open addressing:
  1) Linear Probing
  2) Quadratic Probing
  3) Double hashing.

1) Linear Probing :-
- This involves doing a linear probe for the following slot when a collision occurs and continuing to do so until an empty slot is discovered.

2) Quadratic Probing :-
- When a collision happens in this, we probe for the i^2nd slot in the ith iteration, continuing to do so until an empty slot is discovered.

3) Double hashing :-
- In this, you employ a different hashing algorithm, and in the ith iteration, you look for (i * has 2(x)).
- The determination of two hash functions requires more time.
- the performance of the cache is relatively poor when using double probing.

Page No.
Date

Imp que- Explain ~~cukoo~~ cuckoo Hashing.

cuckoo Hashing :-
- It was described by Rasmus pagh and Flemming Friche Rodler in the year 2001.
- cuckoo hashing is a type of closed hashing.
- It uses two hash functions and two tables to avoid collisions.
- We pass our key to the first hash function to get a location in the first table. If that location is empty, we store the key and stop.
- cuckoo hashing is a collision resolution technique used in hash tables that guarantees O(1) worst-case lookup time.
- The main idea is to use two hash functions instead of one and store each key in one of two possible locations.
- If a collision occurs, one of the existing keys is "kicked out" and reinserted using the alternate hash function. This process may repeat until all keys are successfully placed.

- Two Hash Functions :- We define two hash function h1 and h2
- Two Tables :- We use two tables, each with n slots.

Example :- insert the following keys into a cuckoo hash table :- 21, 10, 31, 14 and 18.

Steps :-
1) Initialization :-
- Two Hash Functions :-

- $h_1(key) = key \bmod 5$
- $h_2(key) = (key/5) \bmod 5$

• Two Tables :- Each with 5 slots.

② Insert 21 :-

i) Compute Positions :-

- $h_1(21) = 21 \bmod 5 = 1$
- $h_2(21) = (21/5) \bmod 5 = 4$

ii) Insert 21 at Position 1 in Table 1.

	Table 1	Table 2
0		
1	21	
2		
3		
4		

③ Insert 10 :-

i) Compute Positions :-

- $h_1(10) = 10 \bmod 5 = 0$
- $h_2(10) = (10/5) \bmod 5 = 2$

ii) Insert 10 at Position 0 in table 1.

	Table 1	Table 2
0	10	
1	21	
2		
3		
4		

4] Insert 31 :-

i] Compute Positions :-

- $h_1(31) = 31 \bmod 5 = 1$
- $h_2(31) = (31/5) \bmod 5 = 1$

ii] Position 1 in Table 1 is occupied by 21. kick out 21 and insert 31 at Position 1 in Table 1.

iii] Reinsert 21 using h_2

- $h_2(21) = 4$

iv] Insert 21 at position 4 in Table 2.

	Table 1	Table 2
0	10	
1	31	
2		
3		
4		21

5] Insert 14 :-

i] Compute Positions :-

- $h_1(14) = 14 \bmod 5 = 4$
- $h_2(14) = (14/5) \bmod 5 = 2$

ii] Position 4 in Table 1 is free, so insert 14 there.

Table 1	Table 2
10	
31	
14	21

Page No.
Date

6) Insert 18 :-

i) Compute Positions :-

- $h_1(18) = 18 \bmod 5 = 3$
- $h_2(18) = (18/5) \bmod 5 = 3$

ii) Position 3 in Table 1 is free, so insert 18 there.

	Table 1	Table 2
0	10	
1	31	
2		
3	18	
4	14	21

- This guarantees O(1) worst-case lookup time.

• Advantages :-

1) Constant Time Lookups
2) Simple and Efficient
3) Cache Efficiency
4) No need for chaining
5) Reduced collision complexity.

• Disadvantages :-

1) Need for Rehashing
2) Space overhead
3) Hash Function Dependency
4) Not suitables for very Large Tables.

• Applications :-

1) Network Routing and switching
2) Database Indexing
3) caching systems.

4) Distributed systems
5) Bioinformatics.

imp que- ⇒ comparison between folding and division method.

Folding method	Division method
1) splits key, combines parts arithmetically	1) Divides key by a prime uses remainder.
2) key split into parts, parts combined	2) key divided by prime, remainder used
3) moderately complex	3) simple and efficient
4) Good for long or structured keys	4) suitable for most key types.
5) Good with well-mixed parts	5) Good with a proper prime number.
6) less predictable	6) more predictable with good prime choice.
7) Handles long keys well, flexible.	7) simple, efficient.
8) Example :- key - 123456 → 12+34+56 = 102	8) Example :- key - 123456 - 123456 % 101 = 45