Practical No:1

Consider telephone book database of N clients. Make use of a hash table implementation to quickly look up client's telephone number. Make use of two collision handling techniques and compare them using number of comparisons required to find a set of telephone numbers. Write a Python program for the same.

Pre-requisite:

- Knowledge of Python programming
- Basic knowledge of hashing and collision handling techniques

Objective:

- To analyze advanced data structure like hash table.
- To understand collision handling techniques.
- To understand practical implementation of hash table.

Input:

Client record: Name and Telephone number

Outcome:

- Hash table with all clients information
- Print number of comparisons required to find a telephone number

Description:

What is Hashing?

- Hashing is finding an address where the data is to be stored as well as located using a key with the help of the algorithmic function
- Hashing is a method of directly computing the address of the record with the help of a key by using a suitable mathematical function called the hash function
- A hash table is an array-based structure used to store

  <key, information> pairs
- The resulting address is used as the basis for storing and retrieving records and this address is called as home address of the record
- For array to store a record in a hash table, hash function is applied to the key of the record being stored, returning an index within the range of the hash table
- The item is then stored in the table of that index position

Functions need to be implemented

1. **create_record**
2. **display_record**
3. **delete_record**
4. **search_record**
5. **update_record**

Need to take Input data from user - Name**, Telephone number**

What is Hash Table?

- Hash table or hash map is a data structure used to store key-value pairs.
- It is a collection of items stored to make it easy to find them later.
- It uses a hash function to compute an index into an array of buckets or slots from which the desired value can be found.
- It is an array of list where each list is known as bucket.
- It contains value based on the key.
- Hash table is used to implement the map interface and extends Dictionary class.
- Hash table is synchronized and contains only unique elements.

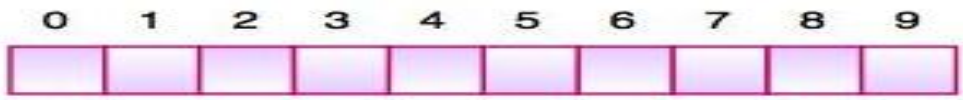

Fig. Hash Table

- The above figure shows the **hash table with the size of n = 10**. Each position of the hash table is called as **Slot**. In the above hash table, there are n slots in the table, names = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Slot 0, slot 1, slot 2 and so on. Hash table contains no items, so every slot is empty.
- As we know the mapping between an item and the slot where item belongs in the hash table is called the hash function. **The hash function takes any item in the collection and returns an integer in the range of slot names between 0 to n-1.**

Collision Resolution Techniques

a. Linear Probing

- Take the above example, if we insert next item 40 in our collection, it would have a hash value of 0 (40 % 10 = 0). But 70 also had a hash value of 0, it becomes a problem. This problem is called as **Collision** or **Clash**. Collision creates a problem for hashing technique.
- **Linear probing is used for resolving the collisions in hash table**, data structures for maintaining a collection of key-value pairs.

- The simplest method is called Linear Probing. Formula to compute linear probing is: **P = (1 + P) % (MOD) Table_size**

b. Double Hashing

Double hashing uses the idea of applying a second hash function to the key when a collision occurs.

The result of the second hash function will be the number of positions form the point of collision to insert.

There are a couple of requirements for the second function:

1. it must never evaluate to 0
2. must make sure that all cells can be probed

A popular second hash function is:

hash2(key) = PRIME – (key % PRIME)

where PRIME is a prime smaller than the TABLE_SIZE.

So**, H(Key) =** ***(hash1(key) + i * hash2(key)) % TABLE_SIZE***

And **hash1(key) = key % TABLE_SIZE**

Algorithms : Write your own algorithms for linear probing and double hashing

Program: Write your own program and attach printouts

Output:

Conclusion:

By this way, we can perform the operations on hash table and use collision handling techniques if needed.

Question Bank:

1. Explain hashing with example
2. What is hash function? Explain with example.
3. Write short note on open hashing.
4. Write short note on closed hashing/open addressing
5. Write the algorithm for linear probing and double hashing