Collision Handling

Collision Handling: Since a hash function gets us a small number for a big key, there is possibility that two keys result in same value. The situation where a newly inserted key maps to an already occupied slot in hash table is called collision and must be handled using some collision handling technique. Following are the ways to handle collisions:

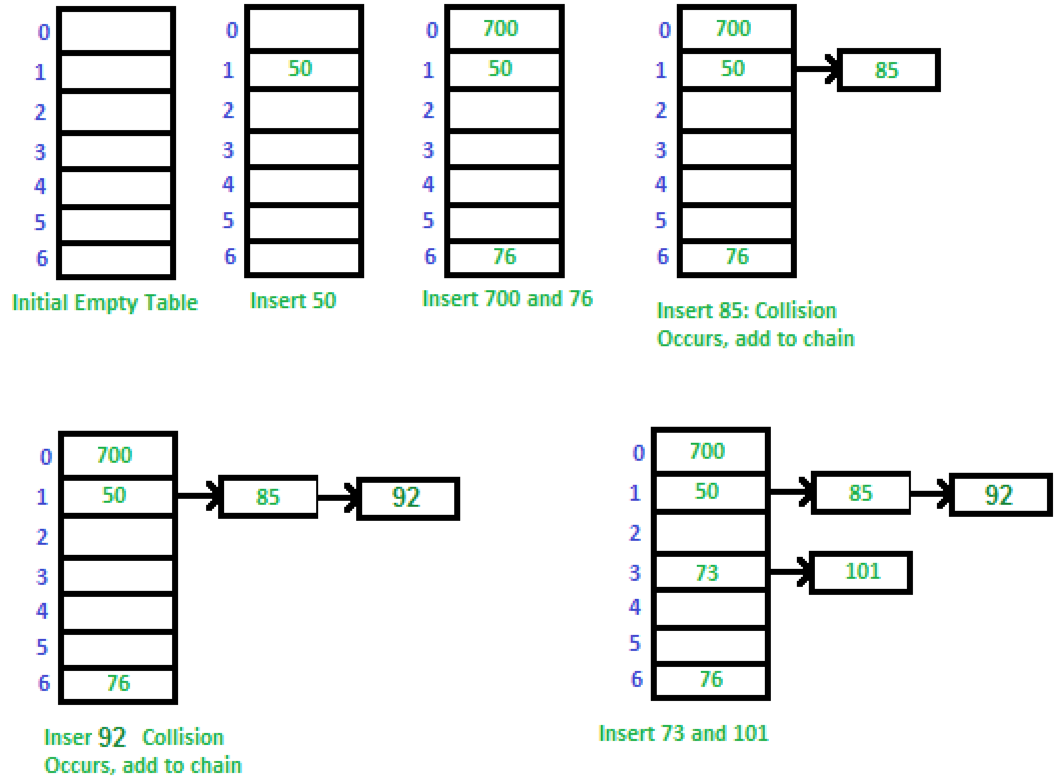
- **Chaining:**The idea is to make each cell of hash table point to a linked list of records that have same hash function value. Chaining is simple, but requires additional memory outside the table.
- **Open Addressing:** In open addressing, all elements are stored in the hash table itself. Each table entry contains either a record or NIL. When searching for an element, we examine the table slots one by one until the desired element is found or it is clear that the element is not in the table.

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.

> *The **linked list** data structure is used to implement this technique. So what happens is, when multiple elements are hashed into the same slot index, then these elements are inserted into a singly-linked list which is known as a chain*

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


Initial Empty Table
Insert 50
Insert 700 and 76
Insert 85: Collision Occurs, add to chain
Inser 92 Collision Occurs, add to chain
Insert 73 and 101

Open Addressing:

Like separate chaining, open addressing is a method for handling collisions. 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 (Note that we can increase table size by copying old data if needed). This approach is also known as closed hashing. This entire procedure is based upon probing. We will understand the types of probing ahead:

1. Linear Probing:

> *The function used for rehashing is as follows: rehash(key) = (n+1)%table-size.*

2. Quadratic Probing

If you observe carefully, then you will understand that the interval between probes will increase proportionally to the hash value. Quadratic probing is a method with the help of which we can solve the problem of clustering that was discussed above. 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.

> *let hash(x) be the slot index computed using hash function.*
>
> *If slot hash(x) % S is full, then we try (hash(x) + 1*1) % S*
> *If (hash(x) + 1*1) % S is also full, then we try (hash(x) + 2*2) % S*
> *If (hash(x) + 2*2) % S is also full, then we try (hash(x) + 3*3) % S*

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 hash2(x) and look for the i*hash2(x) slot in the **i^{th}** rotation.

Marked as Read

Dash
All
Articles
Videos
Problems
Quiz
Contest