Overflow handling - Chaining

Overflow handling - Chaining

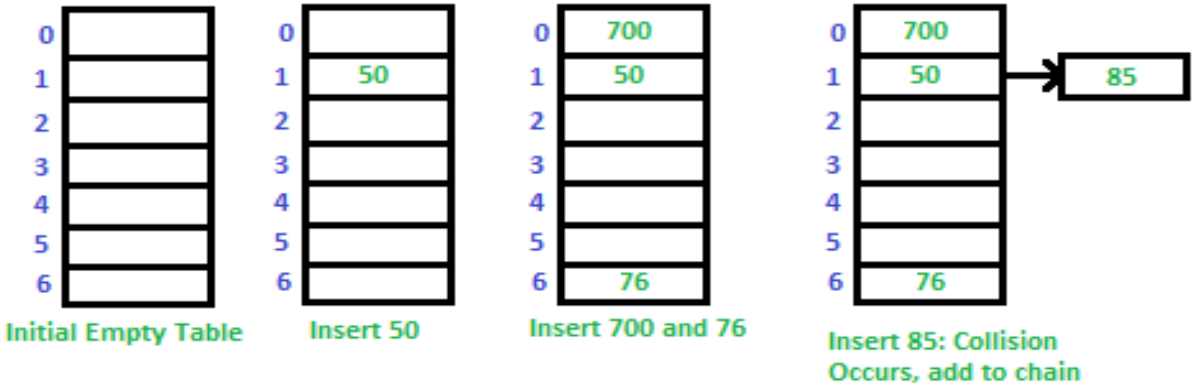
- Since a hash function gets us a small number for a key which is a big integer or string,
- There is possibility that two keys result in same value.
- The situation where a newly inserted key maps to an already occupied slot in hash table is called collision
- It must be handled using some collision handling technique.

- There are mainly two methods to handle collision:
  1) Separate Chaining
  2) Open Addressing
- **Separate 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 Simpler to implement.
- In chaining, Hash table never fills up, we can always add more elements to chain.
- Chaining is Less sensitive to the hash function or load factors.
- Chaining is mostly used when it is unknown how many and how frequently keys may be inserted or deleted.

- Cache performance of chaining is not good as keys are stored using linked list.
- Wastage of Space (Some Parts of hash table in chaining are never used).
- Chaining uses extra space for links.

Chaining


0
1
2
3
4
5
6
Initial Empty Table
50
Insert 50
700
76
Insert 700 and 76
85
Insert 85: Collision
Occurs, add to chain
92
Inser 92 Collision
Occurs, add to chain
73
101
Insert 73 and 101

- **Advantages:**
  1) Simple to implement.
  2) Hash table never fills up, we can always add more elements to chain.
  3) Less sensitive to the hash function or load factors.
  4) It is mostly used when it is unknown how many and how frequently keys may be inserted or deleted.

- **Disadvantages:**
  1) Cache performance of chaining is not good as keys are stored using linked list. Open addressing provides better cache performance as everything is stored in same table.
  2) Wastage of Space (Some Parts of hash table are never used)
  3) If the chain becomes long, then search time can become O(n) in worst case.
  4) Uses extra space for links.

- **Performance of Chaining:**
  m = Number of slots in hash table
- n = Number of keys to be inserted in hash table Load factor $\alpha = n/m$
- Expected time to search = $O(1 + \alpha)$
- Expected time to insert/delete = $O(1 + \alpha)$
- Time complexity of search insert and delete is $O(1)$ if α is $O(1)$