Programme : F. Y. B. Sc.
Subject : Computer Science
Semester : II
Course Code : CSC 102
Course Title : Data Structures
Unit 7 : Hashing
Module Name : Coalesced Hashing, Separate Chaining, Dynamic and Extendible Hashing Choosing a Hash Function, Perfect Hashing Function

Module Number : 39

Collision in Hashing

- Hash function is used to compute the hash value for a key.
- Hash value is then used as an index to store the key in the hash table.
- Hash function may return the same hash value for two or more keys.
    - ❖ When the hash value of a key maps to an already occupied bucket(address) of the hash table, it is called as a Collision.

Coalesced hashing

- Coalesced hashing is a collision avoidance technique when there is a fixed sized data.
- It is a combination of both Separate chaining and Open addressing.
- It uses the concept of Open Addressing(linear probing) to find first empty place for colliding element from the bottom of the hash table and the concept of Separate Chaining to link the colliding elements to each other through pointers.
- The hash function used is :
  h(key)=(key)%(total number of keys).

Example:
n = 10 Input : {20, 35, 16, 40, 45, 25, 32, 37, 22, 55}
Hash function
h(key) = key%10
h(20)=0
h(35)=5
h(16)=6

Hash value	Data	Next
0	20	NULL
1	–	NULL
2	–	NULL
3	–	NULL
4	–	NULL
5	35	NULL
6	16	NULL
7	–	NULL
8	–	NULL
9	–	NULL

Insert 40
h(40)=0 which is already occupied so we search for the first empty block from the bottom and insert it there i.e 9 index value.
Also the address of this newly inserted node(index value of a node) i.e(9)is initialised in the next field of 0th index value node.

Hash value	Data	Next
0	20	9
1	–	NULL
2	–	NULL
3	–	NULL
4	–	NULL
5	35	NULL
6	16	NULL
7	–	NULL
8	–	NULL
9	40	NULL

h(45)=5
h(25)=5
h(32)=2

h(37)=7
h(22)=2
h(55)=5

Hash value	Data	Next
0	20	9
1	55	NULL
2	32	3
3	22	NULL
4	37	1
5	35	8
6	16	NULL
7	25	4
8	45	7
9	40	NULL

Separate Chaining

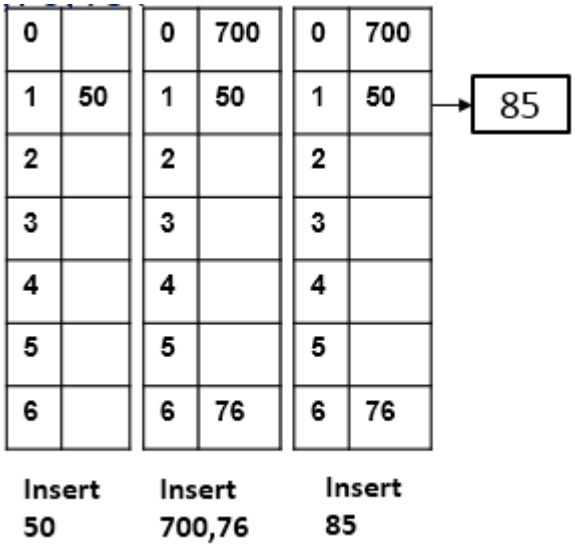
To handle the collision,

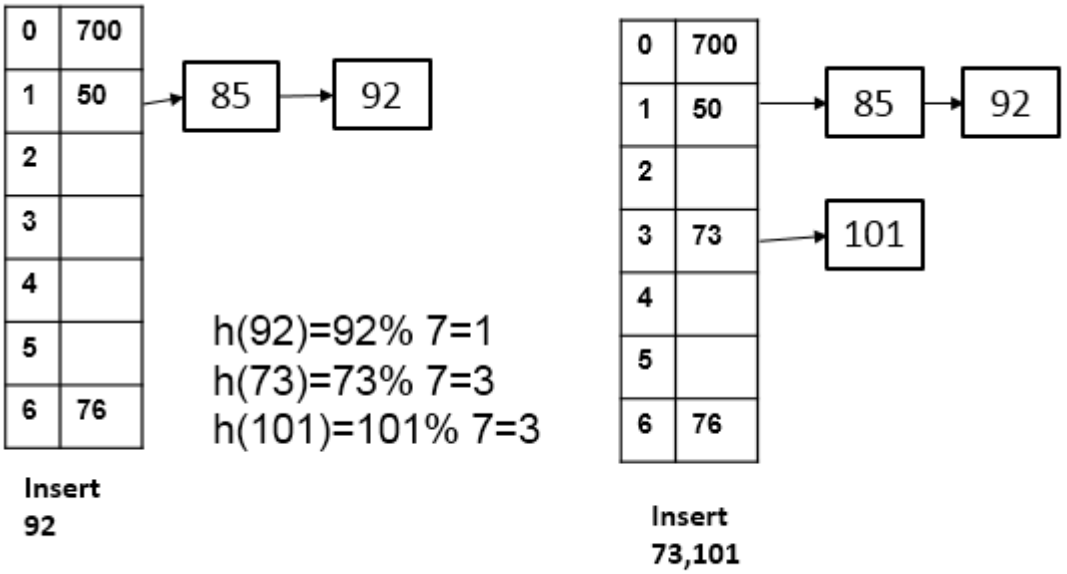
- This technique creates a linked list to the slot for which collision occurs.
- The new key is then inserted in the linked list.
- These linked lists to the slots appear like chains.

That is why, this technique is called as separate chaining.

Example:

let us take a table of size 7 and insert some keys in it taking a hash function h(key)=key% 7
Keys :50,700,76,85,92,73,101
h(50)=50% 7=1
h(700)=700% 7=0
h(76)=76% 7=6
h(85)=85% 7=1



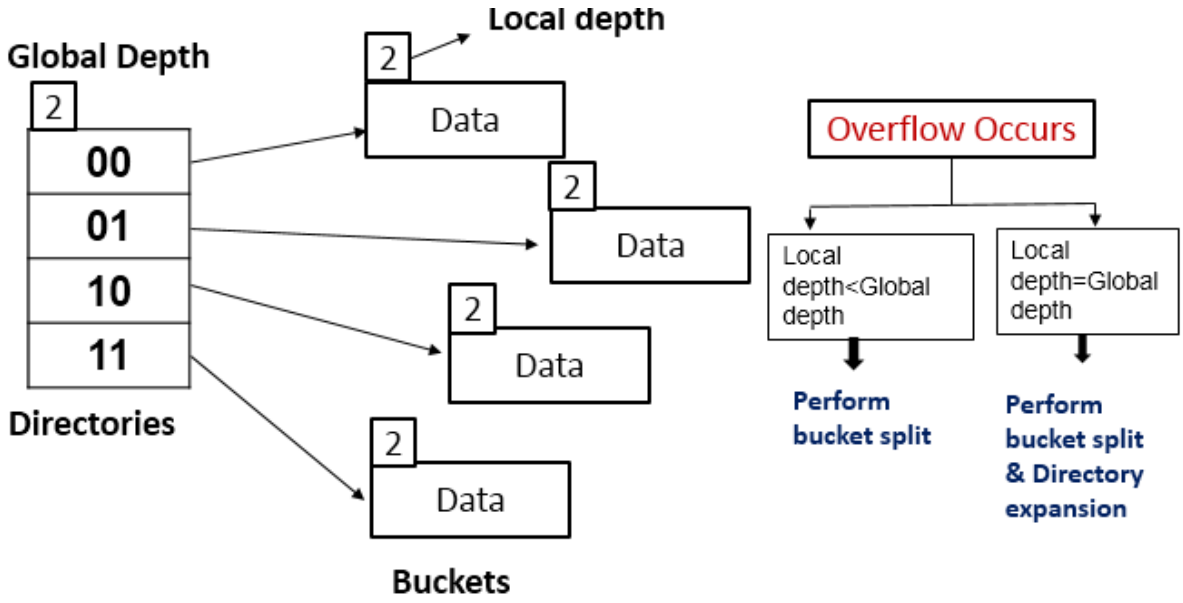


Dynamic and Extendible Hashing

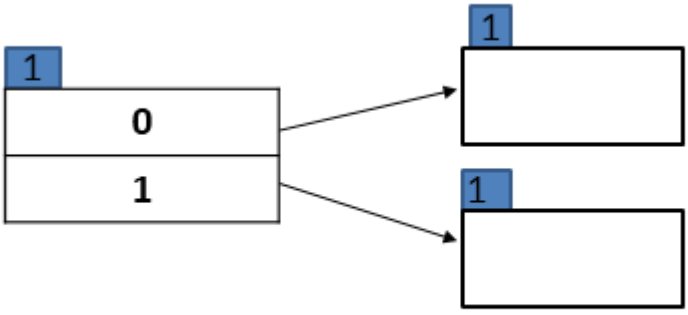
- Extendible Hashing is a dynamic hashing method in which directories, and buckets are used to hash data.
- It is a flexible method in which the hash function also experiences dynamic changes.

The main features in this hashing technique are:

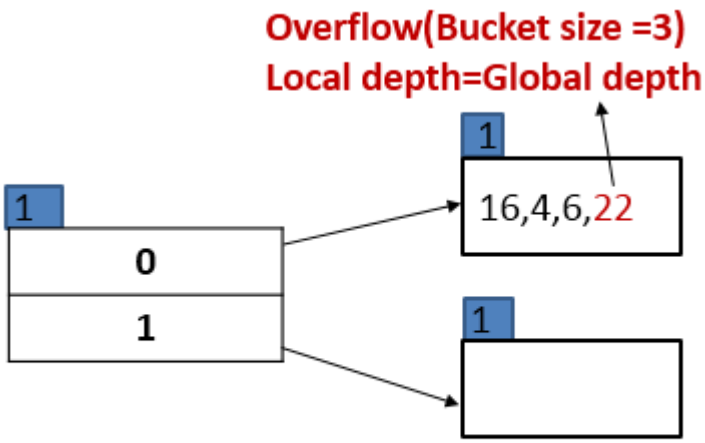
- Directories: The directories store addresses of the buckets in pointers. An id is assigned to each directory which may change each time when Directory Expansion takes place.
- Buckets: The buckets are used to hash the actual data.



Example :
Keys: 16,4,6,22,24,10.
Bucket Size: 3 (Assume)
Hash Function: Suppose the global depth is X. Then the Hash Function returns X LSBs.
Solution: First, calculate the binary forms of each of the given numbers.
16- 10000
4- 00100
6- 00110
22- 10110
24- 11000
10- 01010
Initially, the global-depth and local-depth is always 1. Thus, the hashing frame looks like this:

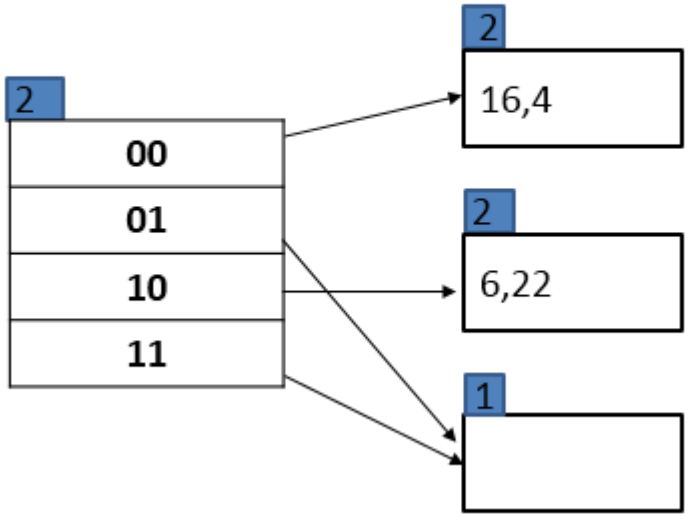


Hash Function: Suppose the global depth is X. Then the Hash Function returns X LSBs.
Solution: First, calculate the binary forms of all numbers.
16- 10000
4- 00100
6- 00110
22- 10110
24- 11000
10- 01010



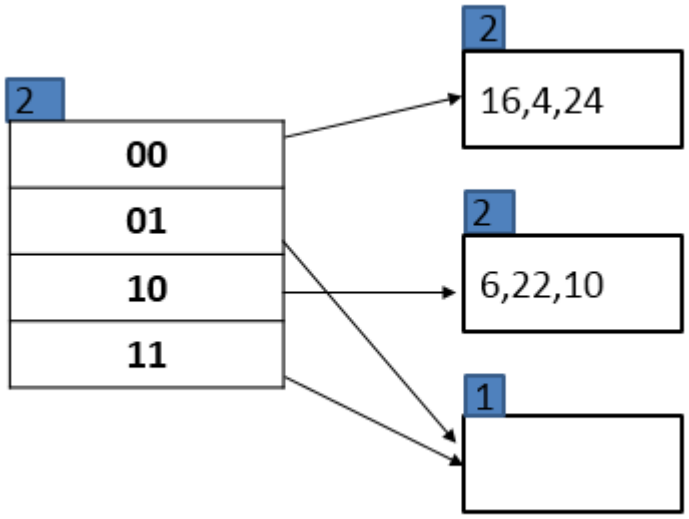
- Since Local Depth = Global Depth, the bucket splits and directory expansion takes place.
- Also, rehashing of numbers present in the overflowing bucket takes place after the split.
- And, since the global depth is incremented by 1, now,the global depth is 2. Hence, 16,4,6,22 are now rehashed w.r.t 2 LSBs.
  16(10000),4(100),6(110),22(10110)]

After Bucket split and Directory Expansion



- **Inserting 24 and 10:**

24(11000) and 10 (1010) can be hashed based on directories with id 00 and 10.



Advantages:

1. No problem of Data-loss since the storage capacity increases dynamically.
2. With dynamic changes in hashing function, associated old values are rehashed w.r.t the new hash function.

Limitations:

1. The directory size may increase significantly if several records are hashed on the same directory while keeping the record distribution non-uniform.
2. Size of every bucket is fixed.

Choosing a Hash Function

Criteria for choosing a good hash function

1. It should be easy to compute.
2. It should generate address with minimum collision i.e. it should distribute the keys as uniformly as possible in the hash table.

Perfect Hashing function

- Given a set of keys k= {k1,k2,k3 ,…..Kn} a perfect hash function is a hash function h such that h(ki)!=h(kj) for all distinct i and j.That is, no hash clashes occur under a perfect hash function.
- A perfect hash function is one to one mapping that guarantees zero collisions.
- Perfect hashing is very efficient if data is close to static and there are no much insertion and deletions.