MODULE 5

SORTING & HASHING

EDULINE
FOR CSE STUDENTS

Prepared By Mr. EBIN PM, AP, IESCE 1

HASHING

❖Hash Tables

- Hash Table is a data structure which stores data in an associative manner.
- In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.
- Hashing function is used to implement hash table
- Hash function returns a location in hash table
- Hash table contains buckets

Prepared By Mr.EBIN PM, AP, IESCE EDULINE 2

- Hash value is a bucket address of hash table
- One bucket can store more than one values
- N number of record can be stored in one bucket.
- One bucket has number of slots. In 2 slotted hash table, one bucket contains 2 values.

n = number of key values

b = number of buckets

s = number of slots

∴ identifier density = n/T, where T is total number of possible identifiers. And

Loading factor = n/sb

Consider 2 identifiers I_1 and I_2.

I_1 and I_2 are applied on a same hashing function and produce same bucket address. So I_1 and I_2 stored in same bucket. Then this identifier is called "synonyms"

0	**I_1**	**I_2**	← **Synonyms**
1			
2			
.			
.			
.			
25			

2 slotted hash table. Hash table with 26 buckets and two slots per bucket

- Suppose, consider the inputs A, B, A_1 , A_2 , C , B_3 , B_4 , A_3 . We apply a hash function **mode8 = r**, and the remainder of A=0, B=1 and C=2. So it can be stored as

0	**A**	**A_1**	← **Synonyms**
1	**B**	**A_2**	← A_2 is stored in the next position
2	**C**	**B_3**	
	B_4	**A_3**	
.			
.			
.			

- If any overflow occurs, the inputs are inserted into the next free space. When overflow occurs , the collision also occurs.

HASHING FUNCTIONS

1. **Mid Square**
2. **Division**
3. **Folding**
4. **Digit Analysis**

❖**Mid Square –** Let k=3205. The hash function squares the k .that is, $k^2 = (3205)^2$

$$= 102]72[025$$

- Take middle value. This middle value is the address of bucket. Mid square is applied , when only the bucket size is a power of 2.

❖Division

f(x) = x % m

- For reducing the collision we use prime numbers for m.
- The range of bucket address is 0 to m-1 (m is a constant, ie, hash table size)

❖Folding

- We divide the key into some parts and add each parts

Eg: 30|25|0

= 30+25+0 = 55

This 55 is take as a bucket address. This method is also called shift folding.

Folding at boundaries

- Here we also divide the key into some parts. We take the alternative reverse of the number and add it

Eg: 30|25|0

= 30+52+0 = 82

- 82 is taken as a bucket address

❖Digit Analysis

- This method is particularly useful in the case of a static file where all the identifiers in the table are known in advance.
- Each identifier x is interpreted as a number using some radix "r"

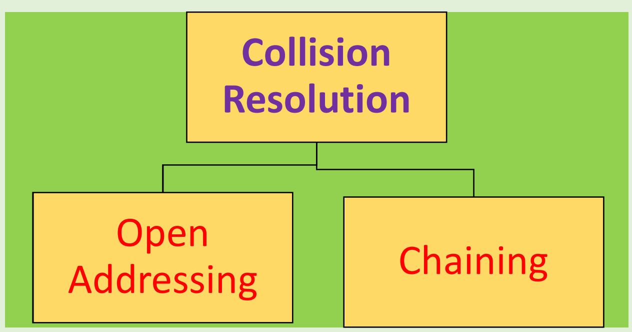
- This hashing function is distribution dependent
- All of the inputs that must be hashed are known in advance
- Here we make a statistical analysis of digits of the key, and select those digits (of fixed position) which occur quite frequently
- Then reverse or shift the digits to get the address

Eg: If the key is 9861234 . If the statistical analysis has revealed the fact that the third and fifth position digits occur quite frequently, then we choose the digits in these positions from the key. So we get 62. Reversing it , we get 26 as the address

Prepared By Mr.EBIN PM, AP, IESCE EDULINE 9

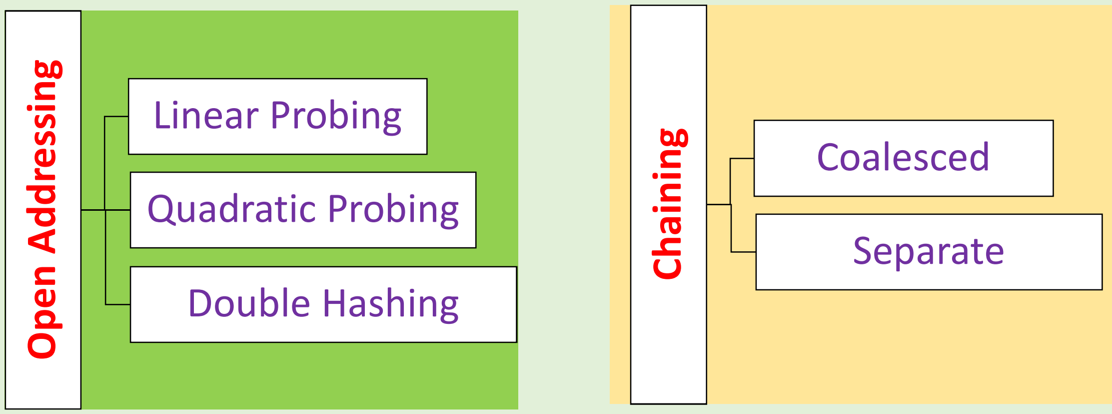
OVERFLOW HANDLING

- It is also called **Collision Resolution**. The main methods are



Prepared By Mr.EBIN PM, AP, IESCE EDULINE 10

Open addressing methods are **Chaining Methods are**



❖Open addressing

- If collision occurs when hashing performs, the values are transferred in to the alternative free location.

1. **Linear Probing** - Here, the values are transferred to the next location

   Eg: Hash Function = K mod 100 and

   Keys:- 50904, 78907, 68403, 86704, 7233

- One disadvantage is that, it form cluster of identifier .
- Here the hash table is consider as a circular list

00	
01	
02	
03	68403
04	50904
05	86704
06	
07	78907
33	7233

2. Quadratic Probing – It avoids the bad clusters. In linear probing, the searching is done in the sequence i+1, i+2, i+3 …….But in quadratic probing, quadratic function is used and the searching is done in the sequence i+2, i+4, i+6……It avoids the thick clusters

- Probe sequence is
  - h(k) mod size
  - (h(k) + 1) mod size
  - (h(k) + 4) mod size
  - (h(k) + 9) mod size
  - …

Eg: 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 and 76	Insert 85: Collision occurs. Insert at 1 + 1*1 position	Insert 92: Collision occurs at 1. Collision occurs at 1 + 1*1 position Insert at 1 + 2*2 position.
0			700	700	700
1		50	50	50	50
2				85	85
3					
4					
5					92
6			76	76	76

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

Insert 73 and 101

3. Double Hashing

- Here we use a series of hash functions $h_1, h_2 \ldots h_x$. Let $H(k) = h$.
- A collision is occurred when the key value is applied in a hash function.
- So the same key is applied in to another hash function $H'(k) = h'$ and the searching is done.
- A popular second hash function is: **Hash2(key) = R - (key % R)** where R is a prime number that is smaller than the size of the table.
- The following function is another example of double hashing:

  (firstHash(key) + i * secondHash(key)) % tableSize

In the computation above, the value of i will keep incrementing (the offset will keep increasing) until an empty slot is found.

❖Chaining

1. Coalesced Chaining

Here, the memory area is divided in to 2 parts.

- Prime area and Overflow area

  Key= 22,31,67,36,29,60

  h(k') = k % 7

- 36%7=1. But 22 is already filled in the position 1. So 36%7=1 is considered as an overflow. So the value is stored in overflow area

Locn	Key	Data	Link
PRIME AREA			
0			
1	22		5
2			
3	31		NULL
4	67		7
OVERFLOW AREA			
5	36		NULL
6	29		NULL
7	60		NULL
8			

- The disadvantage of coalesced chaining is that the table is fixed one.
- If the number of key values are increased, the table is not able to hold the keys.
- To avoid this problem, we use separate chaining method.

2. Separate chaining

- The hash table is implemented using some header nodes and list nodes . The table is a header node.

Eg: 80, 92, 25, 76, 48, 150, 215, 216, 38

hash function = mod10

The remaining positions are filled with null values