Mahavir Education Trust's

SHAH & ANCHOR KUTCHHI ENGINEERING COLLEGE

Affiliated to University of Mumbai, Approved by D.T.E. & A.I.C.T.E.| Awarded 'A' Grade by D.T.E., M.S. | Electronics Engineering Program Accredited by N.B.A., New Delhi for 2 years w.e.f. 6th Aug.,2014 | Computer Engineering Program Re-Accreditated by N.B.A.,New Delhi for 3 yeas w.e.f. 1st July 2019 | Information Technology Program Accredited by N.B.A.,New Delhi for 3 years w.e.f. 1st July 2019

DEPARTMENT OF COMPUTER ENGINEERING

ISO 9001 Certified


# Patil Shweta

# Asst.Prof

# Computer Engineering Department

# SAKEC, Chembur

# Data Structures

# Searching Techniques

# Module-6

# Lecture Objectives

1. Understand Concept of Hashing
2. Understand various Hash Functions
3. Understand collision and various collision handling techniques

# Lecture Outline

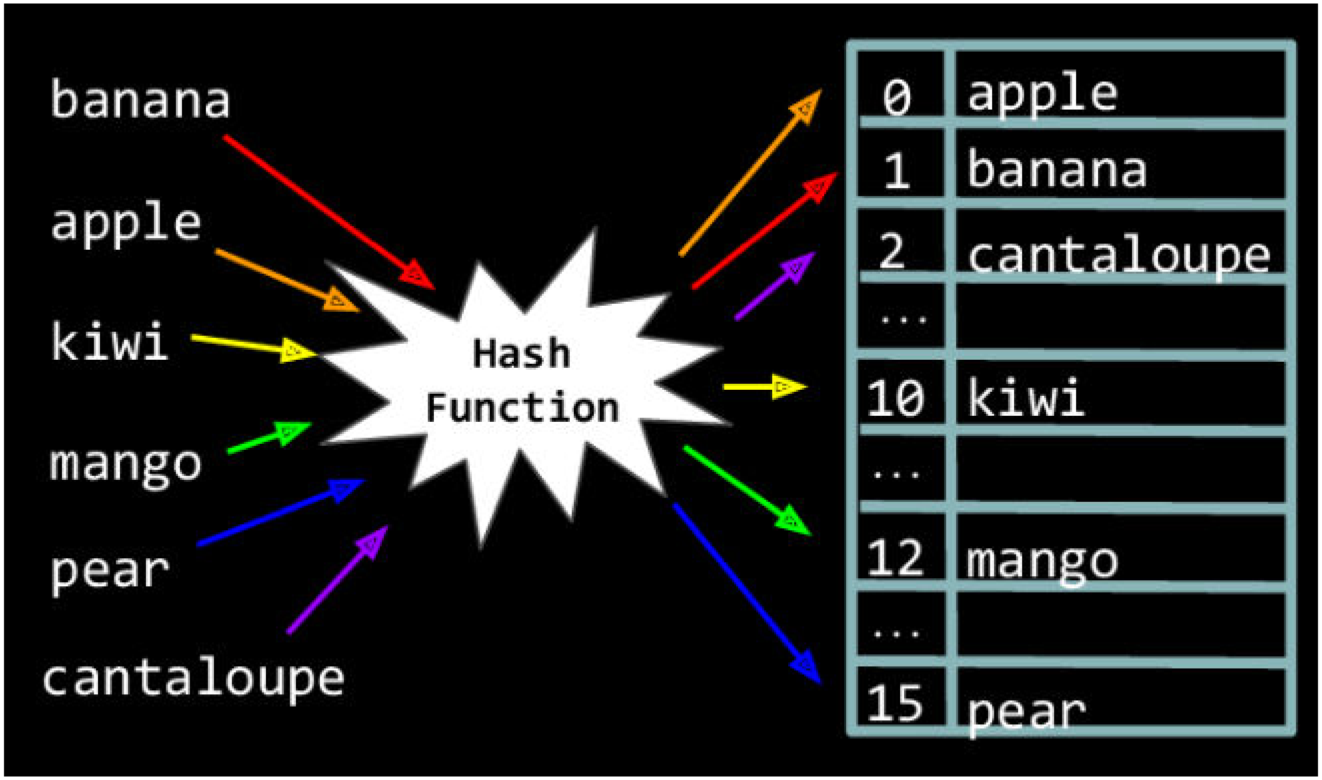
01 Hashing: Concept

02 Hashing functions

03 Collision handling techniques

# Hashing

- Hash table is a data structure in which keys are mapped to array positions by a hash function.




https://www.youtube.com/watch?v=t2a5UVb6SFo

# Hashing

- In hash table, an element with key k is stored at index h(k).
- That means hash function h is used to calculate the index in which the key will be stored.
- This process of mapping key to index position is called as **Hashing**
- When two or more keys are mapped to same index then that problem is called as **Collision**
- Main aim of using hash function is to reduce the range of array indices i.e reduce the storage space.

# Hashing

- A **hash function** is a mathematical formula which, when applied to a key, produces an integer which can be used as an index for the key in the hash table
- **Several types of Hash functions:**
  - Division Method
  - Multiplication Method
  - Mid-square Method
  - Folding Method

# Hash Function

**Division Method**

- This method divides x by M and then uses the remainder obtained.

$$h(x) = x \bmod M$$

- It is always wise to choose M as even number
- Drawback: consecutive keys map to consecutive hash values.
- **Example:** Calculate the hash values of keys 1324 and 5642 (Hint: use M as prime number eg:97)

**Ans:** h(1324) = 63, h(5642) = 16

# Hash Function

**Multiplication Method**

- It works in steps
  1. Choose A as 0< A < 1
  2. Multiply key k by A
  3. Extract fractional part of kA
  4. Multiply result of  by size of hash table (m)
- Knuth has suggested the best value to be used for A as 0.618033
- **Example:** Given a hash table of size 1000, map key 12345 to an appropriate location in the hash table

**Ans:** h(12345) = 617

# Hash Function

**Mid-Square Method**

- It works in steps
  1. Square the value of key
  2. Extract the middle r digits of the result obtained in 1.
- **Example:** Given a hash table of with 100 memory locations, map key 1234 and 5642 to an appropriate location in the hash table

**Ans:** h(1234) = 27, h(5642) = 32

# Hash Function

**Folding Method**

- It works in steps
  1. Divide key k value into number of parts, where each part has equal digits except the last part can have lesser digits
  2. Add all parts. The hash value is the one after ignoring the carry if any
- **Example:** Given a hash table of with 100 memory locations, map key 5678, 321 and 34567 to an appropriate location in the hash table

**Ans:** h(5678) = 34, h(321) = 33, h(34567) =97

# Collisions

- **Collision Resolution Techniques:**
  - Open Addressing
  - Chaining

# Collision Resolution Techniques: Open Addressing

- **Linear Probing:**
  - Function: h(k , i) = [h' (k) + i] mod m

    where m is size of hash table, h'(k) = (k mod m) and I is the probe number
  - Example: Consider a hash table of size 10. Using Linear probing, insert the keys 72, 27, 36, 24, 63, 81, 92 and 101 into the table

| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| -1 | 81 | 72 | 63 | 24 | 92 | 36 | 27 | 101 | -1 |

  - Drawback: Primary Clustering

  - Solved problem

# Collision Resolution Techniques: Open Addressing

- **Quadratic Probing:**
  - Function: h(k , i) = [h' (k) + c1i + c2 (i)2] mod m

    where m is size of hash table, h'(k) = (k mod m) and i is the probe number which varies from 0 to m-1, c1 and c2 are constants other than 0
  - Example: Consider a hash table of size 10. Using Linear probing, insert the keys 72, 27, 36, 24, 63, 81 and 101 into the table. Take c1= 1 and c2 = 3

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| -1 | 81 | 72 | 63 | 24 | 101 | 36 | 27 | -1 | -1 |

  - Drawback: Successive probing explore only fraction of the table
  - Application: Widely used in **Berkley Fast File System** to allocate free blocks

– -Solved Problems

# Collision Resolution Techniques: Open Addressing

- **Double Hashing:**
  - Function: h(k , i) = [h1 (k) + i h2 (k)] mod m

    where m is size of hash table, h1(k) = (k mod m), h2(k) = (k mod m') and i is the probe number which varies from 0 to m-1, m' is value less than m.
  - Example: Consider a hash table of size 10. Using Linear probing, insert the keys 72, 27, 36, 24, 63, 81, 92 and 101 into the table. Take h1= (k mod 10) and h2= (k mod 8)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **92** | **81** | **72** | **63** | **24** | **-1** | **36** | **27** | **-1** | **-1** |

  - Advantage: Minimizes repeated collisions and the effects of clustering

– -Solved Problems

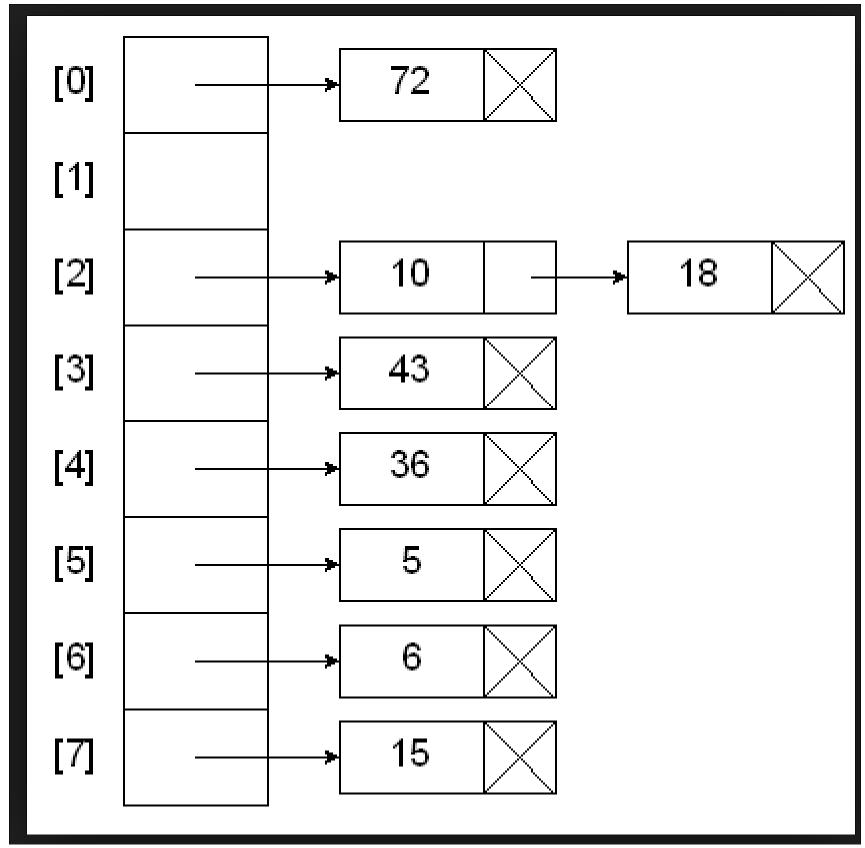
# Collision Resolution Techniques: Open Addressing

- **Rehashing:**
  - Hash table nearly full; collision increases; thereby degrading performance.
  - Create new hash table with double value
  - Move all values from old hash table to new one
  - Cons: To expensive

# Collision Resolution Technique: Chaining

- **Advantages:** Number of key values wont affect the number of locations in the hash table

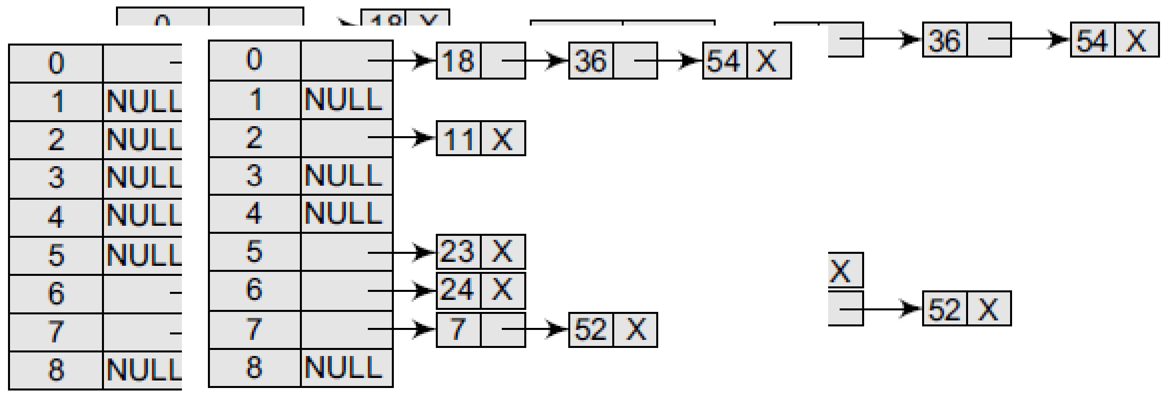
- **Disadvantages:**
  - Overhead of storing pointers
  - Poor cache performance since traversing the linked list



# Collision Resolution Technique: Chaining

Insert the keys 7, 24, 18, 52, 36, 54, 11, and 23 in a chained hash table of 9 memory locations. Use h(k) = k mod m.

| | |
|---|---|
| 0 | NULL |
| 1 | NULL |
| 2 | NULL |
| 3 | NULL |
| 4 | NULL |
| 5 | NULL |
| 6 | NULL |
| 7 | NULL |
| 8 | NULL |



# Bucket Hashing

- Divide M slots of hash table into B buckets
- So, each bucket has M / B slots
- Calculate position for key using hash function
- Slot free; allocate
- Else put the key in "***Overflow bucket***"

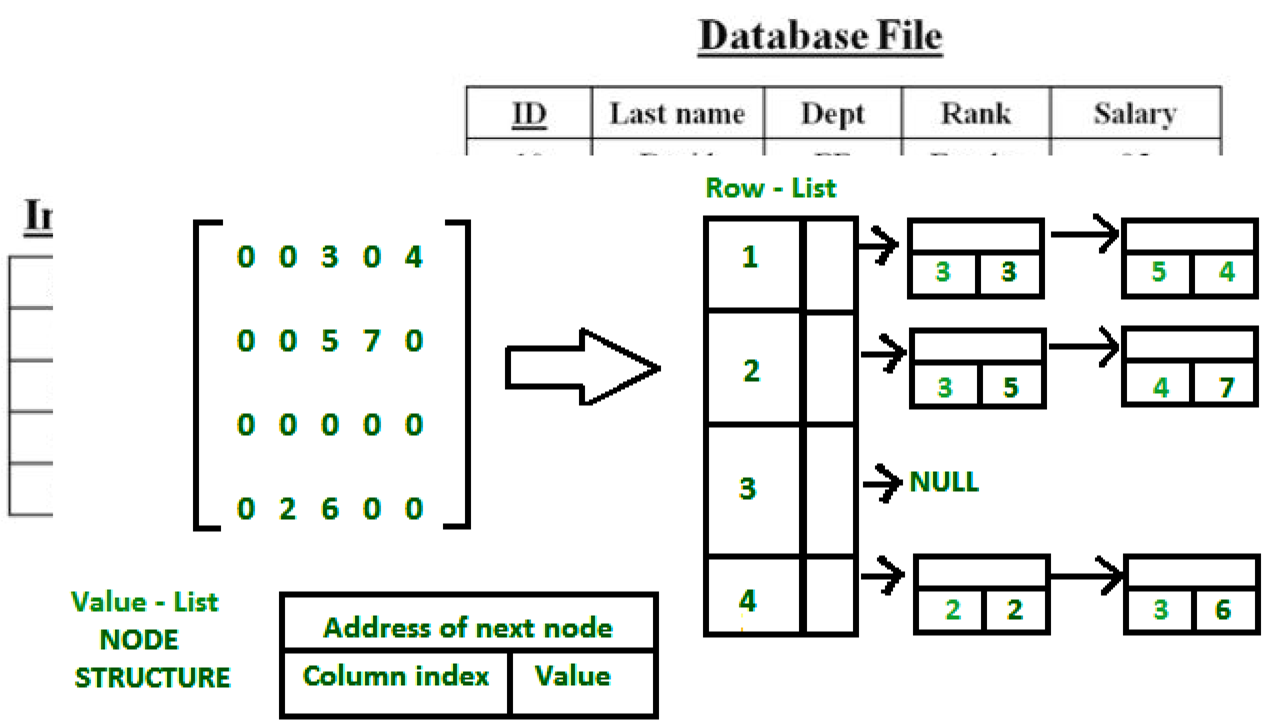


- Cons: If key not found in bucket; searching key in

Overflow bucket is expensive

# Applications

- Database Indexing
- Compiler: Symbol table
- Driving Card
- Sparse Matrix



# *THANK YOU..*

**Any queries**

**(shweta.patil@sakec.ac.in)**

**Reference:**