# DATA STRUCTURES & ALGORITHMS

Hash Tables

Instructor: Engr. Laraib Siddiqui

# Hash Table

A data structure that provides a mapping from key to value using a technique called hashing.

The keys are used for indexing the values/data.

Effective way of implementing dictionaries.

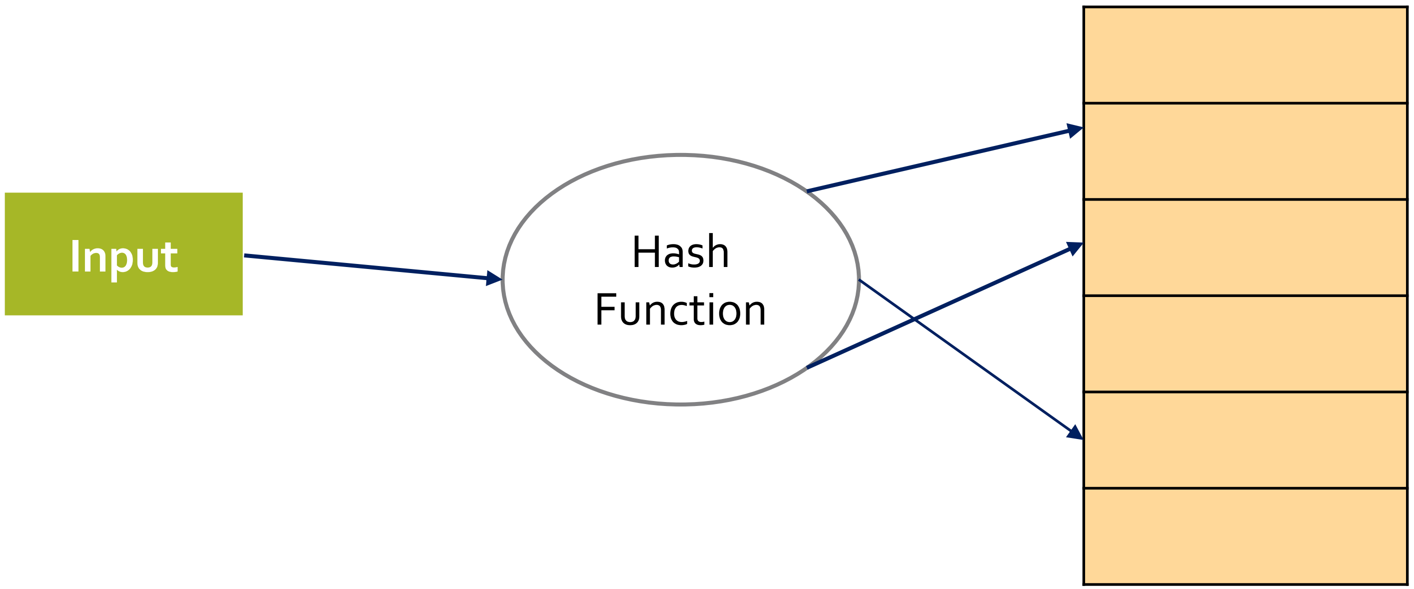
| Key | Value |
|---|---|

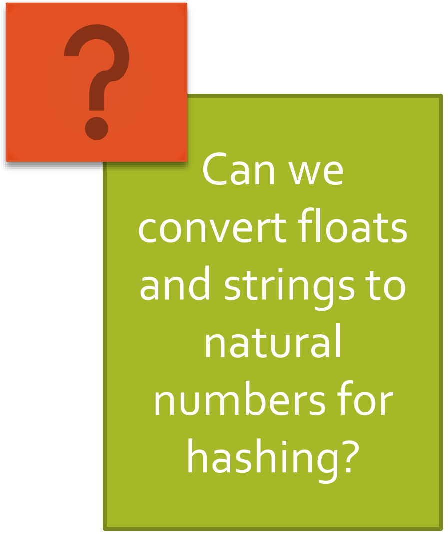
Key - value pairs must be unique, but values can be repeated.

# Hash Function

A hash function H(x) is a function that maps a key 'x' to a whole numbers in fixed range.

Choosing an efficient hash function is a crucial part of creating a good hash table.



# Hashing Methods

- ✓ Direct hashing
- ✓ Modulo division
- ✓ Mid square method
- ✓ Folding method
- ✓ Rotation Method
- ✓ Pseudo Random Methods

# Modulo Division Method

Map a key into one of the m slots by taking the remainder of k divided by m that is

$$H(k) = k \bmod m$$

Where, m can be the list size

*Example:*

*m= 31 and k = 78*

*78 mod 31*

*H(K) = 16*

*<u>With Strings</u>*

*Cat*

*ASCII : c = 99, a = 97, t = 116*

*99 + 95 + 116 = 312*

*If, m= 11 and k = 312*

*Then, H(K) = 4*

# Mid Square Method

The key k is multiplied by itself and the address is obtained by selecting an appropriate number if digits from the middle of the square.

*Example:*

$9452 * 9452 = 89340304$

$H(k) = 3403$

# Folding Method

The key is partitioned into a number of parts, each of which has the same length as the required address with the possible exception of the last part.

*Example:*

*123456789*

$123 + 456 + 789 = \cancel{1}368$

$H(k) = 368$

# Digit Exaction Method

Selected digits are extracted from the key and used as the address.

Address = extracted digits from key

*Example:*

*121267*

*H(k) = 112*

# Rotation Method

Usually used with the combination of other hashing mechanisms.

Example:

*25008891*

*H(k) = 1250089*

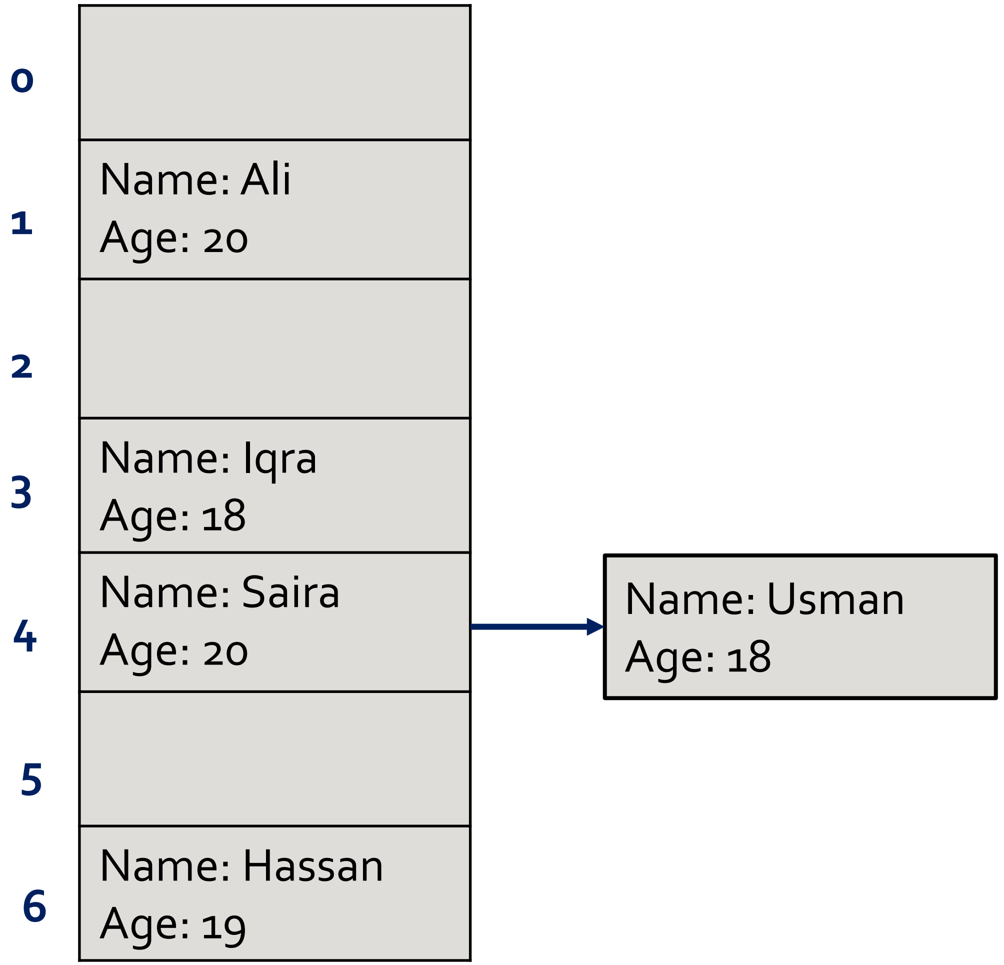
# Collision

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

# Collision Resolutions

## Separate Chaining

Deals with collisions by maintaining a data structure to hold all the different values which hashed to a particular value.



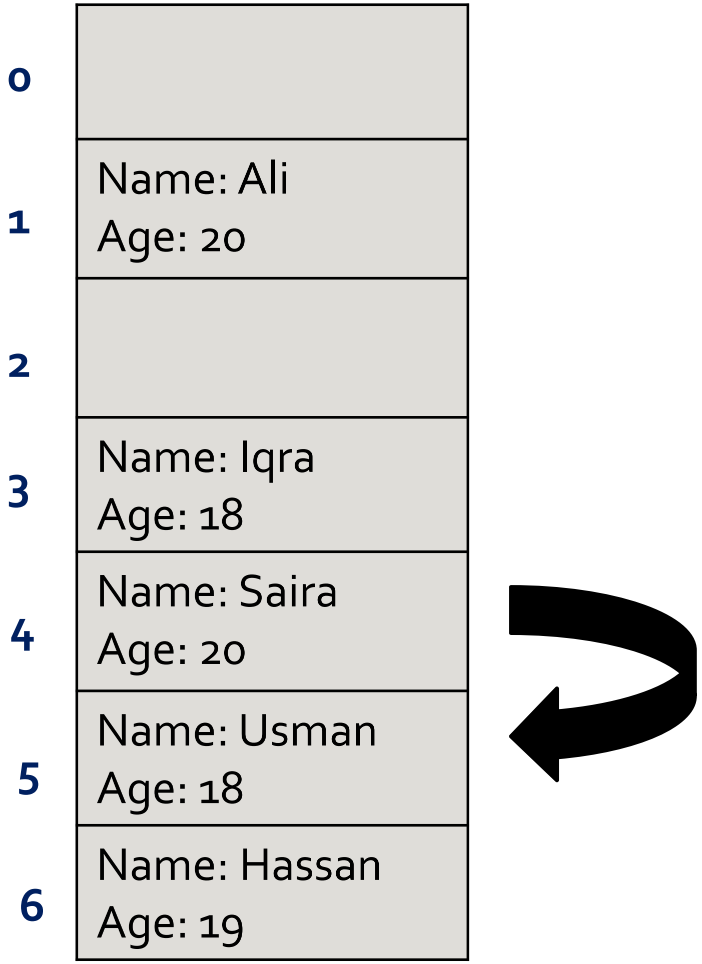
# Collision Resolutions

## Open Addressing

Deals with hash collisions the hash table for all the object to go by offsetting if from the position to which it hashed to.

Types

- Linear probing
- Quadratic probing
- Double Hashing



# Collision Resolutions

Open addressing

- ✓ Linear Probing

  If collision occurs, we linearly probe for next slot

- ✓ Quadratic Probing

  If collision occurs:

  (hash value + $1^2$) % table size

  if again collision occurs

  (hash value + $2^2$) % table size

  If again

  (hash value + $3^2$) % table size

  In general : (hash value + $i^2$) % table size

**Insert {89, 18, 49, 58, 69, 78} with h(x) =x mod 10 using separate chaining.**

x = 89

h(89) = 89 mod 10 => 9

x = 18

h(18) = 18 mod 10 => 8

x = 49

h(49) = 49 mod 10 => 9 (C)

x= 58

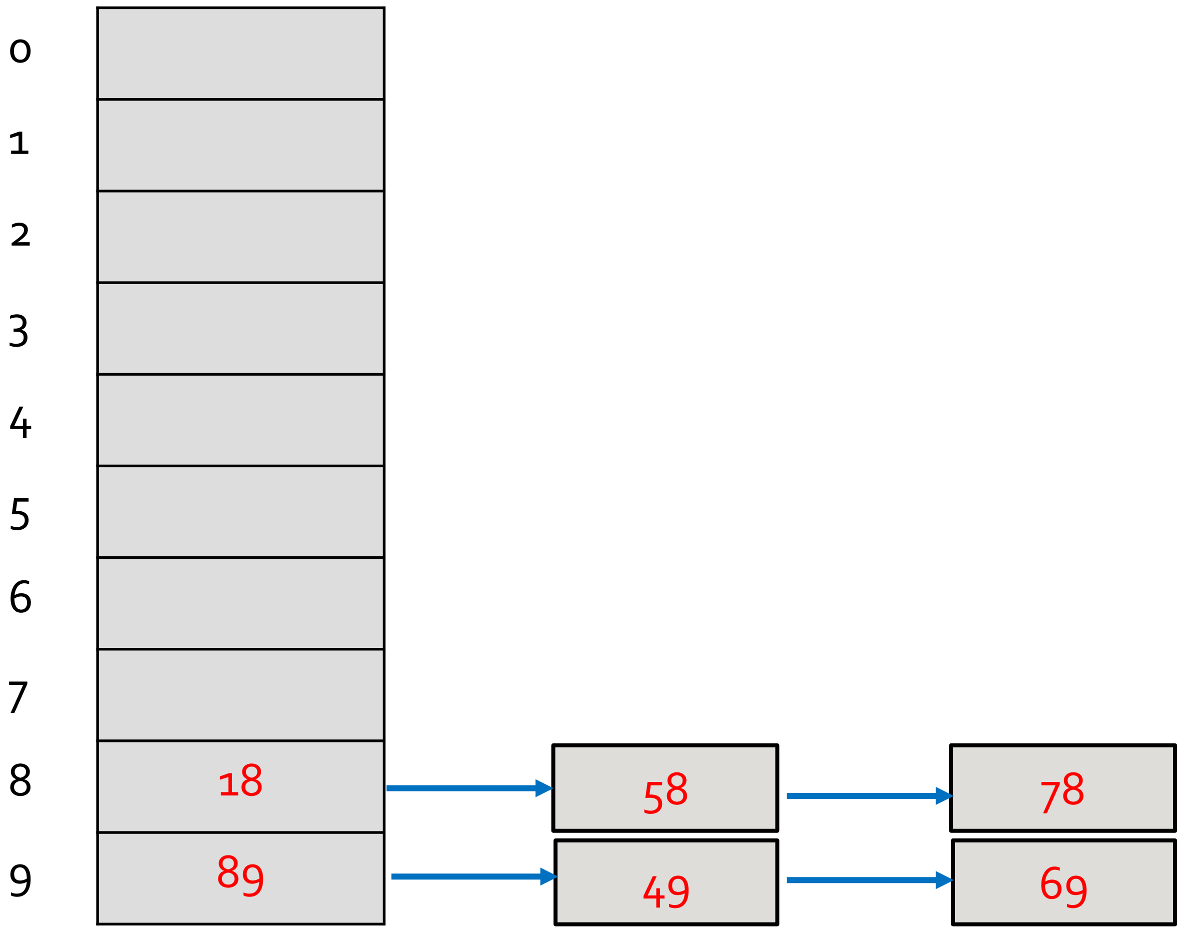
h(58) = 58 mod 10 => 8 (C)

x = 69

h(69) = 69 mod 10 => 9 (C)

x = 78

h(78) = 78 mod 10 => 8 (C)



**Insert {89, 18, 49, 58, 69, 78} with h(x) =x mod 10 using linear probing.**

x = 89

h(89) = 89 mod 10 => 9

x = 18

h(18) = 18 mod 10 => 8

x = 49

h(49) = 49 mod 10 => 9 (C)

x= 58

h(58) = 58 mod 10 => 8 (C)

x = 69

h(69) = 69 mod 10 => 9 (C)

x = 78

h(78) = 78 mod 10 => 8 (C)

| | |
|---|---|
| 0 | 49 |
| 1 | 58 |
| 2 | 69 |
| 3 | 78 |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | 18 |
| 9 | 89 |

**Insert {89, 18, 49, 58, 69, 78} with h(x) =x mod 10 using quadratic probing.**

x = 89

$h(89) = 89 \bmod 10 \Rightarrow 9$

x = 18

$h(18) = 18 \bmod 10 \Rightarrow 8$

x = 49

$h(49) = 49 \bmod 10 \Rightarrow 9$ (collision)

$h1(49) = (9 + 1^2) \bmod 10 \Rightarrow 0$

x= 58

$h(58) = 58 \bmod 10 \Rightarrow 8$ (collision)

$h1(58) = (8 + 1^2) \bmod 10 \Rightarrow 9$ (collision)

$h2(58) = (8 + 2^2) \bmod 10 \Rightarrow 2$

x = 69

$h(69) = 69 \bmod 10 \Rightarrow 9$ (collision)

$h1(69) = (9 + 1^2) \bmod 10 \Rightarrow 0$ (collision)

$h2(69) = (9 + 2^2) \bmod 10 \Rightarrow 3$

x = 78

$h(78) = 78 \bmod 10 \Rightarrow 8$ (collision)

$h1(78) = (8 + 1^2) \bmod 10 \Rightarrow 9$ (collision)

$h2(78) = (8 + 2^2) \bmod 10 \Rightarrow 2$ (collision)

$h3(78) = (8 + 3^2) \bmod 10 \Rightarrow 7$

| | |
|---|---|
| 0 | 49 |
| 1 | |
| 2 | 58 |
| 3 | 69 |
| 4 | |
| 5 | |
| 6 | |
| 7 | 78 |
| 8 | 18 |
| 9 | 89 |

# Exercise

Map all integer key to the range (0,9 ), $h(x) = (x^2 - 6x + 9) \bmod 10$

x = 4

$h(4) = ( 4^2 - 6(4) + 9 ) \bmod 10 \Rightarrow 1$

x = 7

???

x = 0

???

x = 2

???

x = 8

???

# Applications

- File management
- Comparing complex values
- Cryptography
- Security systems

# Hash Table Complexity

| | |
|---|---|
| Search | O(n) |
| Insertion | O(n) |
| Deletion | O(n) |