# Hashing, Hash Function and Hash Table

# What is Hashing

- Hashing is a technique that is used to uniquely identify a specific object from a group of similar objects.

- Examples of how hashing is used in our lives:
  - In universities, each student is assigned a unique roll number that can be used to retrieve information about them.
  - In libraries, each book is assigned a unique number that can be used to determine information about the book

- **Hashing** is the solution that can be used in almost all such situations and performs extremely well compared to above data structures like Array, Linked List, BST in practice

# What is Hashing (Cond…)

- With hashing we get O(1) search time on average and O(n) in worst case.

- The idea is to use **hash function** that converts a given input number or any other key to a smaller number and uses the small number as index in a table called **hash table**.

- Hash Function
  - Hash function maps a big number or string to a small integer that can be used as index in hash table

- Hash Table
  - An array that stores pointers to records corresponding to a given input number

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Mia    M    77    i    105    a    97    279    4

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

0    1    2    3    4    5    6    7    8    9    10

Mia      M    77    i    105    a    97     279     4

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|----|
|   |   |   |   | Mia |   |   |   |   |   |   |

| Mia | M | 77 | i | 105 | a | 97 | 279 | 4 |
| Tim | T | 84 | i | 105 | m | 109 | 298 | 1 |

| | Tim | | | Mia | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

| Name | Letter | | Char | | Char | | Sum | Index |
|------|--------|----|------|-----|------|-----|-----|-------|
| Mia | M | 77 | i | 105 | a | 97 | 279 | 4 |
| Tim | T | 84 | i | 105 | m | 109 | 298 | 1 |
| Bea | B | 66 | e | 101 | a | 97 | 264 | 0 |
| Zoe | Z | 90 | o | 111 | e | 101 | 302 | 5 |
| Jan | J | 74 | a | 97 | n | 110 | 281 | 6 |
| Ada | A | 65 | d | 100 | a | 97 | 262 | 9 |
| Leo | L | 76 | e | 101 | o | 111 | 288 | 2 |

| Bea | Tim | Leo | | Mia | Zoe | Jan | | | Ada | |
|-----|-----|-----|---|-----|-----|-----|---|---|-----|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

| Mia | M | 77 | i | 105 | a | 97 | 279 | 4 |
|-----|---|----|----|-----|----|-----|-----|----|
| Tim | T | 84 | i | 105 | m | 109 | 298 | 1 |
| Bea | B | 66 | e | 101 | a | 97 | 264 | 0 |
| Zoe | Z | 90 | o | 111 | e | 101 | 302 | 5 |
| Jan | J | 74 | a | 97 | n | 110 | 281 | 6 |
| Ada | A | 65 | d | 100 | a | 97 | 262 | 9 |
| Leo | L | 76 | e | 101 | o | 111 | 288 | 2 |
| Sam | S | 83 | a | 97 | m | 109 | 289 | 3 |
| Lou | L | 76 | o | 111 | u | 117 | 304 | 7 |
| Max | M | 77 | a | 97 | x | 120 | 294 | 8 |
| Ted | T | 84 | e | 101 | d | 100 | 285 | 10 |

| Bea | Tim | Leo | Sam | Mia | Zoe | Jan | Lou | Max | Ada | Ted |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

**Index number** = *sum ASCII codes* Mod *size of array*

| Bea | Tim | Leo | Sam | Mia | Zoe | Jan | Lou | Max | Ada | Ted |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  |

Find Ada

Ada = (65 + 100 + 97) = 262

262 Mod 11 = 9

myData = Array(9)

| Bea | Tim | Leo | Sam | Mia | Zoe | Jan | Lou | Max | Ada | Ted |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

| Bea | Tim | Leo | Sam | Mia | Zoe | Jan | Lou | Max | Ada | Ted |
|------|------|------|------|------|------|------|------|------|------|------|
| 27/01/1941 | 08/06/1955 | 31/12/1945 | 27/04/1791 | 20/02/1986 | 19/06/1978 | 13/02/1956 | 27/12/1822 | 23/04/1858 | 10/12/1815 | 17/06/1937 |
| English | English | American | American | Russian | American | Polish | French | German | English | American |
| Astronomer | Inventor | Mathematician | Inventor | Space Station | Actress | Logician | Biologist | Physicist | Mathematician | Philosopher |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

# Hashing Algorithm

- Calculation applied to a key to transform it into an address

- For numeric keys, divide the key by the number of available addresses, $n$, and take the remainder

$$\text{address} = key \text{ Mod } n$$

- For alphanumeric keys, divide the sum of ASCII codes in a key by the number of available addresses, $n$, and take the remainder

- Folding method divides key into equal parts then adds the parts together
  - The telephone number 01452 8345654, becomes $01 + 45 + 28 + 34 + 56 + 54 = 218$
  - Depending on size of table, may then divide by some constant and take remainder

# Understand Collision

| Mia | M | 77 | i | 105 | a | 97 | 279 | 4 |
| Tim | T | 84 | i | 105 | m | 109 | 298 | 1 |
| Bea | B | 66 | e | 101 | a | 97 | 264 | 0 |
| Zoe | Z | 90 | o | 111 | e | 101 | 302 | 5 |

| Bea | Tim | | | Mia | Zoe | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

# Understand Collision

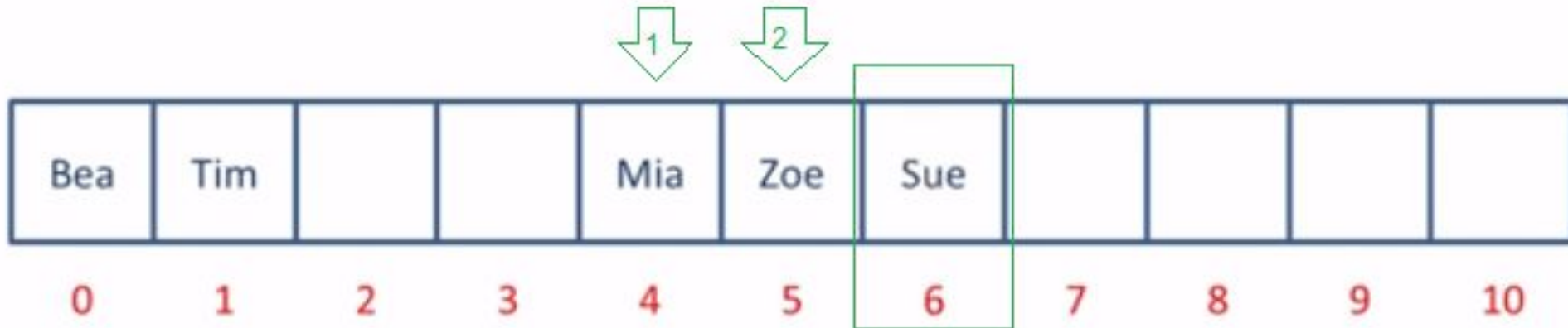| Name | Letter | Code | Letter | Code | Letter | Code | Sum | Mod |
|------|--------|------|--------|------|--------|------|-----|-----|
| Mia | M | 77 | i | 105 | a | 97 | 279 | 4 |
| Tim | T | 84 | i | 105 | m | 109 | 298 | 1 |
| Bea | B | 66 | e | 101 | a | 97 | 264 | 0 |
| Zoe | Z | 90 | o | 111 | e | 101 | 302 | 5 |
| Sue | S | 83 | u | 117 | e | 101 | 301 | 4 |

| Bea | Tim |  |  | Mia | Zoe |  |  |  |  |  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

# Collision Handling

- Irrespective of how good a hash function is, collisions are bound to occur.

- Therefore, to maintain the performance of a hash table, it is important to manage collisions through various collision resolution techniques.
  - Open addressing
    - Linear Probing
    - Quadratic Probing
    - Double Hashing
  - Closed addressing

# Open Addressing (Linear)

| Mia | M | 77 | i | 105 | a | 97 | 279 | 4 |
|-----|---|----|---|-----|---|-----|-----|---|
| Tim | T | 84 | i | 105 | m | 109 | 298 | 1 |
| Bea | B | 66 | e | 101 | a | 97 | 264 | 0 |
| Zoe | Z | 90 | o | 111 | e | 101 | 302 | 5 |
| Sue | S | 83 | u | 117 | e | 101 | 301 | 4 |

| Mia | M | 77 | i | 105 | a | 97 | 279 | 4 |
| Tim | T | 84 | i | 105 | m | 109 | 298 | 1 |
| Bea | B | 66 | e | 101 | a | 97 | 264 | 0 |
| Zoe | Z | 90 | o | 111 | e | 101 | 302 | 5 |
| Sue | S | 83 | u | 117 | e | 101 | 301 | 4 |
| Len | L | 76 | e | 101 | n | 110 | 287 | 1 |

| Bea | Tim | Len | | Mia | Zoe | Sue | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

| | | | | | | | | |
|------|---|----|---|-----|---|-----|-----|---|
| Mia  | M | 77 | i | 105 | a | 97  | 279 | 4 |
| Tim  | T | 84 | i | 105 | m | 109 | 298 | 1 |
| Bea  | B | 66 | e | 101 | a | 97  | 264 | 0 |
| Zoe  | Z | 90 | o | 111 | e | 101 | 302 | 5 |
| Sue  | S | 83 | u | 117 | e | 101 | 301 | 4 |
| Len  | L | 76 | e | 101 | n | 110 | 287 | 1 |
| Moe  | M | 77 | o | 111 | e | 101 | 289 | 3 |
| Lou  | L | 76 | o | 111 | u | 117 | 304 | 7 |
| Rae  | R | 82 | a | 97  | e | 101 | 280 | 5 |
| Max  | M | 77 | a | 97  | x | 120 | 294 | 8 |
| Tod  | T | 84 | o | 111 | d | 100 | 295 | 9 |

| Bea | Tim | Len | Moe | Mia | Zoe | Sue | Lou | Rae | Max | Tod |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  |

Find Rae

$$Rae = (82 + 97 + 101) = 280$$

$$280 \text{ Mod } 11 = 5$$

$$myData = Array(5)$$

| Bea | Tim | Len | Moe | Mia | Zoe | Sue | Lou | Rae | Max | Tod |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Find Rae

$$Rae = (82 + 97 + 101) = 280$$

$$280 \text{ Mod } 11 = 5$$

$$myData = Array(5)$$

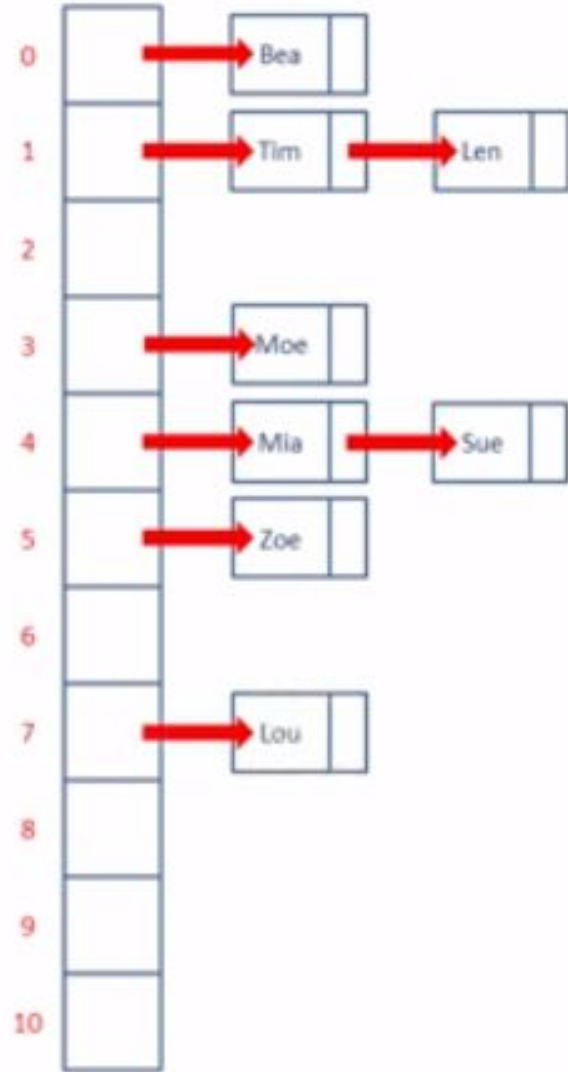| Bea | Tim | Len | Moe | Mia | Zoe | Sue | Lou | Rae | Max | Tod |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

# Closed Addressing (Non-Linear)

| | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | → Mia |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |

Mia        M    77      i    105      a    97      279      4

| Mia | M | 77 | i | 105 | a | 97 | 279 | 4 |
| Tim | T | 84 | i | 105 | m | 109 | 298 | 1 |
| Bea | B | 66 | e | 101 | a | 97 | 264 | 0 |
| Zoe | Z | 90 | o | 111 | e | 101 | 302 | 5 |
| Sue | S | 83 | u | 117 | e | 101 | 301 | 4 |
| Len | L | 76 | e | 101 | n | 110 | 287 | 1 |
| Moe | M | 77 | o | 111 | e | 101 | 289 | 3 |
| Lou | L | 76 | o | 111 | u | 117 | 304 | 7 |
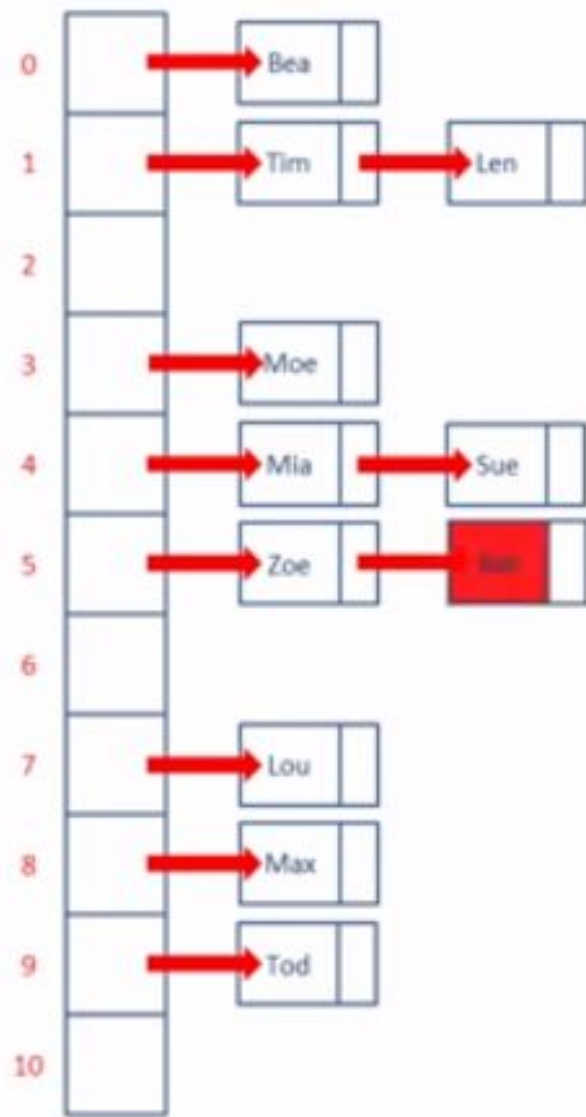
Find Rae      280 Mod 11 = 5

myData = Array(5)

# Objectives of Hash Function

- Minimize collisions

- Uniform distribution of hash values

- Easy to calculate

- Resolve any collisions