Hash Table

ITD62-124 Data Structure

Outline:

- Introduction
- Hash Functions
- Collision Resolution
  - ✓Separate Chaining
  - ✓Open Addressing
- Rehashing

Introduction:

- Hashing is a method for storing large data in the table
- Hashing functions are arithmetic methods for address finding of hash table
- Hash table is a table to store data
- Key is a code of data that will be stored in the table
- Address is an index of the array
- Collision:
  - ✓ address of two data are similar
  - ✓ the address already stored other data

Introduction:

- Each cell of the table will have an address for referencing
- Hash table can be implemented by array or linked list
- Operations of Hash table:
  - ✓ Insert
  - ✓ Delete
  - ✓ Find

Hash Function:

- Direct Hashing
- Subtraction
- Modulo-Division
- Mid-Square
- Shift Folding
- Boundary Folding

Hash Function:

❑ Direct Hashing

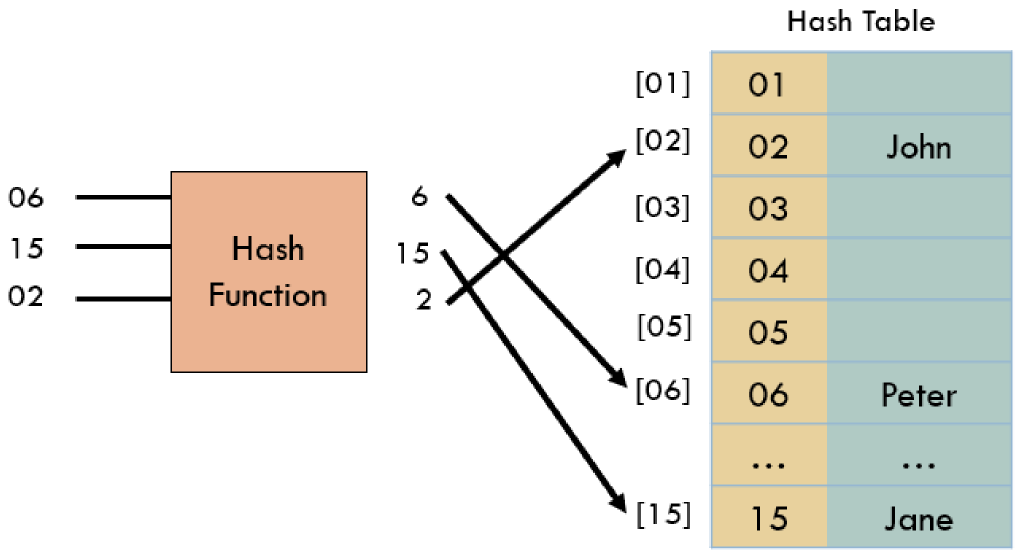
- Advantage: Easy and collision free
- Disadvantage: boundary of key
- Assumptions:
  - ✓ Key values are distinct
  - ✓ Each key is drawn from a universe U = {0, 1, . . . , m - 1}
- Idea:
  - ✓ Store the items in an array, indexed by keys

Hash Function:

- Direct-hashing table representation:
  - ✓ An array T[0 . . m - 1]
  - ✓ Each slot, or position, in T corresponds to a key in U
  - ✓ For an element x with key k, a pointer to x will be placed in location T[k]
  - ✓ If there are no elements with key k in the set, T[k] is empty, represented by NIL

Hash Function:

✓ Example:



Hash Function:

- Subtraction
  - Close to Direct Hashing
  - Using this technique when key does not start with the value 01
  - Solution:

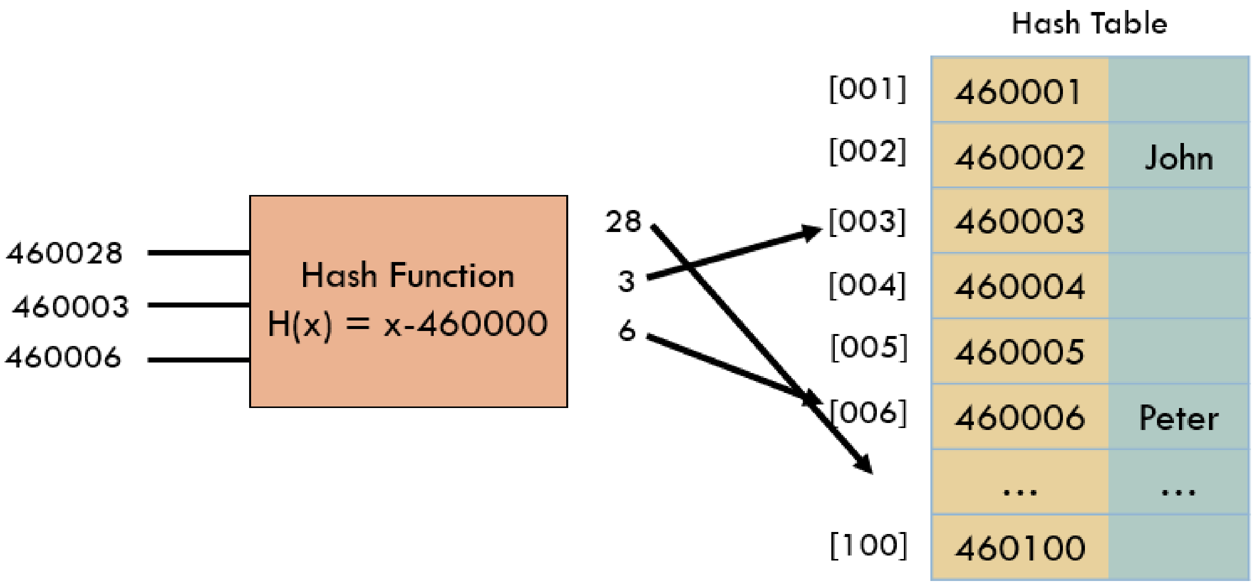
    Hash function: Address = key – constant

    or

    H(x) = x - constant

Hash Function:

✓ Example:



Hash Function:

- Modulo-Division
  - Flexible technique
  - Solution:

Hash function: key MOD table size

or

H(x) = x MOD table size

Remark: table size is usually set as a prime number to reduce the collision

Hash Function:

- Hash Function for Modulo-Division:

  1. Hash function = key MOD table size

     where table size = prime number

     result = address [000-(table size -1)]

  2. Hash function = (key MOD table size)+1

     where table size = prime number

     result = address [001- table size]

Remark: starting address of solution1 is 000 while starting address of solution 2 is 001

Hash Function:

- Algorithm:

```
Function IndexHash(Key:ElementType;TableSize:Integer):integer;
 Begin
     IndexHash: = Key Mod TableSize;
 End;
```

Hash Function:

✓ Example:

Create a hash table to store employee names of ABC company. The approximate size of the table is 100.

- Solution :

table size = 97 → prime number

Hash Function:

- Solution 1:

  H(x) = x MOD 97

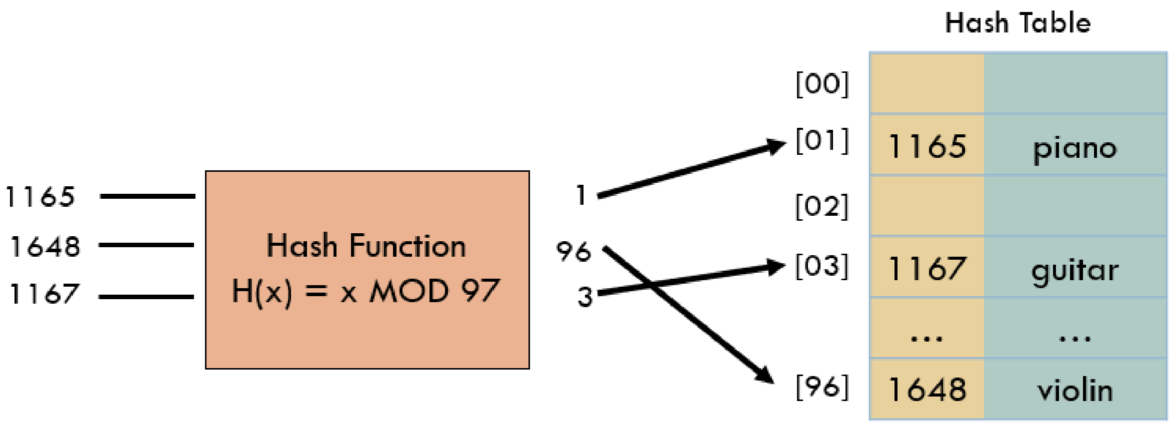
- Solution 2:

  H(x) = (x MOD 97) + 1

Hash Function:

- Solution 1:

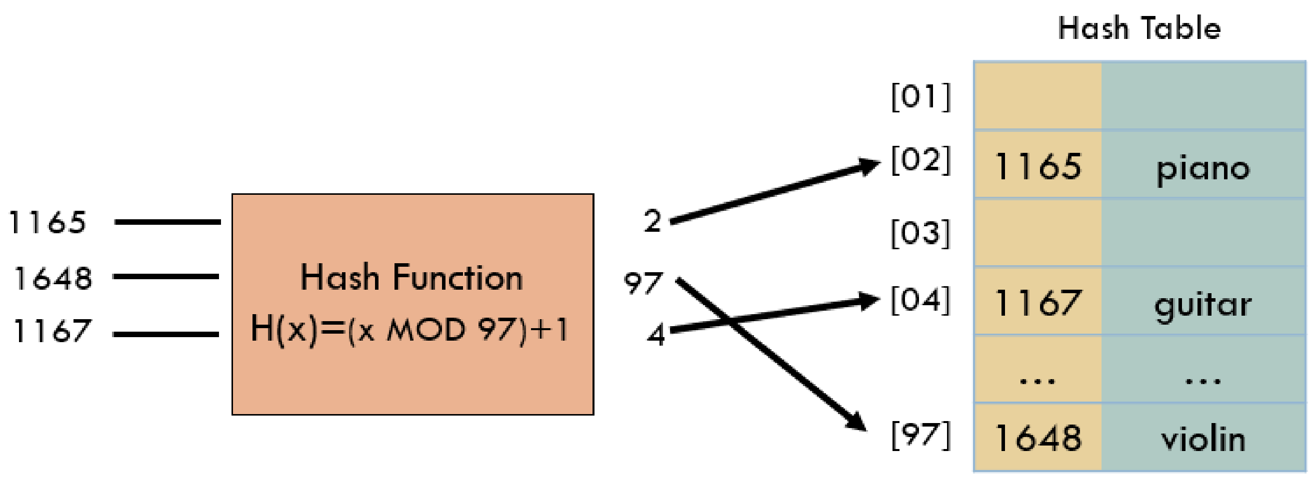
$$H(x) = x \text{ MOD } 97$$



Hash Function:

- Solution 2:

$$H(x) = (x \text{ MOD } 97) + 1$$



Hash Function:

❑ Mid-Square

- Middle of square
- Solution:

  H(x) = return middle digits of x^2

- Example: key = {315, 124, 541, 048}, address 00-99

  ✓ Solution:

Key	Square	Address
315	099225	92
124	015376	53
541	292681	26
048	002304	23

Hash Function:

- Shift Folding

1. Partition the identifier x into 3 parts (left, middle, right)
2. Move left and right to middle part
3. Add the parts together to obtain the hash address
4. Result = address in the hash table

Remark: If the result is bigger than address size, remove the left excess digit

Hash Function:

- Example: x = 123203241 and the address size = 3
  - ✓ Solution:

    Partition x into 123, 203, 241

    Address in the hash table :

$$\left.\begin{matrix}123\\203\\241\end{matrix}\right\} \quad +$$

Result = 567

Hash Function:

❑ Boundary Folding

1. Partition the identifier x into 3 parts (left, middle, right)
2. Fold left and right to middle part
3. Add the parts together to obtain the hash address

Remark: If the result is bigger than address size, remove the left excess digit

Hash Function:

- Example: x = 138967243 and the address size = 3
  - ✓ Solution:

    Partition x into 138, 967, 243

    Address in the hash table :

$$\left.\begin{array}{r} 831 \\ 967 \\ 342 \end{array}\right\} +$$

Result = 2140

Delete 2

Address in the hash table is 140

Collision Resolution:

- There are only two methods (direct hashing and subtraction) that have no collision
- Load factor : percentage of data in the table

$$\lambda = (n/m) \times 100$$

where

n = # of elements stored in the table

m = # of slots in the table

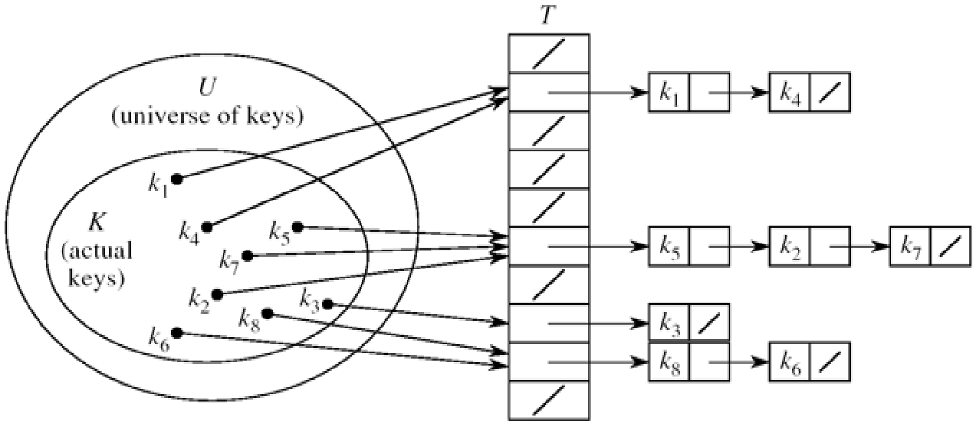
Collision Resolution:

- Separate Chaining
- Open Addressing
  - Linear Probing
  - Quadratic Probing
  - Double Hashing

Collision Resolution:

❑ Separate Chaining

- Put all elements that hash to the same slot into a linked list



- Slot j contains a pointer to the head of the list of all elements that hash to j

Collision Resolution:

- Choosing the size of the table
  - Small enough not to waste space
  - Large enough such that lists remain short
  - Typically 1/5 or 1/10 of the total number of elements

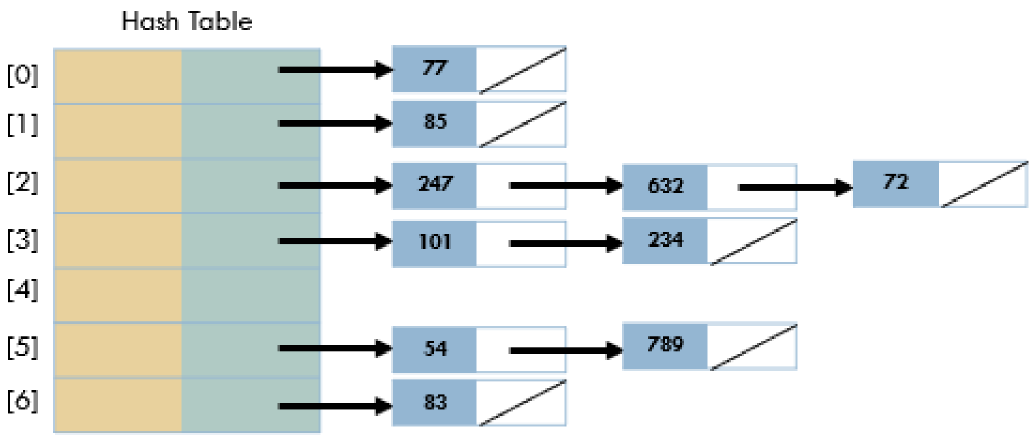
Collision Resolution:

- Example: Create a hash table for the keys (234, 789, 101, 83, 72, 54, 632, 85, 77, 247) using hash function h(x) = x mod 7
  - ✓ Solution:

h(234) = 234 mod 7 = 3
h(789) = 789 mod 7 = 5
h(101) = 101 mod 7 = 3 → collision
h(83) = 83 mod 7 = 6
h(72) = 72 mod 7 = 2
h(54) = 54 mod 7 = 5 → collision
h(632) = 632 mod 7 = 2 → collision
h(85) = 85 mod 7 = 1
h(77) = 77 mod 7 = 0
h(247) = 247 mod 7 = 2 → collision

Collision Resolution:

✓Solution:



Collision Resolution:

- Open Addressing
  - Use two hash functions
  - Solution:
    - ✓the result of first hash function ($H(x)$)
    - ✓the result of second hash function ($h_i(x)$)
  - The size of Hash Table should be a prime number

Collision Resolution:

- Open Addressing
  - Hash Function 1:

    $H(x) = x \text{ MOD table size}$
  - Hash Function 2:

    $(h_i(x)) = (H(x) + f(i)) \text{ MOD table size}$

Remark!!!

- H(x) in the Hash Function 2: is the result of Hash Function 1

- i: the number of collision where i >= 1

Collision Resolution:

- Open Addressing Methods:
  - Linear probing
  - Quadratic probing
  - Double hashing

Collision Resolution:

- Open Addressing Methods: Linear probing
  - ✓ $f(i) = i$
  - ✓ Idea: when there is a collision, check the next available position in the table

Collision Resolution:

- Example: Insert product number into a hash table and the size of the hash table is 13. The keys are 156, 85, 42, 54, 189, 125, 34, 99 and 151
  - Solution:
    - ✓ Hash Function 1:

      $H(x) = x \text{ MOD } 13$

    - ✓ Hash Function 2:

      $(h_i(x)) = (H(x) + f(i)) \text{ MOD } 13$

      $f(i) = i$

Collision Resolution:

- Solution:
  - ✓ H(156) = 156 MOD 13 = 0
  - ✓ H(85) = 85 MOD 13 = 7
  - ✓ H(42) = 42 MOD 13 = 3
  - ✓ H(54) = 54 MOD 13 = 2
  - ✓ H(189) = 189 MOD 13 = 7

$(h_1(189)) = (H(189) + f(1))$ MOD 13 where $f(1) = 1$

$= (7 + 1)$ MOD 13

$= 8$

Collision Resolution:

- ✓ $H(125) = 125 \text{ MOD } 13 \quad = 8$

  $(h_1(125)) = (H(125) + f(1)) \text{ MOD } 13$ where $f(1) = 1$

  $= (8 + 1) \text{ MOD } 13 \quad = 9$

- ✓ $H(34) = 34 \text{ MOD } 13 \quad = 8$

  $(h_1(34)) = (H(34) + f(1)) \text{ MOD } 13$ where $f(1) = 1$

  $= (8 + 1) \text{ MOD } 13 \quad = 9$

  $(h_2(34)) = (H(34) + f(2)) \text{ MOD } 13$ where $f(2) = 2$

  $= (8 + 2) \text{ MOD } 13 \quad = 10$

Collision Resolution:

✓ $H(99) = 99 \text{ MOD } 13 \quad = 8$

$(h_1(99)) = (H(99) + f(1)) \text{ MOD } 13$ where $f(1) = 1$

$= (8 + 1) \text{ MOD } 13 \quad = 9$

$(h_2(99)) = (H(99) + f(2)) \text{ MOD } 13$ where $f(2) = 2$

$= (8 + 2) \text{ MOD } 13 \quad = 10$

$(h_3(99)) = (H(99) + f(3)) \text{ MOD } 13$ where $f(3) = 3$

$= (8 + 3) \text{ MOD } 13 \quad = 11$

Collision Resolution:

✓ $H(151) = 151 \text{ MOD } 13 \quad = 8$

$$(h_1(151)) = (H(151) + f(1)) \text{ MOD } 13 \quad \text{where } f(1) = 1$$
$$= (8 + 1) \text{ MOD } 13 \quad = 9$$
$$(h_2(151)) = (H(151) + f(2)) \text{ MOD } 13 \quad \text{where } f(2) = 2$$
$$= (8 + 2) \text{ MOD } 13 \quad = 10$$
$$(h_3(151)) = (H(151) + f(3)) \text{ MOD } 13 \quad \text{where } f(3) = 3$$
$$= (8 + 3) \text{ MOD } 13 \quad = 11$$
$$(h_4(151)) = (H(151) + f(4)) \text{ MOD } 13 \quad \text{where } f(4) = 4$$
$$= (8 + 4) \text{ MOD } 13 \quad = 12$$

Collision Resolution:

- Solution:

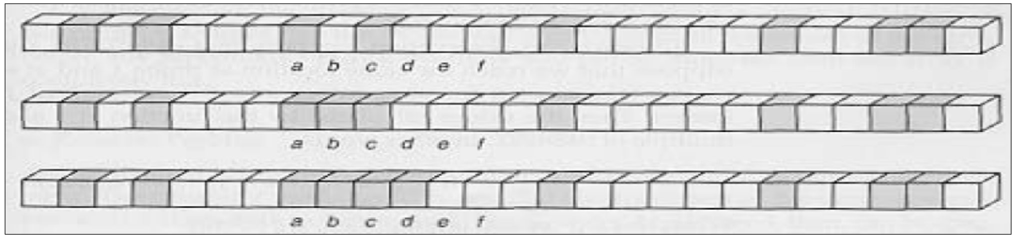
Hash Table

[0]	156	
[1]		
[2]	54	
[3]	42	
[4]		
[5]		
[6]		
[7]	85	
[8]	189	
[9]	125	
[10]	34	
[11]	99	
[12]	151	

Collision Resolution:

- Open Addressing Methods: Linear probing
  - Primary Clustering Problem
    - ✓Some slots become more likely than others
    - ✓Long chunks of occupied slots are created

      → search time increases



Collision Resolution:

- Open Addressing Methods: Linear probing
  - Deleting
    - Example: Delete the key 45 from the given hash table

Hash Table

[0]		
[1]	57	
[2]		
[3]	45	
[4]	38	
[5]	89	
[6]	11	

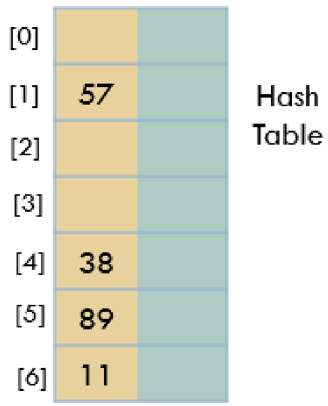
Collision Resolution:

- Solution:
  - ✓ compute address of key 45 in hash table

    H(45) = 45 MOD 7 = 3
  - ✓ check the value in address 3, whether 45 or not ?

    1) If the value is 45, delete it

    2) If the value is not 45, report "could not find 45"



Collision Resolution:

❑ Open Addressing Methods: Linear probing

- Searching
  - ✓Solution:
    - compute address of key x in hash table

      H(x) = x MOD table size

      = Y
    - check address Y, whether x or not ?

      1) If the value is x, report “found”

      2) If the value is not x, check the value of address Y

      - If Y is empty, report “could not find x in hash table”.
      - If Y is not empty, compute address using hash function 2 and do step 2 again (until x is found or Y is empty)

Collision Resolution:

- Open Addressing Methods: Linear probing
  - Searching
    - Example: Search the key 11 from the given hash table

Hash Table

[0]		
[1]	57	
[2]		
[3]	45	
[4]	38	
[5]	89	
[6]	11	

Collision Resolution:

- Solution:
  - ✓compute address of key 11 in hash table

    $H(11) = 11 \text{ MOD } 7 = 4$
  - ✓check the value in address 4, whether 11 or not ? → address 4 = 38

    $(h_1(11)) = (H(11) + f(1)) \text{ MOD } 7$ where $f(1) = 1$

    $= (4 + 1) \text{ MOD } 7 = 5$
  - ✓check the value in address 5, whether 11 or not ? → address 5 = 89

    $(h_1(11)) = (H(11) + f(2)) \text{ MOD } 7$ where $f(2) = 2$

    $= (4 + 2) \text{ MOD } 7 = 6$
  - ✓check the value in address 6, whether 11 or not ? → address 6 = 11

    report “found”

Collision Resolution:

- Open Addressing Methods: Quadratic probing
  - $f(i) = i^2$
  - Idea: when there is a collision, check the next double position in the table

Collision Resolution:

- Example:

Insert employee number into a hash table and the size of hash table is 11. The keys are 85, 55, 42, 96, 132, 140 and 173.

- Solution:
  - ✓ Hash Function 1:

    $H(x) = x \text{ MOD } 11$

  - ✓ Hash Function 2:

    $(h_i(x)) = (H(x) + f(i)) \text{ MOD } 11$

    $f(i) = i^2$

Collision Resolution:

- Solution:

✓H(85) = 85 MOD 11 = 8

✓H(55) = 55 MOD 11 = 0

✓H(42) = 42 MOD 11 = 9

✓H(96) = 96 MOD 11 = 8

$(h_1(96)) = (H(96) + f(1))$ MOD 11 where $f(1) = 1$

$= (8 + 1)$ MOD 11 $= 9$

$(h_2(96)) = (H(96) + f(2))$ MOD 11 where $f(2) = 4$

$= (8 + 4)$ MOD 11 $= 1$

Collision Resolution:

✓ $H(132) = 132 \text{ MOD } 11 \quad = 0$

$(h_1(132)) = (H(132) + f(1)) \text{ MOD } 11$ where $f(1) = 1$

$= (0 + 1) \text{ MOD } 11 \quad = 1$

$(h_2(132)) = (H(132) + f(2)) \text{ MOD } 11$ where $f(2) = 4$

$= (0 + 4) \text{ MOD } 11 \quad = 4$

✓ $H(140) = 140 \text{ MOD } 11 \quad = 8$

$(h_1(140)) = (H(140) + f(1)) \text{ MOD } 11$ where $f(1) = 1$

$= (8 + 1) \text{ MOD } 11 \quad = 9$

$(h_2(140)) = (H(140) + f(2)) \text{ MOD } 11$ where $f(2) = 4$

$= (8 + 4) \text{ MOD } 11 \quad = 1$

$(h_3(140)) = (H(140) + f(3)) \text{ MOD } 11$ where $f(3) = 9$

$= (8 + 9) \text{ MOD } 11 \quad = 6$

Collision Resolution:

✓ $H(173) = 173 \text{ MOD } 11 \quad = 8$

$(h_1(173)) = (H(173) + f(1)) \text{ MOD } 11$ where $f(1) = 1$

$= (8 + 1) \text{ MOD } 11 \quad = 9$

$(h_2(173)) = (H(173) + f(2)) \text{ MOD } 11$ where $f(2) = 4$

$= (8 + 4) \text{ MOD } 11 \quad = 1$

$(h_3(173)) = (H(173) + f(3)) \text{ MOD } 11$ where $f(3) = 9$

$= (8 + 9) \text{ MOD } 11 \quad = 6$

$(h_4(173)) = (H(173) + f(4)) \text{ MOD } 11$ where $f(4) = 16$

$= (8 + 16) \text{ MOD } 11 \quad = 2$

Hash Table

[0]	55	
[1]	96	
[2]	173	
[3]		
[4]	132	
[5]		
[6]	140	
[7]		
[8]	85	
[9]	42	
[10]		

Collision Resolution:

- Open Addressing Methods: Double Hashing
  - $f(i) = i * hash_2(x)$
  - $hash_2(x) = R - (x \text{ MOD } R)$ where R is a prime number which $R <$ table size

Collision Resolution:

- Example:

  Insert employee number into a hash table and the size of hash table is 11. The keys are 73, 152, 211 and 189

Collision Resolution:

- Solution:

✓**R = 7**

✓H(73) = 73 MOD 11 = 9

✓H(152) = 152 MOD 11 = 9

$(h_1(152)) = (H(152) + f(1))$ MOD 11

$hash_2(152) = 7 - (152$ MOD $7)$

$= 7 - 5 = 2$

$f(1) = 1 * 2 = 2$

$(h_1(152)) = (9 + 2)$ MOD 11

$= 0$

Collision Resolution:

- ✓H(211) = 211 MOD 11 = 2
- ✓H(189) = 189 MOD 11 = 2

$(h_1(189)) = (H(189) + f(1))$ MOD 11

$hash_2(189) = 7 - (189$ MOD $7)$

$= 7 – 0 = 7$

$f(1) = 1 * 7 = 7$

$(h_1(189)) = (2 + 7)$ MOD 11 $= 9$

$f(2) = 2 * 7 = 14$

$(h_2(189)) = (2 + 14)$ MOD 11 $= 5$

Rehashing:

- Rehashing when:
  - ✓Case 1: Hash Table has keys more than 50%
  - ✓Case 2: Could not insert key into hash table
  - ✓Case 3: load factor close to 1

Rehashing:

- Example:

  Insert these keys (20, 22, 31, 13 and 30) into the hash table using the Linear Probing. The size of the hash table is 7. The hash function is H(x) = x MOD 7

Rehashing:

- Solution:

  $H(x) = x \text{ MOD } 7$

  $h_i(x) = (H(x) + f(i)) \text{ MOD } 7$ where $f(i) = i$

Rehashing:

- Solution:
  - ✓ $H(20) = 20 \text{ MOD } 7$
    $= 6$
  - ✓ $H(22) = 22 \text{ MOD } 7$
    $= 1$
  - ✓ $H(31) = 31 \text{ MOD } 7$
    $= 3$
  - ✓ $H(13) = 13 \text{ MOD } 7$
    $= 6$ (collision)

    $h_1(13) = (H(13) + f(1)) \text{ MOD } 7$
    $= (6 + 1) \text{ MOD } 7$
    $= 0$
  - ✓ $H(30) = 30 \text{ MOD } 7$
    $= 2$

Rehashing:

- Solution:

Hash Table

[0]	13	
[1]	22	
[2]		
[3]	31	
[4]	38	
[5]		
[6]	20	

Rehashing:

- Solution: size of the hash table = 17

  $H(x) = x \text{ MOD } 17$

  $h_i(x) = (H(x) + f(i)) \text{ MOD } 17$ where $f(i) = i$

Rehashing:

- Solution:
  - ✓ $H(20) = 20 \text{ MOD } 17 = 3$
  - ✓ $H(22) = 22 \text{ MOD } 17 = 1$
  - ✓ $H(31) = 31 \text{ MOD } 17 = 14$
  - ✓ $H(13) = 13 \text{ MOD } 17 = 13$
  - ✓ $H(30) = 30 \text{ MOD } 17 = 13$ (collision)

    $h_1(30) = (H(30) + f(1)) \text{ MOD } 17$

    $= (13 + 1) \text{ MOD } 17 = 14$ (collision)

    $h_2(30) = (H(30) + f(2)) \text{ MOD } 17$

    $= (13 + 2) \text{ MOD } 17 = 15$

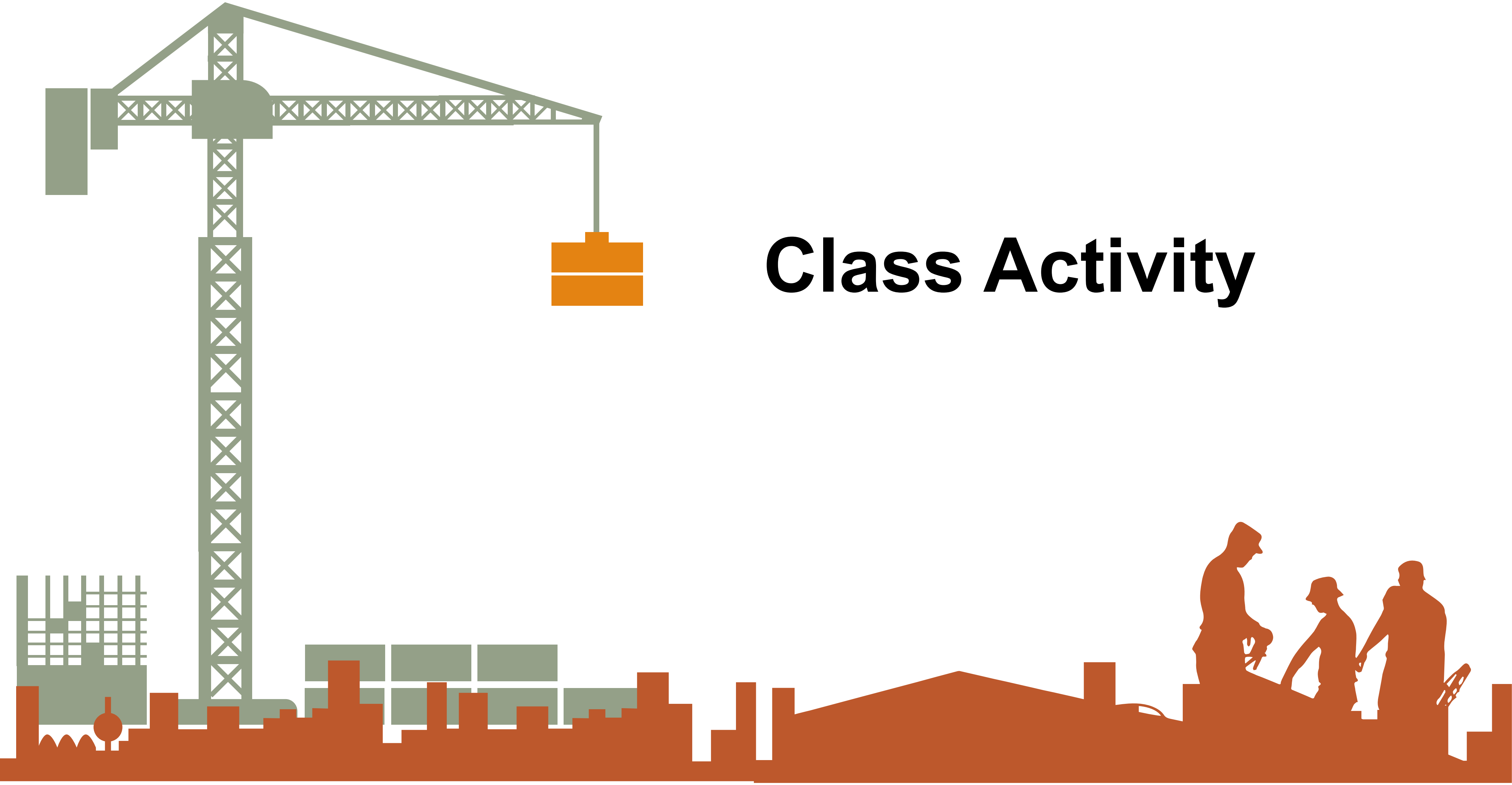
Rehashing:

- Solution:

Hash Table

[0]		
[1]	22	
[2]		
[3]	20	
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		
[10]		
[11]		
[12]		
[13]	13	
[14]	31	
[15]	30	
[16]		

Class Activity



Question & Answer

Formative Assessment