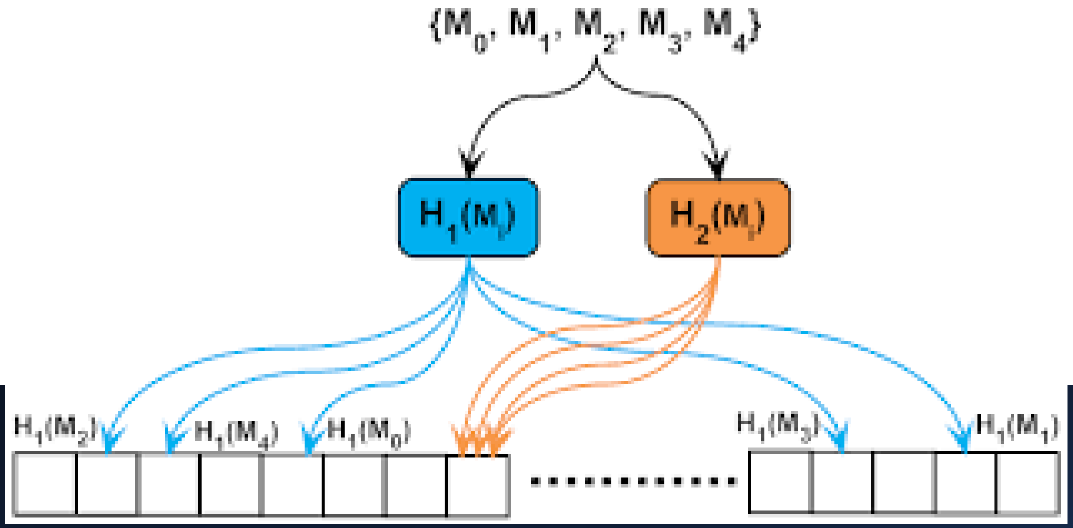
Algorithms

Lecture 9
Hash Algorithms



A. S. M. Sanwar Hosen

Email: sanwar@wsu.ac.kr

Date: 18 May, 2023

Hash Algorithms

- **Hashing:** It generates a fixed size output from a variable size input based on the mathematical formulas called hash function. It determines the location to store information in a data structure.
- **Components of Hashing**

The components are as follows:

- ✓ **Key:** A key is an input in the hash function. It can be a string or an integer.
- ✓ **Hash Function:** It generates an index/location of an input key in an array known as has table.
- ✓ **Hash Table:** It maps input keys to values in an array where each data value has its own unique index.

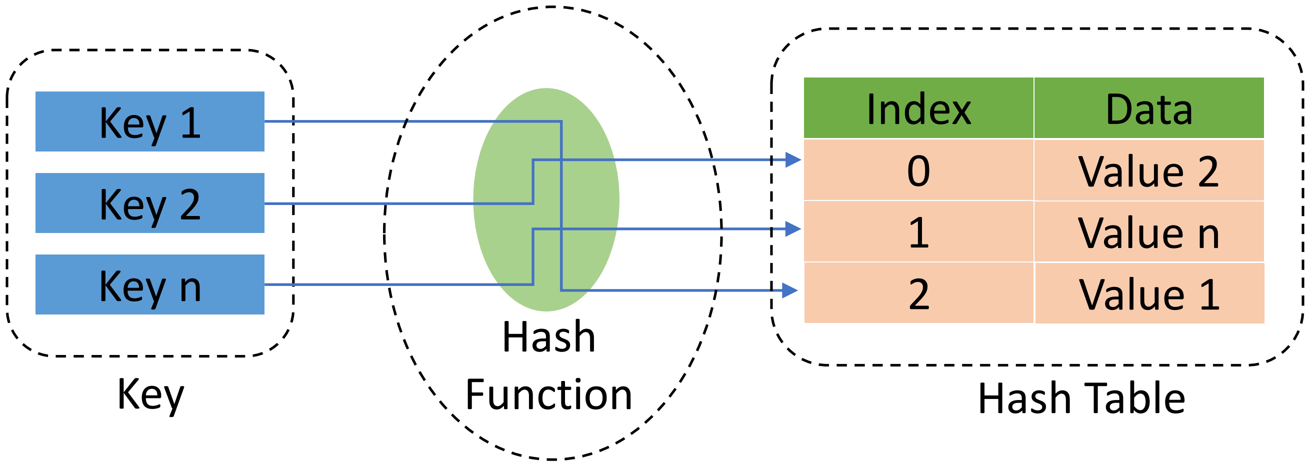


Figure: Hash Components

Hash Algorithms

❑ How It Works?

It works as follows:

- ✓ **Step 1:** A hash function (a methodical formula) is used to calculate the hash value (index) of each key. Suppose, we have a set of strings {‘ab’, ‘bc’, ‘cd’} to store in a table.
- ✓ **Step 2:** Assign an unique value corresponding to each string. Lets assign ‘a’ = 1; ‘b’ = 2; ‘c’ = 3; and ‘d’ = 4 and so on.
- ✓ **Step 3:** Calculate the numerical value by adding all the value of the characters of the: ‘ab’ = 1 + 2 = 3; ‘bc’ = 2 + 3 = 5; ‘cd’ = 3 + 4 = 7
- ✓ **Step 4:** Generate the indices of the strings (keys) using a mathematical formula. For example, we have a table of size 5 and the function uses the formula as $mod(key, table\_size)$ to generate an index. Therefore, the indices are ‘ab’: $mod(3,5) = 3$; ‘bc’: $mod(5,5) = 0$; and ‘cd’: $mod(7,5) = 2$.
- ✓ **Step 5:** Store the values to the generated indices in the table. An example as bellow:

0	1	2	3	4
bc	cd	ab		

Mapping the keys with indices in the hash table

Hash Algorithms

❑ **Types of Hash Functions**

The different hash functions used are:

- ✓ Division Method
- ✓ Mid Square Method
- ✓ Folding Method
- ✓ Multiplication Method

Hash Algorithms

- **Division Method:** In this method, the hash function calculates the reminder of value k dividing by M, and uses the remainder to generate an index. It uses the following formula:

$$H(k) = mod(k, M)$$

Where, H is the hash function, k is the key value, and M is the size of the hash table. M as a prime number is best suited to generate uniformly distributed indices.

For example: Let's assume, $k = 123$, and $M = 23$, therefore, $H(123) = mod(123, 23) = 8$

- **Advantages and Disadvantages of Division Method**

✓ **Advantages:**

1. This method is simple, easy to implement, and quite suitable for any value of M.
2. Time complexity is less, as it requires only a single division operation generating an index.

✓ **Disadvantages:**

1. The performance of this method is poor due to the keys are mapped to consecutive hash values.
2. The value of M needs to be selected carefully.

Hash Algorithms

- **Mid Square Method:** The hash function calculates the square of a given key k and takes the middle digits of the result to use as the index. The number of the middle digits is decided by the size of the table and required digits to align the indices in the table.

  For example: Let's assume, $k = 12$, and $M = 10$, calculates, $H(k^2) = H(12 \times 12) = H(144) = 4$ (the middle of the number is 4 as the maximum 1 digit (0-9) is needed to align the 10 indices in the table).

- **Advantages and Disadvantages of Mid Square Method**

✓ **Advantages:**

1. The performance is good as all the digits in the key produces the middle digits of the squared result.
2. The result is not depend on the top digit or bottom digit of the key value.

✓ **Disadvantages:**

1. The size of the key, as the size of a key is increased, the square of its double is too big.
2. There is collisions in generating indices. Although, there is solution of the problem.

Hash Algorithms

- **Folding Method:** It divides the key-value k into a number of parts where each part has same number of digits except the last part, then it generates an index by adding the numerical values of the parts.

  For example: Let's assume, $k = 13456$, and $k_1 = 13, k_2 = 45, k_3 = 6$. Calculate $H(k) = H(k_1 + k_2 + k_3) = H(13 + 45 + 6) = 64.$

  The number of digits in each part is based on the hash table size. For a hash table of size 100, each part contains 2 digits of the key, unlike the last part.

- **Advantages and Disadvantages of Folding Method**

✓ **Advantages:**

1. Breaks up the key value into precise equal-sized segments for an easy hash value.
2. Independent of distribution in a hash table.

✓ **Disadvantages:**

1. Sometimes inefficient if there are too many collisions.

Hash Algorithms

- **Multiplication Method:** It multiplies the key with a constant fractional number between 0 and 1 and divides the product by 1, and extracts the fractional part of the result. Then it multiplies the fractional part by the size of hash table to generates an index of the key. The formula as follows:

$H(k) = \lceil M \times mod(kc, 1) \rceil$, where c is a constant.

For example: Let's assume, $k = 1235$, $c = 0.125456$, and $M = 100$.

$$H(1235) = H\lceil 100 \times mod(1235 \times 0.125436{,}1) \rceil = H\lceil 91.34599999999 \rceil = 92$$

- **Advantages and Disadvantages of Multiplication Method**

✓ **Advantages:**

1. It can generate infinite constant values between 0 and 1.

✓ **Disadvantages:**

1. The time complexity is increased with increasing in the size of the hash table.

Hash Algorithms

❑ **Hash Algorithms**

The popular hash algorithms used are as follow:

✓ Message Digest (MD)

✓ Secure Hash Function (SHA)

✓ RIPEMD

✓ Whirlpool

❑ **Applications of Hash Functions**

Some of the major applicational of hash function as follows:

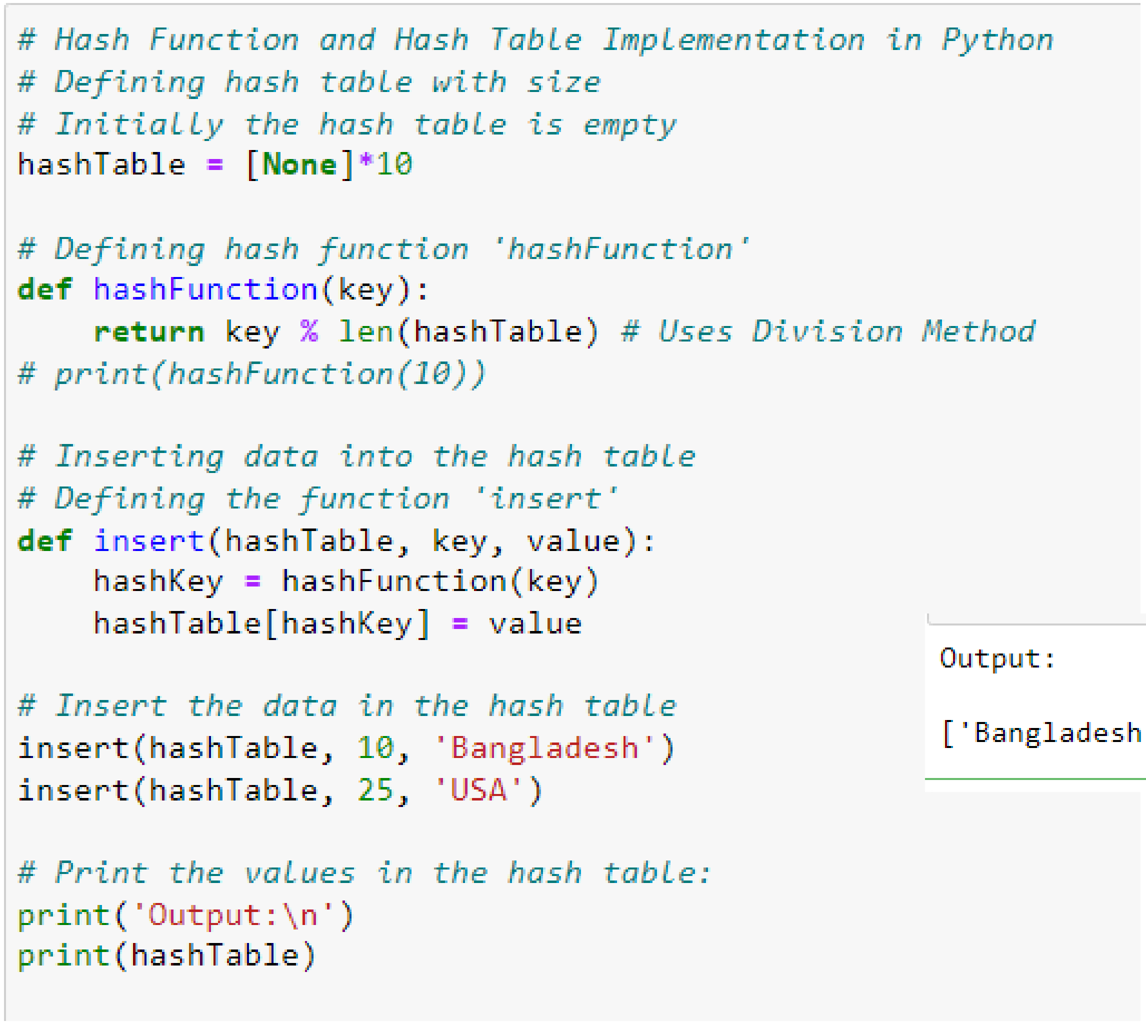
✓ Database Systems

✓ Password Storage

✓ Data Integrity Check

Hash Algorithms Implementation in Python (1)

Division Method Hash Function in Python



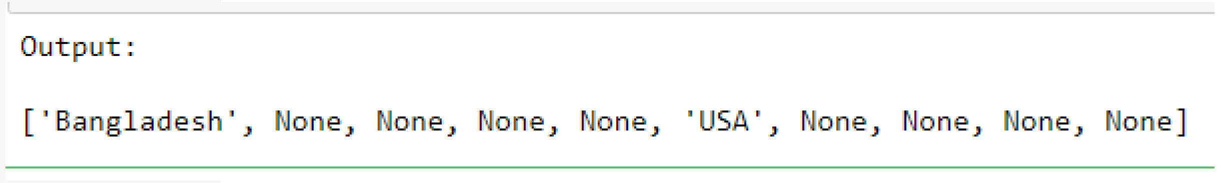
```python
# Hash Function and Hash Table Implementation in Python
# Defining hash table with size
# Initially the hash table is empty
hashTable = [None]*10

# Defining hash function 'hashFunction'
def hashFunction(key):
    return key % len(hashTable) # Uses Division Method
# print(hashFunction(10))

# Inserting data into the hash table
# Defining the function 'insert'
def insert(hashTable, key, value):
    hashKey = hashFunction(key)
    hashTable[hashKey] = value

# Insert the data in the hash table
insert(hashTable, 10, 'Bangladesh')
insert(hashTable, 25, 'USA')

# Print the values in the hash table:
print('Output:\n')
print(hashTable)
```



```
Output:

['Bangladesh', None, None, None, None, 'USA', None, None, None, None]
```

Hash Algorithms Implementation in Python (2)

- **Division Method Hash Function in Python (Collision Occurs, an Example)**

```python
# An example of Collision Occurs
# Collision occurs when multiple items get the same index
# Defining hash table with size
# Initially the hash table is empty
hashTable = [None]*10

# Defining hash function 'hashFunction'
def hashFunction(key):
    return key % len(hashTable) # Uses Division Method
# print(hashFunction(10))

# Inserting data into the hash table
# Defining the function 'insert'
def insert(hashTable, key, value):
    hashKey = hashFunction(key)
    hashTable[hashKey] = value

# Insert the data in the hash table
insert(hashTable, 10, 'Bangladesh')
insert(hashTable, 25, 'USA')
insert(hashTable, 25, 'Korea') #

# Print the values in the hash table:
print('Output:\n')
print('The USA is replaced by Korea, because of the result of hash function for same keys.')
print(hashTable)
```

```
Output:

The USA is replaced by Korea, because of the result of hash function for same keys.
['Bangladesh', None, None, None, None, 'Korea', None, None, None, None]
```

Hash Algorithms Implementation in Python (3)

❑ Division Method Hash Function in Python (Collision Resolution)

```python
# An example of collision resolution
# There are many solutions: 1. Linear Probing and 2. Chaining
# We are using here Chaining Mehtod
# Defining hash table with size:
# Initially the hash table is empty and a nested list
hashTable = [[] for _ in range(10)]
print(hashTable)

# Defining hash function:
def hashFunction(key):
    return key % len(hashTable) # Using Division Method
# print(hashFunction(10))

# Inserting data into the hash table:
# Defining the function 'insert'
def insert(hashTable, key, value):
    hashKey = hashFunction(key)
    hashTable[hashKey].append(value)

# Insert the data in the hash table
insert(hashTable, 10, 'Bangladesh')
insert(hashTable, 25, 'USA')
insert(hashTable, 25, 'Korea') #

# Print the values in the hash table:
print('Output:\n')
print('The USA and Korea are in the same list.')
print(hashTable)
```

```
Output:

The USA and Korea are in the same list.
[['Bangladesh'], [], [], [], [], ['USA', 'Korea'], [], [], [], []]
```