


Nama : Mutiara Novianti Rambe NIM : 064002300029	 Algoritma dan Pemrograman Dasar	Modul 12 Nama Dosen: 1. Abdul Rochman 2. Anung B. Ariwibowo
Hari/Tanggal: Kamis/16 Mei 2024		Nama Aslab: 1. Nathanael W. (064002100020) 2. Adrian Alfajri (064002200009)

MODUL 11 : Hashing

Deskripsi Modul : Memahami dan menerapkan ilmu struktur data dan algoritma untuk menyelesaikan masalah yang disajikan dengan menggunakan program berbasis bahasa Python.

No.	Elemen Kompetensi	Indikator Kinerja	Halaman
1.	Mampu memahami dan mengimplementasikan Hasing pada Python	Membuat dan memahami sebuah program yang menerapkan struktur data Hashing.	

TEORI SINGKAT

Hash table merupakan salah satu struktur data yang menggunakan fungsi khusus yang dikenal sebagai fungsi hash yang dapat memetakan nilai yang diberikan dengan menggunakan *key* untuk mengakses elemen lebih cepat. Hash table menyimpan beberapa informasi, di mana informasi tersebut memiliki dua komponen utama, yaitu *key* dan *value/data*. Hash table dapat diimplementasikan dengan bantuan array asosiatif.

DAFTAR PERTANYAAN

1. Apa yang dimaksud dengan collision?
2. Sebutkan cara apa saja yang dapat diterapkan untuk menangani collision!
3. Apakah kelebihan dari hash table?

JAWABAN

1. Collision atau tabrakan dalam hash table adalah situasi di mana dua atau lebih kunci yang berbeda dipetakan ke slot yang sama dalam tabel hash. Hal ini terjadi karena fungsi hash tidak selalu menghasilkan nilai yang unik untuk setiap kunci.

2. – chaining
 - Open addressing
 - Rehashing
3. - Pencarian elemen dalam hash table umumnya sangat cepat, dengan rata-rata waktu akses $O(1)$.
 - Hash table dapat menyimpan data dengan cara yang efisien, terutama untuk data dengan kunci yang sering digunakan.
 - Implementasi hash table relatif sederhana dibandingkan dengan struktur data lain seperti pohon.

LAB SETUP

Hal yang harus disiapkan dan dilakukan oleh praktikan untuk menjalankan praktikum modul ini, antara lain:

1. Menyiapkan IDE untuk membangun program python (Spyder, Sublime, VSCode, dll);
2. Python sudah terinstal dan dapat berjalan dengan baik di laptop masing-masing;
3. Menyimpan semua dokumentasi hasil praktikum pada laporan yang sudah disediakan.

ELEMEN KOMPETENSI I

Deskripsi : Mampu membuat program tentang hash table sesuai perintah yang ada.

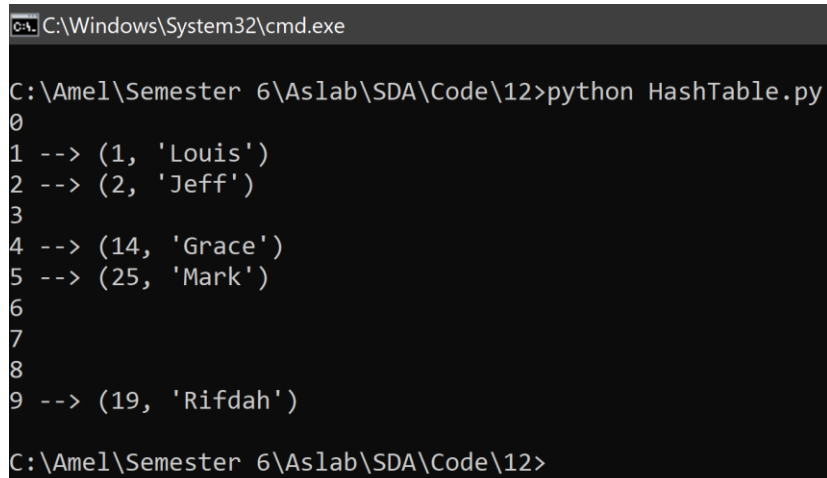
Kompetensi Dasar : Membuat program yang mengimplementasikan hash table

LATIHAN 1

1. Buatlah sebuah program yang mengimplementasikan insert untuk menginput data pada hash table dan display untuk menampilkan isi hash table.
2. Setiap program wajib menampilkan nama dan nim di bagian atas program.
3. Hash table wajib diisi dengan 5 data, di mana data pertama merupakan nama praktikan dan keynya adalah dua digit terakhir nim.

Misal: Jeff memiliki nim 064001900002, maka keynya adalah 2 dan datanya adalah Jeff.

Output:



```
C:\Windows\System32\cmd.exe

C:\Amel\Semester 6\Aslab\SDA\Code\12>python HashTable.py
0
1 --> (1, 'Louis')
2 --> (2, 'Jeff')
3
4 --> (14, 'Grace')
5 --> (25, 'Mark')
6
7
8
9 --> (19, 'Rifdah')

C:\Amel\Semester 6\Aslab\SDA\Code\12>
```

LATIHAN 2

1. Tambahkan fungsi search yang dapat digunakan untuk mencari data di dalam hash table dengan menginput key dari data yang ingin dicari.

Output:

```
C:\Windows\System32\cmd.exe
C:\Amel\Semester 6\Aslab\SDA\Code\12>python HashTable.py
0
1 --> (1, 'Louis')
2 --> (2, 'Jeff')
3
4 --> (14, 'Grace')
5 --> (25, 'Mark')
6
7
8
9 --> (19, 'Rifdah')
Data yang dicari dengan key adalah Louis
C:\Amel\Semester 6\Aslab\SDA\Code\12>
```

LATIHAN 3

1. Tambahkan fungsi delete yang dapat digunakan untuk menghapus data di dalam hash table dengan menginput key dari data yang ingin dihapus

```

C:\Windows\System32\cmd.exe
C:\Amel\Semester 6\Aslab\SDA\Code\12>python HashTable.py
0
1 --> (1, 'Louis')
2 --> (2, 'Jeff')
3
4 --> (14, 'Grace')
5 --> (25, 'Mark')
6
7
8
9 --> (19, 'Rifdah')

Key 2 deleted
Hash table setelah menghapus data menjadi:
0
1 --> (1, 'Louis')
2
3
4 --> (14, 'Grace')
5 --> (25, 'Mark')
6
7
8
9 --> (19, 'Rifdah')

C:\Amel\Semester 6\Aslab\SDA\Code\12>

```

Source Code

LATIHAN.1:

```

class HashTable:
    def __init__(self, size):
        self.size = size
        self.table = [None] * size

    def hash_function(self, key):
        return key % self.size

    def insert(self, key, value):
        index = self.hash_function(key)
        original_index = index
        while self.table[index] is not None:
            index = (index + 1) % self.size
            if index == original_index:
                print("Hash table is full")
                return
        self.table[index] = (key, value)

    def display(self):

```

```
for index, item in enumerate(self.table):
    if item is not None:
        print(f"{index} --> {item}")
    else:
        print(f"{index}")

# Menampilkan nama dan NIM praktikan
print("Nama: Mutiara")
print("NIM: 064002300029")

# Ukuran hash table
size = 10
hash_table = HashTable(size)

# Data untuk dimasukkan ke dalam hash table
data = [
    (29, "Mutiara"), # Key berdasarkan dua digit terakhir NIM
    (1, "Louis"),
    (2, "Jeff"),
    (14, "Grace"),
    (25, "Mark")
]

# Insert data ke dalam hash table
for key, value in data:
    hash_table.insert(key, value)

# Display isi hash table
hash_table.display()
```

LATIHAN.2:

```
class HashTable:
    def __init__(self, size):
        self.size = size
        self.table = [None] * size

    def hash_function(self, key):
        return key % self.size

    def insert(self, key, value):
        index = self.hash_function(key)
        original_index = index
        while self.table[index] is not None:
            index = (index + 1) % self.size
            if index == original_index:
```

```
        print("Hash table is full")
        return
    self.table[index] = (key, value)

def search(self, key):
    index = self.hash_function(key)
    original_index = index
    while self.table[index] is not None:
        if self.table[index][0] == key:
            return self.table[index]
        index = (index + 1) % self.size
    if index == original_index:
        break
    return None

def display(self):
    for index, item in enumerate(self.table):
        if item is not None:
            print(f"{index} --> {item}")
        else:
            print(f"{index}")

# Menampilkan nama dan NIM praktikan
print("Nama: Mutiara")
print("NIM: 064002300029")

# Ukuran hash table
size = 10
hash_table = HashTable(size)

# Data untuk dimasukkan ke dalam hash table
data = [
    (29, "Mutiara"), # Key berdasarkan dua digit terakhir NIM
    (1, "Louis"),
    (2, "Jeff"),
    (14, "Grace"),
    (25, "Mark")
]

# Insert data ke dalam hash table
for key, value in data:
    hash_table.insert(key, value)

# Display isi hash table
hash_table.display()
```

```
# Menerima input kunci dari pengguna untuk pencarian
search_key = int(input("\nMasukkan key yang ingin dicari: "))
result = hash_table.search(search_key)
if result:
    print(f>Data dengan key {search_key} ditemukan: {result}")
else:
    print(f>Data dengan key {search_key} tidak ditemukan")
```

LATIHAN.3:

```
class HashTable:
    def __init__(self, size):
        self.size = size
        self.table = [None] * size

    def hash_function(self, key):
        return key % self.size

    def insert(self, key, value):
        index = self.hash_function(key)
        original_index = index
        while self.table[index] is not None:
            index = (index + 1) % self.size
            if index == original_index:
                print("Hash table is full")
                return
        self.table[index] = (key, value)

    def search(self, key):
        index = self.hash_function(key)
        original_index = index
        while self.table[index] is not None:
            if self.table[index][0] == key:
                return self.table[index]
            index = (index + 1) % self.size
            if index == original_index:
                break
        return None

    def delete(self, key):
        index = self.hash_function(key)
        original_index = index
        while self.table[index] is not None:
            if self.table[index][0] == key:
                self.table[index] = None
```

```

        print(f"Data dengan key {key} telah dihapus")
        return
    index = (index + 1) % self.size
    if index == original_index:
        break
    print(f"Data dengan key {key} tidak ditemukan")

def display(self):
    for index, item in enumerate(self.table):
        if item is not None:
            print(f"{index} --> {item}")
        else:
            print(f"{index}")

# Menampilkan nama dan NIM praktikan
print("Nama: Mutiara")
print("NIM: 064002300029")

# Ukuran hash table
size = 10
hash_table = HashTable(size)

# Data untuk dimasukkan ke dalam hash table
data = [
    (29, "Mutiara"), # Key berdasarkan dua digit terakhir NIM
    (1, "Louis"),
    (2, "Jeff"),
    (14, "Grace"),
    (25, "Mark")
]

# Insert data ke dalam hash table
for key, value in data:
    hash_table.insert(key, value)

# Display isi hash table
hash_table.display()

# Menerima input kunci dari pengguna untuk penghapusan
delete_key = int(input("\nMasukkan key yang ingin dihapus: "))
hash_table.delete(delete_key)

# Display isi hash table setelah penghapusan
print("\nIsi hash table setelah penghapusan:")
hash_table.display()

```


Screenshot

LATIHAN.1:

```
Nama: Mutiara  
NIM: 064002300029  
0  
1 --> (1, 'Louis')  
2 --> (2, 'Jeff')  
3  
4 --> (14, 'Grace')  
5 --> (25, 'Mark')  
6  
7  
8  
9 --> (29, 'Mutiara')
```

LATIHAN.2:

```
1 --> (1, 'Louis')
2 --> (2, 'Jeff')
3
4 --> (14, 'Grace')
5 --> (25, 'Mark')
6
7
8
9 --> (29, 'Mutiarara')
```

Masukkan key yang ingin dicari: 25

Data dengan key 25 ditemukan: (25, 'Mark')

LATIHAN.3:

```
Nama: Mutiara
NIM: 064002300029
0
1 --> (1, 'Louis')
2 --> (2, 'Jeff')
3
4 --> (14, 'Grace')
5 --> (25, 'Mark')
6
7
8
9 --> (29, 'Mutiarara')
```

```
Masukkan key yang ingin dihapus: 14
Data dengan key 14 telah dihapus
```

```
Isi hash table setelah penghapusan:
```

```
0
1 --> (1, 'Louis')
2 --> (2, 'Jeff')
3
4
5 --> (25, 'Mark')
6
7
8
9 --> (29, 'Mutiarara')
```

KESIMPULAN

Saya dapat mengetahui apa itu hash table yaitu merupakan salah satu struktur data yang menggunakan fungsi khusus yang dikenal sebagai fungsi hash yang dapat memetakan nilai yang diberikan dengan menggunakan *key* untuk mengakses elemen lebih cepat.

Dan belajar juga membuat program hash table.

CEKLIST

1. Memahami dan mengimplementasikan hash table pada Python (✓)

REFERENSI

<https://www.programiz.com/dsa/hash-table>

<https://www.javatpoint.com/hash-table>

<https://www.guru99.com/hash-table-data-structure.html>

<https://www.geeksforgeeks.org/implementation-of-hashing-with-chaining-in-python/>