LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL V
HASH TABLE

Disusun Oleh :

NAMA : RIFKY DWI MAHARDIKA

NIM : 2311102043

Dosen

Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

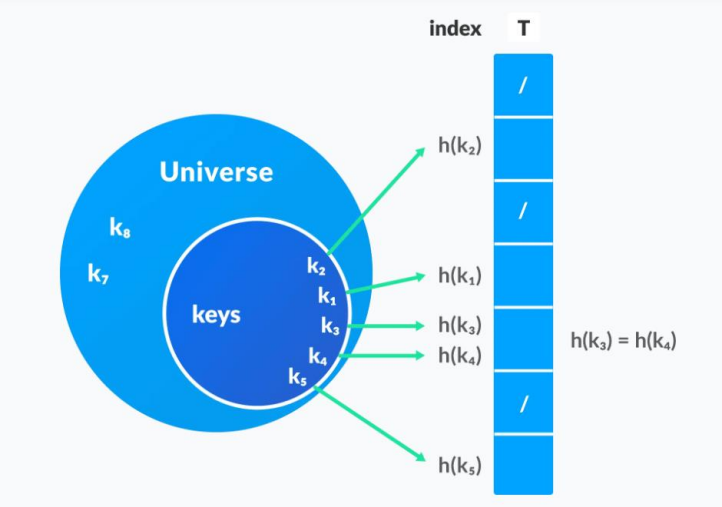
A. Dasar Teori

a. HASH TABLE

Hash table merupakan struktur data yang secara asosiatif menyimpan data. Dalam hal ini, data disimpan dalam format array, di mana setiap nilai data memiliki nilai indeks uniknya sendiri. Akses data akan menjadi sangat cepat jika Anda mengetahui indeks dari data yang diinginkan.

b. FUNGSI HASH TABLE PADA DATA

Fungsi utamanya pada data adalah mempercepat proses akses data. Hal ini berkaitan dengan peningkatan data dalam jumlah besar yang diproses oleh jaringan data global dan lokal. Hash table adalah solusi untuk membuat proses akses data lebih cepat dan memastikan bahwa data dapat dipertukarkan dengan aman.



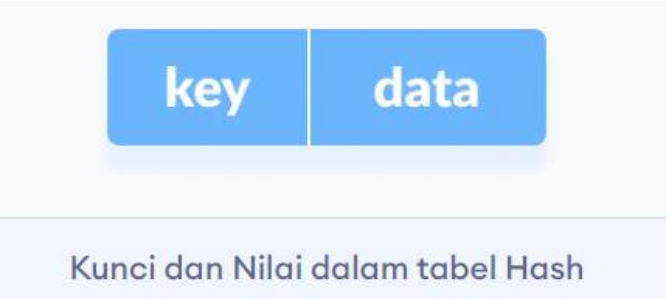
c. TEKNIK-TEKNIK HASH TABLE

Dalam praktiknya, setidaknya ada dua teknik yang umum digunakan saat data scientist melakukan hash table. Berikut ini penjelasannya:

1. HASHING

Hashing merupakan sebuah proses mengganti kunci yang diberikan atau string karakter menjadi nilai lain. Penggunaan hashing paling populer adalah pada hash table. Hash table menyimpan pasangan kunci dan nilai dalam daftar yang

dapat diakses melalui indeksnya. Karena pasangan kunci dan nilai tidak terbatas, maka fungsinya akan memetakan kunci ke ukuran tabel dan kemudian nilainya menjadi indeks untuk elemen tertentu.



Kunci dan Nilai dalam tabel Hash

- Kunci - bilangan bulat unik yang digunakan untuk mengindeks nilai
- Nilai - data yang berhubungan dengan kunci.

2. LINEAR PROBING

Linear probing merupakan skema dalam pemrograman komputer untuk menyelesaikan collision pada hash table. Dalam skema ini, setiap sel dari hash table menyimpan satu pasangan kunci-nilai. Saat fungsi hash menyebabkan collision dengan memetakan kunci baru ke sel hash table yang sudah ditempati oleh kunci lain, maka linear probing akan mencari tabel untuk lokasi bebas terdekat dan menyisipkan kunci baru.

Pencarian dilakukan dengan cara yang sama, yaitu dengan mencari tabel secara berurutan, mulai dari posisi yang diberikan oleh fungsi hash, hingga menemukan sel dengan kunci yang cocok atau sel kosong. Hash table adalah struktur data non trivial yang paling umum digunakan. Linear probing dapat memberikan kinerja tinggi karena lokasi referensi yang baik, namun lebih sensitif terhadap kualitas fungsi hash daripada beberapa skema resolusi collision lainnya.

B. Guided

Guided 1

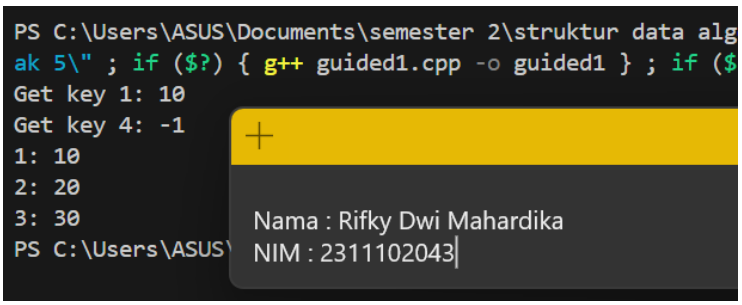
```cpp
#include <iostream>
using namespace std;
const int MAX_SIZE = 10;
// Fungsi hash sederhana
int hash_func(int key)
{
    return key % MAX_SIZE;
}
// Struktur data untuk setiap node
struct Node
{
    int key;
    int value;
    Node *next;
    Node(int key, int value) : key(key), value(value),
                               next(nullptr) {}
};
// Class hash table
class HashTable
{
private:
    Node **table;

public:
    HashTable()
    {
        table = new Node *[MAX_SIZE]();
    }
    ~HashTable()
    {
        for (int i = 0; i < MAX_SIZE; i++)
        {
            Node *current = table[i];
            while (current != nullptr)
            {
                Node *temp = current;
                current = current->next;
                delete temp;
            }
        }
        delete[] table;
    }
```

```
    // Insertion
    void insert(int key, int value)
    {
        int index = hash_func(key);
        Node *current = table[index];
        while (current != nullptr)
        {
            if (current->key == key)
            {
                current->value = value;
                return;
            }
            current = current->next;
        }
        Node *node = new Node(key, value);
        node->next = table[index];
        table[index] = node;
    }
    // Searching
    int get(int key)
    {
        int index = hash_func(key);
        Node *current = table[index];
        while (current != nullptr)
        {
            if (current->key == key)
            {
                return current->value;
            }
            current = current->next;
        }
        return -1;
    }

    // Deletion
    void remove(int key)
    {
        int index = hash_func(key);
        Node *current = table[index];
        Node *prev = nullptr;
        while (current != nullptr)
        {
            if (current->key == key)
            {
                if (prev == nullptr)
```

```
                {
                    table[index] = current->next;
                }
                else
                {
                    prev->next = current->next;
                }
                delete current;
                return;
            }
            prev = current;
            current = current->next;
        }
    }
    // Traversal
    void traverse()
    {
        for (int i = 0; i < MAX_SIZE; i++)
        {
            Node *current = table[i];
            while (current != nullptr)
            {
                cout << current->key << ": " << current->value
                     << endl;
                current = current->next;
            }
        }
    }
};

int main()
{
    HashTable ht;
    // Insertion
    ht.insert(1, 10);
    ht.insert(2, 20);
    ht.insert(3, 30);
    // Searching
    cout << "Get key 1: " << ht.get(1) << endl;
    cout << "Get key 4: " << ht.get(4) << endl;

    // Deletion
    ht.remove(4);

    // Traversal
```

```
    ht.traverse();

    return 0;
}
```

Screenshots Output



Deskripsi

Program tersebut adalah implementasi dari Tabel Hash.Fungsi Hash, akan menghitung indeks dalam tabel hash berdasarkan kunci (key) yang diberikan. Menggunakan operasi modulo untuk memastikan indeks berada dalam rentang ukuran tabel (MAX_SIZE). Pada struktur data node, setiap node memiliki dua atribut: `key` dan `value`. Di gunakan untuk menyimpan pasangan kunci-nilai dalam tabel hash. Class HashTable array dari pointer ke node. Konstruktor HashTable menginisialisasi tabel dengan ukuran `MAX_SIZE`. Destruktor HashTable membersihkan memori dengan menghapus node. Pada operasi pada Tabel Hash, antara lain :

a. Insertion, Menambahkan pasangan kunci-nilai ke tabel hash.
b. Searching, Mencari nilai berdasarkan kunci.
c. Deletion, Menghapus pasangan kunci-nilai berdasarkan kunci.
d. Traversal, Menampilkan seluruh isi tabel hash.

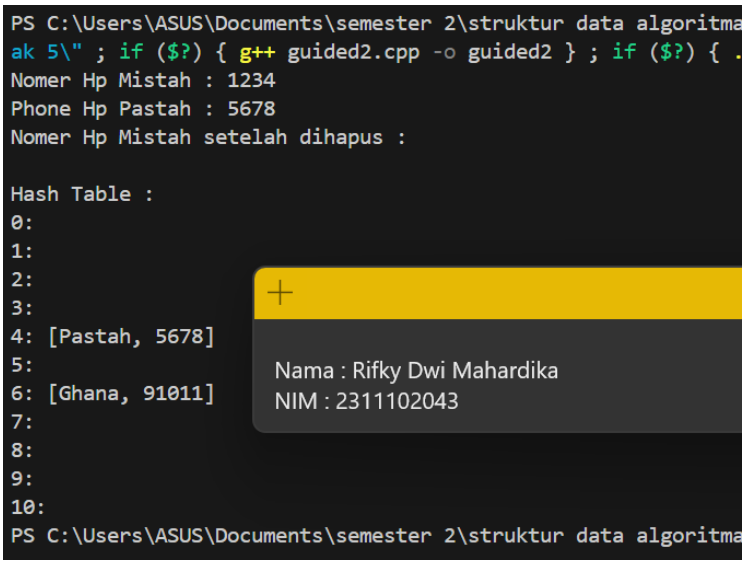
Guided 2

```cpp
#include <iostream>
#include <string>
#include <vector>
using namespace std;
const int TABLE_SIZE = 11;
string name;
string phone_number;
class HashNode
{
public:
    string name;
    string phone_number;
    HashNode(string name, string phone_number)
    {
        this->name = name;
        this->phone_number = phone_number;
    }
};
class HashMap
{
private:
    vector<HashNode *> table[TABLE_SIZE];

public:
    int hashFunc(string key)
    {
        int hash_val = 0;
        for (char c : key)
        {
            hash_val += c;
        }
        return hash_val % TABLE_SIZE;
    }
    void insert(string name, string phone_number)
    {
        int hash_val = hashFunc(name);
        for (auto node : table[hash_val])
        {
            if (node->name == name)
            {
                node->phone_number = phone_number;
                return;
            }
```

```
        }
        table[hash_val].push_back(new HashNode(name,
                                               phone_number));
    }
    void remove(string name)
    {
        int hash_val = hashFunc(name);

        for (auto it = table[hash_val].begin(); it !=
                                               table[hash_val].end();
             it++)
        {
            if ((*it)->name == name)
            {
                table[hash_val].erase(it);
                return;
            }
        }
    }
    string searchByName(string name)
    {
        int hash_val = hashFunc(name);
        for (auto node : table[hash_val])
        {
            if (node->name == name)
            {
                return node->phone_number;
            }
        }
        return "";
    }
    void print()
    {
        for (int i = 0; i < TABLE_SIZE; i++)
        {
            cout << i << ": ";
            for (auto pair : table[i])
            {
                if (pair != nullptr)
                {
                    cout << "[" << pair->name << ", " << pair-
>phone_number << "]";
                }
            }
            cout << endl;
```

```
        }
    }
};
int main()
{
    HashMap employee_map;
    employee_map.insert("Mistah", "1234");
    employee_map.insert("Pastah", "5678");
    employee_map.insert("Ghana", "91011");
    cout << "Nomer Hp Mistah : "
         << employee_map.searchByName("Mistah") << endl;
    cout << "Phone Hp Pastah : "
         << employee_map.searchByName("Pastah") << endl;
    employee_map.remove("Mistah");
    cout << "Nomer Hp Mistah setelah dihapus : "
         << employee_map.searchByName("Mistah") << endl
         << endl;
    cout << "Hash Table : " << endl;
    employee_map.print();
    return 0;
}
```

Screenshots Output



Deskripsi

Program tersebut implementasi tabel hash yang menggunakan dengan chaining. Struktur HashNode, Setiap node memiliki dua atribut: "name" dan "phone_number". Node ini digunakan untuk menyimpan pasangan nama-nomor telepon dalam tabel hash. Class

HashMap, Menggunakan array dari vektor pointer ke node vector HashNode table[TABLE_SIZE]`) sebagai tabel hash. HashFunc menghitung nilai hash berdasarkan nama. Insert menambahkan pasangan nama-nomor telepon ke tabel hash. Remove, menghapus pasangan nama-nomor telepon berdasarkan nama. SearchByName, mencari nomor telepon berdasarkan nama. Print, menampilkan seluruh isi tabel hash.

C. Unguided

```cpp
#include <iostream>
#include <string>
#include <vector>
using namespace std;
const int TABLE_SIZE = 11;
string nim;
int nilai;
class HashNode
{
public:
    string nim;
    int nilai;
    HashNode(string nim, int nilai)
    {
        this->nim = nim;
        this->nilai = nilai;
    }
};
class HashMap
{
private:
    vector<HashNode *> table[TABLE_SIZE];

public:
    int hashFunc(string key)
    {
        int hash_val = 0;
        for (char c : key)
        {
            hash_val += c;
        }
        return hash_val % TABLE_SIZE;
    }
    void insert(string nim, int nilai)
    {
        int hash_val = hashFunc(nim);
        for (auto node : table[hash_val])
        {
            if (node->nim == nim)
            {
                node->nilai = nilai;
                return;
            }
```

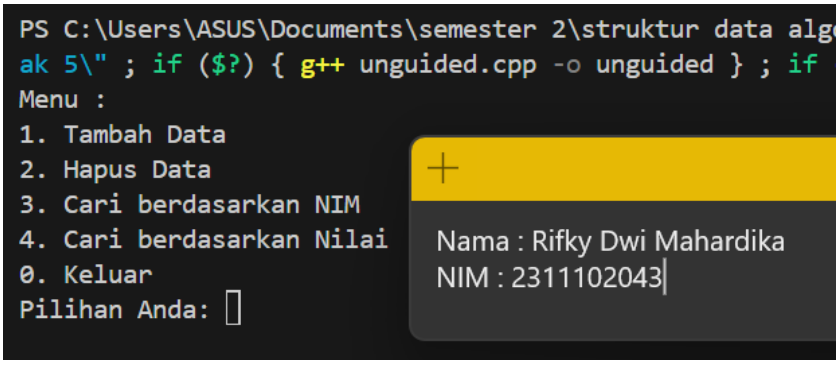
```
        }
        table[hash_val].push_back(new HashNode(nim,nilai));
    }
    void remove(string nim)
    {
        int hash_val = hashFunc(nim);
        for (auto it = table[hash_val].begin(); it !=
table[hash_val].end();
             it++)
        {
            if ((*it)->nim == nim)
            {
                table[hash_val].erase(it);
                cout << "==================================\n";
                cout << "Data Mahasiswa telah dihapus!\n";
                cout << "==================================\n";
                return;
            }
            else
            {
                cout << "Tidak Ditemukan" << endl;
            }
        }
    }
    int searchByNim(string nim)
    {
        int hash_val = hashFunc(nim);
        for (auto node : table[hash_val])
        {
            if (node->nim == nim)
            {
                // return node->nilai;
                cout << "\nMahasiswa dengan NIM " << node->nim
                     << " memiliki nilai " << node->nilai << endl;
            }
        }
        return 0;
    }
    int searchByNilai(int minNilai, int maxNilai)
    {
        for (const auto &bucket : table)
        {
            for (auto node : bucket)
            {
                if (node->nilai >= minNilai && node->nilai <= maxNilai)
```

```
                {
cout << "[ NIM : " << node->nim << ", NILAI : " << node->nilai << " ]"
<< endl;
                }
            }
        }
        return 0;
    }
    void print()
    {
        for (int i = 0; i < TABLE_SIZE; i++)
        {
            cout << i << ": ";
            for (auto pair : table[i])
            {
                if (pair != nullptr)
                {
cout << "[ NIM : " << pair->nim << ", NILAI : " << pair->nilai << " ]";
                }
            }
            cout << endl;
        }
    }
};
int main()
{
    HashMap mahasiswa_map;
    bool menu = true;
    int choice;
    do
    {
        cout << "Menu :\n";
        cout << "1. Tambah Data" << endl;
        cout << "2. Hapus Data" << endl;
        cout << "3. Cari berdasarkan NIM" << endl;
        cout << "4. Cari berdasarkan Nilai" << endl;
        cout << "0. Keluar" << endl;
        cout << "Pilihan Anda: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            cout << "Masukkan NIM: ";
            cin >> nim;
            cout << "Masukkan Nilai: ";
```

```
            cin >> nilai;
            mahasiswa_map.insert(nim, nilai);
            break;
        case 2:
            cout << "Masukkan NIM: ";
            cin >> nim;
            mahasiswa_map.remove(nim);
            break;
        case 3:
            cout << "Masukkan NIM: ";
            cin >> nim;
            mahasiswa_map.searchByNim(nim);
            break;
        case 4:
            int maxNilai, minNilai;
            cout << "Masukkan Nilai Tertinggi: ";
            cin >> maxNilai;
            cout << "Masukkan Nilai Terendah: ";
            cin >> minNilai;
            mahasiswa_map.searchByNilai(minNilai, maxNilai);
            break;
        case 0:
            menu = false;
            break;
        default:
            break;
        }
    } while (menu == true);
    return 0;
}
```

Screenshots Output

- MENU



- Menambahkan Data

```
PS C:\Users\ASUS\Documents\semester 2\struktur data algorit
ak 5\" ; if ($?) { g++ unguided.cpp -o unguided } ; if ($?)
Menu :
1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 1
Masukkan NIM: 2311102043
Masukkan Nilai: 90
```

Nama : Rifky Dwi Mahardika
NIM : 2311102043

- Menghapus Data

```
Menu :
1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 2
Masukkan NIM:  2311102043
================================
Data Mahasiswa telah dihapus!
================================
```

Nama : Rifky Dwi Mahardika
NIM : 2311102043

- Mencari Data Berdasarkan NIM

```
Menu :
1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 3
Masukkan NIM: 2311102043

Mahasiswa dengan NIM 2311102043 memiliki nilai 90
```

Nama : Rifky Dwi Mahardika
NIM : 2311102043

- Mencari data berdasarkan rentang nilai (80 – 90)

```
Menu :
1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 4
Masukkan Nilai Tertinggi: 90
Masukkan Nilai Terendah: 80
[ NIM : 2311102043, NILAI : 90 ]
[ NIM : 2311103624, NILAI : 80 ]
```

Nama : Rifky Dwi Mahardika
NIM : 2311102043

Deskripsi

Program diatas merupakan program sederhana untuk menyimpan informasi nilai siswa berdasarkan NIM. Program ini mengimplementasikan struktur data hashtable untuk menyimpan dan mengelola data mahasiswa. Program mendefinisikan class HashNode. Setiap node mempunyai dua atribut yaitu nama untuk menyimpan NIM dan nilai untuk menyimpan nilai. Lalu ada class HashMap, yaitu kelas dasar yang mewakili tabel hash. Array vektor yang digunakan untuk menyimpan node. Fungsi HashFunc digunakan untuk menghitung hash NIM. Lalu ada fungsi penambahan untuk menambahkan data mahasiswa ke hash table. Jika terjadi collision, data disimpan dalam vektor dengan indeks yang sama di array. Fungsi remove digunakan untuk menghapus data mahasiswa berdasarkan NIM yang disediakan. Fungsi search By Nim digunakan untuk mencari dan menampilkan data mahasiswa dari NIM. Terakhir, terdapat fungsi search By Value yang mencari dan menampilkan data siswa berdasarkan rentang nilai tertentu. Di dalam fungsi utama terdapat perulangan do-while yang menampilkan menu kepada pengguna.

D. Kesimpulan

Keuntungan utama dari hash table dibandingkan struktur data lainnya adalah efisiensi dan kecepatan. Waktu yang dibutuhkan untuk mengakses sebuah elemen cukup cepat sehingga bisa lebih diandalkan. Jadi, tidak perlu memakan waktu atau usaha besar untuk menyimpan dan mencari data yang diperlukan.

E. Referensi

[1]Asisten Praktikum.2024. Modul V : Hash Table

[2]Algoritma Data Science Education Center. 2022.Pengertian HASH TABLE dan Cara Penggunaanya. Di akses pada 13 Mei 2024, dari https://algorit.ma/blog/hash-table-adalah-2022/

[3]Programiz. (n.d.). Hash Table. Diakses pada 13 Mei 2024, dari https://www.programiz.com/dsa/hash-table