LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL V
HASH TABLE

Disusun Oleh :

NAMA : ANISA YASAROH

NIM : 2311102063

Dosen :

WAHYU ANDI SAPUTRA, S.Pd., M.Eng.

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024

A. DASAR TEORI

a. Pengertian Hash Table

Hash Table merupakan struktur data yang secara asosiatif menyimpan data. Dalam hal ini, data disimpan dalam format array, dimana setiap nilai data memiliki nilai indeks uniknya sendiri. Akses data akan menjadi sangat cepat jika anda mengetahui indeks dari data yang diinginkan. Dengan demikian, has table memiliki struktur data dimana operasi penyisipan dan pencarian data terjadi sangat cepat terlepas dari ukuran data tersebut. Hash table menggunakan array sebagai media penyimpanan dan tekniknya untuk menghasilkan indeks suatu elemen yang dimasukkan atau ditempatkan.

b. Fungsi Hash Table

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.

Di dalam banyak bidang, Hash Table dikembangkan dan digunakan karena menawarkan kelebihan dalam efisiensi waktu operasi, mulai dari pengarsipan hingga pencarian data. Contohnya adalah bidang jaringan komputer yang mengembangkannya menjadi hybrid open Hash Table, yang kemudian dipakai untuk memproses jaringan komputer.

c. Teknik – Teknik Hash Table

berikut beberapa teknik yang umum digunakan saat data scientist melakukan hash table. Berikut ini penjelsannya :

1. Hashing

   Merupakan sebuah proses menganti kunci yang diberikan atau string karakter menjadi nilai lain. Penggunaan hashing paling popular 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.

2. Linear Probing

   Linear probing merupakan metode untuk menangani collision dalam hash table dengan cara mencari lokasi terdekat yang masih kosong untuk menyisipkan kunci baru. Meskipun dapat memberikan kinerja tinggi karena efisien secara referensi, pendekatan ini sensitif terhadap kualitas fungsi hash.

1. GUIDED

1. Guided 1 – Hash Table

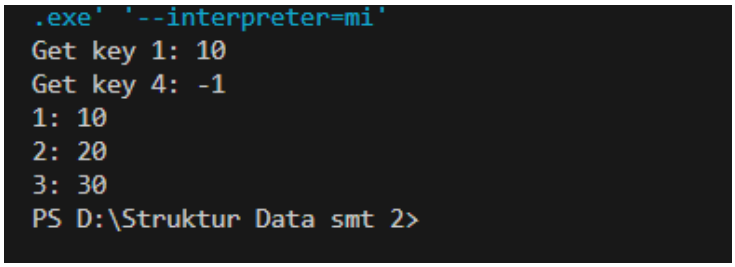
Source Code

```
#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;
}
```

Screenshoot Output



Deskripsi Program

Program diatas yaitu contoh sederhana implementasi Hash Table dengan menggunakan teknik linear probing untuk menangani collision. Hash Table dibuat dengan ukuran maksimum yang sudah ditentukan sebelumnya. Setiap sel dalam tabel hash adalah linked list yang menyimpan pasangan kunci-nilai. Fungsi hash sederhana digunakan untuk menentukan indeks dimana pasangan kunci-nilai akan disimpan. Program menyediakan fungsi untuk menyisipkan, mencari, dan menghapus data dalam tabel, serta menelusuri seluruh isi tabel. Saat menjalankan program, beberapa pasangan kunci-nilai disisipkan ke dalam tabel, kemudian nilai dari beberapa kunci dicari, dan akhirnya, data yang tidak diperlukan dihapus dari tabel.

2. Guided 2 – Hash Table

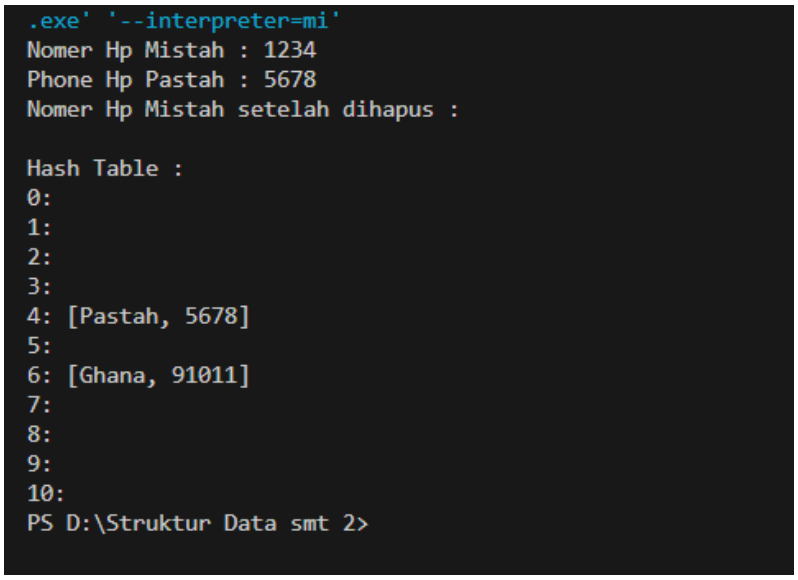
Source Code

```
#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;
}
```

Screenshoot Output



Deskripsi Program

Program diatas menggunakan sebuah hash table dengan ukuran yang telah ditentukan sebelumnya, dimana setiap elemen dalam tabel merupakan sebuah vektor yang berisi pointer ke objek HashNode. Setiap HashNode menyimpan sebuah pasangan nama dan nomor telepon. Fungsi hash digunakan untuk menghasilkan indeks dalam tabel berdasarkan nama. Program menyediakan fungsi untuk menyisipkan data baru, mencari nomor telepon berdasarkan nama, dan menghapus entri dari hash map. Pada saat eksekusi, beberapa pasangan nama dan nomor telepon disisipkan ke dalam hash map, kemudian pencarian dilakukan untuk beberapa nama yang telah ditambahkan, dan terakhir, satu entri dihapus dari hash map. Hasil dari operasi-operasi ini ditampilkan untuk menverifikasi keberhasilan operasi-insert, search, dan remove. Program juga mencetak seluruh isi hash map setelah operasi-operasi tersebut dilakukan.

2. UNGUIDED

Implementasikan Hash Table untuk menyimpan data mahasiswa. Setiap mahasiswa memiliki NIM dan nilai. Implementasikan fungsi untuk menambahkan data baru, menghapus data, mencari data berdasarkan NIM, dan mencari data berdasarkan nilai. Dengan ketentuan :

a. Setiap mahasiswa memiliki NIM dan nilai.
b. Program memiliki tampilan pilihan menu berisi poin C.
c. Implementasikan fungsi untuk menambahkan data baru, menghapus data, mencari data berdasarkan NIM, dan mencari data berdasarkan rentang nilai (80 – 90).

Source Code

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

const int TABLE_SIZE = 11;
string NIM_63;
int nilai;
class HashNode
{
public:
    string NIM_63;
    int nilai;
    HashNode(string NIM_63, int nilai)
    {
        this->NIM_63 = NIM_63;
        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_63, int nilai)
    {
        int hash_val = hashFunc(NIM_63);
        for (auto node : table[hash_val])
        {
            if (node->NIM_63 == NIM_63)
            {
                node->nilai = nilai;
                return;
            }
        }
        table[hash_val].push_back(new HashNode(NIM_63,
nilai));
    }
    void remove(string NIM_63)
    {
        int hash_val = hashFunc(NIM_63);
        for (auto it = table[hash_val].begin(); it !=
table[hash_val].end(); it++)
        {
            if ((*it)->NIM_63 == NIM_63)
            {
                table[hash_val].erase(it);
                return;
            }
        }
    }
    int search(string NIM_63)
    {
        int hash_val = hashFunc(NIM_63);
        for (auto node : table[hash_val])
        {
            if (node->NIM_63 == NIM_63)
            {
                return node->nilai;
            }
        }
        return -1; // return -1 if NIM is not found
    }
    void findNimByScore(int minScore, int maxScore)
    {
        bool found = false;
        for (int a = 0; a < TABLE_SIZE; a++)
        {
            for (auto pair : table[a])
            {
                if (pair != nullptr && pair->nilai >=
minScore && pair->nilai <= maxScore)
```

```
                {
                    cout << pair-> NIM_63 << " memiliki
nilai " << pair->nilai << endl;
                    found = true;
                }
            }
        }
        if (!found)
        {
            cout << "Tidak ada mahasiswa yang memiliki nilai
antara " << minScore << " and " << maxScore << endl;
        }
    }
    void print()
    {
        for (int a = 0; a < TABLE_SIZE; a++)
        {
            cout << a << ": ";
            for (auto pair : table[a])
            {
                if (pair != nullptr)
                {
                    cout << "[" << pair-> NIM_63 << ", " <<
pair->nilai << "]";
                }
            }
            cout << endl;
        }
    }
};

int main()
{
    HashMap data;
    string NIM;
    int nilai_mhs;
    while (true)
    {
        int menu;
        cout << "\nMenu" << endl;
        cout << "1. Tambah data" << endl;
        cout << "2. Hapus data" << endl;
        cout << "3. Mencari data berdasarkan NIM" << endl;
        cout << "4. Mencari data berdasarkan nilai" << endl;
        cout << "5. Cetak data" << endl;
        cout << "6. Exit" << endl;
        cout << "Masukkkan pilihan : ";
        cin >> menu;
```

```
        switch (menu)
        {
        case 1:
            cout << "Masukkan NIM : ";
            cin >> NIM;
            cout << "Masukkan nilai : ";
            cin >> nilai_mhs;
            data.insert(NIM, nilai_mhs);
            break;
        case 2:
            cout << "Masukkan NIM yang akan dihapus : ";
            cin >> NIM;
            data.remove(NIM);
            cout << "Data berhasil dihapus" << endl;
            break;
        case 3:
            cout << "Masukkan NIM yang akan dicari : ";
            cin >> NIM;
            cout << "NIM " << NIM << " memiliki nilai " <<
data.search(NIM) << endl;
            break;
        case 4:
            int x, y;
            cout << "Masukkan rentang nilai minimal : ";
            cin >> x;
            cout << "Masukkan rentang nilai maksimal : ";
            cin >> y;
            data.findNimByScore(x, y);
            break;
        case 5:
            data.print();
            break;
        case 6:
            return 0;
        default:
            cout << "Menu tidak tersedia!" << endl;
            break;
        }
    }
return 0;
}
```

Screenshoot Output

```
.exe' '--interpreter=mi'

Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 1
Masukkan NIM : 2311102063
Masukkan nilai : 89

Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 1
Masukkan NIM : 2311102068
Masukkan nilai : 90

Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 1
Masukkan NIM : 2311102073
Masukkan nilai : 85
```

```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 1
Masukkan NIM : 2311102007
Masukkan nilai : 75

Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 1
Masukkan NIM : 2311102079
Masukkan nilai : 65

Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 5
0: [2311102079, 65]
1:
2: [2311102007, 75]
3:
4: [2311102063, 89]
5: [2311102073, 85]
6:
7:
8:
9: [2311102068, 90]
10:
```

```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 2
Masukkan NIM yang akan dihapus : 2311102073
Data berhasil dihapus

Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 5
0: [2311102079, 65]
1:
2: [2311102007, 75]
3:
4: [2311102063, 89]
5:
6:
7:
8:
9: [2311102068, 90]
10:
```

```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 3
Masukkan NIM yang akan dicari : 2311102063
NIM 2311102063 memiliki nilai 89

Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 4
Masukkan rentang nilai minimal : 75
Masukkan rentang nilai maksimal : 90
2311102007 memiliki nilai 75
2311102063 memiliki nilai 89
2311102068 memiliki nilai 90
```

```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 5
0: [2311102079, 65]
1:
2: [2311102007, 75]
3:
4: [2311102063, 89]
5:
6:
7:
8:
9: [2311102068, 90]
10:

Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 6
PS D:\Struktur Data smt 2>
```

Deskripsi Program

Program diatas yaitu menggunakan hash map yang digunakan untuk menyimpan data mahasiswa berupa NIM dan nilai. Program menyediakan beberapa fitur interaktif, termasuk menambahkan data baru, menghapus data berdasarkan NIM, mencari data berdasarkan NIM, mencari data berdasarkan rentang nilai, dan mencetak seluruh data yang tersimpan. Saat program dijalankan, pengguna diminta untuk memilih opsi yang diinginkan dari menu, dan program akan menanggapi input sesuai dengan opsi yang dipilih oleh pengguna.

3. KESIMPULAN

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa :

1. Unordered_map, merupakan struktur data yang efisien untuk menyimpan data dengan kunci-nilai (key-value) pairs.
2. Program unu memanfaatkan Hash Table untuk menyimpan data mahasiswa berdasarkan NIM sebagai kunci dan nilai mahasiswa sebagai nilai.
3. Dengan fungsi hash table langsung menetapkan lokasi penyimpanan berdasarkan kunci, memungkinkan akses cepat tanpa pencarian melalui seluruh struktur. Hasilnya, kompleksitas waktu rata-rata pencarian data menjadi (0(1)) dalam kasus terbaik.

4. REFERENSI

Avini, T., Kom, S., Kom, M., Alamin, Z., Kom, M., Perkasa, E. B., & Kom, M. (2024). *Fundamental Algorithma*. Sada Kurnia Pustaka.

Holle, K. F. H. (2022). Modul praktikum struktur data.

Kaiser, R., & Kaiser, R. (2021). Containerklassen der C++-Standardbibliothek. *C++ mit Visual Studio 2019: C++ 17 für Studierende und erfahrene Programmierer*, 591-645.

Komalasari, R., Widians, J. A., Meilani, B. D., Arifin, N. Y., Sepriano, S., Syam, S., ... & Darwin, D. (2023). *PENGANTAR ILMU KOMPUTER: TEORI KOMPREHENSIF PERKEMBANGAN ILMU KOMPUTER TERKINI*. PT. Sonpedia Publishing Indonesia.

Mulyana, A. (2023). E-Books Cara Mudah Mempelajari Algoritma dan Struktur Data.