

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

**MODUL V
HASH TABLE**



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

Disusun oleh:

FALAH ASBI PANGESTU

(2311102045)

IF-11-B

**PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

BAB I

DASAR TEORI

A. Pengertian Hash Table

Hash Table adalah struktur data yang mengorganisir data ke dalam pasangan kunci-nilai. Hash table biasanya terdiri dari dua komponen utama: array (atau vektor) dan fungsi hash. Hashing adalah teknik untuk mengubah rentang nilai kunci menjadi rentang indeks array.

Array menyimpan data dalam slot-slot yang disebut bucket. Setiap bucket dapat menampung satu atau beberapa item data. Fungsi hash digunakan untuk menghasilkan nilai unik dari setiap item data, yang digunakan sebagai indeks array. Dengan cara ini, hash table memungkinkan pencarian data dalam waktu yang konstan ($O(1)$) dalam kasus terbaik.

Sistem hash table bekerja dengan cara mengambil input kunci dan memetakannya ke nilai indeks array menggunakan fungsi hash. Kemudian, data disimpan pada posisi indeks array yang dihasilkan oleh fungsi hash. Ketika data perlu dicari, input kunci dijadikan sebagai parameter untuk fungsi hash, dan posisi indeks array yang dihasilkan digunakan untuk mencari data. Dalam kasus hash collision, di mana dua atau lebih data memiliki nilai hash yang sama, hash table menyimpan data tersebut dalam slot yang sama dengan Teknik yang disebut chaining.

B. Fungsi Hash Tabel

Fungsi hash membuat pemetaan antara kunci dan nilai, hal ini dilakukan melalui penggunaan rumus matematika yang dikenal sebagai fungsi hash. Hasil dari fungsi hash disebut sebagai nilai hash atau hash. Nilai hash adalah representasi dari string karakter asli tetapi biasanya lebih kecil dari aslinya. \

C. Operasi Hash Tabel

Insertion: Memasukkan data baru ke dalam hash table dengan memanggil fungsi hash untuk menentukan posisi bucket yang tepat, dan kemudian menambahkan data ke bucket tersebut.

Deletion: Menghapus data dari hash table dengan mencari data menggunakan fungsi hash, dan kemudian menghapusnya dari bucket yang sesuai.

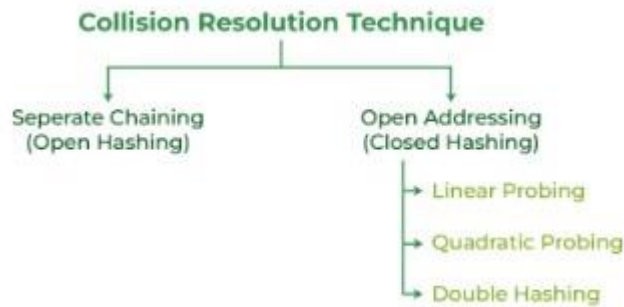
Searching: Mencari data dalam hash table dengan memasukkan input kunci ke fungsi hash untuk menentukan posisi bucket, dan kemudian mencari data di dalam bucket yang sesuai.

Update: Memperbarui data dalam hash table dengan mencari data menggunakan fungsi hash, dan kemudian memperbarui data yang ditemukan.

Traversal: Melalui seluruh hash table untuk memproses semua data yang ada dalam tabel.

D. Collision Resolution

Keterbatasan tabel hash adalah jika dua angka dimasukkan ke dalam fungsi hash menghasilkan nilai yang sama. Hal ini disebut dengan collision. Ada dua teknik untuk menyelesaikan masalah ini diantaranya :



1. Open Hashing (Chaining)

Metode chaining mengatasi collision dengan cara menyimpan semua item data dengan nilai indeks yang sama ke dalam sebuah linked list. Setiap node pada linked list merepresentasikan satu item data. Ketika ada pencarian atau penambahan item data, pencarian atau penambahan dilakukan pada linked list yang sesuai dengan indeks yang telah dihitung dari kunci yang di hash. Ketika linked list memiliki banyak node, pencarian atau penambahan item data menjadi lambat, karena harus mencari di seluruh linked list. Namun, chaining dapat mengatasi jumlah item data yang besar dengan efektif, karena keterbatasan array dihindari.

2. Closed Hashing • Linear Probing

Pada saat terjadi collision, maka akan mencari posisi yang kosong di bawah tempat terjadinya collision, jika masih penuh terus ke bawah, hingga ketemu tempat yang kosong. Jika tidak ada tempat yang kosong berarti HashTable sudah penuh.

• Quadratic Probing

Penanganannya hampir sama dengan metode linear, hanya lompatannya tidak satu-satu, tetapi quadratic (12, 22, 32, 42, ...)

• Double Hashing

Pada saat terjadi collision, terdapat fungsi hash yang kedua untuk menentukan posisinya kembali.

BAB II

GUIDED

LATIHAN – GUIDED

1. Guided 1

Guided (berisi screenshot source code & output program disertai penjelasannya)

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;

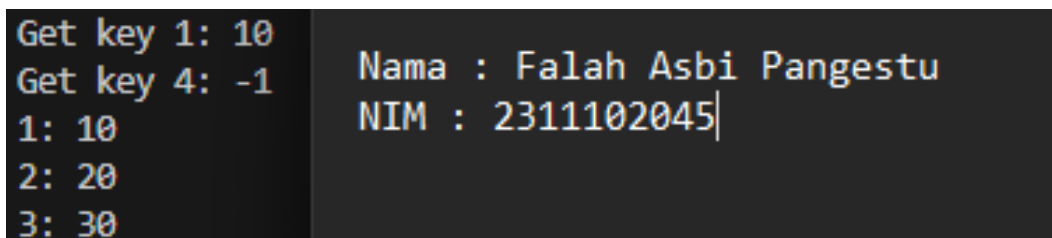
    // Deletion
    ht.remove(4);

    // Traversal
    ht.traverse();

    return 0;
}

```

Screenshoot program



```

Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
Nama : Falah Asbi Pangestu
NIM : 2311102045|

```

Deskripsi program

Hash table adalah struktur data yang digunakan untuk menyimpan pasangan kunci-nilai, di mana setiap elemennya dapat diakses menggunakan kunci yang spesifik. Kode ini menyediakan fungsi dasar untuk menyisipkan (insert), mencari (get), menghapus (remove), dan melakukan traversal terhadap elemen-elemen dalam hash table. Algoritma hash yang digunakan adalah hash_func, yang melakukan operasi modulo pada kunci untuk menentukan indeks di mana elemen tersebut akan disimpan dalam array yang mewakili hash table. Setiap elemen dalam hash table direpresentasikan oleh struktur data Node, yang mencakup kunci, nilai, dan pointer ke node berikutnya jika terjadi tumpang tindih (collision).

2. Guided 2

Source code

```
#include <iostream>

#include <string>
#include <vector>
using namespace std;

const int TABLE_SIZE = 11;

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");

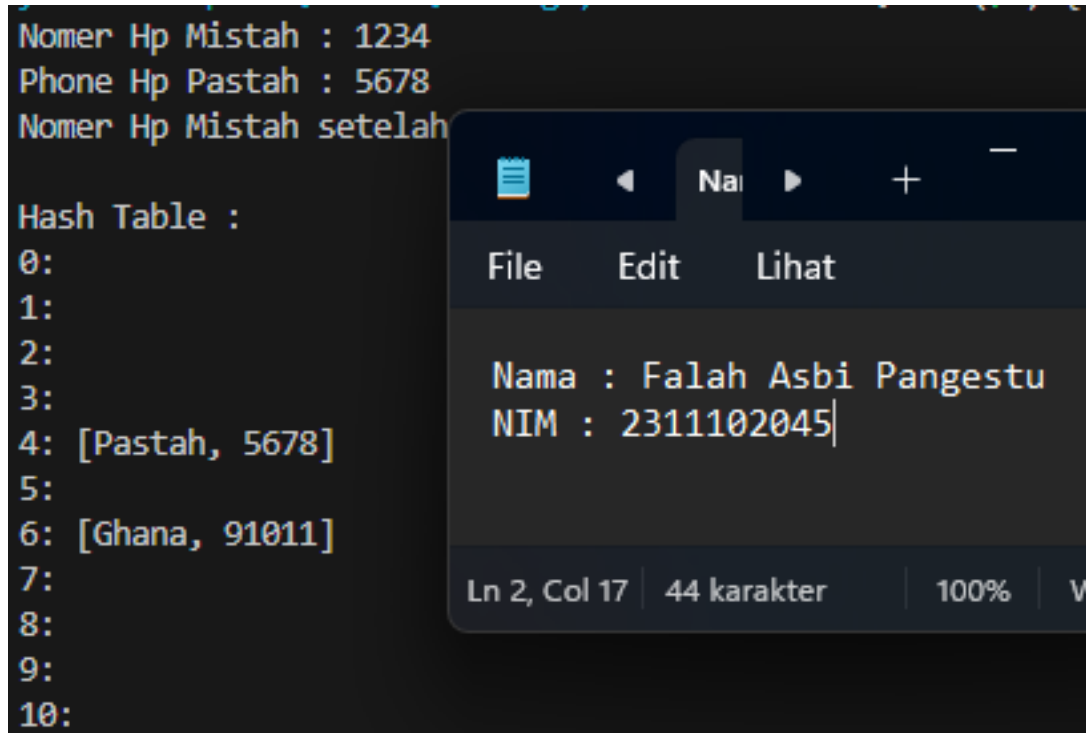
    cout << "Nomer Hp Mistah setelah dihapus : " <<
employee_map.searchByName("Mistah") << endl << endl;

    cout << "Hash Table : " << endl;
    employee_map.print();

    return 0;
}

```

Screenshoot program



The screenshot shows a Java program's output in a terminal window. The output displays phone numbers for 'Mistah' and 'Pastah', and a 'Hash Table' with 11 slots. Slot 4 contains '[Pastah, 5678]' and slot 6 contains '[Ghana, 91011]'. Overlaid on the terminal is a text editor window titled 'Nama' with a menu bar (File, Edit, Lihat) and a text area containing 'Nama : Falah Asbi Pangestu' and 'NIM : 2311102045'. The status bar at the bottom of the editor shows 'Ln 2, Col 17', '44 karakter', '100%', and a window icon.

```
Nomer Hp Mistah : 1234
Phone Hp Pastah : 5678
Nomer Hp Mistah setelah

Hash Table :
0:
1:
2:
3:
4: [Pastah, 5678]
5:
6: [Ghana, 91011]
7:
8:
9:
10:
```

Nama

File Edit Lihat

Nama : Falah Asbi Pangestu
NIM : 2311102045

Ln 2, Col 17 | 44 karakter | 100% | W

Deskripsi program

Dalam struktur hash map ini, setiap elemen terdiri dari pasangan kunci-nilai. Kunci berupa string yang merepresentasikan nama, sedangkan nilai adalah nomor telepon terkait. Implementasinya memanfaatkan array yang berisi vektor. Tiap vektor mengandung pointer ke objek `HashNode` yang menyimpan pasangan kunci-nilai. Fungsi hash dipakai untuk menentukan indeks di mana elemen akan disimpan dalam array. Fungsi-fungsi seperti `insert`, `remove`, dan `searchByName` digunakan untuk operasi penambahan, penghapusan, dan pencarian berdasarkan nama. Metode `print` digunakan untuk mencetak seluruh isi hash map, menampilkan pasangan kunci-nilai yang disimpan di setiap slot tabel hash.

BAB III

UNGUIDED

TUGAS – UNGUIDED

1. Unguided 1

Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Buatlah program menu Linked List Non Circular untuk menyimpan Nama dan NIM mahasiswa, dengan menggunakan input dari user

Source Code

```
#include <iostream>

#include <string>
#include <vector>
using namespace std;

// Struktur data untuk setiap mahasiswa
struct Mahasiswa {
    string nim;
    string nama;
    int nilai;

    Mahasiswa(string nim, string nama, int nilai) : nim(nim),
nama(nama), nilai(nilai) {}
};

// Class HashNode untuk setiap node pada hash table
class HashNode {
public:
    string nim;
    string nama;
    int nilai;

    HashNode(string nim, string nama, int nilai) : nim(nim),
nama(nama), nilai(nilai) {}
};

// Class HashMap untuk hash table
```

```

class HashMap {
private:
    static const int TABLE_SIZE = 10;
    vector<HashNode*> table[TABLE_SIZE];

public:
    // Fungsi hash sederhana
    int hashFunc(string key) {
        int hash_val = 0;
        for (char c : key) {
            hash_val += c;
        }
        return hash_val % TABLE_SIZE;
    }

    // Menambahkan data mahasiswa baru
    void insert(string nim, string nama, int nilai) {
        int hash_val = hashFunc(nim);
        table[hash_val].push_back(new HashNode(nim, nama, nilai));
    }

    // Menghapus data mahasiswa berdasarkan NIM
    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);
                return;
            }
        }
    }

    // Mencari data mahasiswa berdasarkan NIM

```

```

int searchByNIM(string nim) {
    int hash_val = hashFunc(nim);
    for (auto node : table[hash_val]) {
        if (node->nim == nim) {
            return node->nilai;
        }
    }
    return -1; // Mahasiswa tidak ditemukan
}

// Mencari data mahasiswa berdasarkan rentang nilai (80 - 90)
vector<string> searchByRange(int minNilai, int maxNilai) {
    vector<string> result;
    for (int i = 0; i < TABLE_SIZE; i++) {
        for (auto node : table[i]) {
            if (node->nilai >= minNilai && node->nilai <=
maxNilai) {
                result.push_back(node->nama);
            }
        }
    }
    return result;
}

};

int main() {
    HashMap mahasiswaMap;
    int choice;
    do {
        cout << "Menu:" << endl;
        cout << endl;
        cout << "1. Tambah data mahasiswa" << endl;
        cout << "2. Hapus data mahasiswa" << endl;
        cout << "3. Cari data mahasiswa berdasarkan NIM" << endl;
    }
}

```

```
        cout << "4. Cari data mahasiswa berdasarkan rentang nilai  
(80 - 90)" << endl;  
  
        cout << "5. Keluar" << endl;  
  
        cout << "Pilih: ";  
  
        cin >> choice;  
  
  
        cout << endl;  
  
        switch (choice) {  
            case 1: {  
                string nim, nama;  
                int nilai;  
                cout << "Masukkan NIM: ";  
                cin >> nim;  
                cout << "Masukkan Nama: ";  
                cin.ignore();  
                getline(cin, nama);  
                cout << "Masukkan nilai: ";  
                cin >> nilai;  
                mahasiswaMap.insert(nim, nama, nilai);  
                cout << "Data mahasiswa ditambahkan." << endl;  
                cout << endl ;  
                break;  
            }  
            case 2: {  
                string nim;  
                cout << "Masukkan NIM yang ingin dihapus: ";  
                cin >> nim;  
                mahasiswaMap.remove(nim);  
                cout << "Data mahasiswa dihapus." << endl;  
                cout << endl ;  
                break;  
            }  
            case 3: {
```

```

        string nim;
        cout << "Masukkan NIM yang ingin dicari: ";
        cin >> nim;
        int nilai = mahasiswaMap.searchByNIM(nim);
        if (nilai != -1) {
            cout << "Nilai mahasiswa dengan NIM " << nim <<
" adalah: " << nilai << endl;
        } else {
            cout << "Mahasiswa dengan NIM " << nim << "
tidak ditemukan." << endl;
        }
        cout << endl ;
        break;
    }
    case 4: {
        vector<string> mahasiswa80_90 =
mahasiswaMap.searchByRange(80, 90);
        if (mahasiswa80_90.size() > 0) {
            cout << "Mahasiswa dengan nilai antara 80 dan 90
adalah: ";
            for (string nama : mahasiswa80_90) {
                cout << nama << " ";
            }
            cout << endl;
        } else {
            cout << "Tidak ada mahasiswa dengan nilai antara
80 dan 90." << endl;
        }
        cout << endl ;
        break;
    }
    case 5: {
        cout << "Program selesai." << endl;
        cout << endl ;
        break;
    }
}

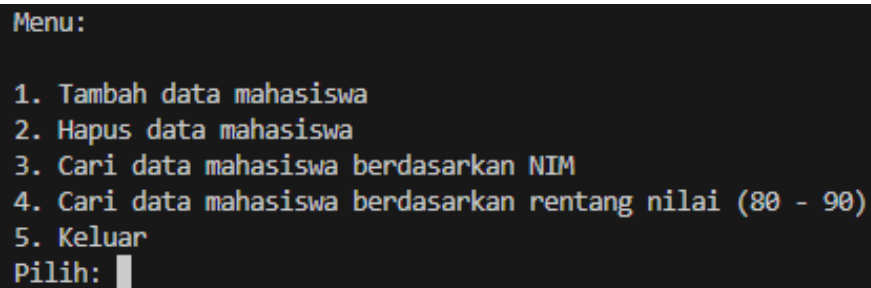
```



```
        default: {  
            cout << "Pilihan tidak valid. Silakan pilih  
kembali." << endl;  
            cout << endl ;  
            break;  
        }  
    }  
    } while (choice != 5);  
  
    return 1;  
}
```

Screenshot Program

- **Tampilan Menu**

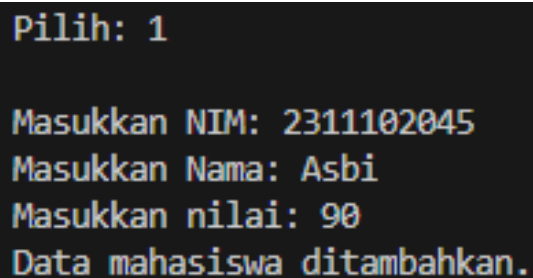


Menu:

1. Tambah data mahasiswa
2. Hapus data mahasiswa
3. Cari data mahasiswa berdasarkan NIM
4. Cari data mahasiswa berdasarkan rentang nilai (80 - 90)
5. Keluar

Pilih: █

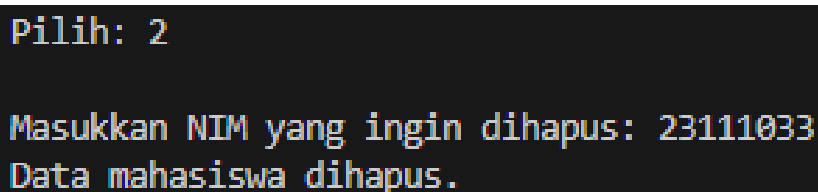
- **Tambah Data**



Pilih: 1

Masukkan NIM: 2311102045
Masukkan Nama: Asbi
Masukkan nilai: 90
Data mahasiswa ditambahkan.

- **Menghapus Data**



Pilih: 2

Masukkan NIM yang ingin dihapus: 23111033
Data mahasiswa dihapus.

- **Cari Berdasarkan NIM**

```
Pilih: 3  
  
Masukkan NIM yang ingin dicari: 2311102045  
Nilai mahasiswa dengan NIM 2311102045 adalah: 90
```

- **Cari Berdasarkan rentan nilai**

```
Pilih: 4  
  
Mahasiswa dengan nilai antara 80 dan 90 adalah: oji Asbi
```

Deskripsi program

Sebuah program yang memanfaatkan peta hash diimplementasikan untuk menyimpan informasi siswa, dengan Nomor Induk Siswa (NIM) sebagai kuncinya. Peta hash menawarkan berbagai metode, antara lain insert untuk menambahkan data siswa baru, delete untuk menghapus data berdasarkan NIM, searchByNIM untuk mengambil data berdasarkan NIM, dan searchByRange untuk mencari data dalam rentang kelas tertentu. Program ini menyajikan menu kepada pengguna, menawarkan opsi untuk menambah, menghapus, dan mencari informasi siswa. Hasil operasi ini kemudian ditampilkan kepada pengguna.

BAB IV

KESIMPULAN

Dalam bidang pemrograman komputer, tabel hash menonjol sebagai struktur data yang penting dan sering digunakan. Tabel ini memanfaatkan kekuatan fungsi hash untuk menetapkan kunci ke indeks tertentu, sehingga menghasilkan penyimpanan data yang efisien dan pengambilan data yang cepat. Keunggulan tabel hash terletak pada kemampuannya untuk menjalankan operasi dasar, seperti memasukkan, mengambil, dan menghapus data, dalam waktu yang konstan ($O(1)$). Namun, penting untuk dicatat bahwa meskipun tabel hash unggul dalam skenario kasus rata-rata, kinerjanya dalam skenario terburuk dapat terhambat oleh fungsi hash yang tidak seimbang atau seringnya tabrakan. Untuk memastikan kinerja yang optimal, sangat penting untuk memilih ukuran tabel hash dengan hati-hati dan menerapkan fungsi hash yang efisien.

DAFTAR PUSTAKA

- [1] Asisten Praktikum, “Modul 5 Hash Table”, Learning Management System, 2024
- [2] Bender, M. A., Conway, A., Farach-Colton, M., Kuszmaul, W., & Tagliavini, G. (2021).
Iceberg Hashing: Optimizing Many Hash-Table Criteria at Once. Computer Science > Data Structures and Algorithms. arXiv preprint arXiv:2109.04548