LAPORAN PRAKTIKUM

MODUL 5 HASH TABLE

Disusun Oleh:

Fahri Ramadhan

2311102024

Dosen

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2023

BAB I

TUJUAN PRAKTIKUM

- Mahasiswa mampu menjelaskan definisi dan konsep dari Hash Code
- Mahasiswa mampu menerapkan Hash Code kedalam pemrogrman

DASAR TEORI

1. Pengertian Hash Table

   Tabel Hash adalah struktur data yang digunakan untuk menyimpan pasangan kunci/nilai. Hash table biasanya terdiri dari dua komponen utama: array (atau vector) dan fungsi hash. Dengan menggunakan fungsi hash yang baik hashing dapat berjalan dengan baik. Hashing adalah teknik untuk mengubah rentang nilai kunci menjadi rentang indeks array

2. Fungsi Hash Table

   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.

3. Operasi Hash Table
   - 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

4. Collision Resolution

BAB II

Keterbatasan tabel hash adalah jika dua angka dimasukkan ke dalam fungsi hash menghasilkan nilai yang sama. Hal ini disebut dengan collision.

GUIDED

BAB III

1. Guided 1 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;
}
```

Output

```
PS C:\Users\fahri\OneDrive\Documents\strukdat tes>  & 'c:
4\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Mi
-phsjjng1.2t1' '--stderr=Microsoft-MIEngine-Error-0powje1
ogram Files\CodeBlocks\MinGW\bin\gdb.exe' '--interpreter=
Get key 1: 10
Get key 4: -1
2: 20
3: 30
PS C:\Users\fahri\OneDrive\Documents\strukdat tes>
```

Deskripsi program

Program tersebut menyimpan bucket dalam hash table menggunakan array dinamis. Sebuah linked list dengan setiap node meprentasikan satu item data mewakili setiap bucket.

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) {
                delete *it;
                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 << "Nomer 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;
```

Output

```
PS C:\Users\fahri\OneDrive\Documents\strukdat tes>  & 'c:\U
4\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Micr
-u2n5fh2t.jny' '--stderr=Microsoft-MIEngine-Error-thcqem13.
ogram Files\CodeBlocks\MinGW\bin\gdb.exe' '--interpreter=mi
Nomer Hp Mistah : 1234
Nomer Hp Pastah : 5678
Nomer Hp Mistah setelah dihapus :

Hash Table :
0:
1:
2:
3:
4: [Pastah, 5678]
5:
6: [Ghana, 91011]
7:
8:
9:
10:
PS C:\Users\fahri\OneDrive\Documents\strukdat tes>
```

Deskripsi program

Program tersebut mempunyai fungsi hashFunc untuk menghitung nilai hash, fungsi insert untuk menambahkan data ke dalam hash table, fungsi remove untuk menghapus data dari hash table, dan fungsi searchByName untuk mencari no telepon menggunakan nama.

UNGUIDED

1. Unguided 1 Source code

```
#include <iostream>
#include
<unordered_map>
#include <vector> using
namespace std;
 struct
Mahasiswa
{     string
NIM;     int
nilai;
};  class
HashTable
{
private:
    unordered_map<string, Mahasiswa> tabel;
 public:     void tambahData(Mahasiswa
mahasiswa)
    {          tabel[mahasiswa.NIM] =
mahasiswa;
    }      void
hapusData(string NIM)
    {
tabel.erase(NIM);
    }

    Mahasiswa cariDataBerdasarkanNIM(string NIM)
```

```
    {          if (tabel.find(NIM) !=
tabel.end())          {              return
tabel[NIM];
        }          else          {
throw "NIM tidak ditemukan";
        }
    }
    vector<Mahasiswa>      cariDataBerdasarkanRentangNilai(int
nilaiMin, int nilaiMax)
    {          vector<Mahasiswa> hasil;          for (auto it =
tabel.begin(); it != tabel.end(); it++)
        {
            if (it->second.nilai >= nilaiMin && it-
>second.nilai <= nilaiMax)
            {
hasil.push_back(it->second);
            }
}          return
hasil;
    }
};  int
main()
{
    HashTable hashTable;     int
pilihan;     do     {
cout << "Menu:" << endl;
```

```
        cout << "1. Tambah data baru" << endl;
cout << "2. Hapus data" << endl;         cout << "3.
Cari data berdasarkan NIM" << endl;
        cout << "4. Cari data berdasarkan rentang nilai (80-
90)" << endl;         cout << "5.
Keluar" << endl;         cout <<
"Masukkan pilihan: ";         cin >>
pilihan;
         if (pilihan ==
1)
        {
            Mahasiswa mahasiswa;             cout <<
"Masukkan NIM: ";             cin >> mahasiswa.NIM;
cout << "Masukkan nilai: ";             cin >>
mahasiswa.nilai;
hashTable.tambahData(mahasiswa);             cout <<
"Data berhasil ditambahkan" << endl;             cout
<< endl;
        }         else if
(pilihan == 2)
        {             string NIM;             cout <<
"Masukkan NIM yang ingin dihapus: ";             cin >>
NIM;             hashTable.hapusData(NIM);
cout << "Data berhasil dihapus" << endl;
cout << endl;
        }         else if
(pilihan == 3)
        {             string NIM;             cout <<
"Masukkan NIM yang ingin dicari: ";
```

```
            cin >> NIM;
try
            {
                Mahasiswa             mahasiswa             =
hashTable.cariDataBerdasarkanNIM(NIM);
                cout << "NIM: " << mahasiswa.NIM << ", Nilai:
" << mahasiswa.nilai << endl;
            }             catch
(const char *msg)
            {                 cerr
<< msg << endl;
            }
cout << endl;
        }
         else if (pilihan ==
4)
        {             int nilaiMin,
nilaiMax;
            cout << "Masukkan rentang nilai yang ingin dicari
(misal: 80 90): ";             cin >>
nilaiMin >> nilaiMax;


            vector<Mahasiswa>             hasil             =
hashTable.cariDataBerdasarkanRentangNilai(nilaiMin,
nilaiMax);             for (Mahasiswa mahasiswa :
hasil)
            {
                cout << "NIM: " << mahasiswa.NIM << ", Nilai:
" << mahasiswa.nilai << endl;
            }
cout << endl;
        }
    } while (pilihan != 5);
return 0;
}
```

Output

a. tambah data

```
4\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=M
-u5cvawaf.2tc' '--stderr=Microsoft-MIEngine-Error-2mk2hp
ogram Files\CodeBlocks\MinGW\bin\gdb.exe' '--interpreter
Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 1
Masukkan NIM: 2311102050
Masukkan nilai: 60
Data berhasil ditambahkan

Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 1
Masukkan NIM: 2311102060
Masukkan nilai: 70
Data berhasil ditambahkan

Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 1
Masukkan NIM: 2311102070
Masukkan nilai: 80
Data berhasil ditambahkan
```

```
Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 1
Masukkan NIM: 2311102080
Masukkan nilai: 90
Data berhasil ditambahkan

Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 1
Masukkan NIM: 2311102090
Masukkan nilai: 100
Data berhasil ditambahkan
```

b. Hapus data

```
Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 2
Masukkan NIM yang ingin dihapus: 2311102090
Data berhasil dihapus
```

c. Mencari data berdasarkan nim

```
Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 3
Masukkan NIM yang ingin dicari: 2311102090
NIM tidak ditemukan
```

d. Mencari data berdasarkan nilai

```
Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 4
Masukkan rentang nilai yang ingin dicari (misal: 80 90): 80 90
NIM: 2311102070, Nilai: 80
NIM: 2311102080, Nilai: 90
```

Deskripsi program

Program tersebut adalah hash table untuk menyimpan data mahasiswa dengan NIM dan nilai. Program tersebut bisa menambah dan menghapus data kemudian bisa juga mencari berdasarkan NIM atau rentang nilai.

BAB IV KESIMPULAN

Tabel Hash adalah struktur data untuk menyimpan pasangan kunci/nilai. Ini menggunakan array dan fungsi hash. Fungsi hash mengubah kunci menjadi indeks dalam array. Ketika mencari atau memasukkan data, kunci digunakan untuk menemukan posisi dalam array. Jika ada konflik hash, data disimpan dalam bucket yang sama dengan metode chaining. Tabel Hash memungkinkan pencarian data dengan cepat dalam waktu konstan.