LAPORAN PRAKTIKUM

PERTEMUAN 14

Nama :

Razhendriya Vania Ramadahan suganjarsarwat (2311104048)

Dosen :

WAHYU ANDI SAPUTRA

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

Memahami cara implementasi graf untuk menghitung jarak antar kota dan merepresentasikan graf tidak berarah menggunakan adjacency matrix.

II. TOOL

VScode

III. DASAR TEORI

Graf merupakan struktur data yang terdiri dari simpul (vertex) dan sisi (edge) yang menghubungkan simpul-simpul tersebut. Dalam representasi graf tidak berarah, hubungan antar simpul bersifat dua arah. Salah satu metode untuk merepresentasikan graf adalah dengan menggunakan matriks ketetanggaan (adjacency matrix), yaitu matriks yang elemen-elemennya menunjukkan apakah dua simpul terhubung.

IV. GUIDED

```cpp
#include <iostream>
#include <queue>

using namespace std;

struct ElmNode;

struct ElmEdge {
    ElmNode *Node;
    ElmEdge *Next;
};

struct ElmNode {
    char info;
    bool visited;
    ElmEdge *FirstEdge;
    ElmNode *Next;
};

struct Graph {
    ElmNode *first;
};

void CreateGraph(Graph &G) {
  G.first = NULL;
}

void InsertNode(Graph &G, char X) {
    ElmNode *newNode = new ElmNode;
    newNode->info = X;
    newNode->visited = false;
    newNode->FirstEdge = NULL;
    newNode->Next = NULL;

    if (G.first == NULL) {
        G.first = newNode;
    } else {
        ElmNode *temp = G.first;
        while (temp->Next != NULL) {
            temp = temp->Next;
        }
        temp->Next = newNode;
    }
}

void ConnectNode(ElmNode *N1, ElmNode *N2) {
    ElmEdge *newEdge = new ElmEdge;
    newEdge->Node = N2;
    newEdge->Next = N1->FirstEdge;
    N1->FirstEdge = newEdge;
}

void PrintInfoGraph(Graph G) {
    ElmNode *temp = G.first;
    while (temp != NULL) {
        cout << temp->info << " ";
        temp = temp->Next;
    }
    cout << endl;
}

void ResetVisited(Graph &G) {
    ElmNode *temp = G.first;
    while (temp != NULL) {
        temp->visited = false;
        temp = temp->Next;
    }
}

void PrintDFS(Graph G, ElmNode *N) {
    if (N == NULL) {
        return;
    }
    N->visited = true;
    cout << N->info << " ";
    ElmEdge *edge = N->FirstEdge;
    while (edge != NULL) {
        if (!edge->Node->visited) {
            PrintDFS(G, edge->Node);
        }
        edge = edge->Next;
    }
}

void PrintBFS(Graph G, ElmNode *N) {
    queue<ElmNode*> q;
    q.push(N);
    N->visited = true;

    while (!q.empty()) {
        ElmNode *current = q.front();
        q.pop();
        cout << current->info << " ";

        ElmEdge *edge = current->FirstEdge;
        while (edge != NULL) {
            if (!edge->Node->visited) {
                edge->Node->visited = true;
                q.push(edge->Node);
            }
            edge = edge->Next;
        }
    }
}

int main() {
    Graph G;
    CreateGraph(G);

    InsertNode(G, 'A');
    InsertNode(G, 'B');
    InsertNode(G, 'C');
    InsertNode(G, 'D');
    InsertNode(G, 'E');
    InsertNode(G, 'F');
    InsertNode(G, 'G');
    InsertNode(G, 'H');

    ElmNode *A = G.first;
    ElmNode *B = A->Next;
    ElmNode *C = B->Next;
    ElmNode *D = C->Next;
    ElmNode *E = D->Next;
    ElmNode *F = E->Next;
    ElmNode *G1 = F->Next;
    ElmNode *H = G1->Next;

    ConnectNode(A, B);
    ConnectNode(A, C);
    ConnectNode(B, D);
    ConnectNode(B, E);
    ConnectNode(C, F);
    ConnectNode(C, G1);
    ConnectNode(D, H);

    cout << "DFS traversal: ";
    ResetVisited(G);
    PrintDFS(G, A);
    cout << endl;

    cout << "DFS traversal: ";
    ResetVisited(G);
    PrintBFS(G, A);
    cout << endl;

    return 0;
}
```

ElmNode: Merepresentasikan simpul (node) dalam graf.

- Menyimpan info simpul, daftar sisi (edges), dan penanda kunjungan.

ElmEdge: Merepresentasikan sisi (edge) yang menghubungkan dua simpul.
Graph: Struktur utama graf yang menyimpan semua simpul.

V. UNGUIDED

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

using namespace std;

int main() {
    int jumlahSimpul;
    cout << "Silakan masukkan jumlah simpul: ";
    cin >> jumlahSimpul;

    vector<string> namaSimpul(jumlahSimpul);
    for (int i = 0; i < jumlahSimpul; i++) {
        cout << "Simpul " << i + 1 << ": ";
        cin >> namaSimpul[i];
       }

    vector<vector<int>> bobot(jumlahSimpul, vector<int>(jumlahSimpul, 0));

    for (int i = 0; i < jumlahSimpul; i++) {
        for (int j = 0; j < jumlahSimpul; j++) {
            if (i != j) {
                cout << namaSimpul[i] << "--> " << namaSimpul[j] << " = ";
                cin >> bobot[i][j];
                   }
             }
       }

    cout << "\nAdjacency Matrix:\n";
    cout << setw(8) << " ";
    for (int i = 0; i < jumlahSimpul; i++) {
        cout << setw(8) << namaSimpul[i];
       }
    cout << endl;

    for (int i = 0; i < jumlahSimpul; i++) {
        cout << setw(8) << namaSimpul[i];
        for (int j = 0; j < jumlahSimpul; j++) {
            cout << setw(8) << bobot[i][j];
             }
        cout << endl;
       }

    return 0;
}
```

1. Program Menghitung Jarak Antar Kota

- Input: Jumlah simpul, nama simpul, dan bobot (jarak) antar simpul.
- Proses:
    - Gunakan matriks bobot untuk menyimpan jarak antar simpul.
    - Jika dua simpul tidak terhubung, bobot = 0.

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

using namespace std;

int main() {
    int jumlahSimpul;
    cout << "Silakan masukkan jumlah simpul: ";
    cin >> jumlahSimpul;

    vector<string> namaSimpul(jumlahSimpul);
    for (int i = 0; i < jumlahSimpul; i++) {
        cout << "Simpul " << i + 1 << ": ";
        cin >> namaSimpul[i];
       }

    vector<vector<int>> bobot(jumlahSimpul, vector<int>(jumlahSimpul, 0));

    for (int i = 0; i < jumlahSimpul; i++) {
        for (int j = 0; j < jumlahSimpul; j++) {
            if (i != j) {
                cout << namaSimpul[i] << "--> " << namaSimpul[j] << " = ";
                cin >> bobot[i][j];
                   }
             }
       }

    cout << "\nAdjacency Matrix:\n";
    cout << setw(8) << " ";
    for (int i = 0; i < jumlahSimpul; i++) {
        cout << setw(8) << namaSimpul[i];
       }
    cout << endl;

    for (int i = 0; i < jumlahSimpul; i++) {
        cout << setw(8) << namaSimpul[i];
        for (int j = 0; j < jumlahSimpul; j++) {
            cout << setw(8) << bobot[i][j];
             }
        cout << endl;
       }

    return 0;
}
```

2. Program Graf Tidak Berarah (Adjacency Matrix)

- Input: Jumlah simpul, jumlah sisi, dan pasangan simpul yang terhubung.
- Proses:
    - Gunakan adjacency matrix biner (0/1) untuk menunjukkan hubungan antar simpul.

VI. KESIMPULAN

memahami konsep graf, penerapannya dalam pemrograman, dan cara merepresentasikan hubungan antar simpul menggunakan adjacency matrix.