LAPORAN PRAKTIKUM

Modul 14

Graph

Disusun Oleh :

Satria Ariq Adelard

Dompas/2211104033 SE 07 2

Asisten Praktikum :

Aldi Putra

Andini Nur Hidayah

Dosen Pengampu :

Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

1. **Tujuan**
    a. Mahasiswa mampu memahami konsep dasar graf (graph) sebagai himpunan simpul (nodes) dan sisi (edges).
    b. Mahasiswa mampu mempelajari representasi graf berarah dan tidak berarah menggunakan pointer.
    c. Mahasiswa mampu mengimplementasikan traversal graf dengan metode BFS dan DFS.
    d. Mahasiswa mampu membuat ADT graf untuk operasi dasar seperti membuat graf, menambah simpul, menghubungkan simpul, dan mencetak graf.
    e. Mahasiswa mampu menerapkan graf dalam kasus praktis seperti topological sorting untuk menyusun keterurutan data.
2. **Landasan Teori**

Graf adalah struktur data non-linear yang terdiri dari simpul (nodes) dan sisi (edges) yang menghubungkan simpul-simpul tersebut. Struktur ini digunakan untuk merepresentasikan hubungan antara berbagai objek. Graf dapat dibedakan menjadi dua jenis utama: graf berarah, di mana setiap sisi memiliki arah tertentu, dan graf tidak berarah, di mana sisi tidak memiliki arah tertentu. Graf memiliki berbagai aplikasi, seperti dalam jaringan komunikasi, peta jalan, dan analisis hubungan sosial.

Untuk merepresentasikan graf, umumnya digunakan dua metode: matriks ketetanggaan dan daftar ketetanggaan. Pemilihan metode bergantung pada jumlah sisi dalam graf, karena masing-masing memiliki kelebihan tersendiri. Proses traversal graf dapat dilakukan menggunakan algoritma Depth First Search (DFS), yang menjelajah secara mendalam, atau Breadth First Search (BFS), yang menjelajah secara melebar. Selain itu, graf juga dimanfaatkan dalam pengurutan topologis untuk menentukan urutan linier elemen-elemen yang memiliki ketergantungan parsial. Dalam implementasinya, graf sering menggunakan struktur pointer dinamis untuk efisiensi penyimpanan dan pengelolaan data yang kompleks.

3. **Guided**

a. Guided 1

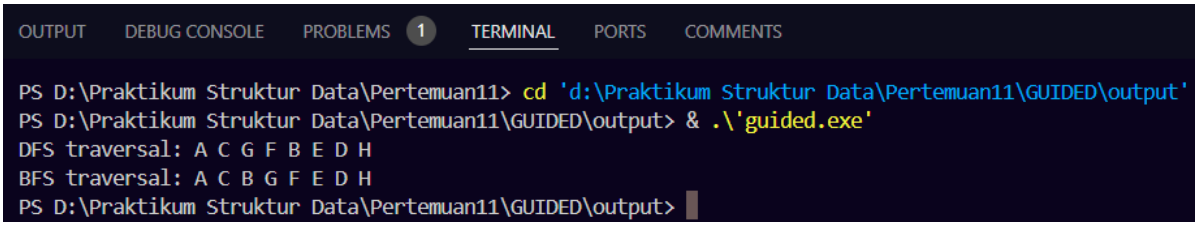
Source Code

```
#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 << "BFS traversal: ";
    ResetVisited(G);
    PrintBFS(G, A);
    cout << endl;

    return 0;
}
```

Output



Deskripsi

Program C++ ini mengimplementasikan graf tak berarah menggunakan struktur data list terhubung (linked list). Program ini mencakup berbagai fungsi, seperti membuat graf, menambahkan simpul (node), menghubungkan simpul melalui sisi (edge), dan traversal graf menggunakan metode Depth-First Search (DFS) serta Breadth-First Search (BFS). Graf disusun dari simpul-simpul yang diwakili oleh struktur ElmNode, yang menyimpan informasi berupa karakter, status kunjungan (visited), dan daftar sisi yang terhubung.

Metode DFS dan BFS dirancang untuk mengunjungi seluruh simpul dalam graf. DFS menjelajah simpul secara mendalam dengan pendekatan rekursi, sedangkan BFS memanfaatkan antrian (queue) untuk menjelajahi simpul secara bertingkat. Program ini juga dilengkapi fungsi untuk mereset status kunjungan simpul serta mencetak hasil traversal menggunakan kedua metode tersebut.

4. Unguided

a. Unguided 1

Source Code

```cpp
#include <iostream>
#include <vector>
#include <iomanip>
using namespace std;

void printMatrix(const vector<vector<int>> &matrix, const vector<string> &nodes)
{
    int n = nodes.size();

    cout << setw(8) << "";
    for (int i = 0; i < n; i++)
    {
        cout << setw(8) << nodes[i];
    }
    cout << endl;

    for (int i = 0; i < n; i++)
    {
```

```
        cout << setw(8) << nodes[i];
        for (int j = 0; j < n; j++)
        {
            cout << setw(8) << matrix[i][j];
        }
        cout << endl;
    }
}

int main()
{
    int numNodes;
    cout << "Silakhan Masukan Jumlah Simpul: ";
    cin >> numNodes;

    vector<string> nodes(numNodes);
    cout << "Silakan Masukan Nama Simpul:\n";
    for (int i = 0; i < numNodes; i++)
    {
        cout << "Simpul " << i + 1 << ": ";
        cin >> nodes[i];
    }

    vector<vector<int>> weights(numNodes, vector<int>(numNodes, 0));

    cout << "Silakan Masukkan Bobot antar Simpul:\n";
    for (int i = 0; i < numNodes; i++)
    {
        for (int j = 0; j < numNodes; j++)
        {
            if (i != j || (i == j && weights[i][j] == 0))
            {
                cout << nodes[i] << " --> " << nodes[j] << " = ";
                cin >> weights[i][j];
            }
        }
    }

    cout << "\n";
    printMatrix(weights, nodes);

    return 0;
}
```

Output

```
OUTPUT  DEBUG CONSOLE  PROBLEMS  TERMINAL  PORTS  COMMENTS

PS D:\Praktikum Struktur Data\Pertemuan11> cd 'd:\Praktikum Struktur Data\Pertemuan11\UNGUIDED\output'
PS D:\Praktikum Struktur Data\Pertemuan11\UNGUIDED\output> & .\'Unguided1.exe'
Silakhan Masukan Jumlah Simpul: 3
Silakan Masukan Nama Simpul:
Simpul 1: Bejo
Simpul 2: Yeni
Simpul 3: Kuncoro
Silakan Masukkan Bobot antar Simpul:
Bejo --> Bejo = 13
Bejo --> Yeni = 23
Bejo --> Kuncoro = 10
Bejo --> Kuncoro = 10
Bejo --> Kuncoro = 10
Bejo --> Kuncoro = 10
Yeni --> Bejo = 2
Yeni --> Yeni = 6
Yeni --> Kuncoro = 8
Kuncoro --> Bejo = 9
Kuncoro --> Yeni = 0
Kuncoro --> Kuncoro = 4

            Bejo    Yeni Kuncoro
    Bejo      13      23      10
    Yeni       2       6       8
 Kuncoro       9       0       4
PS D:\Praktikum Struktur Data\Pertemuan11\UNGUIDED\output>
```

Deskripsi

Program C++ ini dirancang untuk membangun dan menampilkan matriks bobot dari graf berbobot berdasarkan input pengguna. Program dimulai dengan meminta pengguna menentukan jumlah simpul (node) dalam graf, diikuti dengan memberikan nama untuk masing-masing simpul. Selanjutnya, pengguna diminta memasukkan bobot untuk setiap pasangan simpul, yang akan disimpan dalam matriks dua dimensi yang merepresentasikan graf. Dalam matriks ini, elemen matrix[i][j] merepresentasikan bobot sisi antara simpul i dan simpul j.

Setelah matriks selesai diisi, program mencetaknya dalam format tabel yang terstruktur. Nama-nama simpul ditempatkan pada header kolom dan baris, sementara bobot sisi antar simpul ditampilkan pada sel-sel matriks. Untuk sisi yang menghubungkan simpul dengan dirinya sendiri (misalnya dari simpul A ke A), program mengasumsikan bobotnya adalah 0, kecuali jika graf mendukung bobot khusus pada diagonalnya, seperti pada graf berarah.

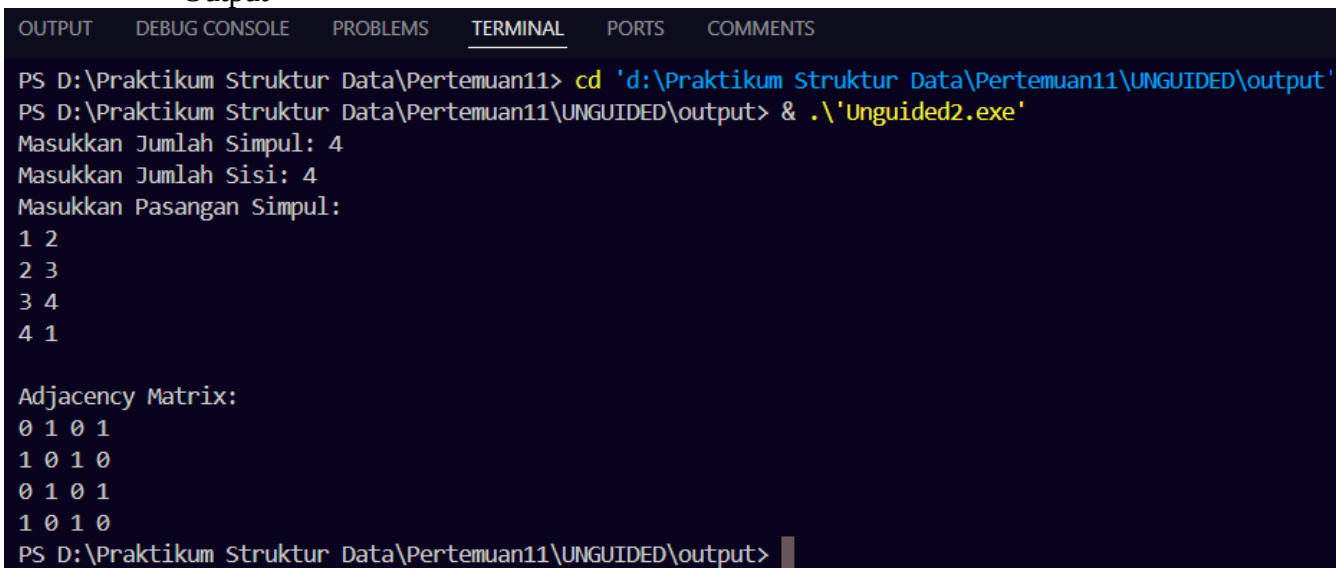
b. Unguided 2

Source Code

```cpp
#include <iostream>
#include <vector>
using namespace std;

void printAdjacencyMatrix(const vector<vector<int>> &matrix)
```

```
{
    int n = matrix.size();
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}

int main()
{
    int numNodes, numEdges;

    cout << "Masukkan Jumlah Simpul: ";
    cin >> numNodes;
    cout << "Masukkan Jumlah Sisi: ";
    cin >> numEdges;

    vector<vector<int>> adjacencyMatrix(numNodes, vector<int>(numNodes, 0));

    cout << "Masukkan Pasangan Simpul:\n";
    for (int i = 0; i < numEdges; i++)
    {
        int node1, node2;
        cin >> node1 >> node2;

        adjacencyMatrix[node1 - 1][node2 - 1] = 1;
        adjacencyMatrix[node2 - 1][node1 - 1] = 1;
    }

    cout << "\nAdjacency Matrix:\n";
    printAdjacencyMatrix(adjacencyMatrix);

    return 0;
}
```

Output



Deskripsi

Program C++ ini dirancang untuk membangun dan menampilkan matriks kedekatan (adjacency matrix) dari sebuah graf tak berarah. Program dimulai dengan meminta pengguna untuk memasukkan jumlah simpul (nodes) dan sisi (edges) dalam graf. Selanjutnya, pengguna diminta untuk memasukkan pasangan simpul yang terhubung oleh setiap sisi. Untuk setiap pasangan simpul yang terhubung, elemen matriks yang sesuai akan diisi dengan nilai 1, menandakan adanya hubungan antara simpul-simpul tersebut, sedangkan elemen lainnya tetap bernilai 0.

Setelah matriks selesai diisi, program mencetak matriks tersebut dengan format yang rapi, di mana baris dan kolom merepresentasikan simpul, dan setiap elemen menunjukkan ada atau tidaknya koneksi antara simpul-simpul terkait. Karena graf yang digunakan adalah graf tak berarah, matriks ini bersifat simetris, yang berarti jika ada sisi dari simpul A ke simpul B, maka sisi dari simpul B ke simpul A juga tercermin dalam matriks.