# LAPORAN PRAKTIKUM
# Modul 14
# GRAPH


**Disusun Oleh:**

Muhammad Shafiq Rasuna - 2311104043

**Kelas :**

S1SE-07-02

**Dosen :**

Wahyu Andi Saputra, S.Pd, M.Eng



**PROGRAM STUDI S1 SOFTWARE ENGINEERING**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY**

**PURWOKERTO**

**2024**

## 1. Tujuan

1. Memahami konsep graph
2. Mengimplementasikan graph dengan menggunakan pointer.

## 2. Dasar Teori

Graph merupakan himpunan tidak kosong dari node (vertec) dan garis penghubung (edge). Contoh sederhana tentang graph, yaitu antara Tempat Kost Anda dengan Common Lab. Tempat Kost Anda dan Common Lab merupakan node (vertec). Jalan yang menghubungkan tempat Kost dan Common Lab merupakan garis penghubung antara keduanya (edge).

Pada dasarnya representasi dari graph berarah sama dengan graph tak-berarah. Perbedaannya apabila graph tak-berarah terdapat node A dan node B yang terhubung, secara otomatis terbentuk panah bolak balik dari A ke B dan B ke A. Pada Graph berarah node A terhubung dengan node B, belum tentu node B terhubung dengan node A

Diberikan urutan partial dari elemen suatu himpunan, dikehendaki agar elemen yang terurut parsial tersebut mempunyai keterurutan linier. Contoh dari keterurutan parsial banyak dijumpai dalam kehidupan sehari-hari, misalnya: 1. Dalam suatu kurikulum, suatu mata pelajaran mempunyai prerequisite mata pelajaran lain. Urutan linier adalah urutan untuk seluruh mata pelajaran dalamkurikulum 2. Dalam suatu proyek, suatu pekerjaan harus dikerjakan lebih dulu dari pekerjaan lain (misalnya membuat fondasi harus sebelum dinding, membuat dinding harus sebelum pintu. Namun pintu dapat dikerjakan bersamaan dengan jendela, dsb 3. Dalam sebuah program Pascal, pemanggilan prosedur harus sedemikian rupa, sehingga peletakan prosedur pada teks program harus seuai dengan urutan (partial) pemanggilan. Dalam pembuatan tabel pada basis data, tabel yang di-refer oleh tabel lain harus dideklarasikan terlebih dulu. Jika suatu aplikasi terdiri dari banyak tabel, maka urutan pembuatan tabel harus sesuai dengan definisinya.

Kenapa disebut salah satu urutan linier ? Karena suatu urutan partial akan mempunyai banyak urutan linier yang mungkin dibentuk dari urutan partial tersebut. Elemen yang membentuk urutan linier disebut sebagai “list”. Proses yang dilakukan untuk mendapatkan urutan linier :

1. Andaikata item yang mempunyai keterurutan partial adalah anggota himpunan S.
2. Pilih salah satu item yang tidak mempunyai predecessor, misalnya X. Minimal adasatu elemen semacam ini. Jika tidak, maka akan looping.
3. Hapus X dari himpunan S, dan insert ke dalam list
4. Sisa himpunan S masih merupakan himpunan terurut partial, maka proses 2-3 dapat dilakukan lagi terhadap sisa dari S
5. Lakukan sampai S menjadi kosong, dan list Hasil mempunyai elemen dengan keterurutan linier

## 3. Guided

### 1. Kode programnya:

```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;
}
```

```cpp
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;
        }
    }
}
```

```cpp
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;
}
```

**Hasil run :**

```
nnerFile
DFS traversal: A C G F B E D H
BFS traversal: A C B G F E D H
```

## 4. Unguided

1. Buatlah program graph dengan menggunakan inputan user untuk menghitung jarak dari sebuah kota ke kota lainnya.

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

int main() {
    int jumlahSimpul;

    cout << "Silakan masukkan jumlah simpul: ";
    cin >> jumlahSimpul;
    cin.ignore();

    vector<string> simpul(jumlahSimpul);

    for (int i = 0; i < jumlahSimpul; i++) {
        cout << "Silakan masukkan nama simpul" << endl;
        cout << "Simpul " << i + 1 << ": ";
        getline(cin, simpul[i]);
    }

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

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

    cout << "\n\t";
    for (const string& namaSimpul : simpul) {
        cout << setw(8) << namaSimpul;
    }
    cout << endl;

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

    return 0;
}
```

Hasil run:

```
Silakan masukkan jumlah simpul: 2
Silakan masukkan nama simpul
Simpul 1: BALI
Silakan masukkan nama simpul
Simpul 2: PALU
BALI--> PALU = 3
PALU--> BALI = 4

          BALI    PALU
   BALI      0       3
   PALU      4       0
```

**2.** Buatlah sebuah program C++ untuk merepresentasikan graf tidak berarah menggunakan adjacency matrix. Program harus dapat:

- Menerima input jumlah simpul dan jumlah sisi.
- Menerima input pasangan simpul yang terhubung oleh sisi.
- Menampilkan adjacency matrix dari graf tersebut.

Kode programnya:

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

int main() {
    int jumlahSimpul, jumlahSisi;

    // Meminta input jumlah simpul dan sisi
    cout << "Masukkan jumlah simpul: ";
    cin >> jumlahSimpul;
    cout << "Masukkan jumlah sisi: ";
    cin >> jumlahSisi;

    // Inisialisasi adjacency matrix dengan nilai 0
    vector<vector<int>> adjacencyMatrix(jumlahSimpul, vector<int>(jumlahSimpul, 0));

    // Meminta pasangan simpul yang terhubung
    cout << "Masukkan pasangan simpul:\n";
    for (int i = 0; i < jumlahSisi; i++) {
        int simpul1, simpul2;
        cin >> simpul1 >> simpul2;

        // Karena graf tak berarah, set adjacencyMatrix[simpul1][simpul2] dan adjacencyMatrix[simpul2][simpul1] ke 1
        adjacencyMatrix[simpul1 - 1][simpul2 - 1] = 1;
        adjacencyMatrix[simpul2 - 1][simpul1 - 1] = 1;
    }

    // Menampilkan adjacency matrix
    cout << "\nAdjacency Matrix:\n";
    for (const auto& row : adjacencyMatrix) {
        for (const auto& val : row) {
            cout << val << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Output nya :

```
C:\Users\ASUS\OneDrive\Dokumen\tugas smt 3\Pemrograman Struktur Data 3\pertemuan14>

0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Adjacency Matrix:

3 4
2 4
1 3
1 2
Masukkan banyaknya simpul:
Masukkan jumlah sisi: 4
Masukkan jumlah simpul: 4
```

## 5. Kesimpulan

Berdasarkan praktikum yang dilakukan, dapat disimpulkan bahwa pemahaman tentang konsep graph sangat penting dalam mengatasi berbagai permasalahan yang melibatkan hubungan antar elemen. Graph merupakan struktur data yang terdiri dari node (verteks) dan edge (garis penghubung), dengan dua jenis utama yaitu graph berarah dan graph tak berarah. Graph berarah memiliki arah yang terdefinisi pada setiap edge, sementara pada graph tak berarah, edge menghubungkan dua node tanpa arah tertentu. Salah satu metode representasi graph yang banyak digunakan adalah adjacency matrix, yang memungkinkan kita untuk menggambarkan hubungan antar simpul dalam graph secara langsung. Dalam praktiknya, graph dapat digunakan untuk memodelkan berbagai hubungan dalam kehidupan nyata, seperti koneksi antar tempat, urutan pekerjaan dalam proyek, atau keterurutan mata pelajaran dalam kurikulum. Selain itu, konsep keterurutan linier yang terkait dengan graph memungkinkan kita untuk menyelesaikan masalah yang melibatkan urutan elemen, seperti dalam perencanaan jadwal atau pemrograman. Dengan demikian, implementasi graph memberikan solusi yang efektif dalam berbagai situasi yang melibatkan hubungan atau ketergantungan antar elemen.