

## Aturan Praktikum Struktur Data

1. **Akun GitHub:** Setiap praktikan wajib memiliki akun GitHub yang aktif dan digunakan selama praktikum berlangsung.
2. **Invite Collaborator:** Setiap praktikan diwajibkan untuk menambahkan collaborator di setiap repository
  - a. Asisten Praktikum: AndiniNH
  - b. Asisten Praktikum: 4ldiputra
3. **Repository Praktikum:** Setiap praktikan diwajibkan untuk membuat satu repository di GitHub yang akan digunakan untuk seluruh tugas dan laporan praktikum. Repository ini harus diatur dengan rapi dan sesuai dengan instruksi yang akan diberikan di lampiran.
4. **Penamaan Folder:** Penamaan folder dalam repository akan dibahas secara rinci di lampiran. Praktikan wajib mengikuti aturan penamaan yang telah ditentukan.

Nomor	Pertemuan	Penamaan
1	Pengalaman Bahasa C++ Bagian Pertama	01_Pengenalan_CPP_Bagian_1
2	Pengenalan Bahasa C++ Bagian Kedua	02_Pengenalan_CPP_Bagian_2
3	Abstract Data Type	03_Abstract_Data_Type
4	Single Linked List Bagian Pertama	04_Single_Linked_List_Bagian_1
5	Single Linked List Bagian Kedua	05_Single_Linked_List_Bagian_2
6	Double Linked List Bagian Pertama	06_Double_Linked_List_Bagian_1
7	Stack	07_Stack
8	Queue	08_Queue
9	Assessment Bagian Pertama	09_Assessment_Bagian_1
10	Tree Bagian Pertama	10_Tree_Bagian_1
11	Tree Bagian Kedua	11_Tree_Bagian_2
12	Asistensi Tugas Besar	12_Asistensi_Tugas_Besar
13	Multi Linked List	13_Multi_Linked_List
14	Graph	14_Graph
15	Assessment Bagian Kedua	15_Assessment_Bagian_2
16	Tugas Besar	16_Tugas_Besar

#### 5. Jam Praktikum:

- Jam masuk praktikum adalah **1 jam lebih lambat** dari jadwal yang tercantum. Sebagai contoh, jika jadwal praktikum adalah pukul 06.30 - 09.30, maka aturan praktikum akan diatur sebagai berikut:
  - **06.30 - 07.30:** Waktu ini digunakan untuk **Tugas Praktikum dan Laporan Praktikum** yang dilakukan di luar laboratorium.
  - **07.30 - 08.30:** Sesi ini mencakup **tutorial, diskusi, dan kasus problem-solving**. Kegiatan ini berlangsung di dalam laboratorium dengan alokasi waktu sebagai berikut:
    - **60 menit pertama:** Tugas terbimbing.
    - **60 menit kedua:** Tugas mandiri.

6. **Pengumpulan Tugasn Pendahuluan:** Tugas Pendahuluan (TP) wajib dikumpulkan melalui GitHub sesuai dengan format berikut:

**nama\_repo/nama\_pertemuan/TP\_Pertemuan\_Ke.md**

Sebagai contoh:

**STD\_Yudha\_Islalmi\_Sulistya\_XXXXXXXX/01\_Running\_Modul/TP\_01.md**

7. **Pengecekan Tugas Pendahuluan:** Pengumpulan laporan praktikum akan diperiksa **1 hari sebelum praktikum selanjutnya** dimulai. Pastikan tugas telah diunggah tepat waktu untuk menghindari sanksi.

8. Struktur Laporan Praktikum

1. Cover :

**LAPORAN PRAKTIKUM**

**Modul 14**

**“Graph”**



**Disusun Oleh:**

**KAFKA PUTRA RIYADI – 2311104041**

**Kelas:**

**SE 07-02**

**Dosen :**

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

**PROGRAM STUDI S1 SOFTWARE ENGINEERING  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY  
PURWOKERTO  
2024**

**2. Tujuan**

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

### 3. Landasan Teori

#### 1. Pengertian Graf

Graf adalah salah satu struktur data yang digunakan dalam ilmu komputer dan matematika diskret untuk merepresentasikan hubungan antar objek. Graf terdiri dari himpunan simpul (vertex atau node) dan himpunan sisi (edge) yang menghubungkan pasangan simpul. Graf sering digunakan untuk memodelkan berbagai masalah di dunia nyata, seperti jaringan komputer, hubungan sosial, peta jalan, dan sebagainya.

Secara formal, graf  $G$  dapat dinyatakan sebagai pasangan  $G=(V,E)$   $G=(V, E)$ , di mana:

- $V$ : himpunan simpul atau node.
- $E$ : himpunan sisi atau edge, yaitu pasangan  $(u,v)$  yang menghubungkan simpul  $u$  dan  $v$ .

#### 2. Jenis-Jenis Graf

Graf dapat dikategorikan ke dalam beberapa jenis berdasarkan sifat dan karakteristiknya:

- **Graf Tak Berarah (Undirected Graph):** Graf di mana setiap sisi tidak memiliki arah. Misalnya, jika  $(u,v)$  adalah sisi, maka  $(v,u)$  juga dianggap sama.
- **Graf Berarah (Directed Graph):** Graf di mana setiap sisi memiliki arah. Misalnya,  $(u,v)$  menunjukkan arah dari simpul  $u$  ke simpul  $v$ .
- **Graf Berbobot (Weighted Graph):** Graf di mana setiap sisi memiliki bobot atau nilai, yang sering digunakan untuk memodelkan jarak, biaya, atau kapasitas.
- **Graf Lengkap (Complete Graph):** Graf di mana setiap simpul terhubung langsung dengan semua simpul lainnya.
- **Graf Bipartit (Bipartite Graph):** Graf di mana himpunan simpul dapat dibagi menjadi dua kelompok terpisah, dan setiap sisi hanya menghubungkan simpul dari kelompok yang berbeda.

### 4. Guided

Code program:

```
#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;  
}
```

Outputannya:

```
PS C:\Users\sleme\AppData\Local\Microsoft\Windows\INetCache\IE\NTJEW48\output> & .\GuidedGraph  
.exe  
DFS traversal: A C G F B E D H  
BFS traversal: A C B G F E D H  
PS C:\Users\sleme\AppData\Local\Microsoft\Windows\INetCache\IE\NTJEW48\output> █
```

## 5. Unguided

Code Program Unguided1

```
#include <iostream>  
#include <vector>  
#include <iomanip>  
using namespace std;  
  
int main() {  
    int jumlah_simpul;  
    cout << "Silakan masukkan jumlah simpul: ";  
    cin >> jumlah_simpul;  
  
    vector<string> simpul(jumlah_simpul);  
    for (int i = 0; i < jumlah_simpul; i++) {  
        cout << "Simpul " << i + 1 << " : ";  
        cin >> simpul[i];  
    }  
  
    vector<vector<int>> bobot(jumlah_simpul, vector<int>(jumlah_simpul, 0));  
  
    cout << "Silakan masukkan bobot antar simpul" << endl;  
    for (int i = 0; i < jumlah_simpul; i++) {  
        for (int j = 0; j < jumlah_simpul; j++) {  
            cout << simpul[i] << " --> " << simpul[j] << " = ";  
            cin >> bobot[i][j];  
        }  
    }  
}
```



```

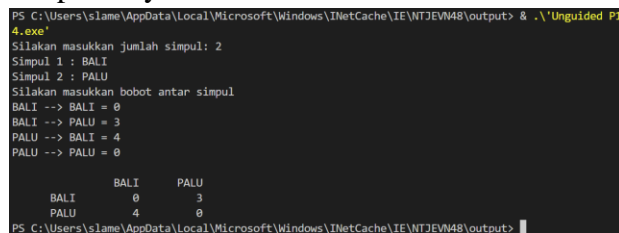
    cout << endl;
    cout << setw(10) << " ";
    for (const auto& s : simpul) {
        cout << setw(10) << s;
    }
    cout << endl;

    for (int i = 0; i < jumlah_simpul; i++) {
        cout << setw(10) << simpul[i];
        for (int j = 0; j < jumlah_simpul; j++) {
            cout << setw(10) << bobot[i][j];
        }
        cout << endl;
    }

    return 0;
}

```

Outputannya:



```

PS C:\Users\sleme\AppData\Local\Microsoft\Windows\INetCache\IE\NTJEVW48\output> & .\Unguided_P1
4.exe'
Silakan masukkan jumlah simpul: 2
Simpul 1 : BALI
Simpul 2 : PALU
Silakan masukkan bobot antar simpul
BALI --> BALI = 0
BALI --> PALU = 3
PALU --> BALI = 4
PALU --> PALU = 0

      BALI    PALU
BALI   0      3
PALU   4      0
PS C:\Users\sleme\AppData\Local\Microsoft\Windows\INetCache\IE\NTJEVW48\output>

```

Code Program Unguided2

```

#include <iostream>
#include <vector>

using namespace std;

int main() {
    int jumlahSimpul, jumlahSisi;
    cout << "Masukkan jumlah simpul: ";
    cin >> jumlahSimpul;
    cout << "Masukkan jumlah sisi: ";
    cin >> jumlahSisi;

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

```

```

    cout << "Masukkan pasangan simpul:\n";
    for (int i = 0; i < jumlahSisi; ++i) {
        int u, v;
        cin >> u >> v;

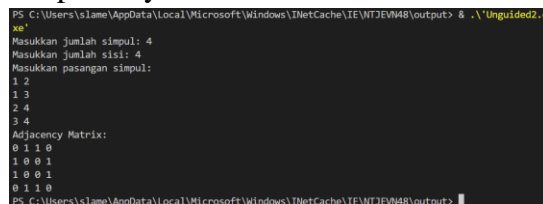
        adjacencyMatrix[u - 1][v - 1] = 1;
        adjacencyMatrix[v - 1][u - 1] = 1;
    }

    cout << "Adjacency Matrix:\n";
    for (int i = 0; i < jumlahSimpul; ++i) {
        for (int j = 0; j < jumlahSimpul; ++j) {
            cout << adjacencyMatrix[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}

```

Outputannya:



```

PS C:\Users\s\ame\AppData\Local\Microsoft\Windows\NetCache\IE\NT3E\W48\output> & .\Ungulded2.exe
xx
Masukkan jumlah simpul: 4
Masukkan jumlah sisi: 4
Masukkan pasangan simpul:
1 2
1 3
2 4
3 4
Adjacency Matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
PS C:\Users\s\ame\AppData\Local\Microsoft\Windows\NetCache\IE\NT3E\W48\output>

```

## 6. Kesimpulan

Graf adalah struktur data yang merepresentasikan hubungan antar objek melalui himpunan simpul (vertex) dan sisi (edge). Graf digunakan untuk memodelkan berbagai masalah, seperti jaringan, hubungan sosial, dan peta. Secara formal, graf dinyatakan sebagai pasangan  $G=(V,E)$   $G = (V, E)$ , dengan  $V$  sebagai himpunan simpul dan  $E$  sebagai himpunan sisi.

Graf memiliki beberapa jenis berdasarkan karakteristiknya:

1. Graf Tak Berarah: Sisi tidak memiliki arah, sehingga hubungan dua simpul bersifat timbal balik.
2. Graf Berarah: Sisi memiliki arah tertentu dari satu simpul ke simpul lain.
3. Graf Berbobot: Setiap sisi memiliki bobot yang merepresentasikan nilai tertentu, seperti jarak atau biaya.
4. Graf Lengkap: Semua simpul saling terhubung.
5. Graf Bipartit: Simpul terbagi dalam dua kelompok, dengan sisi hanya menghubungkan simpul dari kelompok yang berbeda.

Jenis-jenis ini memungkinkan graf digunakan untuk berbagai aplikasi sesuai kebutuhan