A. Tujuan

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

1. Mahasiswa mampu menjelaskan definisi dan konsep dari Graph
2. Mahasiswa mampu membuat representasi graph dalam matriks dan linked list
3. Mahasiswa mampu mengembangkan program untuk membuat graph (baik dalam representasi matriks maupun linked list)

B. Dasar Teori

2.1 Definisi dan Konsep Graph

Graph adalah kumpulan dari simpul dan busur yang secara matematis dinyatakan sebagai :

$$G = (V, E)$$

Dimana

G = Graph

V = Simpul/Vertex/Node/Titik

E = Edge/Busur/Arc/Ruas

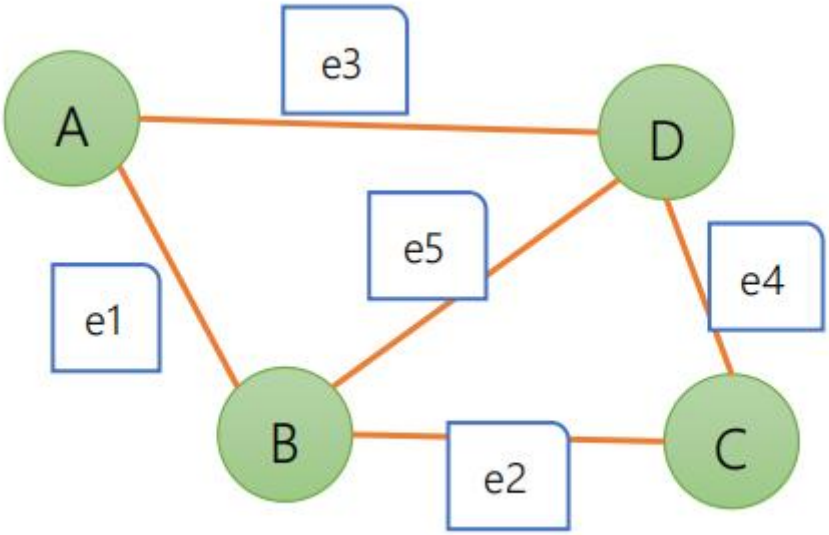
Contoh: Suatu graph G(E,V) dengan elemen-elemen sbb:

V mengandung 4 simpul : A, B, C, D

E mengandung 5 ruas :

e1 = (A,B) e2 = (B,C) e3 = (A,D)

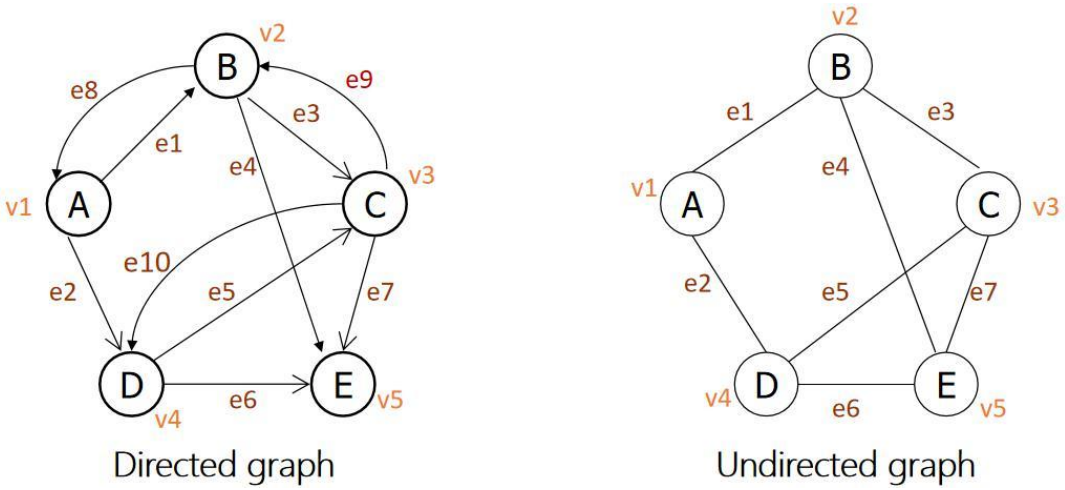
e4 = (C,D) e5 = (B,D)Maka dapat digambarkan :



Gambar 01. Contoh Graph

2.2 Jenis-Jenis Graph

2.2.1. Graph Berarah dan Tak Berarah



Gambar 02. Directed dan Undirected Graph

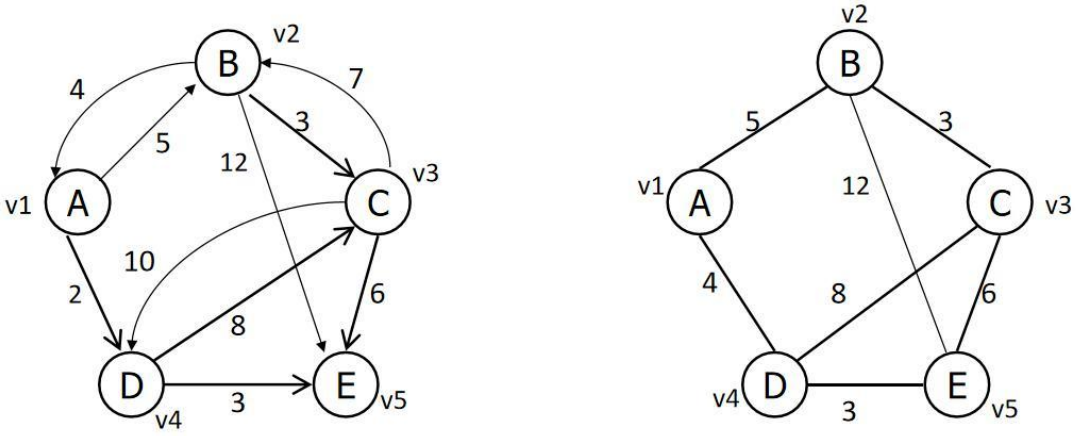
Graph berarah (directed graph) :

Urutan simpul mempunyai arti. Misal busur AB adalah e1 sedangkan busur BA adalah e8.

Graph tak berarah (undirected graph) :

Urutan simpul dalam sebuah busur tidak dipentingkan. Misal busur e1 dapat disebut busur AB atau BA

2.2.2. Grap Berbobot

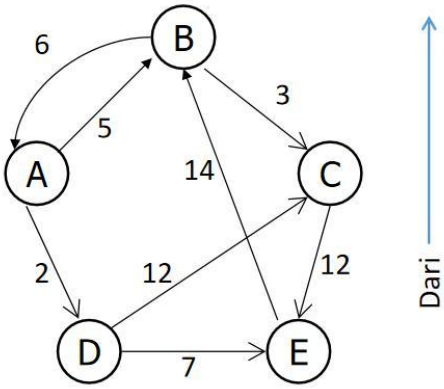


Gambar 03. Graph Berbobot

Jika setiap busur mempunyai nilai yang menyatakan hubungan antara 2 buah simpul, maka busur tersebut dinyatakan memiliki bobot.

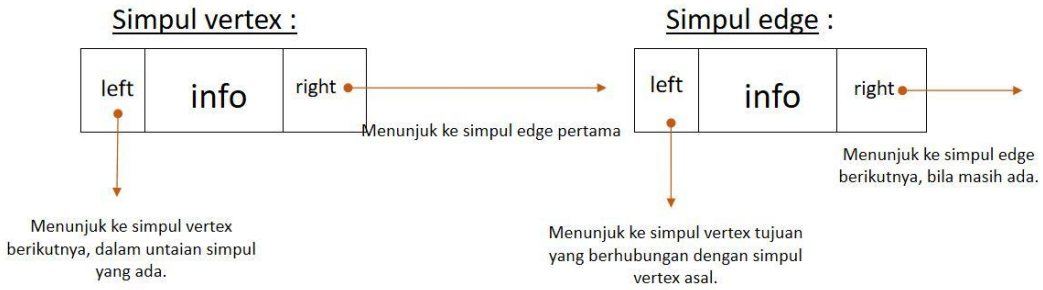
2.3 Representasi Graph

2.3.1 Matriks

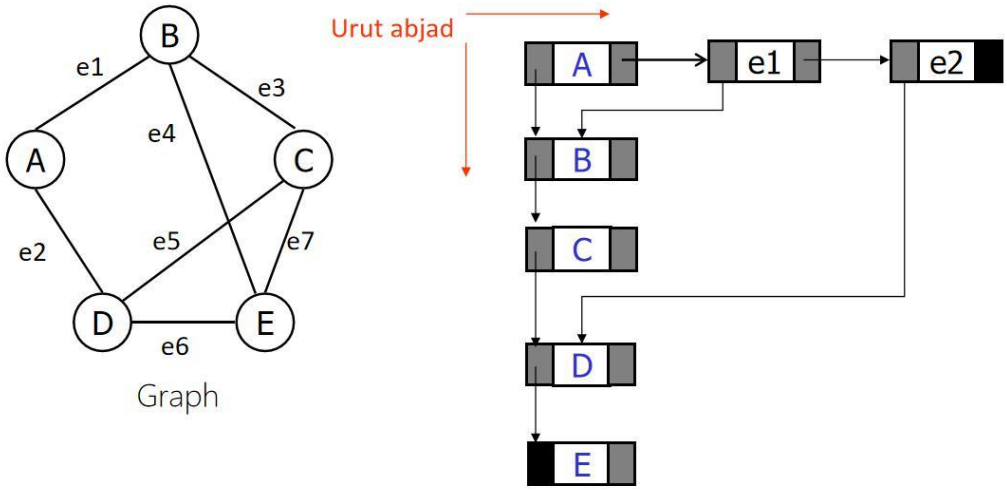


Dari \ Ke	A 0	B 1	C 2	D 3	E 4
A 0	0	5	0	2	0
B 1	6	0	3	0	0
C 2	0	0	0	0	9
D 3	0	0	12	0	7
E 4	0	14	0	0	0

2.3.2 Linked List



Yang perlu diperhatikan dalam membuat representasi graph dalam bentuk linked list adalah kita perlu membedakan antara simpul vertex dengan simpul edge. Simpul vertex menyatakan simpul atau vertex dan simpul edge menyatakan busur (hubungan antar simbol). Struktur keduanya bisa sama bisa juga berbeda tergantung kebutuhan, namun biasanya disamakan. Yang membedakan antara simpul vertex dengan simpul edge nantinya adalah anggapan terhadap simpul tersebut juga fungsinya masing-masing.



Gambar 04. Representasi Graph dengan Linked List

C. GUIDED

Silahkan dijalankan, amati, dan lakukan analisis.

Program pertama : Manual Input

Menghasilkan matrik representasi dari graph tersebut

```
#include <iostream>
#include <string>

using namespace std;

string simpul[7] = {"Bandung", "Bekasi", "Jakarta", "Purwokerto", "Semarang",
"Tasikmalaya", "Yogyakarta"};

int busur[7][7] = {
{0,5,0,15,0,0,0},
{6,0,0,0,5,0,0},
{7,8,0,0,0,0,0},
{0,0,0,0,7,0,3},
{0,0,23,0,0,10,8},
{5,0,0,4,0,0,0},
{0,0,0,4,9,0,0}};

void tampilGraph(){
    if(simpul && busur){
        for(int baris = 0; baris < 7; baris++){
            cout<<simpul[baris]<<" : ";
            for(int kolom = 0; kolom < 7;
            kolom++){
                if(busur[baris][kolom] != 0){
                    cout<<simpul[kolom]<<"("<<busur[baris][kolom]<<")"<<"
";
                }
```

```
            }
            cout<<endl;
        }
    }
}

int main(){
    tampilGraph();
    return 0;
}
```

Program kedua: Input dari User

Menghasilkan matrik representasi dari graph tersebut

```
#include <iostream>
#include <string>

using namespace std;

int jumlahSimpul = 5;
string *dataSimpul;
int **dataBusur;
bool cekMatrik = false;

void buatMatriks(){
    dataSimpul = new string[jumlahSimpul];
    //Membuat simpul dengan jumlah yang dimasukkan user
    dataBusur = new int*[jumlahSimpul];
    dataBusur[0] = new int[jumlahSimpul * jumlahSimpul];
    for(int i = 1; i < jumlahSimpul; i++){
        dataBusur[i] = dataBusur[i-1] + jumlahSimpul;
    }
    //Membuat matrik 2 dimensi dengan ukuran sesuai masukkan user
```

```
    cout<<"Silahkan masukkan nama simpul "<<endl;
    for(int i = 0; i < jumlahSimpul; i++){
        cout<<"Simpul "<<i+1<<" : ";
        cin>>dataSimpul[i];
    }
    //Proses input data simpul

    cout<<"Silahkan masukkan bobot antar simpul "<<endl;
    for(int baris = 0; baris < jumlahSimpul; baris++){
        for(int kolom = 0; kolom < jumlahSimpul; kolom++){
            cout<<dataSimpul[baris]<<" --> "<<dataSimpul[kolom]<<" : ";
            cin>> dataBusur[baris][kolom];
        }
    }
    //Prose input data matriks (busur)

    cekMatrik = true;
    //Atur keberadaan matrik menjadi ada/true
}

void tampilMatriks(){
    if(cekMatrik){
        for(int i = 0; i < jumlahSimpul; i++){
            cout<<dataSimpul[i]<<" ";
        }
        cout<<endl;
        for(int baris = 0; baris < jumlahSimpul; baris++){
            for(int kolom = 0; kolom < jumlahSimpul; kolom++){
                cout<<dataBusur[baris][kolom]<<" ";
            }cout<<endl;
```

```
            }
        }else{
            cout<<"Tidak ada matriks"<<endl;
        }
}

int main(){
    cout<<"Silahkan masukkan jumlah simpul : ";
    cin>>jumlahSimpul;
    buatMatriks();
    tampilMatriks();
    return 0;
}
```

Program ketiga: Input dari User

Menghasilkan list representasi dari graph tersebut

```
#include <iostream>
#include <string>

using namespace std;

int jumlahSimpul = 5;
string *dataSimpul;
int **dataBusur;
bool cekMatrik = false;

struct graph{
    graph *kanan;
    string data;
    graph *kiri;
};
```

```
graph *simpul;
graph *busur;
graph *awal;
graph *akhir;
graph **alamat;
graph *helperA;
graph *helperB;

void inisiasi(){
	awal = NULL;
	akhir = NULL;
}

bool graphKosong(){
	if(awal == NULL && akhir == NULL){
		return true;
	}else{
		return false;
	}
}

void buatMatriks(){
dataSimpul = new string[jumlahSimpul];
dataBusur = new int*[jumlahSimpul];
dataBusur[0] = new int[jumlahSimpul * jumlahSimpul];
for(int i = 1; i < jumlahSimpul; i++){
	dataBusur[i] = dataBusur[i-1] + jumlahSimpul;
}

cout<<"Silahkan masukkan nama simpul "<<endl;
for(int i = 0; i < jumlahSimpul; i++){
```

```
        cout<<"Kota "<<i+1<<" : ";
        cin>>dataSimpul[i];
}

cout<<"Silahkan masukkan bobot antar simpul "<<endl;
        for(int baris = 0; baris < jumlahSimpul; baris++){
                for(int kolom = 0; kolom < jumlahSimpul; kolom++){
                        cout<<dataSimpul[baris]<<" --> "<<dataSimpul[kolom]<<" : ";
                        cin>> dataBusur[baris][kolom];
                }
        }
        cekMatrik = true;
}

void buatSimpulGraph(){
        alamat = new graph*[jumlahSimpul];
        // Membuat pointer alamat sebanyak jumlah simpul
        buatMatriks();
        // Membuat representasi graph berupa matriks dengan memanggil fungsi
buatMatriks()
        for(int i = 0; i < jumlahSimpul; i++){
                if(graphKosong()){
                        simpul = new graph; simpul->data = dataSimpul[i];
                        simpul->kanan = NULL;
                        simpul->kiri = NULL;
                        awal = simpul;
                        akhir = simpul;
                        alamat[i] = awal;
                        // Simpan alamat simpul
                }else{
                        simpul = new graph;
```

```
                simpul->data = dataSimpul[i];
                akhir->kiri = simpul;
                akhir = simpul;
                simpul->kiri = NULL;
                simpul->kanan = NULL;
                alamat[i] = akhir;
            }
        }

        helperA = awal;
        for(int baris = 0; baris < jumlahSimpul; baris++){
            helperB = helperA;
            for(int kolom = 0; kolom < jumlahSimpul; kolom++){
                if(dataBusur[baris][kolom] != 0){
                    simpul = new graph;
                    simpul->data = to_string(dataBusur[baris][kolom]);
                    helperB->kanan = simpul;
                    simpul->kiri = alamat[kolom];
                    simpul->kanan = NULL;
                    helperB = simpul;
                }
            }
            helperA = helperA->kiri;
        }
    }

    void tampilGraph(){
        if(!graphKosong()){
            helperA = awal;
            while(helperA != NULL){
                cout<<helperA->data<<" : ";
```

```
            helperB = helperA->kanan;
            while(helperB != NULL){
                cout<<helperB->kiri->data<<" : "<<helperB->data<<" ";
                helperB = helperB->kanan;
            }
            cout<<endl;
            helperA = helperA->kiri;
        }
    }else{
        cout<<"Graph kosong...!!!"<<endl;
    }
}

int main(){
    inisiasi();
    cout<<"Silahkan masukkan jumlah kota : ";
    cin>>jumlahSimpul;
    buatSimpulGraph();
    tampilGraph();
    return 0;
}
```

Program keempat: Manual Input

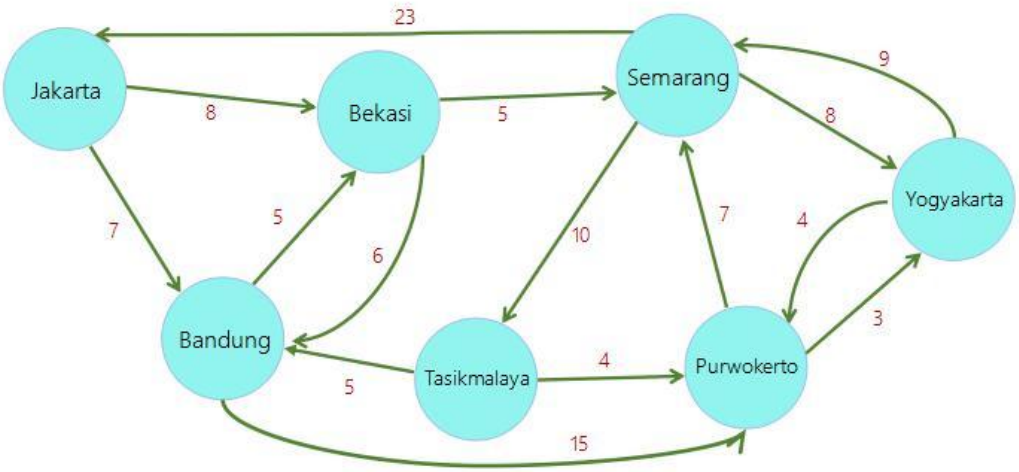
Menghasilkan list representasi dari graph tersebut

```
#include <stdio.h>
#include <stdlib.h>
#define N 6 //misal maksimum node adalah 6
// Struktur data untuk menyimpan adjacency list dari node pada graph
struct Node{
    int dest;
    struct Node* next;
};
```

```
typedef struct Node *ptrNode;
//Struktur data untuk menyimpan onjek graph
struct Graph{
// array pointer ke node untuk representasi adjacency list
ptrNode head[N];
};
typedef struct Graph *ptrGraph;
// Struktur data untuk menyimpan edge graph
struct Edge {
	int src, dest;
};
// Fungsi untuk membuat adjacency list dari edge tertentu
ptrGraph createGraph (struct Edge edges[], int n){
// alokasi memori untuk menyimpan struktur data graph
	ptrGraph graph = (ptrGraph)malloc(sizeof(struct Graph));
	// inisialiasi semua pointer head ke null
	for (int i = 0; i < N; i++) {
		graph->head[i] = NULL;
	}
	// menambahkan edge satu demi satu
	for (int i = 0; i < n; i++){
		// ambil source dan destination dari node
		int src = edges[i].src;
		int dest = edges[i].dest;
		// buat node baru dari src ke dest
		ptrNode newNode = (ptrNode)malloc(sizeof(struct Node));
		newNode->dest = dest;
		// point node baru ke head
		newNode->next = graph->head [src];
		// point head ke node baru
		graph->head [src] = newNode;
```

```
    }
    return graph;
}
// Fungsi print representasi adjacency list
void printGraph(ptrGraph graph) {
    int i;
    for (i = 0; i < N; i++){
    // print node dan semua yang terhubung
        ptrNode ptr = graph->head[i];
        while (ptr != NULL) {
            printf("(%d -> %d)\t", i, ptr->dest);
            ptr = ptr->next;
        }
        printf("\n");
    }
}
int main() {
//input array pasangan dari x ke y
struct Edge edges[] =
{{ 0, 1}, {1, 2}, {2, 0}, {2, 1}, {3, 2}, {4, 5}, {5, 4 }};
// menghitung jumlah edge
int n = sizeof(edges) /sizeof(edges[0]);
// membuat graph
ptrGraph graph = createGraph (edges, n);
// print graph
printGraph (graph);
}
```

D. UNGUIDED



Gambar 05. Ilustrasi Graph

1. Buatlah repsentasi graphnya dengan matrik(tabel) dan list(gambar).
2. Sebutkan dan jelaskan apakah graph ini termasuk Undirected Graph/ Directed Graph (Digraph)/ Weigth Graph?