

# **LAPORAN PRAKTIKUM**

**LAPORAN PRAKTIKUM INI DITUJUKAN UNTUK MEMENUHI**

**UJIAN AKHIR SEMESTER (UAS)**

**MATA KULIAH STRUKTUR DATA**

**Dosen : I Gde Agung Sri Sidhimantra, S.Kom., M.Kom.**



**Disusun Oleh:**

**Revina Aurigha Firdaus**

**21091397003**

**PROGRAM STUDI D IV MANAJEMEN INFORMATIKA**

**FAKULTAS VOKASI**

**UNIVERSITAS NEGERI SURABAYA**

**2021/2022**

## A. Tujuan

Setelah mempraktekkan ini diharapkan mahasiswa akan mampu :

- Memahami mengenai pengertian Graph, representasikan Graph dan istilah-istilah yang terdapat pada Graph
- Memahami mengenai Dijkstra

## B. Penjelasan & Jawaban Soal No 1

### 1. Pengertian Graph

**Graph** adalah kumpulan node (simpul) di dalam bidang dua dimensi yang dihubungkan dengan sekumpulan garis (sisi). Graph dapat digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari graph adalah dengan menyatakan objek sebagai node, bulatan atau titik (vertex), sedangkan hubungan antara objek dinyatakan dengan garis (edge).

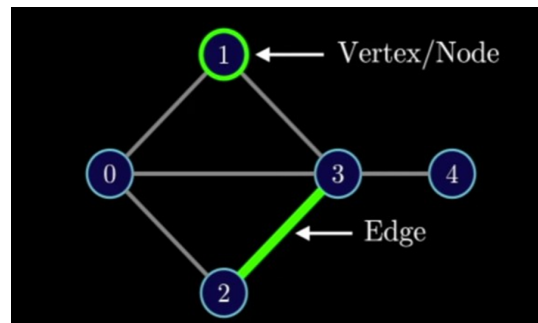
$$G = (V, E)$$

Dengan :

$G$  = Graph

$V$  = Simpul atau vertex, atau node, atau Titik

$E$  = Busur atau edge



### 2. Istilah-istilah dalam graph

#### ○ Incident

Jika  $e$  merupakan busur dengan simpul-simpulnya adalah  $v$  dan  $w$  yang ditulis  $e = (v,w)$ , maka  $v$  dan  $w$  disebut “terletak” pada  $e$  dan  $e$  disebut incident dengan  $v$  dan  $w$ .

#### ○ Degree(Indegree dan Outdegree)

Sebuah simpul adalah jumlah busur yang incident dengan simpul tersebut. Indegree sebuah simpul pada graph berarah adalah jumlah

busur yang kepalanya incident dengan simpul tersebut, atau jumlah busur yang “masuk” atau menuju simpul tersebut. Outdegree sebuah simpul pada graph berarah adalah jumlah busur yang ekornya incident dengan simpul tersebut, atau jumlah busur yang “keluar” atau berasal dari simpul tersebut.

- **Adjacent**

Pada graph tidak berarah, 2 buah simpul disebut adjacent bila ada busur yang menghubungkan kedua simpul tersebut.

- **Successor dan Predecessor**

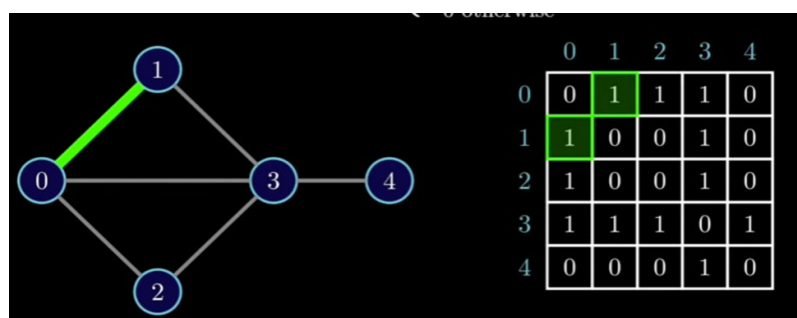
Pada graph berarah, bila simpul  $v$  adjacent dengan simpul  $w$ , maka simpul  $v$  adalah successor simpul  $w$ , dan simpul  $w$  adalah predecessor dari simpul  $v$ .

- **Path**

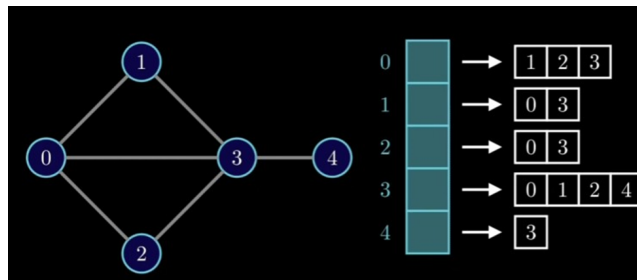
Sebuah path adalah serangkaian simpul-simpul yang berbeda, yang adjacent secara berturut-turut dari simpul satu ke simpul berikutnya.

### 3. Representasi Graph

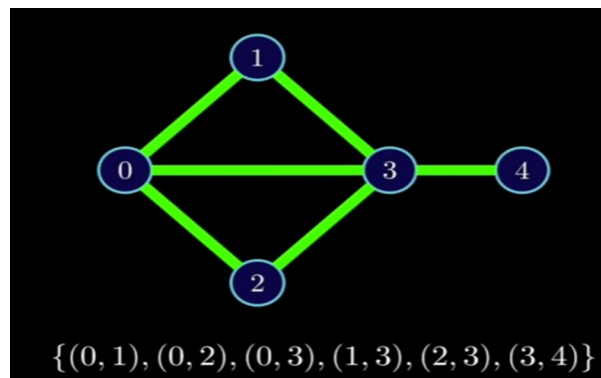
- **Representasi Graph dalam bentuk Adjacency matrix**



- **Representasi Graph dalam bentuk Adjacency List**



- Representasi Graph dalam bentuk Edge Set



#### 4. Latihan Soal 1

**Bahasa pemrograman** : C++

**Compiler** : DevC++

**Input :**

nts\Semester 2\TUGAS BESAR\SD\fiks\A003\_RevinaAurighaFirdaus\_UASNo1.cpp -

Project Execute Tools AStyle Window Help

```
1 //Revina Aurigha Firdaus 21091397003
2
3 #include<iostream>
4 #define Max 100
5
6 using namespace std;
7
8 int adjMat[Max][Max];
9
10 // instalasi matriks ke nol
11 void initializeMat(int v)
12 {
13     for(int i = 0; i < v; i++)
14     {
15         for(int j = 0; j < v; j++)
16         {
17             adjMat[i][j] = 0;
18         }
19     }
20 }
21 // menambahkan edges
22 void addEdge(int u, int v, int w)
23 {
24     adjMat[u][v] = w;
25     adjMat[v][u] = w;
26 }
27 // mencetak matriks
28 void displayMat(int v)
29 {
30     for (int i = 1; i <= v; i++)
31     {
32         cout << "\t";
33
34         for (int j = 1; j <= v; j++)
35         {
36             cout << adjMat[i][j] << "\t";
```

```
36             cout << adjMat[i][j] << "\t";
37         }
38
39         cout << endl;
40     }
41 }
42 int main()
43 {
44
45     int vertice = 4;
46
47     initializeMat(vertice);
48
49     addEdge(1,2,5);
50     addEdge(2,3,1);
51     addEdge(4,1,3);
52     addEdge(2,4,1);
53     addEdge(3,1,1);
54     displayMat(vertice);
55
56     return 0;
57 }
```

Output :

```
C:\Users\Asuspro\Documents\Semester 2\TUGAS BESAR\SD\file\A003_RevinaAurighaFirdaus_UASNo1.exe

0      5      1      3
5      0      1      1
1      1      0      0
3      1      0      0

-----
Process exited after 0.06437 seconds with return value 0
Press any key to continue . . .
```

## C. Penjelasan & Jawaban Soal No 2

### 1. Pengertian Algoritma Dijkstra

Algoritma yang dinamai menurut penemunya, Edsger Dijkstra pada tahun 1959, adalah sebuah algoritma rakus(greedy algorithm) dalam memecahkan permasalahan jarak terpendek untuk sebuah graf berarah (directed graph) ataupun graph tidak berarah (undirected graph) . Dijkstra adalah algoritma yang digunakan untuk mencari lintasan terpendek pada sebuah graf berarah. Contoh penerapan algoritma djikstra adalah lintasan terpendek yang menghubungkan antara dua kota berlainan tertentu (Single-source Single Destination Shortest Path Problems). Cara kerja algoritma Dijkstra memakai strategi greedy, di mana pada setiap langkah dipilih sisi dengan bobot terkecil yang menghubungkan sebuah simpul yang sudah terpilih dengan simpul lain yang belum terpilih. Algoritma Dijkstra membutuhkan parameter tempat asal, dan tempat tujuan. Hasil akhir dari algoritma ini adalah jarak terpendek dari tempat asal ke tempat tujuan beserta rutenya.

### 2. Latihan Soal 2

**Bahasa pemrograman** : C++




**Compiler** : DevC++

**Input :**

A003\_RevinaurighaFirdaus\_UASNo1.cpp

A003\_RevinaurighaFirdaus\_UASNo2.cpp

```
1 // Revina Aurigha Firdaus 21091397003
2
3 #include <iostream>
4 #include <conio.h>
5 #include <string.h>
6 using namespace std;
7 int main()
8 {
9     char kota1,kota2,kota3,kota4,kota5;
10    int jumlah,panjang, hasil1,hasil2,hasil3,hasil4,hasil5,hasil6,hasil7;
11
12    // memasukkan jumlah kota di kerajaan Britan
13    cout<<"* Jumlah kota yang terdapat di kerajaan Britan : "<< endl;
14    cin>>jumlah;
15
16    // deklarasi vertex
17    // menampilkan masing-masing vertex
18    // input nama kota
19    cout<<"Kota Pertama : ";
20    cin>>kota1;
21    cout<<"Kota Kedua : ";
22    cin>>kota2;
23    cout<<"Kota Ketiga : ";
24    cin>>kota3;
25    cout<<"Kota Keempat : ";
26    cin>>kota4;
27    cout<<"Kota kelima : ";
28    cin>>kota5;
29
30    cout<<endl;
31
32    // deklarasi edge
33    // menampilkan setiap edge yang terjadi
34    cout<<"* Sisi-sisinya adalah : "<<endl<<endl;
35    cout<<kota1<<kota2<<",";
36    cout<<kota1<<kota4<<",";
```

 Compile Log  Debug  Find Results

Sel: 0

Lines: 100

Length: 2966

Insert

Done parsing in 0,031 seconds



ents\Semester 2\TUGAS BESAR\SD\file\A003\_RevinaAurighaFirdaus\_UASNo2.cpp - [Executing] - Dev-C++ 5.11

Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Rel

A003\_RevinaAurighaFirdaus\_UASNo1.cpp A003\_RevinaAurighaFirdaus\_UASNo2.cpp

```
37 cout<<kota1<<kota5<<" ";
38 cout<<kota2<<kota3<<" ";
39 cout<<kota3<<kota5<<" ";
40 cout<<kota3<<kota4<<" ";
41 cout<<kota4<<kota5<<endl<<endl;
42
43 // deklarasi weight
44 // menampilkan panjang jalan yang menghubungkan vertex
45 cout<<"* Panjang jalan antar kota : "<<endl;
46 cout<<"panjang "<<kota1<<" ke "<<kota2<<" : "; cin>> hasil1;
47 cout<<"panjang "<<kota1<<" ke "<<kota4<<" : "; cin>> hasil2;
48 cout<<"panjang "<<kota1<<" ke "<<kota5<<" : "; cin>> hasil3;
49 cout<<"panjang "<<kota2<<" ke "<<kota3<<" : "; cin>> hasil4;
50 cout<<"panjang "<<kota3<<" ke "<<kota5<<" : "; cin>> hasil5;
51 cout<<"panjang "<<kota3<<" ke "<<kota4<<" : "; cin>> hasil6;
52 cout<<"panjang "<<kota4<<" ke "<<kota5<<" : "; cin>> hasil7;
53
54 cout<<endl;
55
56 // deklarasi adjecnt
57 // menampilkan jalan yang menghubungkan kedua simpul (x,y,z)
58 cout<<"* seluruh jalan yang ada dalam kerajaan britan dan panjang jalannya : "<<endl;
59 cout<<"("<<kota1<<","<<kota2<<","<<hasil1<<") ";
60 cout<<"("<<kota1<<","<<kota4<<","<<hasil2<<") ";
61 cout<<"("<<kota1<<","<<kota5<<","<<hasil3<<") ";
62 cout<<"("<<kota2<<","<<kota3<<","<<hasil4<<") ";
63 cout<<"("<<kota3<<","<<kota5<<","<<hasil5<<") ";
64 cout<<"("<<kota3<<","<<kota4<<","<<hasil6<<") ";
65 cout<<"("<<kota4<<","<<kota5<<","<<hasil7<<") ";
66
67 cout<<endl<<endl;
68
69 // hasil yang dikeluarkan
70 // menampilkan kota tempat pedagang berada
71 cout<<"* kota tempat pedagang sekarang berada : "<<endl<<endl;
72 cout<<kota1;
```

Compile Log Debug Find Results

Sel: 0 Lines: 100 Length: 2966 Insert Done parsing in 0,031 seconds

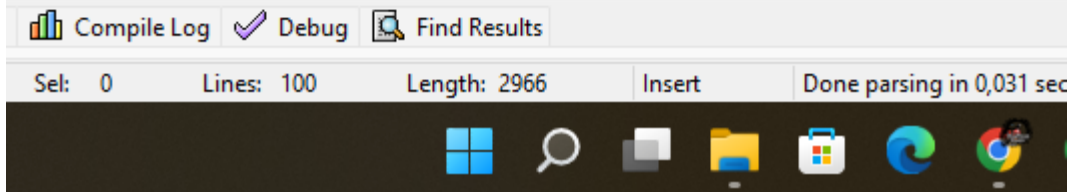




```

73
74     cout<<endl<<endl;
75
76     // menampilkan kota yang diserang oleh naga
77     cout<<"* kota yang diserang naga : "<<endl<<endl;
78     cout<<kota3;
79
80     cout<<endl<<endl;
81
82     // menampilkan kota yang terdapat kastil
83     cout<<"* kota yang memiliki kastil : "<<endl<<endl;
84     cout<<kota5;
85
86     cout<<endl<<endl;
87
88     // menampilkan jalan tecepat mencapai istana
89     cout<<"* jalur yang paling cepat ditempuh : "<<endl<<endl;
90     cout<<kota1<<"- "<<kota4<<"- "<<kota5<<endl;
91
92     cout<<endl<<endl;
93
94     cout<<"* dengan jarak : "<<endl<<endl;
95     cout<<hasil2+hasil7<<endl<<endl;
96
97
98     getch();
99     return 0;
100 }

```



## Output

```

C:\Users\Asuspro\Documents\Semester 2\TUGAS BESAR\SD\fiks\A003_RevinAurighaFirdaus_UASNo2.exe
* Jumlah kota yang terdapat di kerajaan Britan :
5
Kota Pertama : 1
Kota Kedua : 2
Kota Ketiga : 3
Kota Keempat : 4
Kota kelima : 5

* Sisi-sisinya adalah :
12,14,15,23,35,34,45

* Panjang jalan antar kota :
panjang 1 ke 2: 12
panjang 1 ke 4: 11
panjang 1 ke 5: 30
panjang 2 ke 3: 14
panjang 3 ke 5: 5
panjang 3 ke 4: 15
panjang 4 ke 5: 10

* seluruh jalan yang ada dalam kerajaan britan dan panjang jalannya :
(1,2,12) (1,4,11) (1,5,30) (2,3,14) (3,5,5) (3,4,15) (4,5,10)

```

\* kota tempat pedagang sekarang berada :

1

\* kota yang diserang naga :

3

\* kota yang memiliki kastil :

5

\* jalur yang paling cepat ditempuh :

1-4-5

\* dengan jarak :

21