LAPORAN STRUKTUR DATA

"PROGRAM GRAF TIDAK BERARAH BERBOBOT"

Disusun oleh :

Muhammad Fikri Ramadhana (21091397033)

KELAS 2021 A

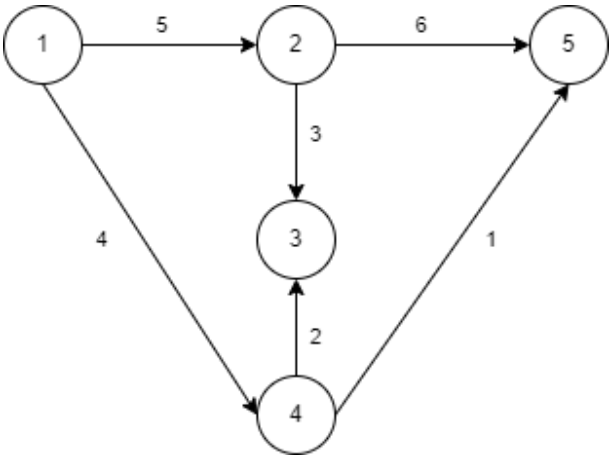
PRODI D4 MANAJEMEN INFORMATIKA

UNIVERSITAS NEGERI SURABAYA

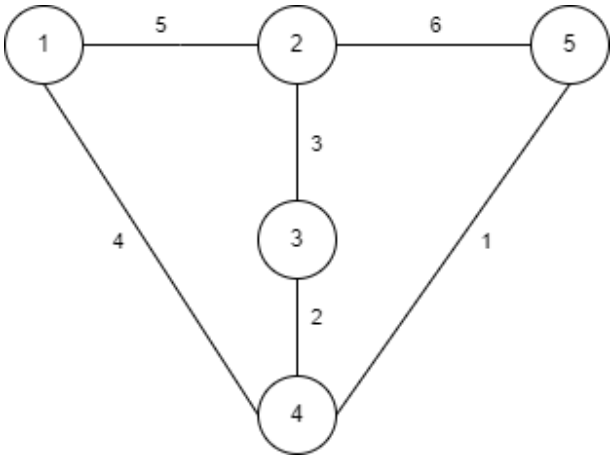
2021/2022

A. Hasil Analisis dan Identifikasi

Graph merupakan sekumpulan dari simpul (*vertices*) dan busur (*edge*) yang dapat dinotasikan dengan G = (V,E). Dengan sebuah garis (*adjacent*) yang menghubungi antara kedua simpul yang dinyatakan sebagai *u* dan *v*. Pada graph berarah (*directed graph*), setiap garis memiliki arah dari simpul *u* mengarah ke simpul *v* dengan notasi u - > v. Kemudian pada graph tak berarah (*undirected graph*), setiap garis tidak memiliki arah yang mentukan diantara garis kedua simpul *u* dan *v* dengan notasi *u* < - > *v*. Implementasi dari tipe graph yaitu dapat menggambarkan pemetaan dari jalur jaringan dan peta jalan.



Gambar 1.1 Graph berarah



Gambar 1.2 Graph tak berarah

Berdasarkan gambar graph berarah diatas simpul 1 memiliki arah ke simpul 2 dengan bobot 5, kemudian simpul 4 ke simpul 5 dengan bobot 1 dan lain-lainnya. Sedangkan pada gambar graph tidak berarah dapat dilihat jika simpul 1 memiliki sebuah garis (*adjacent*) ke simpul 2 dengan bobot 5. Namun program yang dibuat berikut menggunakan konsep *undirected weighted graph* dengan representasi *adjency matriks* input nilai simpul (*vertices*), busur (*edge*), dan bobot (*weight*). Program ini juga disajikan cerita dengan sinopsis sebagai berikut :

“Terdapat seorang pedagang Rahmad, Rahmad setiap bulan berkeliling di kerajaan Britan untuk berdagang. Tetapi suatu hari, pedagang ini mendapat berita bahwa ada seekor naga yang sedang menyerang salah satu kota. Jadi pedagang ini bergegas menuju ke istana untuk memberitahu raja bahwa ada kota yang sedang diserang sambil menghindari kota yang diserang tersebut. Sehingga raja bisa mengirimkan pasukan untuk menyerang kota tersebut.”

Uas SD 2.cpp UAS_STRUKDAT_!.cpp

```cpp
//Program undirected graph cerita UAS SD
#include <iostream>
#include <stdio.h>
#include <conio.h>
#define N 100
using namespace std;

int main()
{
    //inialisasi variabel data
    int graph[N][N];
    int vertex, edge;
    int jumlah,vertex_1,vertex_2,bobot,naga,pedagang,kastil;

    printf("=========================\n");
    printf(" Program Undirected Graph \n");
    printf("=========================\n\n");

    printf("*Invasi Naga yang Menyerang Kota*\n\n");

    //input data vertex dan edge yang diinginkan
    printf("Masukkan vertex & edge (jumlah kota britain) : \n");
    scanf("%d%d", &vertex, &edge);

    //proses untuk pengulangan graph
    for(int i=0;i<vertex;++i)
    {
        for(int j=0;j<vertex;++j)
        {
            graph[i][j]=0;
        }
    }
```

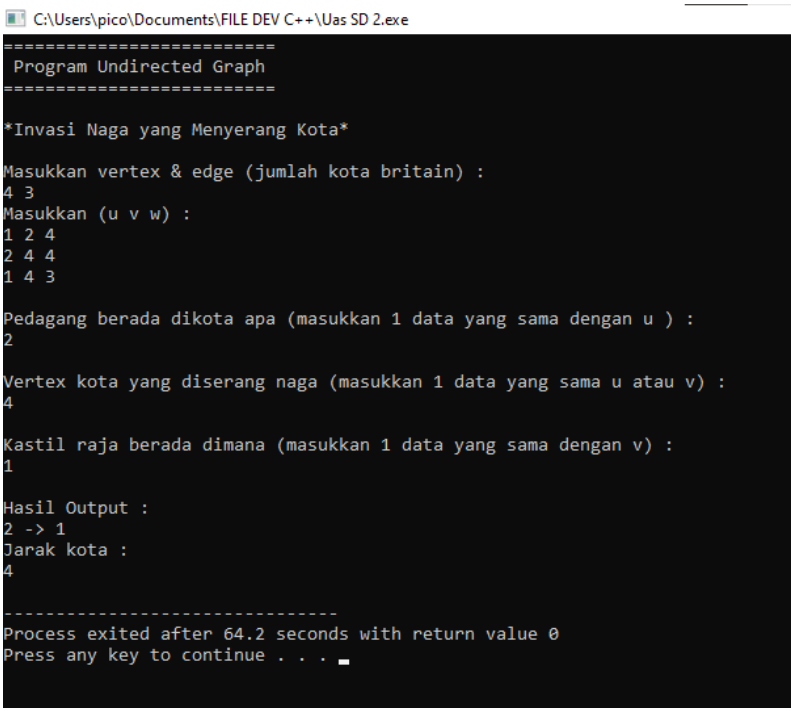
Uas SD 2.cpp UAS_STRUKDAT_!.cpp

```cpp
        {
            graph[i][j]=0;
        }
    }

    //inilisasi vertex_1 = u, vertex_2 = v, dan bobot = w kemudian input datanya
    printf("Masukkan (u v w) : \n");
    //proses pengulangan berdasarkan edge yang dimasukkan
    for(int i=0;i<edge;++i)
    {
        scanf("%d%d%d", &vertex_1, &vertex_2, &bobot);
        //proses untuk fungsi undirected graph
        graph[vertex_1][vertex_2] = graph[vertex_2][vertex_1] = bobot;
    }

    //input lokasi pedagang berada pada vertex berapa
    printf("\n");
    printf("Pedagang berada dikota apa (masukkan 1 data yang sama dengan u ) :\n");
    scanf("%d", &pedagang);

    //input lokasi kota yang diserang naga
    printf("\n");
    printf("Vertex kota yang diserang naga (masukkan 1 data yang sama u atau v) : \n");
    scanf("%d", &naga);

    //input lokasi kastil
    printf("\n");
    printf("Kastil raja berada dimana (masukkan 1 data yang sama dengan v) : \n");
    scanf("%d", &kastil);

    //mencetak hasil output dari jalur tercepat pedagang menuju kastil
    printf("\n");
```

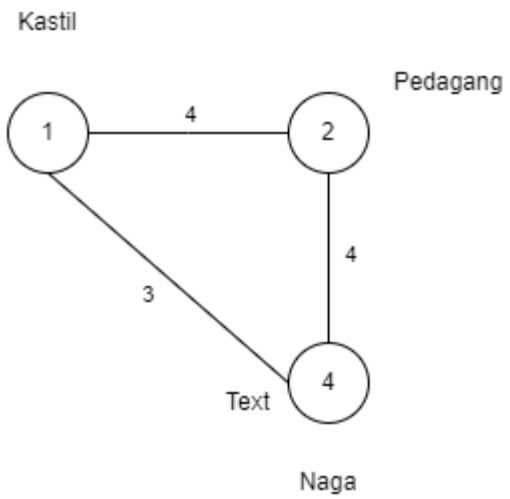
Uas SD 2.cpp UAS_STRUKDAT_!.cpp

```cpp
        graph[vertex_1][vertex_2] = graph[vertex_2][vertex_1] = bobot;
    }

    //input lokasi pedagang berada pada vertex berapa
    printf("\n");
    printf("Pedagang berada dikota apa (masukkan 1 data yang sama dengan u ) :\n");
    scanf("%d", &pedagang);

    //input lokasi kota yang diserang naga
    printf("\n");
    printf("Vertex kota yang diserang naga (masukkan 1 data yang sama u atau v) : \n");
    scanf("%d", &naga);

    //input lokasi kastil
    printf("\n");
    printf("Kastil raja berada dimana (masukkan 1 data yang sama dengan v) : \n");
    scanf("%d", &kastil);

    //mencetak hasil output dari jalur tercepat pedagang menuju kastil
    printf("\n");
    printf("Hasil Output : \n");
    printf("%d -> %d ", pedagang, kastil);

    //input total jarak kota pedagan dengan kastil
    printf("\n");
    printf("Jarak kota : \n");
    scanf("%d", graph[bobot]);

    return 0;
    getch();
}
```

Hasil Output :



Pada hasil kodinganya mengenai *undirected weighted graph* dengan representasi *adjency matriks* input nilai simpul (*vertices*), busur (*edge*), dan bobot (*weight*) dimulai dari deklarasi dari file header yang diperlukan untuk menjalankan sistem operasi, kemudian inialisasi variabel data. Langkah selanjutnya membuat proses input vertex dan edge (sebagai jumlah kerajaan Britain) yang selanjutnya akan dilakukan perulangan dengan memasukkan graph[i][j] yang nantinya digunakan sebagai tempat menyimpan data *u* dan *v*. Kemudian membuat proses fungsi input nilai *u*, *v*, dan *w* (vertex_1, vertex_2, dan bobot) yang dimasukkan dalam perulangan, langkah selanjutnya membuat sistem operasi *undirected weighted graph* dengan cara graph[vertex_1][vertex_2] = graph[vertex_2][vertex_1] = bobot yang dimasukkan dalam perulangan fungsi input nilai *u*, *v*, dan *w*. Selanjutnya membuat fungsi input nilai untuk pedagang yang sedang berada divertex ke berapa. Kemudian membuat fungsi input nilai lokasi kota yang sedang diserang oleh naga. Langkah berikutnya membuat fungsi input nilai lokasi kastil berada di vertex berapa. Kemudian membuat proses cetak hasil dari fungsi dan sistem operasi yang telah dijalankan.



Gambar diatas dapat digunakan sebagai penggambaran mengenai graph yang dibentuk berdasarkan kodingan dengan cerita yang dimasukkan. Kodingan tersebut dijalankan menggunakan algoritma dijkstra yang digunakan sebagai menemukan jarak terpendek dalam sebuah graph. Pada gambar terlihat bahwa lokasi pedagang berada di kota 2, lokasi naga di kota 4, dan lokasi kastil berada di kota 1. Sehingga dalam menyelesaikan kasus dalam cerita jika pedagang berada di kota 2 yang ingin pergi ke kastil namun harus menghindari kota yang diserang naga, maka pedagang Rahmad harus pergi ke kota 1 dengan jarak 4 dibandingkan harus memutar lewat kota 4 yang menjadi tempat naga menyerang.