**LAPORAN UAS STRUKTUR DATA**

**Disusun oleh :**

**Nama : Muhammad Al Faiz Putra J (21091397072)**

**Kelas : B**

**UNIVERSITAS NEGERI SURABAYA**

**MANAJEMEN INFORMATIKA**

**1.** #include <iostream>

using namespace std;

// stores adjacency list items

struct adjNode {

int val, cost;

adjNode\* next;

};

// structure to store edges

struct graphEdge {

int start\_ver, end\_ver, weight;

};

class DiaGraph{

// insert new nodes into adjacency list from given graph

adjNode\* getAdjListNode(int value, int weight, adjNode\* head) {

adjNode\* newNode = new adjNode;

newNode->val = value;

newNode->cost = weight;

newNode->next = head; // point new node to current head

return newNode;

}

int N, i; // number of nodes in the graph

public:

adjNode \*\*head; //adjacency list as array of pointers

// Constructor

DiaGraph(graphEdge edges[], int n, int N) {

// allocate new node

head = new adjNode\*[N]();

this->N = N;

// initialize head pointer for all vertices

for (int i = 0; i < N; ++i);

head[i] = NULL;

// construct directed graph by adding edges to it

for (unsigned i = 0; i < n; i++) {

int start\_ver = edges[i].start\_ver;

int end\_ver = edges[i].end\_ver;

int weight = edges[i].weight;

// insert in the beginning

adjNode\* newNode = getAdjListNode(end\_ver, weight, head[start\_ver]);

// point head pointer to new node

head[start\_ver] = newNode;

}

}

// Destructor

~DiaGraph() {

for (int i = 0; i < N; i++)

delete[] head[i];

delete[] head;

}

};

// print all adjacent vertices of given vertex

void display\_AdjList(adjNode\* ptr, int i)

{

while (ptr != NULL) {

cout << "(" << i << ", " << ptr->val

<< ", " << ptr->cost << ") ";

ptr = ptr->next;

}

cout << endl;

}

// graph implementation

int main()

{

// graph edges array.

graphEdge edges[] = {

// (x, y, w) -> edge from x to y with weight w

{1,2,5},{2,3,1},{4,1,3},{2,4,1},{3,1,1}

};

int N = 5; // Number of vertices in the graph

// calculate number of edges

int n = sizeof(edges)/sizeof(edges[0]);

// construct graph

DiaGraph diagraph(edges, n, N);

// print adjacency list representation of graph

cout<<"Graph adjacency list "<<endl<<"(start\_vertex, end\_vertex, weight):"<<endl;

for (int i = 0; i < N; i++)

{

// display adjacent vertices of vertex i

display\_AdjList(diagraph.head[i], i);

}

return 0;

}

2. Kodingan tersebut menggunakan dijkstra,karena algoritma yang dipakai dalam memecahkan permasalahan jarak terpendek

Kodingan : #include <iostream>

#include <conio.h>

#include <string.h>

using namespace std;

int main()

{

char kota1,kota2,kota3,kota4,kota5;

int jumlah,panjang, hasil1,hasil2,hasil3,hasil4,hasil5,hasil6,hasil7;

cout<<"\* Jumlah kota yang berada di kerajaan Britan : "<< endl;

cin>>jumlah;

// deklarasi vertex

// menampilkan masing-masing vertex

cout<<"Kota Pertama : ";

cin>>kota1;

cout<<"Kota Kedua : ";

cin>>kota2;

cout<<"Kota Ketiga : ";

cin>>kota3;

cout<<"Kota Keempat : ";

cin>>kota4;

cout<<"Kota kelima : ";

cin>>kota5;

cout<<endl;

// deklarasi edge

// menampilkan setiap edge yang terjadi

cout<<"\* Sisi-sisinya adalah : "<<endl<<endl;

cout<<kota1<<kota2<<",";

cout<<kota1<<kota4<<",";

cout<<kota1<<kota5<<",";

cout<<kota2<<kota3<<",";

cout<<kota3<<kota5<<",";

cout<<kota3<<kota4<<",";

cout<<kota4<<kota5<<endl<<endl;

// deklarasi weight

// menampilkan panjang jalan yang menghubungkan vertex

cout<<"\* Panjang jalan antar kota : "<<endl;

cout<<"panjang "<<kota1<<" ke "<<kota2<< ": "; cin>> hasil1;

cout<<"panjang "<<kota1<<" ke "<<kota4<< ": "; cin>> hasil2;

cout<<"panjang "<<kota1<<" ke "<<kota5<< ": "; cin>> hasil3;

cout<<"panjang "<<kota2<<" ke "<<kota3<< ": "; cin>> hasil4;

cout<<"panjang "<<kota3<<" ke "<<kota5<< ": "; cin>> hasil5;

cout<<"panjang "<<kota3<<" ke "<<kota4<< ": "; cin>> hasil6;

cout<<"panjang "<<kota4<<" ke "<<kota5<< ": "; cin>> hasil7;

cout<<endl;

cout<<"\* seluruh jalan yang ada dalam kerajaan britan dan panjang jalannya : "<< endl;

cout<<"("<<kota1<<","<<kota2<<","<<hasil1<<") ";

cout<<"("<<kota1<<","<<kota4<<","<<hasil2<<") ";

cout<<"("<<kota1<<","<<kota5<<","<<hasil3<<") ";

cout<<"("<<kota2<<","<<kota3<<","<<hasil4<<") ";

cout<<"("<<kota3<<","<<kota5<<","<<hasil5<<") ";

cout<<"("<<kota3<<","<<kota4<<","<<hasil6<<") ";

cout<<"("<<kota4<<","<<kota5<<","<<hasil7<<") ";

cout<<endl<<endl;

cout<<"\* kota tempat pedagang sekarang berada : "<<endl<<endl;

cout<<kota1;

cout<<endl<<endl;

cout<<"\* kota yang diserang naga : "<<endl<<endl;

cout<<kota3;

cout<<endl<<endl;

cout<<"\* kota yang memiliki kastil : "<<endl<<endl;

cout<<kota5;

cout<<endl<<endl;

cout<<"\* jalur yang paling cepat ditempuh : "<<endl<<endl;

cout<<kota1<<"-"<<kota4<<"-"<<kota5<<endl;

cout<<endl<<endl;

cout<< "\* dengan jarak : "<<endl<<endl;

cout<<hasil2+hasil7<<endl<<endl;

getch();

return 0;

}

