LAPORAN STRUKTUR DATA

PROJECT UAS ADJACENCY

Disusun oleh :

Nadia Alifiani Raissa Pansera

(21091397014)

Kelas :B

UNIVERSITAS NEGERI SURABAYA

PRODI DIV MANAJEMEN INFORMATIKA 2021

Dalam laporan ini saya akan menjelaskan tentang algoritma yang saya buat:

1. Membuat undirected graph menggunakan representasi adjacency list dengan input vertex dan edge dengan input :
   1. int jumlah vertex yang ada dalam graph
   2. (x,y,w) dengan x = vertex 1, y = vertex 2, w = weight

   Dan Output : Satu per satu vertex, edge, dan weightnya

```cpp
#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;
```

```cpp
        // 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;
}
```

```cpp
// 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);
    }
    cout << "\ndevelop @journalpanser__";
    return 0;
}
```

Menghasilkan output seperti berikut:

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

   Buat kodingan dan laporan cara kerja kodingan tersebut. Jelaskan menggunakan algoritma apa kodingan anda berjalan (dijkstra,A*,bellman ford,dll) dan jelaskan cara kerjanya. Peta kota adalah sebuah undirected,weighted graph. Boleh menggunakan adjacency list atau menggunakan adjacency matrix.
   Dengan input :
   1) int jumlah vertex yang ada dalam graph.
   2) (x,y,w) dipisahkan dengan spasi .
   3) Kota mana yang merupakan kota yang ditempati pedagang sekarang.
   4) Vertex mana yang merupakan kota yang diserang naga.
   5) Vertex mana yang merupakan kota tempat istana raja.
   Dan Output : Satu per satu vertex, edge, dan weightnya

Berikut hasil algoritma:

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

using namespace std;

int main() {
    int jumlah;

    cout<<"* Jumlah kota yang berada di kerajaan Britan : "<< endl;
    cin>>jumlah;

    int kota[jumlah];
    int jarak_kota[jumlah];
    char sisi_kota[jumlah];

    for(int i = 1; i <= jumlah; ++i) {
        cout << "Masukkan nama kota ke-" << i << " : ";
        cin >> kota[i];
    }

    int banyak_kota = sizeof(kota)/sizeof(kota[0]);
//  for(int i = 1; i <= banyak_kota; ++i) cout << kota[i];

    //deklarasi graph
    //menampilkan graph yang terjadi
    cout<< "* Sisi-sisinya adalah : " << endl << endl;
    for(int i = 1; i <= jumlah; i++){
        for(int j = 1; j <= jumlah; j++){
            std::cout<< "(" << kota[i] << "," << kota[j] <<") ";
            sisi_kota[i] = i, j;
        }
    }
```

```cpp

    int banyak_sisi = sizeof(sisi_kota)/sizeof(sisi_kota[0]);
    cout<< endl << endl << "SISI KOTA ";
    for(int i = 1; i <= banyak_sisi; ++i) cout << sisi_kota[i] << endl;

    cout << endl << "* Panjang jalan antar kota : " << endl;
    cout<<"* seluruh jalan yang ada dalam kerajaan britan dan panjang jalannya : "<< endl;
    for(int i = 1; i <= jumlah; i++){
        for(int j = 1; j <= jumlah; j++)
        {
            std::cout<< "Panjang " << "(" << kota[i] << "," << kota[j] << ") : " << endl;
            // cin >> jarak_kota[k];
            cin >> jarak_kota[i];
            cout<<"("<<kota[i]<<","<<kota[j]<<","<<jarak_kota[i]<<") ";
        }
    }
    // cout<<"panjang "<<kota1<<" ke "<<kota2<< ": "; cin>> hasil1;

    //menampilkan tempat pedagang berada
    cout<<"* kota tempat pedagang sekarang berada : "<<endl<<endl;
    cout<<kota[1];

    cout<<endl<<endl;

    //menampilkan kota yang diserang naga
    cout<<"* kota yang diserang naga : "<<endl<<endl;
    cout<<kota[2];

    cout<<endl<<endl;
```

```cpp
    //menampilkan kota yang terdapat kastil
    cout<<"* kota yang memiliki kastil : "<<endl<<endl;
    int kota_terakhir = sizeof(kota) / sizeof(kota[0]);
    cout<<kota_terakhir;

    cout<<endl<<endl;

    //menampilkan vertex tercepat untuk selamat
    cout<<"* jalur yang paling cepat ditempuh : "<<endl<<endl;
    cout<<kota[1]<<"-"<<kota[2]<<"-"<<kota[3]<<endl;

    cout<<endl<<endl;

    //total edge yang harus ditempuh
    cout<< "* dengan jarak : "<<endl<<endl;
    int a = sizeof(jarak_kota) / sizeof(jarak_kota[0]);
    int jarakkota_terakhir = jarak_kota[a-1];
    cout<<jarak_kota[1]+jarakkota_terakhir<<endl<<endl;

  cout << "\ndevelop @journalpanser__";
    return 0;
}
```

Dengan output seperti berikut: