Nama : Maula Izza Azizi

NRP : 5025201104

Laporan Praktikum Struktur Data ke-4

Soal 1 Sunda Empire

Soal :

Setelah bumi dikembalikan ke titik 0 peradaban usai perang dunia kedua, bumi diatur menurut tatanan dunia oleh sunda empire yang diputuskan di bandung. Dunia ini terbagi menjadi 7 benua dan menjadi wilayah terbesar di bumi, tetapi seluruh wilayah di bumi ini belum terhubung seluruhnya. Kami akan memberi mu N daratan yang merupakan pulau yang telah didata. Di sini kalian akan membantu US "uncle SAM" dan pemimpinnya SAM "Sultan Abdul Mufakkir" untuk menghubungkan seluruh wilayah di bumi, tidak hanya benua saja tetapi seluruh daratan di bumi !!!!!!

**Input Format**

baris pertama integer P dan T yang merupakan jumlah pulau dan berapa yang terhubung. kemudian T baris setelahnya adalah integer A dan B merupakan pulau yang telah terhubung

**Constraints**

Semua angka yang digunakan di soal ini dalam range int

**Output Format**

angka int yang merupakan jumlah koneksi yang diperlukan agar semua daratan terhubung

**Sample Input 0**

7 5

5 1

7 4

3 6

3 2

7 3

**Sample Output 0**

1

Jawaban :

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <iterator>

#include <list>

#include <queue>

#include <array>

#include <set>

#include <algorithm>

#include <vector>

#include <map>

#include <utility>

#include <tuple>

#include <iostream>

#include <stack>

typedef long long int ll;

typedef long double ld;

using namespace std;

class Graph {

int V;

list<int>\* adj;

void DFSUtil(int v, bool visited[]);

public:

Graph(int V);

void addEdge(int v, int w);

void NCC();

};

// Number of Connection Component

void Graph::NCC()

{

bool\* visited = new bool[V];

int count = 0;

for (int v = 0; v < V; v++)

visited[v] = false;

for (int v = 0; v < V; v++) {

if (!visited[v]) {

DFSUtil(v, visited);

count += 1;

}

}

cout << count - 1 << endl;

}

void Graph::DFSUtil(int v, bool visited[])

{

visited[v] = true;

for (auto i = adj[v].begin(); i != adj[v].end(); ++i)

if (!visited[\*i])

DFSUtil(\*i, visited);

}

Graph::Graph(int V)

{

this->V = V;

adj = new list<int>[V];

}

void Graph::addEdge(int v, int w)

{

adj[v].push\_back(w);

adj[w].push\_back(v);

}

int main(int argc, char const \*argv[]){

// p = vertex

int p, t;

cin >> p >> t;

int a, b;

Graph g(p);

for (int i = 0; i < t; i++){

cin >> a >> b;

// indexing --> segmentation fault

g.addEdge(a-1, b-1);

}

g.NCC();

return 0;

}

Penjelasan :

Pertama kita lihat di fungsi int. main () dimana terdapat berbagai statement yang harus di eksekusi. Dimulai dari input p dan t sebagai jumlah pulau (vertex) dan jumlah berapa yang terhubung (edge) kemudian terdapat inputan a dan yang merupakan hubungan pulau satu dengan yang lainnya. Kemudian aku buat class Graph untuk make graph dimana aku memanggil constructor di dalam class dengan parameter int v (vertex). Saya assign vertex tersebut kedalam this->v dimana v yang dimaksud berada di dalam class. Kemudian ada statement adj = new list<int>[V]; fungsinya agar kita bisa membuat node berdasarkan list <int> yang bertipe int sebanya v (node atau vertex). Kemudian di fungsi main tersebut buat perulangan sebanyak t (jumlah vertex) dengan memanggil cin >> a >> b; kemudian memanggil fungsi yang berada di dalam class yaitu addEdge(a-1, b-1); di dalam nya terdapat statement void

Graph::addEdge(int v, int w)

{

adj[v].push\_back(w);

adj[w].push\_back(v);

}

Yang artinya adjacent sebanyak v (adj[v]) akan kita isi dengan fungsi push\_back(w). begitu pun sebaliknya. Hal ini berfungsi untuk menghubungkan vertex satu dengan vertex lainnya. Lalu kenapa saya memanggil fungsi addEdge(a-1, b-1); -> di minus 1 ??? karena untuk menghindari segmentation fault ☺. Dan setelah di loop sebanyak t. kita dapat memanggil method g.NCC artinya Number of Connection Component.

Dimana pada method ini kita akan menandakan semua bernilai false dengan statement

for (int v = 0; v < V; v++)

visited[v] = false;

kemudian kita akan mengunjungi nya dengan statement

for (int v = 0; v < V; v++) {

if (!visited[v]) {

DFSUtil(v, visited);

count += 1;

}

}

cout << count - 1 << endl;

}

Di dalam statement if (!visited[v]) artinya jika bernilai false. Maka akan menjalankan statement berikutnya dengan memanggil fungsi DFSUtil (v, visited) terdapat 2 parameter int v (jumlah vertex) dan tipe bool (visited). Berikut adalah method DFSUtil

void Graph::DFSUtil(int v, bool visited[])

{

visited[v] = true;

for (auto i = adj[v].begin(); i != adj[v].end(); ++i)

if (!visited[\*i])

DFSUtil(\*i, visited);

}

Kita akan menakan bahwa ketika fungsi ini di call maka aka semua yang berada di dalam graph tersebut di beri tanda true. Yang artinya sudah di kunjungi

visited[v] = true

kemudian kita akan merekursinya dengan loop dari awal

for (auto i = adj[v].begin(); i != adj[v].end(); ++i)

if (!visited[\*i])

DFSUtil(\*i, visited);

Setelah selesai execute maka akan di jumlahkan untuk menandakn berapa kali node tersebut di kunjungi dengan statement sebagai berikut

count += 1;

Bukti AC :

