Министерство образования Республики Беларусь

Учреждения образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине МОИС за 1 семестр

«***Нахождение минимального остового дерева связанного неориентированного графа.***

Выполнил

Студент группы ИИ-21

Корпач Д.Р.

Проверил

Козинский А.А.

Брест 2022

**Задание :** Найти минимальное остовое дерево для заданного графа *G* алгоритмом Прима и Крускаля. Варианты графов указаны в таблице 1. Граф задан списком ребер.

Ребра: (1,2),(1,3),(2,5),(2,4),(3,4),(3,6),(4,5),(4,6),(5,7),(5,8),(7,8),(8,6)

Вес ребер: (4,6,3,1,1,3,3,4,5,2,1,5)

#include <iostream>

#include <iomanip>

#include "../graph\_LIB.hh"

int main(){

    convert c;

    std::string file\_path = "inf5.txt";

    std::string file\_weight = "weight5.txt";

    std::vector<int> nodes = c.reading\_file(file\_path);

    std::vector<int> weight = c.reading\_file(file\_weight);

    int max\_node = c.count\_of\_nodes(nodes);

    std::vector<std::vector<int>> adjacencyMatrix = c.adjancy(nodes, max\_node);

    alg calculation;                                                                                ///если флаг равен 1 то матрица смежности весов сделана для алгоритма прима

    std::vector<std::vector<int>> adjancy\_prim=c.adjancy\_for\_deikstra\_floid(nodes, weight, max\_node,1);//ну и так же нужна 1 для алгоритма крускала

    std::cout<<std::endl<<"algorithm prim"<<std::endl<<"vertex"<<"\t"<<"distance"<<std::endl;

    calculation.prima(adjancy\_prim);//выводится вес ребра и вершины к которой оно принадлежит

    std::cout<<std::endl;

    std::cout<<"algorith kruskal"<<std::endl;

    std::vector<std::vector<int>> adjancy\_kruskal=c.adjancy\_for\_deikstra\_floid(nodes, weight, max\_node,1);

    std::vector<std::vector<int>> distance\_kryskal = calculation.kruskal(adjancy\_kruskal);

    //изменение получившейся матрицы смежности по арлгоритму крускала в массив вершин по которым прошелся алгоритм

    std::vector<int> vertices;

    for(int i=0;i<distance\_kryskal.size();i++){

        for(int j=0;j<distance\_kryskal.size();j++){

            if(i<j){

            if(distance\_kryskal[i][j]!=0){

                vertices.push\_back(i+1);

                vertices.push\_back(j+1);

            }}

        }

    }

    for(int i=0;i<vertices.size();i++){

        if(i%2==0){

            std::cout<<"("<<vertices[i]<<",";

        }else{

            std::cout<<vertices[i]<<"), ";

        }

    }

    return 0;

}