Министерство образования Республики Беларусь

Учреждения образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине МОИС за 1 семестр

«***Нахождение эйлеровых и гамильтоновых циклов в неориентированном графе.***»

Выполнил

Студент группы ИИ-21

Корпач Д.Р.

Проверил

Козинский А.А.

Брест 2022

**Задание 1-2:**Построить матрицу смежности и инцидентности для заданного графа. Изобразить граф.

1. Написать программу нахождения эйлерова цикла в графе. Результатом программы является последовательность вершин в цикле. Предусмотреть проверку критерия на наличия эйлерова цикла в графе.
2. Написать программу нахождения гамильтонова цикла в графе. Результат программы – последовательность вершин цикла.
3. Варианты заданий указаны в таблице. Графы заданы списком ребер, *n*– количество вершин, -- проверка графа на эйлеров цикл,  - на гамильтонов цикл.
4. Изобразить графы и циклы в них.

****

#include <iostream>

#include<vector>

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

using namespace std;

int main()

{

    convert c;

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

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

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

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

    queue que;

    alg calculation;

    std::cout<<"hamiltonian cycle"<<std::endl;

    que = calculation.Hamilton\_cycle(adjacencyMatrix);

    que.print();

    std::cout<<"check eulerian cycle"<<std::endl;

    file\_path = "inf2.txt";

    nodes = c.reading\_file(file\_path);

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

    std::vector<std::vector<int>> adjacencyMatrix2 = c.adjancy(nodes, max\_node);

    if (calculation.find\_component\_with\_bfc(adjacencyMatrix2) != 1)

    {

        std::cout << "The graph not coherent";

        exit(0);

    }

    // second requirement

    std::vector<int> degrees(max\_node);

    for (int i = 0; i < max\_node; i++){

        for (int j = 0; j < max\_node; j++){

            if (adjacencyMatrix2[i][j]){

                degrees[i]++;

            }

        }

    }

    for(int i=0;i<max\_node;i++){

        std::cout<<degrees[i]<<" ";

    }std::cout<<std::endl;

    for (int i = 0; i < max\_node; i++){

        if (degrees[i] % 2 != 0){

            std::cout << "The graph not Eulerian";

            exit(0);

        }

    }

    std::vector<int> cycle\_euler;

    cycle\_euler = calculation.Euleran\_cyrcle(adjacencyMatrix2);

    std::cout << "The euleran cycle is: ";

    for (int i = cycle\_euler.size() - 1; i >= 0; i--){

        std::cout << cycle\_euler[i] + 1<<"-";

    }

return 0;

}

