Министерство науки и высшего образования

Пензенский государственный университет

Кафедра “Вычислительная техника”

**Отчет**

по лабораторной работе №5

по курсу “ Логика и основы алгоритмизации в инженерных задачах”

на тему “Определение характеристик графов”

Выполнили

студенты группы 22ВВВ3:

Бормотов А.А.

Кочетков А.М.

Кузьмин Д.В.

Приняли

Акифьев И.В.

Юрова О.В.

Пенза 2023

**Задание 1:**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.
2. Определите размер графа *G*, используя матрицу смежности графа.
3. Найдите изолированные, концевые и доминирующие вершины.

### **Задание 2:**

1. Постройте для графа G матрицу инцидентности.
2. Определите размер графа *G*, используя матрицу инцидентности графа.
3. Найдите изолированные, концевые и доминирующие вершины.

**Решение заданий**

**Задание 1.1:**

Реализовали матрицу как вектор векторов. Заполнение случайными числами осуществили через два вложенных цикла для строк и строчек соответственно. Значения генерируется только для треугольной матрицы, потом они дублируются по другую сторону главной диагонали, чтобы получилась матрица смежности.

**Задание 1.2:**

Рассчитали размер графа как количество единиц в треугольной матрице.

**Задание 1.3:**

Нашли изолированные, концевые и доминирующие вершины путем подсчета единиц в каждой строчке матрицы смежности.

**Задание 1.1:**

Реализовали матрицу как вектор векторов, с размерами (кол-во вершин) x (количество ребер).

**Задание 1.2:**

Рассчитали размер графа.

**Задание 1.3:**

Нашли изолированные, концевые и доминирующие вершины путем подсчета единиц в каждой строчке матрицы инцидентности.

**Листинг**

**#include <iostream>**

**#include <vector>**

**#include <cstdlib>**

**#include <time.h>**

**#include <list>**

**using namespace std;**

**int main() {**

**int size, graph\_size = 0, counter = 0, inc\_counter = 0;**

**bool check\_isolated = true;**

**list<int> isolated = {0}, inc\_isolated = {0};**

**list<int> end\_nodes = {0}, inc\_end\_nodes = {0};**

**list<int> dominant\_nodes = {0}, inc\_dominant\_nodes = {0};**

**srand(time(0));**

**cout << "Enter the amount of Graph's nodes" << endl;**

**cin >> size;**

**vector<vector<int>> matrix(size, vector<int> (size));**

**for(int i=0 ; i != size; i++){**

**for(int j=i ; j != size; j++){**

**matrix[i][j] = rand()%2;**

**if (matrix[i][j] == 1){**

**graph\_size++;**

**} // counting the amount of nodes AKA Graph size**

**matrix[j][i] = matrix[i][j];**

**}**

**}**

**for(int i=0 ; i != size; i++){**

**for(int j=0 ; j != size; j++){**

**if(matrix[i][j] == 1) counter++;**

**}**

**if(counter == 0) isolated.push\_back(i);**

**else if(counter == 1) end\_nodes.push\_back(i);**

**else if(counter == size) dominant\_nodes.push\_back(i);**

**counter = 0;**

**}**

**cout << "Adjacency matrix for graph G:" << endl;**

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

**for (int j = 0; j < size; j++){**

**cout << matrix[i][j] << " ";**

**}**

**cout << endl;**

**}**

**printf("This is a TOYOTA, erm a Graph consiting of %d nodes.\nGraph size is %d\n", size, graph\_size);**

**//std::cout << \_\_cplusplus << std::endl;**

**cout << "List of isolated nodes of this Graph:" << endl;**

**isolated.erase(isolated.begin());**

**for(int numberh: isolated){**

**cout << numberh << " ";**

**};**

**cout << endl;**

**cout << "List of end nodes of this Graph:" << endl;**

**end\_nodes.erase(end\_nodes.begin());**

**for(int numberh: end\_nodes){**

**cout << numberh << " ";**

**};**

**cout << endl;**

**cout << "List of dominant nodes of this Graph:" << endl;**

**dominant\_nodes.erase(dominant\_nodes.begin());**

**for(int numberh: dominant\_nodes){**

**cout << numberh << " ";**

**};**

**cout << endl;**

**//the incidence matrix**

**vector<vector<int>> incidence\_matrix(size, vector<int> (graph\_size, 0));**

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

**for (int j = i; j < size; j++){**

**if (matrix[i][j] == 1) {**

**incidence\_matrix[i][inc\_counter] = 1;**

**incidence\_matrix[j][inc\_counter] = 1;**

**inc\_counter++; // counts the amount of paths which have been already added**

**}**

**}**

**}**

**cout << endl << endl << endl;**

**cout << "Incidence matrix for graph G:" << endl;**

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

**for (int j = 0; j < graph\_size; j++){**

**cout << incidence\_matrix[i][j] << " ";**

**}**

**cout << endl;**

**}**

**printf("Mooonday left me broken, Tues- Erm.. This is a Graph consiting of %d nodes.\nGraph size is %d\n", size, inc\_counter);**

**for(int i=0 ; i != size; i++){**

**for(int j=0 ; j != inc\_counter; j++){**

**if(incidence\_matrix[i][j] == 1) counter++;**

**}**

**if(counter == 0) inc\_isolated.push\_back(i);**

**else if(counter == 1) inc\_end\_nodes.push\_back(i);**

**else if(counter == graph\_size) inc\_dominant\_nodes.push\_back(i);**

**counter = 0;**

**}**

**cout << "List of isolated nodes of this Graph:" << endl;**

**inc\_isolated.erase(inc\_isolated.begin());**

**for(int numberh: inc\_isolated){**

**cout << numberh << " ";**

**};**

**cout << endl;**

**cout << "List of end nodes of this Graph:" << endl;**

**inc\_end\_nodes.erase(inc\_end\_nodes.begin());**

**for(int numberh: inc\_end\_nodes){**

**cout << numberh << " ";**

**};**

**cout << endl;**

**cout << "List of dominant nodes of this Graph:" << endl;**

**inc\_dominant\_nodes.erase(inc\_dominant\_nodes.begin());**

**for(int numberh: inc\_dominant\_nodes){**

**cout << numberh << " ";**

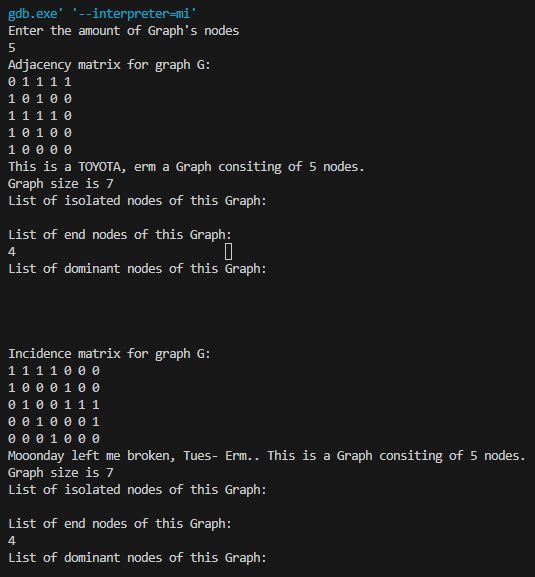
**};**

**cout << endl;**

**return 0;**

**}**

**Результаты работы программы**



**Вывод**

В результате выполнения лабораторной работы были успешно реализованы представление графов через матрицы смежности и инцидентности, их заполнение, а также нахождение различных типов вершин, содержащихся в них.