Министерство науки и высшего образования

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №5

по дисциплине: "Логика и основы алгоритмизации в инженерных задачах"

на тему: "Определение характеристик графов"

Выполнили:

студенты группы 23ВВВ4

Королёв Д.В.

Алешин К.А.

Приняли:

Юрова О.В.

Деев М.В.

Пенза, 2024

**Общие сведения.**

Если *G –* граф (рисунок 1), содержащий непустое множество *n* вершин *V* и множество ребер *E,* где *e(vi, vj) –* ребро между двумя произвольными вершинами *vi* и *vj*, тогда **размер** графа G есть мощность множества ребер |E(G)| или, количество ребер графа.

**Степенью**вершины графа *G* называется число инцидентных ей ребер. Степень вершины *vi* обозначается через *deg(vi).*

**Задание 1.**

**Листинг**

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

int\*\* create\_adjacency\_matrix(size\_t size)

{

int\*\* matrix = (int\*\*)malloc(size \* sizeof(int\*));

if (!matrix)

{

return nullptr;

}

for (size\_t i = 0; i < size; ++i)

{

matrix[i] = (int\*)malloc(size \* sizeof(int));

if (!matrix[i])

{

for (size\_t j = 0; j < i; ++j)

{

free(matrix[j]);

}

free(matrix);

return nullptr;

}

}

for (size\_t i = 0; i < size; ++i)

{

for (size\_t j = i + 1; j < size; ++j)

{

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

}

}

for (size\_t i = 1; i < size; ++i)

{

for (size\_t j = 0; j < i; ++j)

{

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

}

}

for (size\_t i = 0; i < size; ++i)

{

matrix[i][i] = 0;

}

return matrix;

}

void print\_matrix(int\*\* matrix, size\_t size)

{

for (size\_t i = 0; i < size; ++i)

{

for (size\_t j = 0; j < size; ++j) {

std::cout << matrix[i][j] << " ";

}

std::cout << std::endl;

}

}

void free\_matrix(int\*\* matrix, size\_t size)

{

if (!matrix)

{

return;

}

else

{

for (size\_t i = 0; i < size; ++i)

{

free(matrix[i]);

}

free(matrix);

}

}

size\_t size\_G\_grapg(int\*\* matrix, size\_t size)

{

size\_t size\_g = 0;

for (size\_t i = 0; i < size; ++i)

{

for (size\_t j = i + 1; j < size; ++j)

{

if (matrix[i][j] == 1) ++size\_g;

}

}

return size\_g;

}

void isolated(int\*\* matrix, size\_t size)

{

size\_t isol = 0;

for (size\_t i = 0; i < size; ++i)

{

int sum = 0;

for (size\_t j = 0; j < size; ++j)

{

sum += matrix[i][j] + matrix[j][i];

}

if (sum == 0) ++isol;

}

std::cout << "Isolated: " << isol << std::endl;

}

void terminal(int\*\* matrix, size\_t size)

{

size\_t term = 0;

for (size\_t i = 0; i < size; ++i)

{

int sum = 0;

for (size\_t j = 0; j < size; ++j)

{

sum += matrix[i][j] + matrix[j][i];

}

if (sum == 1) ++term;

}

std::cout << "Terminal: " << term << std::endl;

}

void dominating(int\*\* matrix, size\_t size)

{

size\_t dom = 0;

for (size\_t i = 0; i < size; ++i)

{

int sum = 0;

for (size\_t j = 0; j < size; ++j)

{

if (i != j)

{

sum += matrix[i][j];

}

}

if (sum == size - 1)

{

++dom;

}

}

std::cout << "Dominating: " << dom << std::endl;

}

void start()

{

srand(static\_cast<unsigned int>(time(nullptr)));

while (true)

{

std::cout << "if you wanna finish - enter '0'\n";

std::cout << "Enter size matrix (type size\_t) : ";

size\_t size;

std::cin >> size;

if (!size)

{

return;

}

int\*\* matrix = create\_adjacency\_matrix(size);

if (!matrix)

{

std::cerr << "Error allocate\n";

continue;

}

print\_matrix(matrix, size);

std::cout << "Size G-praph - " << size\_G\_grapg(matrix, size) << std::endl;

isolated(matrix, size);

terminal(matrix, size);

dominating(matrix, size);

free\_matrix(matrix, size);

std::cout << std::endl;

std::cin.clear();

}

}

int main() {

start();

}

Результат работы программы

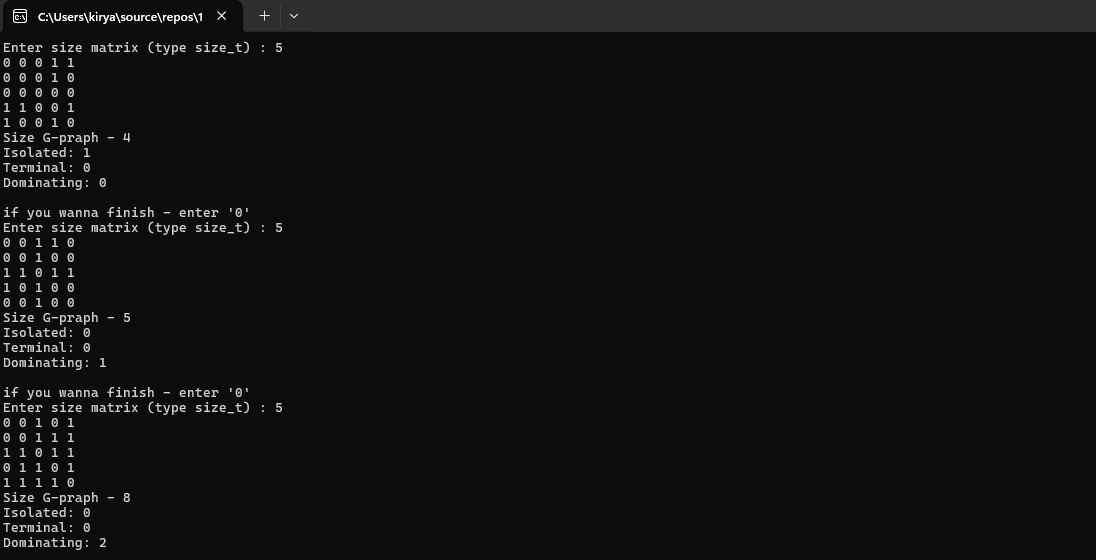


Рисунок № 1

**Вывод -** были получены навыки реализации неориентированных графов,

Матрицы смежности, также получены навыки нахождения изолированных, доминирующих, концевых вершин.