МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7**

**по дисциплине  
 «АЛГОРИТМЫ И АНАЛИЗ СЛОЖНОСТИ»**

Выполнил студент группы 25/2                                       А.А. Козин

Направление подготовки  02.03.03  Математическое обеспечение и администрирование информационных систем

Курс    2

Отчет принял доктор физико-математических наук, профессор                                                                                       А.И. Миков

Краснодар

2021 г.

**Задание**: определить изоморфизм графов G1 и G2

**Решение**: создадим случайный граф. Потом сделаем отображение вершин, то есть какая вершина во втором графе будет соответствовать вершине в первом и получается, что графы всегда будут изоморфны.

**Текст программы:**

#include <iostream>

#include <vector>

#include <set>

#include <algorithm>

using namespace std;

struct Vertez {

vector<int> neighbours;

};

vector<Vertez> G1, G2;

int counter = 0;

void fin(vector<int>& v, int n) {

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

int random = rand() % n;

while (v[random] != 0) {

random++;

random %= n;

}

v[random] = i;

}

}

void randomGraph(int n) {

G1.resize(n);

int z = 1 + rand() % (n \* (n - 1)) / 2;

set<pair<int, int>> q;

while (q.size() != z) {

int x = rand() % n;

int y = rand() % n;

if (q.find({ x, y }) == q.end() && q.find({ y, x }) == q.end() && x != y) {

G1[x].neighbours.push\_back(y);

G1[y].neighbours.push\_back(x);

q.insert({ x, y });

}

}

}

void mapz(int n) {

G2.resize(n);

//вершина inz[i] соответствует i

vector<int> inz(n);

fin(inz, n);

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

G2[i] = G1[inz[i]];

for (int j = 0; j < G2[i].neighbours.size(); j++) {

vector<int> a = G2[i].neighbours;

G2[a[j]] = G1[inz[a[j]]];

}

}

}

bool is(vector<int> ind, int n) {

vector<Vertez> G(n);

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

counter += 2;

G[i] = G2[ind[i]];

counter++;

for (int j = 0; j < G[i].neighbours.size(); j++) {

vector<int> neighbours = G[i].neighbours;

counter++;

G[neighbours[j]] = G2[ind[neighbours[j]]];

counter++;

}

}

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

if (G[i].neighbours.empty() && G1[i].neighbours.empty()) {

counter++;

continue;

}

if (G[i].neighbours.empty() || G1[i].neighbours.empty()) {

counter++;

return false;

}

else {

if (G[i].neighbours.size() != G1[i].neighbours.size()) {

counter++;

return false;

}

}

}

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

for (int j = 0; j < G[i].neighbours.size(); j++) {

if (G[i].neighbours[j] != G1[i].neighbours[j]) {

counter++;

return false;

}

}

}

return true;

}

void main() {

srand(time(NULL));

for (int n = 2; n <= 20; n++) {

long long sum = 0;

for (int j = 0; j < 1000; j++) {

counter = 0;

vector<int> ind(n);

randomGraph(n);

mapz(n);

fin(ind, n);

while (!is(ind, n)) {

next\_permutation(ind.begin(), ind.end());

}

sum += counter;

G1.clear(); G2.clear();

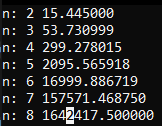
}

cout << fixed << "n: " << n << " " << (float)sum / 1000 << "\n";

}

}

**Программа выдает в консоли результаты в следующем виде:**



(На рисунке представлен вывод n до 8, но программа выполняется до n = 20)

**Для наглядности построим график:**

При решении были использованы все возможные перестановки отображения вершин, поэтому сложность алгоритма не может быть полиномиальной.