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 25/1

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.

**Задача**: написать алгоритм проверки графов на изоморфизм.

**Решение**: сначала создадим случайный граф. Затем придумаем отображение вершин, то есть какая вершина во втором графе будет соответствовать вершине в первом. Получается, что графы всегда будут изоморфны.

struct Ver {

vector<int> neigh;

};

vector<Ver> g1, g2;

int counter = 0;

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

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

int i = rand() % n;

while (v[i] != 0) {

i++;

i %= n;

}

v[i] = ver;

}

}

void randG(int n) {

g1.resize(n);

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

set<pair<int, int>> r;

while (r.size() != c) {

int a = rand() % n;

int b = rand() % n;

if (r.find({ a, b }) == r.end() && r.find({ b, a }) == r.end() && a != b) {

g1[a].neigh.push\_back(b);

g1[b].neigh.push\_back(a);

r.insert({ a, b });

}

}

}

void Map(int n) {

g2.resize(n);

//вершина ind[i] соответствует i

vector<int> ind(n);

f(ind, n);

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

g2[i] = g1[ind[i]];

for (int j = 0; j < g2[i].neigh.size(); j++) {

vector<int> a = g2[i].neigh;

g2[a[j]] = g1[ind[a[j]]];

}

}

}

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

vector<Ver> g(n);

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

counter += 2;

g[i] = g2[ind[i]];

counter++;

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

vector<int> neigh = g[i].neigh;

counter++;

g[neigh[j]] = g2[ind[neigh[j]]];

counter++;

}

}

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

if (g[i].neigh.empty() && g1[i].neigh.empty()) {

counter++;

continue;

}

if (g[i].neigh.empty() || g1[i].neigh.empty()) {

counter++;

return false;

}

else {

if (g[i].neigh.size() != g1[i].neigh.size()) {

counter++;

return false;

}

}

}

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

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

if (g[i].neigh[j] != g1[i].neigh[j]) {

counter++;

return false;

}

}

}

return true;

}

int main() {

srand(time(NULL));

ofstream file("file.txt");

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

long long sum = 0;

cout << "n: " << n << "\n";

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

counter = 0;

vector<int> ind(n);

randG(n);

Map(n);

f(ind, n);

while (!is(ind, n)) {

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

}

sum += counter;

g1.clear(); g2.clear();

}

file << fixed << n << " " << (double)sum / 1000 << "\n";

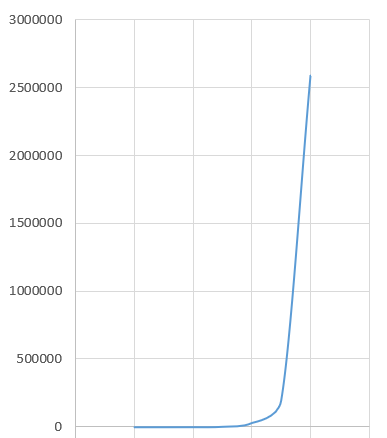
}

file.close();

return 0;

}

По данным из файла построим график:



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