Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

Национальный исследовательский центр университет ИТМО

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

ЛАБОРАТОРНАЯ РАБОТА №5

ПО ДИСЦИПЛИНЕ «ИСПРАВЛЕНИЕ ОШИБОК В ТЕКСТЕ»

Выполнил Чечулин Лев Олегович

Проверил _____

САНКТ-ПЕТЕРБУРГ, 2020Г.

ОТВЕТЫ

Количество рёбер: 2524

Количество изолят: 7

Изоляты: 144 247 409 721 791 826 862

Максимум из степеней каждой вершины: 12

Вершины с максимальной степенью: 236 309 910

Диаметр графа: 8

AB = 3

Путь от A до B: 553 708 618 415

CD = 5

Путь от C до D: 202 131 971 810 123 39

EF = 3

Путь от E до F: 488 27 446 929

CLEARING...

Количество рёбер: 2341

Количество изолят: 7

Изоляты: 144 247 409 721 791 826 862

Максимум из степеней каждой вершины: 12

Вершины с максимальной степенью: 236 309

Диаметр графа: 10

AB = 5

Путь от A до B: 553 239 292 426 338 415

CD = 6

Путь от C до D: 202 30 145 601 809 158 39

EF = 5

Путь от E до F: 488 369 346 876 446 929

ХОД РАБОТЫ

Граф храним в двумерном векторе G.

Все вершины, связанные с i-ой будут храниться в векторе G[i].

Степень вершины, это кол-во вершин, с которыми она связана, т.е. степень i-ой вершины, это G[i].size()

Кол-во рёбер, это сумма G[i].size() по всем i от 0 до 999, пополам. То есть сумма степеней всех вершин пополам, ибо каждое ребро связано с двумя вершинами и считается дважды.

Кратчайшие пути находим алгоритмом BFS, поиск в ширину. Но для данной задачи требуется немного усложнённый BFS, хранящий пути вершин.

Соответственно диаметр графа будет максимальным расстоянием при всех запусках вершин из всех максимальных расстояний до этой вершины.

КОД

```
#include<bits/stdc++.h>

#define int long long

#define ld long double

#define pii pair<int,int>

#define vi  vector<int>

#define f   first

#define s   second

#define pb  push_back

using namespace std;

int n = 1000;

vector<vector<int> > G(n);

vector<int> presence(n, 1);

vector<vector<int> > bfs(int a)

{

   vector<vector<int> > D(n);

   queue<int> q;
```

```
    q.push(a);
    D[a].pb(a);
    while(!q.empty())
    {
        int u = q.front();
        q.pop();
        for(int i = 0; i < G[u].size(); i++)
        {
            int to = G[u][i];
            if(D[to].empty() || D[to].size() > D[u].size() + 1)
            {
                D[to] = D[u];
                D[to].pb(to);
                q.push(to);
            }
        }
    }
    return D;
}
void work()
{
    int counter_edge = 0;
    for(int i = 0; i < n; i++) counter_edge += G[i].size();
    cout << "Quantity of edges is " << counter_edge / 2 << endl;
    vector<int> id_isolates;
    for(int i = 0; i < n; i++) if(G[i].empty() && presence[i]) id_isolates.pb(i);
    cout << "Quantity of isolates is " << id_isolates.size() << endl;
```

```
    cout << "There are: "; for(int i = 0; i < id_isolates.size(); i++) cout << id_isolates[i] << " "; cout << endl;

    int ma = 0;

    for(int i = 0; i < n; i++) if(ma < G[i].size()) ma = G[i].size();

    cout << "Maximum of degrees is " << ma << endl << "These vertexes include max degree: ";

    for(int i = 0; i < n; i++) if(ma == G[i].size()) cout << i << " "; cout << endl;

    ma = 0;

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

    {

        vector<vector<int> > len = bfs(i);

        for(int j = 0; j < len.size(); j++)

        {

            int s = len[j].size();

            if(len[j].size()) ma = max(ma, s);

        }

    }

    cout << "The diameter of this graph is " << ma - 1 << endl;

    cout << "The length between A and B is " << bfs(553)[415].size() - 1 << endl;

    cout << "Way from A to B: "; for(int i = 0; i < bfs(553)[415].size(); i++) cout << bfs(553)[415][i] << " "; cout << endl;

    cout << "The length between C and D is " << bfs(202)[39].size() - 1 << endl;

    cout << "Way from C to D: "; for(int i = 0; i < bfs(202)[39].size(); i++) cout << bfs(202)[39][i] << " "; cout << endl;

    cout << "The length between E and F is " << bfs(488)[929].size() - 1 << endl;

    cout << "Way from E to F: "; for(int i = 0; i < bfs(488)[929].size(); i++) cout << bfs(488)[929][i] << " "; cout << endl;

}

void solve()
```

```
{
    int a, b;
    while(cin >> a)
    {
        cin >> b;
        G[a].pb(b);
        G[b].pb(a);
    }
    work();
    cout << "CLEARING..." << endl;
    for(int i = 0; i < n; i += 17)
    {
        for(int j = 0; j < G[i].size(); j++)
        {
            vector<int> newneighbours;
            for(int k = 0; k < G[G[i][j]].size(); k++)
            {
                if(G[G[i][j]][k] != i) newneighbours.pb(G[G[i][j]][k]);
            }
            G[G[i][j]] = newneighbours;
        }
        G[i].clear();
        presence[i] = 0;
    }
    int DEL[6] = {131, 708, 971, 123, 910, 27};
    for(int i = 0; i < 6; i++)
    {
```

```
        for(int j = 0; j < G[DEL[i]].size(); j++)
        {
            vector<int> newneighbours;
            for(int k = 0; k < G[G[DEL[i]][j]].size(); k++)
            {
                if(G[G[DEL[i]][j]][k] != i) newneighbours.pb(G[G[DEL[i]][j]][k]);
            }
            G[G[DEL[i]][j]] = newneighbours;
        }
        G[DEL[i]].clear();
        presence[DEL[i]] = 0;
    }
    work();
}
signed main()
{
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
    int t = 1;
    while(t--)
    {
        solve();
    }
}
```