Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

По лабораторной работе №8

по дисциплине: «Логика и основы оптимизации в инженерных задачах»

на тему «Определение характеристик графов»

Выполнили студенты группы 19ВВ2:

Муромский Д.А.

Кобзев М.И.

Приняли:

Митрохин М. А.

Юрова О.В.

Пенза 2020

**Цель работы:** закрепить работу с алгоритмом поиска расстояний в графе в ширину с помощью реализации данного алгоритма на матрице инцидентности. Научиться определять радиус и диаметр графа и находить периферийные и центральные вершины графа, используя матрицу смежности и матрицу инцидентности. Также изучить методы нахождения изолированных, концевых и доминирующих вершин графа.

**Практическая часть**

1. **Задание 1.1** Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа *G*. Выведите матрицу на экран.

srand(time(NULL));

//Взвешенный неориентированный

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

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

if (rand() % 2) {

a[i][j] = 1 + rand() % 9;

a[j][i] = a[i][j];

}

else {

a[i][j] = 0;

a[j][i] = a[i][j];

}

}

}

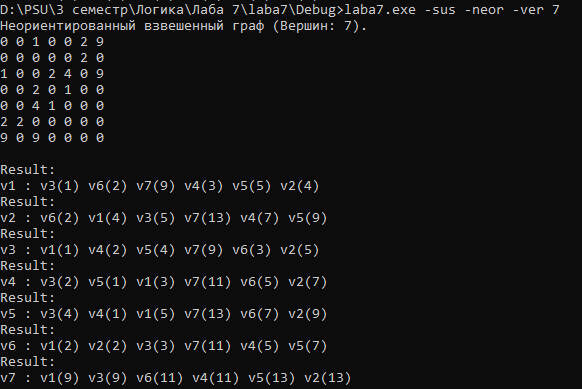


Рисунок 1 - Вывод матрицы смежности для неориентированного взвешенного графа

1. **Задание 1.2-1.4** Определите радиус и диаметр графа *G*, используя матрицу смежности графа. Определите подмножества периферийных и центральных вершин графа *G*, используя матрицу смежности. Найдите изолированные, концевые и доминирующие вершины.

void BFSD(int start, int\* dist, int\*\* a, int n) {

queue<int> Steck;

Steck.push(start);

dist[start] = 0;

int v;

while (!Steck.empty()) {

v = Steck.front();

Steck.pop();

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

if (a[v][i] && (dist[i] > (dist[v] + a[v][i]))) {

Steck.push(i);

dist[i] = dist[v] + a[v][i];

}

}

}

1. }
2. {...}

int radius = 10000;

int diametr = -1;

vector<int> max;

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

BFSD(i, vis, a, n);

printf("v%d : ", i + 1);

max.push\_back(-1);

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

if (vis[j] != 0 && vis[j] != 10000) {

printf("v%d(%d) ", j + 1, vis[j]);

if (max[i] < vis[j]) {

max[i] = vis[j];

}

}

vis[j] = 10000;

}

cout << endl;

}

vector<int> centr;

vector<int> peref;

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

if (radius > max[j]) {

centr.clear();

centr.push\_back(j);

radius = max[j];

}

else {

if (radius == max[j]) {

centr.push\_back(j);

}

}

if (diametr < max[j]) {

peref.clear();

peref.push\_back(j);

diametr = max[j];

}

else {

if (diametr == max[j]) {

peref.push\_back(j);

}

}

}

cout << endl;

cout << "Радиус: " << radius << endl;

cout << "Диаметр: " << diametr << endl;

cout << endl;

cout << "Переферийные вершины(diametr): ";

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

cout << "v" << peref[i] + 1 << " ";

}

cout << endl;

cout << "Центральные вершины(radius): ";

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

cout << "v" << centr[i] + 1 << " ";

}

cout << endl;

cout << endl;

int counter;

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

counter = 0;

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

if (a[i][j]) {

counter++;

}

}

if (counter == 0) {

cout << "V" << i + 1 << " - изолированная" << endl;

}

else {

if (counter == 1) {

cout << "V" << i + 1 << " - концевая" << endl;

}

else {

if (counter == (n - 1)) {

cout << "V" << i + 1 << " - доминирующая" << endl;

}

}

}

}

{...}



Рисунок 2 - Вывод радиуса , диаметра, периферийных и центральных вершин графа. Также вывод изолированным, концевых и доминирующих вершин.

**Задание 2.1\*** Постройте для графа G матрицу инцидентности.

printf("Реализация в матрице инцидентности\n");

vector<vector<int>> incen;

vector<int> temp;

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

incen.push\_back(temp);

}

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

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

if (a[i][j]) {

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

if (k == i || k == j) {

incen[k].push\_back(1);

}

else {

incen[k].push\_back(0);

}

}

temp.push\_back(a[i][j]);

}

}

}

int fantom\_n = incen[0].size();

cout << endl;

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

for (int j = 0; j < fantom\_n; j++) {

cout << incen[i][j] << " ";

}

cout << endl;

}

cout << endl;

for (int i = 0; i < fantom\_n; i++) {

cout << temp[i] << " ";

}

cout << endl;

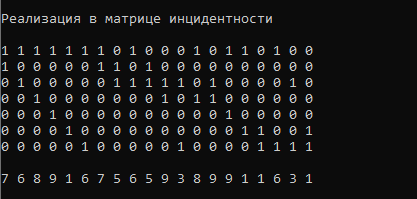


Рисунок 3 - Вывод матрицы инцидентности и массива весов

1. **Задание 2.2-2.3\*** Определите радиус и диаметр графа *G*, используя матрицу инцидентности графа. Определите подмножества периферийных и центральных вершин графа *G*, используя матрицу инцидентности.

void DFSD(int start, int\* dist,vector<vector<int>> a,vector<int> wes, int n, int fantom\_n) {

queue<int> Steck;

Steck.push(start);

dist[start] = 0;

int v;

while (!Steck.empty()) {

v = Steck.front();

Steck.pop();

for (int i = 0; i < fantom\_n; i++) {

if (a[v][i]) {

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

if (a[j][i] && (dist[j] > (dist[v] + wes[i]))) {

Steck.push(j);

dist[j] = dist[v] + wes[i];

}

}

}

}

}

1. }

{...}

radius = 10000;

diametr = -1;

max.clear();

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

DFSD(i, vis, incen, temp, n, fantom\_n);

printf("v%d : ", i + 1);

max.push\_back(-1);

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

if (vis[j] != 0 && vis[j] != 10000) {

printf("v%d(%d) ", j + 1, vis[j]);

if (max[i] < vis[j]) {

max[i] = vis[j];

}

}

vis[j] = 10000;

}

cout << endl;

}

centr.clear();

peref.clear();

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

if (radius > max[j] && max[j] != -1) {

centr.clear();

centr.push\_back(j);

radius = max[j];

}

else {

if (radius == max[j]) {

centr.push\_back(j);

}

}

if (diametr < max[j]) {

peref.clear();

peref.push\_back(j);

diametr = max[j];

}

else {

if (diametr == max[j]) {

peref.push\_back(j);

}

}

}

cout << endl;

cout << "Переферийные вершины(diametr): ";

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

cout << "v" << peref[i] + 1 << " ";

}

cout << endl;

cout << "Центральные вершины(radius): ";

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

cout << "v" << centr[i] + 1 << " ";

}

cout << endl;

cout << endl;

cout << "Радиус: " << radius << endl;

cout << "Диаметр: " << diametr << endl;

cout << endl;

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

counter = 0;

for (int j = 0; j < fantom\_n; j++) {

if (incen[i][j]) {

counter++;

}

}

if (counter == 0) {

cout << "V" << i + 1 << " - изолированная" << endl;

}

else {

if (counter == 2) {

cout << "V" << i + 1 << " - концевая" << endl;

}

else {

if (counter == (n - 1)\*2) {

cout << "V" << i + 1 << " - доминирующая" << endl;

}

}

}

1. }

{...}

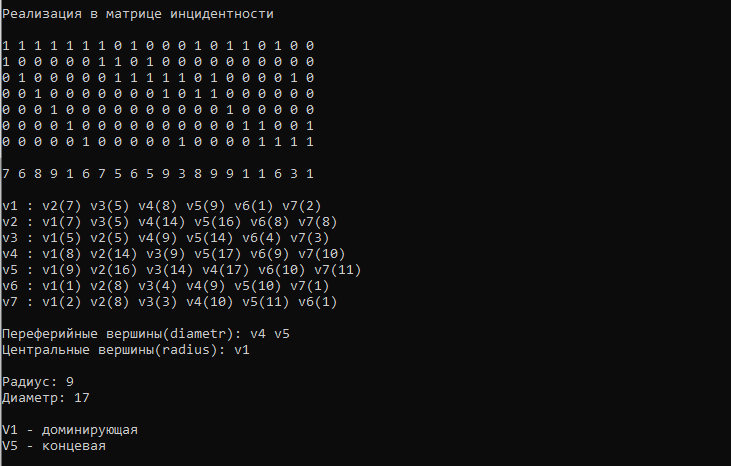


Рисунок 4 - Результат работы программы с матрицей инцидентности

**Вывод:** закрепили работу с алгоритмом поиска расстояний в графе в ширину с помощью реализации данного алгоритма на матрице инцидентности. Научились определять радиус и диаметр графа и находить периферийные и центральные вершины графа, используя матрицу смежности и матрицу инцидентности. Также изучили методы нахождения изолированных, концевых и доминирующих вершин графа.