Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №9

# по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний в графе»

**Выполнили**

студенты группы 21ВВ3

Чинов Даниил

Тюкалов Василий

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2022

# Название

Поиск расстояний в графе

**Цель работы** – изучение алгоритма поиск расстояний в графе.

# Лабораторное задание

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу

смежности для неориентированного графа G. Выведите матрицу на экран.

2. Для сгенерированного графа осуществите процедуру обхода в

глубину, реализованную в соответствии с приведенным выше описанием.

3.\* Реализуйте процедуру обхода в глубину для графа, представленного

списками смежности.

**Задание 2\***

1. Реализуйте процедуру поиска расстояний на основе обхода в глубину.

2. Реализуйте процедуру поиска расстояний на основе обхода в глубину

для графа, представленного списками смежности.

3. Оцените время работы реализаций алгоритмов поиска расстояний на

основе обхода в глубину и обхода в ширину для графов разных порядков.

# Листинг

//Выполнили студенты группы 21ВВ3: Тюкалов В.Е. и Чинов Д.Д.

#include<iostream>

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<queue>

#include<math.h>

#include<windows.h>

using namespace std;

queue <int> Q;

typedef struct Node {

int data = 0;

Node\* next = NULL;

};

void BFSD(int\*\* G, int\* dist, int v, int total) {

Q = {};

Q.push(v);

dist[v] = 0;

while (Q.empty() != true) {

v = Q.front();

Q.pop();

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

if (G[v][i] == 1 && dist[i] == -1) {

Q.push(i);

dist[i] = dist[v] + 1;

}

}

}

}

void BFSDL(int\* dist, int v, int total, Node\*\* start) {

Node\* swCell;

Q = {};

Q.push(v);

dist[v] = 0;

while (Q.empty() != true) {

v = Q.front();

Q.pop();

swCell = start[v];

while (swCell != NULL) {

if (dist[swCell->data] == -1) {

Q.push(swCell->data);

dist[swCell->data] = dist[v] + 1;

}

swCell = swCell->next;

}

}

}

void DFSD(int num, int v, int\* dist, int\*\* G, int\* numV) {

numV[v] = 1;

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

if (G[v][i] == 1) {

if (numV[i] == 0) {

dist[i] = dist[v] + 1;

DFSD(num, i, dist, G, numV);

}

}

}

}

void DFSDL(int num, int v, int\* dist, int\*\* G, int\* numV, Node\*\* start) {

Node\* sw = start[v];

numV[v] = 1;

while (sw != NULL) {

if (G[v][sw->data] == 1) {

if (numV[sw->data] == 0) {

dist[sw->data] = dist[v] + 1;

DFSDL(num, sw->data, dist, G, numV, start);

}

}

sw = sw->next;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int num = 0;

Node\* Cell = NULL;

printf("Кол-во ребер: ");

scanf\_s("%d", &num);

int\*\* G = (int\*\*)malloc(num \* sizeof(int\*));

int\* dist = (int\*)malloc(num \* sizeof(int));

int\* numV = (int\*)malloc(num \* sizeof(int));

Node\*\* start = (Node\*\*)malloc(num \* sizeof(Node\*));

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

G[i] = (int\*)malloc(num \* sizeof(int));

start[i] = (Node\*)malloc(num \* sizeof(Node));

dist[i] = -1;

numV[i] = 0;

}

srand(time(NULL));

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

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

if (i == j) {

G[i][j] = 0;

}

else {

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

G[j][i] = G[i][j];

}

}

}

printf("\n Матрица смежности \n");

printf("\n ");

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

printf(" %c", (i + 97));

}

printf("\n ");

for (int i = 0; i < num \* 3; i++) {

printf("\_");

}

printf("\n");

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

printf(" %c | ", (i + 97));

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

printf("%d ", G[i][j]);

}

printf("\n");

}

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

BFSD(G, dist, i, num);

printf("\n%c: ", i + 97);

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

printf(" %d", dist[j]);

dist[j] = -1;

}

}

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

Cell = NULL;

start[i] = NULL;

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

if (G[i][j] == 1) {

Node\* newCell = (Node\*)malloc(sizeof(Node));

newCell->data = j;

newCell->next = NULL;

if (Cell != NULL) Cell->next = newCell;

else start[i] = newCell;

Cell = newCell;

}

}

}

printf("\n\n Список смежности \n");

printf("\n ");

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

Node\* sw = (Node\*)malloc(sizeof(Node));

sw = start[i];

printf("%c:", i + 97);

while (sw != NULL) {

printf(" %c", sw->data + 97);

sw = sw->next;

}

free(sw);

printf("\n ");

}

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

BFSDL(dist, i, num, start);

printf("\n%c: ", i + 97);

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

printf(" %d", dist[j]);

dist[j] = -1;

}

}

printf("\n\n Задание 2:\n");

printf("\n");

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

dist[j] = 0;

}

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

DFSD(num, i, dist, G, numV);

printf("\n%c: ", i + 97);

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

printf(" %d", dist[j]);

dist[j] = 0;

numV[j] = 0;

}

}

printf("\n");

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

DFSDL(num, i, dist, G, numV, start);

printf("\n%c: ", i + 97);

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

printf(" %d", dist[j]);

dist[j] = 0;

numV[j] = 0;

}

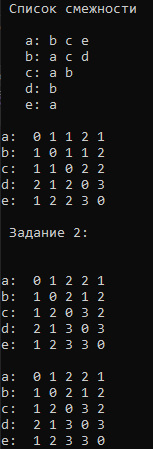
}

return 0;

# }

# Результат работы программы

Результаты работы программы показаны на рисунке 1.



**Рисунок 1 – Результат работы программы**

# Вывод

В ходе выполнения лабораторной работы была разработана программа, в который был реализован алгоритм поиск расстояний в графе.