Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний в графе»

Выполнили студенты группы 20ВВ3:

Мальков И.А.

Педай Н.Д.

Приняли:

Митрохин М.А.

Юрова О.В.

Пенза 2021

**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 поиска

расстояний, реализованную в соответствии с приведенным выше описанием.

3.\* Реализуйте процедуру поиска расстояний 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.

**Ход работы:**

1. Код программы

#include <stdio.h>

#include <iostream>

#include <locale.h>

#include <stdlib.h>

#include <time.h>

#include <queue>

using namespace std;

typedef struct List {

int Data;

struct List\* next;

};

List\* init(int num)

{

List\* node = (List\*)malloc(sizeof(List));

node->Data = num;

node->next = NULL;

return node;

}

List\* add(List\* lst, int num)

{

struct List\* temp, \* p;

temp = (List\*)malloc(sizeof(List));

p = lst->next;

lst->next = temp;

temp->Data = num;

temp->next = p;

return temp;

}

void initialize(int n, List\*\* vrt)

{

int i;

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

{

vrt[i] = init(i);

}

}

void list(int n, int i, int\*\* A1, List\* lst)

{

int j;

for (j = n; j >= 0; j--)

{

if (A1[i][j] == 1)

add(lst, j);

}

}

void result(List\*\* vrt, int\*\* A1, int n)

{

int i;

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

{

list(n, i, A1, vrt[i]);

}

}

void print(List\*\* vrt, int n)

{

int i, j;

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

{

cout << endl;

List\* x = vrt[i];

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

{

if (vrt[i] != NULL)

{

cout << vrt[i]->Data<< " ";

vrt[i] = vrt[i]->next;

}

}

vrt[i] = x;

}

cout << endl;

}

void BFSD\_list(int\* A2, int n, List\*\* vrt, int a3)

{

queue<int> q;

List\* h;

int i = a3;

q.push(a3); // помещаем в очередь вершину

A2[i] = 0;

cout << "BFSD list result: ";

while (!q.empty()) // пока не пуст

{

i = q.front(); // извлекаем вершину

cout << i << " ";

q.pop();

h = vrt[i];

//for (int j = n - 1; j >= 0; j--) // проверяем для нее все смежные вершины

while(vrt[i] != 0)

{

if(A2[vrt[i]->Data] == -1)

{

A2[vrt[i]->Data] = A2[i] + 1;

q.push(vrt[i]->Data);

}

vrt[i] = vrt[i]->next;

}

vrt[i] = h;

}

cout << endl;

cin.get();

}

void BFSD(int\*\* A1, int n, int\* A2, int a0)

{

queue<int> q;

q.push(a0); // помещаем в очередь вершину

A2[a0] = 0;

cout << "BFSD result: ";

while (!q.empty()) // пока не пуст

{

a0 = q.front(); // извлекаем вершину

q.pop();

cout << a0 << " ";

//for (int j = n - 1; j >= 0; j--) // проверяем для нее все смежные вершины

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

{

if (A1[a0][j] == 1 && A2[j] == -1) // если вершина смежная и не обнаружена

{

q.push(j); // добавляем ее в очередь

A2[j] = A2[a0] + 1; // отмечаем вершину как обнаруженную

}

}

}

cin.get();

}

void DFSD\_list(int\* A2, int n, int a4, List\*\* vrt, int k)

{

int j;

List\* p = vrt[a4];

A2[a4] = k;

cout << a4 << " ";

while (p != NULL)

{

if (A2[p->Data] == -1)

{

DFSD\_list(A2, n, p->Data, vrt, k + 1);

}

if (A2[p->Data] > k)

{

DFSD\_list(A2, n, p->Data, vrt, k + 1);

}

p = p->next;

}

}

void DFSD(int\* A2, int n, int a1, int\*\* A1, int s)

{

A2[a1] = s;

cout << a1 << " ";

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

{

if (A1[a1][j] == 1 && A2[j] == -1)

{

DFSD(A2, n, j, A1, s + 1);

}

if (A1[a1][j] == 1 && A2[j] > s)

{

DFSD(A2, n, j, A1, s + 1);

}

}

}

int\*\* create(int n)

{

int\*\* A1, k = 1;

A1 = (int\*\*)malloc(n \* sizeof(int\*));

srand(time(NULL));

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

{

A1[i] = (int\*)malloc(n \* sizeof(int));

A1[i][i] = 0;

for (int j = k; j < n; j++)

{

if (rand() % 100 > 65)

{

A1[i][j] = 0;

}

else {

A1[i][j] = 1;

}

}

k++;

}

k = 1;

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

{

for (int j = k; j < n; j++)

{

A1[j][i] = A1[i][j];

}

k++;

}

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

cout << "V" << i << " ";

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

{

cout << endl;

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

{

cout << A1[i][j] << " ";

}

}

return A1;

}

int main()

{

clock\_t start, end;

int n, \*\* t, a0, a1, a3, a4;

cout << "Enter array size: ";

cin >> n;

List\*\* vrt = (List\*\*)malloc(n \* sizeof(List\*));

int\* A2 = (int\*)malloc(n \* sizeof(int));

t = create(n);

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

A2[i] = -1;

cout << endl;

cout << "Enter vertex: ";

cin >> a0;

start = clock(); // старт таймера

BFSD(t, n, A2, a0);

end = clock(); // остановка таймера

float diff1 = (end - start);

cout << endl;

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

cout << A2[i] << endl;

cout << endl << "time is: " << diff1 << endl;

cout << endl;

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

A2[i] = -1;

cout << "Enter vertex: ";

cin >> a1;

int s = 0;

cout << "DFSD result: ";

start = clock(); // старт таймера

DFSD(A2, n, a1, t, s);

end = clock(); // остановка таймера

float diff2 = (end - start);

cout << endl;

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

cout << A2[i] << endl;

cout << endl << "time is: " << diff2 << endl;

initialize(n, vrt);

result(vrt, t, n);

cout << endl;

cout << "list:";

print(vrt, n);

cout << endl;

cout << "Enter vertex: ";

cin >> a3;

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

A2[i] = -1;

start = clock(); // старт таймера

BFSD\_list(A2, n, vrt, a3);

end = clock(); // остановка таймера

float diff3 = (end - start);

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

cout << A2[i] << endl;

cout << endl;

cout << "time is: " << diff3 << endl;

cout << endl;

cout << "Enter vertex: ";

cin >> a4;

cout << "DFSD list result: ";

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

A2[i] = -1;

int k = 0;

start = clock(); // старт таймера

DFSD\_list(A2, n, a4, vrt, k);

end = clock(); // остановка таймера

float diff4 = (end - start);

cout << endl;

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

cout << A2[i] << endl;

cout << endl << "time is: " << diff4 << endl;

cout << endl;

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

free(t[i]);

free(t);

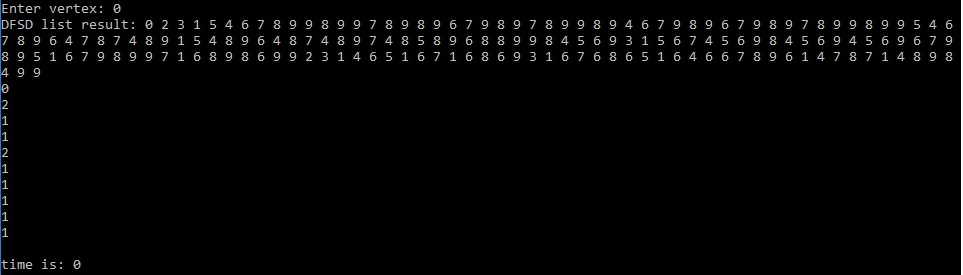
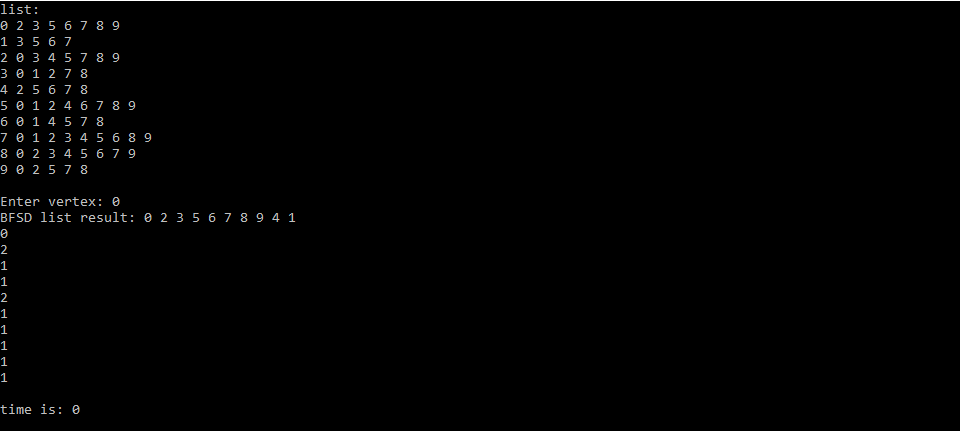
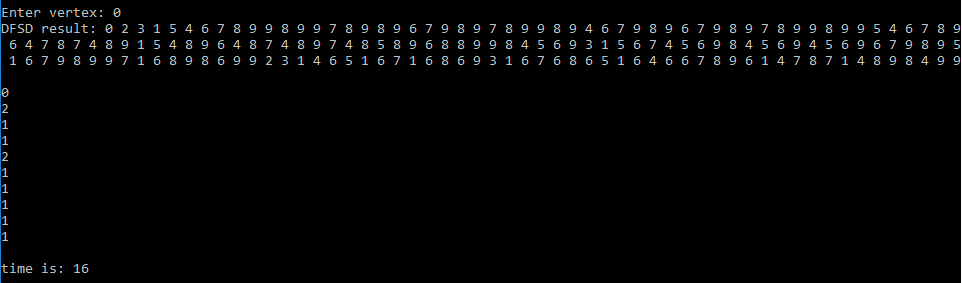
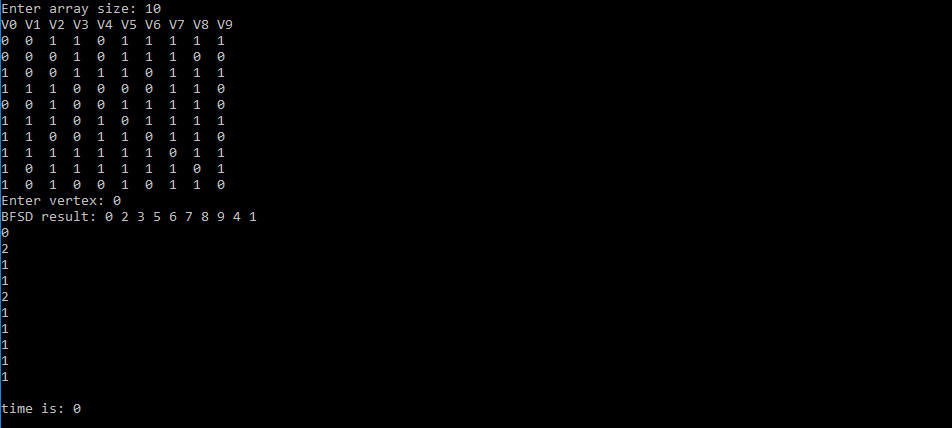
free(A2);

system("pause");

return 0;

}

1. Результат работы программы.



Матрица 100\*100

t(обход в ширину матрицы смежности) = 16 мс

t(обход в глубину матрицы смежности) = 20962 мс

t(обход в ширину списка смежности) = 16 мс

t(обход в глубину списка смежности) = 20914 мс

Вывод: в ходе выполнения данной лабораторной работы разработали программу, согласно заданию.