Министерство науки и высшего образования

Российской федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Новосибирский Государственный Технический Университет»

Кафедра теоретической и прикладной информатики

Курсовая работа по дисциплине

«Информационные технологии и основы программирования»

Факультет: прикладной математики и информатики

Группа: ПМИ-12

Студент: Швадченко Артём Валерьевич

Преподаватель: Еланцева Ирина Леонидовна

Новосибирск, 2022

1. **Условие задачи**

По графу G построить граф G’ следующим образом: вершины G’ это ребра G; вершины G’ смежны, если рёбра G смежны; найти все вершины, расстояние от которых до выделенной вершины равно 2.

1. **Анализ задачи**
2. Исходные данные задачи

Содержимое текстового файла in.txt:

В первой строке вводится размерность матрицы смежности графа G;

Далее в каждой строке записывается нижняя треугольная матрица смежности G (значения матрицы записываются через пробел, а строки матрицы вводятся с новой строки каждая).

Также после построения графа G’ пользователю будет предложено ввести в консоль номер вершины, от которой в дальнейшем найдутся вершины, расстояние до которых 2.

*Пример входных данных файла in.txt:*

4

0

1 0

1 0 0

0 1 1 0

*Пример входных данных консоли:*

2

1. Результат

Результат работы программы будет записан в текстовый файл out.txt.

В случае обнаружения во входном файле ошибок, то на экран будет выведено сообщение о существовании ошибки. Также на экран будет выводиться сообщение о состоянии выходных данных: удалось ли получить ответ на задачу или же нет.

*Пример выходных данных файла out.txt:*

Матрица смежности рёберного графа:

0 1 1 0

1 0 0 1

1 0 0 1

0 1 1 0

От вершины 2 расстояние 2 до следующих вершин:

3

1. Решение

Математическая модель – граф, невзвешенный, неориентированный, связный, непомеченный, простой.

*Определение:*

Невзвешенный граф – граф, в котором все вершины и ребра (дуги) равнозначны. В данной задаче для построения графа G’ и нахождения вершин до которых расстояние от выделенной равно 2 не нужно ставить в соответствие каждому ребру или вершине некоторую информацию

Неориентированный граф – граф, ребра которого не имеют направления.

В данной задаче не имеет значения направление рёбер.

Связный граф – граф, не допускающий вершин, не связанных с другими вершинами.

Непомеченный граф – граф, между вершинами которого нет различий.

Простой граф – граф, в котором нет кратных рёбер и петлей.

*Анализ графа:*

На этапе считывания данных из файла, создадим динамический массив, в котором элементов столько же, сколько во вводимой нижней треугольной матрице смежности. Далее создадим динамический массив, который заполним матрицей инцидентности исходного графа.  
По созданной матрице инцидентности строится нижняя треугольная матрица смежности рёберного графа G’ и записывается в массив.

*Анализ входных данных:*

1. Проверка корректности введённых элементов матрицы G. Если хотя бы один элемент будет одновременно неравен нулю и единице, то на экран выведется сообщение о безуспешной работе программы, а в выходной файл соответствующее сообщение, т. к. программа работает с невзвешенным графом.
2. Проверка на соответствие введенной размерности матрицы смежности и фактической размерности матрицы. Если эти размерности будут отличаться, то на экран выведется сообщение о безуспешной работе программы, а в выходной файл соответствующее сообщение.
3. Проверка наличия нулей на главной диагонали матрицы смежности графа G. Если на диагонали будет стоять хотя бы одна единица, то на экран выведется сообщение о безуспешной работе программы, а в выходной файл соответствующее сообщение, т. к. граф простой.
4. Проверка вводимой пользователем вершины. Если пользователь в консоль введёт несуществующую вершину, то на экран выведется сообщение о безуспешной работе программы, а в выходной файл соответствующее сообщение.

*Формальная постановка задачи*

По невзвешенному, неориентированному, связному, непомеченному, простому графу построить рёберный граф и для введённой вершины найти вершины, до которых расстояние равно двум.

Данную задачу можно разделить на две основные подзадачи:

1. Построение рёберного графа;
2. Нахождение вершин, до которых расстояние 2 от введённой.

*Алгоритм решения поставленных подзадач*

Чтобы приступить к решению поставленных подзадач, необходимо чтобы не было обнаружено ошибок, описанных выше.

1. Построение рёберного графа.

Сначала создаем двумерный массив *a*, который будет являться нижней треугольной матрицей смежности графа G. Затем по мере считывания данных из файла этот массив будет заполняться. Во время заполнения проверяем, чтобы введённые элементы были равны нулю или единице (пункт 1 анализа входных данных). После заполнения массива проверяется правильность введения размерности матрицы (пункт 2 анализа входных данных) и проверяется наличие на главной диагонали нулей (пункт 3 анализа входных данных).

После заполнения нижней треугольной матрицы смежности, если количество рёбер равно нулю, то на экран выведется сообщение о успешной работе программы, а в выходной файл сообщение, что рёберного графа не существует.

Если же количество рёбер будет больше нуля, то создаётся двумерный массив *b*, в который заполняется матрицей инцидентности графа G. Далее создаётся двумерный массив *c,* который заполняется нижней треугольной матрицей смежности рёберного графа G’ через матрицу инцидентности. А в файл выводится полная матрица смежности графа G’.

Таким образом получаем ответ на подзадачу №1.

1. Нахождение вершин, до которых расстояние 2 от введённой.

После вывода матрицы смежности рёберного графа G’ в файл, пользователю будет предложено ввести номер вершины. Если будет введена несуществующая вершина, то программа завершит свою работу с соответствующим сообщением (пункт 4 анализа входных данных).

Если будет введена существующая вершина, то будет создан одномерный массив *f* ,который будет хранить информацию о вершинах графа G’, до которых будет найден путь длиной два (если от введённой вершины до вершины *i* будет найден путь длиной два, то значение в ячейке *f[i-1]* будет увеличено на один).

Программа будет искать вершины, до которых путь длиной 2 от введённой, следующим образом: если будет введена вершина *k*, то программа будет проходить в массиве, содержащем нижнюю треугольную матрицу смежности графа G’, строку с индексом *k-1.* Если значение ячейки будет равно единице, то программа пройдёт столбец массива, в котором находится данная ячейка, и будет искать ячейки, в которых значение равно единице. Если значение ячейки с индексом *j* будет равно единице, то в ячейке *f[j]* будет увеличено значение на единицу. После прохождения столбца программа продолжит проходить *k-1* строку. Так как мы работаем с нижней треугольной матрицей, то при прохождении *k-1* строки, если у элемента a[k-1][j] матрицы смежности графа G’ k-1 будет меньше j, то будет рассмотрен элемент a[j][k-1], т. к. матрица смежности симметрична.

После окончания заполнения массива *f* будут выведены те вершины, для которых соответствующие значения массива *f* не будут равны 0, или, что таких вершин нет.

Замечание. Мы работаем только с простыми путями, поэтому мы не можем вывести, что от введённой вершины до неё самой расстояние равно двум.

Таким образом получаем ответ на подзадачу №2.

1. **Структуры данных, используемых для представления исходных данных и результатов задачи**
2. Входные данные:

Внешнее представление входных данных:

* Целое число – размерность матрицы смежности G;
* Нижняя диагональная матрица смежности – между элементами одной строки – пробелы, каждая строка матрицы, начиная с новой строки в файле;
* Целое число – вершина, от которой необходимо найти другие вершины, до которых расстояние 2.

Внутреннее представление входных данных:

* Размерность матрицы смежности G - переменная типа int (int d);
* Матрица смежности – двумерный массив с количеством элементов d\*d/2.
* Вводимая вершина – переменная типа int (int ver)

1. Выходные данные:

Внешнее представление выходных данных:

* Текстовые сообщения на экран об успешности/безуспешности открытия вводного файла;
* Текстовые сообщения на экран об успешном/безуспешном получении ответа на поставленную задачу.
* Вывод в текстовый файл сообщения: «Таблица смежности рёберного графа: “*матрица смежности G’ ”;* От вершины *ver* расстояние 2 до следующих вершин: “*номера вершин* ”» - в случае нахождения ответа на поставленную задачу.

Внутреннее представление выходных данных:

* “*матрица смежности G’ ”* из внешнего представления – двумерный массив, в котором хранится нижняя треугольная матрица смежности графа G’;
* *ver* из внешнего представления – переменная типа int;
* “*номера вершин* ” из внешнего представления – вершины, у которых числовые значения в массиве *f* не равны нулю, т.е. вершины, до которых расстояние от введённой равно двум.

1. **Укрупненный алгоритм решения задачи**
2. Укрупненный алгоритм решения задачи

{

Открываем файл;

Считываем данные из файла;

Заполняем двумерный массив, хранящий нижнюю треугольную матрицу смежности графа G;

Проверяем корректный ввод нижней треугольной матрицы смежности и её размерности;

Создаем и заполняем матрицу инцидентности;

Создаём и заполняем нижнюю треугольную матрицу смежности графа G’;

Выводим матрицу смежности графа G’ в файл;

Считываем вершину:

Находим вершины, до которых расстояние 2 от введённой;

Выводим найденные вершины в файл;

}

1. Укрупнённый алгоритм нахождения нижней треугольной матрицы смежности рёберного графа G’

{

Создаём массив, который заполняем нижней треугольной матрицей смежности графа G;

Создаём массив, который заполняем матрицей инцидентности графа G по его матрице смежности;

Создаём массив, который заполняем нижней треугольной матрицей смежности графа G’ через матрицу инцидентности графа G;

}

1. Укрупненный алгоритм поиска вершин, до которых расстояние 2 от введённой;

{

Вводим вершину;

Проверяем существование введённой вершины;

Создаём и заполняем массив хранящий информацию о вершинах, до которых расстояние 2;

Выводим найденные вершины в файл;

}

1. **Структура программы**

Текст программы разбит на два модуля.

Main.cpp – содержит главную функцию main осуществляющую ввод и вывод информации, а также вызывает вспомогательные функции.

Source.cpp - содержит функции осуществляющие правильную работу основного модуля.

1. Состав модуля main.cpp::

* Назначение: чтение данных из файла, определение входных данных, анализ входных данных, реализация алгоритма решения задачи, организация связи с пользователем, определение выходных данных, запись данных в файл.
* Прототип функции: int main()

1. Состав модуля Source.cpp:

Функция *MatCreate*:

* Назначение: создание нижних треугольных матриц смежности.
* Прототип функции: void MatCreate(int\*\* hw, int y, int x);.
* Параметры: *hw* – массив, *y* – размерность матрицы, x – значение отвечающее за заполнение нулями созданной матрицы.

Функция *MatInc*:

* Назначение: создание матрицы инцидентности.
* Прототип функции: void MatInc(int\*\* hw, int\*\* wh, int y, int z).
* Параметры: *hw* – нижняя треугольная матрица смежности графа G, *wh* – матрица инцидентности, *y* – размерность матрицы смежности графа G, *z* – количество вершин в графе G’.

Функция *MatRebr*:

* Назначение: создание матрицы смежности рёберного графа G’.
* Прототип функции: void MatRebr(int\*\* hw, int y, int z, int\*\* wh).
* Параметры: *hw* – нижняя треугольная матрица смежности графа G’, *y* – размерность матрицы смежности графа G, *z* – количество вершин в графе G’, *wh* – матрица инцидентности графа G.

Функция *MatVer*:

* Назначение: поиск вершин, до которых расстояние равно двум до введённой.
* Прототип функции: void MatVer(int\*\* hw, int\*\* wh, int y, int x).
* Параметры: *wh* – нижняя треугольная матрица смежности графа G’, *hw* – матрица, хранящая информацию о вершинах, до которых расстояние 2 от введённой, *y* – номер вершины, уменьшенный на один, *x* – количество вершин в графе G’.

1. **Текст программы на языке Си (С++)**
2. Заголовочный файл header.h

void MatCreate(int\*\* hw, int y, int x);

void MatInc(int\*\* hw, int\*\* wh, int y, int z);

void MatRebr(int\*\* hw, int y, int z, int\*\* wh);

void MatVer(int\*\* hw, int\*\* wh, int y, int x);

1. Текст модуля Source.cpp

#include "header.h"

//создание нижних треугольных матриц

void MatCreate(int\*\* hw, int y, int x)

{

int z = 1;

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

{

hw[i] = new int[z];

z++;

}

int f = 1;

if (x)

{

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

{

for (int j = 0; (j < f) and (f <= y); j++)

{

hw[i][j] = 0;

}

f++;

}

}

}

//создание и заполнение матрицы инцидентности

void MatInc(int\*\* hw, int\*\* wh, int y, int z)

{

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

{

wh[i] = new int[z];

}

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

{

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

{

wh[i][j] = 0;

}

}

for (int i = 0, k = 0; i < y; i++)

for (int j = i + 1; j < y; j++)

if (i > j)

{

if (hw[i][j])

{

wh[i][k] = wh[j][k] = 1;

k++;

}

}

else

{

if (hw[j][i])

{

wh[i][k] = wh[j][k] = 1;

k++;

}

}

}

//заполнение нижней треугольной матрицы смежности графа G'

void MatRebr(int\*\* hw, int y, int z, int\*\* wh)

{

for (int v = 0; v < y; v++) {

for (int e = 0; e < z; e++) {

if (wh[v][e]) {

for (int e1 = e + 1; e1 < z; e1++) {

if (wh[v][e1]) {

if (e>e1)

hw[e][e1] = 1;

else

hw[e1][e] = 1;

}

}

}

}

}

}

// заполнение массива, хранящего информацию о вершинах, до которых расстояние 2

void MatVer(int\*\* hw, int\*\* wh, int y, int x)

{

int m = 0;

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

{

if (y>j)

{

if (wh[y][j] == 1)

{

m = j;

for (int l = 0; l < x; l++)

{

if (l > m)

{

if (wh[l][m] == 1)

{

hw[l]++;

}

}

else

if (wh[m][l] == 1)

{

hw[l]++;

}

}

}

}

else

{

if (wh[j][y] == 1)

{

m = j;

for (int l = 0; l < x; l++)

{

if (l > m)

{

if (wh[l][m] == 1)

{

hw[l]++;

}

}

else

if (wh[m][l] == 1)

{

hw[l]++;

}

}

}

}

}

}

1. Содержание модуля main.cpp

#include <iostream>

#include <fstream>

#include "header.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "ru");

fstream file;

file.open("in.txt");

ofstream fout;

fout.open("out.txt");

if (!file.is\_open())

cout << "Error!" << endl;

else

{

int d, q, i, j, r, ver;

char ch = 0;

cout << ch;

int sum = 0;

bool flag = true;

r = 0;

q = 1;

ver = 0;

cout << "Successful opening enter file!" << endl;

file >> d;

if (d > 0) //проверка введения неотрицательной размерности

{

int\*\* a = new int\* [d];

MatCreate(a, d, r);

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

{

for (j = 0; (j < q) and (j < d); j++)

{

file >> a[i][j];

// заполнение матрицы смежности

sum = sum + a[i][j];

if (((a[i][j] != 0) and a[i][j] != 1))

{   
// проверка матрицы смежности на корректные значения

cout << "Output passed unsuccessful!" << endl;

fout << "Матрица смежности введена неверно!" << endl;

flag = false;

break;

}

}

q++;

if (!flag)

break;

}

if (flag)

{

file >> ch;//проверка размерности

if (ch != '\0')

{

cout << "Output passed unsuccessful!" << endl;

fout << "Матрица смежности введена неверно!" << endl;

}

else

{

if ((sum == 0) and (d == 1))

{

cout << "Output passed successful!" << endl;

fout << "Рёберного графа для введённого графа не существует!" << endl;

}

else

{

if (flag)

{

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

{

if (a[i][i] != 0)   
// проверка наличия на диагонали только нулей

{

cout << "Output passed unsuccessful!" << endl;

fout << "Матрица смежности введена неверноd!" << endl;

flag = false;

break;

}

}

}

if (flag)

{

r = 1;

int\*\* b = new int\* [sum];

MatInc(a, b, d, sum);  
//создание и заполнение матрицы инцидентности

int\*\* c = new int\* [sum];

MatCreate(c, sum, r);

MatRebr(c, d, sum, b);  
//создание и заполнение нижней диагональной матрицы смежности рёберного графа

r = 0;

fout << "Матрица смежности рёберного графа:" << endl;

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

{

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

{

if (i >= j)

{

fout << c[i][j] << " ";//вывод матрицы смежности графа G' в файл

r = r + c[i][j];

}

else

{

fout << c[j][i] << " ";

r = r + c[j][i];

}

}

fout << endl;

}

cout << "Output passed successful!" << endl;

if (sum > 2)

{

cout << "Введите номер вершины от 1 до " << sum << endl;

cin >> ver; // ввод вершины

flag = true;

for (i = 1; i < sum + 1; i++)

{

if (ver == i) // проверка существования вершины

{

flag = true;

break;

}

else

flag = false;

}

if (flag)

{

ver--;

int\*\* f = new int\* [r];   
// создание массива, для хранения информации о вершинах до которых расстояние 2

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

f[j] = 0;

MatVer(f, c, ver, sum); //заполнение массива

cout << "Output of the vertices successful!" << endl;

q = 0;

f[ver] = 0;

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

if (f[j] != 0)

{

q++;

}

if (q > 0)

{

fout << "От вершины " << ver + 1 << " расстояние 2 до следующих вершин:" << endl;

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

if (f[j] != 0)

{

fout << j + 1 << " "; // вывод найденных вершин в файл

}

}

else

{

fout << "У вершины " << ver + 1 << " нет вершин, до которых расстояние 2.";

}

}

else

{

cout << "Output of the vertices unsuccessful!" << endl;

fout << "Вершины " << ver << " в рёберном графе нет!" << endl;

}

}

else

{

cout << "Output of the vertices successful!" << endl;

fout << "В графе меньше двух рёбер!";

}

}

}

}

}

}

else

{

cout << "Output passed unsuccessful!" << endl;

fout << "Введена некорректная размерность!" << endl;

}

}

file.close();

fout.close();

}

1. **Набор тестов**
2. В данном тесте происходит проверка пункта 1 анализа входных данных (корректность введённых элементов матрицы G):

|  |  |  |
| --- | --- | --- |
| Входные данные | Сообщения в консоль | Вывод в файл |
| 3  0  1 0  2 1 0 | Successful opening enter file!  Output passed unsuccessful! | Матрица смежности введена неверно! |

1. В данном тесте происходит проверка пункта 2 анализа входных данных  
   (соответствие введенной размерности матрицы смежности и фактической размерности матрицы):

|  |  |  |
| --- | --- | --- |
| Входные данные | Сообщения в консоль | Вывод в файл |
| 3  0  1 0 | Successful opening enter file!  Output passed unsuccessful! | Матрица смежности введена неверно! |

1. В данном тесте происходит проверка пункта 3 анализа входных данных  
   (наличие нулей на главной диагонали матрицы смежности графа G):

|  |  |  |
| --- | --- | --- |
| Входные данные | Сообщения в консоль | Вывод в файл |
| 3  0  1 1  1 1 0 | Successful opening enter file!  Output passed unsuccessful! | Матрица смежности введена неверно! |

1. В данном тесте происходит проверка пункта 4 анализа входных данных (проверка вводимой пользователем вершины):

|  |  |  |
| --- | --- | --- |
| Входные данные | Сообщения в консоль | Вывод в файл |
| 3  0  1 0  1 1 0 | Successful opening enter file!  Output passed successful!  Введите номер вершины от 1 до 3  4  Output of the vertices unsuccessful! | Матрица смежности рёберного графа:  0 1 1  1 0 1  1 1 0  Вершины 4 в рёберном графе нет! |

1. В данном тесте проверяем корректность работы программы, если в рёберном графе G’ будет менее двух рёбер:

|  |  |  |
| --- | --- | --- |
| Входные данные | Сообщения в консоль | Вывод в файл |
| 2  0  1 0 | Successful opening enter file!  Output passed successful!  Output of the vertices successful! | Матрица смежности рёберного графа:  0  В графе меньше двух рёбер! |

1. В данном тесте проверяем корректность работы программы, если граф G содержит одну вершину:

|  |  |  |
| --- | --- | --- |
| Входные данные | Сообщения в консоль | Вывод в файл |
| 1  0 | Successful opening enter file!  Output passed successful! | Рёберного графа для введённого графа не существует! |

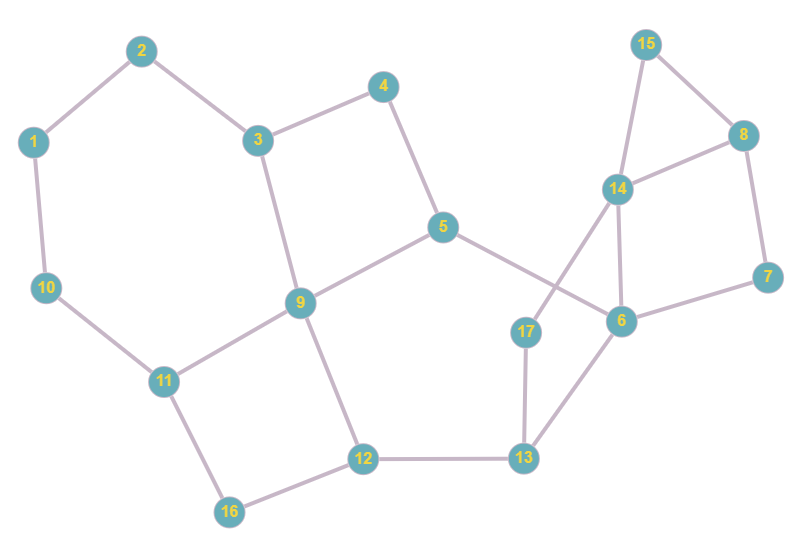
1. В данном тесте проверяем корректность работы программы, если у введённой вершины нет вершин, до которых расстояние 2:

|  |  |  |
| --- | --- | --- |
| Входные данные | Сообщения в консоль | Вывод в файл |
| 4 0 1 0 0 1 0 1 0 0 0 | Successful opening enter file!  Output passed successful!  Введите номер вершины от 1 до 3  1  Output of the vertices successful! | Матрица смежности рёберного графа:  0 1 1 1 0 0 1 0 0  У вершины 1 нет вершин, до которых расстояние 2. |

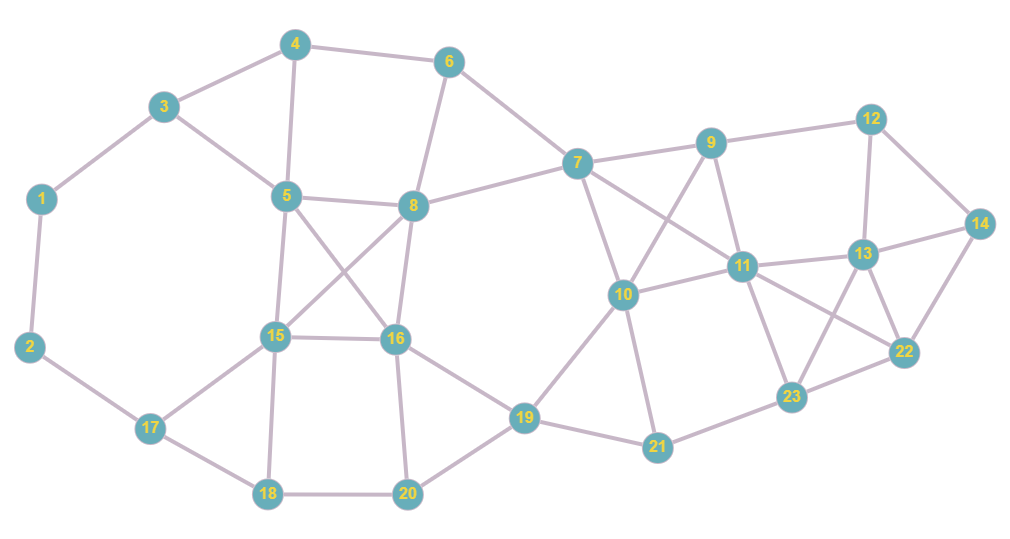
1. В данном тесте проверяем корректность работы программы с большим количеством вершин графа G:

|  |  |
| --- | --- |
| Входные данные | Сообщения в консоль |
| 17  0  1 0  0 1 0  0 0 1 0  0 0 0 1 0  0 0 0 0 1 0  0 0 0 0 0 1 0  0 0 0 0 0 0 1 0  0 0 1 0 1 0 0 0 0  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 1 1 0  0 0 0 0 0 0 0 0 1 0 0 0  0 0 0 0 0 1 0 0 0 0 0 1 0  0 0 0 0 0 1 0 1 0 0 0 0 0 0  0 0 0 0 0 0 0 1 0 0 0 0 0 1 0  0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 | Successful opening enter file!  Output passed successful!  Введите номер вершины от 1 до 23  23  Output of the vertices successful! |
| Вывод в файл: |  |
| Матрица смежности рёберного графа:  0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0  1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0  0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0  0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0  0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 1 1  0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1  0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0  0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0  0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0  0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0  0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0  0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1  0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1  0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0  От вершины 23 расстояние 2 до следующих вершин:  7 9 10 11 12 13 14 19 22 |

Граф G



Граф G’



1. **На всех тестах программа правильно работает и выдаёт ожидаемый результат.**
2. **Источники информации**
3. Хиценко В.П. Структуры данных и алгоритмы: методические указания к курсовой работе для 1 курса ФПМиИ (направление 010500 – Прикладная математика и информатика, специальность 010503 – Математическое обеспечение и администрирование информационных систем) дневного отделения / В.П. Хиценко, Т.А. Шапошникова. – Новосибирск: Изд-во НГТУ, 2008. – 55 с.
4. <https://prog-cpp.ru/data-graph/>
5. <https://ru.wikipedia.org/wiki/%D0%A0%D1%91%D0%B1%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84>
6. <https://function-x.ru/graphs3_structures.html>