

Министерство образования и науки Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К курсовому проектированию
по курсу «Логика и основы алгоритмизации в инженерных задачах»
на тему «Алгоритм Флойда»

Выполнил студент группы 21вв1.1:

Гришин С.В.

Принял

Ахифьев И.В.

Отлично

28.12.22г.

Пенза 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ 

«30» сентября 2022

ЗАДАНИЕ

на курсовое проектирование по курсу

Логика и основы алгоритмизации в инженерных задачах
Студенту Тришину Сергею Вадимовичу Группа 21БВТ
Тема проекта реализация алгоритма Флойда

Исходные данные (технические требования) на проектирование

Разработка алгоритмов и программного обеспечения в соответствии с данным заданием курсовое проекта.
Пояснительная записка должна содержать:

- 1) Постановку задачи.
- 2) Теоретическую часть задания.
- 3) Описание алгоритма поставленной задачи.
- 4) Пример ручного расчёта задачи и вычислений.
- 5) Описание самой программы.
- 6) Иллюстрации.
- 7) Список литературы.
- 8) Листинг программы.
- 9) Результаты работы программы.

Объем работы по курсу

1. Расчетная часть

Ручной расчёт работы алгоритма

2. Графическая часть

Схема алгоритма в формате "блок-схема"

3. Экспериментальная часть

Тестирование программы;

Результат работы программы на тестовых данных

Срок выполнения проекта по разделам

- 1 Исследование теоретической части курсового проекта
- 2 Разработка алгоритмов программы.
- 3 Разработка программы
- 4 Тестирование и завершение разработки программы
- 5 Оформление пояснительной записки.
- 6
- 7
- 8

Дата выдачи задания " 30 " сентября

Дата защиты проекта " " "

Руководитель

Алифьев И.В.

Задание получил

" 30 " сентября

2022 г.

Студент

Гришин Сергей Вадимович

Реферат.....	5
Введение	6
Постановка задачи.....	7
Теоретическая часть задания	8
Описание алгоритма программы	9
Описание программы	11
Тестирование	14
Ручной расчёт задачи	17
Заключение	18
Список литературы	19
Листинг программы	20

Реферат

Отчет 28 стр

Цель исследования – разработка программы, способную находить кратчайшие пути из одной точки в другую.

В работе рассмотрен алгоритм Флойда, на основе которого находится матрица кратчайших путей.

Введение

алгоритм Флойда (также известный как алгоритм Флойда-Уоршелла, алгоритм Роя–Уоршелла, алгоритм Роя–Флойда или алгоритм WFI) – это алгоритм поиска кратчайших путей во взвешенном графе с положительным или отрицательным весом ребер (но без отрицательных циклов). За одно выполнение алгоритма будут найдены длины (суммарные веса) кратчайших путей между всеми парами вершин. Хотя он не возвращает детали самих путей, можно реконструировать пути с помощью простых модификаций алгоритма.

В качестве среды разработки мною была выбрана среда Microsoft Visual Studio 2022, язык программирования – C#.

Постановка задачи

Требуется разработать программу, которая построит матрицу кратчайших путей.

Исходный граф в программе должен задаваться матрицей смежности. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы смежности. После обработки этих данных на экран должна выводиться матрица смежности графа и матрица кратчайших путей.

Устройство ввода – клавиатура.

Теоретическая часть задания

Рассмотрим граф G с вершинами V , пронумерованными от 1 до N . Алгоритм Флойда сравнивает все возможные пути через граф между каждой парой вершин. Он может сделать это за $O(n^3)$ сравнений в графе, даже если в графе может быть до V^2 ребер, и каждая комбинация ребер проверяется. Это достигается путем постепенного улучшения оценки кратчайшего пути между двумя вершинами, пока оценка не станет оптимальной.

Далее рассмотрим функцию $\text{shortestPath}(i,j,k)$, которая возвращает кратчайший возможный путь от i до j с использованием вершин только из множества $\{1,2,\dots,k\}$ в качестве промежуточных точек на этом пути. Теперь, учитывая эту функцию, наша цель - найти кратчайший путь от каждого i до каждого j , используя любую вершину в $\{1,2,\dots,k\}$.

Для каждой из этих пар вершин $\text{shortestPath}(i,j,k)$ может быть либо

- (1) путь, который не проходит через k (использует только вершины из набора $\{1,\dots,k-1\}$),

или

- (2) путь, который проходит через k (от i до k и затем от k до j , в обоих случаях используются только промежуточные вершины в $\{1,\dots,k-1\}$).

Мы знаем, что лучший путь от i до j , это путь который использует только вершины с 1 по $k-1$, определяется как $\text{shortestPath}(i,j,k-1)$, и ясно, что если бы существовал лучший путь от i до k до j , тогда длина этого пути была бы цепочкой состоящей из самого короткого пути от i до k (только с использованием промежуточных вершин в $\{1,\dots,k-1\}$) и кратчайшего пути от k до j (только с использованием промежуточных вершин в $\{1,\dots,k-1\}$).

Если $w(i,j)$ - вес ребра между вершинами i и j , мы можем определить $\text{shortestPath}(i,j,k)$ в терминах следующей рекурсивной формулой:

базовый случай

$$\text{shortestPath}(i,j,0) = w(i,j)$$

и рекурсивный случай

$$\text{shortestPath}(i,j,k) = \min(\text{shortestPath}(i,j,k-1), \text{shortestPath}(i,j,k-1) + \text{shortestPath}(i,j,k-1)).$$

Эта формула составляет основу алгоритма Флойда. Алгоритм работает, сначала вычисляя $\text{shortestPath}(i,j,k)$ для всех пар (i,j) для $k=1$, а затем $k=2$, и так далее. Этот процесс

продолжается до тех пор, пока $k=N$ не будет найден кратчайший путь для всех пар (i,j) с использованием любых промежуточных вершин.

Описание алгоритма программы

В начале программы создаётся основная матрица(*grf*) и массив матриц для сохранения результата и промежуточных матриц. Далее пользователь, выбрав пункт в меню “Сгенерировать граф”, вводит размер (из-за ограниченного размера монитора и особенностей консоли размер матрицы ограничен от 2 до 14 для “красивого” вывода, при желании данные ограничения можно убрать). После выбирает тип ввода данных (ручной или автоматический). При выборе ручного пользователь в ручную заполнит матрицу смежности графа, а при выборе автоматического, матрица заполнится с помощью генератора случайных чисел. Сразу после построения матрицы смежности активируется функция (*floid*), которая, с помощью алгоритма Флойда, строит матрицу кратчайших путей.

Сама функция представляет собой тройной вложенный цикл, работающий следующим образом:

V – целочисленная переменная, в которой хранится информация о том, какая строка и столбец проходятся циклом

Ras – целочисленная переменная, хранит информацию о размере матрицы

grf – исходный граф

matr – массив матриц, содержащий исходный граф, результат работы алгоритма, и промежуточные матрицы

matr[0]=*grf*

от $v = 0$ до *ras*-1

{

matr(*v*+1) = *matr*(*v*)

от $i = 0$ до *ras* – 1

{

От $j = 0$ до *ras* -1

{

Если($i = v$ или $j = v$) { пропустить итерацию цикла; }

Если($i = j$) { пропустить итерацию цикла; }

Если(*matr*[*v*][*v*,*j*] != 0 и *matr*[*v*][*i*,*v*] != 0)

```

{
    Если(matr[v][i,j] = 0 ){ matr[v+1][i, j] = matr[v][v, j] + matr[v][i, v];}
Иначе
    {
        Если(matr[v][i, j] > matr[v][v, j] + matr[v][i, v])
            {
                matr[v+1][i, j] = matr[v][v, j] + matr[v][i, v];
            }
    }
}
}
}
}

```

Описание программы

Для написания данной программы использован язык программирования

С#. Язык программирования С# - объектно-ориентированный язык программирования общего назначения.

Проект был создан в виде консольного приложения Win32 (Visual C#).

Данная программа является многомодульной, поскольку состоит из нескольких функций: `consize`, `sc`, `floid`, `graf_gen`, `graf_print`.

Работа программы начинается с запроса генерации графа. Если пользователь выбрал сгенерировать граф, то на экран выводится запрос на размер графа. Затем выводятся уведомление о том, что граф успешно создан .

[illegible]

Далее выбрав “Вывести граф” пользователь получен первоначальный граф , где еще не найдены все пути .

[illegible]

```

    }
    Console.Clear();
    consize(w, h);
    goto MenuMain;
}

```

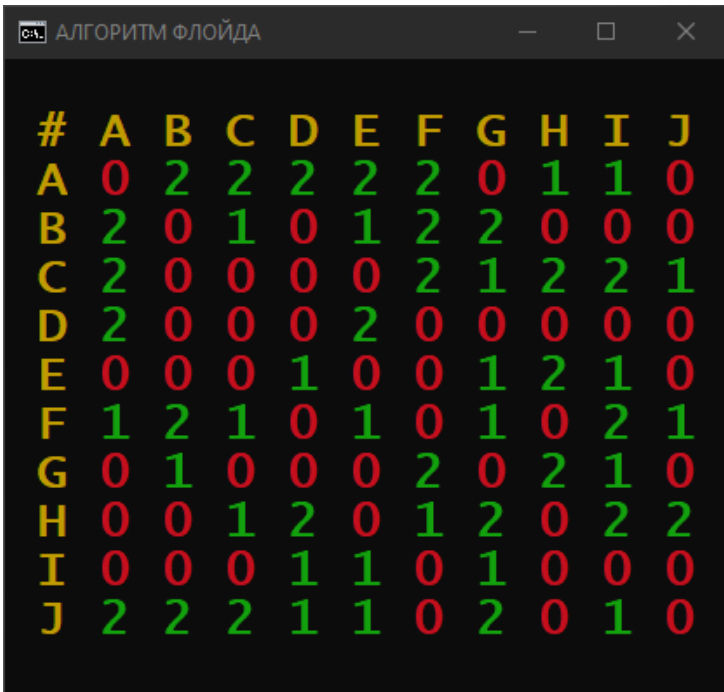
Затем нажимая на любую клавишу граф начнется заполняться недостающими значениями , и далее будет выбираться кратчайший путь и с заменять собой старый более длинный.

```

int ras = Convert.ToInt32(Math.Sqrt(grf.Length));
for (int i = 0; i < ras; i++)
{
    for (int j = 0; j < ras; j++)
    {
        matr[0][i, j] = grf[i, j];
    }
}
for (int v = 0; v < ras; v++)
{
    for (int i = 0; i < ras; i++)
    {
        for (int j = 0; j < ras; j++)
        {
            matr[v + 1][i, j] = matr[v][i, j];
        }
    }
    ras++;
    ras--;
    for (int i = 0; i < ras; i++)
    {
        for (int j = 0; j < ras; j++)
        {
            if (i == v || j == v) { continue; }
            if (i == j) { continue; }
            if (matr[v][v, j] != 0 && matr[v][i, v] != 0)
            {
                if (matr[v][i, j] == 0)
                {
                    matr[v + 1][i, j] = matr[v][v, j] + matr[v][i, v];
                }
                else
                {
                    if (matr[v][i, j] > matr[v][v, j] + matr[v][i, v]) { matr[v + 1][i, j] = matr[v][v, j] + matr[v][i, v]; }
                }
            }
        }
    }
    ras++;
    ras--;
}
}

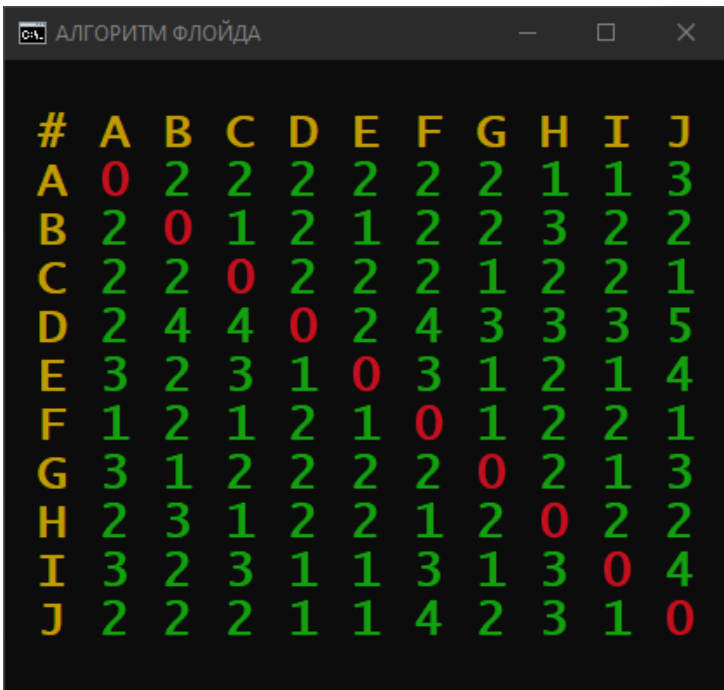
```

Ниже можно увидеть оформление начального запроса и дальнейшие действия с ним.



#	A	B	C	D	E	F	G	H	I	J
A	0	2	2	2	2	2	0	1	1	0
B	2	0	1	0	1	2	2	0	0	0
C	2	0	0	0	0	2	1	2	2	1
D	2	0	0	0	2	0	0	0	0	0
E	0	0	0	1	0	0	1	2	1	0
F	1	2	1	0	1	0	1	0	2	1
G	0	1	0	0	0	2	0	2	1	0
H	0	0	1	2	0	1	2	0	2	2
I	0	0	0	1	1	0	1	0	0	0
J	2	2	2	1	1	0	2	0	1	0

Ниже уже представлена полностью заполненная матрица кратчайших путей



#	A	B	C	D	E	F	G	H	I	J
A	0	2	2	2	2	2	2	1	1	3
B	2	0	1	2	1	2	2	3	2	2
C	2	2	0	2	2	2	1	2	2	1
D	2	4	4	0	2	4	3	3	3	5
E	3	2	3	1	0	3	1	2	1	4
F	1	2	1	2	1	0	1	2	2	1
G	3	1	2	2	2	2	0	2	1	3
H	2	3	1	2	2	1	2	0	2	2
I	3	2	3	1	1	3	1	3	0	4
J	2	2	2	1	1	4	2	3	1	0

Тестирование

Среда разработки Microsoft Visual Studio 2010 предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

Ниже продемонстрирован результат тестирования программы при вводе пользователем минимального и максимального количества вершин и вывод их матрицы кратчайших путей.

МИНИМУМ(2):

Матрица смежности:



#	A	B
A	0	2
B	1	0

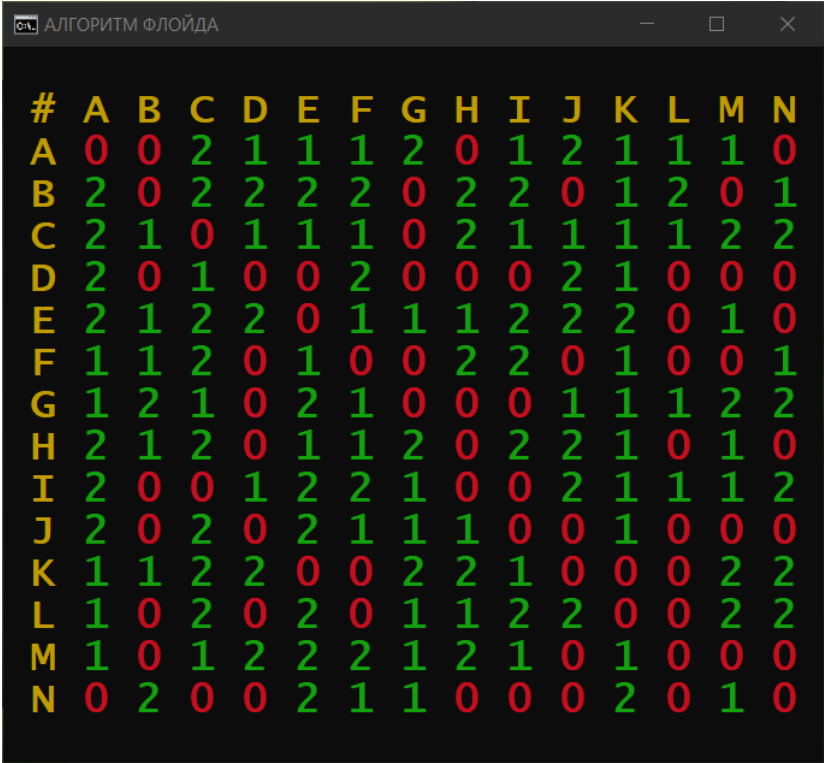
Результат:



#	A	B
A	0	2
B	1	0

МАКСИМУМ(14):

Матрица смежности:



#	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A	0	0	2	1	1	1	2	0	1	2	1	1	1	0
B	2	0	2	2	2	2	0	2	2	0	1	2	0	1
C	2	1	0	1	1	1	0	2	1	1	1	1	2	2
D	2	0	1	0	0	2	0	0	0	2	1	0	0	0
E	2	1	2	2	0	1	1	1	2	2	2	0	1	0
F	1	1	2	0	1	0	0	2	2	0	1	0	0	1
G	1	2	1	0	2	1	0	0	0	1	1	1	2	2
H	2	1	2	0	1	1	2	0	2	2	1	0	1	0
I	2	0	0	1	2	2	1	0	0	2	1	1	1	2
J	2	0	2	0	2	1	1	1	0	0	1	0	0	0
K	1	1	2	2	0	0	2	2	1	0	0	0	2	2
L	1	0	2	0	2	0	1	1	2	2	0	0	2	2
M	1	0	1	2	2	2	1	2	1	0	1	0	0	0
N	0	2	0	0	2	1	1	0	0	0	2	0	1	0

Результат:

#	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A	0	2	2	1	1	1	2	2	1	2	1	1	1	2
B	2	0	2	2	2	2	2	2	2	3	1	2	2	1
C	2	1	0	1	1	1	2	2	1	1	1	1	2	2
D	2	2	1	0	2	2	3	3	2	2	1	2	3	3
E	2	1	2	2	0	1	1	1	2	2	2	2	1	2
F	1	1	2	2	1	0	2	2	2	3	1	2	2	1
G	1	2	1	2	2	1	0	2	2	1	1	1	2	2
H	2	1	2	3	1	1	2	0	2	2	1	3	1	2
I	2	2	2	1	2	2	1	2	0	2	1	1	1	2
J	2	2	2	3	2	1	1	1	2	0	1	2	2	2
K	1	1	2	2	2	2	2	2	1	3	0	2	2	2
L	1	2	2	2	2	2	1	1	2	2	2	0	2	2
M	1	2	1	2	2	2	1	2	1	2	1	2	0	3
N	2	2	2	3	2	1	1	3	2	2	2	2	1	0

Таблица 1 – Описание поведения программы при тестировании

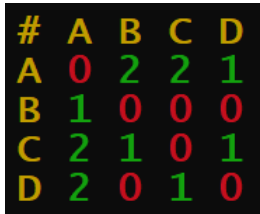
Описание теста	Ожидаемый результат	Полученный результат
Запуск программы	Вывод меню для выбора операции.	Верно
Генерация нового графа	Программа вывод меню для настройки нового графа, генерируется граф и выполняется алгоритм Флойда. Результат записывается в отдельную переменную и файл.	Верно
Вывод матрицы смежности	Выводит матрицу смежности	Верно
Вывод результата работы алгоритма Флойда	Выводит матрицу кратчайших путей	Верно

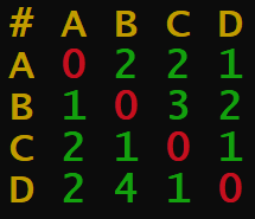
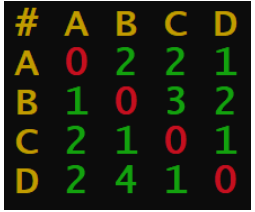
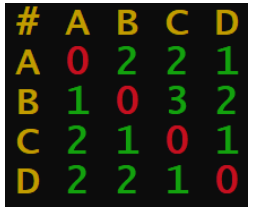
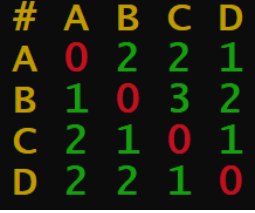
Ввод некорректных данных пользователем	Программа не реагирует или выводит информацию об ошибке ввода	Верно
--	---	-------

В результате тестирования было выявлено, что программа успешно проверяет данные на соответствие необходимым требованиям.

Ручной расчёт задачи

Проведем проверку программы посредством ручных вычислений на примере графа с 4 вершинами:

М	Результат при ручном вычислении	Результат работы программы
М ₀	$\begin{pmatrix} 0 & 2 & 2 & 1 \\ 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 1 \\ 2 & 0 & 1 & 0 \end{pmatrix}$	

M ₁	$\begin{pmatrix} 0 & 2 & 2 & 1 \\ 1 & 0 & 3 & 2 \\ 2 & 1 & 0 & 1 \\ 2 & 4 & 1 & 0 \end{pmatrix}$	 <pre> # A B C D A 0 2 2 1 B 1 0 3 2 C 2 1 0 1 D 2 4 1 0 </pre>
M ₂	$\begin{pmatrix} 0 & 2 & 2 & 1 \\ 1 & 0 & 3 & 2 \\ 2 & 1 & 0 & 1 \\ 2 & 4 & 1 & 0 \end{pmatrix}$	 <pre> # A B C D A 0 2 2 1 B 1 0 3 2 C 2 1 0 1 D 2 4 1 0 </pre>
M ₃	$\begin{pmatrix} 0 & 2 & 2 & 1 \\ 1 & 0 & 3 & 2 \\ 2 & 1 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{pmatrix}$	 <pre> # A B C D A 0 2 2 1 B 1 0 3 2 C 2 1 0 1 D 2 2 1 0 </pre>
M ₄	$\begin{pmatrix} 0 & 2 & 2 & 1 \\ 1 & 0 & 3 & 2 \\ 2 & 1 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{pmatrix}$	 <pre> # A B C D A 0 2 2 1 B 1 0 3 2 C 2 1 0 1 D 2 2 1 0 </pre>

В итоге результат работы программы совпал с результатом ручного расчёта.

Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Флойда для поиска кратчайшего пути в Microsoft Visual Studio 2022. При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания графа. Приобретены навыки по осуществлению алгоритма Флойда. Углублены знания языка программирования C#. Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка

сложность программного оконного интерфейса. Программа имеет небольшой, но достаточный для использования функционал возможностей.

Список литературы

Теория графов: <https://skysmart.ru/articles/mathematic/osnovnye-ponyatiya-teorii-grafov>

Алгоритм Флойда: <https://habr.com/ru/post/105825/>

Алгоритм Флойда: https://e-maxx.ru/algo/floyd_warshall_algorithm

Краткий обзор языка C#: <https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/>

C# Чтение и запись файла: <https://metanit.com/sharp/tutorial/5.4.php>

Листинг программы

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Net.Mime.MediaTypeNames;

namespace kurs2
{
    internal class Program
    {
        struct graf
```

```

{
    public int[,] matrSM;
}
const int w = 30;
const int h = 8;

//АЛГОРИТМ ФЛОЙДА √ √ √
static void floid(int[,],grf , int[,] matr)
{
    int ras = Convert.ToInt32(Math.Sqrt(grf.Length));
    for(int i = 0; i < ras; i++)
    {
        for (int j = 0; j < ras; j++)
        {
            matr[0][i,j] = grf[i,j];
        }
    }
    for (int v = 0; v < ras; v++)
    {
        for (int i = 0; i < ras; i++)
        {
            for (int j = 0; j < ras; j++)
            {
                matr[v + 1][i, j] = matr[v][i, j];
            }
        }
        for (int i = 0; i < ras; i++)
        {
            for (int j = 0; j < ras; j++)
            {
                if (i == v || j == v) { continue; }
                if (i == j) { continue; }
                if (matr[v][v, j] != 0 && matr[v][i, v] != 0)
                {
                    if (matr[v][i, j] == 0)
                    {
                        matr[v+1][i, j] = matr[v][v, j] + matr[v][i, v];
                    }
                    else
                    {
                        if (matr[v][i, j] > matr[v][v, j] + matr[v][i, v]) { matr[v+1][i, j] = matr[v][v, j]
+ matr[v][i, v]; }
                    }
                }
            }
        }
    }
}

```

```

//ГЕНЕРАЦИЯ ГРАФА \ \ \
static void graf_gen(int[,] grf)
{
    int ras = Convert.ToInt32(Math.Sqrt(grf.Length));
    Random rand = new Random(ras);
    for (int i = 0; i < ras; i++)
    {
        for (int j = 0; j < ras; j++)
        {
            if (i == j) { grf[i, j] = 0; }
            else { grf[i, j] = rand.Next(3); }
        }
    }
}

//ВЫВОД МАТРИЦЫ ГРАФА \ \ \
static void graf_print(int[,] grf, int x = 1, int y = 1)
{
    Console.SetCursorPosition(x, y);
    int ras = Convert.ToInt32(Math.Sqrt(grf.Length));
    for (int i = 0; i <= ras; i++)
    {
        for (int j = 0; j <= ras; j++)
        {
            if (i == 0 && j != 0)
            {
                Console.Write(" " + Convert.ToChar(64 + j));
                continue;
            }
            if (j == 0 && i != 0)
            {
                Console.Write(Convert.ToChar(64 + i));
                continue;
            }
            if (i == 0 && j == 0)
            {
                Console.Write("#");
                continue;
            }
            if (true) { if (grf[i-1, j-1] == 0) { sc(2); } else { sc(1); } }
            Console.Write(" " + grf[i-1, j-1]);
            sc(0);
        }
        Console.SetCursorPosition(x, y+1+i);
    }
}

//ИЗМЕНЕНИЕ РАЗМЕРА КОНСОЛИ \ \ \
static void consize(int weigh, int hight)
{
    Console.SetWindowSize(weigh - 1, hight - 1);
}

```


[illegible]

[illegible]

```

mod = Convert.ToInt32(Console.ReadKey().KeyChar);
if (mod != 65 && mod != 97 && mod != 68 && mod != 100)
{
    Console.Clear();
    consize(w, h);
    goto MenuMain;
}
if(mod == 65 || mod == 97)
{
    Console.Write(" ");
    if (mc == 0) { goto Point; }
    else
    {
        mc--;
        Console.SetCursorPosition(1, 3);
        Console.Write(" ");
    }
}

Console.SetCursorPosition(Convert.ToInt32(Math.Sqrt(grf.matrSM.Length)) * 2 + 7, 3);
    Console.Write(" ");
    goto Point;
}
}
if (mod == 68 || mod == 100)
{
    Console.Write(" ");
    if (mc+1 == Convert.ToInt32(Math.Sqrt(grf.matrSM.Length))) { goto Point; }
    else
    {
        mc++;
        Console.SetCursorPosition(1, 3);
        Console.Write(" ");
    }
}

Console.SetCursorPosition(Convert.ToInt32(Math.Sqrt(grf.matrSM.Length)) * 2 + 7, 3);
    Console.Write(" ");
    goto Point;
}
}
break;
}
}
}
}
}
}
}

```

