Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Лабораторная работа №7**

ПРОЕКТИРОВАНИЕ СЕТЕЙ РАДИОСВЯЗИ НА ЯЗЫКЕ С#

Выполнил:

Студент 2 курса 7 группы ФИТ

Бобрович Г.С.

**2022 г.**

**Цель работы**: расчет взаимовлияния базовых станций и построение графа сети радиосвязи на одном из языков программирования.

Задачи 3, 7

**Построение графа связи с помощью координационных колец**

Шаг 1. Вокруг каждой станции строятся окружности радиусом, равным координационному расстоянию, в нашем случае 5 км. На рис. 3.2 приведен пример координационного кольца для станции 8.

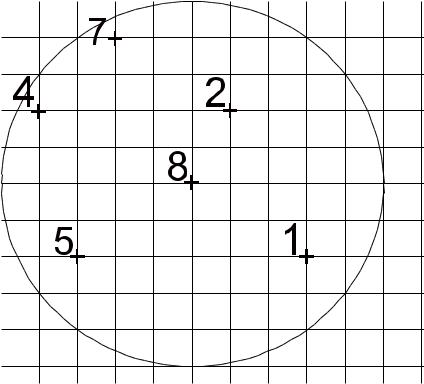


Рис. 1. Координационное кольцо

Шаг 2. Станции, лежащие внутри данного кольца, могут создавать помехи для станции 8, если работают с ней на одной частоте. Поэтому проверяем, какие частоты принадлежат станциям, лежащим внутри кольца. В данном случае станции 7 и 8 имеют одинаковые рабочие частоты, следовательно, оказывают друг на друга мешающее влияние. Ребром соединяем вершины 7 и 8.

Шаг 3. Аналогичные операции выполняются и для остальных станций сети. В результате построения колец и анализа помех, новых ребер в нашем случае не обнаруживается. Следовательно, получаем граф связи, изображенный на рис. 3.3.

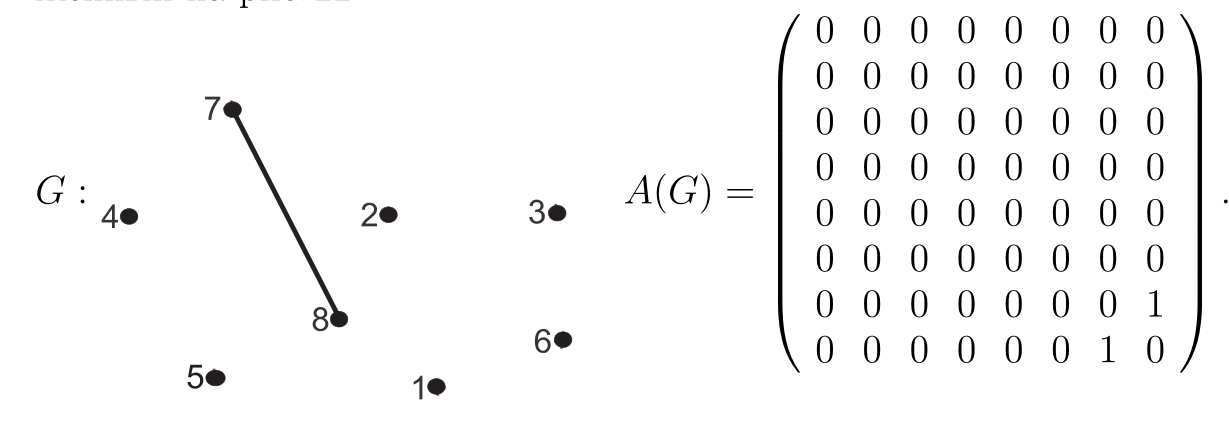


Рис. 2. Граф связи

*Замечание.* На рисунке, задающем граф сети связи, нет необходимости соблюдать порядок расположения вершин, соответствующий реальной сети, и выдерживать расстояния. Наличие помех отмечается ребрами.

**Построение графа связи с помощью матриц**

Анализируя условия, связывающие частоты, строим матрицу *T*:

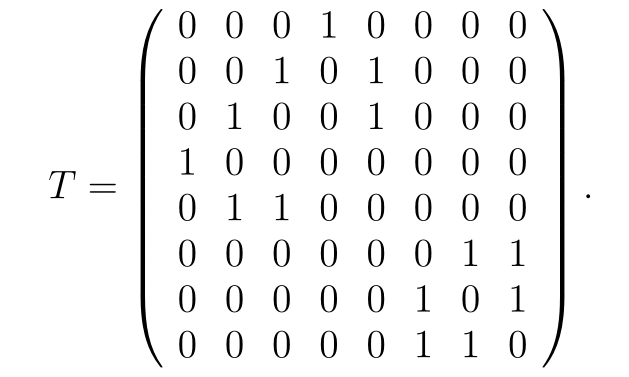


Рис. 3. Матрица смежности

Матрица симметрична, для построения матрицы смежности *A(G)* достаточно вычислить расстояния между станциями, которым соответствуют единицы в треугольнике над диагональю матрицы *T*:

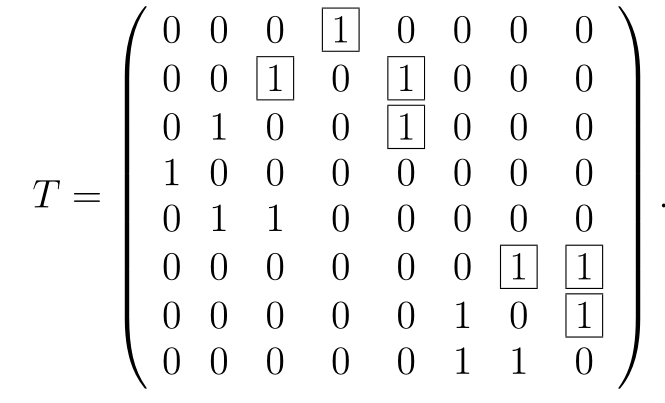


Рис. 4. Матрица смежности











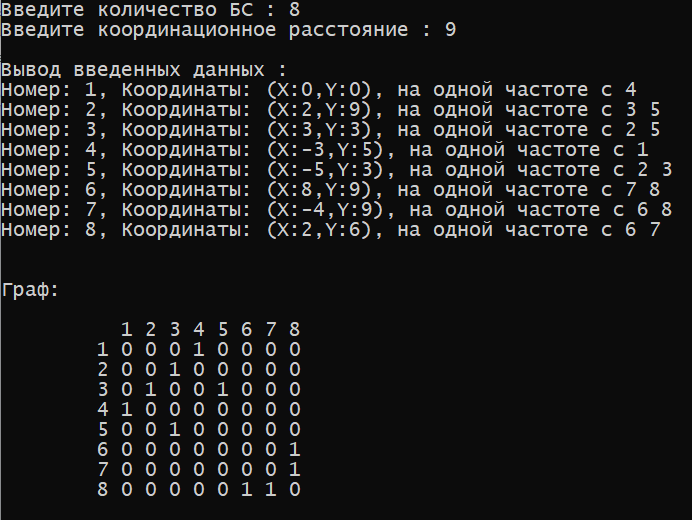


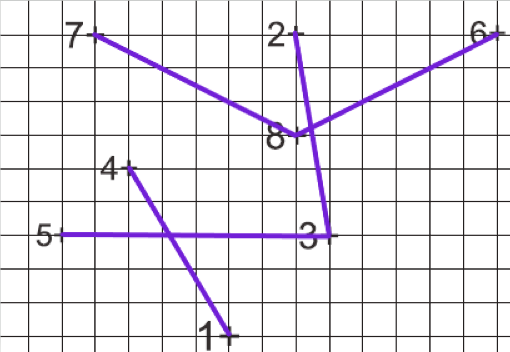


Алгоритм:

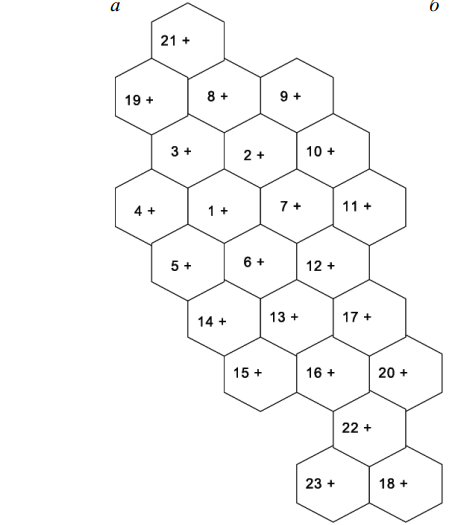
1. Вводим в программу БС с координатами и вышками работающими на той же частоте
2. Считаем матрицу смежности
3. Строим графы

3. Для радиопокрытия города N установлено восемь базовых станций (рис. 5, в). Координационное расстояние равно 9 км, рабочие частоты станций следующие: f1= f4, f2= f3 = f5, f6 = f7 = f8. Построить граф сети.





7. Структура сети сотовой связи города N представлена на рис. 6, в. Радиус зон покрытия базовых станций равен 1,6 км, координационное расстояние равно 10 км. Определить, какие станции будут оказывать взаимные влияния друг на друга, и построить граф сети.



Зная радиус зоны обслуживания, определим модуль сети:

R0 = 1,6\*=2,77

Определим расстояние R1 до ближайшей станции 2, 3, 4, 5, 6, 7. Оно будет равно:

R1 = 1,6\*=2,77

Определим расстояние R2 до ближайшей станции 8, 9, 10, 11, 12, 13, 14, 15,19. Оно будет равно:

R2 = 1,6\*=4,79

Определим расстояние R3 до ближайшей станции. Оно будет равно:

R3 = 2\*1,6\*=5,54

Определим расстояние R4 до ближайшей станции. Оно будет равно:

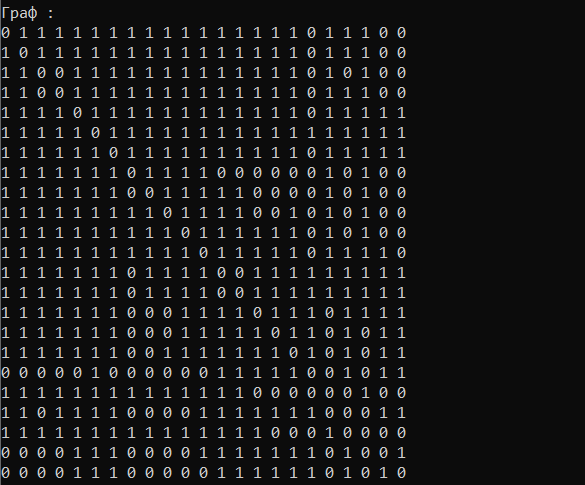
R4 = 1,6\*=7,32

Определим расстояние R5 до ближайшей станции. Оно будет равно:

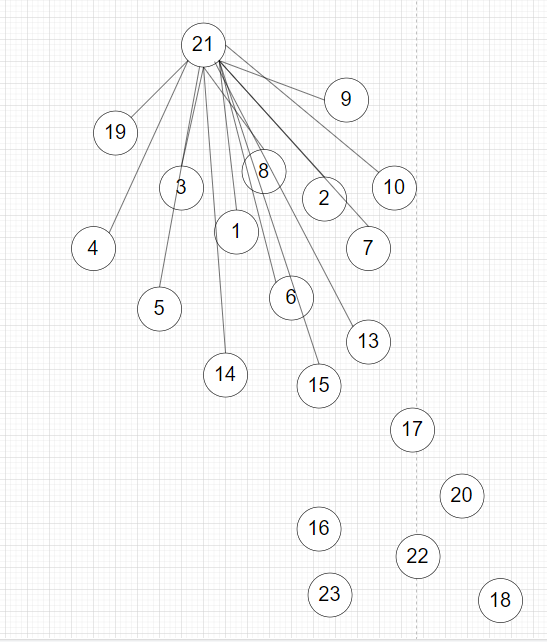
R5 = 2\*1,6\*=14,64 > 10

Так как остальные станции располагаются ещё дальше 16, а расстояние между 1 и 16 уже больше координационного, то дальнейшие расчеты для первой станции бесполезны. Таким образом, первая станция влияет на все станции, кроме 16, 20, 22, 18, 23.

Матрица смежности:



Построенный граф влияния первой станции:



**Контрольные вопросы**

1. **Что представляет собой сеть сотовой связи?**

**Сеть сотовой связи** представляет собой совокупность приемопередающих станций, обслуживающих определенную территорию.

1. **Что такое модуль сети радиосвязи?**

**Модуль сети радиосвязи** — расстояние между соседними стан­циями сети сотовой структуры.

1. **Что называют координационным расстоянием в планировании мобильных сетей?**

**Координационное расстояние D**— расстояние между станциями, работающими в одном частотном канале.

1. **Что такое радиус зоны обслуживания?**

**Радиус зоны обслуживания** — область, в которой обеспечивается прием сигналов с заданным качеством.

1. **Как строится граф сети радиосвязи и задается его матрица?**

**Граф сети радиосвязи** — граф, вершины которого соответствуют пунктам установки передающих станций, а ребрами соединены те передатчики, которые создают помехи приему в соответствующей зоне обслуживания друг друга.

**Построение графа связи с помощью координационых колец**

Задано расположение станций на местности, рабочие частоты, координационное расстояние D. Строим вокруг первой станции окружность радиу­са D. Для всех станций, попавших внутрь круга, проверяем, совпадают ли их рабочие частоты с частотой первой станции. Если частоты совпадают, то соответствующие данным станциям вершины в графе связи соединяем ребром с первой вершиной. Повторяем все вышеописанные действия для остальных станций.

**Построение графа связи с помощью матриц**

Задано расположение станций на местности, рабочие частоты, координационное расстояние D. Формируем матрицу ***T*** такую, что ***tij=1***(*i≠****j***), если рабочие частоты станций ***i*** и ***j*** совпадают, и ***tij=*** 0 при ***i=j*** или при несовпадении частот. Вычисляем расстояния между станциями с одинаковыми частотами ***(tij***= **1**) и на основе матрицы ***T*** формируем матрицу ***A,*** у которой элемент ***aij*** равен 1, если станции ***i***и ***j (****i≠****j***) имеют одинаковые частоты (т.е. ***tij***= 1) и расстояние между ними не больше координационного. Во всех остальных случаях ***aij=*0**. Полученная матрица ***A***— матрица смежности графа сети связи.

1. **Объясните назначение координационных колец.**

**Координационные кольца –** окружности, с центром в точке расположения станции, показывающая область влияния данной станции на другие, работающие с такой же частотой.

**Приложение (код)**

Код для первых 4 заданий (консольное приложение C#):

using System;

using static ConsoleApp1.BS;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

// ---------- Получаем необходимые данные и заполняем ими объекты БС

Console.Write("Введите количество БС : ");

int amountOfBS = int.Parse(Console.ReadLine());

Console.Write("Введите координационное расстояние : ");

int radius = int.Parse(Console.ReadLine());

// массив для хранения объектов станций

BS[] bs = new BS[amountOfBS];

//for (int i = 0; i < amountOfBS; i++)

//{

// bs[i] = new BS();

// // номер БС

// Console.Write("\nВведите номер БС : ");

// bs[i].Number = int.Parse(Console.ReadLine());

// // координаты БС

// Console.Write("Введите координаты x, y (Пример \"5 1\", где 5 по x, 1 по y) : ");

// string coordsArray = Console.ReadLine();

// string[] coords = coordsArray.Split(' '); // массив координат (x, y)

// bs[i].CoordinatesXY = new BS.Coordinate(int.Parse(coords[0]), int.Parse(coords[1]));

// // БС, работающие на той же частоте

// Console.Write("Введите номера БС, которые работают на той же частоте, что и данная (Пример: 1 4 5) : ");

// string sameFrequencyBSArray = Console.ReadLine();

// string[] sameFrequencyBS = sameFrequencyBSArray.Split(' ');

// // так как в классе мы храним массив БС на той же частоте, то определяем новый массив для заполнения

// // номерами базовых станций той же частоты

// bs[i].ArrayOfFrequencySameBS = new int[sameFrequencyBS.Length];

// // заполняем массив номерами БС

// for (int a = 0; a < sameFrequencyBS.Length; a++)

// {

// bs[i].ArrayOfFrequencySameBS[a] = int.Parse(sameFrequencyBS[a]);

// }

//}

bs[0] = new BS(1, new Coordinate(1, 5), new int[] { 2, 3, 4, 5, 6, 7, 8 });

bs[1] = new BS(2, new Coordinate(3, 3), new int[] { 1, 3, 4, 5, 6, 7, 8 });

bs[2] = new BS(3, new Coordinate(7, 5), new int[] { 1, 2, 4, 5, 6, 7, 8 });

bs[3] = new BS(4, new Coordinate(4, 7), new int[] { 1, 2, 3, 5, 6, 7, 8 });

bs[4] = new BS(5, new Coordinate(10, 7), new int[] { 1, 2, 3, 4, 6, 7, 8 });

bs[5] = new BS(6, new Coordinate(10, 2), new int[] { 1, 2, 3, 4, 5, 7, 8 });

bs[6] = new BS(7, new Coordinate(13, 4), new int[] { 1, 2, 3, 4, 5, 6, 8 });

bs[7] = new BS(8, new Coordinate(13, 11), new int[] { 1, 2, 3, 4, 5, 6, 7 });

Console.WriteLine("\nВывод введенных данных :");

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

Console.WriteLine(bs[i].ToString());

// ---------- Построение графа связи с помощью координационых колец

// создаем матрицу смежности, по умолчанию она заполнится нулями

int[,] matrix = new int[amountOfBS, amountOfBS];

// i - индекс БС в массиве bs объектов базовых станций

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

{

// a = индекс номера БС, которая записана в массив базовых станций той же частоты, что рассматриваемая

for (int a = 0; a < bs[i].ArrayOfFrequencySameBS.Length; a++)

{

// расчет дистанции между рассматриваемой БС (bs[i]) и БС той же частоты (bs[bs[i].ArrayOfFrequencySameBS[a] - 1])

// bs[i].ArrayOfFrequencySameBS[a] - 1 =>

// bs[i] - берем рассматриваемую БС,

// ArrayOfFrequencySameBS[a] - 1 - берем номер БС из массива базовых станций той же частоты и отнимаем 1

// для получения индекса данной БС в массиве объектов базовых станций (bs)

double dist = BS.Distance(bs[i].CoordinatesXY, bs[bs[i].ArrayOfFrequencySameBS[a] - 1].CoordinatesXY);

if (dist < radius) // если Координационное расстояние больше расстояние то есть пересечение и в матрицу записывается 1

matrix[i, bs[bs[i].ArrayOfFrequencySameBS[a] - 1].Number - 1] = 1;

}

}

// ---------- Выводим матрицу смежности полученного графа

Console.Write("\n\nГраф:\n\n\t ");

for (int i = 1; i <= amountOfBS; i++)

Console.Write(i + " ");

Console.WriteLine();

for (int row = 0; row < amountOfBS; row++)

{

Console.Write("\t" + (row + 1).ToString() + " "); // номер строки матрицы

for (int column = 0; column < amountOfBS; column++)

{

Console.Write(matrix[row, column] + " ");

}

Console.WriteLine();

}

Console.ReadKey();

}

}

internal class BS

{

public int Number { get; set; }

public Coordinate CoordinatesXY { get; set; }

public int[] ArrayOfFrequencySameBS;

public BS(int number, Coordinate cord, int[] neib)

{

Number = number;

CoordinatesXY = cord;

ArrayOfFrequencySameBS = neib;

}

public BS()

{

}

public override string ToString()

{

string coordsArray = "";

for (int i = 0; i < this.ArrayOfFrequencySameBS.Length; i++)

{

coordsArray += ArrayOfFrequencySameBS[i] + " ";

}

return

"Номер: " + this.Number +

", Координаты: (X:" + this.CoordinatesXY.X + ",Y:" + this.CoordinatesXY.Y +

"), на одной частоте с " + coordsArray;

}

public static double Distance(Coordinate A, Coordinate B)

{

// высчитываем расстояние между станциями как гипотенузу по теореме Пифагора

return Math.Sqrt(Math.Pow((B.X - A.X), 2) + Math.Pow((B.Y - A.Y), 2));

}

internal class Coordinate

{

public int X { get; set; }

public int Y { get; set; }

public Coordinate(int \_x, int \_y)

{

X = \_x;

Y = \_y;

}

}

}

}

Код для 7 задания (консольное приложение С++):

#include <iostream>

#include <stdlib.h>

#include <cmath>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

const int n = 23;//Кол-во вышек

int a[n][n] = //Матрица смежности

{

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0

};

double D = 10.0;//Координационное расстояние

double R0 = 1.6\*(sqrt(3.0));// модуль сети

double p12 = R0 \* (1 \* (sqrt(3.0)));

double p13 = R0 \* (1 \* (sqrt(3.0)));

double p14 = R0 \* (1 \* (sqrt(3.0)));

double p15 = R0 \* (1 \* (sqrt(3.0)));

double p16 = R0 \* (1 \* (sqrt(3.0)));

double p17 = R0 \* (1 \* (sqrt(3.0)));

double p18 = R0 \* (1 \* (sqrt(3.0)));

double p19 = R0 \* (1 \* (sqrt(3.0)));

double p110 = R0 \* (2 \* (sqrt(3.0)));

double p111 = R0 \* (2 \* (sqrt(3.0)));

double p112 = R0 \* (2 \* (sqrt(3.0)));

double p113 = R0 \* (2 \* (sqrt(3.0)));

double p114 = R0 \* (2 \* (sqrt(3.0)));

double p115 = R0 \* (2 \* (sqrt(3.0)));

double p116 = R0 \* (2 \* (sqrt(3.0)));

double p117 = R0 \* (2 \* (sqrt(3.0)));

double p118 = R0 \* (5 \* (sqrt(3.0)));

double p119 = R0 \* (2 \* (sqrt(3.0)));

double p120 = R0 \* (2 \* (sqrt(3.0)));

double p121 = R0 \* (2 \* (sqrt(3.0)));

double p122 = R0 \* (2 \* (sqrt(3.0)));

double p123 = R0 \* (5 \* (sqrt(3.0)));

//

double p23 = R0 \* (1 \* (sqrt(3.0)));

double p24 = R0 \* (2 \* (sqrt(3.0)));

double p25 = R0 \* (2 \* (sqrt(3.0)));

double p26 = R0 \* (2 \* (sqrt(3.0)));

double p27 = R0 \* (1 \* (sqrt(3.0)));

double p28 = R0 \* (1 \* (sqrt(3.0)));

double p29 = R0 \* (1 \* (sqrt(3.0)));

double p210 = R0 \* (1 \* (sqrt(3.0)));

double p211 = R0 \* (2 \* (sqrt(3.0)));

double p212 = R0 \* (2 \* (sqrt(3.0)));

double p213 = R0 \* (2 \* (sqrt(3.0)));

double p214 = R0 \* (2 \* (sqrt(3.0)));

double p215 = R0 \* (2 \* (sqrt(3.0)));

double p216 = R0 \* (2 \* (sqrt(3.0)));

double p217 = R0 \* (2 \* (sqrt(3.0)));

double p218 = R0 \* (6 \* (sqrt(3.0)));

double p219 = R0 \* (2 \* (sqrt(3.0)));

double p220 = R0 \* (2 \* (sqrt(3.0)));

double p221 = R0 \* (2 \* (sqrt(3.0)));

double p222 = R0 \* (5 \* (sqrt(3.0)));

double p223 = R0 \* (6 \* (sqrt(3.0)));

//

double p34 = R0 \* (1 \* (sqrt(3.0)));

double p35 = R0 \* (2 \* (sqrt(3.0)));

double p36 = R0 \* (2 \* (sqrt(3.0)));

double p37 = R0 \* (2 \* (sqrt(3.0)));

double p38 = R0 \* (1 \* (sqrt(3.0)));

double p39 = R0 \* (2 \* (sqrt(3.0)));

double p310 = R0 \* (2 \* (sqrt(3.0)));

double p311 = R0 \* (2 \* (sqrt(3.0)));

double p312 = R0 \* (2 \* (sqrt(3.0)));

double p313 = R0 \* (2 \* (sqrt(3.0)));

double p314 = R0 \* (2 \* (sqrt(3.0)));

double p315 = R0 \* (2 \* (sqrt(3.0)));

double p316 = R0 \* (2 \* (sqrt(3.0)));

double p317 = R0 \* (2 \* (sqrt(3.0)));

double p318 = R0 \* (6 \* (sqrt(3.0)));

double p319 = R0 \* (1 \* (sqrt(3.0)));

double p320 = R0 \* (5 \* (sqrt(3.0)));

double p321 = R0 \* (2 \* (sqrt(3.0)));

double p322 = R0 \* (6 \* (sqrt(3.0)));

double p323 = R0 \* (6 \* (sqrt(3.0)));

//

double p45 = R0 \* (1 \* (sqrt(3.0)));

double p46 = R0 \* (2 \* (sqrt(3.0)));

double p47 = R0 \* (2 \* (sqrt(3.0)));

double p48 = R0 \* (2 \* (sqrt(3.0)));

double p49 = R0 \* (2 \* (sqrt(3.0)));

double p410 = R0 \* (2 \* (sqrt(3.0)));

double p411 = R0 \* (2 \* (sqrt(3.0)));

double p412 = R0 \* (2 \* (sqrt(3.0)));

double p413 = R0 \* (2 \* (sqrt(3.0)));

double p414 = R0 \* (2 \* (sqrt(3.0)));

double p415 = R0 \* (2 \* (sqrt(3.0)));

double p416 = R0 \* (2 \* (sqrt(3.0)));

double p417 = R0 \* (2 \* (sqrt(3.0)));

double p418 = R0 \* (5 \* (sqrt(3.0)));

double p419 = R0 \* (2 \* (sqrt(3.0)));

double p420 = R0 \* (2 \* (sqrt(3.0)));

double p421 = R0 \* (2 \* (sqrt(3.0)));

double p422 = R0 \* (5 \* (sqrt(3.0)));

double p423 = R0 \* (6 \* (sqrt(3.0)));

//

double p56 = R0 \* (1 \* (sqrt(3.0)));

double p57 = R0 \* (2 \* (sqrt(3.0)));

double p58 = R0 \* (2 \* (sqrt(3.0)));

double p59 = R0 \* (2 \* (sqrt(3.0)));

double p510 = R0 \* (2 \* (sqrt(3.0)));

double p511 = R0 \* (2 \* (sqrt(3.0)));

double p512 = R0 \* (2 \* (sqrt(3.0)));

double p513 = R0 \* (2 \* (sqrt(3.0)));

double p514 = R0 \* (1 \* (sqrt(3.0)));

double p515 = R0 \* (2 \* (sqrt(3.0)));

double p516 = R0 \* (2 \* (sqrt(3.0)));

double p517 = R0 \* (2 \* (sqrt(3.0)));

double p518 = R0 \* (6 \* (sqrt(3.0)));

double p519 = R0 \* (2 \* (sqrt(3.0)));

double p520 = R0 \* (2 \* (sqrt(3.0)));

double p521 = R0 \* (2 \* (sqrt(3.0)));

double p522 = R0 \* (2 \* (sqrt(3.0)));

double p523 = R0 \* (2 \* (sqrt(3.0)));

//

double p67 = R0 \* (1 \* (sqrt(3.0)));

double p68 = R0 \* (2 \* (sqrt(3.0)));

double p69 = R0 \* (2 \* (sqrt(3.0)));

double p610 = R0 \* (2 \* (sqrt(3.0)));

double p611 = R0 \* (2 \* (sqrt(3.0)));

double p612 = R0 \* (1 \* (sqrt(3.0)));

double p613 = R0 \* (1 \* (sqrt(3.0)));

double p614 = R0 \* (1 \* (sqrt(3.0)));

double p615 = R0 \* (2 \* (sqrt(3.0)));

double p616 = R0 \* (2 \* (sqrt(3.0)));

double p617 = R0 \* (2 \* (sqrt(3.0)));

double p618 = R0 \* (2 \* (sqrt(3.0)));

double p619 = R0 \* (2 \* (sqrt(3.0)));

double p620 = R0 \* (2 \* (sqrt(3.0)));

double p621 = R0 \* (2 \* (sqrt(3.0)));

double p622 = R0 \* (2 \* (sqrt(3.0)));

double p623 = R0 \* (2 \* (sqrt(3.0)));

//

double p78 = R0 \* (2 \* (sqrt(3.0)));

double p79 = R0 \* (2 \* (sqrt(3.0)));

double p710 = R0 \* (1 \* (sqrt(3.0)));

double p711 = R0 \* (1 \* (sqrt(3.0)));

double p712 = R0 \* (1 \* (sqrt(3.0)));

double p713 = R0 \* (2 \* (sqrt(3.0)));

double p714 = R0 \* (2 \* (sqrt(3.0)));

double p715 = R0 \* (2 \* (sqrt(3.0)));

double p716 = R0 \* (1 \* (sqrt(3.0)));

double p717 = R0 \* (2 \* (sqrt(3.0)));

double p718 = R0 \* (5 \* (sqrt(3.0)));

double p719 = R0 \* (2 \* (sqrt(3.0)));

double p720 = R0 \* (2 \* (sqrt(3.0)));

double p721 = R0 \* (2 \* (sqrt(3.0)));

double p722 = R0 \* (2 \* (sqrt(3.0)));

double p723 = R0 \* (2 \* (sqrt(3.0)));

//

double p89 = R0 \* (1 \* (sqrt(3.0)));

double p810 = R0 \* (2 \* (sqrt(3.0)));

double p811 = R0 \* (2 \* (sqrt(3.0)));

double p812 = R0 \* (2 \* (sqrt(3.0)));

double p813 = R0 \* (5 \* (sqrt(3.0)));

double p814 = R0 \* (5 \* (sqrt(3.0)));

double p815 = R0 \* (5 \* (sqrt(3.0)));

double p816 = R0 \* (5 \* (sqrt(3.0)));

double p817 = R0 \* (5 \* (sqrt(3.0)));

double p818 = R0 \* (8 \* (sqrt(3.0)));

double p819 = R0 \* (1 \* (sqrt(3.0)));

double p820 = R0 \* (8 \* (sqrt(3.0)));

double p821 = R0 \* (1 \* (sqrt(3.0)));

double p822 = R0 \* (8 \* (sqrt(3.0)));

double p823 = R0 \* (9 \* (sqrt(3.0)));

//

double p910 = R0 \* (1 \* (sqrt(3.0)));

double p911 = R0 \* (2 \* (sqrt(3.0)));

double p912 = R0 \* (2 \* (sqrt(3.0)));

double p913 = R0 \* (2 \* (sqrt(3.0)));

double p914 = R0 \* (2 \* (sqrt(3.0)));

double p915 = R0 \* (6 \* (sqrt(3.0)));

double p916 = R0 \* (7 \* (sqrt(3.0)));

double p917 = R0 \* (6 \* (sqrt(3.0)));

double p918 = R0 \* (9 \* (sqrt(3.0)));

double p919 = R0 \* (2 \* (sqrt(3.0)));

double p920 = R0 \* (9 \* (sqrt(3.0)));

double p921 = R0 \* (2 \* (sqrt(3.0)));

double p922 = R0 \* (9 \* (sqrt(3.0)));

double p923 = R0 \* (10 \* (sqrt(3.0)));

//

double p1011 = R0 \* (1 \* (sqrt(3.0)));

double p1012 = R0 \* (2 \* (sqrt(3.0)));

double p1013 = R0 \* (2 \* (sqrt(3.0)));

double p1014 = R0 \* (2 \* (sqrt(3.0)));

double p1015 = R0 \* (5 \* (sqrt(3.0)));

double p1016 = R0 \* (5 \* (sqrt(3.0)));

double p1017 = R0 \* (2 \* (sqrt(3.0)));

double p1018 = R0 \* (6 \* (sqrt(3.0)));

double p1019 = R0 \* (2 \* (sqrt(3.0)));

double p1020 = R0 \* (6 \* (sqrt(3.0)));

double p1021 = R0 \* (2 \* (sqrt(3.0)));

double p1022 = R0 \* (6 \* (sqrt(3.0)));

double p1023 = R0 \* (6 \* (sqrt(3.0)));

//

double p1112 = R0 \* (1 \* (sqrt(3.0)));

double p1113 = R0 \* (2 \* (sqrt(3.0)));

double p1114 = R0 \* (2 \* (sqrt(3.0)));

double p1115 = R0 \* (2 \* (sqrt(3.0)));

double p1116 = R0 \* (2 \* (sqrt(3.0)));

double p1117 = R0 \* (2 \* (sqrt(3.0)));

double p1118 = R0 \* (7 \* (sqrt(3.0)));

double p1119 = R0 \* (2 \* (sqrt(3.0)));

double p1120 = R0 \* (5 \* (sqrt(3.0)));

double p1121 = R0 \* (2 \* (sqrt(3.0)));

double p1122 = R0 \* (7 \* (sqrt(3.0)));

double p1123 = R0 \* (8 \* (sqrt(3.0)));

//

double p1213 = R0 \* (1 \* (sqrt(3.0)));

double p1214 = R0 \* (2 \* (sqrt(3.0)));

double p1215 = R0 \* (2 \* (sqrt(3.0)));

double p1216 = R0 \* (2 \* (sqrt(3.0)));

double p1217 = R0 \* (1 \* (sqrt(3.0)));

double p1218 = R0 \* (5 \* (sqrt(3.0)));

double p1219 = R0 \* (2 \* (sqrt(3.0)));

double p1220 = R0 \* (2 \* (sqrt(3.0)));

double p1221 = R0 \* (2 \* (sqrt(3.0)));

double p1222 = R0 \* (2 \* (sqrt(3.0)));

double p1223 = R0 \* (5 \* (sqrt(3.0)));

//

double p1314 = R0 \* (1 \* (sqrt(3.0)));

double p1315 = R0 \* (1 \* (sqrt(3.0)));

double p1316 = R0 \* (1 \* (sqrt(3.0)));

double p1317 = R0 \* (1 \* (sqrt(3.0)));

double p1318 = R0 \* (2 \* (sqrt(3.0)));

double p1319 = R0 \* (2 \* (sqrt(3.0)));

double p1320 = R0 \* (2 \* (sqrt(3.0)));

double p1321 = R0 \* (2 \* (sqrt(3.0)));

double p1322 = R0 \* (2 \* (sqrt(3.0)));

double p1323 = R0 \* (2 \* (sqrt(3.0)));

//

double p1415 = R0 \* (1 \* (sqrt(3.0)));

double p1416 = R0 \* (1 \* (sqrt(3.0)));

double p1417 = R0 \* (2 \* (sqrt(3.0)));

double p1418 = R0 \* (2 \* (sqrt(3.0)));

double p1419 = R0 \* (2 \* (sqrt(3.0)));

double p1420 = R0 \* (2 \* (sqrt(3.0)));

double p1421 = R0 \* (2 \* (sqrt(3.0)));

double p1422 = R0 \* (2 \* (sqrt(3.0)));

double p1423 = R0 \* (2 \* (sqrt(3.0)));

//

double p1516 = R0 \* (1 \* (sqrt(3.0)));

double p1517 = R0 \* (1 \* (sqrt(3.0)));

double p1518 = R0 \* (2 \* (sqrt(3.0)));

double p1519 = R0 \* (5 \* (sqrt(3.0)));

double p1520 = R0 \* (2 \* (sqrt(3.0)));

double p1521 = R0 \* (2 \* (sqrt(3.0)));

double p1522 = R0 \* (2 \* (sqrt(3.0)));

double p1523 = R0 \* (2 \* (sqrt(3.0)));

//

double p1617 = R0 \* (1 \* (sqrt(3.0)));

double p1618 = R0 \* (2 \* (sqrt(3.0)));

double p1619 = R0 \* (5 \* (sqrt(3.0)));

double p1620 = R0 \* (1 \* (sqrt(3.0)));

double p1621 = R0 \* (6 \* (sqrt(3.0)));

double p1622 = R0 \* (1 \* (sqrt(3.0)));

double p1623 = R0 \* (2 \* (sqrt(3.0)));

//

double p1718 = R0 \* (2 \* (sqrt(3.0)));

double p1719 = R0 \* (5 \* (sqrt(3.0)));

double p1720 = R0 \* (1 \* (sqrt(3.0)));

double p1721 = R0 \* (6 \* (sqrt(3.0)));

double p1722 = R0 \* (2 \* (sqrt(3.0)));

double p1723 = R0 \* (2 \* (sqrt(3.0)));

//

double p1819 = R0 \* (8 \* (sqrt(3.0)));

double p1820 = R0 \* (2 \* (sqrt(3.0)));

double p1821 = R0 \* (10 \* (sqrt(3.0)));

double p1822 = R0 \* (1 \* (sqrt(3.0)));

double p1823 = R0 \* (1 \* (sqrt(3.0)));

//

double p1920 = R0 \* (7 \* (sqrt(3.0)));

double p1921 = R0 \* (1 \* (sqrt(3.0)));

double p1922 = R0 \* (8 \* (sqrt(3.0)));

double p1923 = R0 \* (9 \* (sqrt(3.0)));

//

double p2021 = R0 \* (7 \* (sqrt(3.0)));

double p2022 = R0 \* (1 \* (sqrt(3.0)));

double p2023 = R0 \* (2 \* (sqrt(3.0)));

//

double p2122 = R0 \* (9 \* (sqrt(3.0)));

double p2123 = R0 \* (10 \* (sqrt(3.0)));

//

double p2223 = R0 \* (1 \* (sqrt(3.0)));

//

if (p12 < D) {

a[0][1] = 1;

a[1][0] = 1;

}

if (p13 < D) {

a[0][2] = 1;

a[2][0] = 1;

}

if (p14 < D) {

a[0][3] = 1;

a[3][0] = 1;

}

if (p15 < D) {

a[0][4] = 1;

a[4][0] = 1;

}

if (p16 < D) {

a[0][5] = 1;

a[5][0] = 1;

}

if (p17 < D) {

a[0][6] = 1;

a[6][0] = 1;

}

if (p18 < D) {

a[0][7] = 1;

a[7][0] = 1;

}

if (p19 < D) {

a[0][8] = 1;

a[8][0] = 1;

}

if (p110 < D) {

a[0][9] = 1;

a[9][0] = 1;

}

if (p111 < D) {

a[0][10] = 1;

a[10][0] = 1;

}

if (p112 < D) {

a[0][11] = 1;

a[11][0] = 1;

}

if (p113 < D) {

a[0][12] = 1;

a[12][0] = 1;

}

if (p114 < D) {

a[0][13] = 1;

a[13][0] = 1;

}

if (p115 < D) {

a[0][14] = 1;

a[14][0] = 1;

}

if (p116 < D) {

a[0][15] = 1;

a[15][0] = 1;

}

if (p117 < D) {

a[0][16] = 1;

a[16][0] = 1;

}

if (p118 < D) {

a[0][17] = 1;

a[17][0] = 1;

}

if (p119 < D) {

a[0][18] = 1;

a[18][0] = 1;

}

if (p120 < D) {

a[0][19] = 1;

a[19][0] = 1;

}

if (p121 < D) {

a[0][20] = 1;

a[20][0] = 1;

}

if (p122 < D) {

a[0][20] = 1;

a[20][0] = 1;

}

if (p123 < D) {

a[0][22] = 1;

a[22][0] = 1;

}

//

if (p23 < D) {

a[1][2] = 1;

a[2][1] = 1;

}

if (p24 < D) {

a[1][3] = 1;

a[3][1] = 1;

}

if (p25 < D) {

a[1][4] = 1;

a[4][1] = 1;

}

if (p26 < D) {

a[1][5] = 1;

a[5][1] = 1;

}

if (p27 < D) {

a[1][6] = 1;

a[6][1] = 1;

}

if (p28 < D) {

a[1][7] = 1;

a[7][1] = 1;

}

if (p29 < D) {

a[1][8] = 1;

a[8][1] = 1;

}

if (p210 < D) {

a[1][9] = 1;

a[9][1] = 1;

}

if (p211 < D) {

a[1][10] = 1;

a[10][1] = 1;

}

if (p212 < D) {

a[1][11] = 1;

a[11][1] = 1;

}

if (p213 < D) {

a[1][12] = 1;

a[12][1] = 1;

}

if (p214 < D) {

a[1][13] = 1;

a[13][1] = 1;

}

if (p215 < D) {

a[1][14] = 1;

a[14][1] = 1;

}

if (p216 < D) {

a[1][15] = 1;

a[15][1] = 1;

}

if (p217 < D) {

a[1][16] = 1;

a[16][1] = 1;

}

if (p218 < D) {

a[1][17] = 1;

a[17][1] = 1;

}

if (p219 < D) {

a[1][18] = 1;

a[18][1] = 1;

}

if (p220 < D) {

a[1][19] = 1;

a[19][1] = 1;

}

if (p221 < D) {

a[1][20] = 1;

a[20][1] = 1;

}

if (p222 < D) {

a[1][21] = 1;

a[21][1] = 1;

}

if (p223 < D) {

a[1][22] = 1;

a[22][1] = 1;

}

//

if (p34 < D) {

a[2][4] = 1;

a[4][2] = 1;

}

if (p35 < D) {

a[2][4] = 1;

a[4][2] = 1;

}

if (p36 < D) {

a[2][5] = 1;

a[5][2] = 1;

}

if (p37 < D) {

a[2][6] = 1;

a[6][2] = 1;

}

if (p38 < D) {

a[2][7] = 1;

a[7][2] = 1;

}

if (p39 < D) {

a[2][8] = 1;

a[8][2] = 1;

}

if (p310 < D) {

a[2][9] = 1;

a[9][2] = 1;

}

if (p311 < D) {

a[2][10] = 1;

a[10][2] = 1;

}

if (p312 < D) {

a[2][11] = 1;

a[11][2] = 1;

}

if (p313 < D) {

a[2][12] = 1;

a[12][2] = 1;

}

if (p314 < D) {

a[2][13] = 1;

a[13][2] = 1;

}

if (p315 < D) {

a[2][14] = 1;

a[14][2] = 1;

}

if (p316 < D) {

a[2][15] = 1;

a[15][2] = 1;

}

if (p317 < D) {

a[2][16] = 1;

a[16][2] = 1;

}

if (p318 < D) {

a[2][17] = 1;

a[17][2] = 1;

}

if (p319 < D) {

a[2][18] = 1;

a[18][2] = 1;

}

if (p320 < D) {

a[2][19] = 1;

a[19][2] = 1;

}

if (p321 < D) {

a[2][20] = 1;

a[20][2] = 1;

}

if (p322 < D) {

a[2][21] = 1;

a[21][2] = 1;

}

if (p323 < D) {

a[2][22] = 1;

a[22][2] = 1;

}

//

if (p45 < D) {

a[3][4] = 1;

a[4][3] = 1;

}

if (p46 < D) {

a[3][5] = 1;

a[5][3] = 1;

}

if (p47 < D) {

a[3][6] = 1;

a[6][3] = 1;

}

if (p48 < D) {

a[3][7] = 1;

a[7][3] = 1;

}

if (p49 < D) {

a[3][8] = 1;

a[8][3] = 1;

}

if (p410 < D) {

a[3][9] = 1;

a[9][3] = 1;

}

if (p411 < D) {

a[3][10] = 1;

a[10][3] = 1;

}

if (p412 < D) {

a[3][11] = 1;

a[11][3] = 1;

}

if (p413 < D) {

a[3][12] = 1;

a[12][3] = 1;

}

if (p414 < D) {

a[3][13] = 1;

a[13][3] = 1;

}

if (p415 < D) {

a[3][14] = 1;

a[14][3] = 1;

}

if (p416 < D) {

a[3][15] = 1;

a[15][3] = 1;

}

if (p417 < D) {

a[3][16] = 1;

a[16][3] = 1;

}

if (p418 < D) {

a[3][17] = 1;

a[17][3] = 1;

}

if (p419 < D) {

a[3][18] = 1;

a[18][3] = 1;

}

if (p420 < D) {

a[3][19] = 1;

a[19][3] = 1;

}

if (p421 < D) {

a[3][20] = 1;

a[20][3] = 1;

}

if (p422 < D) {

a[3][21] = 1;

a[21][3] = 1;

}

if (p423 < D) {

a[3][22] = 1;

a[22][3] = 1;

}

//

if (p56 < D) {

a[4][5] = 1;

a[5][4] = 1;

}

if (p57 < D) {

a[4][6] = 1;

a[6][4] = 1;

}

if (p58 < D) {

a[4][7] = 1;

a[7][4] = 1;

}

if (p59 < D) {

a[4][8] = 1;

a[8][4] = 1;

}

if (p510 < D) {

a[4][9] = 1;

a[9][4] = 1;

}

if (p511 < D) {

a[4][10] = 1;

a[10][4] = 1;

}

if (p512 < D) {

a[4][11] = 1;

a[11][4] = 1;

}

if (p513 < D) {

a[4][12] = 1;

a[12][4] = 1;

}

if (p514 < D) {

a[4][13] = 1;

a[13][4] = 1;

}

if (p515 < D) {

a[4][14] = 1;

a[14][4] = 1;

}

if (p516 < D) {

a[4][15] = 1;

a[15][4] = 1;

}

if (p517 < D) {

a[4][16] = 1;

a[16][4] = 1;

}

if (p518 < D) {

a[4][17] = 1;

a[17][4] = 1;

}

if (p519 < D) {

a[4][18] = 1;

a[18][4] = 1;

}

if (p520 < D) {

a[4][19] = 1;

a[19][4] = 1;

}

if (p521 < D) {

a[4][20] = 1;

a[20][4] = 1;

}

if (p522 < D) {

a[4][21] = 1;

a[21][4] = 1;

}

if (p523 < D) {

a[4][22] = 1;

a[22][4] = 1;

}

//

if (p67 < D) {

a[5][6] = 1;

a[6][5] = 1;

}

if (p68 < D) {

a[5][7] = 1;

a[7][5] = 1;

}

if (p69 < D) {

a[5][8] = 1;

a[8][5] = 1;

}

if (p610 < D) {

a[5][9] = 1;

a[9][5] = 1;

}

if (p611 < D) {

a[5][10] = 1;

a[10][5] = 1;

}

if (p612 < D) {

a[5][11] = 1;

a[11][5] = 1;

}

if (p613 < D) {

a[5][12] = 1;

a[12][5] = 1;

}

if (p614 < D) {

a[5][13] = 1;

a[13][5] = 1;

}

if (p615 < D) {

a[5][14] = 1;

a[14][5] = 1;

}

if (p616 < D) {

a[5][15] = 1;

a[15][5] = 1;

}

if (p617 < D) {

a[5][16] = 1;

a[16][5] = 1;

}

if (p618 < D) {

a[5][17] = 1;

a[17][5] = 1;

}

if (p619 < D) {

a[5][18] = 1;

a[18][5] = 1;

}

if (p620 < D) {

a[5][19] = 1;

a[19][5] = 1;

}

if (p621 < D) {

a[5][20] = 1;

a[20][5] = 1;

}

if (p622 < D) {

a[5][21] = 1;

a[21][5] = 1;

}

if (p623 < D) {

a[5][22] = 1;

a[22][5] = 1;

}

//

if (p78 < D) {

a[6][7] = 1;

a[7][6] = 1;

}

if (p79 < D) {

a[6][8] = 1;

a[8][6] = 1;

}

if (p710 < D) {

a[6][9] = 1;

a[9][6] = 1;

}

if (p711 < D) {

a[6][10] = 1;

a[10][6] = 1;

}

if (p712 < D) {

a[6][11] = 1;

a[11][6] = 1;

}

if (p713 < D) {

a[6][12] = 1;

a[12][6] = 1;

}

if (p714 < D) {

a[6][13] = 1;

a[13][6] = 1;

}

if (p715 < D) {

a[6][14] = 1;

a[14][6] = 1;

}

if (p716 < D) {

a[6][15] = 1;

a[15][6] = 1;

}

if (p717 < D) {

a[6][16] = 1;

a[16][6] = 1;

}

if (p718 < D) {

a[6][17] = 1;

a[17][6] = 1;

}

if (p719 < D) {

a[6][18] = 1;

a[18][6] = 1;

}

if (p720 < D) {

a[6][19] = 1;

a[19][6] = 1;

}

if (p721 < D) {

a[6][20] = 1;

a[20][6] = 1;

}

if (p722 < D) {

a[6][21] = 1;

a[21][6] = 1;

}

if (p723 < D) {

a[6][22] = 1;

a[22][6] = 1;

}

//

if (p89 < D) {

a[7][8] = 1;

a[8][6] = 1;

}

if (p810 < D) {

a[7][9] = 1;

a[9][7] = 1;

}

if (p811 < D) {

a[7][10] = 1;

a[10][7] = 1;

}

if (p812 < D) {

a[7][11] = 1;

a[11][7] = 1;

}

if (p813 < D) {

a[7][12] = 1;

a[12][7] = 1;

}

if (p814 < D) {

a[7][13] = 1;

a[13][7] = 1;

}

if (p815 < D) {

a[7][14] = 1;

a[14][7] = 1;

}

if (p815 < D) {

a[7][14] = 1;

a[14][7] = 1;

}

if (p816 < D) {

a[7][15] = 1;

a[15][7] = 1;

}

if (p817 < D) {

a[7][16] = 1;

a[16][7] = 1;

}

if (p818 < D) {

a[7][17] = 1;

a[17][7] = 1;

}

if (p819 < D) {

a[7][18] = 1;

a[18][7] = 1;

}

if (p820 < D) {

a[7][19] = 1;

a[19][7] = 1;

}

if (p821 < D) {

a[7][20] = 1;

a[20][7] = 1;

}

if (p822 < D) {

a[7][21] = 1;

a[21][7] = 1;

}

if (p823 < D) {

a[7][22] = 1;

a[22][7] = 1;

}

//

if (p910 < D) {

a[8][9] = 1;

a[9][8] = 1;

}

if (p911 < D) {

a[8][10] = 1;

a[10][8] = 1;

}

if (p912 < D) {

a[8][11] = 1;

a[11][8] = 1;

}

if (p913 < D) {

a[8][12] = 1;

a[12][8] = 1;

}

if (p914 < D) {

a[8][13] = 1;

a[13][8] = 1;

}

if (p915 < D) {

a[8][14] = 1;

a[14][8] = 1;

}

if (p916 < D) {

a[8][15] = 1;

a[15][8] = 1;

}

if (p918 < D) {

a[8][17] = 1;

a[17][8] = 1;

}

if (p919 < D) {

a[8][18] = 1;

a[18][8] = 1;

}

if (p920 < D) {

a[8][19] = 1;

a[19][8] = 1;

}

if (p921 < D) {

a[8][20] = 1;

a[20][8] = 1;

}

if (p922 < D) {

a[8][21] = 1;

a[21][8] = 1;

}

if (p923 < D) {

a[8][22] = 1;

a[22][8] = 1;

}

//

if (p1011 < D) {

a[9][10] = 1;

a[10][9] = 1;

}

if (p1012 < D) {

a[9][11] = 1;

a[11][9] = 1;

}

if (p1013 < D) {

a[9][12] = 1;

a[12][9] = 1;

}

if (p1014 < D) {

a[9][13] = 1;

a[13][9] = 1;

}

if (p1015 < D) {

a[9][14] = 1;

a[14][9] = 1;

}

if (p1016 < D) {

a[9][15] = 1;

a[15][9] = 1;

}

if (p1017 < D) {

a[9][16] = 1;

a[16][9] = 1;

}

if (p1018 < D) {

a[9][17] = 1;

a[17][9] = 1;

}

if (p1019 < D) {

a[9][18] = 1;

a[18][9] = 1;

}

if (p1020 < D) {

a[9][19] = 1;

a[19][9] = 1;

}

if (p1021 < D) {

a[9][20] = 1;

a[20][9] = 1;

}

if (p1022 < D) {

a[9][21] = 1;

a[21][9] = 1;

}

if (p1023 < D) {

a[9][22] = 1;

a[22][9] = 1;

}

//

if (p1112 < D) {

a[10][11] = 1;

a[11][10] = 1;

}

if (p1113 < D) {

a[10][12] = 1;

a[12][10] = 1;

}

if (p1114 < D) {

a[10][13] = 1;

a[13][10] = 1;

}

if (p1115 < D) {

a[10][14] = 1;

a[14][10] = 1;

}

if (p1116 < D) {

a[10][15] = 1;

a[15][10] = 1;

}

if (p1117 < D) {

a[10][16] = 1;

a[16][10] = 1;

}

if (p1118 < D) {

a[10][17] = 1;

a[17][10] = 1;

}

if (p1119 < D) {

a[10][18] = 1;

a[18][10] = 1;

}

if (p1120 < D) {

a[10][19] = 1;

a[19][10] = 1;

}

if (p1121 < D) {

a[10][20] = 1;

a[20][10] = 1;

}

if (p1122 < D) {

a[10][21] = 1;

a[21][10] = 1;

}

if (p1123 < D) {

a[10][22] = 1;

a[22][10] = 1;

}

//

if (p1213 < D) {

a[11][12] = 1;

a[12][11] = 1;

}

if (p1214 < D) {

a[11][13] = 1;

a[13][11] = 1;

}

if (p1215 < D) {

a[11][14] = 1;

a[14][11] = 1;

}

if (p1216 < D) {

a[11][15] = 1;

a[15][11] = 1;

}

if (p1217 < D) {

a[11][16] = 1;

a[16][11] = 1;

}

if (p1218 < D) {

a[11][17] = 1;

a[17][11] = 1;

}

if (p1219 < D) {

a[11][18] = 1;

a[18][11] = 1;

}

if (p1220 < D) {

a[11][19] = 1;

a[19][11] = 1;

}

if (p1221 < D) {

a[11][20] = 1;

a[20][11] = 1;

}

if (p1222 < D) {

a[11][21] = 1;

a[21][11] = 1;

}

if (p1223 < D) {

a[11][22] = 1;

a[22][11] = 1;

}

//

if (p1314 < D) {

a[9][10] = 1;

a[10][9] = 1;

}

if (p1315 < D) {

a[12][14] = 1;

a[14][12] = 1;

}

if (p1316 < D) {

a[12][15] = 1;

a[15][12] = 1;

}

if (p1317 < D) {

a[12][16] = 1;

a[16][12] = 1;

}

if (p1318 < D) {

a[12][17] = 1;

a[17][12] = 1;

}

if (p1319 < D) {

a[12][18] = 1;

a[18][12] = 1;

}

if (p1320 < D) {

a[12][19] = 1;

a[19][12] = 1;

}

if (p1321 < D) {

a[12][20] = 1;

a[20][12] = 1;

}

if (p1322 < D) {

a[12][21] = 1;

a[21][12] = 1;

}

if (p1323 < D) {

a[12][22] = 1;

a[22][12] = 1;

}

//

if (p1415 < D) {

a[13][14] = 1;

a[14][13] = 1;

}

if (p1416 < D) {

a[13][15] = 1;

a[15][13] = 1;

}

if (p1417 < D) {

a[13][16] = 1;

a[16][13] = 1;

}

if (p1418 < D) {

a[13][17] = 1;

a[17][13] = 1;

}

if (p1419 < D) {

a[13][18] = 1;

a[18][13] = 1;

}

if (p1420 < D) {

a[13][19] = 1;

a[19][13] = 1;

}

if (p1421 < D) {

a[13][20] = 1;

a[20][13] = 1;

}

if (p1422 < D) {

a[13][21] = 1;

a[21][13] = 1;

}

if (p1423 < D) {

a[13][22] = 1;

a[22][13] = 1;

}

//

if (p1516 < D) {

a[14][15] = 1;

a[15][14] = 1;

}

if (p1517 < D) {

a[14][16] = 1;

a[16][14] = 1;

}

if (p1518 < D) {

a[14][17] = 1;

a[17][14] = 1;

}

if (p1519 < D) {

a[14][18] = 1;

a[18][14] = 1;

}

if (p1520 < D) {

a[14][19] = 1;

a[19][14] = 1;

}

if (p1521 < D) {

a[14][20] = 1;

a[20][14] = 1;

}

if (p1522 < D) {

a[14][21] = 1;

a[21][14] = 1;

}

if (p1523 < D) {

a[14][22] = 1;

a[22][14] = 1;

}

//

if (p1617 < D) {

a[15][16] = 1;

a[16][15] = 1;

}

if (p1618 < D) {

a[15][17] = 1;

a[17][15] = 1;

}

if (p1619 < D) {

a[15][18] = 1;

a[18][15] = 1;

}

if (p1620 < D) {

a[15][19] = 1;

a[19][15] = 1;

}

if (p1621 < D) {

a[15][20] = 1;

a[20][15] = 1;

}

if (p1622 < D) {

a[15][21] = 1;

a[21][15] = 1;

}

if (p1623 < D) {

a[15][22] = 1;

a[22][15] = 1;

}

//

if (p1718 < D) {

a[16][17] = 1;

a[17][16] = 1;

}

if (p1719 < D) {

a[16][18] = 1;

a[18][16] = 1;

}

if (p1720 < D) {

a[16][19] = 1;

a[19][16] = 1;

}

if (p1721 < D) {

a[16][20] = 1;

a[20][16] = 1;

}

if (p1722 < D) {

a[16][21] = 1;

a[21][16] = 1;

}

if (p1723 < D) {

a[16][22] = 1;

a[22][16] = 1;

}

//

if (p1819 < D) {

a[17][18] = 1;

a[18][17] = 1;

}

if (p1820 < D) {

a[17][19] = 1;

a[19][17] = 1;

}

if (p1821 < D) {

a[17][20] = 1;

a[20][17] = 1;

}

if (p1822 < D) {

a[17][21] = 1;

a[21][17] = 1;

}

if (p1823 < D) {

a[17][22] = 1;

a[22][17] = 1;

}

//

if (p1920 < D) {

a[18][19] = 1;

a[19][18] = 1;

}

if (p1921 < D) {

a[18][20] = 1;

a[20][18] = 1;

}

if (p1922 < D) {

a[18][21] = 1;

a[21][18] = 1;

}

if (p1923 < D) {

a[18][22] = 1;

a[22][18] = 1;

}

//

if (p2021 < D) {

a[19][20] = 1;

a[20][19] = 1;

}

if (p2022 < D) {

a[19][21] = 1;

a[21][19] = 1;

}

if (p2023 < D) {

a[19][22] = 1;

a[22][19] = 1;

}

//

if (p2122 < D) {

a[20][21] = 1;

a[21][20] = 1;

}

//

if (p2223 < D) {

a[21][22] = 1;

a[22][21] = 1;

}

//

cout << "Граф :" << endl;

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

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

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

}

cout << endl;

}

system("pause");

}