Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Омский государственный технический университет»

Кафедра «Информатика и вычислительная техника»

Отчёт по лабораторной работе № 4

по дисциплине

«Проектирование и тестирование программного обеспечения»

|  |  |
| --- | --- |
|  | Выполнил  Студент гр. МО-181  Грачев Д. В.    (подп., дата)  Проверил  Старший преподаватель каф. ИВТ  Звонов А.О.    (подп., дата) |

Омск 2020

**Задача 7.3 (Станции ТО)**

PC/UVaIDs: 110804/10160

Компания занимается продажей персональных компьютеров в N городах  
(3 <N< 35), обозначаемых 1, 2, ..., N. М пар этих городов соединены прямыми дорогами. Компания решила построить несколько станций техобслуживания так, чтобы для любого города X станция находилась либо непосредственно в X, либо в каком-то городе, напрямую соединенном с Х,

Напишите программу, определяющую минимальное число станций, необходимых компании.

**Входные данные**

Входные данные состоят из нескольких вариантов условия. Каждый вариант начинается с числа городов N и числа пар М, разделенных пробелом. Каждая из следующих М строк содержит пару целых чисел, определяющих соединенные города, причем на одну строку приходится ровно одна пара чисел, разделенных пробелом. Входные данные завершаются при N = 0 и М = 0.

**Выходные данные**

Для каждого блока тестовых данных выведите строку, содержащую число необходимых станций техобслуживания.

Ниже представлена программа:

namespace L41

{

class Program

{

class Town : IComparable<Town>

{

public int id;

public Town(int a) { id = a; }

public List<Town> AdjTowns = new List<Town>();

public bool mark = false;

static public void LinkTowns(Town t1, Town t2)

{

t1.AdjTowns.Add(t2);

t2.AdjTowns.Add(t1);

}

static public Town GetTownFromList(Town[] towns, int townId)

{

foreach (var town in towns)

if (townId == town.id) return town;

return null;

}

public int CompareTo(Town t)

{

return this.AdjTowns.Count.CompareTo(t.AdjTowns.Count);

}

}

static void Main(string[] args)

{

using (StreamReader str = new StreamReader("file.txt"))

{

string input = str.ReadLine();

int n = Convert.ToInt32(input.Split(' ')[0]);

int m = Convert.ToInt32(input.Split(' ')[1]);

Town[] towns = new Town[n];

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

towns[i] = new Town(i + 1);

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

{

input = str.ReadLine();

int t1 = Convert.ToInt32(input.Split(' ')[0]);

int t2 = Convert.ToInt32(input.Split(' ')[1]);

Town.LinkTowns(Town.GetTownFromList(towns, t1), Town.GetTownFromList(towns, t2));

}

Array.Sort(towns);

int stations = 0;

for (int i = n - 1; i >= 0; i--)

{

bool Linked = false;

foreach (var adjTown in towns[i].AdjTowns)

{

if (adjTown.mark) Linked = true;

}

if (!Linked)

{

towns[i].mark = true;

stations++;

}

}

Console.WriteLine(stations);

}

}

}

}

Выбранный путь решения задачи:

1. Считыванием данные и по ним соединяем города между собой

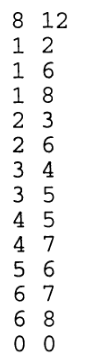
2. Сортируем города по количеству соседей

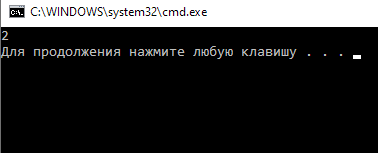
3. Для каждого города, от города с самым большим кол-вом городов соседей к городу с самым маленьким:если среди соседей есть город со станцией, пропустить город, если нет, поставить в нем станции.

4. Вывести получившееся количество станций.

Используемые структуры данных для решения задачи: массивы, списки, классы

Результат выполнения программы для входных данных:





**Задача 8.3 (Однонаправленная задача коммивояжера)**

РС/UVaIDs: 111104/116

Для заданной матрицы размером m х n вы должны написать программу, которая вычисляет путь с минимальным весом с левого края матрицы до правого. Путь начинается в любой строке первого столбца и состоит из последовательности шагов, обрывающихся в столбце n. Каждый шаг состоит в переходе из столбца i в столбец i + 1 в соседнюю (по горизонтали или диагонали) ячейку. Первая и последняя строки (строки 1 и m) матрицы считаются соседними; то есть матрица «сворачивается», образуя горизонтальный цилиндр.

Вес пути рассчитывается как сумма целых чисел, записанных в каждой из n посещенных ячеек.

Пути с минимальным весом через две немного отличные матрицы размером 5x6 показаны ниже. Значения матриц отличаются только в нижнем ряду. Путь, показанный для правой матрицы, использует соседство первого и последнего рядов.

**Входные данные**

Входные данные состоят из последовательности описаний матриц. Каждая матрица задается строкой с количеством строк и столбцов, которые обозначаются m и n соответственно. Далее следуют m х n целых чисел, идущих по строкам; то есть первые n целых чисел составляют первую строку матрицы, следующие n целых чисел составляют вторую строку и т. д. Целые числа, стоящие на одной строке, отделены друг от друга одним или более пробелами. Обратите внимание: целые числа не обязаны быть положительными. Во входном файле могут быть заданы одна или более матриц. Входные данные завершаются маркером конца файла.

Для каждого описания число рядов составляет от 1 до 10 включительно; число столбцов - от 1 до 100 включительно. Вес любого пути не превышает целого числа, для хранения которого потребуется больше 30 бит.

**Выходные данные**

Для каждого описания матрицы требуется вывести две строки. Первая строка задает путь минимальной стоимости, а вторая - собственно стоимость этого пути. Путь состоит из последовательности n целых чисел (разделенных одним или более пробелами), задающих ряды, из которых состоит путь минимальной стоимости. Если путей минимальной стоимости больше одного, то должен быть выведен лексикографически меньший путь.

Ниже представлена программа:

namespace L42

{

class Program

{

static void Main(string[] args)

{

using (StreamReader str = new StreamReader("file.txt"))

{

string input = null;

while ((input = str.ReadLine()) != null)

{

int m = Convert.ToInt32(input.Split(new char[] { ' ' })[0]);

int n = Convert.ToInt32(input.Split(new char[] { ' ' })[1]);

int[,] matrix = new int[m, n];

int[,] dp = new int[m, n];

int[,] next = new int[m, n];

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

{

input = str.ReadLine();

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

{

matrix[i, j] = Convert.ToInt32(input.Split(new char[] { ' ' })[j]);

}

}

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

{

dp[i, n - 1] = matrix[i, n - 1];

}

int y = 0;

for (int j = n - 2; j >= 0; j--)

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

{

int cur = Int32.MaxValue;

for (int k = -1; k <= 1; k++)

{

y = (i + k + m) % m;

if (dp[y, j + 1] < cur || (dp[y, j + 1] == cur && y < next[i, j]))

{

cur = dp[y, j + 1];

next[i, j] = y;

}

}

dp[i, j] = matrix[i, j] + cur;

}

int chainWeight = Int32.MaxValue;

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

if (dp[i, 0] < chainWeight)

{

chainWeight = dp[i, 0];

y = i;

}

Console.Write(y + 1);

for (int i = 0; i < n - 1; i++)

{

y = next[y, i];

Console.Write(" ");

Console.Write(y + 1);

}

Console.WriteLine();

Console.WriteLine(chainWeight);

}

}

}

}

}

Выбранный путь решения задачи:

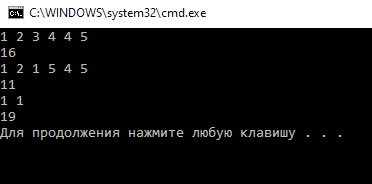
1. Считывание данных  
2. Обработка доп. матрицы с последнего столбца, т.к. требуется лексикографический порядок  
3. Если ячейка удовлетворяет условиям, она становиться следующей в пути

4. Вывод результатов

Используемые структуры данных для решения задачи: массивы, строки, классы

Результат работы программы при следующих входных данных:





Выводы:

В процессе выполнения лабораторной работы повторил материал теории графов, разобрался в сложных алгоритмах.