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 № 5

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 9.3 (Лесенки редактирования)**

PC/UVa IDs: 110905/10029

Назовем ступенькой редактирования (edit step) такое преобразование слова *x* в слово *y*, что слова *х* и *у* принадлежат словарю и слово *х* может быть преобразовано в слово *у* путем добавления, удаления или изменения одной буквы. Преобразования dig в dog и dog в do являются ступеньками редактирования. Лесенка ступенек редактирования (edit step ladder) - это отсортированная в описанном порядке последовательность слов. Для заданного словаря вы должны найти самую большую лесенку ступенек редактирования.

**Входные данные**

Входные данные состоят из словаря: набора слов в нижнем регистре в лексикографическом порядке по одному слову на строку. Длина каждого слова не превышает 16 букв, и в словаре не более 25 000 слов.

**Выходные данные**

Выходные данные состоят из одного целого числа, количества слов в самой большой лесенке ступенек редактирования.

Ниже представлена программа:

namespace L51

{

class Program

{

static List<int> chains = new List<int>();

static int CompareCount(string s1, string s2)

{

int difs = 0;

List<char> diffLetters = new List<char>();

if (s1.Length == s2.Length)

{

for (int i = 0; i < s1.Length; i++)

if (s1[i] != s2[i]) difs++;

}

else

{

int minL = s1.Length > s2.Length ? s2.Length : s1.Length;

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

if (s1[i] != s2[i]) difs++;

difs += Math.Abs(s1.Length - s2.Length);

}

return difs;

}

static void Chain(List<string> words, string prev, int curChain, string schain)

{

bool changed = false;

List<string> newWords = new List<string>(words);

newWords.Remove(prev);

foreach (var word in newWords)

{

if (CompareCount(prev, word) == 1)

{

changed = true;

Chain(newWords, word, curChain + 1, schain + word + " ");

}

}

if (!changed)

{

chains.Add(curChain);

Console.WriteLine(schain);

}

}

static int Start(List<string> words)

{

foreach(var word in words)

{

Chain(words, word, 1, word + " ");

}

return chains.Max();

}

static void Main()

{

List<string> words = new List<string>();

int maxChainLength = 0;

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

{

string input = null;

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

{

words.Add(input);

}

}

Console.WriteLine(Start(words));

}

}

}

**Важное замечание: я не совсем понял условие задачи. Можно ли ходить только соседним словам? Можно ли ходить вверх? Поэтому длина наибольшей цепочке у меня получилась больше, чем в примере.**

Выбранный путь решения задачи:

1. Считывание данных

2. Для каждого слова из словаря, запускаем рекурсию, на каждом шагу, если различия между словами – 1 символ, заносим слово в цепочку. Если такое сделать уже нельзя, т.е. либо отличий больше чем 1, заносим длину цепочки в массив, и переходим к другим словам.

3. Находим наибольшее число в массиве длин цепочек  
4. Выводим результат

Используемые структуры данных для решения задачи: массивы, кортежи, списки, строки

Результат выполнения программы для входных данных:

cat

dig

dog

fig

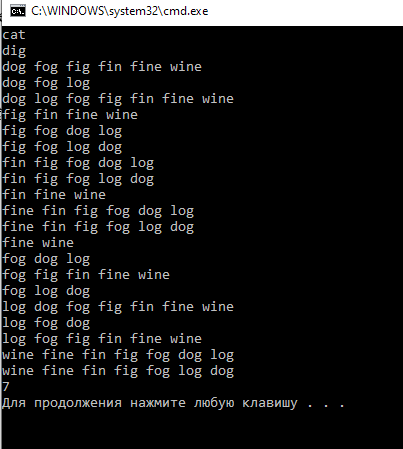
fin

fine

fog

log

wine



**На скриншоте можно увидеть все возможные цепочки. В последней строке – длина максимальной**

**Задача 10.3 (Большой обед)**

PC/UVa IDs: 111007/10249

Все команды, участвовавшие в ACM World Finals, приглашаются на большой банкет, который устраивается после церемонии награждения. Чтобы члены разных команд общались как можно больше, за одним столом не разрешается сидеть двум членам одной и той же команды.

Определите по заданному числу членов каждой команды (включая участников, тренеров, запасных и гостей) и числу мест за каждым столом, можно ли рассадить команды так, как описано выше. Если это возможно, то выведите один из возможных вариантов рассадки. Если существует несколько вариантов решения, то подойдет любой.

**Входные данные**

Входной файл может состоять из нескольких тестовых блоков. Первая строка каждого блока содержит два целых числа, 1 ≤ М ≤ 70 и 1 ≤ N ≤ 50, задающих число команд и столов соответственно. Вторая строка каждого тестового блока содержит М чисел: i-е число mi, показывает количество членов команды номер i. Число членов любой команды не превышает 100 человек. Третья строка содержит N чисел, причем j-е число nj, 2 ≤ nj ≤ 100, означает число мест за j-м столом.

Входные данные заканчиваются тестовым блоком, содержащим нули в качестве М и N.

**Выходные данные**

Для каждого тестового блока выведите строку, содержащую 1 или 0, указывающую на то, существует ли корректная рассадка членов команд. В случае положительного ответа выведите еще М строк, i-я из которых содержит номера столов (от 1 до N) для членов команды номер i.

Ниже представлена программа:

namespace L52

{

class Program

{

class Team : IComparable<Team>

{

public int Number;

public int Members;

public Team(int n, int members)

{

Number = n;

Members = members;

}

public int CompareTo(Team b)

{

return -this.Members.CompareTo(b.Members);

}

public List<int> Seats = new List<int>();

}

class Table

{

public int Number;

public int FreeSeats;

public Table(int n, int c)

{

Number = n;

FreeSeats = c;

}

}

static void Main()

{

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

{

string input = null;

while ((input = stream.ReadLine()) != "0 0")

{

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

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

List<Team> teams = new List<Team>();

int totalMembers = 0;

int[] teamsMembers = stream.ReadLine().Split(new char[] { ' ' }).Select(s => Convert.ToInt32(s)).ToArray();

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

{

teams.Add(new Team(i + 1, teamsMembers[i]));

totalMembers += teamsMembers[i];

}

List<Table> tables = new List<Table>();

int[] tableCapacity = stream.ReadLine().Split(new char[] { ' ' }).Select(s => Convert.ToInt32(s)).ToArray();

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

{

tables.Add(new Table(i + 1, tableCapacity[i]));

}

List<Team> sortedTeams = new List<Team>(teams);

sortedTeams.Sort();

foreach (var table in tables)

{

foreach (var team in sortedTeams)

{

if (table.FreeSeats > 0)

{

if (team.Members > 0)

{

table.FreeSeats--;

team.Members--;

totalMembers--;

team.Seats.Add(table.Number);

}

}

}

}

if (totalMembers == 0)

{

Console.WriteLine(1);

teams.OrderBy(x => x.Number);

foreach (var team in teams)

{

foreach (int seat in team.Seats)

Console.Write($"{seat} ");

Console.WriteLine();

}

}

else Console.WriteLine(0);

}

}

}

}

}

Выбранный путь решения задачи:

1. Считывание данных, создание массивов команд и столов.  
2. Сортировка команд по количеству людей в ней  
3. Для каждого стола и команды – если за столом есть свободные места, а в команде еще не сидящие люди, уменьшаем на 1 количество свободных мест за столом, количество несидящих участников команды, количество несидящих людей вообще на мероприятии, добавляем в список столов команды текущий стол

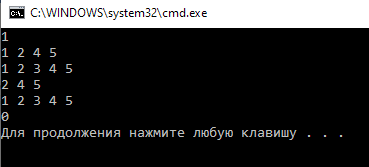
4. Если количество несидящих участников = 0, выводим 1 и порядок столов.

В обратном случае - 0

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

Результат работы программы при следующих входных данных:





Выводы:

В процессе выполнения лабораторной работы развил навыки работы с массивами и списками, потренировался в подобных видах задач.