|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО УЧЕБНО-ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ**

Студент Козлова Ирина Васильевна

Группа ИУ7-12Б

Тип практики Распределительная практика

Название предприятия МГТУ им. Н. Э. Баумана

Студент \_­\_\_\_\_\_\_\_\_\_\_\_ Козлова И.В.

*подпись, дата фамилия, и.о.*

Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_ Борисов С.В.

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2019 г.*

**Оглавление**

Введение……..…..……………………………………...…………………………3

1. Условие задачи……………………………………...…………………………4
2. Схема программы……………………………………………………………...7
3. Описание программы………………………………………………………...21
4. Текст программы……………………………………………………………..22
5. Заключение…………………………………………………………………...27
6. Список литературы…………………………………………………………..28

**Введение**

**Цели и задачи**: составить программу, которая составляет двойной линейный кроссворд из считанных из файла слов в количестве до 1000 и длиной от 2 до 20 символов. Кроссворд должен иметь как минимум 2 расшифровки и быть определенной длины, которая задается в начале файла (до 50 символов). Слова из первой расшифровки могут не появиться во второй и наоборот. Кроме того, ни одно слово не может быть повторено в любой расшифровке.

**Условие задачи**

*1997-98 ACM North-Eastern European Regional Programming Contest*

Problem D  
Crossword

|  |  |
| --- | --- |
| Input file | INPUT.TXT |
| Time-limit/Test | 20 seconds |

A double linear crossword of length L is a string of L lowercase alphabetic characters arranged in a line in such a way that there are at least two methods (so called decompositions) to split this string into the words from the given list. Look at the example for L=17:

| | | |

a n d a t a r e a l l a s t a s k

| | | | |

The words were taken from the following list: all, an, and, are, area, as, ask, at, data, last, or, read, real, task.

The words from the first decomposition may not appear in the second one and vice versa. In addition, no word can be repeated in any decomposition.

No word in one decomposition can end in the same place of the string where a word in the other composition ends, except, naturally, for the end of the string (otherwise the crossword can be separated into two independent crosswords). One of the compositions may consist of a single word.

You should write a program to construct the first, in lexicographic order, double linear crossword of length L for a given list of words.

Strings are arranged in the lexicographic order with respect of the following rules:

* If the first letter of a string appears in latin alphabet before the first letter of another string, then the former string precedes in lexicographic order.
* If the first letters of some strings match, then the corresponding letters of these strings are compared until they stop matching.
* If a mismatching is not found, the shorter string goes first.

Input

The first line of the input file consists of the single integer number L (4<=L<=50) denoting the desired crossword length. The second line consists of the single positive integer N (at most 1000) indicating the number of words in the list. Each of the followings N lines consists of a string of 20 or less (but at least 2) latin lowercase alphabetic characters. The words in the list are arranged in lexicographic order and no word is repeated.

Output

For the given input data set your program should write to the output file the first, in lexicographic order, double linear crossword with the given length. If it is impossible for the given input file to construct a double linear crossword with the given length, the program should write only the message "NO SOLUTION" (without the quotation marks).

Sample input

17

19

all

an

and

area

as

ask

at

data

do

for

last

of

or

ort

read

real

task

to

tor

Output for the sample input

andatareadofortor

**Перевод**

*1997-98 ACM Северо-восточноевропейский региональный конкурс по программированию*

Проблема D

Кроссворд

|  |  |
| --- | --- |
| Входной файл | INPUT.TXT |
| Ограничение по времени | 20 секунд | |

Двойной линейный кроссворд длины L представляет собой строку из L строчных буквенных символов, расположенных в строке таким образом, что существует по крайней мере два метода (так называемые декомпозиции) для разбиения этой строки на слова из данного списка. Посмотрите на пример для L=17:

| | | |

a n d a t a r e a l l a s t a s k

| | | | |

Слова были взяты из следующего списка: all, an, and, are, area, as, ask, at, data, last, or, read, real, task.

Слова из первого разложения могут не появиться во втором и наоборот. Кроме того, ни одно слово не может быть повторено в любом разложении.

Ни одно слово в одной декомпозиции не может заканчиваться в том же месте строки, где заканчивается слово в другой композиции, за исключением, естественно, конца строки (в противном случае кроссворд может быть разделен на два независимых кроссворда). Одна из композиций может состоять из одного слова.

Вы должны написать программу для построения первого, в лексикографическом порядке, двойного линейного кроссворда длины L для данного списка слов.

Строки располагаются в алфавитном порядке с соблюдением следующих правил:

* Если первая буква строки появляется в латинском алфавите перед первой буквой другой строки, то первая строка предшествует в алфавитном порядке.
* Если первые буквы некоторых строк совпадают, то соответствующие буквы этих строк сравниваются до тех пор, пока они не перестанут совпадать.
* Если несоответствие не найдено, то более короткая строка идет первой.

Ввод

Первая строка входного файла состоит из одного целого числа L (4<=L<=50), обозначающего нужную длину кроссворда. Вторая строка состоит из одного положительного целого числа N (не более 1000), указывающего количество слов в списке. Каждая из следующих N строк состоит из строки из 20 или менее (но не менее 2) латинских строчных буквенных символов. Слова в списке расположены в алфавитном порядке, и ни одно слово не повторяется.

Выход

Для данного набора входных данных ваша программа должна записать в выходной файл первый, в лексикографическом порядке, двойной линейный кроссворд заданной длины. Если для данного входного файла невозможно построить двойной линейный кроссворд заданной длины, то программа должна написать только сообщение " NO SOLUTION " (без кавычек).

Входной файл (образец)

17

19

all

an

and

area

as

ask

at

data

do

for

last

of

or

ort

read

real

task

to

tor

Результат:

andatareadofortor

**Схема программы**

Начало

Создаем список Words

Открываем файл input.txt

Считываем первую строчку и записываем ее целое значение в n

Считываем все строки из файла и записываем в список Word

Удаляем поcледний элемент каждого элемента из списка word

Закрываем файл F

String1,String2,UsedWords,

End = FindFirstFonema

(Words,AllFonems,n)

Создаем список AllFonems

Цикл i

i от 0 до L-1

A

А

CombineAllFonems

(AllFonems,

FonemsInCurrentWord)

i:=i+1

Цикл i

FonemsInCurrentWord = FindFonemsInWord

(Words[i],Words)

Открываем файл output.txt и привязываем его к переменной Output

Конец

End1

False True

В файл Output записать значение переменной String1

В файл Output записать ‘NO SOLUTION’

1. Функция FindFonemsInWord:

Начало

analyzable\_word,

word

lendth\_word := длина analyzable\_word

index\_analyzable\_word := индекс analyzable\_word в списке words

Список analyzable\_words :=элементы списка Word c index\_analyzable\_word+1

Цикл i

От 0 до length\_word-1

Цикл j

От i+1 до length\_word

Fonema - элементы строки analyzable\_word c i по j

да

fonema есть в

names\_fonems

нет

В fonems\_words добавить список из fonema и analyzable\_word

В конец names\_fonema добавить fonema

В

Б

Б

В

index\_fonema = (длина fonems\_words) - 1

j := j+1

Цикл j

Цикл word

word элементы списка analyzable\_words

да

нет

fonema есть в

word

в fonems\_words[index\_fonema][1] добавить word

Цикл Word

нет

Fonems\_words[index\_fonema][1]=

[analyzable\_word]

да

Удалить

fonems\_words[index\_fonema]

j:=j+1

Цикл j

Г

Г

i:=i+1

Цикл i

i := 0

нет

i< длины fonems­\_words

да

E :=TestNewFonema

(fonems\_words[i], fonems\_words)

True

E

False

Удалить fonems\_word[i]

i: = i-1

i := i+1

fonems\_words

Конец

1. Функция TestNewFonems

Начало

fonema,fonems

condition := False

Сортируем fonema[1] и i[1]

(fonema[0] есть в i[0]) and (fonema[0] ≠ i[0]) and

(fonema[1] = i[1])

да

condition := True

Цикл i

i элементы списка fonems

нет

Цикл i

condition

Конец

1. Функция CombineAllFonems

Начало

old\_fonems,

new\_fonems

Цикл i

от 0 до длины списка old\_fonems

В names\_of\_old\_fonems добавить old\_fonems[i][0]

i:=i+1

Цикл i

Цикл i

i элементы списка new\_fonems

i[0] есть в

names\_of\_old\_fonems

да

В old\_fonems добавить i

нет

Цикл i

old\_fonems

Конец

1. Функция FindFirstFonema

Начало

words,fonems,

num

end1:=False

Цикл word1

word1 элементы

списка words

length := длина word1

Цикл word2

word2 элементы

списка words

нет

word1 = word2

word1=word2

(с 0 по length)

да

нет

да

string1 := word1

string2 := word2

В used\_words добавляем word1 и word2

string1,string2,en := NewFonema(string1,string2, fonems,used\_words,num)

Е

Ж

Д

Е

Ж

Д

True

en

end1 := True

False

Цикл word2

True

end1

False

Цикл word1

string1,string2,

used\_words,en

Конец

1. Функция NewFonema

Начало

string1,string2,

fonems,

used\_words,q1

Длина string1>

длина string2

нет

да

str1 := string1

str2 := string2

str1 := string2

str2 := string2

index\_start :=длина строки str1

index\_finish := (длина строки str2)+1

delta := разница длины str2 и длины str1

end1 := False

Цикл i

От index\_start+1 до index\_finish

old\_fonema- элементы str2 c index\_start до i

Цикл fonema

fonema in fonems

З

З

нет

М

да

old\_fonema=

fonema[0]

Цикл word

word элементы из fonema[1]

нет

да

Длина строки word =delta

quantity := (длина str1)+ (длина word)

нет

да

quantity = q1

нет

word=элементы

строки str2

c index\_start

condition

:=False

нет

да

да

Длина word > delta

condition:=

False

нет

condition:=

True

Элементы

строки str2 c index\_start есть

в word

да

conditio:=

False

condition:=

True

да

word есть в элементах строки str2 c

index\_start

нет

condition:=

True

К

И

И

К

condition:=

False

T:= длина old\_fonema

old\_fonema

= элементы строки

word до T

нет

да

p:=True

p:=False

нет

да

В used\_word нет word и p=True и condition=True

str1 := str1 + word

В used\_words добавить word

str1,str2,e := NewFonema(str1,str2,

fonems,used\_words,q1)

Л

Л

нет

да

(str1 = str2) и

(длина str1 = q1)

end :=True

Удалить word из used\_words

str1 := string1

str2 := string2

Цикл word

М

True

end1

False

Цикл fonema

True

end1

False

Цикл i

str1, str2,end

Конец

**Описание программы**

Задача: Составить двойной линейный кроссворд определенной длины из заданных слов (считанных из файла в список).

Шаги программы

1. В начале программы считываем длину кроссворда из входного файла.
2. Считываем все слова в список.
3. Обрабатываем этот список, удаляя последний символ, так как это “\n”.
4. Для каждого слова из списка находим фонемы.
5. Записываем все найденные фонемы в список всех фонем данного слова.
6. Проверяем найденные фонемы на совпадение с фонемами, найденными ранее, если находим совпадения, то удаляем такую фонему из списка найденных, иначе оставляем в общем списке.
7. Все фонемы из списка найденных фонем добавляем в общий список фонем.
8. Находим первое слово для кроссворда такое, чтобы оно являлось частью какого-либо другого слова.
9. Далее ищем каждое следующее слово, соблюдая правила построения кроссворда, до тех пор, пока длина кроссворда не будет равняться заданной или закончатся слова для составления кроссворда.
10. Если кроссворд не удалось составить по данным правилам, то выводим сообщение (NO SOLUTION) об этом, иначе выводим получившийся кроссворд в выходном файле.

**Текст программы**

’’’Функция FindFonemsInWord находит фонемы в слове, которые есть в других словах заданного списка и возвращает их[фонемы].

Используемые переменные:

analyzable\_word – анализируемое слово

analyzable\_words – все слова после анализируемого в общем списке слов

fonems\_words – фонемы в анализируемого слова

index\_analyzable\_word – индекс анализируемого слова в списке всех слов

lenght\_word - длина анализируемого слова

names\_fonems –фонемы, которые есть в анализируемом слове ’’’

def FindFonemsInWord(analyzable\_word, words):

fonems\_words = []

length\_word = len(analyzable\_word)

index\_analyzable\_word = words.index(analyzable\_word)

analyzable\_words = words[index\_analyzable\_word+1:]

names\_fonems = []

for i in range(length\_word):

for j in range(i+1,length\_word+1):

fonema = analyzable\_word[i:j]

if fonema in names\_fonems:

continue

fonems\_words.append([fonema[:],[analyzable\_word[:]]])

names\_fonems.append(fonema[:])

index\_fonema = len(fonems\_words) - 1

for word in analyzable\_words:

if fonema in word:

fonems\_words[index\_fonema][1].append(word)

if fonems\_words[index\_fonema][1] ==[analyzable\_word[:]]:

del fonems\_words[index\_fonema]

i = 0

while i < len(fonems\_words):

if TestNewFonema(fonems\_words[i], fonems\_words):

del fonems\_words[i]

i -= 1

i += 1

return fonems\_words

’’’Функция TestNewFonema проверяет, чтобы фонема не являлась частью другой фонемы и возвращает значение True или Falsе.

Использованные переменные:

fonems – список всех фонем

fonema – проверяемая фонема’’’

def TestNewFonema(fonema, fonems):

condition = False

for i in fonems:

fonema[1].sort()

i[1].sort()

if ((fonema[0] in i[0]) and (fonema[0]!=i[0]) and

(fonema[1] == i[1])):

condition = True

return condition

’’’Функция CombineAllFonems добавляет новые фонемы в общий список фонем и возвращает тот же список фонем с новыми (если такие есть) или неизмененный (если не было обнаружено новых фонем).

Использованные переменные:

name\_of\_old\_fonems – копия фонем до добавления

old\_fonems – список фонем до добавления

new\_fonems – новые фонемы анализируемого слова ’’’

def CombineAllFonems(old\_fonems, new\_fonems):

names\_of\_old\_fonems = []

for i in range(len(old\_fonems)):

names\_of\_old\_fonems.append(old\_fonems[i][0])

for i in new\_fonems:

if not (i[0] in names\_of\_old\_fonems):

old\_fonems.append(i)

return old\_fonems

‘’’Функция FindFirstFonema находит первые слова для кроссворда, одно из которых должно быть частью другого слова и возвращает две расшифровки, список использованных слов и значение True, если длина кроссворда равняется заданной, или значение False, если длина не равняется заданной.

Использованные переменные:

end1 – для определения, можно добавить или нет

word1, word2 - слова

length – длина word1

used\_words – использованные слова в кроссворде

string1, string2 – кроссворды‘’’

def FindFirstFonema(words,fonems,num):

end1 = False

for word1 in words:

length = len(word1)

for word2 in words:

if word1 == word2:

continue

if word1 == word2[:length]:

string1 = word1

string2 = word2

used\_words = [word1[:],word2[:]]

string1,string2,en = NewFonema(string1,string2,

fonems,used\_words,num)

if en:

end1 = True

break

if end1:

break

return string1,string2,used\_words,en

‘’’Функция NewFonema добавляет слова в кроссворд и возвращает обе расшифровки и значение True, если длина кроссворда равняется заданной, или значение False, если длина не равняется заданной.

Использованные переменные:

delta – количество анализируемых символов

quantity – длина получаемого кроссворда

index\_start – длина короткой строки

index\_finish – длина длинной строки

old\_fonema – анализируемая фонема

string1,string2 – кроссворды

condition – для определения, можно составить кроссворд или нет’’’

def NewFonema(string1,string2,fonems,used\_words,q1):

if len(string1)>len(string2):

string1,string2 = string2,string1

str1 = string1

str2 = string2

index\_start = len(str1)

index\_finish = len(str2)+1

delta = len(str2)-len(str1)

end1 = False

for i in range(index\_start+1,index\_finish):

old\_fonema = str2[index\_start:i]

for fonema in fonems:

if old\_fonema == fonema[0]:

for word in fonema[1]:

if len(word) == delta:

quantity = len(str1)+len(word)

if quantity == q1:

condition = (word ==str2[index\_start:])

else:

condition = False

elif len(word) > delta:

condition = (str2[index\_start:] in word)

else:

condition = (word in str2[index\_start:])

if (old\_fonema == word[:len(old\_fonema)]

and not(word in used\_words) and condition):

str1 += word

used\_words.append(word)

str1,str2,e =NewFonema(str1,str2,fonems,used\_words,q1)

if (str1 == str2) and (len(str1)==q1):

end1 = True

break

else:

del\

used\_words[used\_words.index(word)]

str1 = string1

str2 = string2

if end1:

break

if end1:

break

return str1,str2,end1

‘’’Основная программа

Использованные переменные:

Words - список слов из которых надо составить кроссворд

n - длина кроссворда

String1 - первая расшифровка

String2 - вторая расшифровка

UsedWords - слова, которые используются в кроссворде

End - для определения, можно составить кроссворд или нет’’’

Words = []

F = open('input.txt')

n = int(F.readline())

i = F.readline()

for i in F:

Words.append(i)

for i in range(len(Words)):

Words[i] = Words[i][:len(Words[i])-1]

F.close()

AllFonems = []

for i in range(len(Words)-1):

FonemsInCurrentWord = FindFonemsInWord(Words[i],Words)

CombineAllFonems(AllFonems,FonemsInCurrentWord)

String1,String2,UsedWords,End = FindFirstFonema(Words,AllFonems,n)

Output = open('Output.txt','w')

if not End1:

Output.write('NO SOLUTION')

else:

Output.write(String1)

Output.close()

**Заключение**

Решая данную задачу, я научилась составлять двойной линейный кроссворд, используя некоторые функции, параметры и методы языка Python, например: списки, строки, срезы, циклы, файлы и операции над ними.

В процессе работы над данной проблемой я научилась применять навыки программирование на языке Python в решении нестандартных задач, а так же приобрела важные практические навыки, которые позволят мне в будущем продолжить изучение своей предметной области более углубленно и досконально.

**Список литературы**

1. Лутц Марк Python. Карманный справочник, 5-е изд. :Пер. с англ. – М. : ООО «И.Д. Вильямс», 2015. – 320с. : ил. – Парал. тит. англ.
2. Васильев А. Н. Python на примерах Практический курс по программированию. – СПб.: Наука и техника, 2016. – 432с.: ил.