**Московский государственный технический университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования» Отчет по лабораторной работе №2

«Расстояние Левенштейна.»

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент группы ИУ5-31Б | преподаватель каф. ИУ5 |
| Зобнин Александр |  |
| Подпись и дата: | Подпись и дата: |

# Постановка задачи

**Задание:**

Разработать программу для нахождения расстояния Левенштейна.

# Текст программы

**main.py**

import sys

def get\_str(index, prompt):

'''

Читаем слово из командной строки или вводим с клавиатуры

Args:

index (int): Номер параметра в командной строке

prompt (str): Приглашение для ввода коэффицента

Returns:

word (str): Слово

'''

try:

# Пробуем прочитать слово из командной строки

word = sys.argv[index]

except:

# Вводим с клавиатуры

print(prompt)

word = input()

return word

def levenstein(str\_1, str\_2):

'''

Вычисление корней квадратного уравнения

Args:

str\_1 (str): Первое слово

str\_2 (str): Второе слово

Returns:

result (int): Расстояние Левенштейна

'''

n, m = len(str\_1), len(str\_2)

if n > m:

str\_1, str\_2 = str\_2, str\_1

n, m = m, n

current\_row = range(n + 1)

for i in range(1, m + 1):

previous\_row, current\_row = current\_row, [i] + [0] \* n

for j in range(1, n + 1):

add, delete, change = previous\_row[j] + 1, current\_row[j - 1] + 1, previous\_row[j - 1]

if str\_1[j - 1] != str\_2[i - 1]:

change += 1

current\_row[j] = min(add, delete, change)

result = current\_row[n]

return result

def main():

'''

Основная функция

'''

a = get\_str(1, 'Введите первое слово:')

b = get\_str(2, 'Введите второе слово:')

# Вычисление расстояния Левенштейна

print("Расстояние Левенштейна =", levenstein(a, b))

# Если сценарий запущен из командной строки

if \_\_name\_\_ == "\_\_main\_\_":

main()

# Примеры запуска

# main.py самолет самокат (2)

**main.swift**

import Foundation

extension String {

subscript(index: Int) -> Character {

return self[self.index(self.startIndex, offsetBy: index)]

}

}

extension String {

public func levenshtein(\_ other: String) -> Int {

guard self != other else {

return 0

}

guard self.count != 0 else {

return other.count

}

guard other.count != 0 else {

return self.count

}

let line: [Int] = Array(repeating: 0, count: other.count + 1)

var matrix: [[Int]] = Array(repeating: line, count: self.count + 1)

for i in 0...self.count {

matrix[i][0] = i

}

for j in 0...other.count {

matrix[0][j] = j

}

for j in 1...other.count {

for i in 1...self.count {

if self[i - 1] == other[j - 1] {

matrix[i][j] = matrix[i - 1][j - 1]

} else {

let deletion = matrix[i - 1][j] + 1

let insertion = matrix[i][j - 1] + 1

let substitution = matrix[i - 1][j - 1] + 1

matrix[i][j] = min(min(deletion, insertion), substitution)

}

}

}

return matrix[self.count][other.count]

}

}

// Usage

print("Input first word:")

var a = readLine()!

print("Input second word:")

var b = readLine()!

// Levenshtein distance calculation

print("Levenshtein distance =", a.levenshtein(b))

// ex: abc, adb (2)

# Анализ результатов

# 

# 