Национальный исследовательский Университет ИТМО
Мегафакультет информационных и трансляционных технологий
Факультет инфокоммуникационных технологий

Алгоритмы и структуры данных

Лабораторная работа №1

Работу выполнил:
А.Э. Филиппов
Группа: К3139
Преподаватель:
Д. Добриборщ

Санкт-Петербург
2022

Содержание

1. Задачи по варианту

1.1. Задача №3. Редакционное расстояние

Редакционное расстояние между двумя строками – это минимальное количе ство операций (вставки, удаления и замены символов) для преобразования одной строки в другую. Это мера сходства двух строк. У редакционного расстояния есть применения, например, в вычислительной биологии, обработке текстов на есте ственном языке и проверке орфографии. Ваша цель в этой задаче – вычислить расстояние редактирования между двумя строками.

- **Формат ввода / входного файла (input.txt).** Каждая из двух строк ввода содержит строку, состоящую из строчных латинских букв. Длина обеих строк - от 1 до 5000.
- **Формат вывода / выходного файла (output.txt).** Выведите расстояние ре дактирования между заданными двумя строками.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt	input.txt	output.txt	input.txt	output.txt
ab ab	0	shorts ports	3	editing distance	5

- Редакционное расстояние во втором примере равно 3:

s	h	o	r	t	-
-	p	o	r	t	s

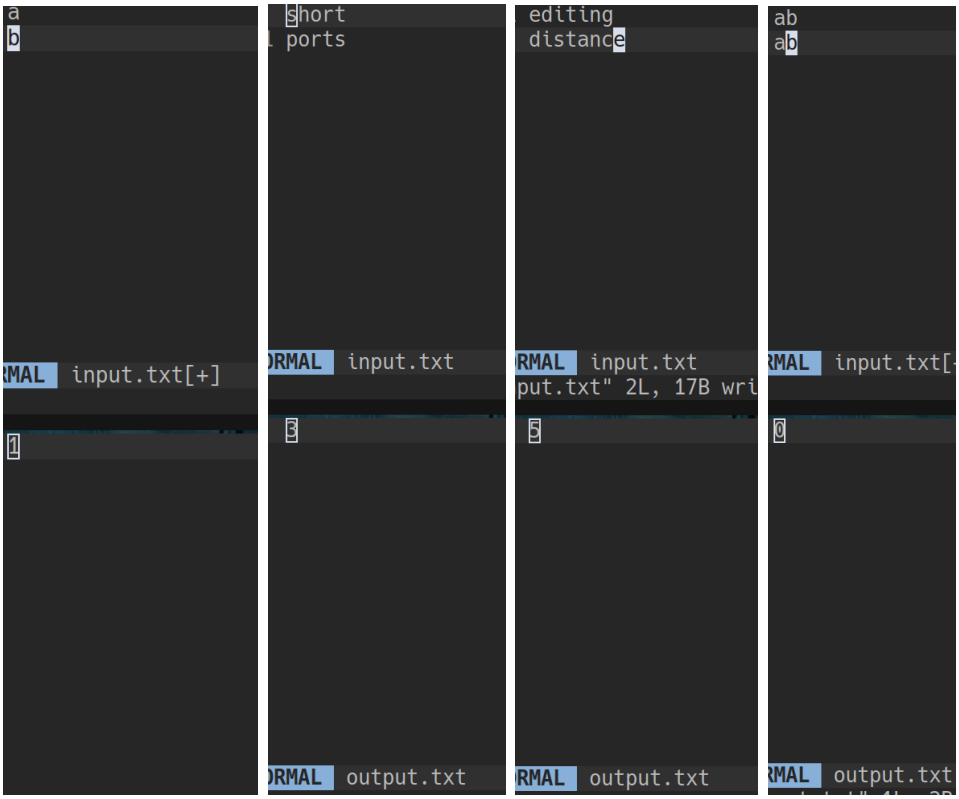
Код:

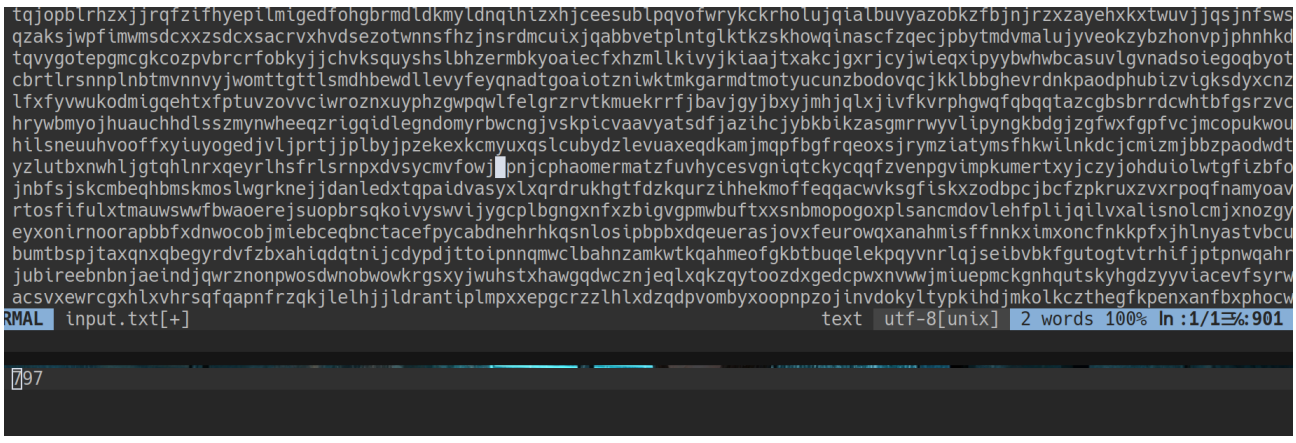
```
import time
import tracemalloc

tracemalloc.start()
t_start = time.perf_counter()
file = open('input.txt')
lines = file.readlines()
out = open("output.txt", "w")

a = lines[0]
b = lines[1]

n, m = len(a), len(b)
if n > m:
    a, b = b, a
    n, m = m, n

curr_row = []
for i in range(n+1):
    curr_row.append(i)

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

```
    prev_row, curr_row = curr_row, [i] + [0] * n
    for j in range(1, n + 1):
        move_aus_a = prev_row[j] + 1
        move_aus_b = curr_row[j-1] + 1
        fortsetzen = prev_row[j-1]
        if a[j-1] != b[i-1]:
            fortsetzen += 1
        curr_row[j] = min(move_aus_a, move_aus_b, fortsetzen)

out.write(str(curr_row[n]))

```

Для решения данной задачи использовалось расстояние Левенштайна. Вместо того, чтобы сохранять массив для ускорения работы сохраняются последняя строка и столбец.

Результат работы кода:





	Время выполнения (с)	Затраты памяти (байт)
Нижняя граница диапазона значений входных данных из текста задачи	0.00017152299915323965	13939
Пример из задачи 1	0.0001822340000217082	13941
Пример из задачи 2	0.00019643999985419214	13947
Пример из задачи 3	0.0002520400003049872	13952
Верхняя граница диапазона значений входных данных из текста задачи	1.9091916950001178	92904

Вывод по задаче: Расстояние Левенштайна считается самым эффективным алгоритмом для расчета редакционного расстояния. Его сложность - O(n^2)

1.2. Задача №4. Наибольшая общая подпоследовательность

Вычислить длину самой длинной общей подпоследовательности из двух по следовательностей. Даны две последовательности A = (a1, a2, ..., an) и B = (b1, b2, ..., bm), найти длину их самой длинной общей подпоследовательности, т.е. наибольшее неотри цательное целое число p такое, что существуют индексы 1 ⊠ i1 < i2 < ... < ip ⊠ n и 1 ⊠ j1 < j2 < ... < jp ⊠ m такие, что ai1 = bj1, ..., aip = bjp.

2. Теоретическая информация

bash [**bash**]

3. Ход выполнения работы

3.1. Список

- первый элемент списка
- второй элемент списка

3.2. Картинка

Рисунок 3.1. название картинки

Текст без отступа (следует за вставкой)

Новый параграф

Новый параграф с принудительно выключенным отступом

3.3. Таблицы

Таблица 3.1

Одна таблица

Element	First	Second
One	-	-
Two	-	-
Three	-	-
Four	-	-

Таблица 3.2

Другая таблица

top left	top center	top right
bot left	bot center	bot right

3.4. Листинг

```
#!/bin/bash

export LANG=en_US.UTF-8

dir_c=-1
file_c=0

cool() {
        dir_c=$(($dir_c + 1))
        local directory=$1
        local prefix=$2

        local leaves=($directory/*)
        local local_c=${#leaves[@]}

        for i in "${!leaves[@]}"; do
                local leaf=${leaves[$i]}
                leaf=${leaf##*/}

                local local_prefix="│   "
                local pointer="├── "

                if [ $i -eq $((local_c - 1)) ]; then
                        pointer="└── "
                        local_prefix="    "
                fi

                echo "${prefix}${pointer}$leaf"
                [ -d "$directory/$leaf" ] && cool "$directory/$leaf"
 ↪  "${prefix}$local_prefix" || file_c=$((file_c + 1))

        done
}

```

Листинг 1: Script.bash – bash в массы!

Заключение

LaTeX удобен для создания отчётов, так как сам следит за нумерацией таблиц, рисунков, листингов и отсылок к ним (так, например, здесь всегда будет указан номер рисунка "sample" не зависимо от того, какой он (1,2 или другой) - это рисунок 3.1). Не менее важно что весь документ оформлен в едином стиле, а исходные материалы подключаются к отчёту, а не хранятся в нём. Всё это позволяет легко получить качественный отчёт без дополнительных трат на его офрмление.

Исключения, пожалуй, составляют таблицы, так как их значительно сложнее создавать кодом, нежели в графическом редакторе. Но здесь никто не запрещает использовать визуальные средства создания таблиц для LaTeX .