САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №7

по курсу «Алгоритмы и структуры данных»

Вариант 1

Выполнил:

Азаренков Георгий Денисович

К34421

Проверил:

Афанасьев А.В.

Санкт-Петербург

2024 г.

# Содержание отчета

[Содержание отчета 2](#_Toc179012155)

[Комментарий к общему коду 3](#_Toc179012156)

[Задача 1 4](#_Toc179012157)

[Текст задачи 4](#_Toc179012158)

[Код решения 4](#_Toc179012159)

[Результат выполнения 5](#_Toc179012160)

[Вывод 6](#_Toc179012161)

# Комментарий к общему коду

При использовании всех задач я использую код из файла “common.py”, который находится в корне репозитория.

from typing import List, Callable  
  
  
def solve(solver: Callable[[List[str]], List[str]]) -> None:  
 with open("../../input.txt", "r") as file:  
 lines = file.readlines()  
 input\_lines = list(map(lambda x: x.strip(), lines))  
  
 output\_lines = solver(input\_lines)  
  
 with open("../../output.txt", "w") as file:  
 for line in output\_lines:  
 file.write(f"{line}\n")

Использование функции “solve” позволило мне избавиться от необходимости раз за разом имплементировать логику чтения и сохранения данных.

# Задача 4

## Текст задачи

A screenshot of a computer

Description automatically generated

## Код решения

from typing import List  
  
from common import solve  
  
  
def get\_solution(input\_lines: List[str]) -> List[str]:  
 *"""  
 Функция для вычисления длины самой длинной общей подпоследовательности двух последовательностей.  
 """* n = int(input\_lines[0])  
  
 first\_sequence = list(map(int, input\_lines[1].split()))  
  
 m = int(input\_lines[2])  
  
 second\_sequence = list(map(int, input\_lines[3].split()))  
  
 *# Инициализация массив для динамического программирования* dp = [[0] \* (m + 1) for \_ in range(n + 1)]  
  
 *# Заполнение массив* for i in range(1, n + 1):  
 for j in range(1, m + 1):  
 if first\_sequence[i - 1] == second\_sequence[j - 1]:  
 dp[i][j] = dp[i - 1][j - 1] + 1  
 else:  
 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])  
  
 *# Длина самой длинной общей подпоследовательности* lcs\_length = dp[n][m]  
  
 return [str(lcs\_length)]  
  
  
solve(get\_solution)

## Результаты выполнения

Пример 1

input.txt

A white background with black numbers

Description automatically generated

output.txt

A number on a calendar

Description automatically generated

Пример 2

input.txt

A close up of numbers

Description automatically generated

output.txt

A screenshot of a computer

Description automatically generated

Пример 3

input.txt

A close up of numbers

Description automatically generated

output.txt

A close up of a calendar

Description automatically generated

# Задача 5

## Текст задачи

A screenshot of a cell phone

Description automatically generated

## Код решения

from typing import List  
  
from common import solve  
  
  
def get\_solution(input\_lines: List[str]) -> List[str]:  
 *"""  
 Вычисляет длину самой длинной общей подпоследовательности из трех последовательностей.  
 """* n = int(input\_lines[0])  
  
 first\_sequence = list(map(int, input\_lines[1].split()))  
  
 m = int(input\_lines[2])  
  
 second\_sequence = list(map(int, input\_lines[3].split()))  
  
 l = int(input\_lines[4])  
  
 third\_sequence = list(map(int, input\_lines[5].split()))  
  
 *# Инициализация 3D массива для динамического программирования* dp = [[[0] \* (l + 1) for \_ in range(m + 1)] for \_\_ in range(n + 1)]  
  
 *# Заполнение массива* for i in range(1, n + 1):  
 for j in range(1, m + 1):  
 for k in range(1, l + 1):  
 if first\_sequence[i - 1] == second\_sequence[j - 1] and first\_sequence[i - 1] == third\_sequence[k - 1]:  
 dp[i][j][k] = dp[i - 1][j - 1][k - 1] + 1  
  
 else:  
 dp[i][j][k] = max(dp[i - 1][j][k], dp[i][j - 1][k], dp[i][j][k - 1])  
  
 *# Получение результата из DP таблицы* result = dp[n][m][l]  
  
 return [str(result)]  
  
  
solve(get\_solution)

## Результаты выполнения

Пример 1

input.txt

A screenshot of a computer

Description automatically generated

output.txt

A screenshot of a computer

Description automatically generated

Пример 2

input.txt

A screenshot of a computer

Description automatically generated

output.txt

A close up of a calendar

Description automatically generated

# Вывод

В ходе данной лабораторной работы я научился решать задачи по теме. Написанные программы были протестированы с измерением затрат времени и оперативной памяти. Все программы работаю корректно и укладываются в установленные ограничения.