САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №7
по курсу «Алгоритмы и структуры данных»
Тема: Динамическое программирование №1
Вариант 14

Выполнила:
Рудникова Виктория Олеговна
К3125

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Задачи по варианту

Задача №4. Наибольшая общая подпоследовательность двух последовательностей

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

• Формат ввода / входного файла (input.txt).

– Первая строка: n - длина первой последовательности.

– Вторая строка: a1, a2, ..., an через пробел.

– Третья строка: m - длина второй последовательности.

– Четвертая строка: b1, b2, ..., bm через пробел.

• Ограничения: $1 \le n, m \le 100$; −10**9 < ai, bi < 10**9.

• Формат вывода / выходного файла (output.txt). Выведите число p.

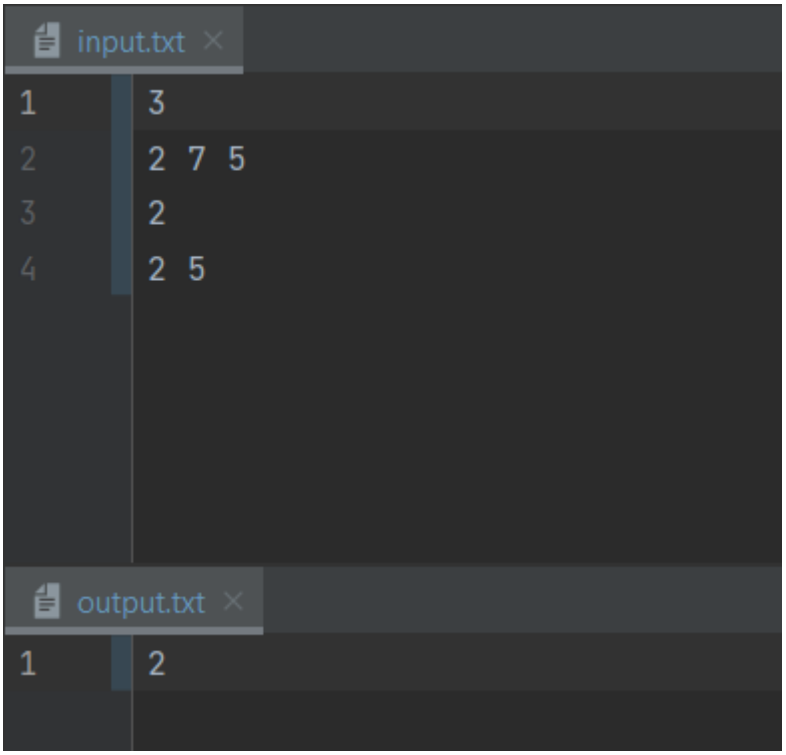
• Ограничение по времени. 1 сек.

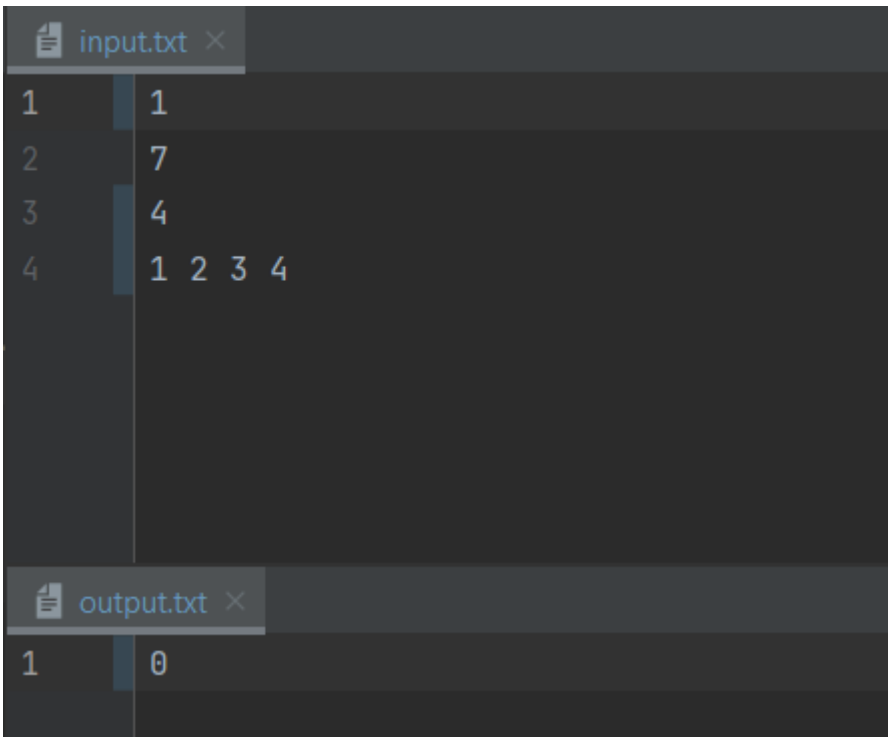
```
from time import process_time
from tracemalloc import start, get_traced_memory


def lcs(arr1, arr2):
    n = len(arr1)
    m = len(arr2)
    dp = [[0] * (m + 1) for i in range(n + 1)]
    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if arr1[i - 1] == arr2[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])
    return dp[n][m]


if __name__ == '__main__':
    start()
    with open('input.txt', 'r') as f:
        a = int(f.readline())
        seq1 = [int(i) for i in f.readline().split()]
        b = int(f.readline())
        seq2 = [int(i) for i in f.readline().split()]
    with open('output.txt', 'w') as g:
```

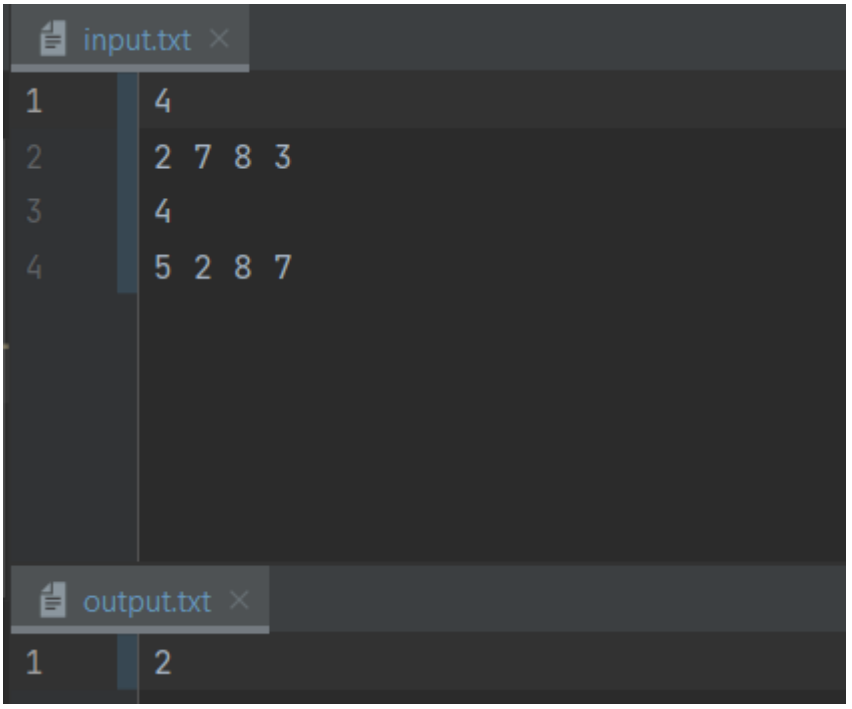
```
        g.write(str(lcs(seq1, seq2)))

    print('Time:', str(process_time()), 'sec')
    print('Memory usage:', str(get_traced_memory()[1] / 1024), 'KB')
```

Длина общих подпоследовательностей хранится в двумерном списке, каждая мера которого отвечает за одну из данных последовательностей. Длина общей подпоследовательности для каждой ячейки списка определяется его предыдущими ячейками.

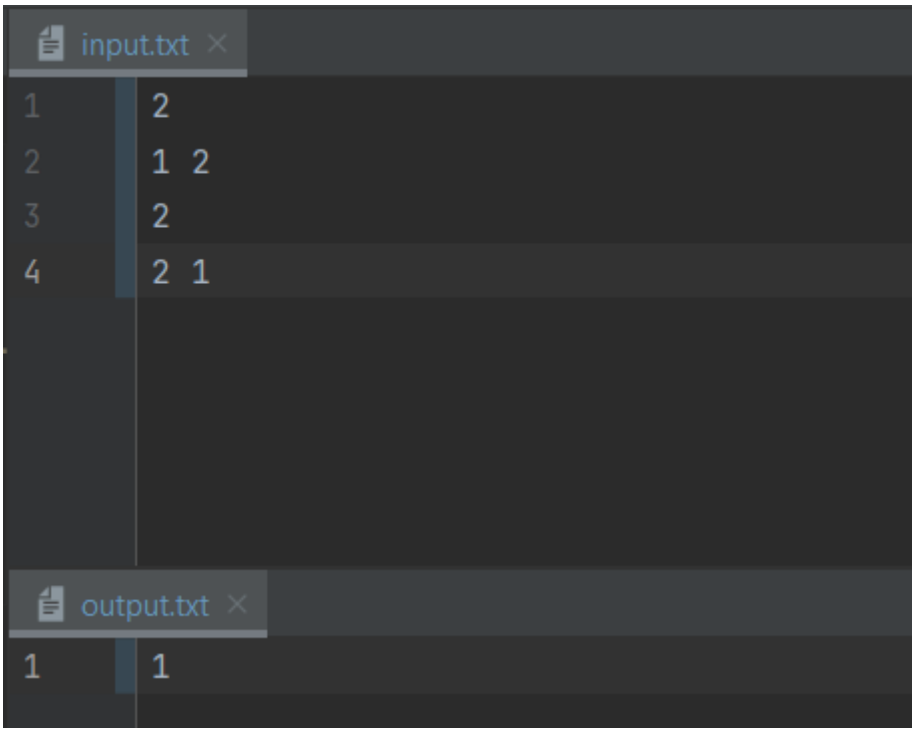
Результат работы кода на примерах из текста задачи:







Результат работы кода на максимальных и минимальных значениях:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.078375 sec	17.244140625 KB

Пример из задачи	0.078375 sec	17.244140625 KB
Пример из задачи	0.078375 sec	17.244140625 KB
Пример из задачи	0.078375 sec	17.244140625 KB
Верхняя граница диапазона значений входных данных из текста задачи	0.078375 sec	17.244140625 KB

Вывод по задаче: в задаче я реализовала алгоритм поиска длины наибольшей общей подпоследовательности для двух последовательностей.

Задача №5. Наибольшая общая подпоследовательность трёх последовательностей

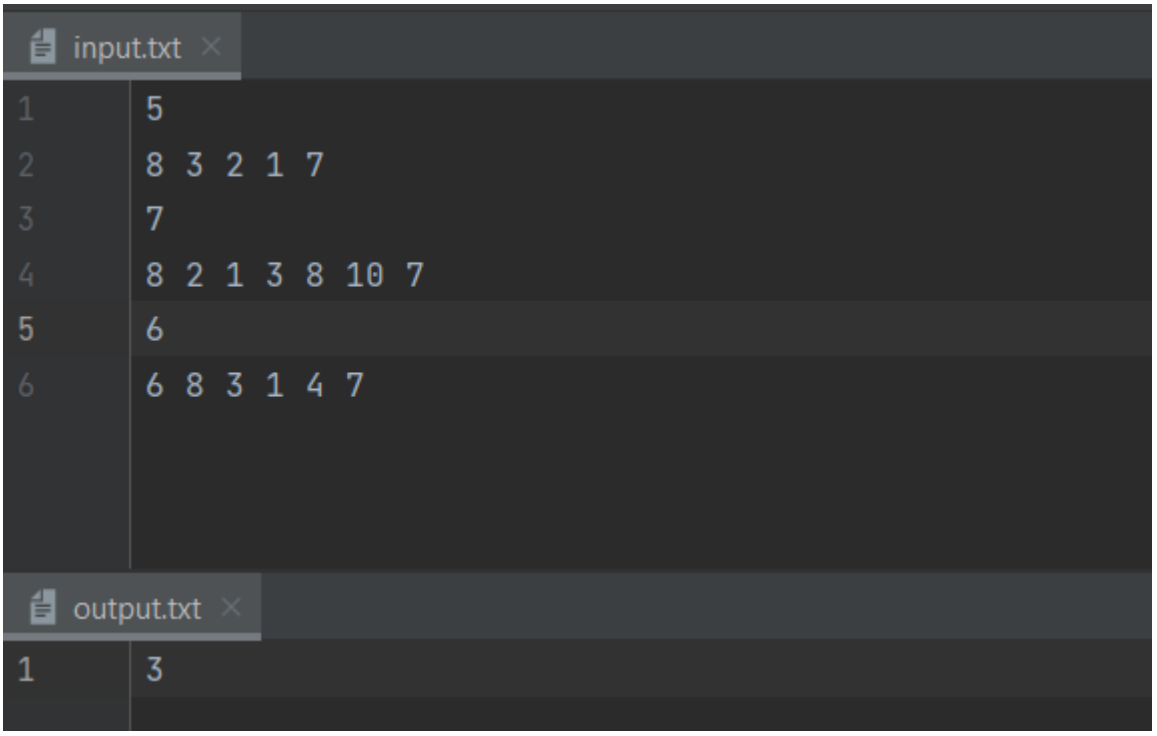
Вычислить длину самой длинной общей подпоследовательности из трех последовательностей.
Даны три последовательности A = (a1, a2, ..., an), B = (b1, b2, ..., bm) и C = (c1, c2, ..., cl), найти длину их самой длинной общей подпоследовательности, т.е. наибольшее неотрицательное целое число p такое, что существуют индексы $1 \le i1 < i2 < ... < ip \le n$, $1 \le j1 < j2 < ... < jp \le m$ и $1 \le k1 < k2 < ... < kp \le l$ такие, что ai1 = bj1 = ck1, ..., aip = bjp = ckp.
• Формат ввода / входного файла (input.txt).
– Первая строка: n - длина первой последовательности.
– Вторая строка: a1, a2, ..., an через пробел.
– Третья строка: m - длина второй последовательности.
– Четвертая строка: b1, b2, ..., bm через пробел.
– Пятая строка: l - длина второй последовательности.
– Шестая строка: c1, c2, ..., cl через пробел.
• Ограничения: $1 \le n, m, l \le 100$; −10**9 < ai, bi, ci < 10**9.
• Формат вывода / выходного файла (output.txt). Выведите число p.
• Ограничение по времени. 1 сек.

```
from time import process_time
from tracemalloc import start, get_traced_memory
```

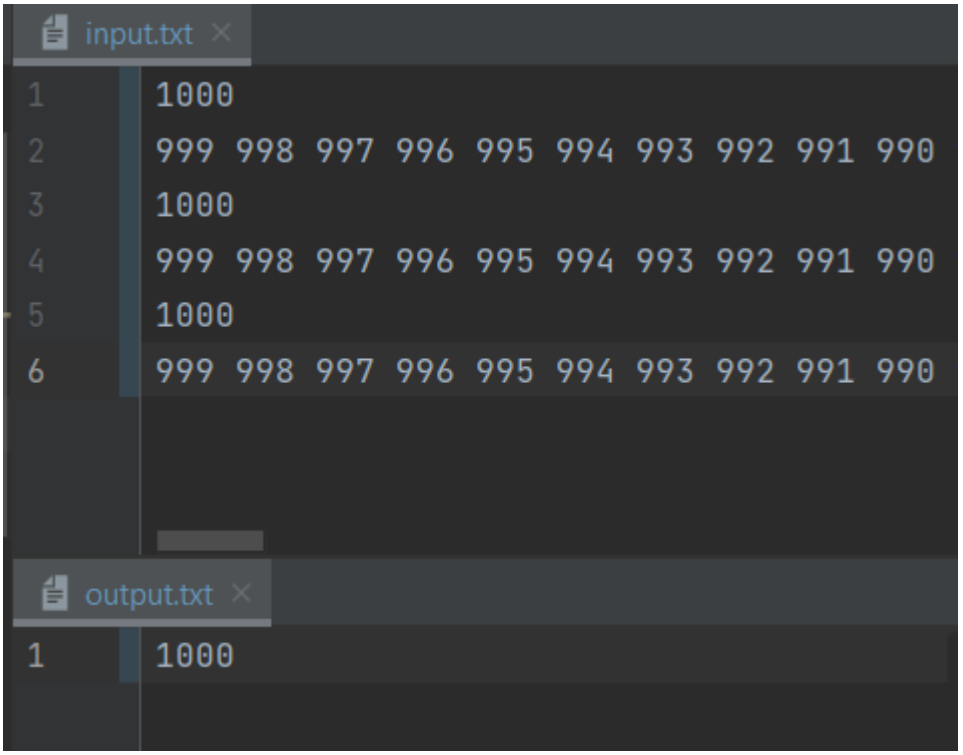
```
def lcs(arr1, arr2, arr3):
    n = len(arr1)
    m = len(arr2)
    l = len(arr3)
    dp = [[[0] * (l + 1) for _ in range(m + 1)] for _ in range(n + 1)]
    for i in range(1, n + 1):
        for j in range(1, m + 1):
            for k in range(1, l + 1):
                if arr1[i - 1] == arr2[j - 1] and arr3[k - 1] == arr2[j - 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])
    return dp[n][m][l]


if __name__ == '__main__':
    start()
    with open('input.txt', 'r') as f:
        a = int(f.readline())
        seq1 = [int(i) for i in f.readline().split()]
        b = int(f.readline())
        seq2 = [int(i) for i in f.readline().split()]
        c = int(f.readline())
        seq3 = [int(i) for i in f.readline().split()]
    with open('output.txt', 'w') as g:
        g.write(str(lcs(seq1, seq2, seq3)))

    print('Time:', str(process_time()), 'sec')
    print('Memory usage:', str(get_traced_memory()[1] / 1024), 'KB')
```

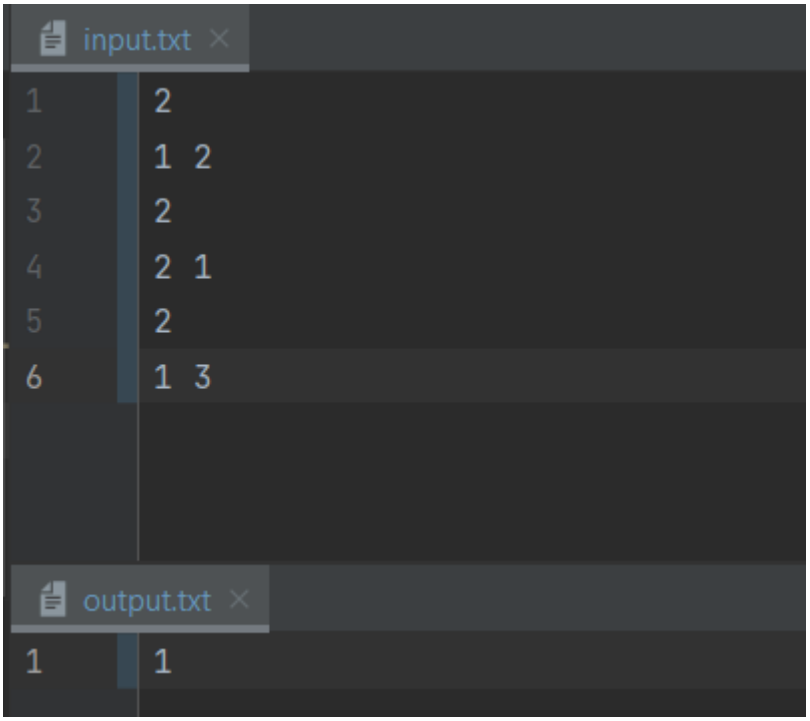
Длина общих подпоследовательностей хранится в трёхмерном списке, каждая мера которого отвечает за одну из данных последовательностей. Длина общей подпоследовательности для каждой ячейки списка определяется его предыдущими ячейками.

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.109375 sec	17.244140625 KB
Пример из задачи	0.109375 sec	17.244140625 KB
Верхняя граница диапазона значений входных данных из текста задачи	0.109375 sec	19.87890625 KB

Вывод по задаче: в задаче я реализовала алгоритм поиска длины наибольшей общей подпоследовательности для трёх последовательностей.

Вывод:

В работе я вспомнила принципы динамического программирования и их применение к задачам про общие подпоследовательности.