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 11

Выполнил:
Кузнецов А.Г.
К3140

Проверила:
Артамонова В.Е.

Санкт-Петербург
2023 г.

Содержание отчета

Задачи по варианту

Задача №5 Наибольшая общая подпоследовательность трех последовательностей

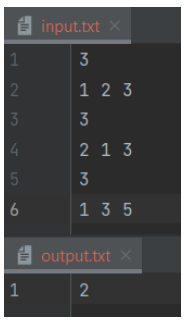
Вычислить длину самой длинной общей подпоследовательности из трех последовательностей.
Даны три последовательности A = (a1, a2, ..., an), B = (b1, b2, ..., bm) и C = (c1, c2, ..., cl), найти длину их самой длинной общей подпоследовательности, т.е. наибольшее неотрицательное целое число p такое, что существуют 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.

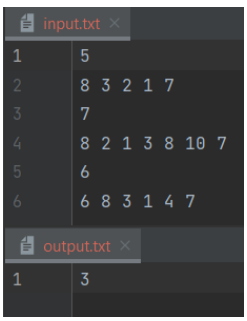
• Формат входного файла (input.txt).
– Первая строка: n - длина первой последовательности.
– Вторая строка: a1, a2, ..., an через пробел.
– Третья строка: m - длина второй последовательности.
– Четвертая строка: b1, b2, ..., bm через пробел.
– Пятая строка: l - длина второй последовательности.
– Шестая строка: c1, c2, ..., cl через пробел.
• Ограничения: $1 \le n, m, l \le 100$; $-109 < ai, bi, ci < 109$
• Формат вывода / выходного файла (output.txt). Выведите число p.
• Ограничение по времени. 1 сек.

```
def lcsOf3(A, B, C, n, m, l):
    array = [[[0 for _ in range(l + 1)] for _ in range(m + 1)] for _ in
range(n + 1)]
    for i in range(n + 1):
        for j in range(m + 1):
            for k in range(l + 1):
                if (i == 0 or j == 0 or k == 0):
                    array[i][j][k] = 0

                elif (A[i - 1] == B[j - 1] and
                      A[i - 1] == C[k - 1]):
                    array[i][j][k] = array[i - 1][j - 1][k - 1] + 1

                else:
                    array[i][j][k] = max(max(array[i - 1][j][k], array[i][j
- 1][k]),array[i][j][k - 1])

    return array[n][m][l]


with open('input.txt', 'r') as f:
    n = int(f.readline())
```

```
    A = [int(x) for x in f.readline().split(' ')]
    m = int(f.readline())
    B = [int(x) for x in f.readline().split(' ')]
    l = int(f.readline())
    C = [int(x) for x in f.readline().split(' ')]

with open('output.txt', 'w') as f:
    f.write(str(lcsOf3(A, B, C, n, m, l)))
```

Из файла input.txt мы получаем длины и сами последовательности. После чего открываем output.txt и записываем туда результат работы функции lcsof3. Функция создаёт трёхмерный массив. Для каждого элемента функция проверяет, является ли текущий элемент первым элементом какой-либо из последовательностей, если да, то она устанавливает значение соответствующей ячейки в массиве array равным 0. Если текущий элемент последовательности A равен текущему элементу последовательности B и текущему элементу последовательности C, функция добавляет 1 к длине LCS, сохраненной в предыдущей ячейке массива array. В противном случае он принимает максимальную длину LCS, сохраненную в ячейке выше, слева и по диагонали к текущей ячейке, и сохраняет ее в текущей ячейке. После обработки всех элементов функция возвращает значение, сохраненное в последней ячейке массива, которое соответствует длине LCS трех последовательностей.





Вывод по задаче: в ходе выполнения пятой задачи было реализовано динамический код, который находит наибольшую общую подпоследовательность трех последовательностей с помощью трёхмерного массива.

Задача №6 Наибольшая возрастающая подпоследовательность

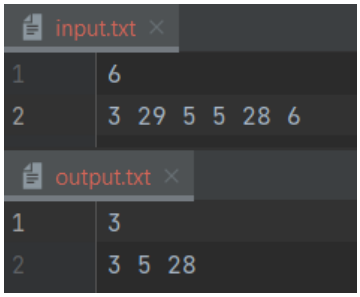
Дана последовательность, требуется найти ее наибольшую возрастающую подпоследовательность.

• Формат ввода / входного файла (input.txt). В первой строке входных данных задано целое число n – длина последовательности ($1 \le n \le 300000$). Во второй строке задается сама последовательность. Числа разделяются пробелом.
Элементы последовательности – целые числа, не превосходящие по модулю 109.
– Подзадача 1 (полегче). $n \le 5000$.
– Общая подзадача. $n \le 300000$.
• Формат вывода / выходного файла (output.txt). В первой строке выведите длину наибольшей возрастающей подпоследовательности, а во второй строке выведите через пробел саму наибольшую возрастающую подпоследовательность данной последовательности. Если ответов несколько – выведите любой.
• Ограничение по времени. 2 сек.
• Ограничение по памяти. 256 мб.

```
with open("input.txt", "r") as f:
    n = int(f.readline())
    A = [int(x) for x in f.readline().split()]

dp = [1] * n
lis = [[x] for x in A]

for i in range(1, n):
    for j in range(i):
        if A[i] > A[j] and dp[i] < dp[j] + 1:
            dp[i] = dp[j] + 1
            lis[i] = lis[j] + [A[i]]

max_length = max(dp)
index = dp.index(max_length)

with open("output.txt", "w") as output_file:
    output_file.write(str(max_length) + "\n")
    output_file.write(" ".join(map(str, lis[index])))
```

На вход получаем из файла input.txt длину последовательности и саму последовательность. Создаём массив dp, который будет хранить в себе

длины массима lis и сам массив lis, в котором будут хранится подпоследовательности. Если текущий элемент последовательности больше, чем предыдущий элемент последовательности. Это условие проверяет порядок возрастания подпоследовательности. Если длина lis, заканчивающаяся на текущем элементе, меньше длины lis, заканчивающейся на предыдущем элементе плюс 1. Это условие проверяет, длиннее ли текущий lis, чем предыдущий lis. Если оба условия выполняются, он обновляет массив dp с индексом текущего элемента до длины lis предыдущего элемента плюс 1 и обновляет массив lis с индексом текущего элемента до lis предыдущего элемента плюс текущий элемент. Находим максимальную длину и по индексу максимальной длины находим наибольшую подпоследовательность, после чего записываем их в output.txt



Вывод по задаче: в ходе выполнения шестой задачи был реализован алгоритм нахождения наибольшей возрастающей подпоследовательности.

Вывод

В ходе лабораторной работы была проведена работа по использованию динамического программирования в решении задач по нахождению наибольшей общей подпоследовательности трёх последовательностей и нахождению наибольшей возрастающей подпоследовательности.