Алгоритмы и структуры данных

Задание к лабораторной работе №7 (последняя в осеннем семестре). Динамическое программирование №1

В данной лабораторной работе изучается мощный инструмент, динамическое программирование. Аналогично предыдущим лабораторным работам, есть **два** способа ее выполнения и защиты, однако присутствуют изменения:

1 **Базовый уровень.** Решается 2 задачи по вариантам. Варианты в табличке внизу, номер вашего варианта соответствует вашему номеру в списке группы. **Посмотреть свой номер нужно, например, в журнале успеваемости по дисциплине**. В этом случае максимум за защиту можно получить 4 балла. В сумме с самой работой (0,5 балла) и отчетом (1 балл) получается 5,5 балла, что достаточно для зачета.

2 **Продвинутый уровень.** Решаются 4 задачb или больше, причем принципы, описанные выше для базового случая тоже учитываются. В этом случае вы сможете получить максимальные 7,5 баллов.

Вариант	Номера задач	Вариант	Номера задач
1	4,5	16	1,2
2	4,6	17	1,3
3	4,7	18	1,4
4	5,6	19	1,5
5	5,7	20	1,6
6	6,7	21	1,7
7	1,7	22	2,3
8	2,7	23	2,4
9	6,7	24	2,5
10	3,5	25	2,6
11	5,6	26	2,7
12	1,5	27	3,4
13	1,3	28	3,5
14	4,5	29	3,6
15	4,6	30	3,7

1 задача. Обмен монет

Как мы уже поняли из лекции, не всегда "жадное"решение задачи на обмен монет работает корректно для разных наборов номиналов монет. Например, если доступны номиналы 1, 3 и 4, жадный алгоритм поменяет 6 центов, используя три монеты (4 + 1 + 1), в то время как его можно изменить, используя всего две монеты (3 + 3). Теперь ваша цель - применить динамическое программирование для решения задачи про обмен монет для разных номиналов.

- **Формат ввода / входного файла (input.txt).** Целое число $money$ ($1 \leq money \leq 10^3$). Набор монет: количество возможных монет k и сам набор $coins = \{coin_1, ..., coin_k\}$. $1 \leq k \leq 100$, $1 \leq coin_i \leq 10^3$. Проверку можно сделать на наборе $\{1, 3, 4\}$. Формат ввода: первая строка содержит через пробел $money$ и k; вторая - $coin_1 coin_2 ... coin_k$.

  - ***Вариация 2:*** Количество монет в кассе ограничено. Для каждой монеты из набора $coins = \{coin_1, ..., coin_k\}$ есть соответствующее целое число - количество монет в кассе данного номинала $c = \{c_1, ..., c_k\}$. Если они закончились, то выдать данную монету невозможно.

- **Формат вывода / выходного файла (output.txt).** Вывести одно число – минимальное количество необходимых монет для размена $money$ доступным набором монет $coins$.

- Ограничение по времени. 1 сек.

- Примеры:

input.txt	output.txt
2 3 1 3 4	2

input.txt	output.txt
34 3 1 3 4	9

2 задача. Примитивный калькулятор

Дан примитивный калькулятор, который может выполнять следующие три операции с текущим числом x: умножить x на 2, умножить x на 3 или прибавить 1 к x. Дано положительное целое число n, найдите минимальное количество операций, необходимых для получения числа n, начиная с числа 1.

- **Формат ввода / входного файла (input.txt).** Дано одно целое число n, $1 \leq n \leq 10^6$. Посчитать минимальное количество операций, необходимых для получения n из числа 1.

- **Формат вывода / выходного файла (output.txt).** В первой строке вывести минимальное число k операций. Во второй – последовательность промежуточных чисел $a_0, a_1, ..., a_{k-1}$ таких, что $a_0 = 1, a_{k-1} = n$ и для всех $0 \leq i < k$ – a_{i+1} равно или $a_i + 1$, $2 \cdot a_i$, или $3 \cdot a_i$. Если есть несколько вариантов, выведите любой из них.

- Ограничение по времени. 1 сек.
- Примеры:

input.txt	output.txt
1	0 1

input.txt	output.txt
5	3 1 2 4 5

input.txt	output.txt
96234	14 1 3 9 10 11 22 66 198 594 1782 5346 16038 16039 32078 96234

- Во втором примере сначала идет умножение на 2 единички два раза, потом прибавление 1. Другой вариант - сначала умножить на 3, а потом добавить 1 два раза. То есть, вариант «1 3 4 5» - тоже верный.
- Аналогично, в третьм примере верным ответом также будет «1 3 9 10 11 33 99 297 891 2673 8019 16038 16039 48117 96234».
- Проход от 1 к n аналогичен проходу от n к 1, каждый раз текущее число либо делится на 2 или 3, либо из него вычитается 1. Поскольку мы хотели бы перейти от n к 1 как можно быстрее, будет естественно многократно уменьшать n, насколько это возможно. То есть на каждом шаге мы заменяем n на $\min\{n/3, n/2, n-1\}$ (деления на 3 и на 2 используются только тогда, когда n делится на них соответственно). Будем так делать пока не достигнем 1. Этот метод приводит к следующему алгоритму:

```
def optimal_sequence(n):
        sequence = []
        while n >= 1:
                sequence.append(n)
                if n % 3 == 0:
                        n = n // 3
                elif n % 2 == 0:
                        n = n // 2
                else:
                        n = n - 1
        return reversed(sequence)
```

Этот, казалось бы, правильный алгоритм на самом деле ***неверен***, в этом случае переход от n к $\min\{n/3, n/2, n-1\}$ небезопасен, можете его проверить.

3 задача. Редакционное расстояние

Редакционное расстояние между двумя строками – это минимальное количество 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 естественном 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 редактирования между заданными двумя строками.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt
ab ab	0

input.txt	output.txt
short ports	3

input.txt	output.txt
editing distance	5

★ Редакционное расстояние во втором примере равна 3:

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

- Дополнительное задание. Выведите в ответ также построчно, какие действия необходимо сделать, чтобы из 1 строки получилась вторая, порядок записи действий - произвольный. Например:

input.txt	output.txt
short ports	3 del s change h p add s

Как вариант, можно вместо действий вывести табличку как в пункте со звездочкой ★.

4 задача. Наибольшая общая подпоследовательность двух последовательностей

Вычислить длину самой длинной общей подпоследовательности из двух последовательностей.

Даны две последовательности $A = (a_1, a_2, ..., a_n)$ и $B = (b_1, b_2, ..., b_m)$, найти длину их самой длинной общей подпоследовательности, т.е. наибольшее неотрицательное целое число p такое, что существуют индексы $1 \leq i_1 < i_2 < ... < i_p \leq n$ и $1 \leq j_1 < j_2 < ... < j_p \leq m$ такие, что $a_{i_1} = b_{j_1}, ..., a_{i_p} = b_{j_p}$.

- **Формат ввода / входного файла (input.txt).**
  - Первая строка: n - длина первой последовательности.
  - Вторая строка: $a_1, a_2, ..., a_n$ через пробел.
  - Третья строка: m - длина второй последовательности.

- Четвертая строка: $b_1, b_2, ..., b_m$ через пробел.

- Ограничения: $1 \leq n, m \leq 100$; $-10^9 < a_i, b_i < 10^9$.

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

- Ограничение по времени. 1 сек.

- Примеры:

input.txt	output.txt
3 2 7 5 2 2 5	2

input.txt	output.txt
1 7 4 1 2 3 4	0

input.txt	output.txt
4 2 7 8 3 4 5 2 8 7	2

- В первом примере одна общая подпоследовательность – $(2, 5)$ длиной 2, во втором примере две последовательности не имеют одинаковых элементов. В третьем примере - длина 2, последовательности – $(2, 7)$ или $(2, 8)$.

5 задача. Наибольшая общая подпоследовательность трех последовательностей

Вычислить длину самой длинной общей подпоследовательности из *трех* последовательностей.

Даны три последовательности $A = (a_1, a_2, ..., a_n)$, $B = (b_1, b_2, ..., b_m)$ и $C = (c_1, c_2, ..., c_l)$, найти длину их самой длинной общей подпоследовательности, т.е. наибольшее неотрицательное целое число p такое, что существуют индексы $1 \leq i_1 < i_2 < ... < i_p \leq n$, $1 \leq j_1 < j_2 < ... < j_p \leq m$ и $1 \leq k_1 < k_2 < ... < k_p \leq l$ такие, что $a_{i_1} = b_{j_1} = c_{k_1}, ..., a_{i_p} = b_{j_p} = c_{k_p}$.

- **Формат ввода / входного файла (input.txt).**

  - Первая строка: n - длина первой последовательности.
  - Вторая строка: $a_1, a_2, ..., a_n$ через пробел.
  - Третья строка: m - длина второй последовательности.
  - Четвертая строка: $b_1, b_2, ..., b_m$ через пробел.
  - Пятая строка: l - длина второй последовательности.
  - Шестая строка: $c_1, c_2, ..., c_l$ через пробел.

- Ограничения: $1 \leq n, m, l \leq 100$; $-10^9 < a_i, b_i, c_i < 10^9$.

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

- Ограничение по времени. 1 сек.

- Примеры:

input.txt	output.txt
3 1 2 3 3 2 1 3 3 1 3 5	2

input.txt	output.txt
5 8 3 2 1 7 7 8 2 1 3 8 10 7 6 6 8 3 1 4 7	3

- В первом примере одна общая подпоследовательность – $(1, 3)$ длиной 2. Во втором примере есть две общие последовательности длиной 3 элемента – $(8, 3, 7)$ и $(8, 1, 7)$.

6 задача. Наибольшая возрастающая подпоследовательность

Дана последовательность, требуется найти ее наибольшую возрастающаю подпоследовательность.

- **Формат ввода / входного файла (input.txt).** В первой строке входных данных задано целое число n – длина последовательности ($1 \leq n \leq 300000$).

  Во второй строке задается сама последовательность. Числа разделяются пробелом.

  Элементы последовательности – целые числа, не превосходящие по модулю 10^9.

  - Подзадача 1 (полегче). $n \leq 5000$.
  - Общая подзадача. $n \leq 300000$.

- **Формат вывода / выходного файла (output.txt).** В первой строке выведите длину наибольшей возрастающей подпоследовательности, а во второй строке выведите через пробел саму наибольшую возрастающаю подпоследовательность данной последовательности. Если ответов несколько - выведите любой.

- Ограничение по времени. 2 сек.

- Ограничение по памяти. 256 мб.

- Пример:

input.txt	output.txt
6 3 29 5 5 28 6	3 3 5 28

7 задача. Шаблоны

Многие операционные системы используют шаблоны для ссылки на группы объектов: файлов, пользователей, и т. д. Ваша задача – реализовать простейший алгоритм проверки шаблонов для имен файлов.

В этой задаче алфавит состоит из маленьких букв английского алфавита и точки («.»). Шаблоны могут содержать произвольные символы алфавита, а также два специальных символа: «?» и «*». Знак вопроса («?») соответствует ровно одному произвольному символу. Звездочка «+» соответствует подстроке произвольной длины (возможно, нулевой). Символы алфавита, встречающиеся в шаблоне, отображаются на ровно один такой же символ в проверяемой строчке. Строка считается подходящей под шаблон, если символы шаблона можно последовательно отобразить на символы строки таким образом, как описано выше. Например, строчки «ab», «aab» и «beda.» подходят под шаблон «*a?», а строчки «bebe», «a» и «ba» –нет.

- **Формат ввода / входного файла (input.txt).** Первая строка входного файла определяет шаблон . Вторая строка S состоит только из символов алфавита. Ее необходимо проверить на соответствие шаблону. Длины обеих строк не превосходят 10 000. Строки могут быть пустыми – будьте внимательны!
- **Формат вывода / выходного файла (output.txt).** Если данная строка подходит под шаблон, выведите YES. Иначе выведите NO.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
k?t*n kitten	YES

input.txt	output.txt
k?t?n kitten	NO