Задание к лабораторной работе №4 (семестр 2). Подстроки

В данной лабораторной работе изучаются строки, поиск подстрок, алгоритмы Рабина-Карпа, Кнута-Морриса-Пратта, префис-функция, Z-функция. Аналогично предыдущим лабораторным работам, есть **два** способа ее выполнения и защиты, однако присутствуют изменения:

1 **Базовый уровень.** ***Решается 3 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**. Вариант в течении семестра менять нальзя!

   Условия базового уровня.

   – **Всего баллов.** За базовый уровень максимум за защиту и отчет **в сумме** можно получить 9 баллов, если сдаете в срок (1 месяц, до **22 июня 2022** года включительно).

   – **Замена.** ***Одну*** задачу можно заменить без потерь **внутри** уровня сложности, или на более сложную. На уровень ниже - нельзя заменить.

   – **Сложность.** У каждой задачи подписано количество баллов, которые можно за нее получить.

   .

   – **Отчет.** На защиту ***без отчета***, оформленного по правилам и загруженного в гитхаб, можно ***не приходить***!

   – Правила оформления отчета:

     * Формат файла - pdf;
     * Титульный лист с названием учреждения и факультета, в котором вы учитесь, курса, названия работы, а также ваши ФИО, группа; ФИО преподавателя; город, дата выполнения;
     * Описание задания к задаче;
     * Описание вашего решения и исходный код;
     * Описание проведенных тестов;
     * Выводы по каждой задаче и в целом по проделанной работе.

   – Хорошо оформленный отчет может добавить до 1 балла к защите, на усмотрение преподавателя. Т.е. можно получить 10 баллов в итоге.

   – За плохо оформленный отчет может сниматься до 2 баллов, также по усмотрению преподавателя. Т.е. можно получить только 7 баллов даже с учетом правильности всех задач.

   – **Дедлайн все-таки будет**. Если вы сдаете лабораторную работу *позже* срока (начиная с 22.06.2022), то суммарная базовая стоимость понижается на 2 балла, т.е. до 7 баллов. Эти 2 балла можно компенсировать решением дополнительных задач. После дедлайна будет еще пара консультаций, на которых еще можно будет прийти и досдать. ☺

2 **Продвинутый уровень.** Решаются ваши задачи по вариантам, причем принципы, описанные выше для базового случая тоже учитываются. До максимальных 15 баллов вы можете набрать, решая другие задачи из лабораторной или, например, из ***дополнительных***.

P.s. На дополнительных задачах нет баллов, но некоторые задачи более сложные чем другие. В целом можно за 2-3 дополнительно решенные задачи получить +6 баллов. Так же можно получить больше баллов в счет экзамена.

Варианты

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

Содержание

1 Задача. Наивный поиск подстроки в строке [2 s, 256 Mb, 1 балл]

Даны строки p и t. Требуется найти все вхождения строки p в строку t в качестве подстроки.

- **Формат ввода / входного файла (input.txt).** Первая строка входного файла содержит p, вторая – t. Строки состоят из букв латинского алфавита.
- **Ограничения на входные данные.** $1 \leq |p|, |t| \leq 10^4$.
- **Формат вывода / выходного файла (output.txt).** В первой строке выведите число вхождений строки p в строку t. Во второй строке выведите в возрастающем порядке номера символов строки t, с которых начинаются вхождения p. Символы нумеруются с единицы.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
aba abaCaba	2 1 5

- Проверяем **обязательно** – на OpenEdu, курс Алгоритмы программирования и структуры данных, неделя 9, задача 1.

2 Задача. Карта [2 s, 256 Mb, 1 балл]

В далеком 1744 году во время долгого плавания в руки капитана Александра Смоллетта попала древняя карта с указанием местонахождения сокровищ. Однако расшифровать ее содержание было не так уж и просто.

Команда Александра Смоллетта догадалась, что сокровища находятся на x шагов восточнее красного креста, однако определить значение числа она не смогла. По возвращению на материк Александр Смоллетт решил обратиться за помощью в расшифровке послания к знакомому мудрецу. Мудрец поведал, что данное послание таит за собой некоторое число. Для вычисления этого числа необходимо было удалить все пробелы между словами, а потом посчитать количество способов вычеркнуть все буквы кроме трех так, чтобы полученное слово из трех букв одинаково читалось слева направо и справа налево.

Александр Смоллетт догадывался, что число, зашифрованное в послании, и есть число x. Однако, вычислить это число у него не получилось.

После смерти капитана карта была безнадежно утеряна до тех пор, пока не оказалась в ваших руках. Вы уже знаете все секреты, осталось только вычислить число x.

- **Формат ввода / входного файла (input.txt).** В единственной строке входного файла дано послание, написанное на карте.
- **Ограничения на входные данные.** Длина послания не превышает $3 \cdot 10^5$. Гарантируется, что послание может содержать только строчные буквы английского алфавита и пробелы. Также гарантируется, что послание не пусто. Послание не может начинаться с пробела или заканчиваться им.
- **Формат вывода / выходного файла (output.txt).** Выведите одно число x – число способов вычеркнуть из послания все буквы кроме трех так, чтобы оставшееся слово одинаково читалось слева направо и справа налево.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt	input.txt	output.txt
treasure	8	you will never find the treasure	146

- Проверяем **обязательно** – на OpenEdu, курс Алгоритмы программирования и структуры данных, неделя 9, задача 2.

3 Задача. Паттерн в тексте [2 s, 256 Mb, 1 балл]

В этой задаче ваша цель – реализовать алгоритм Рабина-Карпа для поиска заданного шаблона (паттерна) в заданном тексте.

- **Формат ввода / входного файла (input.txt).** На входе две строки: паттерн P и текст T. Требуется найти все вхождения строки P в строку T в качестве подстроки.
- **Ограничения на входные данные.** $1 \leq |P|, |T| \leq 10^6$. Паттерн и текст содержат только латинские буквы.
- **Формат вывода / выходного файла (output.txt).** В первой строке выведите число вхождений строки P в строку T. Во второй строке выведите в возрастающем порядке номера символов строки T, с которых начинаются вхождения P. Символы нумеруются с единицы.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input	output
aba abacaba	2 1 5

input	output
Test testTesttesT	1 5

input	output
aaaaa baaaaaaa	3 2 3 4

- В первом примере паттерн *aba* можно найти в позициях 1 (**aba**caba) и 5 (abac**aba**) текста *abacaba*.

  Паттерн и текст в этой задаче чувствительны к регистру. Поэтому во втором примере паттерн *Test* встречается только в 45 позиции в тексте *testTesttesT*.

  Обратите внимание, что вхождения шаблона в тексте могут перекрываться, и это нормально, вам все равно нужно вывести их все.
- Используйте оператор == в Python вместо реализации собственной функции AreEqual для строк, потому что встроенный оператор == будет работать намного быстрее.
- Проверяем **обязательно** – на OpenEdu, курс Алгоритмы программирования и структуры данных, неделя 9, наблюдаемая задача.

4 Задача. Равенство подстрок [10 s, 512 Mb, 1.5 балла]

В этой задаче вы будете использовать хеширование для разработки алгоритма, способного предварительно обработать заданную строку s, чтобы ответить эффективно на любой запрос типа «равны ли эти две подстроки s?» Это, в свою очередь, является основной частью во многих алгоритмах обработки строк.

- **Формат ввода / входного файла (input.txt).** Первая строка содержит строку s, состоящую из строчных латинских букв. Вторая строка содержит количество запросов q. Каждая из следующих q строк задает запрос тремя целыми числами a, b и l.
- **Ограничения на входные данные.** $1 \le |s| \le 500000, 1 \le q \le 100000, 0 \le a, b \le |s| - l$ (следовательно, индексы a и b начинаются с 0).
- **Формат вывода / выходного файла (output.txt).** Для каждого запроса выведите «Yes», если подстроки $s_a s_{a+1} ... s_{a+l-1} = s_b s_{b+1} ... s_{b+l-1}$ равны, и «No» – если не равны.
- Ограничение по времени. 10 сек.
- Ограничение по памяти. 512 мб.
- Пример:

input	output
trololo 4 0 0 7 2 4 3 3 5 1 1 3 2	Yes Yes Yes No

- Что в примере?
  - 0 0 7 → trololo = trololo
  - 2 4 3 → trololo = trololo
  - 3 5 1 → trololo = trololo
  - 1 3 2 → trololo ≠ trololo
- Что делать?

  Для строки $t = t_0 t_1 ... t_{m-1}$ длиной m и для целого числа x определим полиномиальную хеш-функцию:

$$H(t) = \sum_{j=0}^{m-1} t_j x^{m-j-1} = t_0 x^{m-1} + t_1 x^{m-2} + ... + t_{m-2} x + t_{m-1}.$$

  Пусть $s_a s_{a+1} ... s_{a+l-1}$ – это подстрока заданной строки $s = s_0 s_1 ... s_{n-1}$. Замечательным свойством полиномиальной хеш-функции H является то, что $H(s_a s_{a+1} ... s_{a+l-1})$ можно выразить через $H(s_0 s_1 ... s_{a+l-1})$ и $H(s_0 s_1 ... s_{a-1})$, т. е. через хеш-значения двух префиксов s:

$$\begin{aligned} H(s_a s_{a+1} ... s_{a+l-1}) &= s_a x^{l-1} + s_{a+1} x^{l-2} + ... + s_{a+l-1} = \\ &= s_0 x^{a+l-1} + s_1 x^{a+l-2} + ... + s_{a+l-1} - \\ &- x^l (s_0 x^{a-1} + s_1 x^{a-2} + ... + s_{a-1}) = \\ &= H(s_0 s_1 ... s_{a+l-1}) - x^l H(s_0 s_1 ... s_{a-1}). \end{aligned}$$

  Это приводит нас к следующей идее: мы предварительно вычисляем и сохраняем хэш-значения всех префиксов s: пусть $h[0] = 0$ и, для $1 \le i \le n$, пусть $h[i] = H(s_0 s_1 ... s_{i-1})$. Тогда тождество, приведенное выше, становится

$$H(s_a s_{a+1} ... s_{a+l-1}) = h[a+l] - x^l h[a].$$

Другими словами, мы можем получить хеш-значение любой подстроки s всего за константное время! Ясно, что если $H(s_a s_{a+1} ... s_{a+l-1}) \neq H(s_b s_{b+1} ... s_{b+l-1})$, то соответствующие две подстроки ($s_a s_{a+1} ... s_{a+l-1}$ и $s_b s_{b+1} ... s_{b+l-1}$) различны. Однако, если хеш-значения совпадают, все же возможно, что подстроки различаются – это называется *коллизией*. Ниже мы обсудим, как уменьшить вероятность коллизии.

Напомним, что на практике никогда не вычисляется точное значение полиномиальной хеш-функции: все вычисляется по модулю m для некоторого фиксированного целого числа m. Это делается для того, чтобы все вычисления были эффективными, а хэш-значения были достаточно малы. Напомним также, что при вычислении $H(s) \mod m$ важно делать каждый промежуточный шаг (а не окончательный результат) по модулю m.

Можно показать, что если s_1 и s_2 – две *разные* строки длины n, а m – простое целое число, то вероятность того, что $H(s_1) \mod m = H(s_2) \mod m$ (при выборе $0 \leq x \leq m-1$) не более чем $\frac{n}{m}$ (примерно, это потому, что $H(s_1) - H(s_2)$ является ненулевым полиномом степени не выше $n-1$ и, следовательно, может иметь не более n корней по модулю m). Чтобы еще больше снизить вероятность столкновения, можно взять два разных модуля.

В целом получается следующий подход.

1. Установите $m_1 = 10^9 + 7$ и $m_2 = 10^9 + 9$.
2. Выберите случайный x от 1 до 10^9.
3. Посчитайте хеш-таблицы $h_1[0..n]$ и $h_2[0..n]$: $h_1[0] = h_2[0] = 0$ и для всех $1 \leq i \leq n$: $h_1[i] = H(s_0 ... s_{i-1}) \mod m_1$ и $h_2[i] = H(s_0 ... s_{i-1}) \mod m_2$. Проиллюстрируем это для h_1. Используя $h_1[0] = 0$ в цикле от 1 до n будем считать:
$$h_1[i] \leftarrow (x \cdot h_1[i-1] + s_i) \mod m_1$$
4. Для каждого запроса (a, b, l):
   (a) Используйте предварительно вычисленные хеш-значения, чтобы вычислить хеш-значения подстрок $s_a s_{a+1} ... s_{a+l-1}$ и $s_b s_{b+1} ... s_{b+l-1}$ по модулю m_1 и m_2.
   (b) Выведите «Yes», если
$$H(s_a s_{a+1} ... s_{a+l-1}) \mod m_1 = H(s_b s_{b+1} ... s_{b+l-1}) \mod m_1, \text{ и}$$
$$H(s_a s_{a+1} ... s_{a+l-1}) \mod m_2 = H(s_b s_{b+1} ... s_{b+l-1}) \mod m_2$$
   (c) В противном случае выведите «No»

Обратите внимание, что, в отличие от алгоритма Карпа–Рабина, мы не сравниваем подстроки наивно, когда их хеши совпадают. Вероятность этого события не превышает $\frac{n}{m_1} \cdot \frac{n}{m_2} \leq 10^{-9}$. (На самом деле для случайных строк вероятность даже намного меньше: 10^{-18}.)

5 Задача. Префикс-функция [2 s, 256 Mb, 1.5 балла]

Постройте префикс-функцию для всех непустых префиксов заданной строки s.

- **Формат ввода / входного файла (input.txt).** Одна строка входного файла содержит s. Строка состоит из букв латинского алфавита.
- **Ограничения на входные данные.** $1 \leq |s| \leq 10^6$.
- **Формат вывода / выходного файла (output.txt).** Выведите значения префикс-функции для всех префиксов строки s длиной $1, 2, ..., |s|$, в указанном порядке.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt	input.txt	output.txt
aaaAAA	0 1 2 0 0 0	abacaba	0 0 1 0 1 2 3

- Проверяем **обязательно** – на OpenEdu, курс Алгоритмы программирования и структуры данных, неделя 10, задача 1.

6 Задача. Z-функция [2 s, 256 Mb, 1.5 балла]

Постройте Z-функцию для заданной строки s.

- **Формат ввода / входного файла (input.txt).** Одна строка входного файла содержит s. Строка состоит из букв латинского алфавита.
- **Ограничения на входные данные.** $2 \leq |s| \leq 10^6$.
- **Формат вывода / выходного файла (output.txt).** Выведите значения Z-функции для всех индексов $1, 2, ..., |s|$ строки s, в указанном порядке.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt
aaaAAA	2 1 0 0 0

input.txt	output.txt
abacaba	0 1 0 3 0 1

- Проверяем **обязательно** – на OpenEdu, курс Алгоритмы программирования и структуры данных, неделя 10, задача 2.

7 Задача. Наибольшая общая подстрока [15 s, 512 Mb, 2 балла]

В задаче на наибольшую общую подстроку даются две строки s и t, и цель состоит в том, чтобы найти строку w максимальной длины, которая является подстрокой как s, так и t. Это естественная мера сходства между двумя строками. Задача имеет применения для сравнения и сжатия текстов, а также в биоинформатике. Эту проблему можно рассматривать как частный случай проблемы расстояния редактирования (Левенштейна), где разрешены только вставки и удаления. Следовательно, ее можно решить за время $O(|s||t|)$ с помощью динамического программирования. Есть также весьма нетривиальные структуры данных для решения этой задачи за линейное время $O(|s| + |t|)$. В этой задаче ваша цель – использовать хеширование для решения почти за линейное время.

- **Формат ввода / входного файла (input.txt).** Каждая строка входных данных содержит две строки s и t, состоящие из строчных латинских букв.
- **Ограничения на входные данные.** Суммарная длина всех s, а также суммарная длина всех s не превышает 100 000.
- **Формат вывода / выходного файла (output.txt).** Для каждой пары строк s_i и t_i найдите ее самую длинную общую подстроку и уточните ее параметры, выведя три целых числа: ее начальную позицию в s, ее начальную позицию в t (обе считаются с 0) и ее длину. Формально выведите целые числа $0 \le i < |s|$, $0 \le j < |t|$ и $l \ge 0$ такие, что и l максимально. (Как обычно, если таких троек с максимальным l много, выведите любую из них.)
- Ограничение по времени. 15 сек.
- Ограничение по памяти. 512 мб.
- Пример:

input	output
cool toolbox	1 1 3
aaa bb	0 1 0
aabaa babbaab	0 4 3

- Объяснение:

  Самая длинная общая подстрока первой пары строк – *ool*, она начинается с первой позиции в *toolbox* и с первой позиции в *cool*. Строки из второй строки не имеют общих непустых общих подстрок (в этом случае $l = 0$ и можно вывести любые индексы i и j). Наконец, последние две строки имеют общую подстроку *aab* длины 3, начинающуюся с позиции 0 в первой строке и с позиции 4 во второй. Обратите внимание, что для этой пары строк также можно вывести 2 3 3.

- Что делать?

  Для каждой пары строк s и t используйте двоичный поиск, чтобы найти длину наибольшей общей подстроки. Чтобы проверить, есть ли у двух строк общая подстрока длины k,

  - предварительно вычислить хеш-значения всех подстрок длины k из s и t;
  - обязательно используйте несколько хэш-функций (но не одну), чтобы уменьшить вероятность коллизии;
  - храните хеш-значения всех подстрок длины k строки s в хеш-таблице; затем пройдитесь по всем подстрокам длины k строки t и проверьте, присутствует ли хеш-значение этой подстроки в хеш-таблице.

8 Задача. Шаблоны с несовпадениями [40 s, 512 Mb, 2 балла]

Естественным обобщением задачи сопоставления паттернов, текстов является следующее: найти все места в тексте, расстояние (различие) от которых до образца достаточно мало. Эта проблема находит применение в текстовом поиске (где несовпадения соответствуют опечаткам) и биоинформатике (где несовпадения соответствуют мутациям).

В этой задаче нужно решить следующее. Для целочисленного параметра k и двух строк $t = t_0t_1...t_{m-1}$ и $p = p_0p_1...p_{n-1}$, мы говорим, что p встречается в t в знаке индекса i с не более чем k несовпадениями, если строки p и $t[i : i + p) = t_it_{i+1}...t_{i+n-1}$ различаются не более чем на k знаков.

- **Формат ввода / входного файла (input.txt).** Каждая строка входных данных содержит целое число k и две строки t и p, состоящие из строчных латинских букв.
- **Ограничения на входные данные.** $0 \le k \le 5$, $1 \le |t| \le 200000$, $1 \le |p| \le \min|t|, 100000$. Суммарная длина строчек t не превышает 200 000, общая длина всех p не превышает 100 000.
- **Формат вывода / выходного файла (output.txt).** Для каждой тройки (k, t, p) найдите все позиции $0 \le i_1 < i_2 < ... < i_l < |t|$ в которых строка p встречается в строке t с не более чем k несоответствиями. Выведите l и $i_1, i_2, ..., i_l$.
- Ограничение по времени. 40 сек. (Python), 2 сек (C++).
- Ограничение по памяти. 512 мб.
- Пример:

input	output
0 ababab baaa	0
1 ababab baaa	1 1
1 xabcabc ccc	0
2 xabcabc ccc	4 1 2 3 4
3 aaa xxx	1 0

- Объяснение:

  Для первой тройки точных совпадений нет. Для второй тройки baaa находится на расстоянии один от паттерна с началом в индексе 1 *ababab*. Для третьей тройки нет вхождений не более чем с одним несовпадением. Для четвертой тройки любая (длина три) подстрока p, содержащая хотя бы одну букву c, находится на расстоянии не более двух от t. Для пятой тройки t и p различаются тремя позициями.

- Начните с вычисления хеш-значений префиксов t и p и их частичных сумм. Это позволяет сравнивать любые две подстроки t и p за ожидаемое постоянное время. Для каждой позиции-кандидата i выполните k шагов вида «найти следующее несоответствие». Каждое такое несоответствие можно найти с помощью бинарного поиска.

9 Задача. Декомпозиция строки [2 s, 256 Mb, 2 балла]

Строка `ABCABCDEDEDEF` содержит подстроку `ABC`, повторяющуюся два раза подряд, и подстроку `DE`, повторяющуюся три раза подряд. Таким образом, ее можно записать как `ABC*2+DE*3+F`, что занимает меньше места, чем исходная запись той же строки.

Ваша задача – построить наиболее экономное представление данной строки s в виде, продемонстрированном выше, а именно, подобрать такие $s_1, a_1, ..., s_k, a_k$, где s_i - строки, а a_i - числа, чтобы $s = s_1 \cdot a_1 + ... + s_k \cdot a_k$. Под операцией умножения строки на целое положительное число подразумевается конкатенация одной или нескольких копий строки, число которых равно числовому множителю, то есть, `ABC*2=ABCABC`. При этом требуется минимизировать общую длину итогового описания, в котором компоненты разделяются знаком `+`, а умножение строки на число записывается как умножаемая строка и множитель, разделенные знаком `*`. Если же множитель равен единице, его, вместе со знаком `*`, допускается не указывать.

- **Формат ввода / входного файла (input.txt).** Одна строка входного файла содержит s. Строка состоит из букв латинского алфавита.
- **Ограничения на входные данные.** $1 \leq |s| \leq 5 \cdot 10^3$.
- **Формат вывода / выходного файла (output.txt).** Выведите оптимальное представление строки, данной во входном файле. Если оптимальных представлений несколько, выведите любое.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt	input.txt	output.txt
ABCABCDEDEDEF	ABC*2+DE*3+F	Hello	Hello

- Проверяем **обязательно** – на OpenEdu, курс Алгоритмы программирования и структуры данных, неделя 10, задача 3.

Дополнительные задачи

1. Последнее слово Джека
2. Басня о строке
3. Имена
4. Суффиксы
5. Поиск подстроки
6. Сдвиг текста
7. Циклическая строка
8. Подстроки из одинаковых букв
9. Преобразование ДНК
10. Abracadabra