

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Мобильные и сетевые технологии**

Направление подготовки **09.03.03 Прикладная информатика**

О Т Ч Е Т

по практической работе

на тему: лабораторная работа №4

Обучающийся Петров Андрей Сергеевич, № группы К3141

Работа выполнена с оценкой _____

Преподаватель: Харьковская А.Т.

(подпись)

(подпись)

Дата _____

Санкт-Петербург, 2022

Задание №2

Условие: 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 .

Решение: Создадим словарь, где ключами будут символы строки, а значениями - список из индексов, где находятся данные символы. Далее итерируемся по значениям словаря - спискам из индексов символов строки, находим x - сколько символов заключено между первым и последним символом в списке, и $variants$ - сколько пар индексов можно составить из индексов в списке. Перемножаем полученные x и $variants$ и получаем число всех возможных палиндромов из трёх букв, где крайними буквами являются ключ для текущего значения в словаре, если бы между всеми символами пар индексов было заключено x символов, и прибавляем к $count$ это число (в $count$ после всех операций будет лежать ответ). Далее итерируемся по всем индексам символов строки из списка кроме крайних. И из $count$ вычитаем количество всех палиндромов, которое мы посчитали лишний раз, это значение находится как сумма ((произведения номера текущей итерации и разности последнего индекса символа строки из списка и текущего индекса символа строки из списка) и (произведения разности текущего индекса символа строки из списка и первого индекса символа строки из списка и длина списка минус номер текущей итерации минус 1)). На каждой итерации по значениям словаря рассматривается список из индексов конкретного символа строки.

Тесты:

Выведите одно число: — число способов вычеркнуть из послания все буквы кроме трех так, чтобы оставшееся слово одинаково читалось слева направо и справа налево.

Примеры

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

Выбрать файлы | Файл не выбран

Верное решение!
Результаты работы Вашего решения

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.296	23191552	300002	16
1	OK	0.031	9003008	10	1
2	OK	0.015	9027584	34	3
3	OK	0.031	9023488	5	1
4	OK	0.031	9007104	6	1
5	OK	0.046	8970240	7	1
6	OK	0.031	9015296	9	2
7	OK	0.031	8998912	7	1
8	OK	0.015	8945664	7	1
9	OK	0.031	8978432	13	2
10	OK	0.000	9007104	202	6
11	OK	0.031	9048064	202	6

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Исходный код:

```
fint = open('input.txt')
fout = open('output.txt', 'w')
x = fint.readline().strip().split()
s = ''.join(x)

letters = {}
for i in range(len(s)):
    if s[i] in letters.keys():
        letters[s[i]].append(i)
    else:
        letters[s[i]] = [i]

count = 0
for i in letters.values():
    if len(i) != 1:
        x = i[-1] - i[0] - 1
        variants = len(i) * (len(i) - 1) // 2
        count += x * variants
        for j in range(1, len(i) - 1):
            count -= (i[-1] - i[j]) * j + (i[j] - i[0]) * (len(i) - j - 1)

fout.write(str(count))
```

Задание №6

Условие: Постройте Z-функцию для заданной строки s.

Решение: Для каждого символа строки, начиная со второго, нужно найти значение z-функции, которое равно максимальной длине подстроки начиная с данного символа, являющейся префиксом исходной строки. Чтобы на каждой итерации не искать по новой значение z-функции будем хранить индексы крайних символов для ранее найденной z-функции. Таким образом построение z-функции будет быстрее.

Тесты:

aaaAAA 2 1 0 0 0
abacaba 0 1 0 3 0 1

Выбрать файлы | Файл не выбран

Верное решение!
Результаты работы Вашего решения

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.546	140230656	1000002	6888887
1	OK	0.015	9007104	8	9
2	OK	0.031	9011200	9	11
3	OK	0.015	9003008	4	1
4	OK	0.031	9052160	4	1
5	OK	0.031	9007104	5	3
6	OK	0.015	9019392	12	17
7	OK	0.000	9035776	12	17
8	OK	0.109	18165760	92672	185337
9	OK	0.171	19509248	99998	264800
10	OK	0.171	19333120	100002	272210
11	OK	0.234	28020736	176391	352775
12	OK	0.296	29171712	199994	474049
13	OK	0.328	29151232	199992	456478
14	OK	0.187	27795456	172864	345721
15	OK	0.437	48443392	300002	1988887

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Исходный код:

```
fint = open('input.txt')
fout = open('output.txt', 'w')
x = fint.readline().strip().split()
s = ''.join(x)

z = [0]*len(s)
l = 0
r = 0
z[0] = 0

for i in range(1, len(s)):
    d = 0
    if i <= r:
        d = min(r-i+1, z[i-1])
    while i+d < len(s) and s[i+d] == s[d]:
        d += 1
    if i+d-1 > r:
        l = i
        r = i+d-1
    z[i] = d

x = [str(i) for i in z]
s = ' '.join(x[1:])
fout.write(s)
```

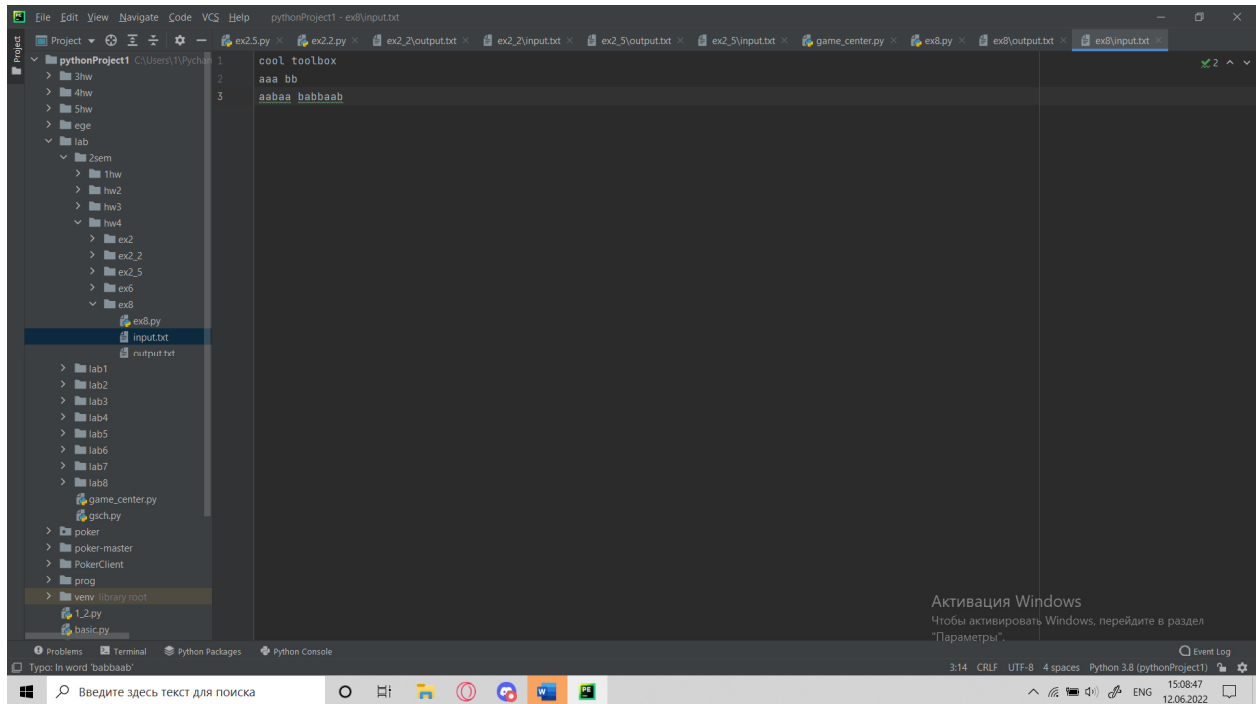
Задание №7

Условие: 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.

Решение: С помощью бинарного поиска будем искать наибольшую длину общей подстроки следующим образом, если после шага рекурсии была найдена общая подстрока заданной длины, то в следующем шаге нижней границей поиска становится длина найденной общей подстроки + 1, иначе изменяется верхняя граница поиска на длину найденной общей подстроки - 1. В каждом шаге рекурсии для каждой строки найдём значение хэш-функций для всех подстрок заданной длины, далее ищем пересечение множеств хэшей двух строк, по полученному пересечению проверяем равенство подстрок первой и второй строки, если они равны, в ответ записываем длину общей подстроки и индекса с которых она начинается в строках. Если после работы всей программы общая подстрока не найдена в ответ запишется строка 0 0 0.

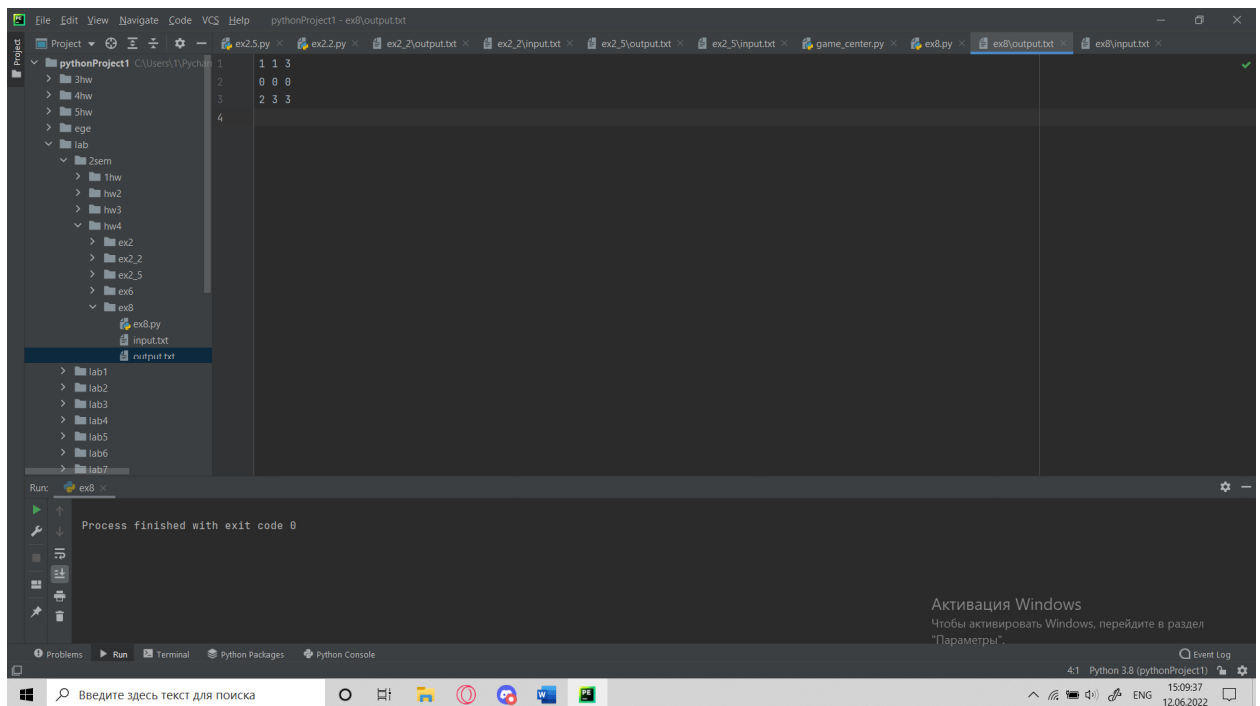
Тесты:



The screenshot shows the Visual Studio Code editor interface. The left sidebar displays a project tree for 'pythonProject1' with a folder structure including '3hw', '4hw', '5hw', 'ege', 'lab', '2sem', '1hw', 'hw2', 'hw3', 'hw4', 'ex2', 'ex2_2', 'ex2_5', 'ex6', 'ex8', 'ex8.py', 'input.txt', 'output.txt', 'lab1' through 'lab8', 'game_center.py', 'gach.py', 'poker', 'poker-master', 'PokerClient', 'prog', 'venv', 'library root', '1.2.py', and 'basic.py'. The main editor window shows a file named 'ex8\input.txt' with the following content:

```
cool toolbox
aaa bb
aabaa babbaab
```

The bottom status bar indicates the file encoding is UTF-8, 4 spaces, and Python 3.8 (pythonProject1). A Windows activation watermark is visible in the bottom right corner.



The screenshot shows the Visual Studio Code editor interface after running a script. The main editor window shows a file named 'ex8\output.txt' with the following content:

```
1 1 3
0 0 0
2 3 3
```

The bottom status bar indicates the file encoding is UTF-8, 4 spaces, and Python 3.8 (pythonProject1). A Windows activation watermark is visible in the bottom right corner. The Run and Debug panel at the bottom shows the message 'Process finished with exit code 0'.

Исходный код:

```
import random

fint = open('../ex8/input.txt', 'r')
fout = open('output.txt', 'w')
answers = []
answer = [0, 0, 0]
x = fint.read().strip().split('\n')

def poly_hash(s, p, x):
    y = 1
    h = 0
    for i in range(0, len(s)):
        h += ord(s[i])*y
        if i != len(s)-1:
            y = (y % p) * x
    return h % p

def precompute(s, k, p, x):
    h = {}
    string = s[-k:]
    hashes = [0]*(len(s)-k+1)
    key = poly_hash(string, p, x)
    y = 1
    for i in range(len(string)):
        y = (y*x) % p
    hashes[len(s)-k] = key
    h[key] = len(s)-k
    for i in range(len(s)-k-1, -1, -1):
        new_hash = (x*hashes[i+1] + ord(s[i]) - y*ord(s[i + k])) % p
        h[new_hash] = i
        hashes[i] = new_hash
    return h

def are_equal(s, t):
    if s == t:
        return True
    else:
        return False

def search_common_subs(s_hash, t_hash, t, s, k):
    common_subs = list(set(s_hash.keys() & set(t_hash.keys())))
    for i in common_subs:
        if are_equal(s[s_hash[i]:s_hash[i]+k], t[t_hash[i]:t_hash[i]+k]):
            return s_hash[i], t_hash[i]

def binary_search(s, t, first, last):
    if first > last:
        return -1
    global answer
    k = (first+last)//2
    p = 10**9+7
    x = random.randint(2, p-1)
    s_hash = precompute(s, k, p, x)
    t_hash = precompute(t, k, p, x)
    indexes = search_common_subs(s_hash, t_hash, t, s, k)
    if indexes:
```



```
        answer = [indexes[0], indexes[1], k]
        binary_search(s, t, k+1, last)
    else:
        binary_search(s, t, first, k-1)

for i in x:
    strings = i.split()
    binary_search(strings[0], strings[1], 1, min(len(strings[0]),
len(strings[1])))
    answers.append(answer)
    answer = [0, 0, 0]

for i in answers:
    fout.write(' '.join([str(j) for j in i])+'\n')
```

Задание №2.2

Условие: Однажды Три Программиста придумали занятную игру для тренировки памяти и умственных способностей. Первый Программист сочинил строку S из N символов и сообщил её Второму и Третьему Программистам. Второй Программист произвёл над этой строкой X ($0 \leq X < N$) последовательных циклических сдвигов (под циклическим сдвигом строки понимается перенос её последнего символа в начало). В результате этих манипуляций получилась строка T , которую он сообщил Третьему Программисту. Задачей Третьего Программиста было определить число X , либо сообщить Второму Программисту, что он ошибся, поскольку строка T не могла быть получена из строки S с помощью циклических сдвигов.

Решение: Найдем значение h хэш-функции строки, полученной после циклических сдвигов, а также множество значений хэш-функций строки после от 0 до $\text{len}(s)-1$ циклических сдвигов. Если h есть в данном множестве, то по данному хэшу найдём количество циклических сдвигов, иначе в ответ запишем -1.

Тесты:

The screenshot shows the Timus Online Judge interface. At the top, there are navigation links for Online Judge, Задачи, Авторы, and Соревнования. Below these, there are links for О системе, Часто задаваемые вопросы, Новости сайта, Форум, and Ссылки. The main section is titled "Последние попытки" (Recent attempts) and displays a table of attempts for problem 1423, "Басня о строке" (A fairy tale about a string).

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
9911592	17:04:24 12 июн 2022	Andrew	1423. Басня о строке	PyPy 3.0 x64	Accepted		0.734	7 060 КБ
9911384	06:17:03 12 июн 2022	Andrew	1423. Басня о строке	PyPy 3.8 x64	Accepted		0.671	7 912 КБ
9911383	06:16:34 12 июн 2022	Andrew	1423. Басня о строке	PyPy 3.8 x64	Accepted		0.718	8 192 КБ
9911382	06:15:49 12 июн 2022	Andrew	1423. Басня о строке	PyPy 3.8 x64	Runtime error	1	0.078	1 376 КБ
9911381	06:14:39 12 июн 2022	Andrew	1423. Басня о строке	PyPy 3.8 x64	Wrong answer	1	0.031	188 КБ
9911380	06:11:27 12 июн 2022	Andrew	1423. Басня о строке	PyPy 3.8 x64	Runtime error	1	0.078	1 340 КБ
9911379	06:10:06 12 июн 2022	Andrew	1423. Басня о строке	PyPy 3.8 x64	Runtime error	1	0.062	1 416 КБ

Показывать по 10 | 30 | 100 строк на странице • Обновлять каждые 15 | 60 | 240 секунд | не обновлять

Активация Windows
© 2000–2022 Timus Online Judge Team. Все права защищены.
"Параметры".

Исходный код:

```
import random

def poly_hash(s, p, x):
    y = 1
    h = 0
    for i in range(0, len(s)):
        h += ord(s[i])*y
        if i != len(s)-1:
            y = (y % p) * x
    return h % p

def check_match(s, k, p, x, t_h):
    hashes = [0]*len(s)
    s_h = poly_hash(s, p, x)
    y = 1
    for i in range(n):
        y = (y*x) % p
        hashes[0] = s_h

    if s_h == t_h:
        return 0

    for i in range(1, len(s)):
        s_h = (x*hashes[i-1] + ord(s[-i]) - y*ord(s[-i])) % p
        hashes[i] = s_h
        if s_h == t_h:
            return i

    return -1

n = int(input())
s = input()
t = input()

p = 10**9+7
x = random.randint(2, p)
h = poly_hash(t, p, x)

print(check_match(s, n, p, x, h))
```

Задание №2.5

Условие: Найти все вхождения строки T в строке S .

Решение: Найдем значение h хэш-функции для строки t . Далее каждую подстроку строки s длиной $\text{len}(t)$ хэшируем и сравниваем с h , если значение хэшей совпадает кладём в список индекс данной подстроки. Ответом будет список из индексов.

Тесты:

The screenshot shows the 'School of Programmer' website interface. The main content area displays a Python solution for finding all occurrences of string T in string S using a polynomial hash algorithm. The code defines functions for calculating the hash of a string, checking for matches, and returning the indices of all occurrences.

On the right side of the page, there is a table showing the test results for the solution:

Тест	Результат	Время	Память
1	Accepted	0,046	2174 КБ
2	Accepted	0,031	2170 КБ
3	Accepted	0,031	2170 КБ
4	Accepted	0,046	2170 КБ
5	Accepted	0,015	2170 КБ
6	Accepted	0,031	2170 КБ
7	Accepted	0,078	3198 КБ
8	Accepted	0,062	3198 КБ
9	Accepted	0,078	3202 КБ
10	Accepted	0,078	2170 КБ
11	Accepted	0,093	5,1 МБ
12	Accepted	0,125	2946 КБ
13	Accepted	0,031	2170 КБ

Исходный код:

```
import random

fint = open('input.txt', 'r')
fout = open('output.txt', 'w')
indexes = []

def poly_hash(s, p, x):
    y = 1
    h = 0
    for i in range(0, len(s)):
        h += ord(s[i])*y
        if i != len(s)-1:
            y = (y % p) * x
    return h % p

def check_match(s, k, p, x, t_h):
    string = s[-k:]
    hashes = [0]*(len(s)-k+1)
    s_h = poly_hash(string, p, x)
    y = 1
    for i in range(len(string)):
        y = (y*x) % p
        hashes[len(s)-k] = s_h

    if s_h == t_h:
        indexes.append(len(s)-k)

    for i in range(len(s)-k-1, -1, -1):
        s_h = (x*hashes[i+1] + ord(s[i]) - y*ord(s[i + k])) % p
        hashes[i] = s_h
        if s_h == t_h:
            indexes.append(i)

s = fint.readline().strip()
t = fint.readline().strip()

p = 10**9+7
x = random.randint(2, p)
h = poly_hash(t, p, x)
check_match(s, len(t), p, x, h)

indexes.reverse()
fout.write(' '.join([str(i) for i in indexes]))
```