САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №4

по курсу «Алгоритмы и структуры данных»

Тема: Подстроки

Вариант 22

Выполнил:

Федюкин М. В.

К3244

Проверила:

Артамонова В.Е.

Санкт-Петербург

2023 г.

Содержание отчета

Задачи по варианту

Задание к лабораторной работе № 2-4:
https://drive.google.com/drive/folders/1hjwL6oDXaZJ8BZqDXJec6fxwhDpgdbmX

Мой вариант – 22.

Обязательные задачи: 1, 4, 8
Можно заменить одну задачу на задачу такого же уровня сложности, поэтому меняю 8-ю на 7-ю. Так же решил 2-ю, 5-ю и 3 дополнительных.

Дополнительные задачи: Поиск подстроки, Сдвиг текста, Подстроки из одинаковых букв.

Обязательные задачи

Задача 1

Для каждого элемента исходной строки последовательно сравнивается кусок исходной строки с заданной подстрокой.

```
with open('input.txt', 'r') as infile:
    sub = infile.readline().strip()
    initstr = infile.readline().strip()
    found_ind = []
    for i in range(len(initstr) - len(sub) + 1):
        if initstr[i:i+len(sub)] == sub:
            found_ind.append(i+1)
with open('output.txt', 'w') as outfile:
    print(len(found_ind), file=outfile)
    print(*found_ind, file=outfile)
```

Input	Output
aba abaCaba	2 1 5
Mike mikeMikemiMifeMike	2 5 15
kek kelolkekekekekekek	6 6 8 10 12 14 16

Задача 2

Я создал словарь символов, куда для каждого символа записал позиции, на которых он встречается в исходной строке. Дальше для каждого получившегося списка нужно было найти сумму разностей пар.

```
with open('input.txt', 'r') as infile:
    initstr = infile.readline().strip().replace(" ", "")
    encoded = {}
    for i, s in enumerate(initstr):
        try:
            encoded[s].append(i)
        except Exception:
            encoded[s] = [i]
    total = 0
    for spisok in encoded.values():
        if len(spisok) > 1:
            for i, val in enumerate(spisok):
                total += val * (2*i - len(spisok) + 1)
            total -= len(spisok) * (len(spisok) - 1) // 2
with open('output.txt', 'w') as outfile:
    print(total, file=outfile)
```

Input	Output
treasure	8
you will never find the treasure	146
that is what you are Honey, you're my golden star I know you can make my wish come true If you let me treasure you If you let me treasure you	39

Задача 4

Алгоритм полностью основан на описанном в ТЗ к лабораторной работе.

```
from random import randint


with open('input.txt', 'r') as infile:
    with open('output.txt', 'w') as outfile:
        string = infile.readline().strip()
        n = int(infile.readline())
        m1 = 10**9 + 7
        m2 = 10**9 + 9
        x = randint(1, 10**9)
        hashes1 = [0]
        hashes2 = [0]
        for i, s in enumerate(string):
            hashes1.append((x * hashes1[i] + ord(s)) % m1)
            hashes2.append((x * hashes2[i] + ord(s)) % m2)
        for _ in range(n):
            a, b, l = map(int, infile.readline().split())
            hasha1 = hashes1[a + l] - x**l * hashes1[a]
            hashb1 = hashes1[b + l] - x**l * hashes1[b]
            if hasha1 % m1 == hashb1 % m1:
                hasha2 = hashes2[a + l] - x ** l * hashes2[a]
                hashb2 = hashes2[b + l] - x ** l * hashes2[b]
                if hasha2 % m2 == hashb2 % m2:
                    print("Yes", file=outfile)
                    continue
            print('No', file=outfile)
```

Input	Output
trololo 4 0 0 7 2 4 3 3 5 1 1 3 2	Yes Yes Yes No
abcabcdeabc 3 1 4 4 0 8 3 4 9 1	No Yes Yes
aaabaa 2 0 1 3 1 4 2	No Yes

Задача 5

Псевдокод взят из лекции и переписан на Python.

```
def pref(S):
    p = [0]*(len(S)+1)
    i, j = 1, 0
    while i < len(S):
        if S[i] == S[j]:
            p[i+1] = j + 1
            i += 1
            j += 1
        else:
            if j > 0:
                j = p[j]
            else:
                p[i+1] = 0
                i += 1
    return p


with open('input.txt', 'r') as infile:
    string = infile.readline().strip()
with open('output.txt', 'w') as outfile:
    print(*pref(string)[1:], file=outfile)
```

Input	Output
aaaAAA	0 1 2 0 0 0
abcabcdeabc	0 0 0 1 2 3 0 0 1 2 3
abacaba	0 0 1 0 1 2 3

Задача 7

Делал по заданию. Чтобы избежать коллизий, предварительно в двойном экземпляре вычислил хеш-значения всех подстрок длины k из s и t, где k от 1 до минимальной из длин s и t. Потом прошел по получившимся хэш-таблицам, начиная с максимального k, в поисках совпадающих хэшей (а значит и совпадающих подстрок).

```
from random import randint


with open('input.txt', 'r') as infile:
    with open('output.txt', 'w') as outfile:
        line = infile.readline().strip()
        while len(line) != 0:
            s, t = line.split()

            m1 = 10**9 + 7
            m2 = 10**9 + 9
            x = randint(1, 10**9)
            hashes_s_1 = [0]
            hashes_s_2 = [0]
            hashes_t_1 = [0]
            hashes_t_2 = [0]
            for i, letter in enumerate(s):
                hashes_s_1.append((x * hashes_s_1[i] +
ord(letter)) % m1)
                hashes_s_2.append((x * hashes_s_2[i] +
ord(letter)) % m2)
            for i, letter in enumerate(t):
                hashes_t_1.append((x * hashes_t_1[i] +
ord(letter)) % m1)
                hashes_t_2.append((x * hashes_t_2[i] +
ord(letter)) % m2)

            dict_with_hashes = {}
            for k in range(1, min(len(s), len(t)) + 1):
                dict_with_hashes[k] = {"s1": [], "t1": [], "s2":
[], "t2": []}
                for i in range(len(hashes_s_1) - k):

dict_with_hashes[k]["s1"].append((hashes_s_1[i + k] - x**k *
hashes_s_1[i]) % m1)

dict_with_hashes[k]["s2"].append((hashes_s_2[i + k] - x ** k *
hashes_s_2[i]) % m2)
                for i in range(len(hashes_t_1) - k):

dict_with_hashes[k]["t1"].append((hashes_t_1[i + k] - x**k *
hashes_t_1[i]) % m1)

dict_with_hashes[k]["t2"].append((hashes_t_2[i + k] - x ** k *
```

```
hashes_t_2[i]) % m2)

            flag = False
            for k in range(min(len(s), len(t)), 0, -1):
                current_dict = dict_with_hashes[k]
                for s_i, s_hash in
enumerate(current_dict["s1"]):
                    for t_i, t_hash in
enumerate(current_dict["t1"]):
                        if s_hash == t_hash:
                            if current_dict["s2"][s_i] ==
current_dict["t2"][t_i]:
                                print(s_i, t_i, k, file=outfile)
                                flag = True
                                break
                    if flag:
                        break
                if flag:
                    break

            if not flag:
                print(0, 0, 0, file=outfile)
            line = infile.readline().strip()
```

Input	Output
cool toolbox aaa bb aabaa babbaab korova korovka onetwothreefour fourthreefive getmeoutofthisplace thisisfineoutthere pottery potter	1 1 3 0 0 0 0 4 3 0 0 5 6 4 6 4 9 4 0 0 6

Дополнительные задачи

Задача: Поиск подстроки

Решение:

Алгоритм Кнута-Морриса-Пратта, адаптированный для поиска всех вхождений подстроки в исходную строку.

```
def pref(S):
    p = [0]*(len(S)+1)
    i, j = 1, 0
    while i < len(S):
        if S[i] == S[j]:
            p[i+1] = j + 1
            i += 1
            j += 1
        else:
            if j > 0:
                j = p[j]
            else:
                p[i+1] = 0
                i += 1
    return p


initstring = input().strip()
substring = input().strip()
mystr = substring + '#' + initstring
prefixes = pref(mystr)
length = len(substring)
found_ind = []
for i, prefix in enumerate(prefixes[length + 1:]):
    if prefix == length:
        found_ind.append(i - length)
print(*found_ind)
```

ID	Дата	Автор	Задача	Язык	Результат	Тест	Время	Память
19899375	19.09.2023 20:40:00	Федюкин Михаил	0202	Python	Accepted		0,14	5 Мб

Задача: Сдвиг текста

Решение:

Если строка является циклическим сдвигом исходной строки, то исходная строка будет найдена, если искать её в двух приложенных друг к другу сдвинутых строках.

```
init = input()
moved = input()
print((moved + moved).find(init))
```

ID	Дата	Автор	Задача	Язык	Результат	Тест	Время	Память
19899388	19.09.2023 20:41:22	Федюкин Михаил	0203	Python	Accepted		0,031	530 Кб

Задача: Подстроки из одинаковых букв

Решение:

Данная задача сводится к тому, что нужно найти максимальное расстояние между двумя одинаковыми символами.

```
string = input()
letters = {}
for i, let in enumerate(string):
    try:
        letters[let].append(i)
    except Exception:
        letters[let] = [i]
mx = 0
for spisok in letters.values():
    if spisok[-1] - spisok[0] > mx:
        mx = spisok[-1] - spisok[0]
print(mx)
```

ID	Дата	Автор	Задача	Язык	Результат	Тест	Время	Память
19899399	19.09.2023 20:43:07	Федюкин Михаил	0361	Python	Accepted		0,031	482 Кб

Выводы по проделанной работе

В работе со строками активно используется динамическое программирование, и есть много неочевидных ходов, позволяющих ускорять алгоритмы, и их трудно понять.

Достаточно интересным является подход с хэшированием. Хоть мы и тратим время на вычисление хэшей и их сравнение, хэширование оправдывает себя для длинных строк, так как посимвольно сравнивать длинную последовательность гораздо затратнее, чем сделать немного предварительных вычислений.