

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №4
по курсу «Алгоритмы и структуры данных»

Вариант 14

Выполнила:

Лаузер Я.П. К3144

Проверил:

Санкт-Петербург

2024 г.

Содержание отчета

Лабораторная №4

Задача №1. Наивный поиск подстроки в строке

Задача №5. Префикс-функция

Задача №6. Z-функция

4 лаба “Подстроки”.

Задача №1. Наивный поиск подстроки в строк

Текст задачи

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.

Листинг кода.

```
f = open('input.txt')
t2 = f.readline()
t1 = t2[:len(t2) - 1]
p1 = f.readline()

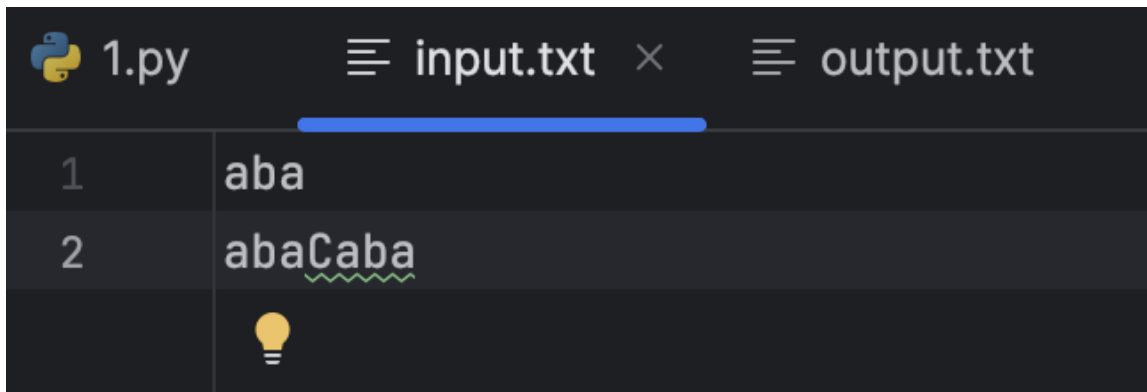
def NaiveStringMatcher(t, p):
    n = len(t)
    m = len(p)
    ans = []
    for i in range(n - m + 1):
        if t[i:i + m] == p:
            ans.append(i + 1)
    return ans

z = open('output.txt', 'w')
itog = NaiveStringMatcher(p1, t1)
z.write(str(len(itog)) + '\n')
itog.sort()
for i in itog:
    z.write(str(i) + ' ')
```

Текстовое объяснение решения.

Этот код представляет собой простую реализацию алгоритма поиска подстроки в тексте. Он принимает два входных аргумента: текст `t` и подстроку `p`, и возвращает список всех позиций, где подстрока `p` встречается в тексте `t`. Алгоритм работает путем пошагового сравнения подстроки `p` с каждым возможным сегментом текста `t` длиной, равной длине подстроки `p`. Если совпадение найдено, то позиция начала этого сегмента добавляется в выходной список. В коде используется файл `input.txt`, содержащий текст и подстроку, а результаты выводятся в файл `output.txt`.

Результат работы кода на примерах из текста задачи:



The screenshot shows a code editor with three tabs: `1.py`, `input.txt`, and `output.txt`. The `input.txt` tab is active and contains two lines of text. The first line is `aba`. The second line is `abaCaba`, where the substring `aba` is highlighted with a green wavy line. Below the text in the second line, there is a yellow lightbulb icon, indicating a suggestion or a tip.

Line	Text
1	aba
2	abaCaba

1.py	input.txt	output.txt
1	2	
2	1 5	

Задача №5. Префикс-функция

Текст задачи

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 $1, 2, \dots, |s|$, в указанном порядке.

Листинг кода

```
f = open('input.txt')
st = f.readline()

def PrefixFunction(s):
    p = [0] * len(s)
    for i in range(1, len(s)):
```

```

        k = p[i - 1]
        while k > 0 and s[i] != s[k]:
            k = p[k - 1]
        if s[i] == s[k]:
            k += 1
        p[i] = k
    return p

z = open('output.txt', 'w')
itog = PrefixFunction(st)
for i in itog:
    z.write(str(i) + ' ')

```

Объяснение кода:

Префикс-функция для строки s - это массив p , где $p[i]$ - длина максимального собственного префикса строки $s[0:i]$, являющегося одновременно суффиксом этой же строки. Алгоритм, который реализован в коде, позволяет эффективно находить префикс-функцию для любой заданной строки. Код обрабатывает входной файл "input.txt", содержащий строку, для которой необходимо найти префикс-функцию, и записывает результат в файл "output.txt".

Результат работы кода на примерах из текста задачи:

input.txt × output.txt 5.py	
1	aaaAAA

≡ input.txt	≡ output.txt	×	5.py
1	0 1 2 0 0 0		

Задача №6. Z-функция

Текст задачи

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 $1, 2, \dots, |s|$ строки s , в указанном порядке.

Листинг кода:

```
def zfunction(s):
    zf = [0] * len(s)
    left, right = 0, 0
    for i in range(1, len(s)):
        zf[i] = max(0, min(right - i, zf[i - left]))
        while i + zf[i] < len(s) and s[zf[i]] == s[i +
zf[i]]:
            zf[i] = zf[i] + 1
        if i + zf[i] > right:
            left = i
            right = i + zf[i]
    return zf
```

```
f = open('input.txt')
stroka1 = f.readline()
stroka = stroka1[:len(stroka1)]
rez = zfunction(stroka)
with open('output.txt', 'w') as z:
    for i in range(1, len(rez)):
        z.write(str(rez[i]) + ' ')
```

Объяснение кода:

Функция `zfunction(s)` принимает строку `s` в качестве аргумента и возвращает список `zf` той же длины, что и `s`, где каждый элемент `zf[i]` представляет собой длину наибольшего префикса-суффикса, начинающегося с позиции `i` в строке `s`.

Результат работы кода на примерах из текста задачи:

input.txt × output.txt 6.py	
1	aaaAAA

≡ input.txt

≡ output.txt ×

6.py

1

2 1 0 0 0