УНИВЕРСИТЕТ ИТМО

Факультет инфокоммуникационных технологий

Мобильные и сетевые технологии

Лабораторная работа №2.4

по дисциплине

«Алгоритмы и структуры данных»

Выполнила:

Юсипов Артем Александрович

К3140

Преподаватель:

Харьковская Т. А.

Санкт-Петербург

2022

Задача №5

Описание задания:

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

Решение:

```python
import sys

sys.stdin = open("input.txt")
sys.stdout = open("output.txt", "w")


def prefix_function(s):
    n = len(s)
    pi = [0 for i in range(n)]
    for i in range(1, n):
        j = pi[i - 1]
        while j > 0 and s[i] != s[j]:
            j = pi[j - 1]
        if s[i] == s[j]:
            j += 1
        pi[i] = j
    return pi


print(*prefix_function(input()))
```

Задача №6

Описание задания:

Постройте Z-функцию для заданной строки s.

Решение:

```
import sys

sys.stdin = open("input.txt")
sys.stdout = open("output.txt", "w")


def z_function(string):
    z = [0 for i in range(len(string))]
    left = right = 0
    for i in range(1, len(string)):
        if i >= right:
            j = 0
            while i + j < len(string) and string[i + j] == string[j]:
                j += 1
            left = i
            right = i + j
            z[i] = j
        else:
            if z[i - left] < right - i:
                z[i] = z[i - left]
            else:
                j = right - i
                while i + j < len(string) and string[i + j] == string[j]:
                    j += 1
                left = i
                right = i + j
                z[i] = j

    print(" ".join(list(map(str, z[1:]))))


m = input()
z_function(m)
```

Задача №7

Описание задания:

Для каждой пары строк si и ti 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.

Решение:

```python
import sys

sys.stdin = open("input.txt")
sys.stdout = open("output.txt", "w")


def hash_table(string):
    hashes = []
    for i in range(len(string)):
        res = []
        cont = 10 ** 9 + 9
        sm_str = string[0: len(string) - i]
        sm_hash = hash(sm_str)
        res.append([0, len(sm_str), sm_hash])
        for j in range(1, len(string) - len(sm_str) + 1):
            sum_1 = (sm_hash - (ord(string[j - 1]) * (13 ** (len(sm_str) - 1) % cont)) % cont) * 13
            sum_2 = ord(string[j + len(sm_str) - 1])
            sm_hash = (sum_1 + sum_2) % cont
            res.append([j, len(sm_str), sm_hash])
        hashes.append(res)
    return list(reversed(hashes))


def hash(string):
    cont = 10 ** 9 + 9
    hash = 0
    for i in string:
        hash = (hash * 13 + ord(i)) % cont
    return hash

def compare(string1, string2):
    table_1 = hash_table(string1)
    table_2 = hash_table(string2)
    for i in range(min(len(string1), len(string2)), 0, -1):
        for j in table_1[i - 1]:
            for t in table_2[i - 1]:
                if j[2] == t[2]:
                    sub_str = string1[j[0]: j[0] + j[1]]
                    return j[0], t[0], len(sub_str), sub_str
    return 0, 0, 0


string1, string2 = input().split()
print(*compare(string1, string2))
```

Задача №1

Описание задания:

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

Решение:

```python
import sys

sys.stdin = open("input.txt")
sys.stdout = open("output.txt", "w")


def KMP(part, string):
    M = len(part)
    N = len(string)
    inserts = list()
    lps = [0] * M
    j = 0
    prefix(part, M, lps)
    i = 0
    while i < N:
        if part[j] == string[i]:
            i += 1
            j += 1
        if j == M:
            inserts.append(i - j + 1)
            j = lps[j - 1]
        elif i < N and part[j] != string[i]:
            if j != 0:
                j = lps[j - 1]
            else:
                i += 1
    return inserts

def prefix(pat, M, pref):
    len = 0
    pref[0] = 0
    i = 1
    while i < M:
        if pat[i] == pat[len]:
            len += 1
            pref[i] = len
            i += 1
        else:
            if len != 0:
                len = pref[len - 1]
            else:
                pref[i] = 0
                i += 1


pattern = input()
string = input()
ans = KMP(pattern, string)
print(len(ans))
print(*ans)
```

Вывод:

Наивный алгоритм неэффективен

Задача №930 (Имена)

Описание задания:

На далекой планете Тау Кита есть непонятные нам обычаи. Например, таукитяне очень необычно для землян выбирают имена своим детям. Родители так выбирают имя ребенку, чтобы оно могло быть получено как удалением некоторого набора букв из имени отца, так и удалением некоторого набора букв из имени матери. Например, если отца зовут «abacaba», а мать — «bbccaa», то их ребенок может носить имена «a», «bba», «bcaa», но не может носить имена «aaa», «ab» или «bbc». Возможно, что имя ребенка совпадает с именем отца и/или матери, если оно может быть получено из имени другого родителя удалением нескольких (возможно, ни одной) букв. Пусть отец по имени X и мать по имени Y выбирают имя своему новорожденному ребенку. Так как в таукитянских школах учеников часто вызывают к доске в лексикографическом порядке имен учеников, то есть в порядке следования имен в словаре, то они хотят выбрать своему ребенку такое имя, чтобы оно лексикографически следовало как можно позже. Формально, строка S лексикографически больше строки T, если выполняется одно из двух условий: • строка T получается из S удалением одной или более букв с конца строки S; • первые (i - 1) символов строк T и S не различаются, а буква в i-й позиции строки T следует в алфавите раньше буквы в i-й позиции строки S. Требуется написать программу, которая по именам отца и матери находит лексикографически наибольшее имя для их ребенка.

Решение:

```
import sys

sys.stdin = open("input.txt")
sys.stdout = open("output.txt", "w")
s1 = input()
s2 = input()
count1 = [0 for i in range(128)]
count2 = [0 for j in range(128)]
for c in s1:
    count1[ord(c)] += 1
for c in s2:
    count2[ord(c)] += 1
ans = ''
pos1 = 0
pos2 = 0
for c in "zyxwvutsrqponmlkjihgfedcba":
    while count1[ord(c)] > 0 and count2[ord(c)] > 0:
        ans += c
        while s1[pos1] != c:
            count1[ord(s1[pos1])] -= 1
            pos1 += 1
        count1[ord(s1[pos1])] -= 1
        pos1 += 1
        while s2[pos2] != c:
            count2[ord(s2[pos2])] -= 1
            pos2 += 1
        count2[ord(s2[pos2])] -= 1
        pos2 += 1
print(ans)
```

Задача №886 (Суффиксы)

Описание задания:

Назовем строкой последовательность из маленьких букв английского алфавита. Строкой, например, является пустая последовательность “”, слово “aabaf” или бесконечная последовательность букв “a”. i-ый суффикс Si строки S – это строка S, из которой вырезаны первые i букв: так, для строки S = “aabaf” суффиксы будут такими: S0 = “aabaf” S1 = “abaf” S2 = “baf” S3 = “af” S4 = “f” S5 = S6 = S7 = . . . = “” Суффиксы определены для всех i > 0. Циклическое расширение S* конечной строки S – это строка, полученная приписыванием ее к самой себе бесконечное количество раз. Так, S* = S0* = “aabafaabafaa...” S1* = “abafabafabaf...” S2* = “bafbafbafbaf...” S3* = “afafafafafaf...” S4* = “fffffffffff...” S5* = S6* = S7*= . . . = “” По данной строке S выясните, сколько ее суффиксов Si имеют такое же циклическое расширение, как и сама строка S, то есть количество таких i, что S*= Si*.

Решение:

```
import sys

sys.stdin = open("input.txt")
sys.stdout = open("output.txt", "w")
s = input()
n = len(s)
prefix = [0 for i in range(1 + n)]
l = 0
for i in range(1, n):
    while True:
        if s[l] == s[i]:
            l += 1
            break
        if l == 0:
            break
        l = prefix[l]
    prefix[i + 1] = l
period = n - prefix[n]
if n % period != 0:
    ans = 1
else:
    ans = n // period
print(ans)
```

Задача №203

Описание задания:

Мальчик Кирилл написал однажды на листе бумаги строчку, состоящую из больших и маленьких английских букв, а после этого ушел играть в футбол. Когда он вернулся, то обнаружил, что его друг Дима написал под его строкой еще одну строчку такой же длины. Дима утверждает, что свою строчку он получил циклическим сдвигом строки Кирилла направо на несколько шагов (циклический сдвиг строки abcde на 2 позиции направо даст строку deabc). Однако Дима известен тем, что может случайно ошибиться в большом количестве вычислений, поэтому Кирилл в растерянности - верить ли Диме? Помогите ему! По данным строкам выведите минимально возможный размер сдвига вправо или -1, если Дима ошибся.

Решение:

```
import sys

sys.stdin = open("input.txt")
sys.stdout = open("output.txt", "w")
s1 = input()
s2 = input()
if s1 == s2:
    print(0)
else:
    for i in range(1, len(s1)):
        s3 = s1[len(s1)-i:] + s1[0:len(s1)-i]
        if s3 == s2:
            print(i)
            break
    else:
        print(-1)
```