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 17

Выполнил:
Прокопец С.Р.
К3139

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Оглавление

Задачи по варианту

Задача №2. Карта [2 s, 256 Mb, 1 балл]

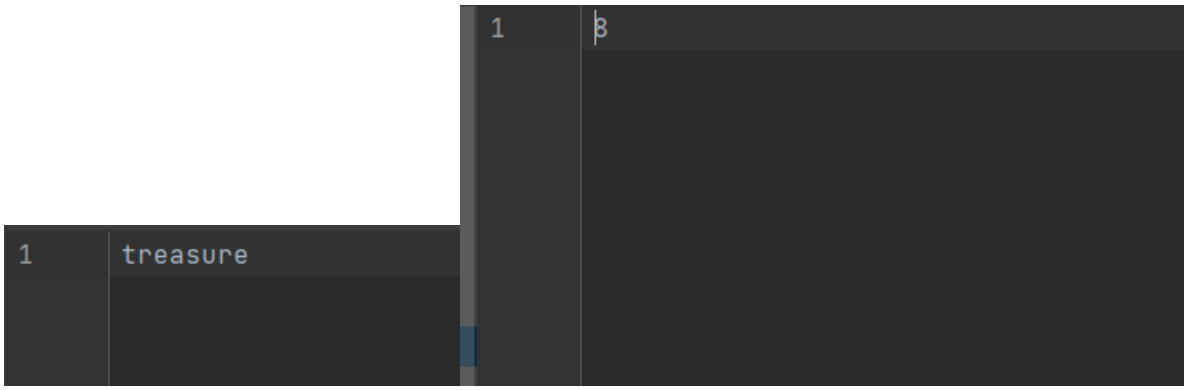
В далеком 1744 году во время долгого плавания в руки капитана Александра Смоллетта попала древняя карта с
указанием местонахождения сокровищ. Однако расшифровать ее содержание было не так уж и просто.

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.

```
with open('input.txt') as inp:
    word = inp.readline().split(' ')
    word = ''.join(word)
alph, q = dict(), dict()

for i in range(len(word)):
    letter = word[i]
    try:
        alph[letter] += 1
    except:
        alph[letter] = 0
        alph[letter] += 1

c = 0
for key in alph:
    copy = word
    if alph[key] >= 2:
        c += 1
        q[key] = []
        for i in range(len(word)):
            if word[i] == key:
                q[key].append(i)
result = 0

for letter in q:
    coef = len(q[letter]) - 1
    for i in reversed(range(len(q[letter]))):
        result += q[letter][i] * coef
        coef -= 2
    result -= (len(q[letter]) - 1) * len(q[letter]) // 2
```

```
with open('output.txt', 'w') as outp:
    outp.write(str(result))
```

Считываем строку и проходимся по ней, записывая в словарь alph количество символов в строке, а в словарь q записываем индексы этих символов, если их больше 1. Затем считываем количество возможных вариантов, считая количество символов между двумя повторяющимися для каждого символа.



Задача №4. 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.

```
def poly_hash(P, p, x=128):
    h = 0
    for i in reversed(range(len(P))):
        h = (h * x + ord(P[i])) % p
    return h % p


def precompute_hashes(T, P, p, x):
    H = [0] * (len(T) - len(P) + 1)
    S = T[len(T) - len(P): len(T)]
    ind = len(T) - len(P)
    H[ind] = poly_hash(S, p, x)
    y = 1
    for i in range(1, len(P) + 1):
        y = (y * x) % p
    for i in range(len(T) - len(P) - 1, -1, -1):
        H[i] = (x * H[i + 1] + ord(T[i]) - y * ord(T[i + len(P)]) + p) % p
    return H


with open("input.txt") as inp:
```

```
    word = inp.readline()[:-1]
    n = int(inp.readline())
    sp = []
    for _ in range(n):
        sp.append(list(map(int, inp.readline().split())))

hashes = dict()
coll = dict()
k = 1
l = len(word)
p, o = 10 ** 9 + 7, 10 ** 9 + 9

while k <= l:
    hashes[k] = []
    coll[k] = []
    for i in range(l - k + 1):
        hashes[k].append(poly_hash(word[i:i + k], p))
        coll[k].append(poly_hash(word[i:i + k], o))
    k += 1

with open('output.txt', 'w') as outp:
    for i in range(n):
        count = sp[i][-1]
        first = sp[i][0]
        second = sp[i][1]
        if hashes[count][first] == hashes[count][second] and
coll[count][first] == coll[count][second]:
            outp.write(f'YES\n')
        else:
            outp.write(f'NO\n')
```

Считываем строку и записываем хэши каждой ее подстроки в словарь с количеством символов этой подстроки. Создаем второй словарь с таким же условием, чтобы избежать коллизии. Затем проходим по командам, считываем индекс первого элемента первой подстроки и индекс первого элемента второй подстроки, и количество элементов этих подстрок. Затем в словарях сравниваем значения индексов первых элементов в ключе количества символов. Если значения совпадают в обоих словарях, то выводим YES, в противном случае NO

```
1  trololo
2  4
3  0 0 7
4  2 4 3
5  3 5 1
6  1 3 2
```

```
1  YES
2  YES
3  YES
4  NO
5
```

Задача №7. Наибольшая общая подстрока [15 s, 512 Mb, 2 балла]

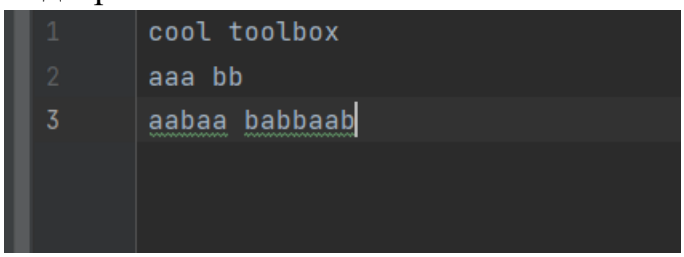
В задаче на наибольшую общую подстроку даются две строки s и t, и цель состоит в том, чтобы найти строку
w максимальной длины, которая является подстрокой как s, так и t. Это естественная мера сходства между двумя

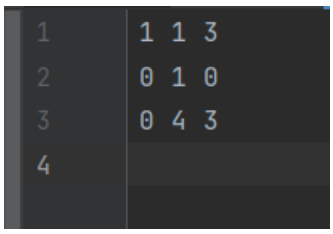
строками. Задача имеет применения для сравнения и сжатия текстов, а также в биоинформатике. Эту проблему
можно рассматривать как частный случай проблемы расстояния редактирования (Левенштейна), где разрешены только
вставки и удаления. Следовательно, ее можно решить за время O(|s||t|) с помощью динамического 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.

```
import random


def PolyHash(P, l, p, x):
    res = 0
    for i in reversed(range(l)):
        res = (res * x + ord(P[i])) % p
    return res % p


def CalculateHashes(T, l, k, p, x):
    H = [0] * (l - k + 1)
    S = T[l - k : l]
    H[l - k] = PolyHash(S, k, p, x)
    y = 1
    for i in range(1, k + 1):
        y = (y * x) % p
    for i in range(l - k - 1, -1, -1):
        H[i] = (x * H[i + 1] + ord(T[i]) - y * ord(T[i + k]) + p) % p
    return H


with open("input.txt") as f_read:
    with open("output.txt", "w") as f_write:
        while True:
            line = f_read.readline()
            if not line:
                exit()
            s, t = map(str, line.split())
            lS, lT = len(s), len(t)
            k = min(lS, lT)
            p = 10**9 + 7
            x = random.randint(1, p - 1)
            flag = False
            for i in reversed(range(1, k + 1)):
                Hs = CalculateHashes(s, lS, i, p, x)
                Ht = CalculateHashes(t, lT, i, p, x)
                for j in range(len(Hs)):
                    for h in range(len(Ht)):
                        if Hs[j] == Ht[h]:
                            f_write.write(str(j) + " " + str(h) + " " +
str(i) + "\n")
                            flag = True
                            break
                    if flag:
```

```
                    break
                if flag:
                    break
            if not flag:
                f_write.write("0" + " " + "1" + " " + "0" + "\n")
```

Считываем строки, первое слово разделяем на подстроки и
хэшируем каждую. Второе слово начинаем разделять на подстроки начиная с 1,
хэшируем и проверяем, есть ли такой хэш в словаре с количеством символов. Если
такой хэш есть, то переменную максим обновляем и увеличиваем количество символов
в подстроке второго слова. На выходе пишем максимальную длину совпавшей
подстроки.





Дополнительные задачи

Задача №1 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.

```
def areEqual(s1, s2):
    if len(s1) != len(s2):
        return False
    for i in range(len(s1)):
        if s1[i] != s2[i]:
            return False
    return True


def FindPattern(t, p):
    result = []
    for i in range(len(t) - len(p) + 1):
        if areEqual(t[i:i + len(p)], p):
            result.append(str(i + 1))
    return result
```

```
with open('input.txt') as inp:
    p = inp.readline()[:-1]
    t = inp.readline()
with open('output.txt', 'w') as outp:
    result = FindPattern(t, p)
    outp.write(f'{len(result)}\n')
    outp.write(' '.join(result))
```

```
1  aba
2  abaCaba
```

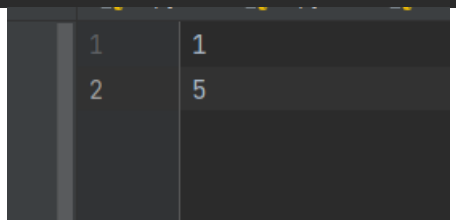
```
1  2
2  1 5
```

Задача №3 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.

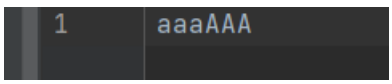
```
def pref(S):
    l = len(S)
    P = [0] * l
    i, j = 0, 1
    while j < l:
        if S[i] == S[j]:
            P[j] = i + 1
            i += 1
            j += 1
        elif i:
            i = P[i - 1]
        else:
            P[j] = 0
            j += 1
    return P


def kmp(text, sub):
    sub_len, text_len = len(sub), len(text)
    if not text_len or sub_len > text_len:
        return []
    P = pref(sub)
    entries = []
    i = j = 0
    while i < text_len and j < sub_len:
        if text[i] == sub[j]:
            if j == sub_len - 1:
                entries.append(str(i - sub_len + 2))
                j = P[j]
            else:
                j += 1
            i += 1
        elif j:
            j = P[j - 1]
```

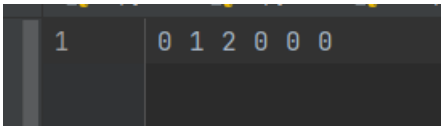
```
        else:
            i += 1
    return entries


with open('input.txt') as inp:
    sub = inp.readline()[:-1]
    s = inp.readline()
    P = kmp(s, sub)
with open('output.txt', 'w') as outp:
    outp.write(f'{len(P)}\n')
    outp.write(f'{" ".join(P)}')
```



Задача №5 Наивный поиск подстроки в строке [2 s, 256 Mb, 1 балл]

```
with open('input.txt', 'r') as inp:
    strr = inp.readline().strip()


def prefixFunction(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[1::]


res = prefixFunction(strr)
with open("output.txt", "w") as outp:
    outp.write(" ".join(map(str, res)))
    print(res)
```

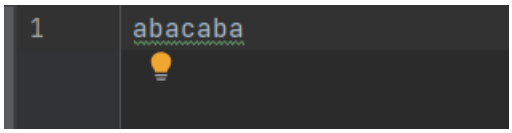


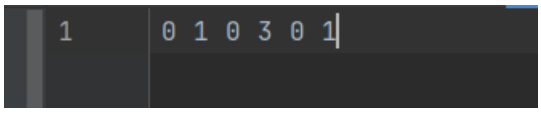


Задача №6 Z-функция [2 s, 256 Mb, 1.5 балла]

Постройте Z-функцию для заданной строки

```
with open('input.txt', 'r') as inp:
    strr = inp.readline().strip()


def zfun(s):
    out = []
    if not s:
        return out
    i, slen = 1, len(s)
    out.append(slen)
    while i < slen:
        left, right = 0, i
        while right < slen and s[left] == s[right]:
            left += 1
            right += 1
        out.append(left)
        i += 1
    return out[1::]


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


res = z_func(strr)

with open("output.txt", "w") as outp:
    outp.write(" ".join(map(str, res)))
```





Задача №8 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 = t0t1...tm−1 и p =
p0p1...pn−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) = titi+1...ti+n−1 различаются не более чем на k знаков.

```
import time


def main():
    with open("input.txt", "r") as input_file:
        results = []

        for line in input_file.readlines():
            line = line.split()

            mismatch_tolerance = int(line[0])
            text = line[1]
            pattern = line[2]

            result = []

            text_index_max = len(text) - len(pattern) + 1

            for text_index in range(text_index_max):
                missed = 0

                for pattern_index in range(len(pattern)):
                    text_char = text[text_index + pattern_index]
                    pattern_char = pattern[pattern_index]

                    if text_char != pattern_char:
                        missed += 1

                    if missed > mismatch_tolerance:
                        break

                if missed == mismatch_tolerance:
                    match = text[text_index : text_index + len(pattern)]
                    result.append(text_index)

            if len(result) == 0:
                results.append([0])
            else:
                results.append([len(result)] + result)

        with open("output.txt", "w") as output_file:
            for result in results:
                output = ""
                for i in result:
                    output += str(i) + " "
                output_file.write(output + "\n")


if __name__ == "__main__":
    t_start = time.perf_counter()
```

```
main()
print(time.perf_counter() - t_start)
```

```
0 abababb baaa
1 ababab baaa
1 xabcabc ccc
2 xabcabc ccc
3 aaa xxx
```

```
0
1 1
0
4 1 2 3 4
1 0
```

Задача №9 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 s1, a1, ..., sk, ak, где si - строки, а ai - числа, чтобы s = s1 · a1 + ... + sk · ak. Под операцией

умножения строки на целое положительное число подразумевается конкатенация одной или нескольких копий строки,

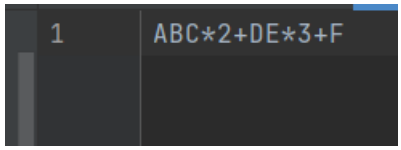
число которых равно числовому множителю, то есть, ABC*2=ABCABC . При этом требуется минимизировать общую

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.
 import sys


def z_func(s):
    n = len(s)
    z = [0] * n
    l = r = 0
    for i in range(1, n):
        if i <= r:
```

```
            z[i] = min(z[i - l], r - i + 1)
        while i + z[i] < n and s[i + z[i]] == s[z[i]]:
            z[i] += 1
        if i + z[i] > r:
            l, r = i, i + z[i] - 1
    return z


def step(ln, k, prev):
    res = k + 2 + len(str(ln // k))
    if ln == k:
        res -= 2
    if prev == n:
        res -= 1
    return res


sys.stdin = open("input.txt", "r")
s = input()
n = len(s)
s += "_"
dp = [n - i for i in range(n + 1)]
to = [[n - i, n - i] for i in range(n + 1)]
for i in range(n - 2, -1, -1):
    z = z_func(s[i:])
    if dp[i] > dp[i + 1] + 2:
        dp[i] = dp[i + 1] + 2
        to[i] = [1, 1]
    for j in range(i + 1, n + 1):
        k = 1
        while k * k <= j - i:
            if (j - i) % k:
                k += 1
                continue
            if z[k] + k >= j - i:
                if dp[i] > dp[j] + step(j - i, k, j):
                    dp[i] = dp[j] + step(j - i, k, j)
                    to[i] = [j - i, k]
            if z[(j - i) // k] + (j - i) // k >= j - i:
                if dp[i] > dp[j] + step(j - i, (j - i) // k, j):
                    dp[i] = dp[j] + step(j - i, (j - i) // k, j)
                    to[i] = [j - i, (j - i) // k]
            k += 1

with open("output.txt", "w") as f_write:
    i = 0
    while i < n:
        if i > 0:
            f_write.writelines("+")
        f_write.writelines(s[i : i + to[i][1]])
        if to[i][0] != to[i][1]:
            f_write.writelines("*" + str(to[i][0] // to[i][1]))
        i += to[i][0]
```



Мега супер пупер Дополнительные задачи

Задача №202 Поиск подстроки [2 s, 256 Mb, много баллов]

Найти все вхождения строки T в строке S.

```
with open('input.txt') as inp:
    t = str(inp.readline().rstrip())
    s = str(inp.readline().rstrip())
x = len(s)
S = s + '#' + t
w = len(S)


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


pi = prefix(S)
ans = 0
tmp = []
for i in range(w):
    if pi[i] == x:
        ans += 1
        tmp.append(i - x - x)

# print(pi)

# print(ans)
for k in range(len(tmp)):
    tmp[k] = str(tmp[k])
stroke = ' '.join(tmp)

with open('output.txt', 'w') as outp:
    outp.write(str(stroke))
```

18983587	09.03.2023 22:33:07	Прокопец Семен Романович	0202	Python	Accepted		0.093	4990 Кб

Задача №203 Сдвиг текста [2 s, 256 Mb, много баллов]

```
with open("input.txt", "r") as inp:
    kirill = inp.readline().replace("\n", "")
```

```
    dima = inp.readline().replace("\n", "")


if (0 < len(kirill) <= 10000 and 0 < len(dima) <= 10000
    and len(kirill) == len(dima)):

    y = open("output.txt", "w")
    result = 0

    if kirill == dima:
        print(0)
        y.write(str(0))
    else:
        for i in range(0, len(kirill)):

            first_part = kirill[len(kirill) - i - 1:len(kirill)]
            second_part = kirill[0:len(kirill) - i - 1]
            sdvig = first_part + second_part

            if sdvig == dima:
                result += 1
                break

            result += 1

        if result != len(kirill):
            print(result)
            y.write(str(result))
        else:
            print(-1)
            y.write(str(-1))

    y.close()
```

18983594	09.03.2023 22:35:09	Прокопец Семен Романовіч	0203	Python	Accepted		0,078	1362 Кб

Задача №361 Подстроки из одинаковых букв**[2 s, 256 Mb, много баллов]**

В заданной строке, состоящей из малых английских букв, необходимо найти пару самых длинных подстрок, состоящих из одних и тех же букв (возможно, в разном порядке). Например, в строке twotwow это будут подстроки wotwo и otwow.

```
with open("input.txt", "r") as inp:
    str1 = inp.readline().split()
str2 = str1[0]
len_of_w = 0
for i in range(len(str2)):
```

```
    lett = str2[i] #перебираем буквы и сравниваем_с_каждой
    for j in range(i, len(str2)):
        if lett == str2[j]:
            len_of_w2 = j - i
            if len_of_w2 > len_of_w:
                len_of_w = len_of_w2

#print(len_of_w)

with open("output.txt", "w") as outp:
    outp.write(str(len_of_w) + "\n")
```

18983623	09.03.2023 22:40:49	Прокопец Семен Романович	0361	Python	Accepted		0,046	482 Кб