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 20

Выполнил:

Галилей Кирилл Дмитриевич

K3240

Проверил:

Афанасьев А.В.

Санкт-Петербург 2024 г.

Задачи по варианту	3
Задача. №2 Карта.	3
Задача №3 Паттерн в тексте.	7
Задача №9 Декомпозиция строки.	10
Reiron	13

Задачи по варианту

Задача. №2 Карта.

2 Задача. 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 *х* 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.

- Формат ввода / входного файла (input.txt). В единственной строке входного файла дано послание, написанное на карте.
- Ограничения на входные данные. Длина послания не превышает 3 · 10⁵. Гарантируется, что послание может содержать только строчные буквы английского алфавита и пробелы. Также гарантируется, что послание не пусто. Послание не может начинаться с пробела или заканчиваться им.
- Формат вывода / выходного файла (output.txt). Выведите одно число x число способов вычеркнуть из послания все буквы кроме трех так, чтобы оставшееся слово одинаково читалось слева направо и справа налево.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

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

Код программы

```
if message[i] == message[k]:
                    count += 1
    return count
with open('input.txt', 'r') as file:
   message = file.readline().strip()
start time = time.time()
tracemalloc.start()
result = count palindromic triplets(message)
end time = time.time()
current, peak = tracemalloc.get traced memory()
tracemalloc.stop()
print(f"Время выполнения: {end time - start time} секунд")
print(f"Использование памяти: {current / 10**6} MB (текущая), {peak /
with open('output.txt', 'w') as file:
    file.write(f"{result}\n")
```

Текстовое объяснение решения

Программа читает входное сообщение из файла input.txt и удаляет из него все пробелы.

Затем она перебирает все возможные комбинации трех букв в строке и проверяет, образуют ли они палиндром.

Если комбинация образует палиндром, счетчик увеличивается.

Результат работы кода на примерах из текста задачи:

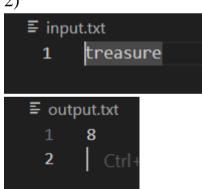
```
1)
```

```
    input.txt
        you will never find the treasure

    output.txt

       146
```

2)



	Время выполнения, с	Затраты памяти, МБ
Пример из задачи	0.001616477966308593	0.000212
Пример из задачи	0.0000824000	0.000144

Вывод по задаче: Программа корректно работает на всех приведенных тестах и укладывается в ограничения по времени и памяти

Задача №3 Паттерн в тексте.

3 Задача. 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.

- Формат ввода / входного файла (input.txt). На входе две строки: паттерн P и текст T. Требуется найти все вхождения строки P в строку T в качестве подстроки.
- Ограничения на входные данные. $1 \le |P|, |T| \le 10^6$. Паттерн и текст содержат только латинские буквы.
- 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.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input	output	input	output	input	output
aba	2	Test	1	aaaaa	3
abacaba	1 5	testTesttesT	5	baaaaaaa	2 3 4

• В первом примере паттерн *aba* можно найти в позициях 1 (abacaba) и 5 (abacaba) текста *abacaba*.

Паттерн и текст в этой задаче чувствительны к регистру. Поэтому во втором примере паттерн Test встречается только в 45 позиции в тексте testTesttesT.

Обратите внимание, что вхождения шаблона в тексте могут перекрываться, и это нормально, вам все равно нужно вывести их все.

 Используйте оператор == в Python вместо реализации собственной функции AreEqual для строк, потому что встроенный оператор == будет работать намного быстрее.

Код программы

```
def rabin_karp(pattern, text):
    p_len = len(pattern)
    t_len = len(text)
    p_hash = hash(pattern)
    result = []

for i in range(t_len - p_len + 1):
        t_sub_hash = hash(text[i:i + p_len])
        if p_hash == t_sub_hash and text[i:i + p_len] == pattern:
            result.append(i + 1) # Нумерация с 1

return result

def main():
    with open('input.txt', 'r') as file:
        pattern = file.readline().strip()
        text = file.readline().strip()

positions = rabin_karp(pattern, text)
```

```
count = len(positions)

with open('output.txt', 'w') as file:
    file.write(f"{count}\n")
    file.write(" ".join(map(str, positions)) + "\n")

if __name__ == "__main__":
    import time
    import tracemalloc

start_time = time.time()
    tracemalloc.start()

main()

current, peak = tracemalloc.get_traced_memory()
    print(f"Время выполнения: {time.time() - start_time} секунд")
    print(f"Использование памяти: {current / 10**6} МВ; Пиковое
использование памяти: (peak / 10**6) МВ")

tracemalloc.stop()
```

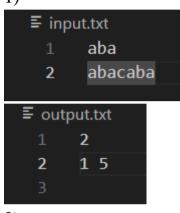
Текстовое объяснение решения

Для поиска всех вхождений шаблона в текст используется вспомогательная функция rabin_karp, которая принимает шаблон и текст. В этой функции инициализируется длина шаблона и текста, а также вычисляется хеш-код шаблона. Создается пустой список для хранения результатов. Затем перебираются все возможные подстроки текста длиной, равной длине шаблона, и для каждой подстроки вычисляется хеш-код. Если хеш-код подстроки совпадает с хеш-кодом шаблона и сама подстрока совпадает с шаблоном, то индекс начала этой подстроки добавляется в список результатов. В конце функция возвращает список всех позиций вхождений шаблона в текст.

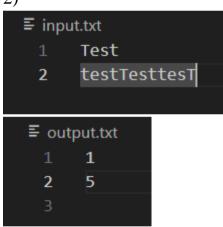
Функция main открывает файл input.txt и считывает первую строку как шаблон и вторую строку как текст. Затем вызывает функцию rabin_karp для поиска всех вхождений шаблона в текст и получает список позиций вхождений. Записывает количество вхождений и сами позиции в файл output.txt.

Результат работы кода на примерах из текста задачи:

1)



2)



	Время выполнения, с	Затраты памяти, МБ
Пример из задачи	0.001010656356811523	0.018031
	0.001014471054077148	0.018043

Вывод по задаче: Программа корректно работает на всех приведенных тестах и укладывается в ограничения по времени и памяти

Задача №9 Декомпозиция строки.

9 Задача. Декомпозиция строки [2 s, 256 Mb, 2 балла]

Ваша задача — построить наиболее экономное представление данной строки s в виде, продемонстрированном выше, а именно, подобрать такие $s_1, a_1, ..., s_k, a_k$, где s_i - строки, а a_i - числа, чтобы $s=s_1\cdot a_1+...+s_k\cdot a_k$. 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.

- Формат ввода / входного файла (input.txt). Одна строка входного файла содержит s. Строка состоит из букв латинского алфавита.
- Ограничения на входные данные. $1 \le |s| \le 5 \cdot 10^3$.
- **Формат вывода / выходного файла (output.txt).** Выведите оптимальное представление строки, данной во входном файле. Если оптимальных представлений несколько, выведите любое.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt	input.txt	output.txt
ABCABCDEDEDEF	ABC*2+DE*3+F	Hello	Hello

Код программы

```
def encode_string(s):
    n = len(s)
    dp = [["" for _ in range(n)] for _ in range(n)]

for l in range(n):
    for i in range(n - l):
        j = i + l
        substr = s[i:j + 1]
        dp[i][j] = substr

    if j - i < 4:
        continue

    for k in range(i, j):
        if len(dp[i][j]) > len(dp[i][k] + '+' + dp[k + 1][j]):
```

```
dp[i][j] = dp[i][k] + '+' + dp[k + 1][j]
            for k in range(1, len(substr)):
                repeat str = substr[:k]
                if repeat str and len(substr) % len(repeat str) == 0 and
substr == repeat str * (len(substr) // len(repeat str)):
                    encoded = f''\{dp[i][i + k - 1]\}*\{len(substr) //
len(repeat str)}"
                   if len(encoded) < len(dp[i][j]):</pre>
                        dp[i][j] = encoded
    return dp[0][n-1]
def main():
    with open('input.txt', 'r') as file:
        s = file.readline().strip()
    result = encode string(s)
   with open('output.txt', 'w') as file:
        file.write(result + '\n')
    start time = time.time()
    tracemalloc.start()
   main()
    current, peak = tracemalloc.get traced memory()
    print(f"Время выполнения: {time.time() - start time:.10f} секунд")
    print(f"Использование памяти: {current / 10**6:.6f} МВ; Пиковое
использование памяти: {peak / 10**6:.6f} MB")
    tracemalloc.stop()
```

Текстовое объяснение решения

Для декомпозиции строки используется вспомогательная функция encode_string, которая принимает строку s. В этой функции инициализируется двумерный список dp для хранения оптимальных представлений подстрок. Затем перебираются все возможные подстроки строки s и для каждой подстроки вычисляется ее оптимальное представление.

Если длина подстроки меньше 4, она сохраняется в исходном виде. Для подстрок длиной 4 и более проверяются все возможные разбиения на две части, и если объединение этих частей дает более короткое представление, чем текущее, оно обновляется. Также проверяются все возможные повторяющиеся подстроки, и если подстрока может быть представлена как повторение более короткой подстроки, это представление сохраняется, если оно короче текущего.

В конце функция возвращает оптимальное представление всей строки з.

Результат работы кода на примерах из текста задачи:

1)

input.txt

input.



	Время выполнения, с	Затраты памяти, МБ
Пример из задачи	0.0020027161	0.017978
Пример из задачи	0.0021336079	0.017994

Вывод по задаче: Программа корректно работает на всех приведенных тестах и укладывается в ограничения по времени и памяти раничения по времени и памяти

Вывод

В ходе данной лабораторной работы я научился решать задачи. Написанные программы были протестированы, а также были измерены потребляемый ими объём памяти и время работы. Все программы работаю корректно и укладываются в установленные ограничения по времени и памяти на примерах из задач.