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

Отчет по лабораторной работе №1
по курсу «Алгоритмы и структуры данных»
Тема: Greedy and DP
Вариант 2

Выполнила:
Бочкарева Е.А.
К3144

Проверила:
Артамонова В.Е.

Санкт-Петербург
2024 г.

Содержание отчета

Задачи по варианту

Задача №3 Максимальный доход от рекламы

У вас есть n объявлений для размещения на популярной интернет-странице. Для каждого объявления вы знаете, сколько рекламодатель готов платить за один клик по этому объявлению. Вы настроили n слотов на своей странице и оценили ожидаемое количество кликов в день для каждого слота. Теперь ваша цель - распределить рекламу по слотам, чтобы максимизировать общий доход.

- **Постановка задачи.** Даны две последовательности $a_1, a_2, ..., a_n$ (a_i - прибыль за клик по i-му объявлению) и $b_1, b_2, ..., b_n$ (b_i - среднее количество кликов в день i-го слота), нужно разбить их на n пар (a_i, b_j) так, чтобы сумма их произведений была максимальной.
- **Формат ввода / входного файла (input.txt).** В первой строке содержится целое число n, во второй - последовательность целых чисел $a_1, a_2, ..., a_n$, в третьей - последовательность целых чисел $b_1, b_2, ..., b_n$.
- **Ограничения на входные данные.** $1 \leq n \leq 10^3$, $-10^5 \leq a_i, b_i \leq 10^5$, для всех $1 \leq i \leq n$.
- **Формат вывода / выходного файла (output.txt).** Выведите максимальное значение $\sum_{i=1}^{n} a_i c_i$, где $c_1, c_2, ..., c_n$ является перестановкой $b_1, b_2, ..., b_n$.
- Ограничение по времени. 2 сек.
- Примеры:

input.txt	output.txt
1 23 39	897

input.txt	output.txt
3 1 3 -5 -2 4 1	23

Во втором примере $23 = 3 \cdot 4 + 1 \cdot 1 + (-5) \cdot (-2)$.

Листинг кода:

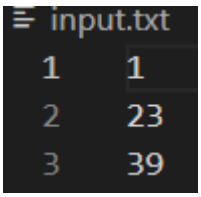
```
import resource
import time



# Starting time measuring
start_time = time.perf_counter()

# Reading from the input file
with open("input.txt") as f:
    n = int(f.readline())
    a = list(map(int, f.readline().split()))
    b = list(map(int, f.readline().split()))

a.sort()
```

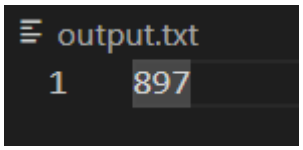
```
b.sort()

answer = 0
for i in range(n):
    answer += a[i] * b[i]

# Writing to the output file
with open("output.txt", "w") as f:
    f.write(str(answer))

# Writing the time measurements to the standard output
end_time = time.perf_counter()
duration = end_time - start_time
print(f"Execution time: {duration}s")

# Writing memory usage to the standard output
memory_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss
print(f"Memory usage: {memory_usage}B")
```

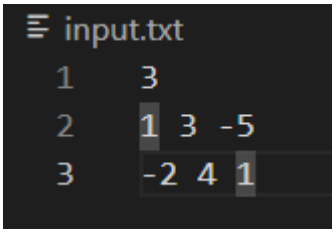
Пояснение к решению:

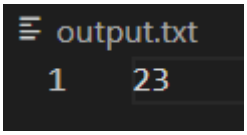
Это простая задача, требующая отсортировать значения для вычисления наиболее выгодного размещения. Когда значения в a и b упорядочены, можно их перемножить – это и будет оптимальным решением.

Результат работы на примерах из текста к задаче:









Вывод по задаче:

Сортировка – наиболее оптимальный способ решения данной задачи.

Задача №5 Максимальное количество призов

Вы организуете веселый конкурс для детей. В качестве призового фонда у вас есть n конфет. Вы хотели бы использовать эти конфеты для раздачи k лучшим местам в конкурсе с естественным ограничением, заключающимся в том, что чем выше место, тем больше конфет. Чтобы осчастливить как можно больше детей, вам нужно найти наибольшее значение k, для которого это возможно.

- **Постановка задачи.** Необходимо представить заданное натуральное число n в виде суммы как можно большего числа попарно различных натуральных чисел. То есть найти максимальное k такое, что n можно записать как $a_1 + a_2 + ... + a_k$, где $a_1, ..., a_k$ - натуральные числа и $a_i \neq a_j$ для всех $1 \leq i < j \leq k$.
- **Формат ввода / входного файла (input.txt).** Входные данные состоят из одного целого числа n.
- **Ограничения на входные данные.** $1 \leq n \leq 10^9$.
- **Формат вывода / выходного файла (output.txt).** В первой строке выведите максимальное число k такое, что n можно представить в виде суммы k попарно различных натуральных чисел. Во второй строке выведите эти k попарно различных натуральных чисел, которые в сумме дают n (если таких представлений много, выведите любое из них).
- Ограничение по времени. 2 сек.
- Примеры:

№	input.txt	output.txt
1	6	3 1 2 3

№	input.txt	output.txt
2	8	3 1 2 5

№	input.txt	output.txt
3	2	1 2

Листинг кода:

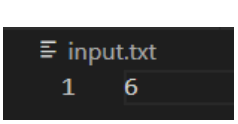
```
import resource
import time



# Starting time measuring
start_time = time.perf_counter()

# Reading from the input file
with open("input.txt") as f:
    n = int(f.readline())

cur_candies = 0
all_candies = []
for k in range(1, n + 1):
    if cur_candies + k <= n:
```

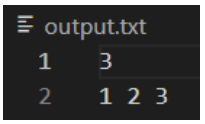
```
        all_candies.append(k)
        cur_candies += k
        answer = k
    else:
        all_candies[-1] += n - cur_candies
        break

# Writing to the output file
with open("output.txt", "w") as f:
    f.write(str(answer) + "\n")
    f.write(" ".join(list(map(str, all_candies))))

# Writing the time measurements to the standard output
end_time = time.perf_counter()
duration = end_time - start_time
print(f"Execution time: {duration}s")

# Writing memory usage to the standard output
memory_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss
print(f"Memory usage: {memory_usage}B")
```

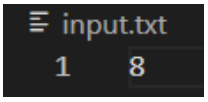
Пояснение к решению:

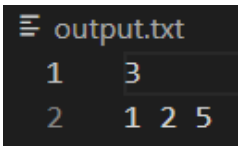
Используем жадные алгоритмы, чтобы детям всё-таки достались конфеты.

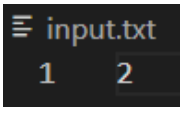
Результат работы на примерах из текста к задаче:

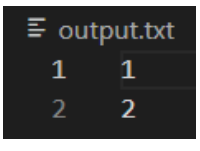












Вывод по задаче: потренировалась в использовании жадных алгоритмов.

Задача №9 Распечатка

- **Постановка задачи.** Диссертация дело сложное, особенно когда нужно ее печатать. При этом вам нужно распечатать не только текст самой диссертации, так и другие материалы (задание, рецензии, отзывы, авторефераты для защиты и т.п.). Вы оценили объём печати в N листов. Фирма, готовая размножить печатные материалы, предлагает следующие финансовые условия. Один лист она печатает за A_1 рублей, 10 листов - за A_2 рублей, 100 листов - за A_3 рублей, 1000 листов - за A_4 рублей, 10000 листов - за A_5 рублей, 100000 листов - за A_6 рублей, 1000000 листов - за A_7 рублей. При этом не гарантируется, что один лист в более крупном заказе обойдется дешевле, чем в более мелком. И даже может оказаться, что для любой партии будет выгодно воспользоваться тарифом для одного листа. Печать конкретного заказа производится или путем комбинации нескольких тарифов, или путем заказа более крупной партии. Например, 980 листов можно распечатать, заказав печать 9 партий по 100 листов плюс 8 партий по 10 листов, сделав 98 заказов по 10 листов, 980 заказов по 1 листу или заказав печать 1000 (или даже 10000 и более) листов, если это окажется выгоднее. Требуется по заданному объему заказа в листах N определить минимальную сумму денег в рублях, которой будет достаточно для выполнения заказа.
- **Формат ввода / входного файла (input.txt).** На вход программе сначала подается число N – количество листов в заказе. В следующих 7 строках ввода находятся натуральные числа $A_1, A_2, A_3, A_4, A_5, A_6, A_7$ соответственно.
- **Ограничения на входные данные.** $1 \leq N \leq 2 \times 10^9, 1 \leq A_i \leq 10^6$
- **Формат вывода / выходного файла (output.txt).** Выведите одно число – минимальную сумму денег в рублях, которая нужна для выполнения заказа. Гарантируется, что правильный ответ не будет превышать 2×10^9.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt
980 1 9 90 900 1000 10000 10000	882

input.txt	output.txt
980 1 10 100 1000 900 10000 10000	900

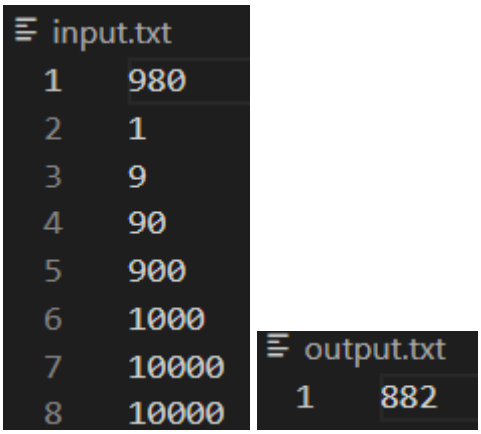
Листинг кода:

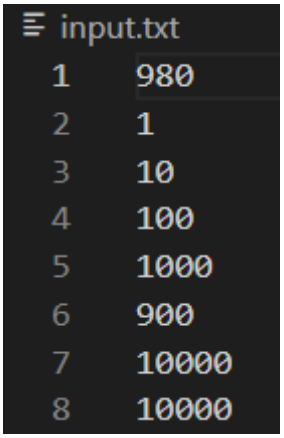
```python
import resource
import time


# Starting time measuring
start_time = time.perf_counter()

p = {}

# Reading from the input file
with open("input.txt") as f:
    n = int(f.readline())
    for i in range(7):
        p[10 ** i] = int(f.readline())

dp = []
dp.append(0)

for i in range(1, max(n + 1, 10 ** 6 + 1)):
    dp.append(p[1] * i)
    for j in p:
        if i - j >= 0:
            dp[i] = min(dp[i], dp[i - j] + p[j])

ans = dp[n]
for i in range(n + 1, 10 ** 6 + 1):
    ans = min(ans, dp[i])

# Writing to the output file
with open("output.txt", "w") as f:
    f.write(str(ans) + "\n")

# Writing the time measurements to the standard output
end_time = time.perf_counter()
duration = end_time - start_time
print(f"Execution time: {duration}s")

# Writing memory usage to the standard output
memory_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss
print(f"Memory usage: {memory_usage}B")
```

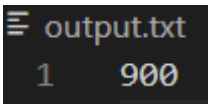
Пояснение к решению:

Для рассчета минимальной стоимости наиболее выгодно использовать дп, как и показано в коде.

Результат работы на примерах из текста к задаче:







Вывод по задаче:

В ходе решения данной задачи я научилась использовать динамическое программирование. Оно помогает наиболее оптимально решить задачу, которая, к примеру, с помощью жадных алгоритмов не решается корректно.

Задача №14 Максимальное значение арифметического выражения

В этой задаче ваша цель - добавить скобки к заданному арифметическому выражению, чтобы максимизировать его значение.

$$max(5 - 8 + 7 \times 4 - 8 + 9) = ?$$

- **Постановка задачи.** Найдите максимальное значение арифметического выражения, указав порядок применения его арифметических операций с помощью дополнительных скобок.
- **Формат ввода / входного файла (input.txt).** Единственная строка входных данных содержит строку s длины $2n + 1$ для некоторого n с символами $s_0, s_1, ..., s_{2n}$. Каждый символ в четной позиции s является цифрой (то есть целым числом от 0 до 9), а каждый символ в нечетной позиции является одной из трех операций из $+, -, *$
- **Ограничения на входные данные.** $0 \leq n \leq 14$ (следовательно, строка содержит не более 29 символов).
- **Формат вывода / выходного файла (output.txt).** Выведите максимально возможное значение заданного арифметического выражения среди различных порядков применения арифметических операций.
- Ограничение по времени. 5 сек.
- Пример:

input.txt	output.txt	input.txt	output.txt
1+5	6	5-8+7*4-8+9	200

Здесь $200 = (5 - ((8 + 7) * (4 - (8 + 9))))$.

Листинг кода:

```
import resource
import time



# Starting time measuring
start_time = time.perf_counter()

# Performs an arithmetic operation between two numbers a and b
def eval(a, b, op):
    if op == '*':
        return a * b
    elif op == '+':
        return a + b
    elif op == '-':
        return a - b

# Function to calculate the minimum and maximum values for a
subexpression
def min_and_max(i, j, min_array, max_array, operators):
```

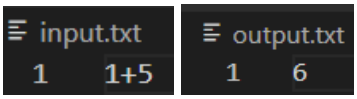
```
    min_value = float('inf')
    max_value = float('-inf')

    # Iterate over all possible operators between i and j
    for k in range(i, j):
        op = operators[k]
        a = eval(min_array[i][k], min_array[k + 1][j], op)
        b = eval(min_array[i][k], max_array[k + 1][j], op)
        c = eval(max_array[i][k], min_array[k + 1][j], op)
        d = eval(max_array[i][k], max_array[k + 1][j], op)

        # Update minimum and maximum values
        min_value = min(min_value, a, b, c, d)
        max_value = max(max_value, a, b, c, d)

    return min_value, max_value

# Returns the maximum possible value of an expression
def get_maximum_value(expression):

    digits = []
    operators = []

    # Separation of numbers and operators
    for i, c in enumerate(expression):
        if i % 2 == 0:
            digits.append(int(c))
        else:
            operators.append(c)

    n = len(digits)

    # Initializing tables for minimum and maximum values
    min_array = [[0] * n for _ in range(n)]
    max_array = [[0] * n for _ in range(n)]

    # Filling the diagonal of the table with the original numbers
    for i in range(n):
        min_array[i][i] = digits[i]
        max_array[i][i] = digits[i]

    # Loop by subexpression size
    for s in range(1, n):
```

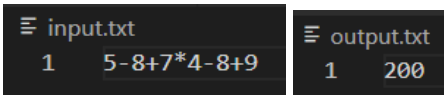
```
        for i in range(n - s):
            j = i + s
            min_array[i][j], max_array[i][j] = min_and_max(i, j,
min_array, max_array, operators)

    # Return the maximum value for the entire expression
    return max_array[0][n - 1]

# Reading data from input.txt file
with open('input.txt', 'r') as f:
    expression = f.readline().strip()

# Finding the maximum value of an expression
result = get_maximum_value(expression)

# Write the result to the output.txt file
with open('output.txt', 'w') as f:
    f.write(str(result))

# Writing the time measurements to the standard output
end_time = time.perf_counter()
duration = end_time - start_time
print(f"Execution time: {duration}s")

# Writing memory usage to the standard output
memory_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss
print(f"Memory usage: {memory_usage}B")
```

Пояснение к решению:

Суть решения в том, что мы используем ДП чтобы посчитать максимальное значение выражения.

Результат работы на примерах из текста к задаче:





Вывод по задаче: Научилась находить максимальное значение выражения с помощью дп.

Задача №18 Кафе

- **Постановка задачи.** Около университета недавно открылось новое кафе, в котором действует следующая система скидок: при каждой покупке более чем на 100 рублей покупатель получает купон, дающий право на один бесплатный обед (при покупке на сумму 100 рублей и меньше такой купон покупатель не получает). Однажды вам на глаза попался прейскурант на ближайшие n дней. Внимательно его изучив, вы решили, что будете обедать в этом кафе все n дней, причем каждый день вы будете покупать в кафе ровно один обед. Однако стипендия у вас небольшая, и поэтому вы хотите по максимуму использовать предоставляемую систему скидок так, чтобы ваши суммарные затраты были минимальны. Требуется найти минимально возможную суммарную стоимость обедов и номера дней, в которые вам следует воспользоваться купонами.
- **Формат ввода / входного файла (input.txt).** В первой строке входного файла дается целое число n - количество дней. В каждой из последующих n строк дано одно неотрицательное целое число s_i – стоимость обеда в рублях на соответствующий день i.
- **Ограничения на входные данные.** $0 \leq n \leq 100$, $0 \leq s_i \leq 300$ для всех $0 \leq i \leq n$.
- **Формат вывода / выходного файла (output.txt).** В первой строке выдайте минимальную возможную суммарную стоимость обедов. Во второй строке выдайте два числа k_1 и k_2 – количество купонов, которые останутся у вас неиспользованными после этих n дней и количество использованных вами купонов соответственно. В последующих k_2 строках выдайте в возрастающем порядке номера дней, когда вам следует воспользоваться купонами. Если существует несколько решений с минимальной суммарной стоимостью, то выдайте то из них, в котором значение k_1 максимально (на случай, если вы когда-нибудь ещё решите заглянуть в это кафе). Если таких решений несколько, выведите любое из них.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 64 мб.
- Пример:

input.txt	output.txt
5 110 40 120 110 60	260 0 2 3 5

input.txt	output.txt
3 110 110 110	220 1 1 2

Листинг кода:

```
import resource
import time



# Starting time measuring
start_time = time.perf_counter()
```

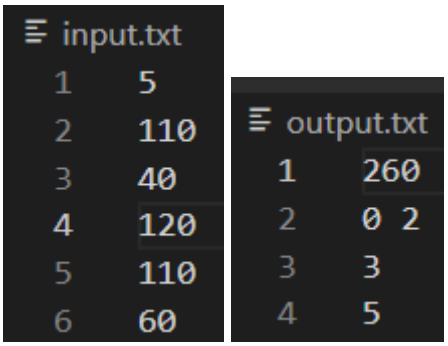
```
# Reading from the input file
with open("input.txt", "r") as f:
    n = int(f.readline())
    costs = []
    for i in range(n):
        costs.append(int(f.readline()))

# Creating dp to store minimum values
dp = [[float('inf')] * (n + 1) for _ in range(n + 1)]
dp[0][0] = 0

# Filling dp
for i in range(1, n + 1):
    for j in range(n + 1):
        if j < n:
            dp[i][j] = dp[i - 1][j + 1]

        if costs[i - 1] > 100:
            dp[i][j] = min(dp[i][j], dp[i - 1][j - 1] + costs[i - 1])
        else:
            dp[i][j] = min(dp[i][j], dp[i - 1][j] + costs[i - 1])

# Finding the minimum cost and the number of unused coupons
min_cost = float('inf')
for j in range(n + 1):
    if dp[n][j] <= min_cost:
        min_cost = dp[n][j]
        unused_coupons = j

# Recovering path and finding days of using coupons
days = []
i, j = n, unused_coupons
while i > 0:
    if dp[i][j] == dp[i - 1][j + 1]:
        days.append(i)
        j += 1
    elif costs[i - 1] > 100 and dp[i][j] == dp[i - 1][j - 1] + costs[i
- 1]:
        j -= 1
    i -= 1
days.reverse()
```

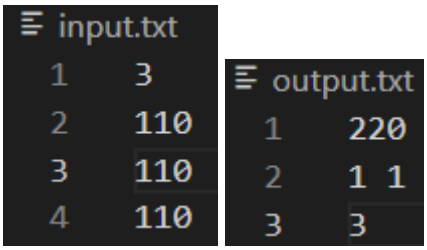
```
# Write the result to the output.txt file
with open('output.txt', 'w') as f:
    f.write(str(min_cost) + "\n")
    f.write(str(unused_coupons) + " " + str(len(days)) + "\n")
    for day in days:
        f.write(str(day) + "\n")

# Writing the time measurements to the standard output
end_time = time.perf_counter()
duration = end_time - start_time
print(f"Execution time: {duration}s")

# Writing memory usage to the standard output
memory_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss
print(f"Memory usage: {memory_usage}B")
```

Пояснение к решению:

Код решает проблему, связанную с минимизацией стоимости покупки товаров при использовании купонов с помощью ДП. Каждый шаг описан в комментариях.

Результат работы на примерах из текста к задаче:





Вывод по задаче:

В ходе решения данной задачи я потренировалась использовать динамическое программирование и генерировать двумерные массивы.

Дополнительные задачи

Задача №17 Ход конем

- **Постановка задачи.** Шахматная ассоциация решила оснастить всех своих сотрудников такими телефонными номерами, которые бы набирались на кнопочном телефоне ходом коня. Например, ходом коня набирается телефон 340-49-27. При этом телефонный номер не может начинаться ни с цифры 0, ни с цифры 8.

1	2	3
4	5	6
7	8	9
.	0	.

  Напишите программу, определяющую количество телефонных номеров длины N, набираемых ходом коня. Поскольку таких номеров может быть очень много, выведите ответ по модулю 10^9.

- **Формат ввода / входного файла (input.txt).** Во входном файле записано одно целое число N.

- **Ограничения на входные данные.** $1 \leq N \leq 1000$.

- **Формат вывода / выходного файла (output.txt).** Выведите в выходной файл искомое количество телефонных номеров по модулю 10^9.

- Ограничение по времени. 1 сек.

- Ограничение по памяти. 256 мб.

- Примеры:

input.txt	output.txt
1	8

input.txt	output.txt
2	16

Листинг кода:

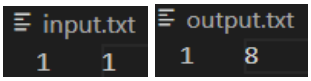
```
import resource
import time



# Starting time measuring
start_time = time.perf_counter()

MOD = 10 ** 9

# Reading from the input file
with open("input.txt") as f:
    n = int(f.readline())

dp = [[0] * 10 for _ in range(n + 1)]
for i in range(10):
    dp[1][i] = 1
```

```
for i in range(2, n + 1):
    dp[i][0] = (dp[i - 1][4] + dp[i - 1][6]) % MOD
    dp[i][1] = (dp[i - 1][6] + dp[i - 1][8]) % MOD
    dp[i][2] = (dp[i - 1][7] + dp[i - 1][9]) % MOD
    dp[i][3] = (dp[i - 1][4] + dp[i - 1][8]) % MOD
    dp[i][4] = (dp[i - 1][0] + dp[i - 1][3] + dp[i - 1][9]) % MOD
    dp[i][6] = (dp[i - 1][0] + dp[i - 1][1] + dp[i - 1][7]) % MOD
    dp[i][7] = (dp[i - 1][2] + dp[i - 1][6]) % MOD
    dp[i][8] = (dp[i - 1][1] + dp[i - 1][3]) % MOD
    dp[i][9] = (dp[i - 1][2] + dp[i - 1][4]) % MOD

ans = 0
for i in range(10):
    ans += dp[n][i]
ans -= dp[n][0] + dp[n][8]

# Write the result to the output.txt file
with open('output.txt', 'w') as f:
    f.write(str(ans))

# Writing the time measurements to the standard output
end_time = time.perf_counter()
duration = end_time - start_time
print(f"Execution time: {duration}s")

# Writing memory usage to the standard output
memory_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss
print(f"Memory usage: {memory_usage}B")
```

Пояснение к решению:

Решение, опять же, является одним из вариантов использования дп.

Результат работы на примерах из текста к задаче:



Вывод по задаче:

Отработала навык применения дп.

Задача №20 Почти палиндром

- **Постановка задачи.** Слово называется палиндромом, если его первая буква совпадает с последней, вторая – с предпоследней и т.д. Например: «abba», «madam», «x».

  Для заданного числа K слово называется почти палиндромом, если в нем можно изменить не более K любых букв так, чтобы получился палиндром. Например, при $K = 2$ слова «$reactor$», «$kolobok$», «$madam$» являются почти палиндромами (подчеркнуты буквы, заменой которых можно получить палиндром).

  Подсловом данного слова являются все слова, получающиеся путем вычеркивания из данного нескольких (возможно, одной или нуля) первых букв и нескольких последних. Например, подсловами слова «cat» являются слова «c», «a», «t», «ca», «at» и само слово «cat» (а «ct» подсловом слова «cat» не является).

  Требуется для данного числа K определить, сколько подслов данного слова S являются почти палиндромами.

- **Формат входного файла (input.txt).** В первой строке входного файла вводятся два натуральных числа: N – длина слова и K. Во второй строке записано слово S, состоящее из N строчных английских букв.

- **Ограничения на входные данные.** $1 \leq N \leq 5000, 0 \leq K \leq N$.

- **Формат выходного файла (output.txt).** В выходной файл требуется вывести одно число – количество подслов слова S, являющихся почти палиндромами (для данного K).

- Ограничение по времени. 2 сек.

- Ограничение по памяти. 256 мб.

- Примеры:

input.txt	output.txt
5 1 abcde	12

input.txt	output.txt
3 3 aaa	6

Листинг кода:

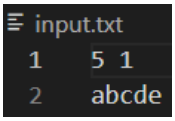
```
import resource
import time



# Starting time measuring
start_time = time.perf_counter()

# Reading from the input file
with open("input.txt", "r") as f:
    n, k = map(int, f.readline().split(" "))
    s = f.readline()

ans = 0

# Odd sized sub-words
for c in range(n):
```

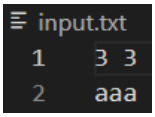
```
    ans += 1
    changes_required = 0
    for l in range(1, n):
        if c - l < 0 or c + l >= n:
            break
        if s[c - l] != s[c + l]:
            if changes_required != k:
                ans += 1
                changes_required += 1
            else:
                break
        else:
            ans += 1

# Even sized sub-words
for c in range(n):
    changes_required = 0
    for l in range(1, n):
        left = c - l + 1
        right = c + l
        if left < 0 or right >= n:
            break
        if s[left] != s[right]:
            if changes_required != k:
                ans += 1
                changes_required += 1
            else:
                break
        else:
            ans += 1

# Write the result to the output.txt file
with open('output.txt', 'w') as f:
    f.write(str(ans))

# Writing the time measurements to the standard output
end_time = time.perf_counter()
duration = end_time - start_time
print(f"Execution time: {duration}s")

# Writing memory usage to the standard output
memory_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss
print(f"Memory usage: {memory_usage}B")
```

Пояснение к решению:

Принцип простой: находим у слова середину, затем постепенно “наращиваем” его, определяя, является ли полученное палиндромом.

Результат работы на примерах из текста к задаче:





Вывод по задаче:

Так работает жадный алгоритм в более сложных задачах – концепция простая, легко освоить (люблю жадин))

Вывод по лабораторной работе:

В данной лабораторной работе я научилась работать с основными структурами данных (стек, очередь). А также научилась их применять и комбинировать в решении различных задач.