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

Отчет по лабораторной работе №0
по курсу «Алгоритмы и структуры данных»
Тема: Введение в Python

Выполнил:
Серова С. А. (фамилия имя)
К3140 (номер группы)

Проверила:
—-

Санкт-Петербург
2024 г.

Содержание отчета:

Задачи

Задачи:
Задача №1. Ввод - вывод
Текст задачи:

Вам необходимо выполнить 4 следующих задачи:

1. Задача $a + b$. В данной задаче требуется вычислить сумму двух заданных чисел. Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \leq a, b \leq 10^9$. Выход: единственное целое число — результат сложения $a + b$.

2. Задача $a + b^2$. В данной задаче требуется вычислить значение $a + b^2$. Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \leq a, b \leq 10^9$. Выход: единственное целое число — результат сложения $a + b^2$.

3. Выполните задачу $a + b$ с использованием файлов.

   - Имя входного файла: input.txt
   - Имя выходного файла: output.txt
   - Формат входного файла. Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \leq a, b \leq 10^9$.
   - Формат выходного файла. Выходной файл единственное целое число — результат сложения $a + b$.

   Примеры.

input.txt	12 25	130 61
output.txt	37	191

4. Выполните задачу $a+b^2$ с использованием файлов аналогично предыдущему пункту.

Листинг кода:

(1)

```
s = input().split()
a = int(s[0])
b = int(s[1])
if (a or b) <= 10**9 and (a or b) >= -10**9:
    print(a + b)
```

(2)

```
s = input().split()
a = int(s[0])
b = int(s[1])
if (a or b) <= 10**9 and (a or b) >= -10**9:
    print(a + b**2)
```

(3)

```
with open("input.txt", 'r') as file:
    data = file.read().replace('\n', '')
a = data.split()
num1 = int(a[0])
num2 = int(a[1])
result = num1 + num2
with open("output.txt", 'w') as file:
    file.write(str(result))
```

(4)

```
with open("input.txt", 'r') as file:
```

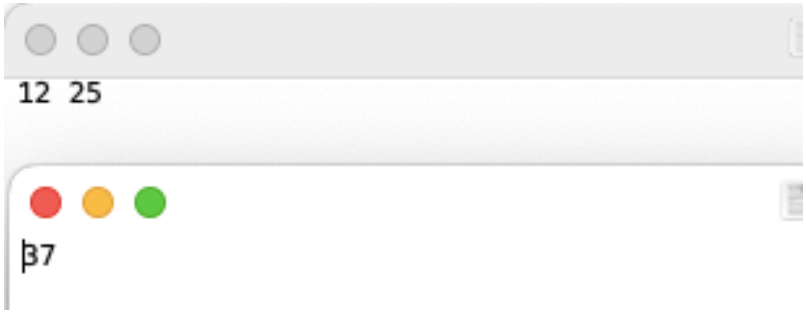
```
    data = file.read().replace('\n', '')
a = data.split()
num1 = int(a[0])
num2 = int(a[1])
result = num1 + (num2 ** 2)
with open("output.txt", 'w') as file:
    file.write(str(result))
```

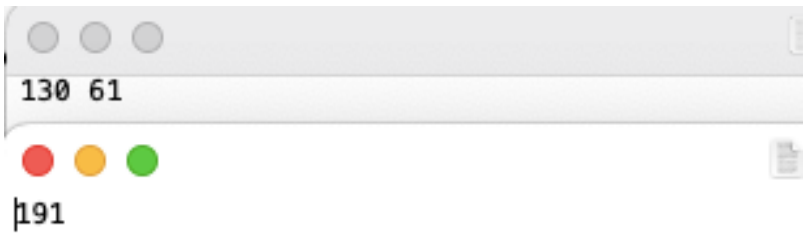
Текстовое объяснение решения:
Я считываю два числа и в пунктах 1 и 3 их складываю, а в пунктах 2 и 4 складываю первое число и квадрат второго.

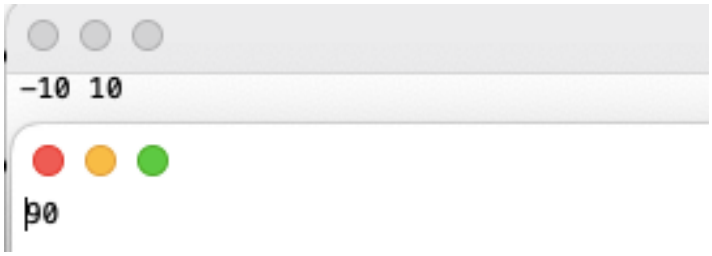
Результат работы кода на примерах из текста задачи:(скрины input output файлов)

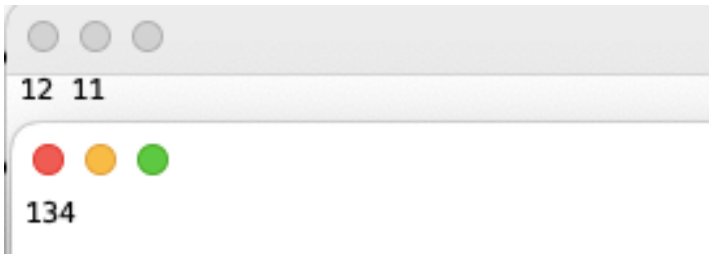
```
MacBook-Air-Serova:subtask_1 serovasofa$ python3 solution.py
10 2
12
MacBook-Air-Serova:subtask_1 serovasofa$ python3 solution.py
20 32
52
MacBook-Air-Serova:subtask_1 serovasofa$ python3 solution.py
-10 100
90
```

```
MacBook-Air-Serova:subtask_2 serovasofa$ python3 solution.py
10 3
19
MacBook-Air-Serova:subtask_2 serovasofa$ python3 solution.py
1 11
122
MacBook-Air-Serova:subtask_2 serovasofa$ python3 solution.py
3 -7
52
```





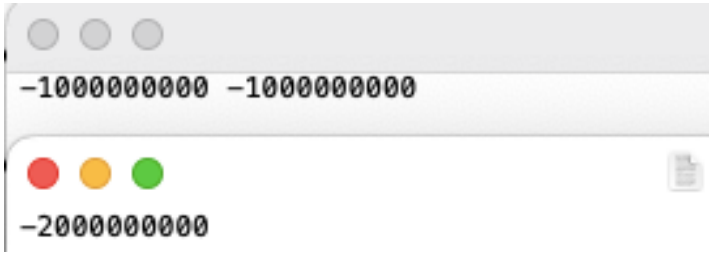


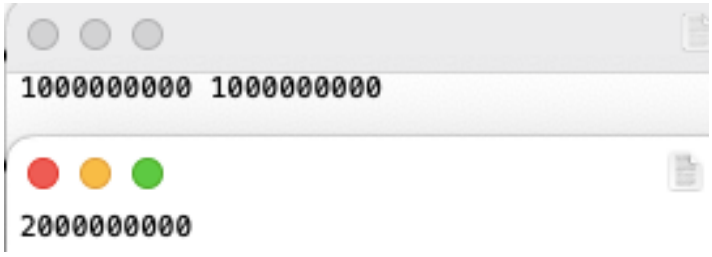


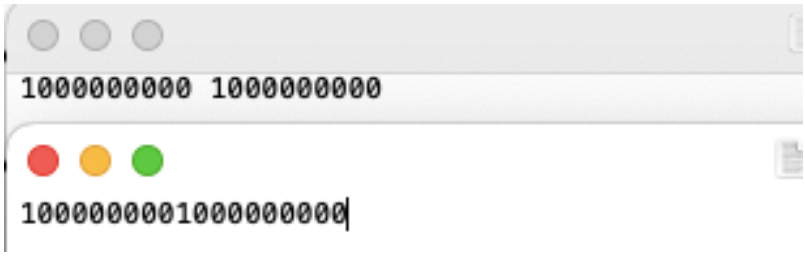
Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)

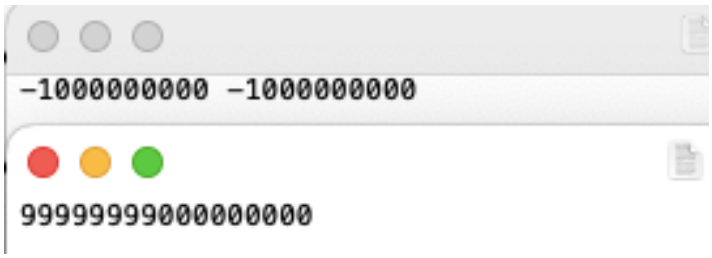
```
[MacBook-Air-Serova:subtask_1 serovasofa$ python3 solution.py
1000000000 1000000000
2000000000
[MacBook-Air-Serova:subtask_1 serovasofa$ python3 solution.py
-1000000000 -1000000000
-2000000000
```

```
[MacBook-Air-Serova:subtask_2 serovasofa$ python3 solution.py
1000000000 1000000000
1000000001000000000
[MacBook-Air-Serova:subtask_2 serovasofa$ python3 solution.py
-1000000000 -1000000000
999999999000000000
```









Нижняя граница диапазона значений входных данных из текста задачи:

-10**9

Верхняя граница диапазона значений входных данных из текста задачи:

10**9

Вывод по задаче:

Я научилась работать с файлами, складывать числа и возводить их в квадрат.

Задача №2. Число Фибоначчи

Текст задачи:

Определение последовательности Фибоначчи:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_i &= F_{i-1} + F_{i-2} \text{ для } i \geq 2. \end{aligned} \tag{1}$$

Таким образом, каждое число Фибоначчи представляет собой сумму двух предыдущих, что дает последовательность

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Ваша цель – разработать эффективный алгоритм для подсчета чисел Фибоначчи. Вам предлагается начальный код на Python, который содержит наивный рекурсивный алгоритм:

```
def calc_fib(n):
        if (n <= 1):
                return n

        return calc_fib(n - 1) + calc_fib(n - 2)

n = int(input())
print(calc_fib(n))
```

- Имя входного файла: input.txt
- Имя выходного файла: output.txt
- Формат входного файла. Целое число n. $0 \leq n \leq 45$.
- Формат выходного файла. Число F_n.

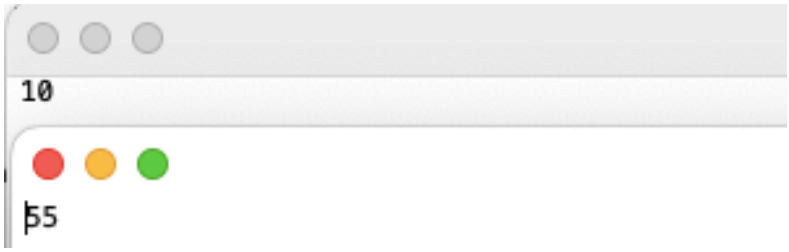
Листинг кода:

```
import time
t_start = time.perf_counter()
def calc_fib(n, fib={}):
    if n in fib :
        return fib[n]
    if n <=1:
        return n
    fib[n] = calc_fib(n - 1, fib) + calc_fib(n - 2, fib)
    return fib[n]

with open("input.txt", 'r') as file:
    data = file.read().replace('\n', '')
n = int(data)
with open("output.txt", 'w') as file:
    file.write(str(calc_fib(n)))
print("Время работы: %s секунд" % (time.perf_counter()-
t_start))
```

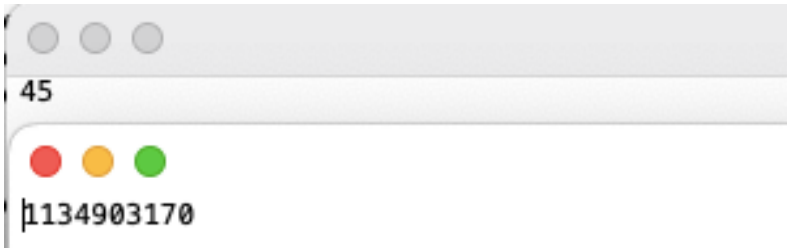
Текстовое объяснение решения:
Я создаю функцию, которая рассчитывает число Фибоначчи и записывает результат в

список для оптимизации алгоритма (из-за этого он вычисляет каждое число Фибоначчи не больше 1 раза).

Результат работы кода на примерах из текста задачи:(скрины input output файлов):



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)



Время выполнения

```
MacBook-Air-Serova:task2 serovasofa$ cd src
MacBook-Air-Serova:src serovasofa$ python3 solution.py
Время работы: 0.0017722939992381725 секунд
```

Нижняя граница диапазона значений входных данных из текста задачи

0

Верхняя граница диапазона значений входных данных из текста задачи

45

Вывод по задаче:

Я научилась оптимизировать функцию для вычисления числа Фибоначчи.

Задача №3. Еще про числа Фибоначчи

Текст задачи:

Определение последней цифры большого числа Фибоначчи. Числа Фибоначчи растут экспоненциально. Например,

$$F_{200} = 280571172992510140037611932413038677189525$$

Хранить такие суммы в массиве, и при этом подсчитывать сумму, будет достаточно долго. Найти последнюю цифру любого числа достаточно просто: `F mod 10`.

- Имя входного файла: input.txt
- Имя выходного файла: output.txt
- Формат входного файла. Целое число n. $0 \leq n \leq 10^7$.
- Формат выходного файла. Одна последняя цифра числа F_n.
- Пример 1.

input.txt	331
output.txt	9

$F_{331} = 668996615388005031531000081241745415306766517246774551964595292186469.$

- Пример 2.

input.txt	327305
output.txt	5

Это число не влезет в страницу, но оканчивается действительно на 5.

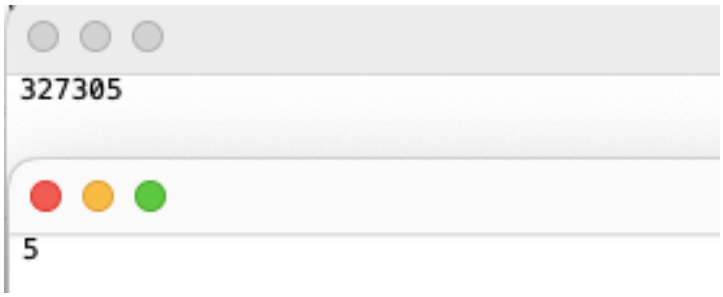
- Ограничение по времени: 5сек.
- Ограничение по памяти: 512 мб.

Листинг кода:

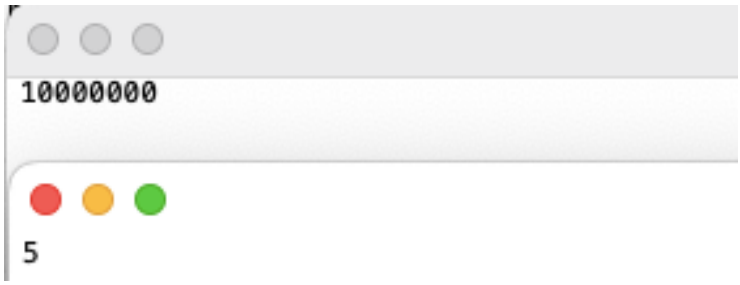
```
import sys
sys.setrecursionlimit(40000000)
import time
t_start = time.perf_counter()

def calc_fib(n, fib={}):
    if n in fib :
        return fib[n]
    if n <= 1:
        return n
    fib[n] = (calc_fib(n - 1, fib) + calc_fib(n - 2, fib)) %
10
    return fib[n]

with open("input.txt", 'r') as file:
    data = file.read().replace('\n', '')
n = int(data)
with open("output.txt", 'w') as file:
    file.write(str(calc_fib(n)))
print("Время работы: %s секунд" % (time.perf_counter()-
t_start))
```

Текстовое объяснение решения:
Я немного изменила алгоритм предыдущей задачи, теперь он вычисляет не само число Фибоначчи, а последнюю цифру (остаток при делении на 10). А также, увеличила максимальную длину рекурсии.

Результат работы кода на примерах из текста задачи:(скрины input output файлов):



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)



Время выполнения

```
MacBook-Air-Serova:task3 serovasofa$ cd src
MacBook-Air-Serova:src serovasofa$ python3 solution.py
Время работы: 0.43801792199883494 секунд
```

Нижняя граница диапазона значений входных данных из текста задачи

0

Верхняя граница диапазона значений входных данных из текста задачи

10000000

Вывод по задаче:

Я научилась оптимизировать функцию для вычисления последней цифры числа Фибоначчи.

Задача №3. Еще про числа Фибоначчи

Текст задачи:

Определение последней цифры большого числа Фибоначчи. Числа Фибоначчи растут экспоненциально. Например,

$$F_{200} = 280571172992510140037611932413038677189525$$

Хранить такие суммы в массиве, и при этом подсчитывать сумму, будет достаточно долго. Найти последнюю цифру любого числа достаточно просто: `F mod 10`.

- Имя входного файла: input.txt
- Имя выходного файла: output.txt
- Формат входного файла. Целое число n. $0 \leq n \leq 10^7$.
- Формат выходного файла. Одна последняя цифра числа F_n.
- Пример 1.

input.txt	331
output.txt	9

$F_{331} = 668996615388005031531000081241745415306766517246774551964595292186469.$

- Пример 2.

input.txt	327305
output.txt	5

Это число не влезет в страницу, но оканчивается действительно на 5.

- Ограничение по времени: 5сек.
- Ограничение по памяти: 512 мб.

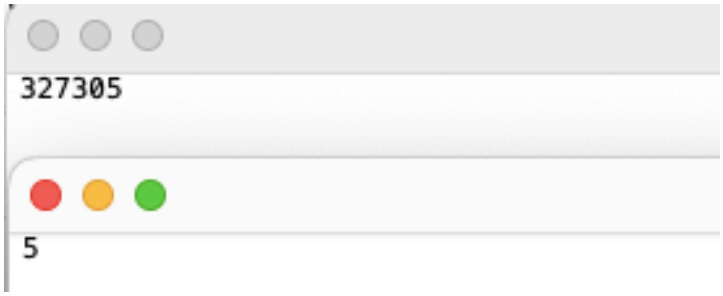
Листинг кода:

```
import sys
sys.setrecursionlimit(40000000)
import time
t_start = time.perf_counter()

def calc_fib(n, fib={}):
    if n in fib :
        return fib[n]
    if n <= 1:
        return n
    fib[n] = (calc_fib(n - 1, fib) + calc_fib(n - 2, fib)) %
10
    return fib[n]

with open("input.txt", 'r') as file:
    data = file.read().replace('\n', '')
n = int(data)
with open("output.txt", 'w') as file:
```

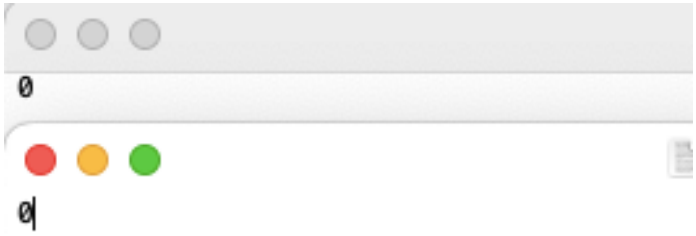
```
    file.write(str(calc_fib(n)))
print("Время работы: %s секунд" % (time.perf_counter()-
t_start))
```

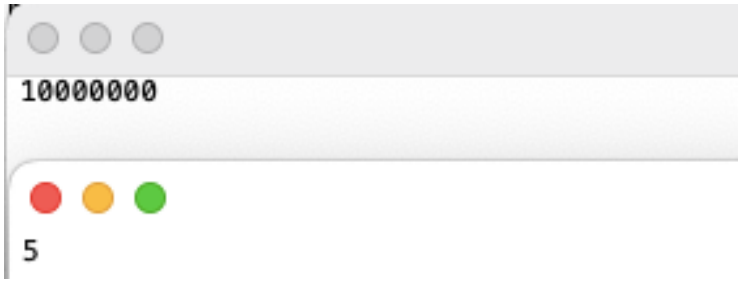
Текстовое объяснение решения:
Я немного изменила алгоритм предыдущей задачи, теперь он вычисляет не само число Фибоначчи, а последнюю цифру (остаток при делении на 10). А также, увеличила максимальную длину рекурсии.

Результат работы кода на примерах из текста задачи:(скрины input output файлов):



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов)





Время выполнения

```
[MacBook-Air-Serova:task3 serovasofa$ cd src
[MacBook-Air-Serova:src serovasofa$ python3 solution.py
 Время работы: 0.43801792199883494 секунд
```

Нижняя граница диапазона значений входных данных из текста задачи

0

Верхняя граница диапазона значений входных данных из текста задачи

10000000

Вывод по задаче:

Я научилась оптимизировать функцию для вычисления последней цифры числа Фибоначчи.

Задача №4. Тестирование ваших алгоритмов

Текст задачи:

Задача: вам необходимо протестировать время выполнения вашего алгоритма в *Задании 2* и *Задании 3*.

Дополнительно: вы можете протестировать объем используемой памяти при выполнении вашего алгоритма.

Листинг кода:

```
import time
t_start = time.perf_counter()


print("Время работы: %s секунд" % (time.perf_counter()-
t_start))
```

Текстовое объяснение решения:
Я добавила таймер, который считывает время выполнения кода для заданий 2 и 3.

Вывод по задаче:

Я научилась выводить время работы кода, чтобы можно было проще оптимизировать алгоритм.

Вывод

При выполнении лабораторной я научилась базовым навыкам работы с кодом в Python: операциям над числами, вычислением времени работы кода, использовании функций и поиску решений по оптимизации алгоритмов.