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

Выполнил:
Макунина Арина
К32421

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Текст заданий

Задание 1

1.1. Задача a + b. Необходимо вычислить сумму двух заданных чисел.

1.2. Задача a + b^2. Необходимо вычислить значение a + b^2.

1.3. Задача a + b с использованием входных/выходных файлов. Необходимо вычислить сумму двух заданных чисел используя внешние файлы.

1.4. Задача a + b^2 с использованием входных/выходных файлов. Необходимо вычислить сумму двух заданных чисел используя внешние файлы.

Задание 2

Число Фибоначчи. Требуется разработать эффективный алгоритм для подсчета чисел Фибоначчи с использованием внешних входных/выходных файлов. На вход поступает файл с порядковым номером необходимого числа Фибоначчи, на выходе необходимо получить файл с этим числом.

Задание 3

Необходимо определить последнюю цифру какого-либо большого числа Фибоначчи по порядковому номеру этого числа Фибоначчи с помощью эффективного алгоритма. При этом ограничение по времени работы алгоритма: 5сек; ограничение по памяти: 512 мб.

Задание 4

Необходимо протестировать время выполнения и объем используемой памяти алгоритма в Задании 2 и Задании 3.

Решение заданий

Задание 1

1.1. Решение данной задачи уместилось в одну строку: сперва мы получаем значение строкового типа str с помощью стандартного ввода input(), учитывая пробел с помощью метода split(). Функция map позволяет выполнить преобразование из строкового типа данных в целочисленный int. Далее получение значения суммируются с помощью функции sum и выводятся на консоль.

```
print(sum(map(int, input('Enter two numbers separated by a space: ').split())))
```

Результат работы кода на примерах из текста задачи:

```
Enter two numbers separated by a space: 12 25
37
```

```
Enter two numbers separated by a space: 130 61
191
```

1.2. Данная задача отличается от предыдущей дополнительной математической операцией. В первой строке с помощью пользовательского ввода input() мы получаем значение строкового типа str, а также используем метод split() для учёта пробела. С помощью функции map применяем преобразование к типу int к каждому из полученных двух значений a и b. Далее выводим на консоль сумму чисел a и квадрат b.

```
a, b = map(int, input('Enter the numbers a and b: ').split())
print(a + (b ** 2))
```

Результат работы кода на примерах из текста задачи:

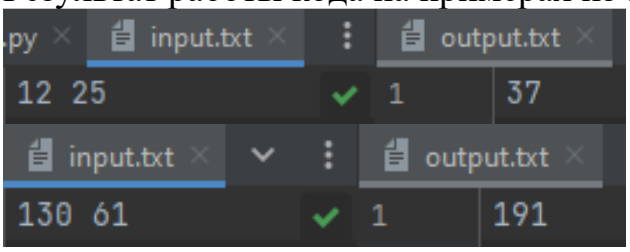
```
Enter the numbers a and b: 12 25
637
```

```
Enter the numbers a and b: 130 61
3851
```

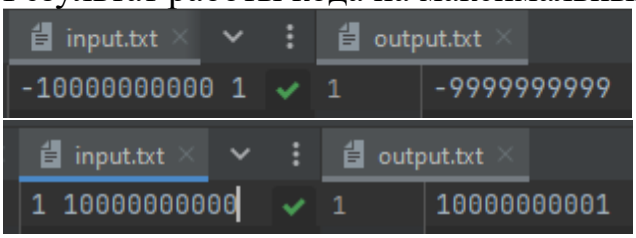
1.3. Для начала мы открываем файл с числами input.txt, находящийся в папке проекта, с помощью функции open(). Он открывается для чтения по умолчанию. Создаём объект файла в переменной f. Далее мы считываем строку из файла с помощью метода readline() и следом учитываем пробел в строке с помощью метода split(). Приводим полученный строковый тип данных к int с помощью функции map() и присваиваем значения к переменным a и b соответственно. Закрываем файл с вводными данными с помощью метода close. После открываем файл output.txt для записи. Для этого используем режим ‘w’. Запись происходит с помощью метода write(), где в аргумент идёт математическая операция, преобразованная к строковому типу. Программа заканчивается закрытием файла.

```
f = open('input.txt')
a, b = map(int, f.readline().split())
f.close()
w = open('output.txt', 'w')
w.write(str(a + b))
w.close()
```

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



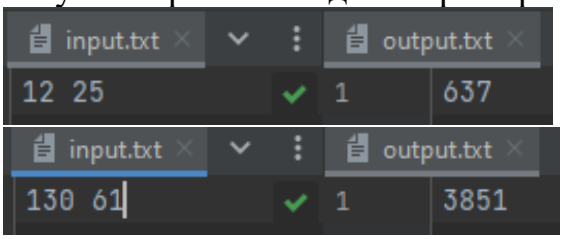
	Время выполнения, с	Максимальные затраты памяти, МБ
Нижняя граница диапазона	0.0013076000000000006	0.0172271728515625

значений входных данных из текста задачи		
Пример из задачи	0.0010282000000000069	0.017210006713867188
Пример из задачи	0.0010172000000000098	0.0172119140625
Верхняя граница диапазона значений входных данных из текста задачи	0.0011001000000000066	0.017225265502929688

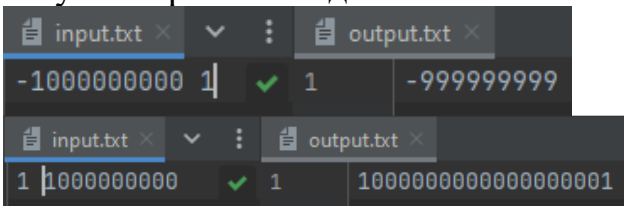
1.4. Данная задача решается точно так же, как и задача 1.3. Отличается только математическая операция: число a складывается с квадратом числа b.

```
f = open('input.txt')
a, b = map(int, f.readline().split())
f.close()
w = open('output.txt', 'w')
w.write(str(a + (b ** 2)))
w.close()
```

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения, с	Максимальные затраты памяти, МБ

Нижняя граница диапазона значений входных данных из текста задачи	0.002149200000000004	0.018770217895507812
Пример из задачи	0.0024697999999999942	0.01878833770751953
Пример из задачи	0.002555799999999997	0.018790245056152344
Верхняя граница диапазона значений входных данных из текста задачи	0.0019370999999999972	0.01883220672607422

Вывод по задаче: интересно…вспомнила как происходит работа с файлом на python!

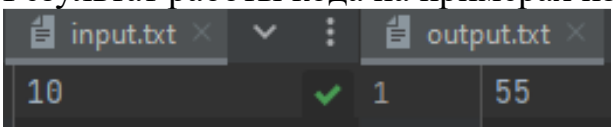
Задание 2

Для начала открываем файл с входными данными и считываем переменную n. Изначально переменные f1 и f2 первые числа последовательности, но далее они служат для хранения чисел последовательности, которые необходимо просуммировать. Переменная sum_f необходима как раз таки для суммы переменных f1 и f2. Мы должны рассмотреть варианты при n=1, n=2, n=0 для корректной работы программы. Сам алгоритм подсчёта чисел Фибоначчи реализован таким образом: суммируем первые числа последовательности, далее f1 присваивается значение f2, а f2 присваивается значение sum_f. Соответственно, f_sum снова можно использовать для подсчета суммы переменных f1 и f2 – процесс пошёл! Далее полученное число записывается в файл input.txt.

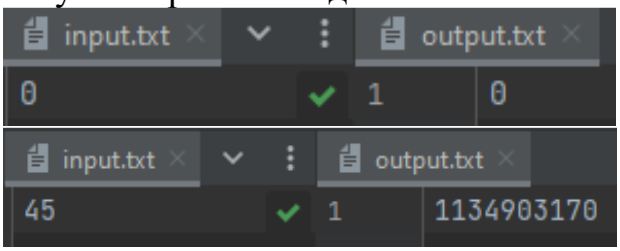
```
f = open('input.txt')
n = int(f.readline())
f.close()
f1, f2 = 0, 1
if n == 1:
```

```
    sum_f = f1
elif n == 2:
    sum_f = f2
elif n == 0:
    sum_f = 0
else:
    for i in range(n - 1):
        sum_f = f1 + f2
        f1 = f2
        f2 = sum_f
w = open('output.txt', 'w')
w.write(str(sum_f))
w.close()
```

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения, с	Максимальные затраты памяти, МБ
Нижняя граница диапазона значений входных данных из текста задачи	0.0011965999999999921	0.017154693603515625
Пример из задачи	0.001483299999999993	0.01720428466796875
Верхняя граница диапазона значений входных	0.0016256000000000048	0.01720428466796875

данных из текста задачи		

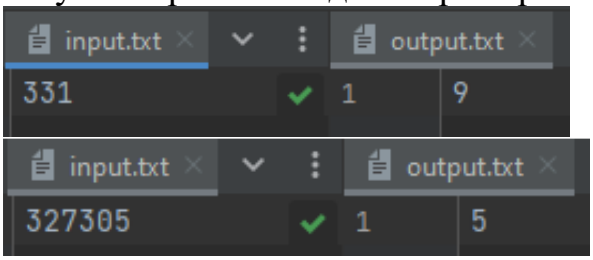
Вывод по задаче: рекурсия – не лучший выход в оптимизации кода. Нужно меньше хранить данных в итерациях.

Задание 3

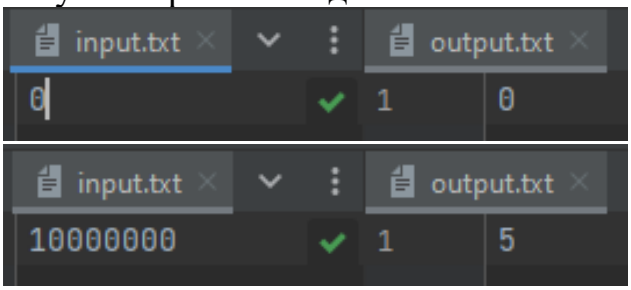
Код программы идентичен задаче 2, но по тексту данного задания нам необходимо сохранять лишь последнюю цифру числа Фибоначчи, что мы делаем с помощью операции «%10» (остаток от деления). По порядковому номеру можно узнать не все число Фибоначчи, а лишь его последнюю цифру.

```
f = open('input.txt')
n = int(f.readline())
f.close()
f1, f2 = 0, 1
if n == 1:
    sum_f = f1
elif n == 2:
    sum_f = f2
elif n == 0:
    sum_f = 0
else:
    for i in range(n - 1):
        sum_f = f1 + f2
        f1 = f2
        f2 = sum_f%10
w = open('output.txt', 'w')
w.write(str(sum_f%10))
w.close()
```

Результат работы кода на примерах из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения, с	Максимальные затраты памяти, МБ
Нижняя граница диапазона значений входных данных из текста задачи	0.0012137000000000012	0.017154693603515625
Пример из задачи	0.0016802999999999957	0.017206192016601562
Пример из задачи	0.30828279999999997	0.0172119140625
Верхняя граница диапазона значений входных данных из текста задачи	3. 533645564632	0.017215728759765625

Вывод по задаче: хранить целиком числа, когда нужна только последняя цифра не имеет смысла.

Задание 4

Тестирование времени происходит через импорт внутренней библиотеки time, а памяти – tracemalloc. Далее задаём начало отсчёта с помощью вводных строчек:

```
t_start = time.perf_counter()
tracemalloc.start()
```

А вывод:

```
print("Time: %s s " % (time.perf_counter() -
t_start))
print("Max memory ",
tracemalloc.get_traced_memory()[1] / 2 ** 20,
"mb")
```

Вывод по задаче: круто, что кто-то написал библиотеки, чтобы мы ими пользовались!

Вывод

В ходе выполнения данной лабораторной работы я вспомнила и применила на практике основы алгоритмов на языке программирования Python. Посмотрела на примере, почему рекурсиия не самый оптимальный вариант в решение задач с помощью кода, способы работы с файлами на ввод и вывод, а также на консоль. Научилась регистрировать время выполнения программы и затраты памяти с помощью импортирования библиотек. Это помогло мне проанализировать эффективность и оптимальность моих алгоритмов.