

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

Отчет по лабораторной работе №6 по
курсу «Алгоритмы и структуры
данных»

Вариант 7

Выполнила:
Пожидаева Е.Р.

Санкт-Петербург

2024 г.

Оглавление

Задача №5. Выборы в США	3
Задача №6. Фибоначчи возвращается.....	6
Задача №7. Драгоценные камни.....	8
Задача №8. Почти интерактивная хеш-таблица.....	11

Задача №5. Выборы в США

Текст задачи

Как известно, в США президент выбирается не прямым голосованием, а путем двухуровневого голосования. Сначала проводятся выборы в каждом штате и определяется победитель выборов в данном штате. Затем проводятся государственные выборы: на этих выборах каждый штат имеет определенное число голосов — число выборщиков от этого штата. На практике, все выборщики от штата голосуют в соответствии с результатами голосования внутри штата, то есть на заключительной стадии выборов в голосовании участвуют штаты, имеющие различное число голосов. Вам известно за кого проголосовал каждый штат и сколько голосов было отдано данным штатом. Подведите итоги выборов: для каждого из участника голосования определите число отданных за него голосов.

Листинг кода.

```
with open("input.txt") as f:
    lines = f.readlines()

votes = {}
for line in lines:
    candidate, count =
line.strip().split(" ")
    count = int(count)
    if candidate in votes:
        votes[candidate] += count
    else:
        votes[candidate] = count

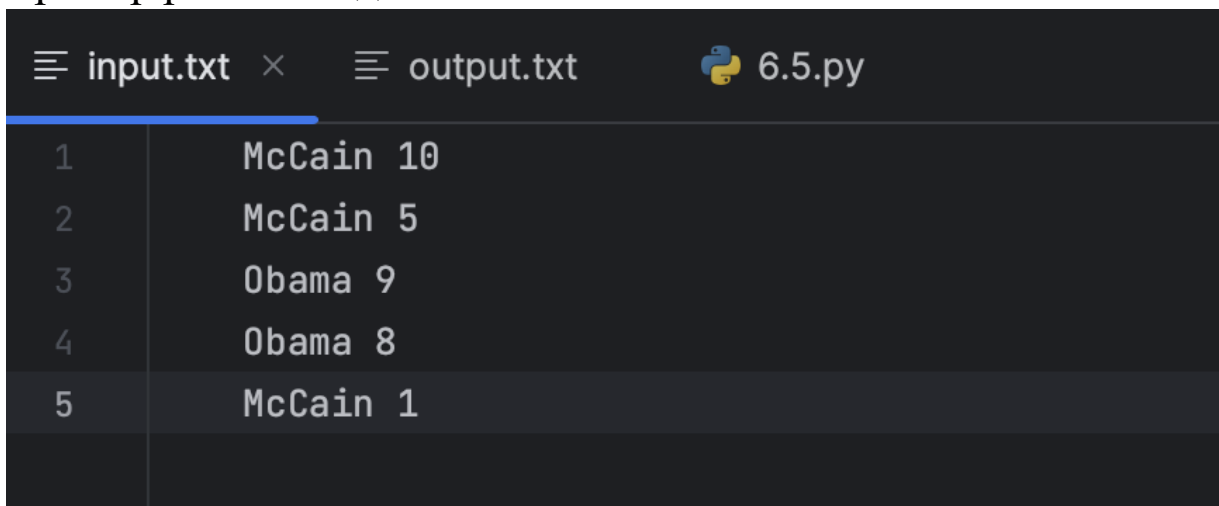
with open("output.txt", "w") as file:
    for candidate in
sorted(votes.keys()):
        count = votes[candidate]
        file.write(f"{candidate}
{count}\n")
```

Текстовое объяснение кода:

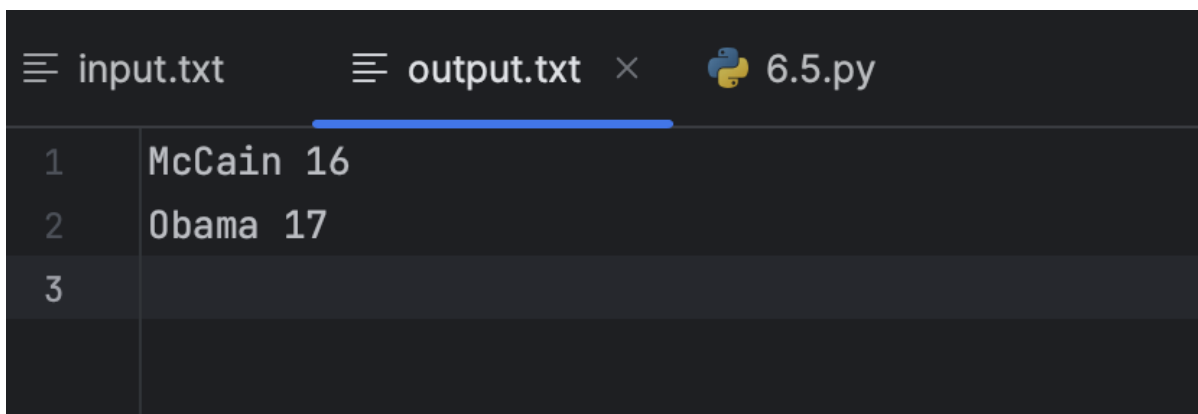
Открывается файл `input` и считываются входные данные. Создается пустой словарь `votes`, в котором ключами будут являться имена кандидатов, а

значениями - количество голосов за каждого кандидата. Если имя кандидата уже присутствует в словаре `votes`, то увеличивает значение голосов для этого кандидата на `count`. Если же имя кандидата отсутствует в словаре, то добавляет его в словарь с начальным количеством голосов `count`. Открывает файл "output" в режиме записи и записывает в него отсортированные по ключу пары ключ-значение из словаря `votes`. Каждая пара ключ-значение записывается на отдельной строке в формате "имя кандидата количество голосов".

Пример работы кода:



```
input.txt × output.txt 6.5.py
1 McCain 10
2 McCain 5
3 Obama 9
4 Obama 8
5 McCain 1
```



```
input.txt output.txt × 6.5.py
1 McCain 16
2 Obama 17
3
```

	Время выполнения (сек)	Затраты памяти (Мб)
Пример из задачи	0.00130701065063476	6193152

Задача №6. Фибоначчи возвращается

Текст задачи

Вам дается последовательность чисел. Для каждого числа определите, является ли оно числом Фибоначчи. Напомним, что числа Фибоначчи определяются, например, так:

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

Листинг кода.

```
def fibbonachi(el):
    a, b = 0, 1
    while a < el:
        a, b = b, a+b
    if a == el or b == el:
        return True
    return False

with open('input.txt') as f:
    _ = int(f.readline())
    digits = list(map(int, f.readlines()))
answer = []
for element in digits:
    if fibbonachi(element):
        answer.append("Yes")
    else:
        answer.append("No")

with open('output.txt', "w") as f:
    f.write("\n".join(answer))
```

Текстовое объяснение решения

Функция `fibbonachi` принимает параметр `el`. Переменным `a` и `b` присваиваем значения 0 и 1 соответственно. В цикле происходит переопределение значений переменных `a` и `b`: `a` присваивается значение `b`, а `b` присваивается сумма `a` и `b`. Если `a` равно `el` или `b` равно `el`, то функция возвращает `True`.

Если ни одно из вышеперечисленных условий не выполняется, функция возвращает False. Для каждого элемента в списке digits выполняется проверка через функцию fibonacci. Если результат True, то в список answer добавляется "Yes", если False, то "No".

Результат работы кода на примере из текста задачи

```

6.6.py  input.txt  output.txt
1      8
2      1
3      2
4      3
5      4
6      5
7      6
8      7
9      8
  
```

```

6.6.py  input.txt  output.txt
1      Yes
2      Yes
3      Yes
4      No
5      Yes
6      No
7      No
8      Yes
  
```

	Время выполнения (сек)	Затраты памяти (Мб)
--	------------------------------	---------------------

Пример из задачи	0.001129150390625	6193152
------------------	-------------------	---------

Задача №7. Драгоценные камни

В одной далекой восточной стране до сих пор по пустыням ходят караваны верблюдов, с помощью которых купцы перевозят пряности, драгоценности и дорогие ткани. Разумеется, основная цель купцов состоит в том, чтобы подороже продать имеющийся у них товар. Недавно один из караванов прибыл во дворец одного могущественного шаха.

Купцы хотят продать шаху n драгоценных камней, которые они привезли с собой. Для этого они выкладывают их перед шахом в ряд, после чего шах оценивает эти камни и принимает решение о том, купит он их или нет. Видов драгоценных камней на Востоке известно не очень много всего 26, поэтому мы будем обозначать виды камней с помощью строчных букв латинского алфавита. Шах обычно оценивает камни следующим образом. Он заранее определил несколько упорядоченных пар типов камней: $(a_1, b_1), (a_2, b_2), \dots, (a_k, b_k)$. Эти пары он называет красивыми, их множество мы обозначим как P . Теперь представим ряд камней, которые продают купцы, в виде строки S длины n из строчных букв латинского алфавита. Шах считает число таких пар (i, j) , что $1 \leq i < j \leq n$, а камни S_i и S_j образуют красивую пару, то есть существует такое число $1 \leq q \leq k$, что $S_i = a_q$ и $S_j = b_q$.

```
def count_beautiful_pairs(S, P):
    beautiful_pairs = 0
    for i in range(len(S)):
        for j in range(i + 1, len(S)):
            if (S[i] + S[j]) in P:
                beautiful_pairs += 1
    return beautiful_pairs
```

```
f = open('input.txt')
s1 = f.readline().split()
n = int(s1[0])
k = int(s1[1])
kamni = f.readline()
good_kamni = []
```



```

for i in range(k):

    para = f.readline().rstrip()

    good_kamni.append(para)

with open("output.txt", "w") as file:

    file.write(str(count_beautiful_pairs(kamni, good_kamni)))

```

Текстовое объяснение решения

Файл открывается, из него считывается первая строка, содержащая число камней, привезенных купцами и число красивых пар. Эта строка делится на переменные n и k . Строка с привезенными камнями записывается в переменную `kamni`. Создается пустой массив `good_kamni`. Запускается цикл, длящийся k раз. Считывается строка с красивой парой и добавляется в массив `good_kamni`. Далее открывается файл `output` для записи, в него записывается результат функции `count_beautiful_pairs`, считающий количество красивых пар в привезенных камнях.


Результат работы кода на примерах


The screenshot shows a code editor with three tabs: '6.7.py', 'input.txt', and 'output.txt'. The 'input.txt' tab is active and displays the following content:


1	7 3
2	abacaba
3	ab
4	ac
5	bb


The screenshot shows the same code editor with the 'output.txt' tab active. It displays the following content:

1	7
---	---

<div>  6.7.py <div> <div>≡ input.txt</div> <div>×</div> <div>≡ output.txt</div> </div> </div>	
1	7 1
2	<u>abacaba</u>
3	aa

<div>  6.7.py <div> <div>≡ input.txt</div> <div>≡ output.txt</div> <div>×</div> </div> </div>	
1	6

<div>  6.7.py <div> <div>≡ input.txt</div> <div>×</div> <div>≡ output.txt</div> </div> </div>	
1	2 2
2	aa
3	aa

<div>  6.7.py <div> <div>≡ input.txt</div> <div>≡ output.txt</div> <div>×</div> </div> </div>	
1	1

	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.006492137908935547	655849
Пример 1	0.007873780250549316	673593
Пример 2	0.008733682737238738	656476

Вывод: Код способен находить количество красивых пар.

Задача №8. Почти интерактивная хеш-таблица

Текст задачи.

В данной задаче у Вас не будет проблем ни с вводом, ни с выводом. Просто реализуйте быструю хеш-таблицу.

В этой хеш-таблице будут храниться целые числа из диапазона $[0; 10^{15} - 1]$. Требуется поддерживать добавление числа x и проверку того, есть ли в таблице число x . Числа, с которыми будет работать таблица, генерируются следующим образом. Пусть имеется четыре целых числа N, X, A, B такие что:

- $1 \leq N \leq 10^7$
- $1 \leq X \leq 10^{15}$
- $1 \leq A \leq 10^3$
- $1 \leq B \leq 10^{15}$

Требуется N раз выполнить следующую последовательность операций:

- Если X содержится в таблице, то установить $A \leftarrow (A + A_C) \bmod 10^3$, $B \leftarrow (B + B_C) \bmod 10^{15}$.
- Если X не содержится в таблице, то добавить X в таблицу и установить $A \leftarrow (A + A_D) \bmod 10^3$, $B \leftarrow (B + B_D) \bmod 10^{15}$.
- Установить $X \leftarrow (X \cdot A + B) \bmod 10^{15}$.

Начальные значения X, A и B , а также N, A_C, B_C, A_D и B_D даны во входном файле. Выведите значения X, A и B после окончания работы.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится четыре целых числа N, X, A, B . Во второй строке содержится еще четыре целых числа A_C, B_C, A_D и B_D такие что $0 \leq A_C, A_D < 10^3$, $0 \leq B_C, B_D < 10^{15}$.
- **Формат выходного файла (output.txt).** Выведите значения X, A и B после окончания работы.

Листинг кода.

```
class HashTable:
    def __init__(self):
        self.table = set()

    def add_key(self, key):
        self.table.add(key)

    def check_key(self, key):
        if key in self.table:
            return True
        return False

with open('input.txt', 'r', encoding='utf-8') as file_input, \
    open('output.txt', 'w', encoding='utf-8') as file_output:

    N, X, A, B = map(int, file_input.readline().split())
    AC, BC, AD, BD = map(int, file_input.readline().split())

    hash_table = HashTable()

    for _ in range(N):
        if hash_table.check_key(X):
            A = (A + AC) % 10**3
            B = (B + BC) % 10**15
        else:
            hash_table.add_key(X)
            A = (A + AD) % 10**3
            B = (B + BD) % 10**15

        X = (X * A + B) % 10**15

    print(X, A, B, file=file_output)
```

Текстовое объяснение решения.

Предоставленный код реализует структуру данных HashSet, которая используется для эффективной проверки наличия ключа в наборе.

Определяется класс HashTable. У него есть набор self.table, который используется для хранения ключей.

Метод __init__ инициализирует набор. Метод add_key добавляет ключ в набор. Метод check_key проверяет, существует ли ключ в наборе. Если да, то метод возвращает True. В противном случае он возвращает False.

Блок with считывает входной файл 'input.txt', содержащий количество операций (n) и сами операции. Код создает объект HashTable hash_table и обрабатывает каждую операцию в зависимости от типа операции ('A', 'D' или '?').

Если тип операции 'A', код вызывает метод add_key для добавления ключа в набор. Если тип операции - 'D', вызывается

метод `check_key`, чтобы проверить, существует ли ключ в наборе. Результат этой проверки записывается в выходной файл 'output.txt'. Если тип операции равен '?', код вызывает метод `check_key`, чтобы проверить, существует ли ключ в наборе. Результат этой проверки записывается в выходной файл 'output.txt'. К концу работы программы набор `hash_table` содержит ключи, полученные в результате выполнения всех операций, а файл 'output.txt' содержит результаты операций '?'.

Результат работы кода на примерах из текста задачи:

6.8.py	input.txt	output.txt
1	4 0 0 0	
2	1 1 0 0	

6.8.py	input.txt	output.txt
1	3 1 1	
2		

6.8.py	input.txt	output.txt
1	1000000 1000000 1000000 1000000	
2	1000000 1000000 1000000 1000000	

6.8.py	input.txt	output.txt
1	170 84 100	
2		

	Время выполнения (сек)	Затраты памяти (Мб)
Верхняя граница диапазона значений входных данных из текста задачи	0.00047478922009876	6082560

Вывод по задаче: Разобралась с работой Хэш таблиц.