

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

Отчет по лабораторной работе №6
по курсу «Алгоритмы и структуры данных»
Тема: Хеширование. Хеш-таблицы.

Выполнил:
Волжева М.И.
К3141

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Содержание отчета	2
Задачи по варианту	3
Задача №1. Множество	3
Задача №2. Телефонная книга	6
Задача №5. Выборы в США	9
Вывод	12

Задачи по варианту

Задача №1. Множество

Реализуйте множество с операциями «добавление ключа», «удаление ключа», «проверка существования ключа».

1)A x – добавить элемент x в множество. Если элемент уже есть в множестве, то ничего делать не надо.

2)D x – удалить элемент x. Если элемента x нет, то ничего делать не надо.

3)? x – если ключ x есть в множестве, выведите «Y», если нет, то выведите «N».

- Формат входного файла (input.txt) В первой строке входного файла находится строго положительное целое число операций N, не превышающее $5 \cdot 10^5$. В каждой из последующих N строк находится одна из операций:
- Формат выходного файла (output.txt) Выведите последовательно результат выполнения всех операций «?». Следуйте формату выходного файла из примера.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

Input.txt	8 A 2 A 5 A 3 ? 2 ? 4 A 2 D 2 ? 2
Output.txt	Y N N

Листинг кода:

```
import time
import os, psutil

def add_number(book, number):
```

```

book.add(number)

def find_number(book, number):
    if number in book:
        return "Y"
    else:
        return "N"

def del_number(book, number):
    if number in book:
        book.remove(number)

t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("1_input.txt")
m = open("1_output.txt", "w")
count = int(f.readline())
book = set()
for i in range(count):
    string = f.readline()
    elements = list(map(str, string.split()))
    if elements[0] == "A":
        add_number(book, int(elements[1]))
    if elements[0] == "?":
        m.write(str(find_number(book, int(elements[1]))) + "\n")
    if elements[0] == "D":
        del_number(book, int(elements[1]))

f.close()
m.close()

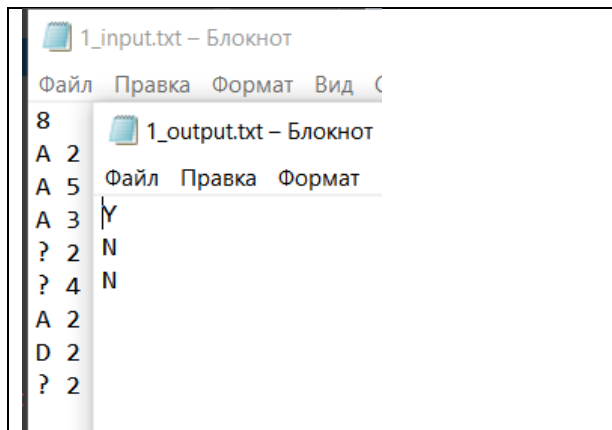
print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")

```

Текстовое объяснение решения:

Сначала мы читаем данные из файла и записываем все данные в set – структура данных, в которой сохраняются только неповторяющиеся элементы. Далее были написаны 3 функции, выполняющие соответствующие операции с set, - del_number, find_number, add_number. В конце результат функции find_number выводится в файл.

Результат работы кода на примерах из текста задачи:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Пример 1 из задачи	0.0005359000000000058 second	13.84765625 mb

Вывод по задаче: Мы узнали, как работает структура данных set и научились добавлять, искать, удалять элементы в нём.

Задача №2. Телефонная книга

В этой задаче ваша цель - реализовать простой менеджер телефонной книги. Он должен уметь обрабатывать следующие типы пользовательских запросов:

1)add number name – это команда означает, что пользователь добавляет в телефонную книгу человека с именем name и номером телефона number. Если пользователь с таким номером уже существует, то ваш менеджер должен перезаписать соответствующее имя.

2)del number – означает, что менеджер должен удалить человека с номером из телефонной книги. Если такого человека нет, то он должен просто игнорировать запрос.

3)find number – означает, что пользователь ищет человека с номером телефона number. Менеджер должен ответить соответствующим именем или строкой «not found» (без кавычек), если такого человека в книге нет.

- Формат входного файла (input.txt) В первой строке входного файла содержится число N ($1 \leq N \leq 10^5$) – количество запросов. Далее следуют N строк, каждая из которых содержит один запрос в формате, описанном выше.
- Формат выходного файла (output.txt) Выведите результат каждого поискового запроса find – имя, соответствующее номеру телефона, или «not found» (без кавычек), если в телефонной книге нет человека с таким номером телефона. Выведите по одному результату в каждой строке в том же порядке, как были заданы запросы типа find во входных данных.
- Ограничение по времени. 6 сек.
- Ограничение по памяти. 512 мб.
- Пример:

Input.txt	12 add 911 police add 76213 Mom add 17239 Bob find 76213 find 910 find 911 del 910 del 911 find 911	8 find 3839442 add 123456 me add 0 granny find 0 find 123456 del 0 del 0 find 0
-----------	--	---

	find 76213 add 76213 daddy find 762130	
Output.txt	Mom not found police not found Mom daddy	not found granny me not found

Листинг кода:

```
import time
import os, psutil

def add_number(book, number, name):
    book[number] = name

def find_number(book, number):
    if number in book:
        return book[number]
    else:
        return "not found"

def del_number(book, number):
    if number in book:
        book.pop(number)

t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("2_input.txt")
m = open("2_output.txt", "w")
count = int(f.readline())
book = {}
for i in range(count):
    string = f.readline()
    elements = list(map(str, string.split()))
    if elements[0] == "add":
        add_number(book, int(elements[1]), elements[2])
    if elements[0] == "find":
        m.write(str(find_number(book, int(elements[1]))) + "\n")
    if elements[0] == "del":
        del_number(book, int(elements[1]))

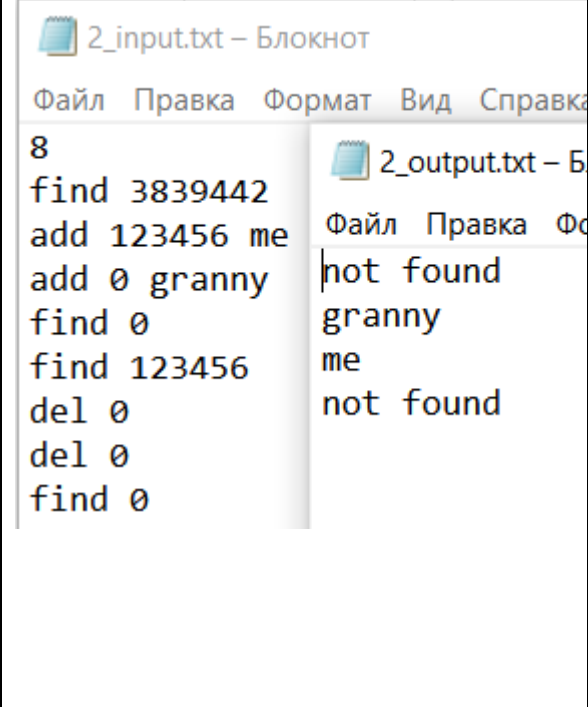
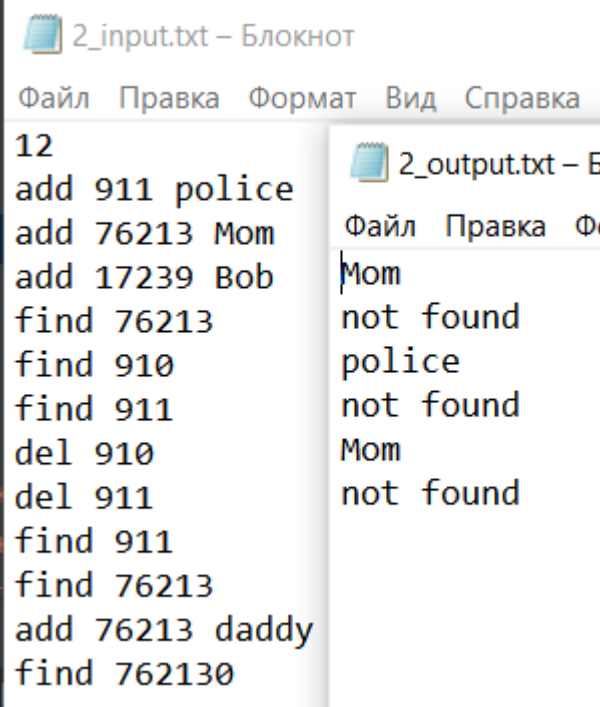
f.close()
m.close()

print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Сначала мы читаем данные из файла и записываем все данные в словарь (хеш-таблицу). Ключи – номера телефонов, значения – имена. Далее были написаны 3 функции, выполняющие соответствующие операции: `del_number` , `find_number`, `add_number`. В конце результат функции `find_number` выводится в файл.

Результат работы кода на примерах из текста задачи:

 <pre>2_input.txt – Блокнот Файл Правка Формат Вид Справка 8 find 3839442 add 123456 me add 0 granny find 0 find 123456 del 0 del 0 find 0 2_output.txt – Блокнот Файл Правка Ф not found granny me not found</pre>	 <pre>2_input.txt – Блокнот Файл Правка Формат Вид Справка 12 add 911 police add 76213 Mom add 17239 Bob find 76213 find 910 find 911 del 910 del 911 find 911 find 76213 add 76213 daddy find 762130 2_output.txt – Блокнот Файл Правка Ф Mom not found police not found Mom not found</pre>
--	---

Тестирование алгоритма:

	Время выполнения	Затраты памяти
Пример 1 из задачи	0.0030673000000000023 second	13.921875 mb
Пример 2 из задачи	0.0005357000000000001 second	13.9375 mb

Вывод по задаче: Мы узнали, что такое хешированные таблицы, в языке питон – `dictionary`, и научились ими пользоваться.

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.

- Формат входного файла(input.txt). Каждая строка входного файла содержит фамилию кандидата, за которого отдают голоса выборщики этого штата, затем через пробел идет количество выборщиков, отдавших голоса за этого кандидата.
- Формат выходного файла (output.txt). Выведите фамилии всех кандидатов в лексикографическом порядке, затем, через пробел, количество отданных за них голосов.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 64 мб.
- Пример:

Input.txt	McCain 10 McCain 5 Obama 9 Obama 8 McCain 1	ivanov 100 ivanov 500 ivanov 300 petr 70 tourist 1 tourist 2	bur 1
Output.txt	McCain 16 Obama 17	ivanov 900 petr 70 tourist 3	bur 1

Листинг кода:

```
import time
import os, psutil

def processing(info, name, voices):
    if name in info:
```

```

        info[name] += voices
    else:
        info[name] = voices

t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("5_input.txt")
m = open("5_output.txt", "w")

info = {}
while True:
    string = f.readline()
    if not string:
        break
    elements = list(map(str, string.split()))
    name = elements[0]
    voices = int(elements[1])
    processing(info, name, voices)

info = dict(sorted(info.items()))
for name, voices in info.items():
    m.write(name + ' ' + str(voices) + "\n")

f.close()
m.close()

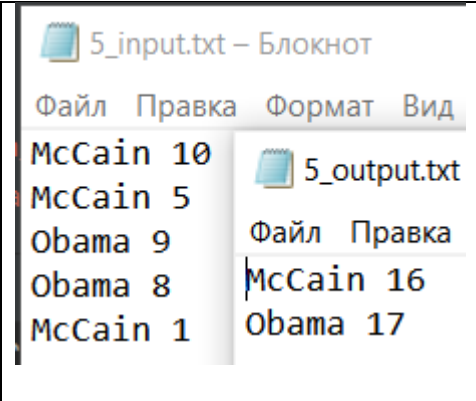
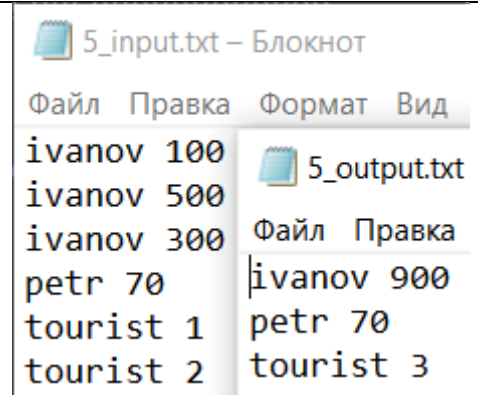
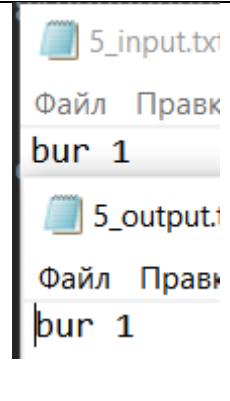
print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss / (1024 * 1024), "mb")

```

Текстовое объяснение решения:

Сначала мы читаем данные из файла и записываем все данные в словарь (хеш-таблицу). Ключи – имена кандидатов, значения – количества голосов. Далее была написана функция, которая изменяет значение ключа, при обращении к ней. В конце выводится весь словарь, то есть результаты выборов – имя кандидата + количество голосов.

Результат работы кода на примерах из текста задачи:

		
---	--	---

Тестирование алгоритма:

	Время выполнения	Затраты памяти
Пример 1 из задачи	0.002566699999999991 second	13.93359375 mb
Пример 2 из задачи	0.00059050000000000077 second	13.9296875 mb
Пример 3 из задачи	0.00281270000000000013 second	3.9296875 mb

Вывод по задаче.: Мы научились обращаться и перезаписывать значения ключей хеш-таблицы.

Вывод

Мы узнали, что такое хеш-таблицы, как они реализованы в питоне и научились ими пользоваться.