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.
Вариант 11

Выполнил:
Тимаков Е.П. (фамилия имя)
К3141 (номер группы)

Проверила:
Артамонова В.Е.

Санкт-Петербург 2024 г.

Содержание отчета

Задачи по варианту

Задача №1. Высота дерева

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

• Формат входного файла (input.txt). В первой строке входного файла находится строго положительное целое число операций N, не превышающее $5 \cdot 105$. В каждой из последующих N строк находится одна из следующих операций: – A x – добавить элемент x в множество. Если элемент уже есть в множестве, то ничего делать не надо. – D x – удалить элемент x. Если элемента x нет, то ничего делать не надо. – ? x – если ключ x есть в множестве, выведите «Y», если нет, то выведите «N». Аргументы указанных выше операций – целые числа, не превышающие по модулю 1018 .

• 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

Листинг кода:

```
class MN:
    def __init__(self):
        self.a: set = set()

    def A(self, value: int):
        self.a.add(value)
        return None

    def D(self, value: int):
        self.a.discard(value)
        return None

    def Q(self, value: int):
        return value in self.a


def main():
```

```
    mn = MN()
    a: list = []
    with open('input.txt') as f:
        while True:
            line = f.readline()
            if not line:
                break
            a.append((line.split()))
    with open("output.txt", "w") as f:
        for i in range(len(a)):
            if a[i][0] == "A":
                mn.A(int(a[i][1]))
            elif a[i][0] == "D":
                mn.D(int(a[i][1]))
            elif a[i][0] == "?":
                print(mn.Q(int(a[i][1])), file=f)

main()
```

Текстовое объяснение решения:

Данный код представляет собой реализацию структуру данных для выполнения операций добавления, удаления и проверки присутствия элемента. Класс MN создает экземпляр структуры данных, содержащий множество a. Метод A добавляет целочисленное значение в множество. Метод D удаляет целочисленное значение из множества, если оно там присутствует. Метод Q проверяет, содержится ли целочисленное значение в множестве.

Результат работы кода на примерах из текста задачи:
input.txt:

```
8
A 2
A 5
A 3
? 2
? 4
A 2
D 2
? 2
```

output.txt:

```
Y
N
N
```

Результат работы кода на максимальных:
input.txt:

```
500000
? 907639874583151183
? -18202344837784092
A 135929979742079561
A -5861522440636214
? -438234841457905552
D 312634484736209852
D -100272724814035304
? 71996682624157449
D 287808577610340444
D -547679730073828590
A -671378767710988550
A -977232759139263904
D 843700646752784122
? -724256494711020349
? 818698999650215680
A 299776389938210387
```

output.txt:

```
1  N
2  N
3  N
4  N
5  N
6  N
7  N
8  N
9  N
10 N
11 N
12 N
13 N
14 N
15 N
16 N
17 N
18 N
```

Результат работы кода на минимальных значениях:

input.txt:

```
1  1
2  ? 1
```

output.txt:

```
1  N
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.001	25
Пример из задачи	0.002	25
Верхняя граница	0.812	160

диапазона значений входных данных из текста задачи		

Вывод по задаче:

В данной задаче мы изучили и реализовали такую структуру, как множество, а также необходимые для работы с этим множеством операции.

Задача №2. Телефонная книга

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

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

• 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 ввода / входного файла (input.txt). В первой строке входного файла содержится число N ($1 \leq N \leq 105$) - количество запросов. Далее следуют N строк, каждая из которых содержит один запрос в формате, описанном выше. Все номера телефонов состоят из десятичных цифр, в них нет нулей в начале номера, и каждый состоит не более чем из 7 цифр. Все имена представляют собой непустые строки из латинских букв, каждая из которых имеет длину не более 15. Гарантируется при проверке, что не будет человека с именем «not found».

• Формат вывода / 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.

Листинг кода:

```
class tel_book:
    def __init__(self):
        self.book: dict = {}

    def add_number_name(self, number: str, name: str):
        self.book[f"{number}"] = name
        return None

    def del_number(self, number: str):
        if number in self.book:
            del self.book[number]
        return None

    def find_number(self, number: str):
        return self.book.get(number)


def main():
    with open("output.txt", "w") as f:
        for i in range(len(a)):
            if a[i][0] == "add":
                book.add_number_name(a[i][1], a[i][2])
            elif a[i][0] == "del":
                book.del_number(a[i][1])
            elif a[i][0] == "find":
                print(book.find_number(a[i][1]), file=f)


if __name__ == "__main__":
    book = tel_book()
    a: list = []
    with open('input.txt') as f:
        while True:
            line = f.readline()
            if not line:
                break
            a.append((line.split()))
```

main()

Текстовое объяснение решения:

Данный код представляет собой простую реализацию телефонной книги. Вот краткое объяснение каждого блока кода:

1. Класс tel_book:

- __init__: Инициализирует объект телефонной книги, создавая пустой словарь book. Метод: add_number_name: Добавляет номер телефона и имя в телефонную книгу. Метод: Удаляет номер телефона из телефонной книги, если он существует.Метод: find_number: Ищет имя по номеру телефона и возвращает его .Программа читает данные из списка a, который содержит операции (добавление, удаление, поиск) и соответствующие аргументы и выполняет соответствующие операции с помощью методов класса tel_book и записывает результаты в файл "output.txt".

Результат работы кода на примерах из текста задачи:

input.txt:

```
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 76213
```

output.txt:

```
Mom
Not found
police
Not found
Mom
daddy
```

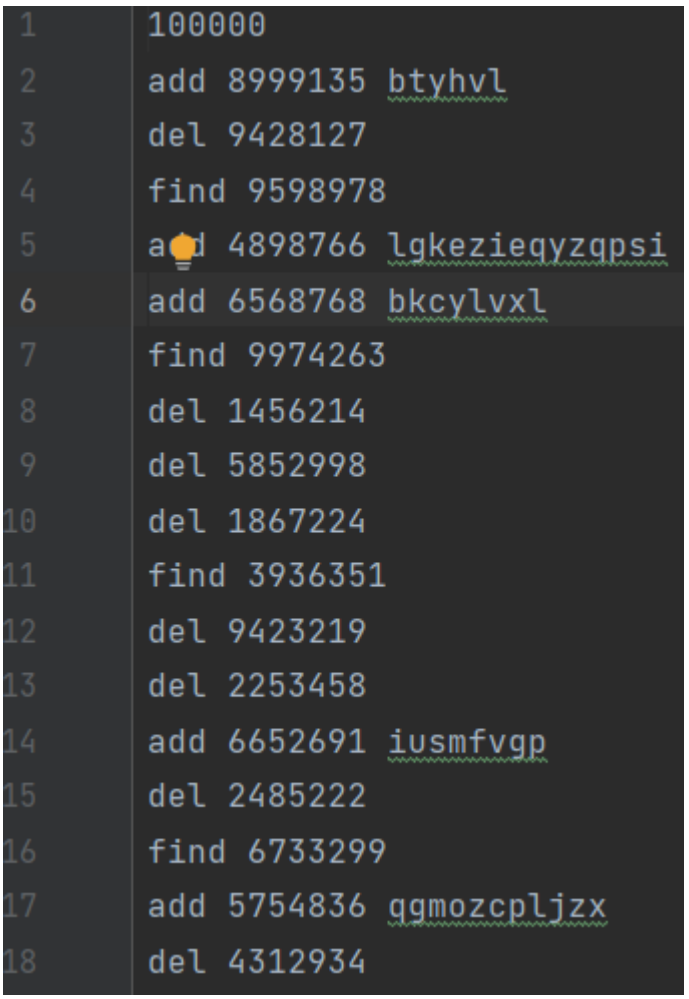
input.txt:

```
8
find 3839442
add 123456 me
add 0 granny
find 0
find 123456
del 0
del 0
find 0
```

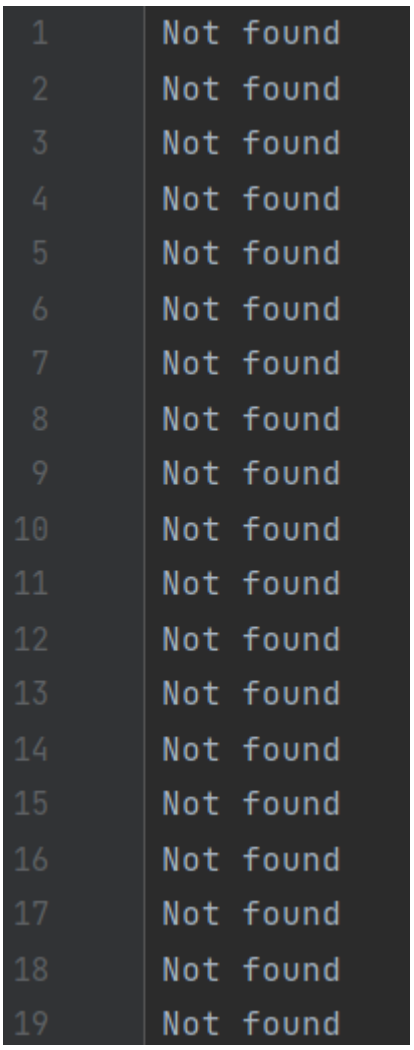
output.txt:

```
Not found
granny
me
Not found
```

Результат работы кода на максимальных:
input.txt:

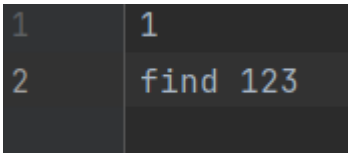

100000
add 8999135 btyhvl
del 9428127
find 9598978
add 4898766 lgkezieqyzqpsi
add 6568768 bkcylvxl
find 9974263
del 1456214
del 5852998
del 1867224
find 3936351
del 9423219
del 2253458
add 6652691 iusmfvgp
del 2485222
find 6733299
add 5754836 qgmozcpljzx
del 4312934

output.txt:

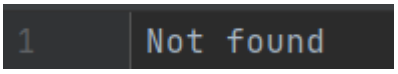


Результат работы кода на минимальных значениях:

input.txt:



output.txt:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.001	25
Пример из задачи	0.001	25

Пример из задачи	0.001	25
Верхняя граница диапазона значений входных данных из текста задачи	0.023	57

Вывод по задаче:

В данной задаче мы изучили и реализовали структуру данных такую, как карта, а также необходимые для работы с картой функции.

Задача №5. Выборы в США

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 ввода / 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 вывода / 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.

Листинг кода:

```
def add_book(bk: dict, key: str, value: int):
    if key in bk:
        bk[key] += value
        return None
```

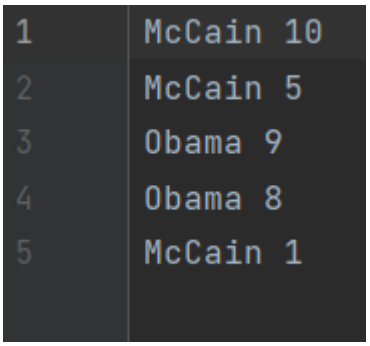
```
        bk[key] = value
        return None

def qs(a):
    return qs([x for x in a[1:] if x <= a[0]]) + [a[0]] + qs([x for x in a if x > a[0]])
if a else []


def main():
    a: list = []
    book: dict = {}
    with open('input.txt') as f:
        while True:
            line = f.readline()
            if not line:
                break
            a.append((line.split()))
    for i in range(len(a)):
        add_book(book, a[i][0], int(a[i][1]))
    result: list = []
    for i in book.items():
        result.append(i)
    result = qs(result)
    with open("output.txt", "w") as f:
        for i in result:
            print(i, file=f)

main()
```

Текстовое объяснение решения:

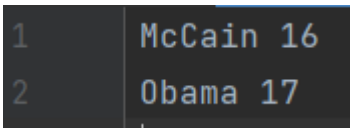
данный код складывает значения для повторяющихся ключей, сортирует эти пары и записывает отсортированный результат в файл output.txt. Функция add_book добавляет ключи и значения в словарь, увеличивая значение, если ключ уже существует. Функция qs реализует быструю сортировку для списка.

Результат работы кода на примерах из текста задачи:

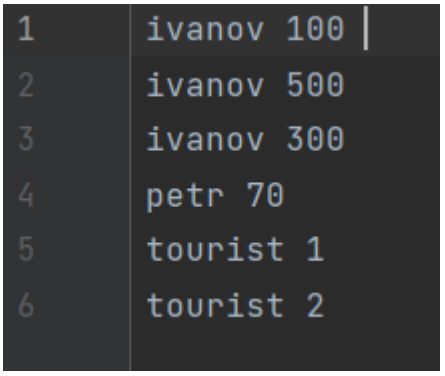
input.txt:



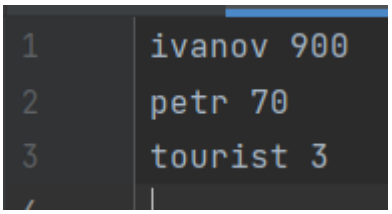
output.txt:



input.txt:

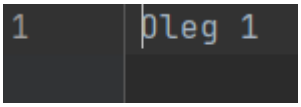


output.txt:

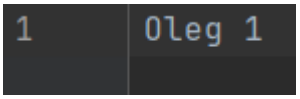


Результат работы кода на минимальных значениях:

input.txt:



output.txt:



	Время выполнения	Затраты памяти
Нижняя граница	0.001	25

диапазона значений входных данных из текста задачи		
Пример из задачи	0.001	25
Пример из задачи	0.001	25

Вывод по задаче:

В данной задаче мы изучили и реализовали структуру данных - карта. Также мы освежили знания алгоритма быстрой сортировки.

Вывод

В данной лабораторной работе мы изучили и реализовали необходимые каждому программисту структуры данных примера ключ - значение.