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 8

Выполнил:
Макунина А.А.
К32421

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Задачи по варианту

Задача №1. Множество

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 $5 \cdot 10^5$. 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».

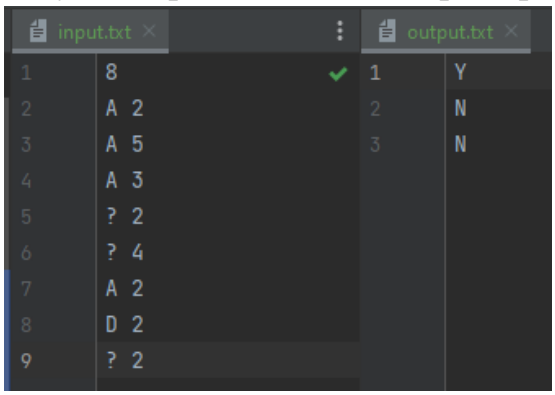
  Аргументы указанных выше операций – **целые числа**, не превышающие по модулю 10^{18}.
- **Формат выходного файла (output.txt).** Выведите последовательно результат выполнения всех операций «?». Следуйте формату выходного файла из примера.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

Я использовала словарь и массив, затем ввела количество операций, а затем через for/in прошлась по операциям. В каждой строке я вводила операцию (A, D, ?), и все числа были включены в массив mas. Первый элемент mas - letter, второй является key. Затем были проверки по первому элементу – операции. A – добавляет ключ в словарь, D – удаляет, а ? – проверяет, есть ли этот ключ в словаре.

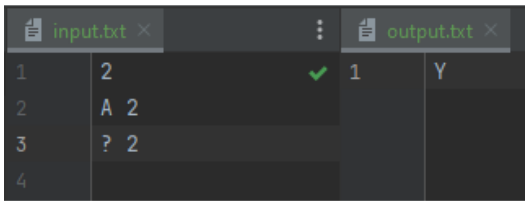
```
sets, result = {}, []
f = open('input.txt')
n = int(f.readline())
for i in range(n):
    arr = f.readline().split()
    letter = arr[0]
    key = arr[1]
```

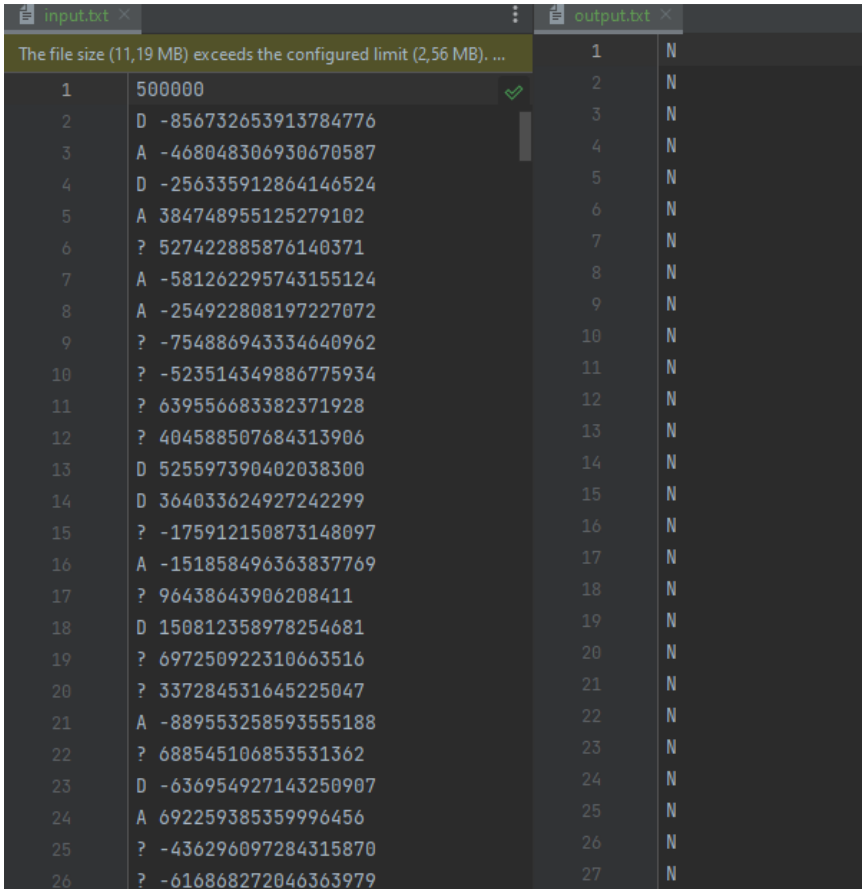
```
    if letter == 'A':
        sets[key] = []
    elif letter == 'D':
        if key in sets:
            del (sets[key])
    elif letter == '?':
        if key in sets:
            result.append('Y')
        else:
            result.append('N')
f.close()
w = open('output.txt', 'w')
w.write('\n'.join(result))
w.close()
```

Результат работы кода на примере из текста задачи:



Результат работы кода на максимальных и минимальных значениях:





	Время выполнения, с	Затраты памяти, мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0014300999999999897	0.017091751098632812
Пример из задачи	0.0018713000000000063	0.01755523681640625
Верхняя граница диапазона значений входных данных из текста задачи	2.0044647	27.262529373168945

Вывод по задаче: я узнала, что можно тестить эти задачи без своих входных данных, но курс на openedu недоступен…задача нот хард.

Задача №4. Прошитый ассоциативный массив

Реализуйте прошитый ассоциативный массив. Ваш алгоритм должен поддерживать следующие типы операций:

- `get` x – если ключ x есть в множестве, выведите соответствующее ему значение, если нет, то выведите <none>.
- `prev` x – вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен позже всех, но до x, или <none>,

если такого нет или в массиве нет x.

- `next` x – вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен раньше всех, но после x , или <none>, если такого нет или в массиве нет x .
- `put` x y – поставить в соответствие ключу x значение y. При этом следует учесть, что
  - если, независимо от предыстории, этого ключа на момент вставки в массиве не было, то он считается только что вставленным и оказывается самым последним среди добавленных элементов – то есть, вызов `next` с этим же ключом сразу после выполнения текущей операции put должен вернуть <none>;
  - если этот ключ уже есть в массиве, то значение необходимо изменить, и в этом случае ключ не считается вставленным еще раз, то есть, не меняет своего положения в порядке добавленных элементов.
- `delete` x – удалить ключ x. Если ключа в ассоциативном массиве нет, то ничего делать не надо.
- **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 приведенных выше операций. Ключи и значения операций - строки из латинских букв длиной не менее одного и не более 20 символов.
- **Формат выходного файла (output.txt).** Выведите последовательно результат выполнения всех операций get, prev, next. Следуйте формату выходного файла из примера.
- Ограничение по времени. 4 сек.
- Ограничение по памяти. 256 мб.

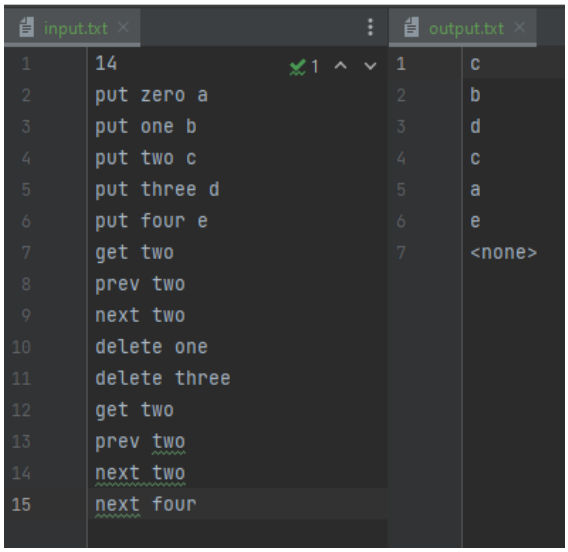
Реализовала функции, которые были описаны в условии задачи. Далее вводила количество функций, словарь и предыдущий элемент. Затем

проходилась по функциям и вводила их и дальше реализовывала их действия со словарем.

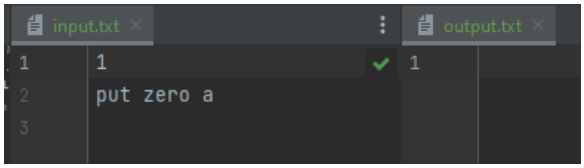
```
def get(x):
    if x in table:
        return table[x][1]
    return '<none>'


def prev(x):
    if x in table:
        return get(table[x][0])
    return '<none>'


def next(x):
    if x in table:
        return get(table[x][2])
    return '<none>'


def put(x, y):
    if x in table:
        table[x][1] = y
    else:
        global before
        table[before][2] = x
        table[x] = [before, y, '<none>']
        before = x
    return table


def delete(x):
    if x in table:
        global before
        if x == before:
            if table[x][0] != '<none>':
                before = table[x][0]
        if table[x][0] != '<none>':
            table[table[x][0]][2] = table[x][2]
        if table[x][2] != '<none>':
            table[table[x][2]][0] = table[x][0]
        del (table[x])
    return table
```

```
f = open('input.txt')
N = int(f.readline())
result, table, before = [], {}, ''
for i in range(N):
    com = list(f.readline().split())
    if com[0] == 'put':
        if len(table) == 0:
            table[com[1]] = ['<none>', com[2],
'<none>']
            before = com[1]
        else:
            put(com[1], com[2])
    elif com[0] == 'get':
        result.append(get(com[1]))
    elif com[0] == 'prev':
        result.append(prev(com[1]))
    elif com[0] == 'next':
        result.append(next(com[1]))
    else:
        delete(com[1])
f.close()
w = open('output.txt', 'w')
w.write('\n'.join(result))
w.close()
```

Результат работы кода на примере из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения, с	Затраты памяти, мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0013147000000000002	0.017740249633789062
Пример из задачи	0.0012018000000000029	0.020751953125
Верхняя граница диапазона значений входных данных из текста задачи	Здесь должен запуститься тест на опенеду вероятно, но у меня нет доступа к курсу…а самой писать много команд…не хочется	аналогично

Вывод по задаче: нормально, интересно, здорово, что в тексте задачи нормально прописаны команды.

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), ..., (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 $1 \le i < j \le n$, а камни S_i и S_j образуют красивую пару, то есть существует такое число $1 \le q \le k$, что $S_i = a_q$ и $S_j = b_q$.

Если число таких пар оказывается достаточно большим, то шах покупает все камни. Однако в этот раз купцы привезли настолько много камней, что шах не может посчитать это число. Поэтому он вызвал своего визиря и поручил ему этот подсчет. Напишите программу, которая находит ответ на эту задачу.

- **Формат ввода / входного файла (input.txt).** Первая строка входного файла содержит целые числа n и k ($1 \le n \le 100000$, $1 \le k \le 676$) – число камней, которые привезли купцы и число пар, которые шах считает красивыми. Вторая строка входного файла содержит строку S, описывающую типы камней, которые привезли купцы.

  Далее следуют k строк, каждая из которых содержит две строчных буквы латинского алфавита и описывает одну из красивых пар камней.

- **Формат вывода / выходного файла (output.txt).** В выходной файл выведите ответ на задачу – количество пар, которое должен найти визирь.

- Ограничение по времени. 1 сек.

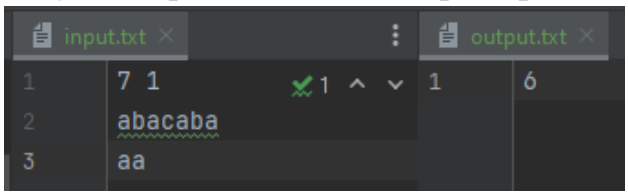
- Ограничение по памяти. 64 мб.

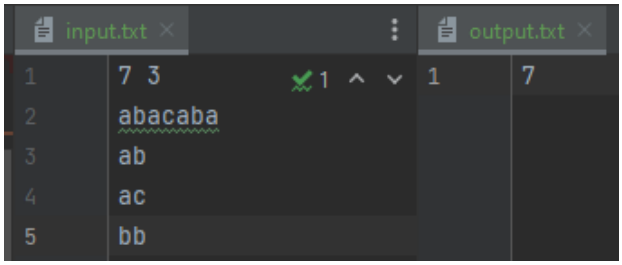
Чтобы определить количество пар, необходимо выяснить, какие пары могут быть вообще, и пробежаться по строке в поисках таких пар. Каждый раз, когда мы находим пару, мы увеличиваем счетчик.

```
def count(stones, couple, counter):
    now = 0
```

```
    for i in range(len(stones) - 1, -1, -1):
        if stones[i] in couple:
            for next in couple[stones[i]]:
                if next in counter:
                    now += counter[next]
        if stones[i] not in counter:
            counter[stones[i]] = 0
        counter[stones[i]] += 1
    w.write(str(now))


f = open('input.txt')
lines = f.readlines()
couple, counter = {}, {}
n, k = map(int, lines[0].split())
stones = lines[1]
f.close()
for j in range(k):
    pair = lines[2 + j]
    if pair[0] not in couple:
        couple[pair[0]] = []
    couple[pair[0]].append(pair[1])
w = open('output.txt', 'w')
count(stones, couple, counter)
w.close()
```

Результат работы кода на примере из текста задачи:





	Время выполнения, с	Затраты памяти, мб
Пример из задачи	0.0023080000000000045	0.01746654510498047

Пример из задачи	0.0017165999999999987	0.01756572723388672

Вывод по задаче: нот хард, может, мне так кажется, на фоне предыдущих задач.

Вывод

В ходе лабораторной работы я поняла, что лабораторная достаточно приятная. Я вспомнила как взаимодействовать с хэш-функциями. Это предпоследняя лабораторная 1 семестра, чему я несомненно рада, ведь у меня ещё 4 предмета в академической разнице…