Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Мегафакультет трансляционных информационных технологий

Факультет инфокоммуникационных технологий

Дисциплина: Алгоритмы и структуры данных

**Отчет по Лабораторной работе №6**

Выполнила: Микулина Алиса Романовна

Группа: K3143, 1 курс

Преподаватель: Харьковская Татьяна Александровна

Санкт-Петербург

29.12.2021

**Описание задания**

**Задание 0.**

Вставляем циферки в уже данные хеш-функции, расшифровываем нужные нам для решения задачи. Радуемся, что одной из нужных задач является первая.

**Задание 1.**

Нужно реализовать множество, в котором должны храниться числа под хешами. Нужно иметь возможность добавлять, удалять элементы и проверять множество на их наличие.

**Задание 2.**

Создаем телефонную книгу, в которую можно добавлять людей с номерами, искать человека по номеру и удалять номера.

**Задание 4.**

Необходимо написать нечто невероятное, наподобие чудища, которое будет проходить тесты с opened и будет считать все функции (get, prev, next, put и delete). Многоуважаемая функция delete, за что вы так со мной…..

**Задание 5.**

Выборы в США. Все легко и просто. Просто пишем мини-код и радуемся жизни.

**Задание 7.**

Еще один мини-код после жуткой 4-й задачи. Пишем нечто про сравнение драгоценных камней, составление их пар и дышим спокойно.

**Описание решения и исходный код**

**Задача 1.**

Алгоритм максимально прост. Две лекции (наша и из Яндекса) спасли мозг от переплавки. Единственное, что вызвало дикое непонимание, это тот факт, что делить надо при хешировании на большое число, иначе выполнение при максимальных значениях will take ages. К функции из лекции Яндекса надо добавить проверку на наличие элемента в множестве. По факту, все, очень просто и со вкусом, и тесты на openedu проходят :)

class MySet():

    def \_\_init\_\_(*self*, *size*):

*self*.size = *size*

*self*.myset = [[] *for* p *in* range(*size*)]

    def find(*self*, *x*):

*for* now *in* *self*.myset[*x* % *self*.size]:

*if* now == *x*:

*return* 'Y'

*return* 'N'

    def add(*self*, *x*):

*if* *self*.find(*x*) == 'Y':

*return*

*self*.myset[*x* % *self*.size].append(*x*)

    def delete(*self*, *x*):

        xlist = *self*.myset[*x* % *self*.size]

*for* i *in* range(len(xlist)):

*if* xlist[i] == *x*:

                xlist[i], xlist[len(xlist) - 1] = xlist[len(xlist) - 1], xlist[i]

                xlist.pop()

*return* *self*.myset

*return* *self*.myset

set = MySet(5 \* (10 \*\* 5))

ans = []

*with* open("input.txt", "r") *as* f:

    num\_actions = int(f.readline())

*for* i *in* range(num\_actions):

        action, number = f.readline().split(' ')

        number = int(number)

*if* action == 'A':

            set.add(number)

*elif* action == 'D':

            set.delete(number)

*elif* action == '?':

            ans.append(set.find(number))

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

*for* answer *in* ans:

        d.write(answer + '\n')

**Задача 2.**

По реализации задача напоминает первую, только класс мы строим на основе словаря, а не массива. В ключах храним номера, в значениях – имена. Поиск по номеру, добавление, удаление – все предельно просто и наивно.

В эту секунду она еще не знала, что ее ждет…

class Phonebook():

    def \_\_init\_\_(*self*):

*self*.items = {}

    def add(*self*, *number*, *name*):

*if* *name* == 'not found':

*raise* AssertionError('Wrong Name!')

*self*.items[*number*] = *name*

    def find(*self*, *number*):

        name = *self*.items.get(*number*)

*if* name == None:

*return* 'not found'

*else*:

*return* name

    def delete(*self*, *number*):

*if* *self*.items.get(*number*) != None:

*del* *self*.items[*number*]

*return*

    def output(*self*):

*return* *self*.items

book = Phonebook()

ans = []

*with* open("input\_2.txt", "r") *as* f:

    num\_actions = int(f.readline())

*for* i *in* range(num\_actions):

        file = f.readline().strip().split(' ')

*if* len(file) > 2:

            number, name = int(file[1]), file[2]

            book.add(number, name)

*else*:

*if* file[0] == 'del':

                number = int(file[1])

                book.delete(number)

*else*:

                number = int(file[1])

                ans.append(book.find(number))

*with* open("output\_2.txt", "w") *as* d:

*for* answer *in* ans:

        d.write(answer + '\n')

**Задача 4.**

Никак не могу назвать эту задачу, кроме как “Ошибка в Nном тесте.” Столько ошибок, как здесь, я еще ни разу не ловила, мне кажется. Код валился чуть ли не на каждой проверке, после чего моя соседка слушала много нехороших слов от меня про мои мозги и неспособность думать. В определенный момент все было стерто, удалено и выкинуто из корзины, с гневными мыслями “За что эта задача у меня обязательная!?!?!??!?”

Ну да не будем о плохом. Вот это чудо прошло все тесты из openedu, что в определенный момент мне казалось totally невозможным. Тут все основано на функциях, и словаре, в котором мы, собственно, храним все элементы со ссылками на предыдущие и последующие. Можно было бы написать класс, в котором будут ссылки (а-ля двусвязный список-словарь), но что-то как-то за 14 часов до защиты настроения не было вообще (желания существовать тоже).

Хитрость всей системы в том, что в значениях словаря хранятся массивы, где нулевой элемент – само значение элемента, первый – ссылка на последующий элемент, а второй – на предыдущий. У нас есть глобальная переменная, которая позволяет отслеживать, какой элемент был добавлен последним, чтобы при добавлении элементов у нас была возможность сразу дать ссылку на элемент, добавленный ранее. В этот же момент предыдущему элементу дается ссылка на вновь поступивший элемент. Таким образом, мы заполняем все ячейки ссылками на элементы. Получается гусеница с ножками (I’m not sorry). У первого элемента нет ссылки на предыдущий, у последнего – на последующий.

Благодаря наличию ссылок мы моем просто найти элемент и вывести его предыдущее или последующее значение.

Поиск элемента тоже очень прост. Если ключ есть в ключах – ура!!! Если нет – выводим <none>.

Самое ужасное что? Правильно, удаление. Почему? Потому что при удалении из центра (ну, из любого места, кроме начала и конца, там весьма просто) нам нужно менять ссылки у соседних двух элементов. Это легко?) Да, легко. Если ты не путаешься в индексах достаточно сильно, чтобы запороть все что только можно и даже больше. Еще очень загадочно, когда оказывается, что часть функции (смена индекса назад) вообще не работала, но часть тестов все равно проходились, и все ломалось уже на 10 +. Это очень озадачивало, потому что валилось не на времени, не на памяти, а непонятно на чем, удачи в поисках, так сказать. Хорошо хоть на opened подсказки есть. Я по ним все дебажила (ну хотя бы пыталась). • – •

Плакать хочется, но терпимо…

def get(*set*, *key*):

*if* *key* in *set*:

*return* *set*[*key*][0]

*return* '<none>'

def prev(*set*, *key*):

*if* *key* in *set*.keys():

*if* *set*[*key*][1] != None:

            local\_key = *set*[*key*][1]

*return* *set*[local\_key][0]

*return* '<none>'

def next(*set*, *key*):

*if* *key* in *set*.keys():

*if* *set*[*key*][2] != None:

            local\_key = *set*[*key*][2]

*return* *set*[local\_key][0]

*return* '<none>'

def put(*set*, *key*, *value*):

    global current\_prev

*if* *key* in *set*.keys():

*set*[*key*][0] = *value*

*return*

*set*[*key*] = [*value*, None, None]

*set*[*key*][1] = current\_prev

*if* current\_prev != None:

        local\_key = *set*[*key*][1]

*set*[local\_key][2] = *key*

    current\_prev = *key*

*return*

def delete(*set*, *key*):

    global current\_prev

*if* *key* in *set*.keys():

*if* *set*[*key*][0] is not None and *set*[*key*][1] is not None and *set*[*key*][2] is not None:

            newnext = *set*[*key*][2]

            newprev = *set*[*key*][1]

*set*[newprev][2] = newnext

*set*[newnext][1] = newprev

*elif* *set*[*key*][1] is None and *set*[*key*][2] is not None:

            local\_key = *set*[*key*][2]

*set*[local\_key][1] = None

*elif* *set*[*key*][2] is None and *set*[*key*][1] is not None:

            local\_key = *set*[*key*][1]

            current\_prev = local\_key

*set*[local\_key][2] = None

*else*:

            current\_prev = None

*del* *set*[*key*]

*return*

set = {}

ans = []

current\_prev = None

*with* open("input.txt", "r") *as* f:

    num\_actions = int(f.readline())

*for* i *in* range(num\_actions):

        file = f.readline().strip().split(' ')

*if* len(file) > 2:

            key, value = file[1], file[2]

            put(set, key, value)

*else*:

*if* file[0] == 'delete':

                key = file[1]

                delete(set, key)

*elif* file[0] == 'get':

                key = file[1]

                ans.append(get(set, key))

*elif* file[0] == 'prev':

                key = file[1]

                ans.append(prev(set, key))

*elif* file[0] == 'next':

                key = file[1]

                ans.append(next(set, key))

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

*for* answer *in* ans:

        d.write(answer + '\n')

**Задача 5.**

Очень просто, очень приятно, мне максимально понравилось (особенно, после 4 задачи!!!). Я, на радостях, даже класс Election создала, потому что так красиво! (и потому что могу). В классе буквально две функции: добавить в словарь и вывести список участников с баллами.

Просто и радостно на душе после 4-й лабы, ыыыыыыыы……

class Election():

    def \_\_init\_\_(*self*):

*self*.names = {}

    def add(*self*, *name*, *votes*):

*if* *name* in *self*.names:

*self*.names[*name*] += *votes*

*return*

*self*.names[*name*] = *votes*

*return*

    def output(*self*):

        ans = ''

*for* name *in* sorted(list(*self*.names.keys())):

            ans += str(name) + ' ' + str(*self*.names[name]) + '\n'

*return* ans

set = Election()

ans = ''

*with* open("input\_5.txt", "r") *as* f:

*for* line *in* f.readlines():

        file = line.strip().split(' ')

        name, votes = file[0], int(file[1])

        set.add(name, votes)

*with* open("output\_5.txt", "w") *as* d:

    d.write(set.output())

**Задача 7.**

Задача на первый взгляд сложная, но по факту очень простая. Долго не могла понять, по какому принципу искать пары, но в итоге все оказалось элементарно. Всего лишь определяем, какие пары могут быть и пробегаем по строке в поиске таких пар. Каждый раз, когда находим пару, увеличиваем счетчик. Все слишком просто после четвертой задачи.

Спать хочется…

def count(*stones*, *pairs*, *counter*):

    ans = 0

*for* i *in* range (len(*stones*)-1, -1, -1):

*if* *stones*[i] in *pairs*:

*for* next *in* *pairs*[*stones*[i]]:

*if* next in *counter*:

                    ans += *counter*[next]

*if* *stones*[i] not in *counter*:

*counter*[*stones*[i]] = 0

*counter*[*stones*[i]] += 1

    print(ans)

*with* open('input\_7.txt') *as* f:

    lines = f.readlines()

pairs = {}

counter = {}

n, k = map(int, lines[0].split())

stones = lines[1]

*for* j *in* range(k):

    pair = lines[2+j]

*if* pair[0] not in pairs:

        pairs[pair[0]] = []

    pairs[pair[0]].append(pair[1])

count(stones, pairs, counter)

**Описание проведенных тестов.**

**1**

Изображение выглядит как стол

Автоматически созданное описание

**2**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Time** |
| 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 | Mom  not found  police  not found  Mom  daddy | 0.0023759000000000002 секунд |
| 8  find 3839442  add 123456 me  add 0 granny  find 0  find 123456  del 0  del 0  find 0 | Mom  not found  police  not found  Mom  daddy | 0.003880300000000003 секунд |

**4**

**Изображение выглядит как стол

Автоматически созданное описание**

**5**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Time** |
| ivanov 100  ivanov 500  ivanov 300  petr 70  tourist 1  tourist 2 | ivanov 900  petr 70  tourist 3 | 0.007675199999999997 секунд |
| McCain 10  McCain 5  Obama 9  Obama 8  McCain 1 | McCain 16  Obama 17 | 0.008546200000000004 секунд |
| bur 1 | bur 1 | 0.002524099999999998 секунд |

**7**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Time** |
| 7 1  abacaba  aa | 6 | 0.002285800000000001 секунд |
| 7 3  abacaba  ab  ac  bb | 7 | 0.0024968000000000004 секунд |

**Выводы по проделанной работе.**

Последняя лаба написана!!! Мне понравилось, я дико устала, но я смогла! Скоро Новый Год, уходим в него без долгооов!!!

Написание отчета было веселым, потому что 1) я его опять писала ночью; 2) Защитин из инстаграмма заманил меня просмотром смешариков на защите, которая началась в 5 утра, и я вот сижу пишу отчет под звуки того, как они там в Доту играют теперь :)

Лаба в целом приятная (кроме 4-й задачи), да и 4-я была бы нормальной, если бы я не была уже на пределе с 3 бессонными ночами подряд. Уже и смысла спать нет, наверное, пойду пить чай :)