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 10

Выполнил:

Говоров П. И.

К3140

Проверил:

Афанасьев А. В.

Санкт-Петербург

2024 г.

Содержание отчета

Оглавление

[Содержание отчета 2](#_Toc185637189)

[1 задача. Множество 3](#_Toc185637190)

[2 задача. Телефонная книга 5](#_Toc185637191)

[4 задача. Прошитый ассоциативный массив 7](#_Toc185637192)

[5 задача. Выборы в США 9](#_Toc185637193)

[7 задача. Драгоценные камни 10](#_Toc185637194)

[Вывод 12](#_Toc185637195)

Задачи по варианту

**1 задача.** 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 превышающее5 · 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 HashSet:  
 def \_\_init\_\_(self, size=6\*10\*\*5):  
 self.size = size  
 self.table = [[] for i in range(size)]  
  
 def add(self, key):  
 ind = self.get\_hash(key)  
 if key not in self.table[ind]:  
 self.table[ind].append(key)  
  
 def pop(self, key):  
 ind = self.get\_hash(key)  
 if key in self.table[ind]:  
 self.table[ind].remove(key)  
  
 def check(self, key):  
 return key in self.table[self.get\_hash(key)]  
  
 def get\_hash(self, key):  
 return hash(key) % self.size

Текстовое объяснение решения.

Класс с операциями “добавление ключа”, “удаление ключа”, “проверка существования ключа”, где обращение происходит через хэш по модулю в массиве.

Результат работы кода на примере из задачи:

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

Результат работы кода на максимальных и минимальных значениях: Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.09136 sec | 14.56 Mb |
| Пример из задачи | 0.11091 sec | 16.71 Mb |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.87454 sec | 36.76 Mb |

Вывод по задаче: Увеличение значений вводимых переменных в пределах ограничений увеличивает время работы программы в зависимости O(n) и используемую память.

2 задача. 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 ≤ N ≤ 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.

• Примеры:Листинг кода

class PhoneAdapter:  
 def \_\_init\_\_(self):  
 self.numb\_dict = {}  
  
 def add\_num(self, number, name):  
 self.numb\_dict[number] = name  
  
 def pop\_num(self, number):  
 if number in self.numb\_dict:  
 del self.numb\_dict[number]  
  
 def find\_num(self, number):  
 return self.numb\_dict.get(number, "not found")

Текстовое объяснение решения.

Создаем класс с помощью, которого управляем и храним номерам через словарь

Результат работы кода на примере из задачи:

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

Результат работы кода на максимальных и минимальных значениях:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.00133 sec | 14.33 Mb |
| Пример из задачи | 0.00267 sec | 14.79 Mb |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.43443 sec | 35.12 Mb |

Вывод по задаче: Увеличение значений вводимых переменных в пределах ограничений увеличивает время работы программы в зависимости O(n) и используемую память.

4 задача. 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 . • prev x – вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен позже всех, но до x, или , 6 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 , если такого нет или в массиве нет x . • put x y – поставить в соответствие ключу x значение y. При этом следует учесть, что – eсли, 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 ; – 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.

Листинг кода

class StitchedHashMap:  
 def \_\_init\_\_(self):  
 self.prev\_dep = {}  
 self.next\_dep = {}  
 self.data = {}  
 self.values\_queue = []  
  
  
 def get(self, item):  
 return self.data.get(item, "<none>")  
  
 def prev(self, val):  
 if val in self.prev\_dep and self.prev\_dep[val] in self.data:  
 return self.data[self.prev\_dep[val]]  
 else: return "<none>"  
  
 def next(self, item):  
 if item in self.next\_dep and self.next\_dep[item] in self.data:  
 return self.data[self.next\_dep[item]]  
 else: return "<none>"  
  
  
 def put(self, key, val):  
 if key not in self.data:  
 if self.values\_queue:  
 last\_key = self.values\_queue[-1]  
 self.next\_dep[last\_key] = key  
 self.prev\_dep[key] = last\_key  
 self.values\_queue.append(key)  
 self.data[key] = val  
  
  
 def delete(self, val):  
 if val in self.data:  
 ind = self.values\_queue.index(val)  
 if ind > 0:  
 self.next\_dep[self.values\_queue[ind - 1]] = self.next\_dep[val]  
 if ind < len(self.values\_queue) - 1:  
 self.prev\_dep[self.values\_queue[ind + 1]] = self.prev\_dep[val]  
 self.values\_queue.pop(ind)  
 self.prev\_dep.pop(val, "")  
 self.next\_dep.pop(val, "")  
 del self.data[val]

Текстовое объяснение решения.

Создаем класс со словарем где хранятся данные, массивом порядка элементов и словарями, где лежит ссылка на следующий и предыдущий элемент

Результат работы кода на примере из задачи:

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

Результат работы кода на максимальных и минимальных значениях: Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.00598 sec | 14.78 Mb |
| Пример из задачи | 0.00641 sec | 15.07 Mb |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.51925 sec | 36.19 Mb |

Вывод по задаче: Увеличение значений вводимых переменных в пределах ограничений увеличивает время работы программы в зависимости времени работы O(n), а также используемую память.

5 задача. 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 результами 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

class ElectionsAdapter:  
 def \_\_init\_\_(self):  
 self.elect\_dict = {}  
  
 def add\_num(self, name, cnt):  
 self.elect\_dict[name] = self.elect\_dict.get(name, 0) + cnt  
  
 def get\_data(self):  
 names\_sorted = sorted(self.elect\_dict.keys())  
 return [f"{name} {self.elect\_dict[name]}" for name in names\_sorted]

Текстовое объяснение решения.

Использую словарь для подсчета голосов

Результат работы кода на примере из задачи:

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Результат работы кода на максимальных и минимальных значениях: Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.00234 sec | 14.77 Mb |
| Пример из задачи | 0.00254 sec | 14.83 Mb |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.47529 sec | 37.17 Mb |

Вывод по задаче: Увеличение значений вводимых переменных в пределах ограничений увеличивает время работы программы в зависимости O(n) и используемую память.

7 задача. 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: (a1, b1), (a2, b2), ..., (ak, bk). 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 ≤ i < j ≤ n, а камни Si и Sj образуют красивую пару, то есть существует такое число 1 ≤ q ≤ k, что Si = aq и Sj = bq. 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.

Листинг кода

def get\_b\_pairs(st\_rocks, arr\_pairs):  
 ans = 0  
 index\_dict = {i:[] for i in alph\_lc}  
  
 for i in range(len(st\_rocks)):  
 index\_dict[st\_rocks[i]].append(i)  
  
 for x, y in arr\_pairs:  
 arr1, arr2 = index\_dict[x], index\_dict[y]  
  
 i = j = 0  
 while i < len(arr1) and j < len(arr2):  
 if arr2[j] > arr1[i]:  
 ans += (len(arr2) - j)  
 i += 1  
 else:  
 j += 1  
  
 return ans

Текстовое объяснение решения.

Подсчитываю в словарь индексы камней, затем через 2 указателя подсчитываю количество комбинаций

Результат работы кода на примере из задачи:

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Результат работы кода на максимальных и минимальных значениях: Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.01938 sec | 14.77 Mb |
| Пример из задачи | 0.02316 sec | 16.02 Mb |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.63208 sec | 36.35 Mb |

Вывод по задаче: Увеличение значений вводимых переменных в пределах ограничений увеличивает время работы программы в зависимости O(n) и используемую память.

Вывод

1. В этой лабораторной мы познакомились и научились создавать множества, хеш-таблицы и хэш-функции, а также реализовывать некоторые алгоритмы для этих структур
2. Алгоритм O(n) и O(n logn) выполняется достаточно быстро, относительно квадратичной сложности, затраты памяти также прямо пропорциональны линейной.
3. С помощью методов *time.perf\_counter()* и *psutil.Process().memory\_info().rss* можно отслеживать ресурсозатратность алгоритмов.