**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Информатика»**

Тема: **Алгоритмы и структуры данных в Python**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Размочаева Н.В. |

Санкт-Петербург

2018

**Цель работы**

Изучить основы работы и применения словарей в Python; изучить алгоритмы сортировки, научиться применять их и оценивать их сложность; применить один из алгоритмов сортировки к словарю.

**Задание**

Вы - колонизатор новой неизведанной планеты. Вокруг Вас огромное количество полезных ископаемых и других ресурсов; каждый день появляются новые виды. Вы понятия не имеете, зачем они нужны, но привязанный за ногу к столу ученый настаивает на необходимости бережно хранить ресурсы и вести их учет. Он также уговаривал Вас ввести классификацию, основанную на различном регистре букв, но Вы не стали слушать чокнутого и для простоты учета решили хранить все названия в **строго нижнем регистре**.

Ваши бестолковые подчиненные, инопланетные крысы и желание покушать приводят к тому, что некоторые ресурсы могут убывать.

На вход программе подается произвольное количество пар, разделенных пробелом

<ресурс> <количество>

Где ресурс - название ресурса, а количество - либо положительно либо отрицательное число. В зависимости от получения или траты ресурса.

Перечень пар заканчивается словом Total. После него программа должна распечатать список ресурсов и их текущее количество (ресурсов может быть отрицательное число. Мы сами не знаем как - считайте, что живете в кредит).

**Вывод должен быть лексикографически упорядочен по названию ресурса.**

**Выполнение работы**

1. Вначале идет считывание данных в словарь dict. Считывание продолжается до слова Total. При вводе наименования ресурсов переводятся в нижний регистр и используются как ключи для словаря. Количество ресурсов сохраняется под соответствующим ключом. Если такого ключа еще нет, он создается.
2. Далее создается список из всех ключей словаря, так как сами словари отсортировать напрямую нельзя.
3. Список ключей сортируется с помощью сортировки выбором. Последовательность проходится в поисках максимального элемента. Найденный максимум меняется местами с последним элементом, таким образом, неотсортированная часть последовательности уменьшается на один элемент. К неотсортированной части применяются аналогичные действия — находится максимум и ставится на последнее место в неотсортированной части массива. Алгоритм продолжается до тех пор, пока неотсортированная часть массива не уменьшится до одного элемента.
4. Далее в цикле for для каждого ключа выводим его значение.

**Выводы**

В ходе лабораторной работы были изучены основы работы словарей, их методы и алгоритмы сортировки. Получен опыт по применению этих алгоритмов на практике в языке Python.

**Приложение А. Оценка сложности алгоритма сортировки выбором**

Данный алгоритм сортировки может быть как устойчивый, то есть он не меняет относительный порядок сортируемых элементов, имеющих одинаковые ключи, так и неустойчивый. Алгоритм проходит по всем еще не отсортированным элементам, и, как только найдет элемент больше, чем последний из неотсортированных, поменяет их местами. Таким образом будет нужно O(*n*2) обменов (для каждого i требуется O(n−i)  обменов). На массиве из n элементов имеет время выполнения в худшем, среднем и лучшем случае O(*n*2), предполагая, что сравнения делаются за постоянное время. В среднем этот алгоритм имеет квадратичную сложность. Сложность по памяти – максимальное число просматриваемых ячеек при обработке входного объекта. Нам не нужно выделять никакой дополнительный массив под элементы, поэтому дополнительная сложность равна O(*1*).

|  |  |  |
| --- | --- | --- |
| Сложность по времени | Худшая | O(*n*2) |
| Средняя | O(*n*2) |
| Лучшая | O(*n*2) |
| Сложность по памяти | Общая | O(*n*) |
| Дополнительная | O(*1*) |

**Приложение Б. Исходный код**

def SelectionSort(x):

for i in range (len(x) - 1):

for j in range (i + 1, len(x)):

if x[i] > x[j]:

c = x[j]

x[j] = x[i]

x[i] = c

return(x)

dict = {}

while True:

s = input().split()

if s[0] != "Total":

s[0] = s[0].lower()

if s[0] in dict:

dict[s[0]] = int(s[1]) + int(dict[s[0]])

else:

dict.update({s[0] : s[1]})

else:

break

l = list(dict.keys())

l = SelectionSort(l)

for i in l:

print(i, dict[i])