**Титульный лист рейтинговой работы**



***Кафедра информационных систем***

|  |  |
| --- | --- |
| ***Рейтинговая работа*** | ***Расчетно-аналитическое задание*** |
| ***по дисциплине*** | ***Алгоритмизация и программирование*** |
|  |  |
| ***Задание / Вариант №*** | ***Индивидуальное задание в соответствие с ФИО и ID*** |
|  |  |
| ***Тема*** | ***Алгоритмы типовых задач*** |
|  |  |
| ***Выполнена обучающимся группы*** | о.ИЗДтс 23.2/Б1-22 |
| ***ФИО обучающегося*** | Духленков Андрей Геннадьевич |
|  |  |
| ***Преподаватель*** | Камолова Анастасия Николаевна |

Москва – 2023 г.

**Содержание**

[1. ЗАДАНИЕ С ИСХОДНЫМИ ДАННЫМИ 3](#_Toc131081662)

[2. ОПИСАНИЕ ПОСЛЕДОВАТЕЛЬНОСТИ ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc131081663)

[3. ОБОСНОВАНИЕ ВЫБОРА АЛГОРИТМА СОРТИРОВКИ ЧИСЛОВОГО НАБОРА ДАННЫХ 4](#_Toc131081664)

[*3.1 Сортировка пузырьком* 4](#_Toc131081665)

[*3.2 Сортировка выбором* 5](#_Toc131081666)

[4. ЛИСТИНГ ПРОГРАММНОГО КОДА 6](#_Toc131081667)

[5. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ 10](#_Toc131081668)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 11](#_Toc131081669)

**1. ЗАДАНИЕ С ИСХОДНЫМИ ДАННЫМИ**

***Исходные данные***

ФИО и ID подготавливаются заранее в текстовом файле с именем source\_data.txt в кодировке Unicode (Юникод). Каждое слово, составляющее ФИО должно быть с заглавной буквы. Слова между собой должны быть разделены пробелом. Точка в конце не ставится.

***Задание***

Требуется написать программу на языке Python, которая выполняет следующие задачи:

1. Считать исходные данные из текстового файла source\_data.txt

2. Вычислить целое значение, получившееся в результате деления ID на количество символов, составляющих ФИО (количество символов подсчитывается без учета пробелов).

3. Определить направление сортировки в зависимости от числа, полученного в п.2:

- сортировка по возрастанию, если число чётное;

- сортировка по убыванию, если число нечётное.

4. Сформировать набор данных (список) из кодов Юникода каждого символа ФИО (исключая пробелы между словами), переведенных в десятичную форму.

5. Выполнить сортировку набора данных по убыванию или по возрастанию. Сортировку выполнить двумя различными способами

6. Вычислить среднее арифметическое значение набора данных. При необходимости, результат округлить до третьего знака после запятой.

7. Вычислить среднее квадратическое значение набора данных. При необходимости, результат округлить до третьего знака после запятой.

**2. ОПИСАНИЕ ПОСЛЕДОВАТЕЛЬНОСТИ ВЫПОЛНЕНИЯ РАБОТЫ**

На основании полученного задания был составлен алгоритм. Далее были изучены материалы различных источников. Был сформирован файл, source\_data.txt, с исходными данными. После этого была написана программа в соответствии с полученным заданием. При помощи написанной программы составлен файл конечных данных, result.txt.

**3. ОБОСНОВАНИЕ ВЫБОРА АЛГОРИТМА СОРТИРОВКИ ЧИСЛОВОГО НАБОРА ДАННЫХ**

***3.1 Сортировка пузырьком***

Алгоритм сортировки методом пузырька является простым в реализации и понимании, что делает его хорошим выбором для начинающих программистов или для задач, где необходима быстрая сортировка небольшого массива.

Его преимущество заключается в том, что он использует небольшой объем дополнительной памяти. Также, если массив уже отсортирован или почти отсортирован, то алгоритм пузырька быстро завершает работу, что делает его быстрым и эффективным в таких случаях.

Наконец, алгоритм пузырька может быть легко модифицирован для реализации других алгоритмов сортировки, таких как шейкерная сортировка или сортировка выбором.

***3.2 Сортировка выбором***

Алгоритм сортировки методом выбора имеет следующие преимущества:

1. Простота. Алгоритм легко понять и реализовать. При определенных условиях может быть эффективнее, чем более сложные алгоритмы.
2. Малое количество перемещений элементов. Сортировка методом выбора требует меньшего количества перемещений элементов, чем другие методы сортировки.
3. Работа с неупорядоченными последовательностями. Алгоритм метода выбора не зависит от того, является ли сортируемая последовательность упорядоченной или нет. Он работает быстро и правильно на любой последовательности.
4. Использование дополнительной памяти. Алгоритм метода выбора не требует использовать дополнительную память, за исключением временной переменной, которую можно выделить в рамках сортируемого массива.

В целом, метод выбора является эффективным алгоритмом сортировки для небольших и средних размеров массивов и прекрасно подходит для решения малых задач и задач, где продуктивность программы не самый приоритетный фактор.

**4. ЛИСТИНГ ПРОГРАММНОГО КОДА**

В данном фрагменте кода происходит чтение данных из файла:

with open('source\_data.txt', 'r', encoding='utf8') as f:

data = (list(f.read().split('\n')))

Здесь происходит извлечение имени и ID и их запись в переменные:

name = data[0]

id = int(data[1])

В данном фрагменте кода происходит удаление пробелов из имени и вычисление значения id деленного на количество символов ФИО, а также создание пустого массива для хранения Unicode-кодов символов:

name\_without\_space = name.replace(' ', '')

name\_div\_id = int(id / len(name\_without\_space))

unicode\_array = []

Далее происходит перебор каждого символа в ФИО (без пробелов) и добавление его Unicode-кода в массив:

for char in name\_without\_space:

unicode\_char = ord(char)

unicode\_array.append(unicode\_char)

В нижеприведённом фрагменте кода происходит выбор алгоритма сортировки в зависимости от значения числа полученного при делении id на количество символов в ФИО, а также запись значение в переменные, которые понадобятся для составления файла с результатом программы:

if name\_div\_id % 2 == 0:

dirrection = 'по возрастанию'

number = 'чётное'

soreted\_arr = bubble\_sort(unicode\_array)

soreted\_arr\_2 = selection\_sort(unicode\_array)

else:

dirrection = 'по убыванию'

number = 'не чётное'

soreted\_arr = bubble\_sort(unicode\_array, False)

soreted\_arr\_2 = selection\_sort(unicode\_array, False)

Далее производится вычисление среднего арифметического и среднего квадратического значения и их запись в переменные:

arithmetic = arithmetic\_mean(unicode\_array)

quadratic = quadratic\_mean(unicode\_array)

В следующем фрагменте кода происходит запись результатов в файл:

with open("result.txt", 'w', encoding='utf8') as f:

f.write(f"""1. Исходные данные: {name}; ID: {id}

2. {name\_div\_id}

3. Направление сортировки: {dirrection}, так как число {name\_div\_id} - {number}

4. Набор данных: {unicode\_array}

5. Отсортированный {dirrection} набор данных {soreted\_arr}

6. Среднее арифметическое значение: {arithmetic}

7. Среднее квадратическое значение: {quadratic}""")

В данном методе представлен алгоритм сортировки пузырьком:

def bubble\_sort(array, direction=True):

n = len(array)

for i in range(n):

for j in range(0, n-i-1):

if not direction:

if array[j] < array[j+1]:

array[j], array[j+1] = array[j+1], array[j]

else:

if array[j] > array[j+1]:

array[j], array[j+1] = array[j+1], array[j]

return array

В следующем методе представлен алгоритм сортировки методом выборки:

def selection\_sort(array, direction=True):

for i in range(len(array)):

first\_value = i

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

if not direction:

if array[j] > array[first\_value]:

first\_value = j

else:

if array[j] < array[first\_value]:

first\_value = j

array[i], array[first\_value] = array[first\_value], array[i]

return array

В нижеприведённом методе придавлен алгоритм для вычисления среднего арифметического:

def arithmetic\_mean(array):

sum = 0

for i in array:

sum += i

result = sum / len(array)

return round(result, 3)

В данном методе представлен алгоритм вычисления среднего квадратического значения:

def quadratic\_mean(array):

sum = 0

for i in array:

sum += i\*\*2

result = (sum / len(array))\*\*(1/2)

return round(result, 3)

**5. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

По выполнению программы мы получаем файл result.txt с данными, которые представлены ниже:

1. Исходные данные: Духленков Андрей Геннадьевич; ID: 70192434

2. 2699709

3. Направление сортировки: по убыванию, так как число 2699709 - не чётное

4. Набор данных: [1100, 1095, 1093, 1091, 1088, 1086, 1085, 1085, 1085, 1085, 1083, 1082, 1081, 1080, 1077, 1077, 1077, 1077, 1076, 1076, 1074, 1074, 1072, 1044, 1043, 1040]

5. Отсортированный по убыванию набор данных [1100, 1095, 1093, 1091, 1088, 1086, 1085, 1085, 1085, 1085, 1083, 1082, 1081, 1080, 1077, 1077, 1077, 1077, 1076, 1076, 1074, 1074, 1072, 1044, 1043, 1040]

6. Среднее арифметическое значение: 1077.923

7. Среднее квадратическое значение: 1078.021

В пункте 1 представлены исходные данные из фала source\_data.txt. В пункте 2 записано число получено при делении ID на количество символов ФИО без учета пробелов. В пункте 3 указано направление сортировки, полученное на основании числа из пункта 2. В пункте 4 представлен массив данных из Unicode-кодов ФИО без учета пробелов. В пункте 5 записан массив из пункта 4 отсортированный в направлении, указанном в пункте 3. В пункте 6 записано среднее арифметическое значение массива из пункта 4. В пункте 7 записано среднее квадратическое значение массива из пункта 4.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Объяснение алгоритмов сортировки с примерами на Python [Электронный ресурс] URL: <https://tproger.ru/translations/sorting-algorithms-in-python/>.
2. Электронный университет МУИВ [Электронный ресурс] URL: <https://e.muiv.ru/my/>.
3. Практикум по алгоритмизации и программированию на Python : курс: учебное пособие / Хахаев И. А. // Сортировка массива – 2016 – С. 83-85.
4. Файлы в python, ввод-вывод [Электронный ресурс] URL: <https://pythonru.com/osnovy/fajly-v-python-vvod-vyvod>.