

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.03 | Прикладная информатика | | |
| Направленность (профиль) | |  | Прикладная информатика в химии | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Разработка программных систем | | |
| Курс | II | | | Группа | 425 |

Отчёт по контрольной работе № 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнители: |  |  |  |  |
| обучающиеся группы 425 |  |  |  | Перьков Алексей  Евсеенко Валентин  Понадысев Тимур |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Дамрин Антон Олегович |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |

СОДЕРЖАНИЕ

[1 Цель работы и задание 3](#_Toc152681431)

[Цель работы 3](#_Toc152681432)

[Задание 3](#_Toc152681433)

[2 Описание хода выполнения 3](#_Toc152681434)

[3 Результаты тестирования 5](#_Toc152681437)

[4 Исходный код программыё 7](#_Toc152681438)

[[sort.py] 7](#_Toc152681439)

[[load.py] 8](#_Toc152681440)

[[interface.py] 9](#_Toc152681441)

# Цель работы и задание

## Цель работы

Разработка программы сортировки согласно варианту.

|  |  |  |
| --- | --- | --- |
| 3 | Поразрядная сортировка / Radix sort (версия least significant digit) | Перьков Алексей Михайлович, Понадысев Тимур Сергеевич,  Евсеенко Валентин Максимович |

## Задание

1. Каждый участник группы разработки должен зарегистрироваться в любой открытой системе контроля версий (к примеру, <https://github.com/> или <https://about.gitlab.com/>).
2. Создать проект будущей программы в выбранной системе контроля версий Каждый участник учебной группы в рамках работы над лабораторной должен внести хотя бы одно изменение в проект.
3. Реализовать модуль загрузки тестовых данных. Это может быть модуль загрузки предварительно созданных данных из файла, либо модуль генерации случайных массивов.
4. Реализовать модуль вывода отсортированных данных. Это может быть модуль загрузки данных в выходной файл, либо модуль вывода построенного графика отсортированных данных, либо модуль сохранения построенного графика отсортированных данных.
5. Описать алгоритм сортировки (согласно варианту) в виде блок-схемы или псевдокода. Данный описанный алгоритм должен быть впоследствии приложен к отчёту.
6. Запрограммировать описанный выше алгоритм сортировки.
7. Протестировать запрограммированный алгоритм, результаты тестирования приложить к отчёту.
8. Рассчитать лучшую и худшую сложность реализованного алгоритма на основании описания, сделанного в пункте 5.
9. Написать отчёт по работе, в котором представить текст программы и результат её работы.

* Текст программы должен быть хорошо прокомментирован. По всем не прокомментированным участкам программы будут заданы вопросы.

# Описание хода выполнения

Результат выполнения задания выложен по адресу (ветка no-recursion):

<https://github.com/LOHA3000/RADIX-LSD-Sort>

Блок-схема алгоритма представлена на рисунке 1.

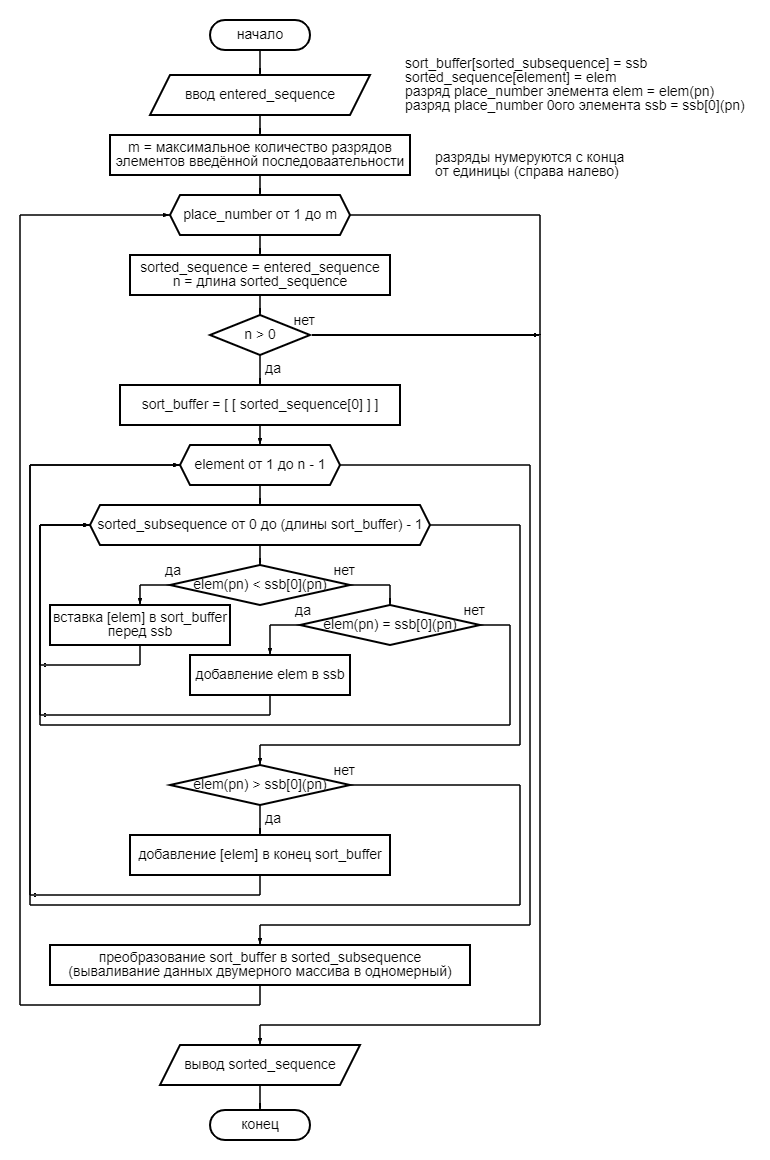


Рисунок 1 – блок-схема алгоритма

Для расчёта сложности учитывается несколько факторов:

- количество элементов (n);

- количество символов в элементе (m);

- количество различных символов, которые может принять каждый символ в определённом месте каждого элемента (k).

Так как перебираются все доступные разряды, первым сомножителем будет O(m­max). Далее перебираются все элементы последовательности, исключая первый, следовательно вторым сомножителем будет O(n - 1), равное O(n). Значение первых двух сомножителей не изменяются для лучшей и худшей ситуаций.

Для каждого элемента производится сравнение с группами уже разделённых элементов. Количество таких групп ограничено ki для i-ой итерации, обрабатывающей i-ый разряд. Поэтому последний сомножитель будет O(1+2+3+…+ki+ki+ki+ki) = O(ki) = O(n) для худшего случая, когда все символы различны и идут в необходимом порядке (по возрастанию), и O(1+1+1+…+1) = O(1) для лучшего случая, когда все элементы идут в обратном порядке (по убыванию).

Для худшего случая выражение имеет вид:

Для лучшего случая выражение принимает вид:



# Результаты тестирования

Тестирование ввода последовательности с клавиатуры, из файла и заполнением случайными последовательностями из цифр представлены на рисунках 4, 5 и 6. Процесс открытия файлов и сохранения файлов представлен на изображениях 2 и 3.

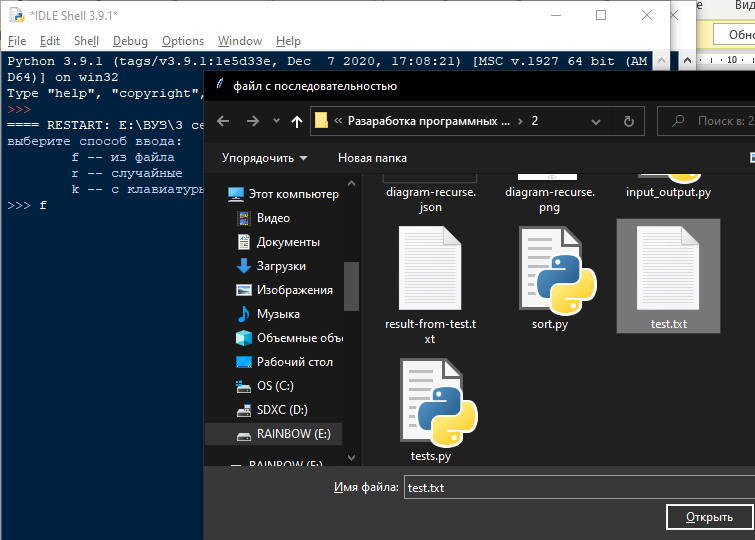


Рисунок 2 – открытие файла

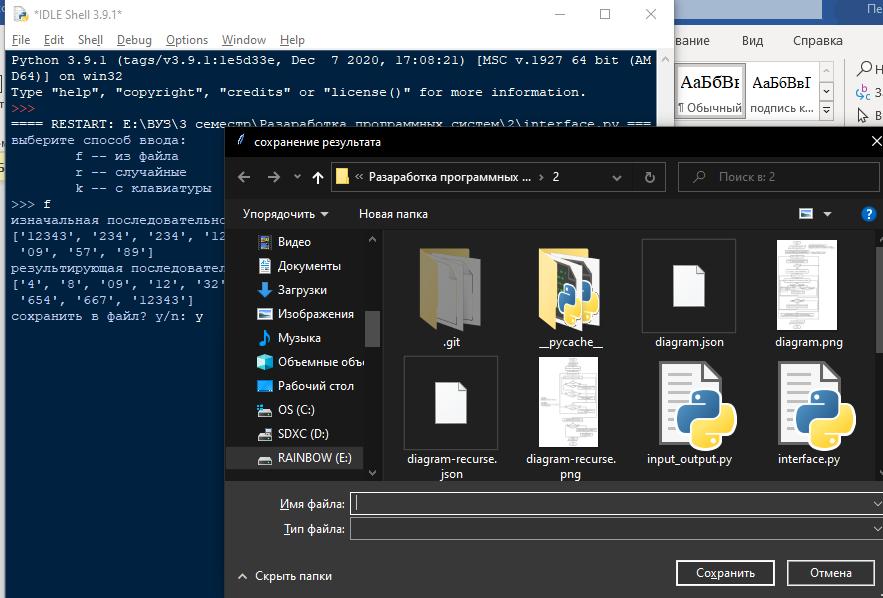


Рисунок 3 – сохранение последовательности

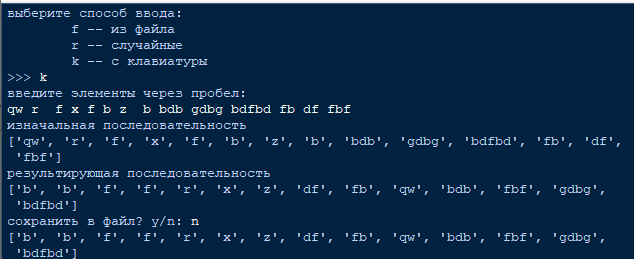


Рисунок 4 – ввод с клавиатуры

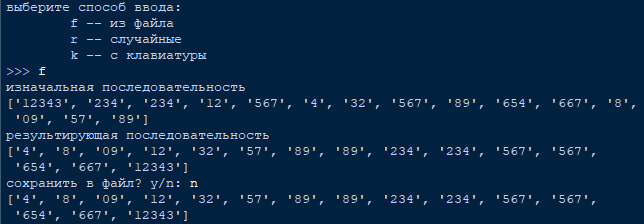


Рисунок 5 – ввод из файла

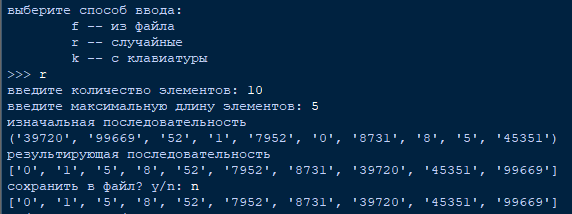


Рисунок 6 – ввод последовательности случайных чисел

# Исходный код программы

## [sort.py]

"""

модуль сортировки

"""

def sort(sorting\_sequence, place\_number=0):

"""

метод RADIX LSD

или

поразрядная по наименее значимой цифре

"""

sorted\_sequence = sorting\_sequence[:]

for place\_number in range(max(map(len, sorting\_sequence))):

def get\_digit\_at\_place\_number(gived\_element):

if len(gived\_element) < place\_number + 1:

return '' # минимальное значение при отсутствии разряда

return gived\_element[-(place\_number + 1)]

if not len(sorted\_sequence): # пустая последовательность

return sorted\_sequence

sort\_buffer = [[sorted\_sequence[0]]] # заполнение начальным значением

for element in sorted\_sequence[1:]:

for index, sorted\_subsequence in enumerate(sort\_buffer): # перебор сохранённых значений

if (get\_digit\_at\_place\_number(element) <

get\_digit\_at\_place\_number(sorted\_subsequence[0])):

# меньше текущего

sort\_buffer.insert(index, [element])

break

elif (get\_digit\_at\_place\_number(element) ==

get\_digit\_at\_place\_number(sorted\_subsequence[0])):

# равно текущему

sorted\_subsequence.append(element)

break

if (get\_digit\_at\_place\_number(element) >

get\_digit\_at\_place\_number(sorted\_subsequence[0])):

# больше последнего

sort\_buffer.append([element])

# объединение строк

sorted\_sequence = [element for subsequence in sort\_buffer for element in subsequence]

return sorted\_sequence

## [load.py]

"""

модуль ввода / вывода

"""

from random import randint

from os.path import exists

from pprint import pprint, pformat

def load\_from\_file(path\_to\_file):

"""

функция загрузки из файла

path\_to\_file: str -- путь до файла

"""

if not exists(path\_to\_file):

print(f'файл "{path\_to\_file}" не существует')

return None

with open(path\_to\_file) as outer\_file:

# вычленение всех подстрок разделённых пробелами

result\_sequence = outer\_file.read().strip().split()

return result\_sequence

def load\_random\_digits():

"""

функция создания последовательности из числовых подстрок

"""

while True:

try:

elements\_number = int(input('введите количество элементов: '))

if elements\_number <= 0:

raise ValueError

break

except ValueError:

print('введите корректное значение!')

while True:

try:

elements\_lenght = int(input('введите максимальную длину элементов: '))

if elements\_lenght <= 0:

raise ValueError

break

except ValueError:

print('введите корректное значение!')

result\_sequence = tuple([str(randint(0, 10\*\*randint(0, elements\_lenght))) for i in range(elements\_number)])

return result\_sequence

def load\_from\_input():

"""

функция загрузки данных вводимых с клавиатуры

"""

# получение элементов из ввода пользователем

result\_sequence = input('введите элементы через пробел:\n').strip().split()

return result\_sequence

def save(sorted\_sequence, path\_to\_file):

"""

метод сохранения отсортированных данных

"""

with open(path\_to\_file, 'w') as out\_file:

out\_file.write(' '.join(sorted\_sequence))

## [interface.py]

"""

модуль интерфейса

"""

import sort

import input\_output

from tkinter import Tk

from tkinter.filedialog import asksaveasfilename, askopenfilename

from os import getcwd as current\_path

class Interface:

"""

класс с функциями управления интерфейсом

"""

def \_\_init\_\_(self):

"""

инициализация объекта класса,

загрузка последовательности

"""

self.entered\_sequence = self.load\_sequence()

self.result\_sequence = []

def load\_sequence(self):

"""

функция загрузки массива

"""

while True:

load\_way = input('выберите способ ввода:\n'

'\tf -- из файла\n'

'\tr -- случайные\n'

'\tk -- с клавиатуры\n>>> ').lower()

if load\_way not in ('f', 'r', 'k'):

print('выберите один из предложенных способов!')

continue

else:

break

if load\_way == 'f':

loaded\_sequence = input\_output.load\_from\_file(askopenfilename(

title='файл с последовательностью', initialdir=current\_path()))

elif load\_way == 'r':

loaded\_sequence = input\_output.load\_random\_digits()

elif load\_way == 'k':

loaded\_sequence = input\_output.load\_from\_input()

return loaded\_sequence

def print(self):

"""

метод вывода введённой последовательности

"""

print('изначальная последовательность')

input\_output.pprint(self.entered\_sequence, compact=True)

def print\_sorted(self):

"""

метод вывода введённой последовательности

"""

print('результирующая последовательность')

input\_output.pprint(self.result\_sequence, compact=True)

def sort\_sequence(self):

"""

метод сортировки загруженного массива

"""

self.result\_sequence = sort.sort(self.entered\_sequence)

def save\_sorted\_sequence(self):

"""

метод сохранения отсортированного массива

"""

if not len(self.result\_sequence):

print('последовательность не отсортирована!')

save = input('сохранить в файл? y/n: ').lower()

if save in ('да', 'д', 'y', 'yes'):

input\_output.save(self.result\_sequence,

asksaveasfilename(title='сохранение результата',

initialdir=current\_path()))

else:

input\_output.pprint(self.result\_sequence, compact=True)

if \_\_name\_\_ == '\_\_main\_\_':

root = Tk()

while True:

i = Interface()

i.print()

i.sort\_sequence()

i.print\_sorted()

i.save\_sorted\_sequence()

root.destroy()