САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1
по курсу «Алгоритмы и структуры данных»
Тема: Сортировка вставками, выбором, пузырьковая.
Вариант 4

Выполнил:
Волжева М.И.
К3141

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Задачи по варианту

Задача №1. Сортировка вставкой

Текст задачи:
Используя код процедуры Insertion-sort, напишите программу и проверьте сортировку массива A = {31, 41, 59, 26, 41, 58}.

- Формат входного файла (input.txt). В первой строке входного файла содержится число n ($1 \le n \le 10^3$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9
- Формат выходного файла (output.txt). Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб
- Пример:

Input.txt	6 31 41 59 26 41 58
Output.txt	26 31 41 41 58 59

Листинг кода:

```
import time
import os, psutil
t_start = time.perf_counter()
process = psutil.Process(os.getpid())


def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >=0 and key < arr[j] :
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key


f = open("1_input.txt")
m = open("1_output.txt", "w")
num = int(f.readline())
string = f.readline()
numbers = list(map(int, string.split()))

insertion_sort(numbers)
m.write(" ".join(map(str, numbers)))
f.close()
m.close()
```

```
print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

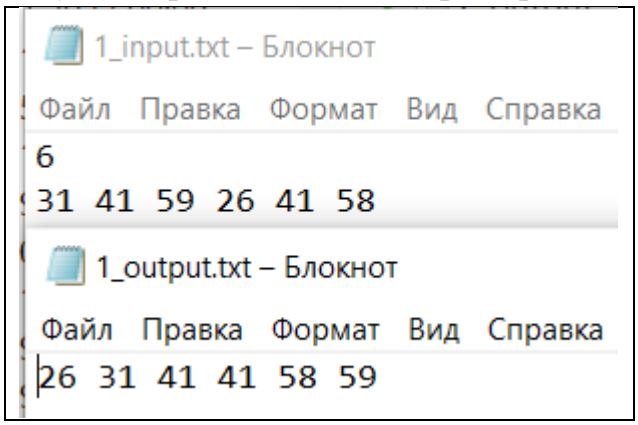
Листинг кода, используемый для тестирования:

```
import random
t_start = time.perf_counter()
process = psutil.Process(os.getpid())

def test(a, b):
    k = []
    for i in range(0, a):
        k.append(random.randint(-b, b))
    return k

def test_test(arr):
    k = False
    for i in range(0, len(arr)-1):
        if arr[i] > arr[i+1]:
            return False
    return True


f.write("1000\n")
numbers = test(1000, 1000000000)
f.write(" ".join(map(str, numbers)))
insertion_sort(numbers)
if test_test(numbers):
    m.write(" ".join(map(str, numbers)))
```

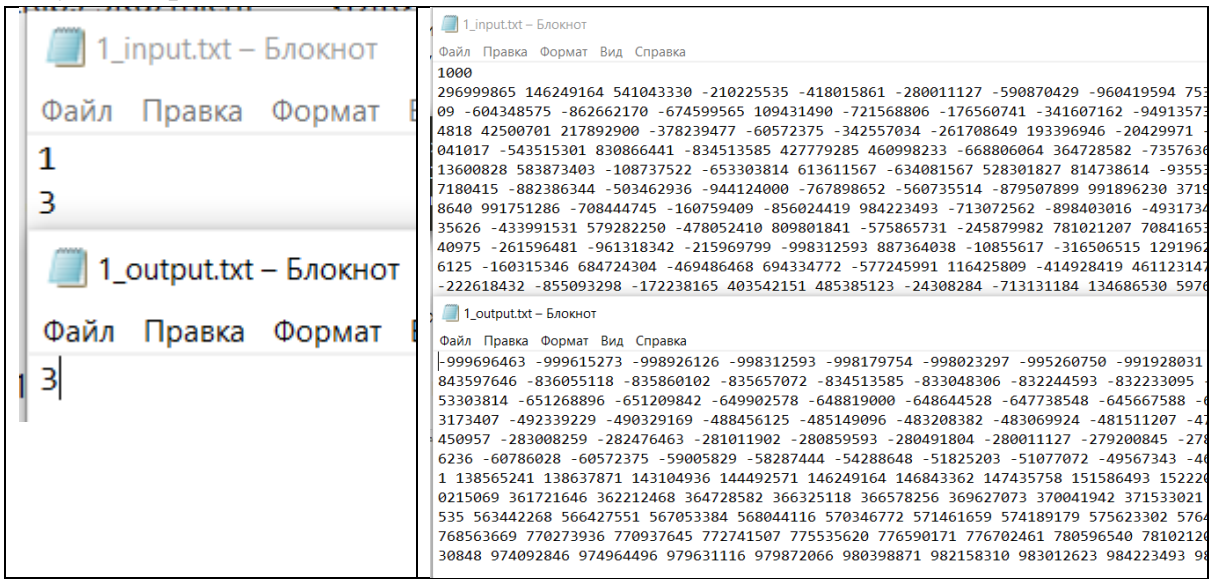
Текстовое объяснение решения:

Сначала была написана функция сортировки массива методом вставки(insertion_sort). Далее я прочла данные из файла, преобразовала их в массив, вызвала для них написанную раннее функцию и результат вывела в файл. Далее были написаны функции для тестирования. Функция test – создает рандомный массив по параметрам (первый – количество элементов второй – модульное ограничение каждого элементов), а функция test_test проверяет правильно ли отсортирован массив.

Результат работы кода на примере из текста задачи:



Результат работы кода на максимальных и минимальных значениях:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.008467700000000022 second	13.34375 mb
Пример из задачи	0.0022447000000000022 second	13.59375 mb
Верхняя граница диапазона значений входных данных из текста задачи	0.028808399999999998 second	13.734375 mb

Вывод по задаче: Я узнали, как работает алгоритм сортировки вставкой, и научилась реализовывать его. Также я проверила эффективность его реализации на питоне на время и память.

Задача №2. Сортировка вставкой +

Текст задачи:
Измените процедуру Insertion-sort для сортировки таким образом, чтобы в выходном файле отображалось в первой строке n чисел, которые обозначают новый индекс элемента массива после обработки.

- Формат выходного файла (input.txt). В первой строке выходного файл выведите n чисел. При этом i-ое число равно индексу, на который, в момент обработки его сортировкой вставками, был перемещен i-ый элемент исходного массива. Индексы нумеруются, начиная с единицы. Между любыми двумя числами должен стоять ровно один пробел.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб
- Пример:

Input.txt	10 1 8 4 2 3 7 5 6 9 0
Output.txt	1 2 2 2 3 5 5 6 9 1 0 1 2 3 4 5 6 7 8 9

Листинг кода:

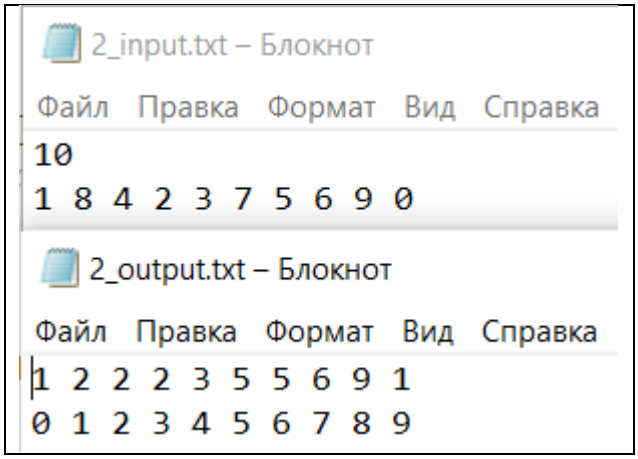
```
import time
import os, psutil
import copy
t_start = time.perf_counter()
process = psutil.Process(os.getpid())


def insertion_sort(arr1, arr2):
    arr2[0] = 0 #первый элемент(с индексом 0) остается на месте
    for i in range(1, len(arr1)):
        key = arr1[i]
        j = i-1
        while j >= 0 and key < arr1[j]:
            arr1[j+1] = arr1[j]
            j -= 1
        arr1[j+1] = key
        arr2[i] = j + 1 #заносим индекс, куда был переставлен
соответсвующий элемент



f = open("2_input.txt")
m = open("2_output.txt", "w")
num = int(f.readline())
string = f.readline()
numbers = list(map(int, string.split()))
numbers_copy = copy.copy(numbers)
```

```
insertion_sort(numbers, numbers_copy)
m.write(" ".join(map(str, numbers_copy)))
m.write('\n')
m.write(" ".join(map(str, numbers)))
f.close()
m.close()


print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Была переделана функция insertion_sort, теперь в нее передается два одинаковых массива, чтобы соблюсти количество элементов. Далее во второй массив вносятся значения куда переставляются элементы массивы соответствующего индекса. (Нумерация индексов в нашем случае начинается с 0). В коде отмечены строчки, в которых заполняется 2 массив.

Результат работы кода на примерах из текста задачи:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0023717999999999934 second	13.5546875 mb
Пример из задачи	0.0035690999999999917 second	13.44140625 mb
Верхняя граница диапазона значений входных данных из	0.031789899999999996 second	13.953125 mb

текста задачи		

Вывод по задаче: Я более детально разобралась, как работает алгоритм сортировкой вставкой, научилась выводить номера индексов, куда переставляются элементы массива при сортировке. Также я проверила эффективность его реализации на питоне на время и память.

Задача №7. Знакомство с жителями Сортлэнда

Текст задачи:

Владелец графства Сортлэнд, граф Бабблсортер, решил познакомиться со своими подданными. Число жителей в графстве нечетно и составляет n, где n может быть достаточно велико, поэтому граф решил ограничиться знакомством с тремя представителями народонаселения: с самым бедным жителем, с жителем, обладающим средним достатком, и с самым богатым жителем. Согласно традициям Сортлэнда, считается, что житель обладает средним достатком, если при сортировке жителей по сумме денежных сбережений он оказывается ровно посередине. Известно, что каждый житель графства имеет уникальный идентификационный номер, значение которого расположено в границах от единицы до n. Информация о размере денежных накоплений жителей хранится в массиве M таким образом, что сумма денежных накоплений жителя, обладающего идентификационным номером i, содержится в ячейке M[i]. Помогите секретарюграфа мистеру Свопу вычислить идентификационные номера жителей, которыебудут приглашены на встречу с графом.

- Формат входного файла (input.txt). Первая строка входного файла содержит число жителей n ($3 \le n \le 9999$, n нечетно). Вторая строка содержитописание массива M, состоящее из положительных вещественных чисел, разделенных пробелами. Гарантируется, что все элементы массива M различны, а их значения имеют точность не более двух знаков после запятой и не превышают 10^6
- Формат выходного файла (output.txt). В выходной файл выведите три целых положительных числа, разделенных пробелами — идентификационные номера беднейшего, среднего и самого богатого жителей Сортлэнда.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб
- Пример:

Input.txt	331	327305
Output.txt	9	5

Листинг кода:

```
import time
import os, psutil
import copy
import random
t_start = time.perf_counter()
process = psutil.Process(os.getpid())
```

```
def insertion_sort(arr): #функция, которая сортирует массив методом вставок
(по возрастанию)
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >= 0 and key < arr[j] :
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key


def selection_sort(arr):
    for i in range(0, len(arr) - 1):
        smallest = i
        for k in range(i + 1, len(arr)):
            if arr[k] < arr[smallest]:
                smallest = k
        arr[i], arr[smallest] = arr[smallest], arr[i]


def search_number(a, index, arr, arr_clone): #функция, которая ищет
определенный элемент в первоначальном массиве и возвращает его место
(индекс+1)
    for i in range(0, a):
        if arr_clone[i] == arr[index]:
            return i + 1


f = open("7_input.txt")
m = open("7_output.txt", "w")
num = int(f.readline())
string = f.readline()
numbers = list(map(float, string.split()))
numbers_clone = copy.copy(numbers)
result = []

selection_sort(numbers)
result.append(search_number(num, 0, numbers, numbers_clone)) #самый бедный
result.append(search_number(num, int((num-1)/2), numbers, numbers_clone))
#средний
result.append(search_number(num, -1, numbers, numbers_clone)) #самый
богатый
m.write(" ".join(map(str, result)))

f.close()
m.close()
print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

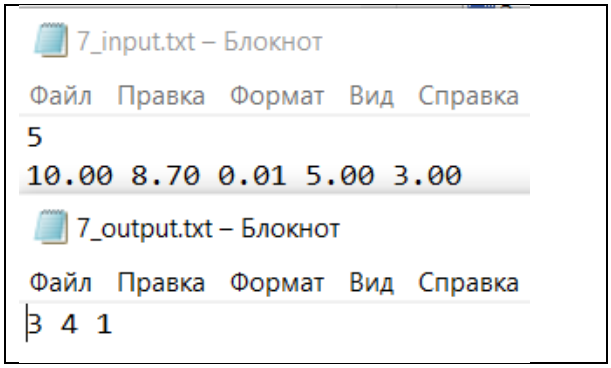
Текстовое объяснение решения:

Для решения задачи было использовано 2 функции. Первая из них (insertion_sort) сортирует массив по возрастанию. Методом вставками. Вторая (search_number) ищет определенный элемент в первоначальном массиве и возвращает его место (индекс+1).

Сначала мы читаем массив из файла и клонируем его, так как нам необходимо сохранить первоначальной порядок элементов, чтобы обратится к нему далее. После этого мы сортируем массив по возрастанию, определяем значение 1, среднего и последнего элемента и с помощью второй функции и клонированного массива опредаеляем его первоначальное место.

Также у меня возникли проблемы с временем выполнения программы, поэтому была попытка использовать сортировку выбором. Но там тоже решение на максимальных значения за временное ограничение.

Результат работы кода на примерах из текста задачи:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0012840999999999964 second	13.796875 mb
Пример из задачи	0.0028587999999999947 second	13.796875 mb
Верхняя граница диапазона значений входных данных из текста задачи	2.2766966 second	14.04296875 mb

Вывод по задаче: Я научилась использовать функции сортировки для решение конкретных задач. Также было замечена, что функции сортировки вставкой и выбором работают достаточно медленно, как и все другие функции с вычислительной сложностью – $O(n^2)$. Поэтом для сортировки больших массивов стоит использовать более эффективные сортировки.

Дополнительные задачи

Вывод

Я узнала, как научилась реализовывать некоторые алгоритмы сортировки массивов вычислительной сложность равной n^2, такие как: сортировки выбором и вставками, а также пузырьковая сортировка. Я протестировала их на эффективность и заметила необходимость изучение более быстрых алгоритмов сортировки.