МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК

Практическое занятие № 7.

Сортировки массивов.

Вариант 3.

Выполнил студент:

Герасимов Константин Сергеевич

МОиАИС

1. **Цель работы**

Получение практических навыков по использованию алгоритмов сортировки одномерных массивов (списков).

1. **Решение задачи 1**

***Постановка задачи***

Используя алгоритмы сортировки, напишите программу сортировки массива из n элементов (n вводиться с клавиатуры) следующим образом: отсортировать массив по убыванию с помощью алгоритма сортировки «пузырек» до элемента массива, индекс которого вводится с клавиатуры (необходимо осуществить проверку введенного индекса), оставшуюся часть массива отсортировать по возрастанию с помощью алгоритма сортировки массива выбором; заполнение массива осуществить случайным образом.

***Текст программы на языке Python***

from random import randint

n = int(input("введите длину массива "))

massive = [0]\*n

for i in range(0, len(massive) - 1):

massive[i] = randint(-100, 100)

k = int(input("введите количество отсортированных элементов "))

for i in range(0, k):

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

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

massive[j], massive[j+1] = massive[j+1], massive[j]

print(f'{massive} - прошла сортировка пузырьком до {k} элемента')

for i in range(k, len(massive) - 1):

m = i

j = i + 1

while j < len(massive):

if massive[j] < massive[m]:

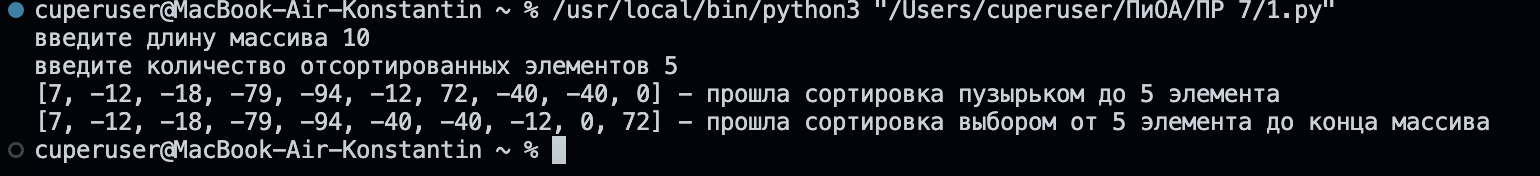
m = j

j = j + 1

massive[i], massive[m] = massive[m], massive[i]

print(f'{massive} - прошла сортировка выбором от {k} элемента до конца массива')

***Результат тестирования программы на языке Python***

****

1. **Решение задачи 2**

***Постановка задачи***

Напишите программу, которая выполняет следующие функции:

- заполнение элементов массива вещественными числами с заданной размерностью случайным образом;

- сортировку массива каждым из 3 способов (пузырьковая сортировка, сортировка выбором, сортировка вставкой);

- подсчет времени сортировки и количества перестановок.

***Аппаратная часть эксперимента***

Компьютер: Apple MacBook Air 2020

Процессор: M1 3.2 ГГц, 8 ядер

Оперативная память: 8 ГБ

Накопитель: SSD 512 ГБ

***Текст программы на языке Python***

from random import randint

from datetime import \*

n = int(input('введите длину массива '))

massive = [0]\*n

k1, k2, k3 = 0, 0, 0

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

massive[i] = randint(-10000, 10000)/randint(1, 100)

massive1 = copy(massive)

massive2 = copy(massive)

def bubble(array):

global k1

start\_time = datetime.now()

for i in range(0, len(array) - 1):

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

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

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

k1 += 1

end\_time = datetime.now()

bubble\_time = end\_time - start\_time

return bubble\_time

def selection(array):

global k2

start\_time = datetime.now()

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

m = i

j = i + 1

while j < len(array):

if array[j] < array[m]:

m = j

j = j + 1

array[i], array[m] = array[m], array[i]

k2 += 1

end\_time = datetime.now()

selection\_time = end\_time - start\_time

return selection\_time

def insertion(array):

global k3

start\_time = datetime.now()

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

key = array[i]

j = i-1

while j >=0 and key < array[j] :

array[j+1] = array[j]

j -= 1

array[j+1] = key

k3 += 1

end\_time = datetime.now()

insertion\_time = end\_time - start\_time

return insertion\_time

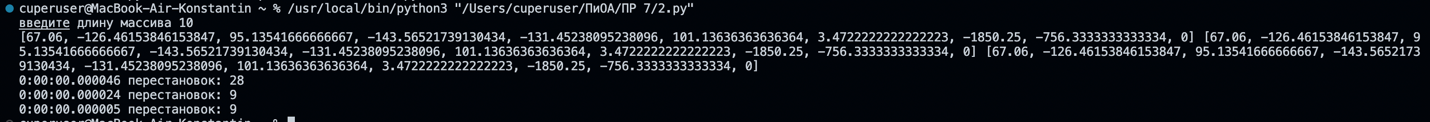
print(massive)

print(f'{bubble(massive)} перестановок: {k1}\

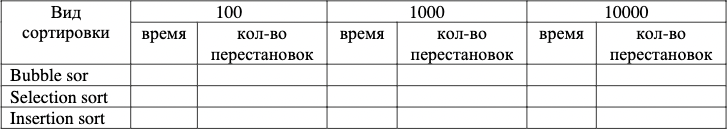
\n{selection(massive1)} перестановок: {k2}\

\n{insertion(massive2)} перестановок: {k3}')

***Результат тестирования программы на языке Python***

******

Провести эксперимент сортировки массива со следующим количеством элементов: 100, 1000 и 10000. Для проведения эксперимента необходимо произвести по 5 запусков каждого алгоритма и выбрать наилучшее время. Выбранный наилучший результат занести в таблицу. Сортировку осуществлять с одним и тем же набором данных.

******

1.740898

3.458067

4.632157

9999

9999

24917888

0.016399

0.036312

0.077064

999

999

245888

0.000518

0.000949

0.001297

99

99

2638

***Вывод***

Сортировка пузырьком при размерности массива 100 среди 5ти запусков имела лучшее время 00.001297 и выполнила 2638 перестановок; при размерности массива 1000 среди 5ти запусков имела лучшее время 00.077064 и выполнила 245888 перестановок; при размерности массива 10000 имела лучшее время 04.632157 и выполнила 24917888 перестановок.

Сортировка выбором при размерности массива 100 среди 5ти запусков имела лучшее время 00.000949 и выполнила 99 перестановок; при размерности массива 1000 среди 5ти запусков имела лучшее время 00.036312 и выполнила 999 перестановок; при размерности массива 10000 имела лучшее время 03.458067 и выполнила 9999 перестановок.

Сортировка вставкой при размерности массива 100 среди 5ти запусков имела лучшее время 00.000581 и выполнила 99 перестановок; при размерности массива 1000 среди 5ти запусков имела лучшее время 00.016399 и выполнила 999 перестановок; при размерности массива 10000 имела лучшее время 01.740898 и выполнила 9999 перестановок.

Общий вывод: Сортировка вставкой показала самые лучшие результаты среди всех сортировок: она имеет наименьшее время выполнения и задействует столько же операций перестановки, что и сортировка выбором.