МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
 федеральное государственное бюджетное образовательное учреждение

высшего образования

АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ЦИФРОВЫХ ТЕХНОЛОГИЙ, ЭЛЕКТРОНИКИ И ФИЗИКИ

Кафедра информационной безопасности

«РЕШЕНИЕ ЗАДАНИЯ № 4»

Отчет по лабораторной работе по дисциплине

«Технологии и методы программирования»

Выполнил студент

2 курса, 5.107-1 группы

Кулеба Никита Александрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Проверил к.ф.-м.н., доцент,

Лепендин Андрей Александрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Барнаул 2023

# ЗАДАНИЕ ПО ЛАБОРАТОРНОЙ РАБОТЕ (ТЕКСТ ЗАДАЧИ)

**Цель работы:** реализация алгоритма из лабораторного задания № 4

**Задачи:**

В данной лабораторной работе вам предстоит реализовать метод сортировки произвольных последовательностей объектов, хранящихся в списке, оценить опытным путем среднее время его работы в зависимости от размера сортируемой последовательности.

**Общая задача сортировки** может быть сформирована следующим образом: пусть дана последовательность некоторых объектов a0,a1,...,an−1 Необходимо переставить эти объекты таким образом, что в новой последовательности каждый последующий будет меньше (или больше) предыдущего. В данной работе предполагается, что по умолчанию последовательность сортируется по нарастанию с возможностью замены операции сравнения своим вариантом.

# ИСХОДНЫЙ КОД ПРОГРАММЫ

*import* timeit

*import* random

*import* numpy

*import* matplotlib *as* mpl

*import* matplotlib.pyplot *as* plt

*from* typing *import* List

generate\_list = lambda *n*: [random.randint(0, *n*) *for* \_ *in* range(*n*)]

def oddEvenSort(*array*: List) -> List:

    arr\_len = len(*array*)

*for* i *in* range(arr\_len):

        start\_point = 0 *if* i % 2 == 0 *else* 1

*for* j *in* range(start\_point, arr\_len - 1, 2):

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

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

*return* *array*

def speed\_tester() -> List:

    speed = []

*for* i *in* range(1\_000, 11\_000, 1\_000):

        arr = generate\_list(i)

        speed.append(timeit.Timer(lambda: oddEvenSort(arr)).repeat(10, 1))

*return* speed

speed = speed\_tester()

np\_arr = numpy.array(speed)

np\_average = numpy.mean(np\_arr, *axis*=1)

np\_std = numpy.std(np\_arr, *axis*=1)

ax = plt.subplot()

x\_line = range(1\_000, 11\_000, 1\_000)

ax.plot(x\_line, np\_average, *label*="вектор средних значений времени работы")

ax.plot(x\_line, np\_std, *label*="среднеквадратичное отклонение времени работы")

ax.set\_xlabel("кол-во сортируемых элементов")

ax.set\_ylabel("Время работы")

ax.legend()

ax.grid()

# 

# Теоретическое время сложности: O(n2)

# Среднее время: O(n2)

Мы оценили опытным путем среднее время работы в зависимости от размера сортируемой последовательности.