Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
дисциплины «Алгоритмизация»

Выполнил:
Кожуховский Виктор Андреевич
1 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных систем
», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Порядок выполнения работы:

Написал программу пузырьковой сортировки массива, автоматического заполнения массива, расчёта ста точек, показывающих время сортировки массива в худшем и среднем случае, вывода графиков, составленных из этих точек, и вывода коэффициентов при параболе, составленной методом наименьших квадратов:

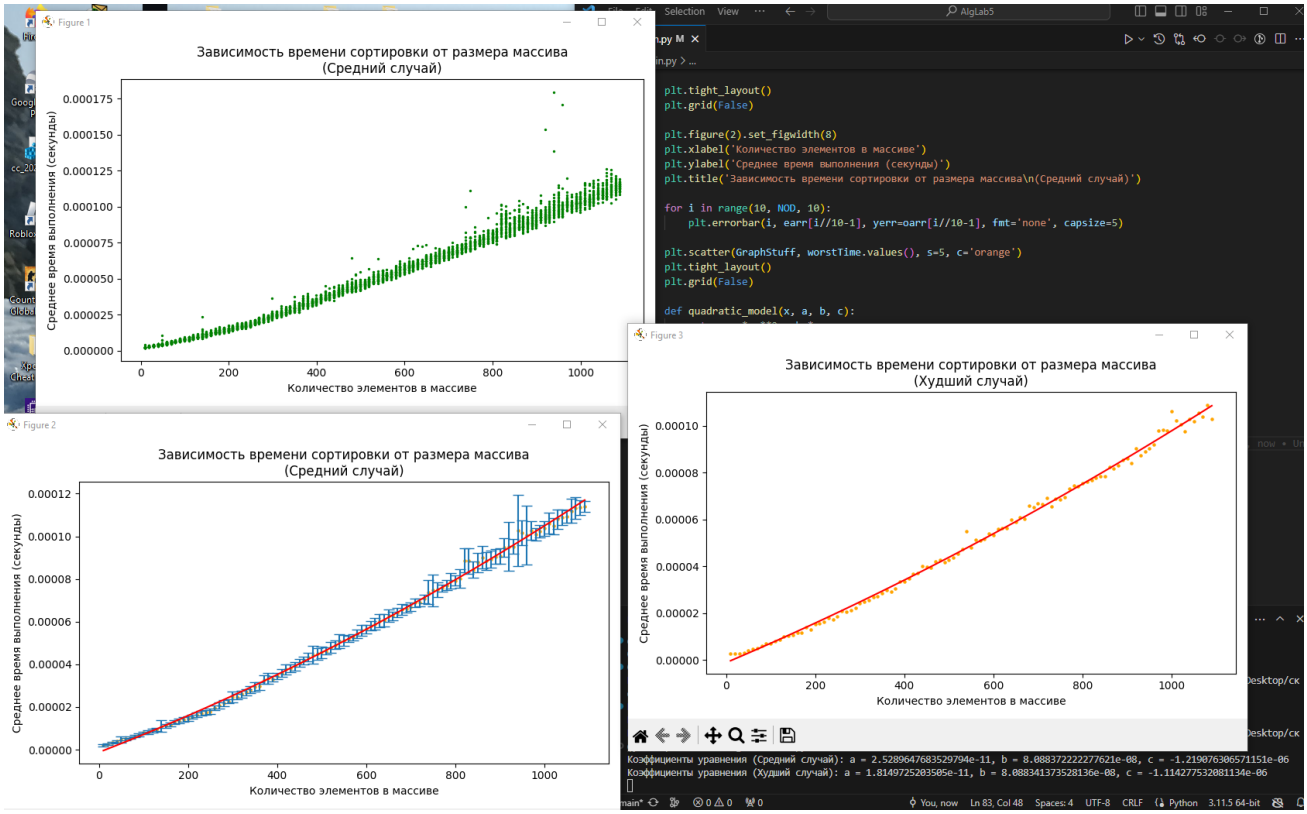

Рисунок 1. Графики сортировки массива в худшем и среднем случае и вывод коэффициентов

```python
import matplotlib.pyplot as plt
import numpy as np
import math as m
from scipy.optimize import curve_fit

NumberOfDots = 100
e, o = 0, 0
NOD = (NumberOfDots + 10) * 10
mid = 0
a = {}
worstTime = {}
GraphStuff = [i for i in range(10, NOD, 10)]
oarr = []
earr = []
something = {}

def sort(a, n):
    for j in range(1, n-1):
        f=0
        for i in range(n-1-j):
            if (a[i]>a[i+1]):
                a[i], a[i+1] = a[i+1], a[i]
        if f == 0:
            return 0

def fillArr(numOfEl):
    a.clear()
    for i in range(numOfEl):
        a[i] = random.randint(0, 100000)

def fillArrWorst(numOfEl):
    a.clear()
    for i in range(numOfEl):
        a[i] = 100 - i

plt.figure(1).set_figwidth(8)
plt.xlabel('Количество элементов в массиве')
plt.ylabel('Среднее время выполнения (секунды)')
plt.title('Зависимость времени сортировки от размера массива\n(Средний случай)')

for i in range(10, NOD, 10):
    fillArr(i)
    mid, e, o = 0, 0, 0
    for j in range(30):
        worstTime[i] = timeit.timeit(lambda:sort(a, i+1), number = 1)
        mid += worstTime[i]
        e += 1 / 30 * worstTime[i]
        plt.scatter(i-1, worstTime[i], s=2, c='green')
        something[j] = worstTime[i]
    worstTime[i] = mid / 30
    for s in something:
        o += 1 / 29 * (something[s] - e) ** 2
    o = m.sqrt(o)
    oarr.append(o)
    earr.append(e)

plt.tight_layout()
plt.grid(False)

plt.figure(2).set_figwidth(8)
plt.xlabel('Количество элементов в массиве')
plt.ylabel('Среднее время выполнения (секунды)')
plt.title('Зависимость времени сортировки от размера массива\n(Средний случай)')

for i in range(10, NOD, 10):
    plt.errorbar(i, earr[i//10-1], yerr=oarr[i//10-1], fmt='none', capsize=5)

plt.scatter(GraphStuff, worstTime.values(), s=5, c='orange')
plt.tight_layout()
plt.grid(False)      You, 6 days ago • Not finished

def quadratic_model(x, a, b, c):
    return a * x**2 + b * x + c

x_data = np.array(GraphStuff)
y_data = np.array(list(worstTime.values()))

params, covariance = curve_fit(quadratic_model, x_data, y_data)

a_fit, b_fit, c_fit = params
print(f'Коэффициенты уравнения (Средний случай): a = {a_fit}, b = {b_fit}, c = {c_fit}')

x_fit = np.linspace(min(x_data), max(x_data), 100)
y_fit = quadratic_model(x_fit, *params)

plt.plot(x_fit, y_fit, 'r-', label='Quadratic Fit')

#Case 3
```

Рисунок 2. Код программы

Вывод: в результате выполнения лабораторной работы был изучен алгоритм пузырьковой сортировки массива и проведено исследование зависимости времени этой сортировки от количества элементов в массиве, в результате которого было установлено, что алгоритмы пузырьковой сортировки массива с увеличением количества элементов в массиве занимают больше времени.