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 №4
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 поиска наименьшего и наибольшего элемента в массиве, автоматического заполнения массива, расчёта тысячи точек, показывающих время поиска элемента в массиве в худшем и среднем случае, вывода графиков, составленных из этих точек, и подсчета корреляции:

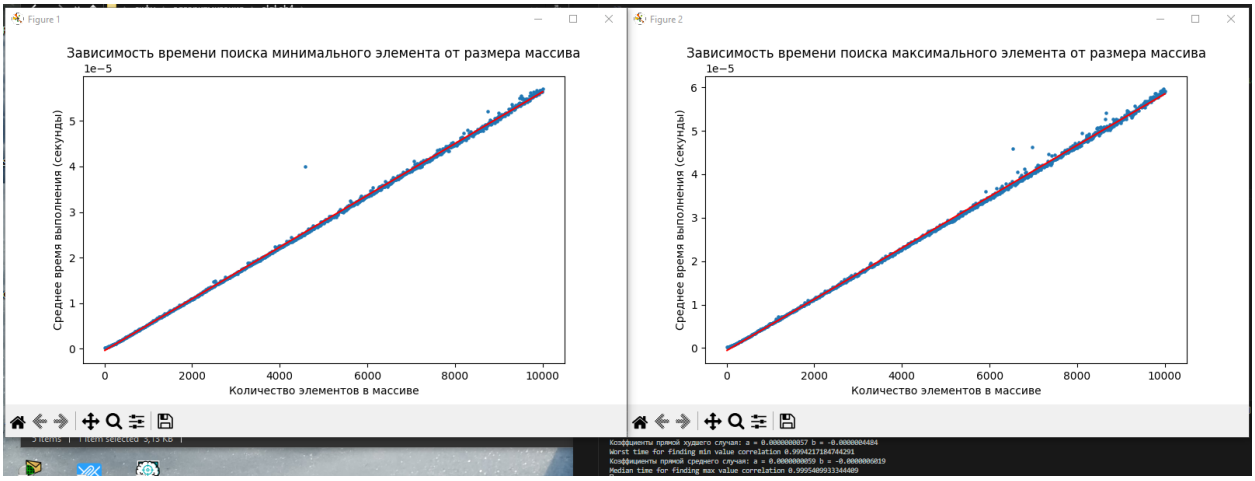
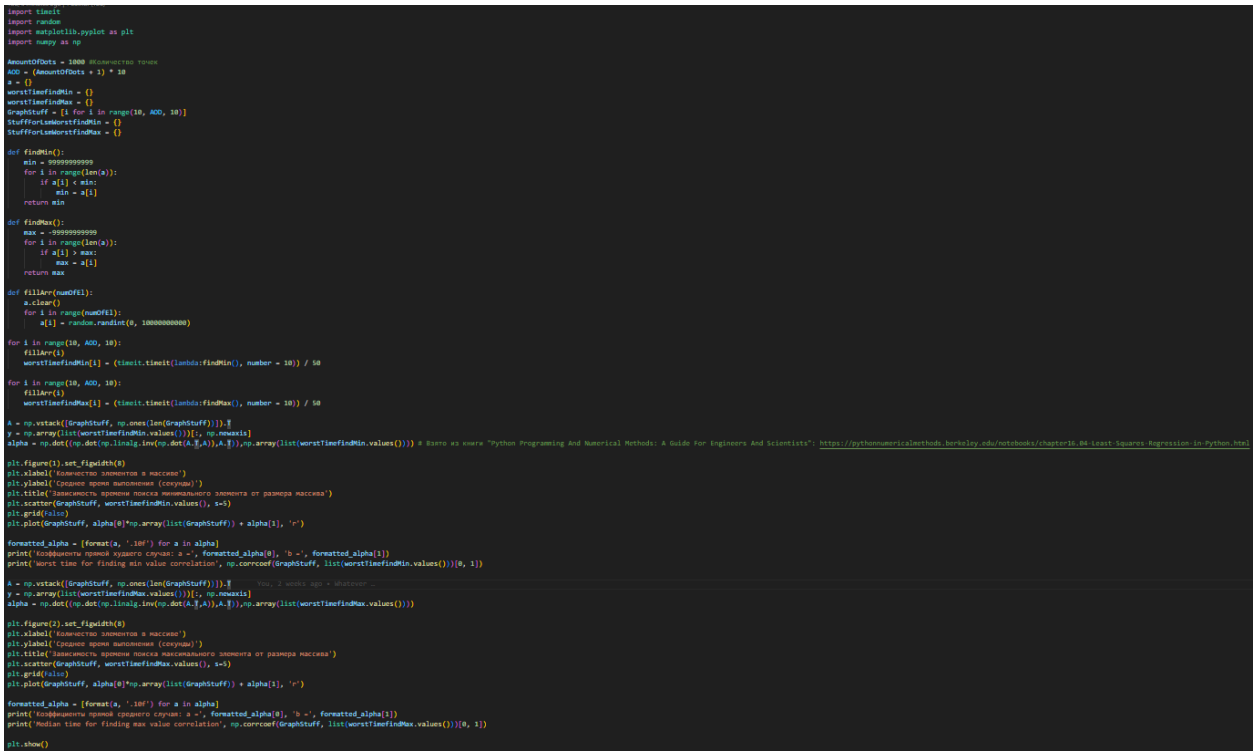


Рисунок 1. Графики времени поиска наименьшего и наибольшего элемента в массиве в худшем и среднем случае и вывод корреляции



```
import timeit
import random
import matplotlib.pyplot as plt
import numpy as np

AmountOfDots = 1000 #Количество точек
ADD = (AmountOfDots + 1) * 10
a = []
worstTimefindMin = []
worstTimefindMax = []
GraphStuff = [i for i in range(10, ADD, 10)]
StuffForLsmWorstfindMin = []
StuffForLsmWorstfindMax = []

def findMin():
    min = 9999999999
    for i in range(len(a)):
        if a[i] < min:
            min = a[i]
    return min

def findMax():
    max = -9999999999
    for i in range(len(a)):
        if a[i] > max:
            max = a[i]
    return max

def fillArr(numOfEl):
    a.clear()
    for i in range(numOfEl):
        a[i] = random.randint(0, 1000000000)

for i in range(10, ADD, 10):
    fillArr(i)
    worstTimefindMin[i] = (timeit.timeit(lambda:findMin(), number = 10)) / 50

for i in range(10, ADD, 10):
    fillArr(i)
    worstTimefindMax[i] = (timeit.timeit(lambda:findMax(), number = 10)) / 50

A = np.vstack([GraphStuff, np.ones(len(GraphStuff))]).T
y = np.array(list(worstTimefindMin.values()))[:, np.newaxis]
alpha = np.dot((np.dot(np.linalg.inv(np.dot(A.T,A)),A.T)),np.array(list(worstTimefindMin.values()))) # Взято из книги "Python Programming And Numerical Methods: A Guide For Engineers And Scientists": https://pythonnumericalmethods.berkeley.edu/notebooks/chapter16.04-Least-Squares-Regression-in-Python.html

plt.figure(1).set_figwidth(8)
plt.xlabel('Количество элементов в массиве')
plt.ylabel('Среднее время выполнения (секунды)')
plt.title('Зависимость времени поиска минимального элемента от размера массива')
plt.scatter(GraphStuff, worstTimefindMin.values(), s=5)
plt.grid(False)
plt.plot(GraphStuff, alpha[0]*np.array(list(GraphStuff)) + alpha[1], 'r')

formatted_alpha = [format(a, '.10f') for a in alpha]
print('Коэффициенты прямой худшего случая: a =', formatted_alpha[0], 'b =', formatted_alpha[1])
print('Worst time for finding min value correlation', np.corrcoef(GraphStuff, list(worstTimefindMin.values()))[0, 1])

A = np.vstack([GraphStuff, np.ones(len(GraphStuff))]).T
y = np.array(list(worstTimefindMax.values()))[:, np.newaxis]
alpha = np.dot((np.dot(np.linalg.inv(np.dot(A.T,A)),A.T)),np.array(list(worstTimefindMax.values())))

plt.figure(2).set_figwidth(8)
plt.xlabel('Количество элементов в массиве')
plt.ylabel('Среднее время выполнения (секунды)')
plt.title('Зависимость времени поиска максимального элемента от размера массива')
plt.scatter(GraphStuff, worstTimefindMax.values(), s=5)
plt.grid(False)
plt.plot(GraphStuff, alpha[0]*np.array(list(GraphStuff)) + alpha[1], 'r')

formatted_alpha = [format(a, '.10f') for a in alpha]
print('Коэффициенты прямой среднего случая: a =', formatted_alpha[0], 'b =', formatted_alpha[1])
print('Median time for finding max value correlation', np.corrcoef(GraphStuff, list(worstTimefindMax.values()))[0, 1])

plt.show()
```

Рисунок 2. Код программы

Вывод: в результате выполнения лабораторной работы был изучен алгоритм линейного поиска наименьшего и наибольшего элемента в массиве и проведено исследование зависимости времени этого поиска от количества

элементов в массиве, в результате которого было установлено, что алгоритмы линейного поиска максимального и минимального элемента массива не зависимо от количества элементов в массиве занимают равное количество времени и это количество времени возрастает с добавлением большего числа элементов в массив.