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 №3
## 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 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:

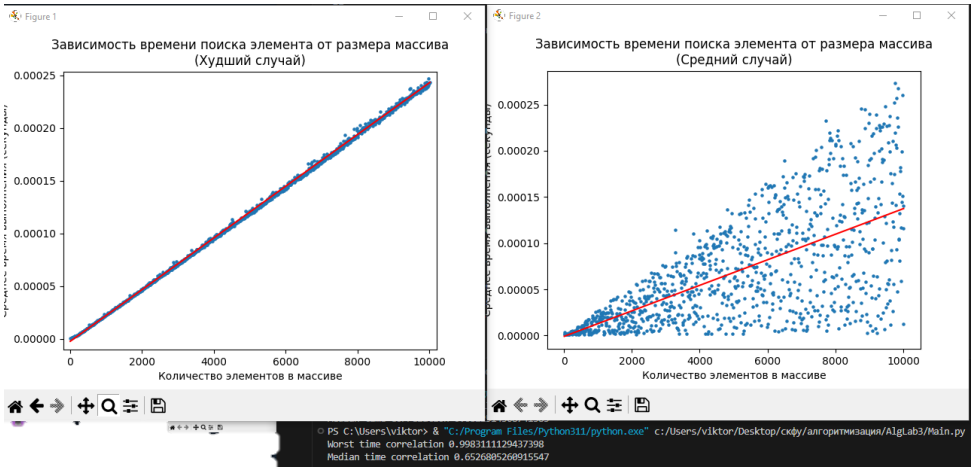


Рисунок 1. Графики времени поиска элемента в массиве в худшем и среднем случае и вывод корреляции

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

a = {}
worstTime = {}
medianTime = {}
GraphStuff = [i for i in range(10, 10020, 10)]
StuffForLsmWorst = {}
StuffForLsmMedian = {}

def find(n):
    for i in range(len(a)):
        if a[i] == n:
            return i
    return -1

def fillArr(numOfEl):
    a.clear()
    for i in range(numOfEl):
        a[i] = random.randint(0, 1000000)

for i in range(10, 10020, 10):
    fillArr(i)
    worstTime[i] = (timeit.timeit(lambda:find(10000000), number = 10)) / 10

for i in range(10, 10020, 10):
    fillArr(i)
    medianTime[i] = timeit.timeit(lambda:find(a[int(random.randint(1, i - 1))]), number = 1)

A = np.vstack([GraphStuff, np.ones(len(GraphStuff))]).T
y = np.array(list(worstTime.values()))[:, np.newaxis]
alpha = np.dot((np.dot(np.linalg.inv(np.dot(A.T,A)),A.T)),np.array(list(worstTime.values()))) # Взято из книги "Python Programming A

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

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

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

print('Worst time correlation', np.corrcoef(GraphStuff, list(worstTime.values()))[0, 1])
print('Median time correlation', np.corrcoef(GraphStuff, list(medianTime.values()))[0, 1])

plt.show()
```

Рисунок 2. Код программы

Вывод: в результате выполнения лабораторной работы был изучен алгоритм линейного поиска и проведено исследование зависимости времени поиска от количества элементов в массиве.