

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:
Епифанов Алексей Александрович
2 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:

1. Написал программу, которая выводит в виде графика зависимость времени выполнения функции линейного поиска от размера массива, я рассмотрел 2 случая: средний и худший, и соответственно моя программа вывела 2 графика. Также на графике есть прямая, построенная по методу меньших квадратов, а в консоль выводится коэффициент парной корреляции.

```
... + Python ... > X
/usr/local/bin/python3 /Users/aleksej
pifanov/Desktop/нары/нары_3_сем/pytgit
/algorithm3/program/main.py

aleksejepifanov@MacBook-Pro algorithm3
% /usr/local/bin/python3 /Users/alekse
jepifanov/Desktop/нары/нары_3_сем/pytgit
it/algorithm3/program/main.py
Коэффициент корреляции в первом случае
= 0.6649685390305776
а во втором случае = 0.980998981883674
aleksejepifanov@MacBook-Pro algorithm3
%

main.py M X Параметры
program > main.py > ...
1 import random as rnd
2 from statistics import correlation
3 import matplotlib.pyplot as plt
4 import numpy as np
5 import timeit
6
7
8 def find(a, b, len):
9     for i in range(len):
10         if b == a[i]:
11             return i
12     return -1
13
14
15 def create_graph(b, c, aur, bur):
16     y_values = np.linspace(0, max(c), num=5)
17     x_values = np.linspace(0, b[-1], num=11)
18     plt.scatter(b, c, s=5)
19
20     y_line = aur * np.array(b) + bur
21     plt.plot(b, y_line, color='red')
22     plt.title("График")
23     plt.xlabel("X-ось")
24     plt.ylabel("Y-ось")
25     plt.xticks(x_values)
26     plt.yticks(y_values)
27     correlation_coefficient = np.corrcoef(c, y_line)[0, 1]
28     return correlation_coefficient
29
30
31 correlation = []
32 # Цикл нужен для создания двух графиков, один при среднем случае, второй при худшем
33 for iter in [1, 2]:
34     x = []
35     time = []
36     x2 = []
37     xtime = []
38     randmax = 1000000
39     for i in range(10, 10001, 10):
40         x.append(i)
41         a = [rnd.randint(1, randmax) for j in range(i)]
42         if iter == 1:
43             b = a[rnd.randint(1, len(a)-1)]
44         else:
45             b = randmax+1
46         timer = timeit.timeit(lambda: find(a, b, i), number=1)
47         time.append(timer)
48         index = find(a, b, i)
49         for i, j in zip(x, time):
50             x2.append(i**2)
51             xtime.append(i*j)
52     # Вычисление коэффициентов в системе уравнений метода наименьших квадратов
53     sx = sum(x)
54     stime = sum(time)
55     sx2 = sum(x2)
56     sxtime = sum(xtime)
57     n = len(x)
58     # k - это коэффициент, при котором вычитание
59     # из первого уравнения второго,
60     # умноженного на него, приводит к нулю в коэффициенте при x.
61     # Таким образом, мы сможем вычислить свободный коэффициент.
62     k = sx2/sx
63     # bur - это свободный коэффициент
64     bur = (sxtime - k*stime)/(sx-k*n)
65     # aur - это коэффициент при x
66     aur = (stime - bur*n)/sx
67     # Создание графических окон
68     plt.figure(iter)
69     # Создание графиков
70     correlation.append(create_graph(x, time, aur, bur))
71
72 print("Коэффициент корреляции в первом случае =",
73       correlation[0], "\nа во втором случае =", correlation[1])
74
75 # Показ графиков
76 plt.show()
```

Рисунок 1. Код программы и вывод в консоли

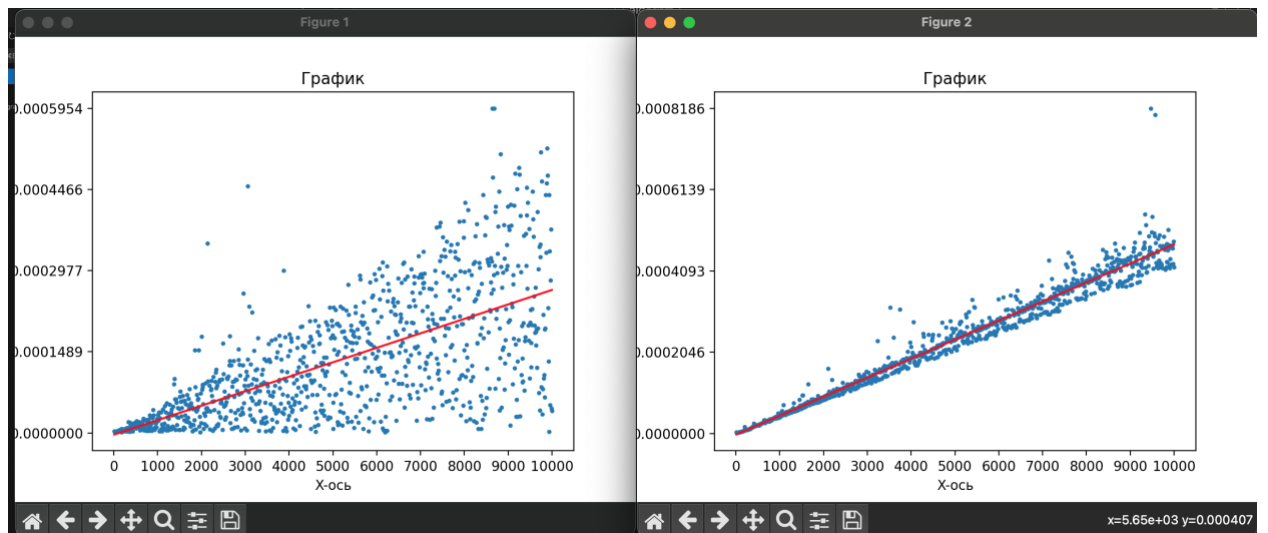


Рисунок 2. Вывод графиков: слева средний случай, справа худший

Вывод: выполняя данную работу, я изучил метод линейного поиска, понял, что при увеличении размера массива, время выполнения данного метода линейно растет, пусть и с небольшими отклонениями. В среднем случае из-за того, что метод может проверить почти все элементы массива и где-то в конце найти нужны, а может найти нужный и в самом начале разброс по времени между соседними проверками получается намного больше, чем в худшем, так как там алгоритм обязан проверить все элементы массива.