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 №9
дисциплины «Алгоритмизация»

Выполнил:
Дзуев Альберт Мухаметович
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 Р.А., доцент кафедры инфокоммуникаций

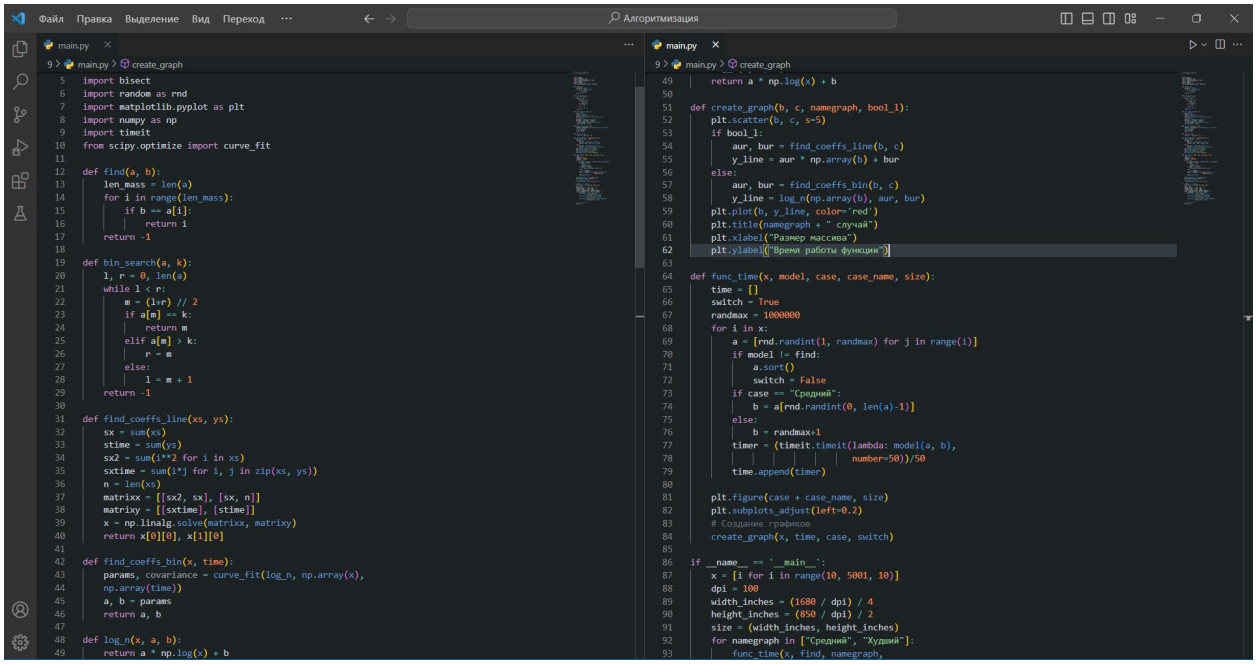
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Порядок выполнения работы:

1. Написал программу сравнения функций поиска, написанных мной: линейный поиск и бинарный поиск, а так же эти функции сравнил со встроенной в Python функцией бинарного поиска bisect:



```python
import bisect
import random as rnd
import matplotlib.pyplot as plt
import numpy as np
import timeit
from scipy.optimize import curve_fit

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

def bin_search(a, k):
    l, r = 0, len(a)
    while l < r:
        m = (l+r) // 2
        if a[m] == k:
            return m
        elif a[m] > k:
            r = m
        else:
            l = m + 1
    return -1

def find_coeffs_line(xs, ys):
    sx = sum(xs)
    stime = sum(ys)
    sx2 = sum(i**2 for i in xs)
    sxtime = sum(i*j for i, j in zip(xs, ys))
    n = len(xs)
    matrixx = [[sx2, sx], [sx, n]]
    matrixy = [[sxtime], [stime]]
    x = np.linalg.solve(matrixx, matrixy)
    return x[0][0], x[1][0]

def find_coeffs_bin(x, time):
    params, covariance = curve_fit(log_n, np.array(x),
    np.array(time))
    a, b = params
    return a, b

def log_n(x, a, b):
    return a * np.log(x) + b

def create_graph(b, c, namegraph, bool_l):
    plt.scatter(b, c, s=5)
    if bool_l:
        aur, bur = find_coeffs_line(b, c)
        y_line = aur * np.array(b) + bur
    else:
        aur, bur = find_coeffs_bin(b, c)
        y_line = log_n(np.array(b), aur, bur)
    plt.plot(b, y_line, color='red')
    plt.title(namegraph + " случай")
    plt.xlabel("Размер массива")
    plt.ylabel("Время работы функции")

def func_time(x, model, case, case_name, size):
    time = []
    switch = True
    randmax = 1000000
    for i in x:
        a = [rnd.randint(1, randmax) for j in range(i)]
        if model != find:
            a.sort()
            switch = False
        if case == "Средний":
            b = a[rnd.randint(0, len(a)-1)]
        else:
            b = randmax+1
        timer = (timeit.timeit(lambda: model(a, b),
                               number=50))/50
        time.append(timer)

    plt.figure(case + case_name, size)
    plt.subplots_adjust(left=0.2)
    # Создание графиков
    create_graph(x, time, case, switch)

if __name__ == '__main__':
    x = [i for i in range(10, 5001, 10)]
    dpi = 100
    width_inches = (1680 / dpi) / 4
    height_inches = (850 / dpi) / 2
    size = (width_inches, height_inches)
    for namegraph in ["Средний", "Худший"]:
        func_time(x, find, namegraph,
```

Рисунок 1. Код программы

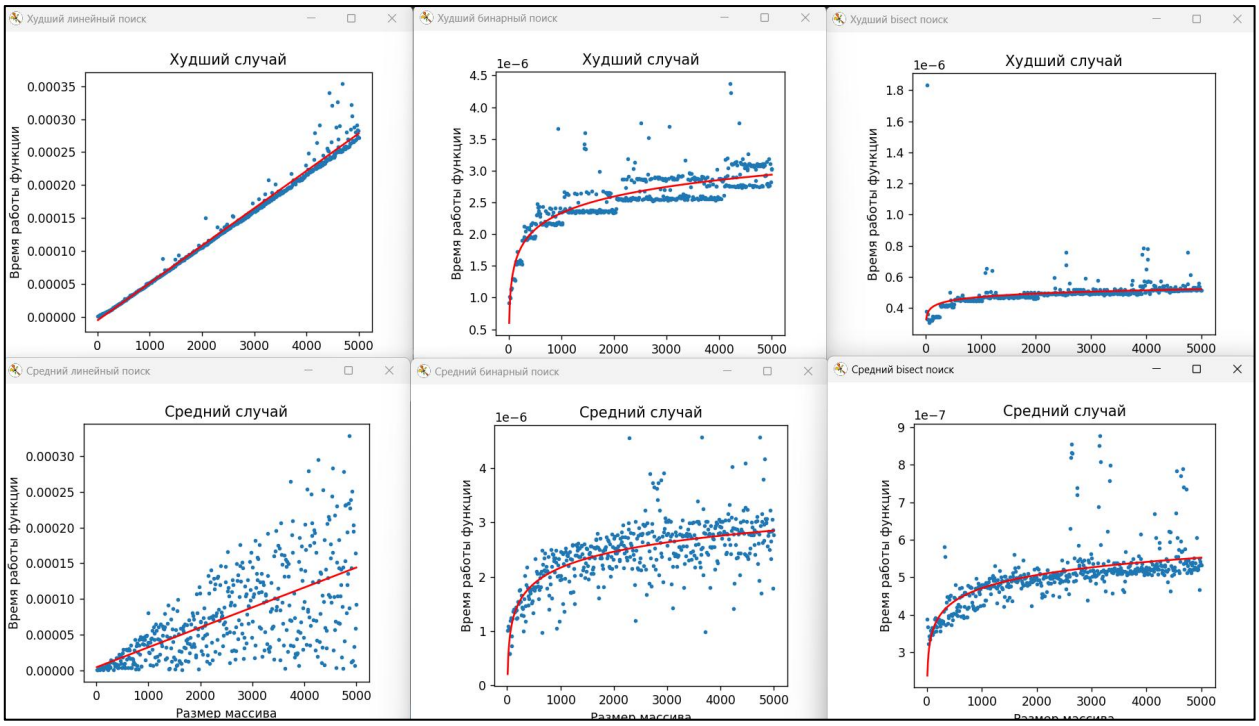


Рисунок 2. Результат работы программы

Вывод: в ходе выполнения лабораторной работы были исследованы функции поиска в двух случаях: среднем и худшем. Лучшей оказалась функция, встроенная в Python: bisect.bisect_left(), а худшей – линейный поиск.