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 ПРАКТИЧЕСКОЙ РАБОТЕ №3
дисциплины «Алгоритмизация»
Вариант 9

Выполнил:
Дудкин Константин Александрович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»
направление «Программное
обеспечение средств вычислительной
техники и автоматизированных
систем»,
очная форма обучения

_____
(подпись)

Руководитель практики:
доцент кафедры инфокоммуникаций,
кандидат технических наук
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Порядок выполнения работы

Изучил алгоритмы линейного поиска, составил программу для анализа данного алгоритма. Провел исследование линейного поиска при лучшем и худшем случае. Построил линейную зависимость с помощью метода наименьших квадратов и нашел коэффициент парной корреляции:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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

# Функции (линейный поиск, фиксация времени)
1 usage  new *
def linear_search(arr, target):  # функция для алгоритма линейного поиска
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1


2 usages  new *
def search_time_experiment(arr_size, target_position):  # функция для записывания времени экспериментов
    arr = list(range(arr_size))
    target = arr[target_position]

    execution_time = timeit.timeit(lambda: linear_search(arr, target), number=1000)
    return execution_time

```

Рисунок 1. Полный код программы (часть 1)

```python
if __name__ == '__main__':

    # Параметры для исследования
    array_sizes = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
                   2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000]  # Параметры для массивов
    num_experiments = 5  # Кол-во экспериментов
    WorstCase_execution_times = []  # Массив для времени выполнения худшего случая
    AvgCase_execution_times = []  # Массив для времени выполнения среднего случая


    # Подготовка и проведение экспериментов
    for size in array_sizes:
        WorstCase_times = []
        AvgCase_times = []

        for _ in range(num_experiments):
            WorstCase_time = search_time_experiment(size, size - 1)
            AvgCase_time = search_time_experiment(size, size // 2)

            WorstCase_times.append(WorstCase_time)
            AvgCase_times.append(AvgCase_time)

        # Вычисление среднего времени выполнения алгоритма
        avg_worst_case_time = sum(WorstCase_times) / num_experiments
        avg_average_case_time = sum(AvgCase_times) / num_experiments

        WorstCase_execution_times.append(avg_worst_case_time)
        AvgCase_execution_times.append(avg_average_case_time)

```

Рисунок 2. Полный код программы (часть 2)

```python
    # Выполнение линейной регрессии и расчет коэффициента корреляции для обоих случаев
    x = np.array(array_sizes)
    y_worst = np.array(WorstCase_execution_times)
    y_avg = np.array(AvgCase_execution_times)

    A = np.vstack([x, np.ones(len(x))]).T
    a_worst, b_worst = np.linalg.lstsq(A, y_worst, rcond=None)[0]
    a_avg, b_avg = np.linalg.lstsq(A, y_avg, rcond=None)[0]

    correlation_coefficient_worst = np.corrcoef(x, y_worst)[0, 1] ** 2
    correlation_coefficient_avg = np.corrcoef(x, y_avg)[0, 1] ** 2

    # Визуализация
    plt.figure(figsize=(12, 6))

    plt.subplot( *args: 1, 2, 1)
    plt.plot( *args: x, y_worst, 'o', label='Худший случай')
    plt.plot( *args: x, a_worst * x + b_worst, 'r', label=f'Линейная зависимость (R^2={correlation_coefficient_worst:.5f})')
    plt.xlabel('Размер массива')
    plt.ylabel('Время выполнения (секунды)')
    plt.title('Аналитика худшего случая')
    plt.legend()

    plt.subplot( *args: 1, 2, 2)
    plt.plot( *args: x, y_avg, 'o', label='Средний случай')
    plt.plot( *args: x, a_avg * x + b_avg, 'g', label=f'Линейная зависимость (R^2={correlation_coefficient_avg:.5f})')
    plt.xlabel('Размер массива')
    plt.ylabel('Время выполнения (секунды)')
    plt.title('Аналитика среднего случая')
    plt.legend()

    plt.tight_layout()
    plt.show()
```

Рисунок 3. Полный код программы (часть 3)

```python
    # Вывод в консоль
    print(f"Худший случай: Линейная зависимость: y = {a_worst:.5f} * x + {b_worst:.5f}")
    print(f"Худший случай: Коэффициент корреляции: {correlation_coefficient_worst:.5f}")

    print(f"Средний случай: Средний случай: y = {a_avg:.5f} * x + {b_avg:.5f}")
    print(f"Средний случай: Коэффициент корреляции: {correlation_coefficient_avg:.5f}")
```

Рисунок 4. Полный код программы (часть 4)

```
/usr/bin/python3.11 /home/code_ra1der/git/Algorithm_PR3/Task.py
Худший случай: Линейная зависимость: y = 0.00002 * x + -0.00121
Худший случай: Коэффициент корреляции: 0.99997
Средний случай: Средний случай: y = 0.00001 * x + -0.00089
Средний случай: Коэффициент корреляции: 0.99993

Process finished with exit code 0
```

Рисунок 5. Вывод результата в консоль

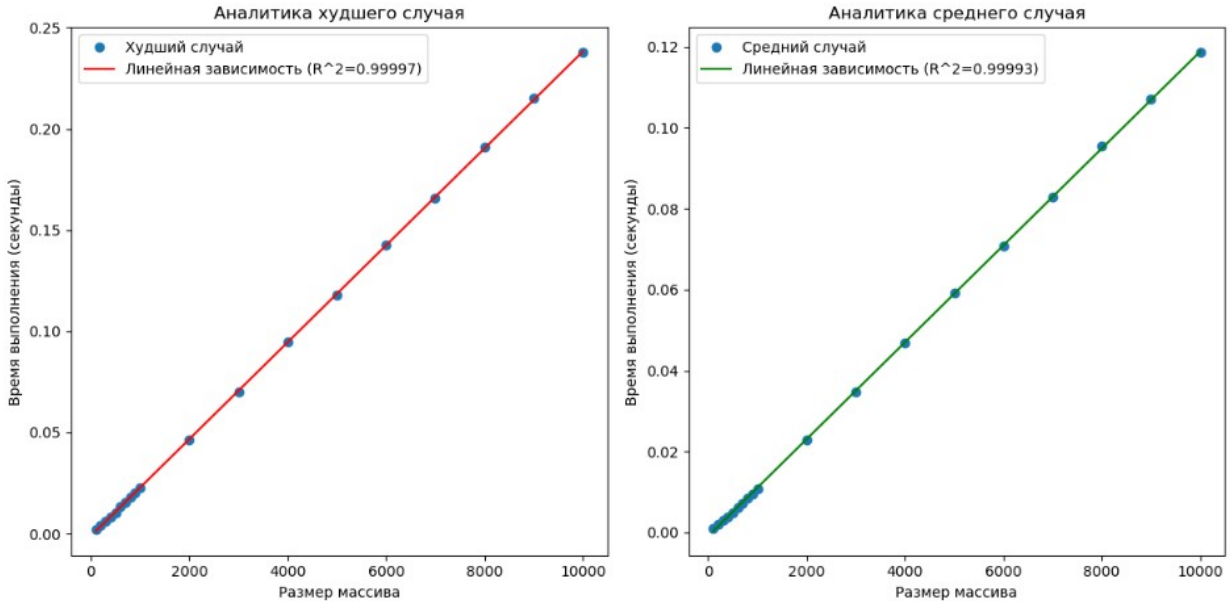

Рисунок 6. Графики линейной зависимости

Вывод: В ходе работы было проведено исследование алгоритма линейного поиска, проведено исследование худшего и среднего случая, построены для них графики и найден коэффициент корреляции