

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 ____

Выполнил:
Репкин Александр Павлович
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., канд. техн. наук,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Порядок выполнения работы:

1. Создана программа на основе поставленного задания – необходимо проанализировать скорость нахождения элемента в списке в зависимости от его положения в нём. Требовалось рассмотреть два варианта – худший (Элемента нет в списке), средний (Элемент находится недалеко от середины списка). На основе полученных данных о затраченном времени необходимо построить график, используя метод наименьших квадратов.

```
public static void main(String[] args) {
    Random rand = new Random();
    while (needed <= 6700000) {
        List<Integer> numbers = new ArrayList<>();
        results.append(needed).append(",");
        for(int i = 0; i < needed; i++) numbers.add(i);
        int max = (int) (needed * 0.6);
        int min = (int) (needed * 0.4);
        // Middle
        int randomNumber = rand.nextInt( bound: (max - min) + 1) + min;
        long startTime = System.nanoTime();
        for (Integer integer : numbers) {
            if (integer == randomNumber) {
                //System.out.println(j);
                break;
            }
        }
        long endTime = System.nanoTime();
        long spent_time = (endTime - startTime); //divide by 1000000 to get milliseconds.
        results.append(spent_time/ 1000000).append(",.").append((spent_time/ 10000)%100).append(",");
        //System.out.println("Spent time is " + spent_time / 10000);
        //System.out.println("\n-1\n");
        // Doesn't exist, -1
        startTime = System.nanoTime();
        for (Integer number : numbers) {
            if (number == needed) {
                System.out.println("Oh no! Why is it here?!");
            }
        }
        endTime = System.nanoTime();
        spent_time = (endTime - startTime); //divide by 1000000 to get milliseconds.
        results.append(spent_time/ 1000000).append(" ").append((spent_time/ 10000)%100).append("\n");
    }
}
```

Рисунок 1. Полученный код.



Рисунок 2. Полученный график для среднего случая.



Рисунок 3. Полученный график для худшего случая.

Код программы:

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Main {

    static int needed = 1000000;

    static StringBuilder results = new StringBuilder();

    public static void main(String[] args) {

        Random rand = new Random();

        while (needed <= 6700000) {

            List<Integer> numbers = new ArrayList<>();

            results.append(needed).append(";");

            for(int i = 0; i < needed; i++) numbers.add(i);

            int max = (int) (needed * 0.6);

            int min = (int) (needed * 0.4);

            // Middle

            int randomNumber = rand.nextInt((max - min) + 1) + min;

            long startTime = System.nanoTime();

            for (Integer integer : numbers) {
```

```

        if (integer == randomNumber) {
            break;
        }
    }
    long endTime = System.nanoTime();
    long spent_time = (endTime - startTime); //divide by 1000000 to get milliseconds.
    results.append(spent_time/          1000000).append(",").append((spent_time/
10000)%100).append(";");
    // Doesn't exist, -1
    startTime = System.nanoTime();
    for (Integer number : numbers) {
        if (number == needed) {
            System.out.println("Oh no! Why is it here?!");
        }
    }
    endTime = System.nanoTime();
    spent_time = (endTime - startTime); //divide by 1000000 to get milliseconds.
    results.append(spent_time/          1000000).append(",").append((spent_time/
10000)%100).append("\n");
    needed += 10000;
}
File outputFile = new File("Results.txt");
try (OutputStream outputStream = new FileOutputStream(outputFile)) {
    outputStream.write(results.toString().getBytes(StandardCharsets.UTF_8));
    outputStream.flush();
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

Вывод: в ходе выполнения практической работы, была рассмотрена зависимость затраты времени на поиск элемента в списке в зависимости от размера списка и положения элемента в нём. Согласно полученным данным, зависимость поиска элемента в массиве — линейная.