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 №4
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.

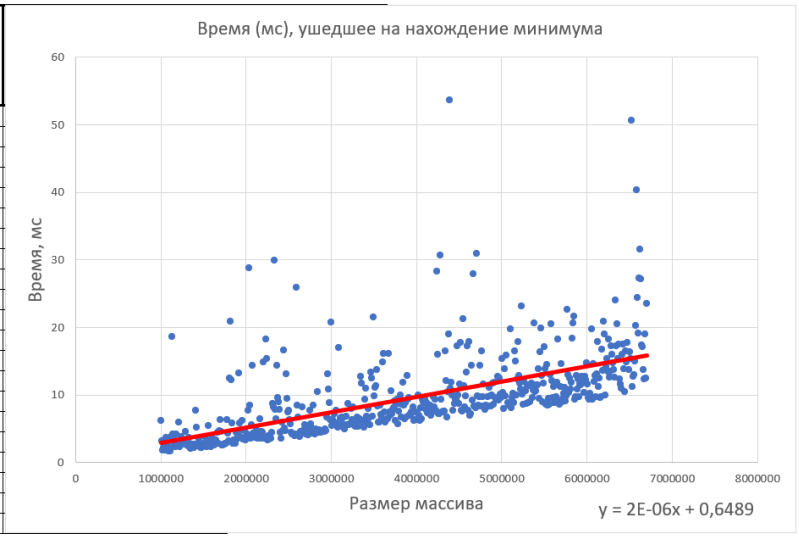
Порядок выполнения работы:

1. Создана программа на основе поставленного задания – необходимо проанализировать скорость нахождения минимального и максимального элементов в списке. На основе полученных данных о затраченном времени необходимо построить график, используя метод наименьших квадратов.

```
public static void main(String[] args) {
    while (needed <= 6700000) {
        int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
        List<Integer> numbers = new ArrayList<>();
        results.append(needed).append(";");
        for (int i = 0; i < needed; i++) numbers.add(i);
        // Maximum
        long startTime = System.nanoTime();
        for (int j = 0; j < needed; j++) {
            if (numbers.get(j) > max) max = numbers.get(j);
        }
        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(";");
        // Minimum
        startTime = System.nanoTime();
        for (int j = 0; j < needed; j++) {
            if (numbers.get(j) < min) min = numbers.get(j);
        }
        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( pathname: "Results.txt");
        try (OutputStream outputStream = new FileOutputStream(outputFile)) {
            outputStream.write(results.toString().getBytes(StandardCharsets.UTF_8));
            outputStream.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Рисунок 1. Полученный код.

Длина массива	Время (мс), ушедшее на нахождение минимума
1000000	6,2
1010000	3,19
1020000	1,77
1030000	2,2
1040000	2,9
1050000	1,84
1060000	3,72
1070000	1,74
1080000	2,4
1090000	1,64
1100000	1,74
1110000	1,69
1120000	3,85
1130000	18,59
1140000	4,24
1150000	2,49
1160000	3,9
1170000	2,31
1180000	4,22
1190000	2,55
1200000	2,89



Коэффициент парной корреляции

0,61300811

Рисунок 2. Полученный график для минимума.

Длина массива	Время (мс), ушедшее на нахождение максимума
1000000	6,6
1010000	2,1
1020000	2,55
1030000	2,22
1040000	2,3
1050000	2,49
1060000	3
1070000	1,86
1080000	2,41
1090000	1,97
1100000	2,84
1110000	2
1120000	7,6
1130000	14,34
1140000	6,25
1150000	3,82
1160000	2,17
1170000	3,3
1180000	4,49
1190000	3,27
1200000	3,96

Коэффициент парной корреляции

0,5851957

Рисунок 3. Полученный график для максимума.

Код программы:

```
package org.example;
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;
public class Main {
static int needed = 1000000;
static StringBuilder results = new StringBuilder();
public static void main(String[] args) {
while (needed <= 6700000) {
int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
List<Integer> numbers = new ArrayList<>();
results.append(needed).append(";");
for (int i = 0; i < needed; i++) numbers.add(i);
// Maximum
long startTime = System.nanoTime();
for (int j = 0; j < needed; j++) {
if (numbers.get(j) > max) max = numbers.get(j);
}
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(";");
        // Minimum
        startTime = System.nanoTime();
        for (int j = 0; j < needed; j++) {
        if (numbers.get(j) < min) min = numbers.get(j);
        }
        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 минимального и максимального элементов в списке в зависимости от размера списка. Согласно полученным данным, зависимость поиска элемента в массиве – линейная.