Министерство цифрового развития связи и массовых коммуникаций РФ

Государственное бюджетное образовательное учреждение высшего образование

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

дисциплина «Структуры и алгоритмы обработки данных»

Отчет по лабораторной работе №1

«Методы сортировки»

Подготовил: студент группы

БВТ1903 Саввин Д.И.

Проверил: Кутейников И.А.

Москва

2021

ОГЛАВЛЕНИЕ

1. ЦЕЛЬ РАБОТЫ 3
2. ВЫПОЛНЕНИЕ 3
3. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 14
4. ВЫВОД 15

1. ЦЕЛЬ РАБОТЫ

Реализовать заданный метод сортировки строк числовой матрицы в соответствии с индивидуальным заданием. Для всех вариантов добавить реализацию быстрой сортировки (quicksort). Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки, используемой в выбранном языке программирования.

Вариант:

5

Турнинрная;

5,11,17,23,29;

2. ВЫПОЛНЕНИЕ

Ниже представлен исходный код программы:

TournamentSort.java

```
package com.company;

import java.util.LinkedList;

public class TournamentSort {

    /** Поля **/

    private LinkedList<Integer> main_array = new LinkedList<>();

    private LinkedList<Integer> wins_array = new LinkedList<>();

    private LinkedList<Integer> wins_array2 = new LinkedList<>();

    private LinkedList<Integer> loses_array = new LinkedList<>();

    private LinkedList<Integer> tree = new LinkedList<>();

    private int loses_count = 0;
```

```
    /** Конструктор **/

    public  TournamentSort(LinkedList<Integer> input_array){
        this.main_array = input_array;
        for (int i = 0; i < 8; i++){
            tree.add(999);
        }
    }


    //ТАК НУ ПРОБЛЕМА БЫЛА В ТОМ ЧТО 1)В МАССИВ ПРОИГРАВШИХ ДОБАВЛЯЕТСЯ ОДИН
И ТОТ ЖЕ ЭЛЕМЕНТ ДВА РАЗА,
    //2) СЛИЯНИЕ НЕ ПРОИСХОДИТ ПРИМЕР: [3, 4, 5, 6, 7, 8, 9, 1, 1, 2]

    /** Методы **/

    //Сравнение первой пары чисел
    public void firtPairSravn(int global_i) {
        if ((tree.get(1) != 999) && (tree.get(2) != 999)) {
            if ((this.tree.get(1)) < (this.tree.get(2))) {
                this.tree.set(5, this.tree.get(1));
                this.tree.set(1,999);
                if (wins_array.size() != 0) {
                    if (global_i < main_array.size()) {
                        while (main_array.get(global_i) <
wins_array.getLast()) {
                            loses_array.add(main_array.get(global_i));
                            loses_count++;
                            global_i++;
                            if (global_i >= main_array.size()){
                                break;
                            }
                        }
                    }
                }
                if ( global_i < main_array.size()){
                    this.tree.set(1, main_array.get(global_i));
                    if (this.tree.get(1) < this.tree.get(5)) {
                        int f = this.tree.get(5);
                        this.tree.set(5, this.tree.get(1));
                        this.tree.set(1, f);
                    }
                    global_i++;
                }
            } else {
                this.tree.set(5, this.tree.get(2));
                this.tree.set(2,999);
                if (wins_array.size() != 0) {
                    if (global_i < main_array.size()) {
                        while (main_array.get(global_i) <
wins_array.getLast()) {
                            loses_array.add(main_array.get(global_i));
                            loses_count++;
                            global_i++;
                            if (global_i >= main_array.size()){
                                break;
                            }
                        }
                    }
                }
                if (global_i < main_array.size()) {
                    this.tree.set(2, main_array.get(global_i));
                    if (this.tree.get(2) < this.tree.get(5)) {
```

```
                    int f = this.tree.get(5);
                    this.tree.set(5, this.tree.get(2));
                    this.tree.set(2, f);
                }
                global_i++;
            }
        }
    }
    else if((tree.get(1) == 999)&&(tree.get(2) != 999)){    //Если первая
ячейка пустая
        this.tree.set(5, this.tree.get(2));
        this.tree.set(2,999);
        if (wins_array.size() != 0) {
            if (global_i < main_array.size()) {
                while (main_array.get(global_i) < wins_array.getLast()) {
                    loses_array.add(main_array.get(global_i));
                    loses_count++;
                    global_i++;
                    if (global_i >= main_array.size()){
                        break;
                    }
                }
            }
        }
        if (global_i < main_array.size()) {
            this.tree.set(2, main_array.get(global_i));
            if (this.tree.get(2) < this.tree.get(5)) {
                int f = this.tree.get(5);
                this.tree.set(5, this.tree.get(2));
                this.tree.set(2, f);
            }
            global_i++;
        }
    }
    else if ((tree.get(2) == 999)&&(tree.get(1) != 999)){   //Если вторая
ячейка пустая
        this.tree.set(5, this.tree.get(1));
        this.tree.set(1,999);
        if (wins_array.size() != 0) {
            if (global_i < main_array.size()) {
                while (main_array.get(global_i) < wins_array.getLast()) {
                    loses_array.add(main_array.get(global_i));
                    loses_count++;
                    global_i++;
                    if (global_i >= main_array.size()){
                        break;
                    }
                }
            }
        }
        if (global_i < main_array.size()) {
            this.tree.set(1, main_array.get(global_i));
            if (this.tree.get(1) < this.tree.get(5)) {
                int f = this.tree.get(5);
                this.tree.set(5, this.tree.get(1));
                this.tree.set(1, f);
            }
            global_i++;
        }
    }
}

//Сравнение второй пары чисел
public void secondPairSravn(int global_i) {
```

```
        if ((tree.get(3) != 999) && (tree.get(4) != 999)) {
            if ((this.tree.get(3)) < (this.tree.get(4))) {
                this.tree.set(6, this.tree.get(3));
                this.tree.set(3,999);
                if (wins_array.size() != 0) {
                    if (global_i < main_array.size()) {
                        while (main_array.get(global_i) <
wins_array.getLast()) {
                            loses_array.add(main_array.get(global_i));
                            loses_count++;
                            global_i++;
                            if (global_i >= main_array.size()){
                                break;
                            }
                        }
                    }
                }
                if (global_i < main_array.size()) {
                    this.tree.set(3, main_array.get(global_i));
                    if (this.tree.get(3) < this.tree.get(6)) {
                        int f = this.tree.get(6);
                        this.tree.set(6, this.tree.get(3));
                        this.tree.set(3, f);
                    }
                    global_i++;
                }
            } else {
                this.tree.set(6, this.tree.get(4));
                this.tree.set(4,999);
                if (wins_array.size() != 0) {
                    if (global_i < main_array.size()) {//-1
                        while (main_array.get(global_i) <
wins_array.getLast()) {
                            loses_array.add(main_array.get(global_i));
                            loses_count++;
                            global_i++;
                            if (global_i >= main_array.size()){
                                break;
                            }
                        }
                    }
                }
                if (global_i < main_array.size()) {
                    this.tree.set(4, main_array.get(global_i));
                    if (this.tree.get(4) < this.tree.get(6)) {
                        int f = this.tree.get(6);
                        this.tree.set(6, this.tree.get(4));
                        this.tree.set(4, f);
                    }
                    global_i++;
                }
            }
        }
        else if ((tree.get(3) == 999)&&(tree.get(4) != 999)){    //Если
третья ячейка пустая
            this.tree.set(6, this.tree.get(4));
            this.tree.set(4,999);
            if (wins_array.size() != 0) {
                if (global_i < main_array.size()) {
                    while (main_array.get(global_i) < wins_array.getLast()) {
                        loses_array.add(main_array.get(global_i));
                        loses_count++;
                        global_i++;
                        if (global_i >= main_array.size()){
```

```
                            break;
                        }
                    }
                }
            }
            if (global_i < main_array.size()) {
                this.tree.set(4, main_array.get(global_i));
                if (this.tree.get(4) < this.tree.get(6)) {
                    int f = this.tree.get(6);
                    this.tree.set(6, this.tree.get(4));
                    this.tree.set(4, f);
                }
                global_i++;
            }
        }
        else if ((tree.get(4) == 999)&&(tree.get(3) != 999)){    //Если
четвертая ячейка пустая
            this.tree.set(6, this.tree.get(3));
            this.tree.set(3,999);
            if (wins_array.size() != 0) {
                if (global_i < main_array.size()) {
                    while (main_array.get(global_i) < wins_array.getLast()) {
                        loses_array.add(main_array.get(global_i));
                        loses_count++;
                        global_i++;
                        if (global_i >= main_array.size()){
                            break;
                        }
                    }
                }
            }
            if (global_i < main_array.size()) {
                this.tree.set(3, main_array.get(global_i));
                if (this.tree.get(3) < this.tree.get(6)) {
                    int f = this.tree.get(6);
                    this.tree.set(6, this.tree.get(3));
                    this.tree.set(3, f);
                }
                global_i++;
            }
        }
    }

    //Сравнение финала
    public void finalPairSravn(int global_i, int wins_count) {
        if ((tree.get(5) != 999) && (tree.get(6) != 999)) {
            if ((this.tree.get(5)) < (this.tree.get(6))) {
                this.tree.set(7, this.tree.get(5));
                this.tree.set(5,999);
                    firtPairSravn(global_i);
                    if (this.tree.get(5) < this.tree.get(7)) {
                        int f = this.tree.get(7);
                        this.tree.set(7, this.tree.get(5));
                        this.tree.set(5, f);
                    }
                global_i++;
                for (int i = 0; i < loses_count; i++)
                    global_i++;
                this.wins_array.add(this.tree.get(7));
                tree.set(7,999);
                wins_count++;
            } else {
                this.tree.set(7, this.tree.get(6));
                this.tree.set(6,999);
```

```
                    secondPairSravn(global_i);//global_i++
                    if (this.tree.get(6) < this.tree.get(7)) {
                        int f = this.tree.get(7);
                        this.tree.set(7, this.tree.get(6));
                        this.tree.set(6, f);
                    }
                global_i++;
                    for (int i = 0; i < loses_count; i++)
                    global_i++;
                this.wins_array.add(this.tree.get(7));
                tree.set(7,999);
                wins_count++;
            }
        }
        else if ((tree.get(5) == 999)&&(tree.get(6) != 999)){   //Если пятая
ячейка пустая
            this.tree.set(7, this.tree.get(6));
            this.tree.set(6,999);
                secondPairSravn(global_i);
                if (this.tree.get(6) < this.tree.get(7)) {
                    int f = this.tree.get(7);
                    this.tree.set(7, this.tree.get(6));
                    this.tree.set(6, f);
                }
            global_i++;
            for (int i = 0; i < loses_count; i++)
                global_i++;
            this.wins_array.add(this.tree.get(7));
            tree.set(7,999);
            wins_count++;
        }
        else if ((tree.get(6) == 999)&&(tree.get(5) != 999)){   //Если шестая
ячейка пустая
            this.tree.set(7, this.tree.get(5));
            this.tree.set(5,999);
                firtPairSravn(global_i);
                if (this.tree.get(5) < this.tree.get(7)) {
                    int f = this.tree.get(7);
                    this.tree.set(7, this.tree.get(5));
                    this.tree.set(5, f);
                }
            global_i++;
            for (int i = 0; i < loses_count; i++)
                global_i++;
            this.wins_array.add(this.tree.get(7));
            tree.set(7,999);
            wins_count++;
        }
    }

    //Сравнение финала 2
    public void finalPairSravn2(int global_i, int wins_count) {
        if ((tree.get(5) != 999) && (tree.get(6) != 999)) {
            if ((this.tree.get(5)) < (this.tree.get(6))) {
                this.tree.set(7, this.tree.get(5));
                this.tree.set(5, 999);
                if ((tree.get(1) != 999) && (tree.get(2) != 999)) {
                    firtPairSravn(global_i);
                    if (this.tree.get(5) < this.tree.get(7)) {
                        int f = this.tree.get(7);
                        this.tree.set(7, this.tree.get(5));
                        this.tree.set(5, f);
                    }
                    global_i++;
```

```
                    for (int i = 0; i < loses_count; i++)
                        global_i++;
                }
                this.wins_array.add(this.tree.get(7));
                tree.set(7,999);
                wins_count++;
            } else {
                this.tree.set(7, this.tree.get(6));
                this.tree.set(6, 999);
                if ((tree.get(3) != 999) && (tree.get(4) != 999)) {
                    secondPairSravn(global_i);
                    if (this.tree.get(6) < this.tree.get(7)) {
                        int f = this.tree.get(7);
                        this.tree.set(7, this.tree.get(6));
                        this.tree.set(6, f);
                    }
                    global_i++;
                    for (int i = 0; i < loses_count; i++)
                        global_i++;
                }
                this.wins_array.add(this.tree.get(7));
                tree.set(7,999);
                wins_count++;
            }
        }
        else if ((tree.get(5) == 999)&&(tree.get(6) != 999)){   //Если пятая
ячейка пустая
            this.tree.set(7, this.tree.get(6));
            this.tree.set(6,999);
            if ((tree.get(3) != 999) && (tree.get(4) != 999)) {
                secondPairSravn(global_i);
                if (this.tree.get(6) < this.tree.get(7)) {
                    int f = this.tree.get(7);
                    this.tree.set(7, this.tree.get(6));
                    this.tree.set(6, f);
                }
                global_i++;
                for (int i = 0; i < loses_count; i++)
                    global_i++;
            }
            this.wins_array.add(this.tree.get(7));
            tree.set(7,999);
            wins_count++;
        }
        else if ((tree.get(6) == 999)&&(tree.get(5) != 999)){   //Если шестая
ячейка пустая
            this.tree.set(7, this.tree.get(5));
            this.tree.set(5,999);
            if ((tree.get(1) != 999) && (tree.get(2) != 999)) {
                firtPairSravn(global_i);
                if (this.tree.get(5) < this.tree.get(7)) {
                    int f = this.tree.get(7);
                    this.tree.set(7, this.tree.get(5));
                    this.tree.set(5, f);
                }
                global i++;
                for (int i = 0; i < loses_count; i++)
                    global_i++;
            }
            this.wins_array.add(this.tree.get(7));
            tree.set(7,999);
            wins_count++;
        }
    }
```

```
    public LinkedList<Integer> Sliyanie (LinkedList<Integer> arr1,
LinkedList<Integer> arr2) {
            LinkedList<Integer> m_arr = new LinkedList<Integer>();
            int a = arr1.size() + arr2.size();

            //Слияние массивов победителей
            for (int i = 0; i < a; i++) {

                if ((arr2.size()!=0)&&(arr1.size()!=0)&&(arr1.get(0) <
arr2.get(0))) {
                    m_arr.add(arr1.get(0));
                    arr1.pop(); //Сдвиг элементов влево
                } else {
                    m_arr.add(arr2.get(0));
                    arr2.pop(); //Сдвиг элементов влево
                }
            }
        return m_arr;
    }

    public void doSort(){
        int global_i = 0;
        int wins_count = 0;

        //Заполнение первых четырех мест
        for (int i = 0; i < 4; i++) {
            if (i < main_array.size()) {
                this.tree.set(i + 1, this.main_array.get(i));
                global_i = i + 1;
            }
        }

        //Первый прогон основного массива
        while
((tree.get(1)!=999)||(tree.get(2)!=999)||(tree.get(3)!=999)||(tree.get(4)!=99
9)||(tree.get(5)!=999)
                ||(tree.get(6)!=999)||(tree.get(7)!=999)){

            if (tree.get(5) == 999) {
                //Сравнение первой пары чисел
                firtPairSravn(global_i);
                global_i++;
            }

            if (tree.get(6) == 999) {
                //Сравнение второй пары чисел
                secondPairSravn(global_i);
                global_i++;
            }

            //Сравнение финала
            finalPairSravn(global_i,wins_count);
            global_i++;
            for (int i = 0; i < loses_count; i++) {
            global i++;
            }
            wins_count++;

        }

        //Замена основного ссмаива, массивом проигравших
        main_array = loses_array;
        for (int i = 0; i < wins_array.size(); i++){
```

```
            wins_array2.add(wins_array.get(i));
        }
        global_i = 0;
        wins_count = 0;
        this.wins_array.clear();

        //Заполнение первых четырех мест
        for (int i = 0; i < 4; i++) {
            if (i < main_array.size()) {
                this.tree.set(i + 1, this.main_array.get(i));
                global_i = i + 1;
            }
        }


        //Второй прогон основного массива(с проигравшими)
        while
((tree.get(1)!=999)||(tree.get(2)!=999)||(tree.get(3)!=999)||(tree.get(4)!=99
9)||(tree.get(5)!=999)
                ||(tree.get(6)!=999)||(tree.get(7)!=999)){
            if (tree.get(5) == 999) {
                //Сравнение первой пары чисел
                firtPairSravn(global_i);
                global_i++;
            }

            if (tree.get(6) == 999) {
                //Сравнение второй пары чисел
                secondPairSravn(global_i);
                global_i++;
            }

            //Сравнение финала
            finalPairSravn(global_i,wins_count);
            global_i++;
            wins_count++;
        }

       //Слияние массивов победителей
        if ((wins_array.size() >= 0) && (wins_array2.size() > 0)) {
            main_array = Sliyanie(wins_array, wins_array2);
        }
        else{
            main_array = wins_array;
        }
    }

    public void toStringMain(){
        System.out.println(main_array);
    }
}
```

QuickSort.java

```
package com.company;

import java.util.LinkedList;

public class QuickSort {

    /** Поля **/

    private LinkedList<Integer> array = new LinkedList<>(); // Основной
массив


    /** Конструктор **/

    public QuickSort(LinkedList<Integer> arr){
        array = arr;
    }


    /** Методы **/

    // Метод выбора опорного элемента
    public static int chooseB(int L, int R){
        int B = (L+R)/2;
        return B;
    }

    // Метод прогона по массиву
    public LinkedList<Integer> startIteration(int L, int R){
        int B = chooseB(L,R);
        int l = L;
        int r = R;

        while (l < r){
            while (l < B) {
                if (array.get(l) <= array.get(B)) { l++; }
                else { break; }
            }
            while (r > B){
                if (array.get(r) >= array.get(B)){ r--; }
                else{ break; }
            }
            // Замена элементов
            int f = array.get(l);
            array.set(l,array.get(r));
            array.set(r,f);
        }
        return array;
    }


    // Метод полной сортировки массива
    public LinkedList<Integer> doQuickSort(LinkedList<Integer> arr) {
        int L = 0;
        int R = arr.size() - 1;
        QuickSort quickSort = new QuickSort(arr);
        arr = quickSort.startIteration(L, R);
```

```
        while ((R - L) >= 1) {
            for (int i = 0; i < arr.size(); i++) {
                for (int j = arr.size()-1; j > 0; j--) {
                    L = i;
                    R = j;
                    arr = quickSort.startIteration(L, R);
                }
            }
        }
        return arr;
    }

    public void toStringQuick(){
        System.out.println(array);
    }
}
```

Main.java

```
package com.company;

import java.lang.reflect.Array;
import java.util.Arrays;
import java.util.Comparator;
import java.util.LinkedList;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        int method;
        Scanner in = new Scanner(System.in);
        LinkedList<Integer> linkedList = new LinkedList<Integer>();
        String str = in.next();
        while (!str.contentEquals("stop")) {
            linkedList.add(Integer.parseInt(str));
            str = in.next();
        }
        System.out.println("Выберите метод сортировки:\n1.Турнирная
сортировка\n2.Быстрая соритровка\n3.Стандартная сортировка");
        str = in.next();
        method = Integer.parseInt(str);
        if (method == 1) {
            TournamentSort tournamentSort = new TournamentSort(linkedList);
            long startTime = System.nanoTime();
            tournamentSort.doSort();
            long time = System.nanoTime() - startTime;
            tournamentSort.toStringMain();
            System.out.println("На сортировку потрачено - " + time + "
наносекунд");
        }
        else if (method == 2){
            QuickSort quickSort = new QuickSort(linkedList);
            long startTime = System.nanoTime();
            quickSort.doQuickSort(linkedList);
            long time = System.nanoTime() - startTime;
            quickSort.toStringQuick();
            System.out.println("На сортировку потрачено - " + time + "
```

```
наносекунд");
        }
        else{
            int[] arr = new int[linkedList.size()];
            for (int i = 0;i < linkedList.size(); i++){
               arr[i] = linkedList.get(i);
            }
            long startTime = System.nanoTime();
            Arrays.sort(arr);
            long time = System.nanoTime() - startTime;
            System.out.print("[");
            for (int i = 0; i < arr.length; i++) {
                if ((i+1)!=arr.length)
                    System.out.print(arr[i]+", ");
                else
                    System.out.print(arr[i]);
            }
            System.out.println("]");
            System.out.println("На сортировку потрачено - " + time + "
наносекунд");
        }
    }
}
```

3. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Ниже представлены рисунки сравнения скорости выполнения различных сортирово.

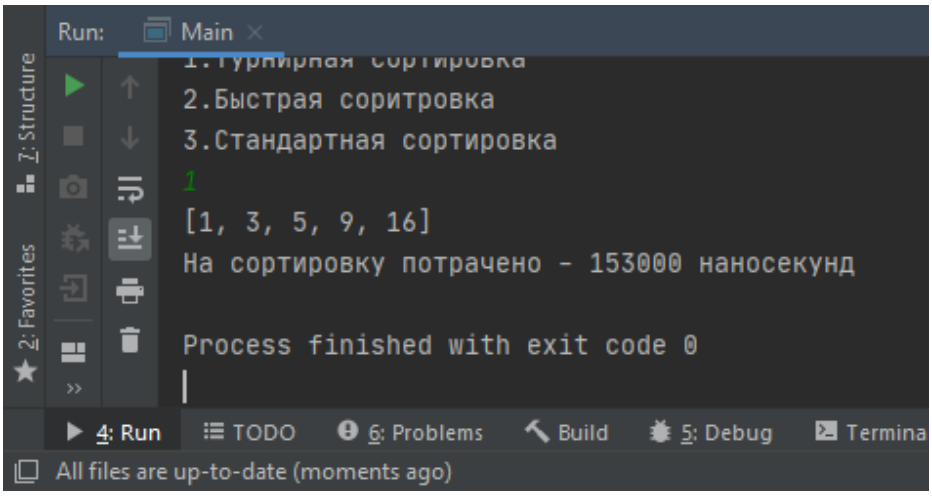

Рис. 1 – Результат и скорость выполнения при помощи турнирной сортировки.

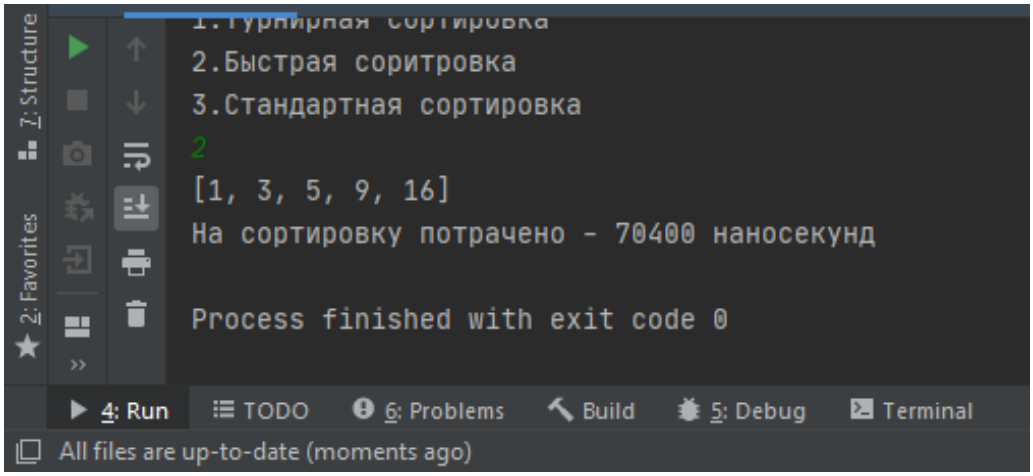

Рис. 2 – Результат и скорость выполнения при помощи быстрой сортировки.

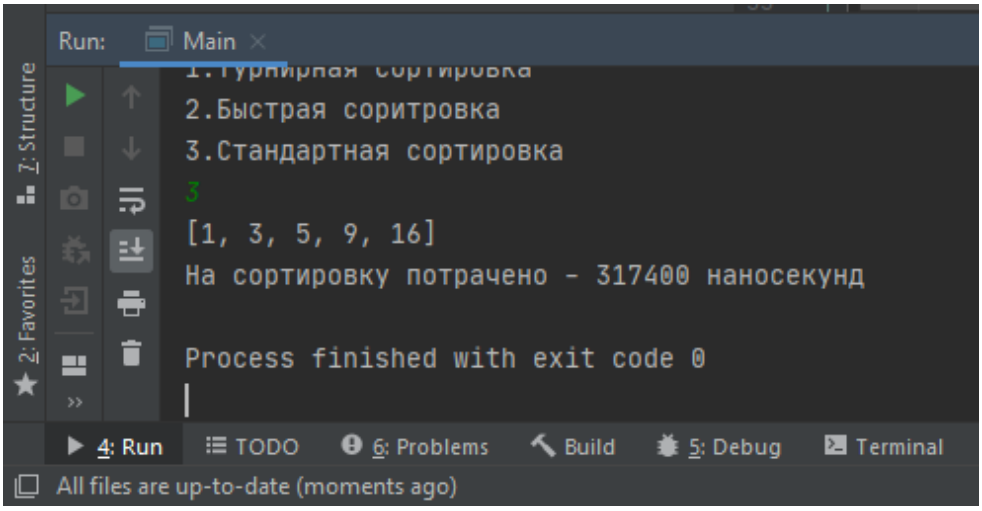

Рис. 3 – Результат и скорость выполнения при помощи стандартной сортировки.

4. ВЫВОД

В ходе выполнения данной лабораторной работы мы познакомились с различными видами сортировки, а также сравнили скорости их быстродействия. Самой быстрой оказалась «быстрая сортировка», затем «турнирная сортировка», и самой медленной оказалась стандартная сортировка метод Arrays.sort().