

**Федеральное агентство связи**  
**Ордена Трудового Красного Знамени**  
**Федеральное государственное бюджетное образовательное учреждение**  
**высшего образования**  
**«Московский технический университет связи и информатики»**

Кафедра «Информатики»

**Отчет по лабораторной работе №1**  
СиАОД

Выполнил: студент группы БВТ1901

Адедиха Коффи Жермен

Руководитель:

Мелехин А.

Москва 2021

## Задание

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. Испытания проводить на сгенерированных матрицах.

## Решение:

Реализуем заданные методы сортировки, используя язык программирования Java. Встроим в программу рандомайзер матрицы размером 50\*50.

Код программы сортировки выбором

```
import java.util.Random;
import java.util.Arrays;

public class Selection {
    public static void main(String[] args) {

        int[][] array2d = new int[50][50];
        Random rand = new Random();

        for (int i = 0; i < 50; i++)
            for (int j = 0; j < 50; j++)
                array2d[i][j] = rand.nextInt(1012);

        for(int[] m : toMatrix(sort(toArray(array2d)))) {
            for(int i : m){
                System.out.print(i + "\t");
            }
            System.out.println();
        }

        System.out.println();

        long time = System.currentTimeMillis();
        sort(toArray(array2d));
        System.out.println(System.currentTimeMillis() - time);
    }

    public static int[] toArray(int[][] arr){
        int[] flat = new int[50 * 50];
```

```

        int ctr = 0;
        for (int row = 0; row < 50; row++) {
            for (int col = 0; col < 50; col++) {
                flat[ctr++] = arr[row][col];
            }
        }
        return flat;
    }

    public static int[][] toMatrix(int[] arr){
        int [][] numbers = new int [50][50];

        int m = 0;
        for(int i = 0 ; i < 50 ; i++)
            for(int j = 0 ; j < 50; j++)
                numbers[i][j] = arr[m++];

        return numbers;
    }

    public static int[] sort(int[] array) {
        for (int i = 0; i < array.length; i++) {        // i - номер текущего шага
            int pos = i;
            int min = array[i];
            // цикл выбора наименьшего элемента
            for (int j = i + 1; j < array.length; j++) {
                if (array[j] < min) {
                    pos = j;        // pos - индекс наименьшего элемента
                    min = array[j];
                }
            }
            array[pos] = array[i];
            array[i] = min;        // меняем местами наименьший с array[i]
        }

        return array;
    }
}

```

## Код программы сортировки вставкой

```

import java.util.Random;

public class Insertion {
    public static void main(String[] args) {

        int[][] array2d = new int[50][50];
        Random rand = new Random();

        for (int i = 0; i < 50; i++)

```

```

        for (int j = 0; j < 50; j++)
            array2d[i][j] = rand.nextInt(1012);

    for(int[] m : toMatrix(sort(toArray(array2d)))) {
        for(int i : m){
            System.out.print(i + "\t");
        }
        System.out.println();
    }

    System.out.println();

    long time = System.currentTimeMillis();
    sort(toArray(array2d));
    System.out.println(System.currentTimeMillis() - time);
}

public static int[] toArray(int[][] arr){
    int[] flat = new int[50 * 50];

    int ctr = 0;
    for (int row = 0; row < 50; row++) {
        for (int col = 0; col < 50; col++) {
            flat[ctr++] = arr[row][col];
        }
    }
    return flat;
}

public static int[][] toMatrix(int[] arr){
    int [][] numbers = new int [50][50];

    int m = 0;
    for(int i = 0 ; i < 50 ; i++)
        for(int j = 0 ; j < 50; j++)
            numbers[i][j] = arr[m++];

    return numbers;
}

public static int[] sort(int[] array) {
    int key;
    for (int i = 1; i < array.length; i++) {
        key = array[i];
        int j = i - 1;
        while (j >= 0 && array[j] > key) {
            array[j + 1] = array[j];
            j = j - 1;
        }
        array[j + 1] = key;
    }
}

```

```
        return array;
    }
}
```

## Код программы сортировки обменом

```
import java.util.Random;

public class Bubble {
    public static void main(String[] args) {

        int[][] array2d = new int[50][50];
        Random rand = new Random();

        for (int i = 0; i < 50; i++)
            for (int j = 0; j < 50; j++)
                array2d[i][j] = rand.nextInt(1012);

        for(int[] m : toMatrix(sort(toArray(array2d)))) {
            for(int i : m){
                System.out.print(i + "\t");
            }
            System.out.println();
        }

        System.out.println();

        long time = System.currentTimeMillis();
        sort(toArray(array2d));
        System.out.println(System.currentTimeMillis() - time);
    }

    public static int[] toArray(int[][] arr){
        int[] flat = new int[50 * 50];

        int ctr = 0;
        for (int row = 0; row < 50; row++) {
            for (int col = 0; col < 50; col++) {
                flat[ctr++] = arr[row][col];
            }
        }
        return flat;
    }

    public static int[][] toMatrix(int[] arr){
        int [][] numbers = new int [50][50];

        int m = 0;
```

```

        for(int i = 0 ; i < 50 ; i++)
            for(int j = 0 ; j < 50; j++)
                numbers[i][j] = arr[m++];

        return numbers;
    }

    public static int[] sort(int[] array) {
        // i - номер прохода
        for (int i = 0; i < array.length - 1; i++) {
            // внутренний цикл прохода
            for (int j = array.length - 1; j > i; j--) {
                if (array[j - 1] > array[j]) {
                    int tmp = array[j - 1];
                    array[j - 1] = array[j];
                    array[j] = tmp;
                }
            }
        }

        return array;
    }
}

```

## Код программы сортировки Шелла

```

import java.util.Random;

public class ShellSort {

    public static void main(String[] args) {

        int[][] array2d = new int[50][50];
        Random rand = new Random();

        for (int i = 0; i < 50; i++)
            for (int j = 0; j < 50; j++)
                array2d[i][j] = rand.nextInt(1012);
        for(int[] m : toMatrix(sort(toArray(array2d)))) {
            for(int i : m){
                System.out.print(i + "\t");
            }
            System.out.println();
        }

        System.out.println();

        long time = System.currentTimeMillis();
    }
}

```

```

        sort(toArray(array2d));
        System.out.println(System.currentTimeMillis() - time);
    }

    public static int[] toArray(int[][] arr){
        int[] flat = new int[50 * 50];

        int ctr = 0;
        for (int row = 0; row < 50; row++) {
            for (int col = 0; col < 50; col++) {
                flat[ctr++] = arr[row][col];
            }
        }
        return flat;
    }

    public static int[][] toMatrix(int[] arr){
        int [][] numbers = new int [50][50];

        int m = 0;
        for(int i = 0 ; i < 50 ; i++)
            for(int j = 0 ; j < 50; j++)
                numbers[i][j] = arr[m++];

        return numbers;
    }

    public static int[] sort(int[] array) {
        int temp;
        int h = 0; //величина интервала

        //вычисляем исходное значение интервала
        while(h <= array.length/3)
            h = 3*h + 1;

        for(int k = h; k > 0; k = (k-1)/3)
            for(int i = k; i < array.length; i++)
            {
                temp = array[i];
                int j;
                for(j = i; j >= k; j -= k)
                {
                    if(temp < array[j - k])
                        array[j] = array[j - k];
                    else
                        break;
                }
                array[j] = temp;
            }

        return array;
    }

```

```

    }

}

```

## Код программы сортировки турнирной

```

import java.util.Random;
public class TournamentSort{
    public static void main(String[] args) {

        int[][] array2d = new int[50][50];
        Random rand = new Random();

        for (int i = 0; i < 50; i++)
            for (int j = 0; j < 50; j++)
                array2d[i][j] = rand.nextInt(1012);

        for(int[] m : toMatrix(sort(toArray(array2d)))) {
            for(int i : m){
                System.out.print(i + "\t");
            }
            System.out.println();
        }

        System.out.println();

        long time = System.currentTimeMillis();
        sort(toArray(array2d));
        System.out.println(System.currentTimeMillis() - time);
    }

    public static int[] toArray(int[][] arr){
        int[] flat = new int[50 * 50];

        int ctr = 0;
        for (int row = 0; row < 50; row++) {
            for (int col = 0; col < 50; col++) {
                flat[ctr++] = arr[row][col];
            }
        }
        return flat;
    }

    public static int[][] toMatrix(int[] arr){
        int [][] numbers = new int [50][50];

```



```

        int m = 0;
        for(int i = 0 ; i < 50 ; i++)
            for(int j = 0 ; j < 50; j++)
                numbers[i][j] = arr[m++];

        return numbers;
    }

    public static int[] sort(int[] array) {
        int[] arr = new int[array.length];
        for (int i = 0; i < array.length; i++) {
            int c;
            for (int a = 1; a < array.length; a = a * 2) {
                for (int k = 0; k * a * 2 < array.length; k++) {
                    if (k * a * 2 + a < array.length) {
                        if (array[k * a * 2 + a] < array[k * a * 2]) {
                            c = array[k * 2 * a + a];
                            array[k * 2 * a + a] = array[k * 2 * a];
                            array[k * 2 * a] = c;
                        }
                    }
                }
            }
            arr[i] = array[0];
            array[0] = Integer.MAX_VALUE;
        }
        return arr;
    }
}

```

## Код программы сортировки пирамидальной

```

import java.util.Random;
public class PyramideSort{
    public static void main(String[] args) {
        int[][] array2d = new int[50][50];
        Random rand = new Random();
        for (int i = 0; i < 50; i++)
            for (int j = 0; j < 50; j++)
                array2d[i][j] = rand.nextInt(1012);
        for(int[] m : toMatrix(sort(toArray(array2d)))) {
            for(int i : m){
                System.out.print(i + "\t");
            }
            System.out.println();
        }

        System.out.println();
    }
}

```

```

    long time = System.currentTimeMillis();
    sort(toArray(array2d));
    System.out.println(System.currentTimeMillis() - time);
}

```

```

public static int[] toArray(int[][] arr){
    int[] flat = new int[50 * 50];

    int ctr = 0;
    for (int row = 0; row < 50; row++) {
        for (int col = 0; col < 50; col++) {
            flat[ctr++] = arr[row][col];
        }
    }
    return flat;
}

```

```

public static int[][] toMatrix(int[] arr){
    int [][] numbers = new int [50][50];

    int m = 0;
    for(int i = 0 ; i < 50 ; i++)
        for(int j = 0 ; j < 50; j++)
            numbers[i][j] = arr[m++];

    return numbers;
}

```

```

// Процедура для преобразования в двоичную кучу поддерева с корневым
// узлом i, что является индексом в arr[]. n - размер кучи
public static void heapify(int arr[], int n, int i)
{

```

```

    int largest = i; // Инициализируем наибольший элемент как корень
    int l = 2*i + 1; // левый потомок = 2*i + 1
    int r = 2*i + 2; // правый потомок = 2*i + 2

```

```

    // Если левый дочерний элемент больше корня
    if (l < n && arr[l] > arr[largest])
        largest = l;

```

```

    // Если правый дочерний элемент больше, чем самый большой элемент на
    // данный момент

```

```

    if (r < n && arr[r] > arr[largest])
        largest = r;

```

```

    // Если самый большой элемент не корень
    if (largest != i)
    {

```

```

        int swap = arr[i];
        arr[i] = arr[largest];
        arr[largest] = swap;

```

```

    // Рекурсивно преобразуем в двоичную кучу затронутое поддерево

```

```

        heapify(arr, n, largest);
    }
}

public static int[] sort(int arr[])
{
    int n = arr.length;

    // Построение кучи (перегруппируем массив)
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // Один за другим извлекаем элементы из кучи
    for (int i=n-1; i>=0; i--)
    {
        // Перемещаем текущий корень в конец
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;

        // Вызываем процедуру heapify на уменьшенной куче
        heapify(arr, i, 0);
    }
    return arr;
}
}

```

## Работа программы

Работа сортировки Выбором.

ers\sony\AppData\Local\Temp\vscode\sws_0b6c0\jdt_ws\jdt.ls-java-project\bin\'																		Insertion'																	
0	1	1	1	1	2	3	3	3	3	5	5	6	7	8	8	9	9	9																	
9	10	10	10	11	11	11	12	12	12	13	13	15	16	16	17	18	18	18																	
9	19	20	20	21	21	22	22	22	22																										
23	24	24	24	24	24	25	25	25	25	26	28	28	28	28	28	29	29	30																	
0	30	30	30	31	31	32	33	33	33	33	34	34	35	35	35	36	36	36																	
7	38	38	38	39	40	40	40	42	42																										
42	42	42	42	43	43	44	44	44	44	45	45	46	46	46	47	47	47	47																	
8	48	48	48	48	49	49	49	50	51	51	51	51	52	52	52	52	53	53																	
3	53	53	53	53	53	53	53	53	54																										
54	54	55	55	55	56	56	56	58	58	58	59	59	60	60	60	60	61	61																	
1	61	61	61	62	62	62	63	65	66	66	67	67	67	67	67	68	68	68																	
8	68	68	68	68	69	69	70	70	71																										
71	71	72	74	75	75	75	75	75	75	77	77	78	78	78	79	79	80	80																	
0	81	81	81	81	82	82	83	83	83	86	86	86	87	87	88	88	89	89																	
9	89	91	92	92	92	93	93	93	94																										
94	94	94	96	97	98	99	99	99	100	100	100	100	100	101	101	101	101	102																	
02	103	103	103	103	104	105	105	105	105	106	106	106	106	107	107	107	108	108																	
08	108	109	109	109	110	110	110	111	112																										
112	113	113	113	113	114	114	115	115	116	116	116	116	117	117	119	119	119	120																	
20	120	121	121	122	122	122	123	124	124	124	125	125	126	126	126	127	128	128																	
29	129	130	130	130	130	131	131	133	134																										
134	134	134	134	134	135	135	136	136	137	137	138	138	138	138	139	139	139	139																	
39	139	141	142	142	142	143	143	143	144	145	145	145	145	146	146	147	147	147																	
47	147	147	148	148	148	148	148	149	150																										
150	151	151	152	152	153	153	154	154	154	154	154	154	156	158	159	160	161	161																	
62	162	162	162	163	164	165	165	165	166	166	166	167	167	167	168	169	169	170																	
71	171	171	172	173	173	173	174	174	174																										
174	175	175	175	175	176	176	177	177	178	178	178	179	180	181	181	182	182	183																	
83	183	183	184	184	184	185	185	185	186	186	187	188	188	188	188	189	190	190																	
91	191	191	191	192	192	193	193	194	195																										
195	195	196	196	197	198	198	199	199	199	199	200	200	200	201	201	201	201	201																	

## Работа вставкой сортировки.

	1	1	1	2	2	3	3	4	5	6	6	6	7	7	7	7	8	8
9	10	10	11	11	11	12	12	13	14	15	16	18	18	18	19	19	19	20
0	20	21	21	21	23	23	24	24										
24	24	25	25	25	26	26	26	26	27	27	27	28	28	28	29	30	31	31
1	31	31	31	31	32	32	32	33	33	33	33	34	34	34	35	35	35	35
5	36	37	37	37	37	37	38	38	38									
39	39	39	39	41	41	41	41	41	42	42	42	42	43	43	43	43	44	44
4	44	45	46	46	46	46	46	46	47	47	47	47	47	48	48	48	50	50
1	51	52	52	52	53	53	55	56	57									
58	59	59	59	59	60	60	60	61	62	62	63	63	63	64	66	66	67	67
7	67	69	69	70	70	70	70	71	71	72	72	72	73	73	74	74	76	76
6	76	77	77	77	77	78	78	78	78									
79	79	80	80	80	80	81	82	82	83	83	83	84	85	85	86	86	86	86
6	86	87	87	87	87	87	88	88	89	89	89	90	91	91	91	91	92	93
3	93	94	94	94	94	95	95	95	95									
96	96	97	97	98	98	99	100	100	100	101	101	101	103	103	103	104	104	104
06	107	107	110	111	112	112	113	114	114	115	115	116	116	116	117	117	117	118
18	119	119	119	119	119	119	120	120	120									
120	120	120	121	122	122	124	124	124	125	125	126	126	126	126	128	129	129	130
31	132	132	133	133	134	134	135	137	138	138	139	139	139	140	140	141	141	142
42	142	144	144	144	145	145	145	146	146									
147	148	149	149	149	149	149	150	150	150	150	151	151	153	154	154	154	154	155
55	157	157	157	158	158	158	158	159	159	160	160	161	161	161	161	163	163	163
64	164	164	165	165	165	165	166	166	167									
167	167	167	168	168	169	169	170	171	171	172	172	172	172	172	172	173	173	173
73	173	174	174	174	175	175	176	176	176	177	178	178	178	178	179	180	181	182
84	184	185	185	185	186	186	187	187	187									
188	189	190	190	191	191	192	192	192	193	193	193	193	193	195	196	196	197	197
07	199	199	200	200	200	201	202	203	203	203	203	204	204	205	205	206	206	207
98	208	208	208	209	209	209	209	210	210									
211	211	212	212	212	212	212	213	213	213	214	214	216	216	216	217	217	218	218
18	218	219	220	221	221	221	222	223	223	223	223	223	223	224	224	225	226	226
26	227	227	227	228	228	229	230	230	230									

## Работа сортировки обменом.

ers\sony\AppData\Local\Temp\vscode\sws_60f02\jdt_ws\jdt.ls-java-project\bin\ShellSort																															
0	0	1	1	1	1	2	2	2	3	3	4	5	5	5	5	6	6	6	7	7	7	7	7	7	7	7	7	7	7	7	7
7	7	9	9	9	9	9	9	10	10	11	12	13	13	14	14	15	15	16	16	16	16	16	16	16	16	16	16	16	16	16	16
6	16	17	17	18	19	19	19	19	19	19	23	24	24	24	25	27	27	27	29	29	29	29	29	29	29	29	29	29	29	29	29
20	20	21	22	22	22	22	22	22	23	23	24	24	24	25	27	27	27	29	29	29	29	29	29	29	29	29	29	29	29	29	29
9	29	30	30	30	30	31	31	31	31	31	31	31	32	33	33	34	34	34	34	35	35	35	35	35	35	35	35	35	35	35	35
5	36	36	36	39	39	39	39	39	39	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
41	41	41	41	42	42	43	44	44	44	44	45	45	46	46	46	48	48	48	49	49	49	49	49	49	49	49	49	49	49	49	49
0	50	50	50	50	51	51	51	52	52	53	53	53	54	54	54	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55
5	56	56	56	56	57	58	59	59	59	59	64	64	64	66	66	66	66	67	67	67	67	67	67	67	67	67	67	67	67	67	67
59	60	60	61	61	62	63	63	63	64	64	64	64	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66
9	69	69	70	71	71	72	72	73	73	74	74	74	75	76	76	76	76	77	77	77	77	77	77	77	77	77	77	77	77	77	77
8	78	79	79	79	81	81	81	81	82	82	82	82	82	82	82	82	82	82	82	82	82	82	82	82	82	82	82	82	82	82	82
83	83	83	83	84	84	84	84	85	85	86	87	87	87	88	89	89	89	89	89	89	89	89	89	89	89	89	89	89	89	89	89
0	90	90	90	91	91	92	93	93	94	94	94	94	94	95	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96
7	99	99	100	101	101	101	101	101	102	102	106	106	106	107	107	107	108	108	108	108	108	108	108	108	108	108	108	108	108	108	108
182	102	102	102	102	103	105	105	105	105	105	114	114	114	114	114	114	114	115	115	115	115	115	115	115	115	115	115	115	115	115	115
69	110	110	111	112	112	112	112	112	113	113	114	114	114	114	114	114	114	114	114	114	114	114	114	114	114	114	114	114	114	114	114
17	117	118	119	119	119	120	120	120	120	120	126	126	127	127	127	127	127	127	128	128	128	128	128	128	128	128	128	128	128	128	128
121	121	121	122	122	122	122	123	125	125	126	126	126	127	127	127	127	127	127	127	127	127	127	127	127	127	127	127	127	127	127	127
29	132	132	132	132	132	133	134	134	134	134	134	135	135	135	135	135	136	137	137	137	137	137	137	137	137	137	137	137	137	137	137
39	139	140	140	140	140	141	141	142	142	142	142	142	142	142	142	142	142	142	142	142	142	142	142	142	142	142	142	142	142	142	142
142	142	144	144	144	144	144	144	144	145	145	146	146	147	148	148	149	149	149	149	149	149	149	149	149	149	149	149	149	149	149	149
50	151	152	152	154	155	155	155	155	155	156	157	157	157	157	157	157	157	157	157	157	157	157	157	157	157	157	157	157	157	157	157
60	160	161	161	162	163	163	163	164	164	164	168	168	168	169	169	169	169	169	169	169	169	169	169	169	169	169	169	169	169	169	169
164	165	166	166	167	167	167	167	168	168	168	168	168	168	168	168	168	168	168	168	168	168	168	168	168	168	168	168	168	168	168	168
71	171	172	172	172	172	173	173	174	175	175	175	175	175	176	176	176	176	176	176	176	176	176	176	176	176	176	176	176	176	176	176
78	178	178	179	179	181	181	181	181	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182	182
182	182	182	182	183	183	184	184	184	184	184	186	186	187	188	188	188	188	188	188	188	188	188	188	188	188	188	188	188	188	188	188
90	191	191	191	192	192	193	194	195	196	196	196	196	197	197	197	197	197	197	197	197	197	197	197	197	197	197	197	197	197	197	197
00	200	201	201	202	202	202	202	202	203	203	207	207	208	208	208	209	210	210	210	210	210	210	210	210	210	210	210	210	210	210	210
203	203	204	204	204	206	206	206	206	207	207	207	207	207	207	207	207	207	207	207	207	207	207	207	207	207	207	207	207	207	207	207
Ln 1, Col 1 Spaces:4 UTF-8 CRLF Java																															

## Работа сортировки Шелла.

ers\sony\AppData\Local\Temp\vscode\sws_77361\jdt_ws\jdt.ls-java-project\bin\TournamentSort																															
0	0	1	1	1	2	2	2	2	3	4	4	5	5	6	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	
8	8	8	9	9	10	10	10	10	12	12	12	12	12	12	13	13	14	15	15	15	15	15	15	15	15	15	15	15	15	15	
5	15	17	17	17	18	18	18	18	18	18	23	23	23	24	25	26	26	27	27	27	27	27	27	27	27	27	27	27	27	27	
19	19	19	20	20	21	21	21	22	22	22	23	23	23	24	25	26	26	27	27	27	27	27	27	27	27	27	27	27	27	27	
7	27	28	29	30	30	30	31	32	32	32	33	33	33	34	34	34	35	35	35	35	35	35	35	35	35	35	35	35	35	35	
6	37	37	37	37	38	38	38	38	38	38	46	47	47	47	47	47	48	49	50	51	51	51	51	51	51	51	51	51	51	51	
39	39	39	40	41	45	45	46	46	46	46	46	47	47	47	47	47	48	49	50	51	51	51	51	51	51	51	51	51	51	51	
1	51	51	51	52	52	52	52	53	54	54	54	54	54	55	56	57	57	58	58	58	58	58	58	58	58	58	58	58	58	58	
8	59	59	59	60	60	60	60	60	60	60	66	66	66	66	66	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	
61	61	63	63	64	64	64	64	65	65	65	66	66	66	66	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	
1	71	71	71	71	72	73	73	74	74	74	74	75	75	75	75	76	76	78	78	78	78	78	78	78	78	78	78	78	78	78	
9	80	80	80	81	81	81	82	82	82	82	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	
82	82	82	82	83	85	85	85	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	86	
0	90	91	91	92	92	92	92	93	94	94	94	95	95	95	95	95	97	98	98	98	98	98	98	98	98	98	98	98	98	98	
9	99	99	100	101	102	103	103	103	104	104	104	104	104	104	104	104	104	104	104	104	104	104	104	104	104	104	104	104	104	104	
104	105	105	105	105	106	106	106	106	107	107	107	108	108	108	109	109	109	110	111	111	111	111	111	111	111	111	111	111	111	111	
11	113	113	113	113	114	114	114	115	115	115	115	115	116	117	117	117	118	118	118	118	118	1									
18	119	120	120	120	122	122	122	123	123	123	123	123	123	123	123	123	123	124	124	124	124	124	124	124	124	124	124	124	124	124	
123	124	124	124	124	124	124	125	125	125	125	125	125	125	125	125	126	126	127	127	127	127	127	127	127	127	127	127	127	127	127	
28	129	129	129	129	129	130	130	130	131	131	131	132	132	133	133	133	134	134	134	134	134	134	134	134	134	134	134	134	134	134	
35	136	136	137	137	137	138	138	140	140	140	140	140	140	140	140	140	140	140	140	140	140	140	140	140	140	140	140	140	140	140	
140	140	141	142	142	143	143	143	143	145	146	147	149	150	150	150	150	151	152	152	153	153	153	153	153	153	153	153	153	153	153	
53	154	155	155	155	155	156	157	157	157	157	158	158	158	158	160	160	160	161	161	161	161	161	161	161	161	161	161	161	161	161	
61	162	163	163	164	165	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	166	
166	167	167	167	168	169	169	170	170	170	170	171	171	171	172	172	173	173	173	174	174	174	174	174	174	174	174	174	174	174	174	
74	175	175	175	175	176	176	176	176	176	176	176	177	177	178	178	178	178	178	178	178	178	178	178	178	178	178	178	178	178	178	
81	181	181	182	182	182	182	182	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	
Ln 1, Col Spaces: 4 UTF-8 CRLF Java JavaSE-14																															

```
ers\sony\AppData\Local\Temp\vscodesws_d7211\jdt_ws\jdt.ls-java-project\bin' 'QuickSort'
```

0	0	0	2	2	2	3	3	4	5	5	5	6	7	7	7	8	9	9	10	1
0	11	12	12	12	12	13	13	14	14	15	16	16	16	16	16	16	17	18	18	1
8	18	19	20	20	22	23	23	24	24											
24	24	25	25	25	26	26	27	27	27	27	27	28	29	29	31	31	31	32	32	3
2	33	34	34	34	35	35	35	37	37	37	37	37	38	38	38	39	39	39	39	4
0	40	41	41	42	42	42	43	43	43											
44	46	46	47	48	48	48	49	50	51	51	51	51	52	52	52	53	53	53	53	5
3	53	54	54	54	55	55	56	56	56	57	58	58	59	59	59	59	59	59	59	6
1	61	61	62	62	63	63	63	63	63											
63	64	64	65	65	66	66	67	67	67	67	68	70	71	73	73	73	77	78	79	7
9	80	81	81	81	81	82	82	82	83	84	85	85	85	85	85	86	86	87	87	8
8	88	88	89	89	90	90	90	90	91	92										
92	92	93	93	94	94	95	95	95	96	96	97	97	97	97	98	98	98	98	100	1
00	101	101	101	102	102	102	102	103	103	103	104	105	106	106	108	108	108	108	109	1
10	111	111	112	112	115	115	115	116	116											
116	116	116	116	117	117	117	118	119	119	121	122	123	123	123	124	124	124	125	125	1
25	125	125	126	126	126	127	127	128	129	129	130	130	130	130	131	131	131	131	132	1
33	133	133	133	133	133	134	134	134	135											
135	136	136	137	138	138	138	138	139	139	139	139	139	140	140	140	142	142	142	142	1
42	143	144	144	144	145	145	146	146	146	146	148	150	150	150	151	152	153	153	153	1
54	154	155	155	155	155	156	156	157	157											
157	158	158	158	159	159	159	160	160	160	161	161	161	163	163	164	164	164	165	166	1
66	166	166	166	166	167	168	169	169	169	169	169	170	170	170	170	171	171	171	171	1
71	172	173	173	174	174	175	175	175	176											
176	177	177	178	178	179	179	179	180	181	181	182	182	183	183	183	183	184	184	184	1
85	185	185	185	185	186	186	187	188	190	190	190	190	191	191	191	192	192	192	192	1
92	193	193	193	193	194	194	194	194	195											
195	195	196	197	197	198	198	198	199	199	199	200	200	201	201	201	201	202	202	202	2
02	203	205	205	205	205	206	206	207	208	209	209	210	211	211	211	212	212	212	212	2
12	212	213	213	214	214	215	216	216	216											
217	218	219	219	219	219	219	219	220	221	221	222	222	222	222	223	225	226	226	226	2

## Работа быстрой сортировки.

### Вывод:

Реализовали методы заданных сортировок, произвели оценку времени работы каждого алгоритма.