

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

Выполнил: студент группы БВТ1901

Кускова А. Е.

Руководитель:

Мелехин А.

Москва 2021

Задание:

Реализовать методы сортировки строк числовой матрицы в соответствии с заданием. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки. Испытания проводить на сгенерированных матрицах.

Решение:

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

Код для пирамидальной сортировки:

```
import java.util.Random;

public class HeapSort {

    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 поддереве
            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;
    }
}

```

Код для быстрой сортировки:

```

import java.util.Random;

public class QuickSort {
    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 i = 0; i < 50; i++) {
    for (int j = 0; j < 50; j++) {
        System.out.print(array2d[i][j] + "\t");
    }
    System.out.println();
}
System.out.println();

for (int v : toArray(array2d))
    System.out.print(v);

System.out.println();
System.out.println();
*/

int low = 0;
int high = toArray(array2d).length - 1;

/*for (int v : quickSort(toArray(array2d), low, high))
    System.out.print(v);

System.out.println();
System.out.println();*/

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

```

```

        long time = System.currentTimeMillis();
        quickSort(toArray(array2d), low, high);
        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[] quickSort(int[] array, int low, int high) {
        // выбрать опорный элемент
        int middle = low + (high - low) / 2;
        int opora = array[middle];

        // разделить на подмассивы, который больше и меньше опорного элемента
        int i = low, j = high;

```

```

        while (i <= j) {
            while (array[i] < opora) {
                i++;
            }

            while (array[j] > opora) {
                j--;
            }

            if (i <= j) { //меняем местами
                int temp = array[i];
                array[i] = array[j];
                array[j] = temp;
                i++;
                j--;
            }
        }

        // вызов рекурсии для сортировки левой и правой части
        if (low < j)
            quickSort(array, low, j);
        if (high > i)
            quickSort(array, i, high);
        return array;
    }
}

```

Код для сортировки обменом:

```

import java.util.Random;

public class BubbleSort {
    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 для сортировки вставкой:

```

import java.util.Random;

public class InsertionSort {
    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 для сортировки выбором:

```

import java.util.Random;

public class SelectionSort {
    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 для 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 для турнирной сортировки:

```

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;
}
}

```

Работа программы

На рисунках 1-2 изображена работа программы.

C:\Users\Анна\jdk\openjdk-14.0.2\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.1\lib\idea_rt.jar=64																																							
2	2	2	2	3	3	3	3	4	4	4	4	5	6	6	6	6	6	7	8	8	8	9	9	10	10	11	11	11	12	13	14	14	15	15	15				
22	22	23	23	24	24	24	25	26	26	26	27	27	27	28	28	28	30	31	31	31	31	32	32	32	33	33	34	35	35	35	35	36	36	36					
44	44	45	46	46	46	46	47	48	48	48	48	48	49	49	50	51	51	51	52	52	53	53	54	54	55	55	57	58	58	59	59	60	61	61	61				
65	66	66	69	69	69	69	70	70	71	71	71	71	71	71	71	71	72	72	72	72	74	74	74	75	75	76	76	77	77	78	78	79	79	80	80				
86	87	87	88	88	88	88	89	89	90	91	92	93	94	95	96	97	98	98	98	100	100	101	101	102	102	102	103	103	104	104	104	105	105	105	105				
109	109	109	109	110	110	110	110	111	112	112	112	112	113	113	113	114	115	116	116	116	118	118	118	118	118	119	119	119	119	120	120	121	121	121	121	121			
129	130	130	130	130	131	131	131	132	133	133	133	135	135	136	136	136	137	137	138	138	138	139	139	139	140	141	142	142	143	144	144	144	145	146	146				
153	153	153	153	153	154	154	155	155	156	156	156	157	157	157	157	158	158	158	159	160	161	161	162	162	163	164	164	164	165	165	165	167	167	167	167				
173	174	174	175	175	175	175	176	176	176	177	177	179	179	179	179	180	180	181	181	181	181	181	182	182	182	183	183	183	184	185	185	185	186	186	186				
189	190	190	190	190	190	191	191	192	192	193	193	194	194	194	194	195	196	196	196	196	197	197	198	198	199	200	200	200	201	201	202	203	203	203	204	204			
209	209	209	211	211	212	212	213	213	214	214	214	215	215	216	217	218	219	220	220	221	221	221	221	222	222	222	222	223	223	223	224	224	227	227	227				
236	236	237	237	238	238	239	239	240	240	240	241	243	243	243	244	244	244	244	244	244	245	245	246	246	246	246	247	247	248	249	250	250	251	251	251				
256	257	257	257	257	257	257	258	258	258	258	259	259	259	261	261	262	262	263	263	263	263	264	264	264	264	265	265	265	266	266	266	266	266	267	267				
272	273	273	273	273	274	275	275	276	277	277	278	279	279	279	279	280	280	280	281	281	281	282	283	283	283	284	286	286	287	287	288	288	288	288					
294	295	295	297	297	297	297	298	298	298	298	299	299	299	299	300	300	300	300	302	302	303	304	305	305	305	306	306	306	306	306	307	307	308	308	308				
313	313	313	313	315	315	316	317	317	317	318	320	320	321	321	321	321	322	323	323	324	324	324	324	324	324	325	325	325	328	328	328	328	329	329	33				
335	335	336	337	337	337	338	339	339	339	340	340	340	340	341	341	342	342	343	343	343	343	343	343	344	344	344	344	344	345	345	346	346	346	346					
349	349	350	350	350	351	352	352	353	353	353	354	354	356	356	357	357	357	358	358	359	359	359	359	359	359	360	361	362	362	362	363	363	363	363					
369	369	371	371	371	371	371	371	371	371	371	372	372	372	373	373	373	374	374	375	376	376	376	377	377	378	378	378	378	379	379	380	380	381	381					
387	387	388	388	389	389	390	390	391	391	391	393	394	394	394	395	395	396	396	397	398	399	399	399	399	399	400	400	400	401	401	402	402	402	402					
408	408	409	410	410	411	411	411	411	411	411	412	412	416	417	417	419	419	420	420	421	421	422	423	424	425	425	426	426	426	426	427	427	428	428					
433	433	434	434	434	435	435	436	436	436	437	437	439	439	439	440	440	440	440	441	441	441	441	441	442	442	442	443	443	443	444	444	444	444	445	445				
449	449	450	450	450	451	451	452	452	452	452	453	454	454	454	454	454	455	455	455	455	456	456	456	456	456	456	456	456	457	457	457	457	458	458					
463	463	463	465	466	466	466	466	469	469	470	470	471	471	472	474	474	474	475	475	475	475	476	476	476	476	477	477	477	478	479	479	480	480	480					

Рисунок 1 – Работа сортировки обменом.

C:\Users\AHNA\.jdk\openjdk-14.0.2\bin\java.exe --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.1\lib\idea_rt.jar=61																																							
0	0	0	0	0	1	2	2	2	4	4	4	4	4	4	7	7	7	7	8	8	8	8	9	10	10	11	11	12	13	13	14	14	15	15	16				
19	20	20	20	22	22	22	23	24	25	26	26	26	27	27	27	27	27	28	28	28	28	29	29	29	30	30	31	32	32	33	33	33	34	34					
40	40	40	40	40	41	42	42	43	43	44	44	44	46	46	46	46	48	49	49	49	51	52	52	52	52	53	53	53	53	53	54	56	56	56					
61	61	62	62	62	63	64	64	64	65	65	66	66	66	68	68	68	69	69	69	69	70	70	70	70	71	72	73	73	73	74	75	75	76	76					
81	82	82	82	83	83	84	84	85	85	85	86	86	86	86	87	87	87	87	89	89	90	90	91	91	92	92	93	93	94	94	95	95	96	96					
98	99	99	99	99	100	100	100	101	101	101	101	102	102	102	102	103	103	103	103	105	105	106	106	106	107	107	107	108	109	110	110	111	111	112	112				
117	118	118	118	119	119	119	120	120	120	121	121	122	122	123	123	123	123	125	125	126	126	126	126	127	127	128	128	128	129	130	130	131	131	131					
136	137	137	137	138	138	139	139	139	139	140	141	141	141	141	142	142	143	143	143	143	143	145	146	146	146	148	148	149	149	149	149	150	150	151	151				
157	157	157	158	158	158	159	159	159	159	160	160	161	161	161	161	162	162	162	162	163	163	163	163	163	164	164	164	164	165	165	165	165	167	168	168				
175	177	178	178	179	179	180	180	180	181	181	181	182	182	183	184	184	184	184	186	186	186	186	187	188	188	188	189	189	189	189	189	190	191	191	191				
198	199	200	200	200	200	200	201	201	202	202	202	202	202	203	203	204	204	204	204	204	205	205	205	205	206	207	207	208	208	210	210	212	212	212	212				
220	228	228	228	228	221	222	222	222	222	222	224	224	224	224	225	225	226	226	226	226	226	226	227	227	228	228	229	229	230	230	232	232	233	233	233				
237	238	238	238	239	239	239	240	240	241	241	241	241	241	241	244	244	245	245	246	247	247	247	247	247	249	249	249	249	250	251	251	252	252	252	252				
257	258	258	258	259	259	259	260	262	262	263	264	264	265	265	265	266	267	267	268	269	269	269	270	270	270	271	271	271	271	271	272	272	273	274	277				
279	279	279	279	281	281	281	282	282	284	284	284	284	285	285	285	286	286	287	287	288	288	288	288	289	291	291	291	291	291	292	292	293	293	293					
299	299	299	299	300	300	300	300	301	301	301	302	302	303	304	304	304	304	304	304	305	305	305	306	306	306	306	307	307	307	308	310	310	311	311	311				
315	316	316	317	317	318	318	319	319	319	320	321	321	321	322	323	323	323	323	324	324	324	324	324	325	325	326	326	326	326	327	328	329	329	330	33				
336	336	336	336	337	337	337	337	338	338	338	339	340	340	340	341	341	342	342	343	343	345	346	346	346	348	348	348	348	349	349	350	350	351	351	35				
357	358	358	358	360	360	361	362	362	363	363	364	364	364	365	365	365	365	366	366	366	366	367	367	368	369	369	369	370	370	371	373	373	373	374	37				
379	379	379	379	380	380	381	381	381	381	381	382	382	382	382	383	383	383	384	385	385	385	386	387	387	388	389	389	389	390	390	392	393	393	393					
481	482	483	483	484	484	484	484	484	484	485	486	486	487	487	487	489	489	489	489	489	489	489	489	489	489	489	489	489	489	489	489	489	489	489					
421	421	422	422	423	424	424	424	424	424	424	425	425	425	425	426	427	427	427	427	428	428	428	429	430	432	432	432	433	433	433	433	433	434	434	435				
440	440	441	441	442	442	442	443	443	443	443	443	443	444	446	446	447	447	447	448	448	449	449	449	450	450	450	450	451	452	453	453	453	453	454	454				
459	459	459	459	460	461	461	461	461	462	463	463	463	463	464	464	464	464	464	465	465	465	465	465	465	465	466	466	467	467	467	468	468	468	469	469				

[illegible]

Рисунок 3 – Работа сортировки выбором.

```
C:\Users\AHHa\jdk\openjdk-14.0.2\bin>java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.1\lib\idea_rt.jar;-x1
0 0 0 1 2 3 4 4 5 6 6 7 7 8 8 8 8 8 9 9 9 9 10 10 10 11 11 11 11 11 12 12 13 13 13 13 14 14 14
8 0 0 20 20 21 21 21 21 22 22 23 23 24 25 25 26 26 26 27 27 28 29 29 29 30 30 32 32 32 33 34 34
40 41 41 41 41 41 41 41 42 42 42 42 43 43 43 43 44 44 44 45 45 46 46 47 47 47 47 48 48 48 49 50 50 50 58
53 53 54 54 54 55 55 55 56 56 58 58 59 60 60 60 61 61 61 62 63 63 64 64 65 65 66 66 67 67 68 68 68 70 70 78
79 79 88 81 81 82 82 84 85 86 87 88 88 88 88 88 89 89 90 90 91 91 91 91 91 92 92 93 93 93 94 94 94 94 95
101 101 101 102 103 103 103 103 103 105 106 106 106 107 107 108 108 108 110 111 111 111 112 112 112 113 113 115 116 117 117 117 118 118 118 11
125 125 126 127 127 128 128 129 131 131 131 131 132 132 132 132 132 133 133 133 133 134 134 135 136 136 137 137 138 138 139 139 14
146 146 146 147 147 148 149 149 150 150 150 151 151 151 151 153 153 153 153 155 155 156 156 156 157 157 158 158 158 158 159 161 161 161 163 16
171 171 171 171 172 172 172 172 173 173 174 174 175 176 177 178 179 179 179 179 179 179 179 180 180 181 181 181 183 183 184 184 184 185 18
189 190 190 190 190 191 191 191 192 192 193 193 193 194 194 195 195 195 195 196 196 197 197 197 198 198 199 199 200 200 201 201 201 202 202 20
211 211 212 212 212 213 213 213 213 214 214 214 214 215 215 215 215 216 216 216 216 217 217 217 218 219 220 220 221 221 221 222 222 222 222 223 22
228 228 229 229 229 230 230 231 231 232 232 232 233 233 233 234 234 235 236 236 237 237 238 238 239 239 240 241 241 242 243 244 244 245 24
252 252 254 254 255 255 255 255 255 256 256 256 257 259 259 260 260 261 261 262 263 263 264 264 264 264 264 265 265 265 265 265 265 26
272 272 272 272 272 273 274 274 275 275 275 276 276 276 277 277 278 278 278 279 280 280 280 280 282 283 283 284 284 285 287 287 288 288 288 28
295 295 296 296 297 297 297 298 298 299 299 299 299 300 300 301 301 301 301 302 302 302 303 304 304 305 306 306 307 307 307 308 308 308 308 30
313 313 313 314 315 315 315 315 316 316 317 317 317 318 319 319 319 319 319 320 321 321 321 322 322 323 323 323 324 324 324 324 326 326 32
332 332 332 333 333 333 333 333 334 334 335 335 335 336 337 337 339 340 340 341 342 342 342 342 343 343 343 343 344 344 344 345 345 346 34
351 352 353 353 354 354 354 354 355 355 356 356 357 357 357 358 358 358 358 359 359 360 360 361 361 362 362 363 363 363 364 364 366 36
372 372 372 373 374 374 375 376 376 376 377 377 377 378 378 378 379 380 380 380 381 381 382 382 382 383 384 386 386 387 387 387 388 388 389 389 38
397 397 397 397 398 398 398 399 399 400 400 400 401 402 402 402 403 403 404 404 404 405 405 405 405 405 405 406 406 406 406 407 408 408 4
413 414 414 414 414 415 415 415 416 416 417 417 418 418 418 418 419 419 420 420 420 421 422 422 422 423 423 423 423 425 426 426 426 427 42
432 432 433 433 433 434 434 435 435 435 436 436 436 437 437 437 438 438 439 440 440 440 441 441 441 441 443 444 445 445 445 445 446 446 447 44
454 454 455 455 456 456 458 459 460 460 461 461 462 462 463 463 464 464 465 467 467 467 467 468 468 468 469 469 470 471 471 471 471 471 472 47
478 478 478 478 478 479 479 479 480 481 482 482 482 483 483 484 484 484 484 485 485 485 485 485 486 486 486 487 487 487 488 488 488 488 488 48
```

Рисунок 4 – Работа сортировки вставкой.

```
C:\Users\AHMA\jdk\openjdk-14.0.2\bin>java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.1\lib\idea_rt.jar-61
0 0 0 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 4 5 5 6 6 6 7 7 8 8 9 9 10 10 11 11 11 12 12 13 13 13
20 21 21 21 22 23 23 23 25 25 26 26 26 26 26 26 27 28 28 29 29 29 30 30 30 31 31 31 32 33 33 33 33 33 34 35
38 38 38 39 39 39 39 42 43 44 44 44 44 45 46 47 47 47 48 49 49 49 50 51 51 51 52 53 53 53 54 54 54 55 55 55
61 63 63 63 64 64 64 65 66 66 66 66 66 67 67 67 68 68 68 68 68 68 68 68 69 69 69 70 70 70 71 71 72 73 73 73
82 83 83 83 83 84 84 86 86 87 87 87 88 88 88 89 89 89 90 90 90 90 91 93 93 93 94 94 95 95 95 95 98 98 98 99 99 99
185 185 185 186 186 186 186 187 187 188 188 189 110 110 110 111 111 112 112 112 112 112 113 113 114 114 114 115 115 115 116 116 116 117 117 117
124 124 124 124 125 125 125 125 126 126 126 126 126 127 127 127 127 127 128 128 128 128 129 129 129 129 130 130 131 131 132 133 134 136 137 13
142 142 142 142 142 142 142 142 143 143 143 143 144 144 145 145 145 146 146 146 147 147 147 147 148 149 149 150 150 150 151 152 152 152 153 15
160 160 160 161 161 162 162 163 163 163 164 165 165 166 166 166 166 167 167 168 168 169 169 169 170 171 171 171 172 172 172 172 172 173 17
178 178 178 179 179 180 180 180 181 182 182 183 183 183 184 185 186 186 187 187 188 188 188 189 189 189 190 190 190 191 191 192 192 193 19
200 200 201 201 201 201 201 202 202 202 202 203 203 203 204 204 204 204 205 205 205 205 206 206 206 208 208 209 209 209 210 210 211 211 211 212 212 21
218 218 219 219 219 220 220 220 221 221 222 222 222 222 223 223 223 223 223 224 224 224 226 227 227 227 227 227 228 229 229 229 229 229 22
235 236 236 236 236 236 237 237 238 238 238 238 238 238 239 239 239 239 240 240 241 241 241 242 242 243 243 243 245 246 247 247 248 249 251 25
256 256 256 256 257 257 258 258 258 259 259 260 260 261 261 261 262 262 262 263 263 264 264 265 265 266 267 267 267 268 268 268 269 270 270 271 27
276 277 277 277 277 278 278 279 279 279 280 281 281 281 282 282 282 283 284 285 285 287 287 287 287 288 288 288 289 290 290 291 291 293 295 295 29
301 301 301 301 303 303 303 304 304 305 305 306 307 308 308 308 310 310 311 312 312 313 313 314 314 314 315 315 317 318 318 318 318 319 319 320 32
326 327 327 328 328 328 329 329 329 329 330 330 331 331 331 332 332 332 333 334 335 335 336 336 337 337 337 337 338 338 338 339 339 339 340 34
345 346 346 346 346 346 346 347 347 347 348 348 349 350 350 351 351 353 353 353 353 353 353 353 354 355 355 355 356 356 356 356 357 357 35
366 367 368 369 370 370 370 371 371 372 372 372 373 373 373 373 373 373 373 374 374 374 374 374 374 375 375 375 375 376 376 377 377 377 377
384 384 384 384 386 386 386 386 387 387 389 389 389 389 390 390 391 391 392 392 392 392 393 393 393 393 394 395 395 396 397 397 398 398 39
407 408 409 409 409 410 410 411 411 412 412 413 413 414 414 415 415 416 416 416 417 418 418 418 419 419 419 420 420 420 420 421 421 42
427 428 428 428 429 429 430 431 431 432 434 434 434 434 434 436 436 436 436 437 437 437 438 438 439 441 441 441 441 441 443 443 443 444 445 44
449 450 451 451 452 452 453 454 454 454 454 455 455 455 456 456 456 457 458 458 459 459 459 459 459 460 460 461 461 462 462 462 462 462 46
469 470 471 472 472 473 473 474 474 474 474 475 475 475 476 477 478 478 478 479 479 480 480 480 481 481 481 482 482 483 483 483 485 485 48
```


Рисунок 5 – Работа пирамидальной сортировки.

Рисунок 6 – Работа турнирной сортировки.

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