Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Лабораторная работа №1

по дисциплине

«Алгоритмы и структуры данных»

Выполнил: Студент группы P3217

Сергачев Данила Дмитриевич

Преподаватели:

Романов Алексей Андреевич и

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2018 г

Задание №1

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дан массив целых чисел. Ваша задача — отсортировать его в порядке неубывания с помощью сортировки слиянием.

Чтобы убедиться, что Вы действительно используете сортировку слиянием, мы просим Вас, после каждого осуществленного слияния (то есть, когда соответствующий подмассив уже отсортирован!), выводить индексы граничных элементов и их значения.

Решение

```
package
week2;

            import mooc.*;
            import java.util.Arrays;

            public class task1 {
                public static void main(String[] args){
                    try (EdxIO io = EdxIO.create()) {
                       int count = io.nextInt();
                       int[] array = new int[count];

                       for(int i = 0; i < count; i++){
                            array[i] = io.nextInt();
                       }
                       StringBuilder st = new StringBuilder();
                       int[] res = sortMerge(0,array, io, st);
                       if(st.length() > 0){
                        io.print(st.toString());
                       }
                      for(int i = 0; i < res.length; i++){
                        io.print(String.format("%d ", res[i]));
                       }
                    }
                }

                private static int[] sortMerge(int startIndex, int[] arr, EdxIO io,
            StringBuilder st) {
                    int len = arr.length;
                    if(len == 1){
```

```
            return arr;
        }

        int[] leftArr = sortMerge(startIndex, Arrays.copyOfRange(arr, 0, len/2),
io, st);
        int[] rightArr = sortMerge(len/2 + startIndex, Arrays.copyOfRange(arr,
len/2, len), io, st);

        return merge(startIndex, leftArr, rightArr, io, st);
    }

    private static int[] merge(int startIndex, int[] arr1, int[] arr2, EdxIO
io,StringBuilder st) {
        int i = 0, j = 0, currCount = 0;

        int[] res = new int[arr1.length + arr2.length];

        while(i < arr1.length || j < arr2.length){
            if(j == arr2.length || (i < arr1.length && arr1[i] <= arr2[j])){
                res[currCount++] = arr1[i++];
            }
            else{
                res[currCount++] = arr2[j++];
            }
        }

        st.append(String.format("%d %d %d %d\n", startIndex + 1, startIndex +
arr1.length +   arr2.length, res[0], res[arr1.length + arr2.length - 1]));



        return res;
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.937	183934976	1039245	4403713
1	OK	0.125	21819392	25	105
2	OK	0.109	21766144	6	2
3	OK	0.093	21762048	8	13
4	OK	0.140	21762048	8	13
5	OK	0.125	21762048	42	154
6	OK	0.140	21798912	43	154
7	OK	0.125	21774336	51	178
8	OK	0.125	21803008	45	161
9	OK	0.140	21827584	105	330
10	OK	0.125	21786624	110	343
11	OK	0.140	21807104	107	336
12	OK	0.156	23072768	461	2044
13	OK	0.156	22925312	560	2331
14	OK	0.171	22818816	388	1822
15	OK	0.140	23097344	408	1883
16	OK	0.140	23011328	1042	3776
17	OK	0.140	22962176	1043	3784
18	OK	0.140	23023616	1044	3775
19	OK	0.281	29118464	5587	25513
20	OK	0.312	29724672	6733	28937
21	OK	0.265	29007872	4737	22960
22	OK	0.281	29503488	5685	25799
23	OK	0.265	29564928	10383	39968
24	OK	0.265	30040064	10421	40060
25	OK	0.281	29491200	10420	40057
26	OK	0.656	75759616	65880	305388
27	OK	0.656	77266944	77550	340376
28	OK	0.656	74608640	57488	280213

Задание №2

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Инверсией в последовательности чисел A называется такая ситуация, когда $i < j$, а $A_i > A_j$.

Дан массив целых чисел. Ваша задача — подсчитать число инверсий в нем.

Подсказка: чтобы сделать это быстрее, можно воспользоваться модификацией сортировки слиянием.

Решение

```
package
week2;

        import mooc.*;
        import java.util.Arrays;

        public class task2 {
            public static void main(String[] args){
                try (EdxIO io = EdxIO.create()) {
                    int count = io.nextInt();
                    int[] array = new int[count];

                    for(int i = 0; i < count; i++){
                         array[i] = io.nextInt();
                    }

                    LongCount inversionCount = new LongCount();
                    int[] res = sortMerge(0,array, inversionCount);

                    io.print(inversionCount.count);
                }
            }

            private static int[] sortMerge(int startIndex, int[] arr, LongCount
        inversionCount) {

                int len = arr.length;
                if(len == 1){
                    return arr;
                }

                int[] leftArr = sortMerge(startIndex, Arrays.copyOfRange(arr, 0, len/2),
        inversionCount);
                int[] rightArr = sortMerge(len/2 + startIndex, Arrays.copyOfRange(arr,
        len/2, len),
                 inversionCount);

                return merge(startIndex, leftArr, rightArr, inversionCount);
            }

            private static int[] merge(int startIndex, int[] arr1, int[] arr2,LongCount
        inversionCount) {
                int i = 0, j = 0, currCount = 0;

                int[] res = new int[arr1.length + arr2.length];
```

```
        while(i < arr1.length || j < arr2.length){

            if(j == arr2.length || (i < arr1.length && arr1[i] <= arr2[j])){
                res[currCount++] = arr1[i++];
            }
            else{
                if(i< arr1.length){
                    inversionCount.count += arr1.length - i;
                }

                res[currCount++] = arr2[j++];
            }
        }

        return res;
    }

    public static class LongCount{
        public LongCount(){
            count = 0;
        }
        public long count;
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.265	44195840	1039245	10
1	OK	0.125	21557248	25	2
2	OK	0.203	21602304	6	1
3	OK	0.125	21585920	8	1
4	OK	0.109	21581824	8	1
5	OK	0.109	21561344	42	1
6	OK	0.140	21581824	43	2
7	OK	0.125	21573632	51	1
8	OK	0.140	21626880	45	2
9	OK	0.125	21544960	105	2
10	OK	0.125	21610496	110	2
11	OK	0.109	21610496	107	2
12	OK	0.125	21622784	461	1
13	OK	0.109	21618688	560	4
14	OK	0.125	21610496	388	1
15	OK	0.125	21630976	408	4
16	OK	0.125	21614592	1042	4
17	OK	0.140	21610496	1043	4
18	OK	0.156	21630976	1044	4
19	OK	0.140	21880832	5587	1
20	OK	0.109	21831680	6733	6
21	OK	0.125	21803008	4737	1
22	OK	0.093	21872640	5685	6
23	OK	0.125	22024192	10383	6
24	OK	0.140	21991424	10421	6
25	OK	0.109	21962752	10420	6
26	OK	0.109	24748032	65880	1
27	OK	0.140	24768512	77550	8
28	OK	0.109	24809472	57488	1
29	OK	0.093	24862720	68090	8

Задание №3

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Для сортировки последовательности чисел широко используется быстрая сортировка — QuickSort. Далее приведена программа, которая сортирует массив a, используя этот алгоритм.

Решение

```
package
week2;

        import mooc.*;

        public class task3 {
            public static void main(String[] args){
                    try(EdxIO io = EdxIO.create()) {
                        int count = io.nextInt();
                        int[] arr = new int[count];

                        for(int i = 0; i < count; i++){
                             arr[i] = i + 1;

                             int m = i / 2;

                             swap(arr, i, m);
                        }

                        // Поиск индекса элемента 2
                        int index2 = 0;
                        while (index2 < arr.length - 1 && arr[index2] != 2) {
                            index2++;
                        }

                        // Поиск индекса элемента 1
                        int index1 = arr.length - 1;
                        while (index1 > 0 && arr[index1] != 1) { index1--; }

                        swap(arr, index1, index2);

                        StringBuilder st = new StringBuilder();

                        for(int i =0; i < count; i++){
                            st.append(arr[i] + " ");
```

```
                }

                io.print(st.toString());
            }
        }

        private static void swap(int[] arr, int i, int j){
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.265	44195840	1039245	10
1	OK	0.125	21557248	25	2
2	OK	0.203	21602304	6	1
3	OK	0.125	21585920	8	1
4	OK	0.109	21581824	8	1
5	OK	0.109	21561344	42	1
6	OK	0.140	21581824	43	2
7	OK	0.125	21573632	51	1
8	OK	0.140	21626880	45	2
9	OK	0.125	21544960	105	2
10	OK	0.125	21610496	110	2
11	OK	0.109	21610496	107	2
12	OK	0.125	21622784	461	1
13	OK	0.109	21618688	560	4
14	OK	0.125	21610496	388	1
15	OK	0.125	21630976	408	4
16	OK	0.125	21614592	1042	4
17	OK	0.140	21610496	1043	4
18	OK	0.156	21630976	1044	4
19	OK	0.140	21880832	5587	1
20	OK	0.109	21831680	6733	6
21	OK	0.125	21803008	4737	1
22	OK	0.093	21872640	5685	6
23	OK	0.125	22024192	10383	6
24	OK	0.140	21991424	10421	6
25	OK	0.109	21962752	10420	6
26	OK	0.109	24748032	65880	1
27	OK	0.140	24768512	77550	8

Задание №4

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дан массив из n элементов. Какие числа являются k_1-ым, (k_1+1)-ым, ..., k_2-ым в порядке неубывания в этом массиве?

Формат входного файла

В первой строке входного файла содержатся три числа: n — размер массива, а также границы интервала k_1 и k_2, при этом $2 \le n \le 4 \cdot 10^7, 1 \le k_1 \le k_2 \le n, k_2 - k_1 < 200$.

Во второй строке находятся числа A, B, C, a_1, a_2, по модулю не превосходящие 10^9. Вы должны получить элементы массива, начиная с третьего, по формуле: $a_i = A \cdot a_{i-2} + B \cdot a_{i-1} + C$. Все вычисления должны производится в 32-битном знаковом типе, переполнения должны игнорироваться.

Обращаем Ваше внимание, что массив из $4 \cdot 10^7$ 32-битных целых чисел занимает в памяти **160 мегабайт**! Будьте аккуратны!

Подсказка: эту задачу лучше всего решать модификацией быстрой сортировки. Однако сортировка массива целиком по времени, скорее всего, не пройдет, поэтому нужно подумать, как модифицировать быструю сортировку, чтобы не сортировать те части массива, которые не нужно сортировать.

> Эту задачу, скорее всего, **нельзя решить ни на Python, ни на PyPy**. Мы не нашли способа сгенерировать $4 \cdot 10^7$ 32-битных целых чисел и при этом уложиться в ограничение по времени. Если у Вас тоже не получается, попробуйте другой язык программирования, например, **Cython** (расширение файла `*.pyx`).

Решение

```
package
week2;
          import mooc.*;

          public class task4 {
             public static void main(String[] args){
                 try (EdxIO io = EdxIO.create()) {
                     int n = io.nextInt();
                     int[] arr = new int[n];

                     int start = io.nextInt() - 1;
                     int end = io.nextInt() - 1;
                     int A = io.nextInt();
                     int B = io.nextInt();
                     int C = io.nextInt();
                     arr[0] = io.nextInt();
                     arr[1] = io.nextInt();

                     for (int i = 2; i < n; i++) {
```

```
            arr[i] = A * arr[i - 2] + B * arr[i - 1] + C;
           }

           StringBuilder st = new StringBuilder();

           searchItems(arr, start, end, 0, arr.length - 1, 0, arr.length - 1,
st);

           io.print(st.toString());
       }
   }

   private static void searchItems(int[] array, int kIndex, int kIndexEnd,
        int startIndex, int endIndex, int absStart, int absEnd, StringBuilder
st){

    if(endIndex < startIndex){
        return;
    }

    int middleIndex = startIndex + (endIndex - startIndex) / 2;
    int start = startIndex, end = endIndex;

    while(start < end){
        while((array[start] <= array[middleIndex]) && start < middleIndex) {
start++; }

        while((array[middleIndex] <= array[end]) && end > middleIndex) { end--; }

        if(start < end){
            swap(array, start, end);
            // Задание новых границ
            if(start == middleIndex){
                middleIndex = end;
            }
            else if(end == middleIndex){
                middleIndex = start;
            }
        }
    }

    if(middleIndex == kIndex){
        st.append(array[middleIndex] + " ");
        if(kIndex == kIndexEnd){
            return;
        }
        else{
            searchItems(array, kIndex + 1, kIndexEnd,
```

```
                middleIndex + 1, absEnd, absStart, absEnd, st);
            }
        }
        else if(middleIndex < kIndex){
            absStart = middleIndex + 1;
            searchItems(array, kIndex, kIndexEnd,
                middleIndex + 1, endIndex, absStart, absEnd, st);
        }
        else{
            if(middleIndex > kIndexEnd){
                absEnd = middleIndex - 1;
            }
            searchItems(array, kIndex, kIndexEnd,
                absStart, middleIndex - 1, absStart, absEnd, st);
        }
    }

    private static void swap(int[] array, int i, int j){
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.156	183869440	54	2400
1	OK	0.109	21581824	18	6
2	OK	0.156	21581824	28	9
3	OK	0.156	21594112	32	4
4	OK	0.109	21565440	33	5
5	OK	0.109	21544960	32	10
6	OK	0.109	21585920	33	5
7	OK	0.140	21565440	32	19
8	OK	0.140	21569536	32	21
9	OK	0.125	21565440	25	300
10	OK	0.093	21577728	22	382
11	OK	0.125	21659648	23	477
12	OK	0.140	21594112	35	12
13	OK	0.109	21573632	38	11
14	OK	0.109	21602304	36	1074
15	OK	0.140	21606400	36	561
16	OK	0.125	21569536	37	220
17	OK	0.125	21856256	24	400
18	OK	0.140	21991424	28	1200
19	OK	0.140	22020096	29	1400
20	OK	0.156	21655552	37	12
21	OK	0.125	21639168	45	11
22	OK	0.125	22040576	38	2400
23	OK	0.109	21954560	39	2400
24	OK	0.156	22089728	44	2200
25	OK	0.125	22102016	43	2200
26	OK	0.125	21704704	41	676
27	OK	0.187	22659072	28	600
28	OK	0.187	22962176	31	1400
29	OK	0.156	23293952	32	1600
30	OK	0.156	22974464	37	12
31	OK	0.140	23126016	48	11

Задание №5

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

«Сортировка пугалом» — это давно забытая народная потешка, которую восстановили по летописям специалисты платформы «Открытое образование» специально для этого курса.

Участнику под верхнюю одежду продевают деревянную палку, так что у него оказываются растопырены руки, как у огородного пугала. Перед ним ставятся n матрёшек в ряд. Из-за палки единственное, что он может сделать — это взять в руки две матрешки на расстоянии k друг от друга (то есть i-ую и $(i+k)$-ую), развернуться и поставить их обратно в ряд, таким образом поменяв их местами.

Задача участника — расположить матрёшки по неубыванию размера. Может ли он это сделать?

Решение

```
package
week2;
          import java.util.ArrayList;
          import java.util.List;

          import mooc.*;

          public class task5 {
             public static void main(String[] args){
                 try (EdxIO io = EdxIO.create()) {
                     int n = io.nextInt();
                     int k = io.nextInt();

                     int[] arr = new int[n];

                     for(int i =0; i < n; i++){
                         arr[i] = io.nextInt();
                     }

                     ArrayList<ArrayList> a = new ArrayList<ArrayList>();
                     for(int i = 0; i < k; i++){
                         a.add(new ArrayList());
                         for (int j = i; j < arr.length; j += k){
                              a.get(i).add(arr[j]);
                         }
                         int[] intArr = new int[a.get(i).size()];
                         for(int j = 0; j < a.get(i).size(); j++){
                             intArr[j] = (int)a.get(i).get(j);
                         }

                         doSort(intArr, 0, intArr.length - 1);
```

```
                ArrayList arrList = new ArrayList();
                for(int j = 0; j < intArr.length;j++){
                    arrList.add(intArr[j]);
                }

                a.set(i, arrList);
            }

            io.print(isSorted(arr, a, k) ? "YES" : "NO");
        }
    }

     private static void doSort(int[] array, int startIndex, int endIndex){
     if(startIndex >= endIndex){
        return;
    }

    int start = startIndex, end = endIndex;
    // Выбор индекса опорного элемента
    int middleIndex = startIndex + (endIndex - startIndex) / 2;
    while(start < end){
        while((array[start] <= array[middleIndex]) && start < middleIndex) {
start++; }

        while((array[middleIndex] <= array[end]) && end > middleIndex) { end--; }

        if(start < end){
            swap(array, start, end);
            // Задание новых границ
            if(start == middleIndex){
                middleIndex = end;
            }
            else if(end == middleIndex){
                middleIndex = start;
            }
        }
    }

    doSort(array, startIndex, middleIndex);
    doSort(array, middleIndex+1, endIndex);
}

    private static void swap(int[] arr, int i, int k){
      int temp = arr[i];
      arr[i] = arr[i+k];
      arr[i+k] = temp;
    }
```

```
    private static boolean isSorted(int[] arr, ArrayList<ArrayList> arrs, int k){
        for(int i = 1; i < arr.length;i++){
            int arrsIndex = i % k;
            int index = i / k;

            if(arr[i - 1] > (int)arrs.get(arrsIndex).get(index)){
                return false;
            }
            else{
                arr[i] = (int)arrs.get(arrsIndex).get(index);
            }
        }
        return true;
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.328	28405760	1039313	3
1	OK	0.109	21622784	12	2
2	OK	0.125	21594112	16	3
3	OK	0.125	21577728	112	3
4	OK	0.109	21618688	111	2
5	OK	0.109	21602304	112	3
6	OK	0.125	21569536	112	2
7	OK	0.140	21626880	109	3
8	OK	0.140	21614592	112	2
9	OK	0.125	21643264	110	3
10	OK	0.156	21581824	111	2
11	OK	0.125	21569536	108	3
12	OK	0.109	22147072	11674	3
13	OK	0.140	22134784	11707	2
14	OK	0.125	21913600	11712	3
15	OK	0.140	21835776	11754	2
16	OK	0.109	21905408	11708	3
17	OK	0.125	21987328	11740	2
18	OK	0.140	21901312	11726	3
19	OK	0.109	21798912	11680	2
20	OK	0.125	21864448	11741	3
21	OK	0.171	23715840	128736	3
22	OK	0.125	23506944	128832	2
23	OK	0.203	24092672	128751	3
24	OK	0.109	23273472	128866	2
25	OK	0.109	23756800	128700	3
26	OK	0.125	23339008	128707	2
27	OK	0.125	23502848	128729	3
28	OK	0.125	23138304	128807	2
29	OK	0.109	23572480	128784	3
30	OK	0.265	25481216	1039313	3
31	OK	0.312	25878528	1038610	2
32	OK	0.312	25812992	1038875	3