САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет: Программной инженерии и компьютерной техники
Направление подготовки: 09.03.04 (Программная инженерия)

Дисциплина «Алгоритмы и структуры данных»

Отчёт по лабораторной работе Неделя №2

OpenEdu

Группа: P3217
Выполнил: Минин Александр

г. Санкт-Петербург
2018г.

Задача №1

Условие

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.

Формат входного файла

В первой строке входного файла содержится число n ($1 \le n \le 10^5$) — число элементов в массиве. Во второй строке находятся n целых чисел, по модулю не превосходящих 10^9.

Формат выходного файла

Выходной файл состоит из нескольких строк.

В **последней строке** выходного файла требуется вывести отсортированный в порядке неубывания массив, данный на входе. Между любыми двумя числами должен стоять ровно один пробел.

Все предшествующие строки описывают осуществленные слияния, по одному на каждой строке. Каждая такая строка должна содержать по четыре числа: I_f I_l V_f V_l, где I_f — индекс начала области слияния, I_l — индекс конца области слияния, V_f — значение первого элемента области слияния, V_l — значение последнего элемента области слияния.

Все индексы начинаются с единицы (то есть, $1 \le I_f \le I_l \le n$). **Индексы области слияния должны описывать положение области слияния в исходном массиве!** Допускается не выводить информацию о слиянии для подмассива длиной 1, так как он отсортирован по определению.

> Приведем небольшой пример: отсортируем массив [9, 7, 5, 8]. Рекурсивная часть сортировки слиянием (процедура SORT(A, L, R), где A — сортируемый массив, L — индекс начала области слияния, R — индекс конца области слияния) будет вызвана с $A = [9, 7, 5, 8]$, $L = 1$, $R = 4$ и выполнит следующие действия:

- разделит область слияния $[1; 4]$ на две части, $[1; 2]$ и $[3; 4]$;
- выполнит вызов SORT(A, $L = 1$, $R = 2$):
  - разделит область слияния $[1; 2]$ на две части, $[1; 1]$ и $[2; 2]$;
  - получившиеся части имеют единичный размер, рекурсивные вызовы можно не делать;
  - осуществит слияние, после чего A станет равным $[7, 9, 5, 8]$;
  - выведет описание слияния: $I_f = L = 1$, $I_l = R = 2$, $V_f = A_L = 7$, $V_l = A_R = 9$.
- выполнит вызов SORT(A, $L = 3$, $R = 4$):
  - разделит область слияния $[3; 4]$ на две части, $[3; 3]$ и $[4; 4]$;
  - получившиеся части имеют единичный размер, рекурсивные вызовы можно не делать;
  - осуществит слияние, после чего A станет равным $[7, 9, 5, 8]$;
  - выведет описание слияния: $I_f = L = 3$, $I_l = R = 4$, $V_f = A_L = 5$, $V_l = A_R = 8$.
- осуществит слияние, после чего A станет равным $[5, 7, 8, 9]$;
- выведет описание слияния: $I_f = L = 1$, $I_l = R = 4$, $V_f = A_L = 5$, $V_l = A_R = 9$.

Описания слияний могут идти в произвольном порядке, необязательно совпадающем с порядком их выполнения. Однако, с целью повышения производительности, рекомендуем выводить эти описания сразу, не храня их в памяти. Именно по этой причине отсортированный массив выводится в самом конце.

Решение

```
package week2;

import java.io.*;

public class Week2_1 {

    private static BufferedReader bufferedReader;
    private static BufferedWriter bufferedWriter;

    // l1 < l2; r1 < r2
    public static void merge(int[] array, int l1, int r1, int l2, int r2) throws IOException {
        bufferedWriter.write((l1+1) + " ");
        bufferedWriter.write((r2) + " ");
        int i = 0;
        int size = r2 - l1;
        int destStart = l1;
        int[] tmp = new int[size];
        while (i < size) {
            while (l1 < r1 && (l2 >= r2 || array[l1] <= array[l2])) { // stable
                tmp[i++] = array[l1];
                l1++;
            }
            while (l2 < r2 && (l1 >= r1 || array[l2] < array[l1])) {
                tmp[i++] = array[l2];
                l2++;
            }
        }


        for (int k = 0; k < size; k++) {
            array[destStart + k] = tmp[k];
        }

        bufferedWriter.write(array[destStart] + " ");
        bufferedWriter.write(array[r2 - 1] + "\n");
    }

    public static void mergeSort(int[] array, int l, int r) throws IOException {
        if (l == r - 1) {
            return;
        }
        mergeSort(array, l, l + (r - l) / 2);
        mergeSort(array, l + (r - l) / 2, r);
        merge(array, l, l + (r - l) / 2, l + (r - l) / 2, r);
    }

    public static void main(String[] args) throws IOException {
        bufferedReader = new BufferedReader(new FileReader("input.txt"));
        bufferedWriter = new BufferedWriter(new FileWriter("output.txt"));

        int arraySize = Integer.valueOf(bufferedReader.readLine());
        String[] arrayStr = bufferedReader.readLine().split(" ");
        int[] array = new int[arraySize];
        for (int i = 0; i < arraySize; i++) {
            array[i] = Integer.valueOf(arrayStr[i]);
        }

        mergeSort(array, 0, array.length);
```

```
        for (int i = 0; i < arraySize; i++) {
            bufferedWriter.write(Integer.toString(array[i]));
            if (i + 1 == arraySize) {
                bufferedWriter.write("\n");
            } else {
                bufferedWriter.write(" ");
            }
        }

        bufferedWriter.flush();
        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Результаты бенчмарка:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1031	7737753	1039245	4303714
1	OK	125	2097152	25	96
2	OK	125	2087321	6	2
3	OK	125	2090188	8	12
4	OK	125	2095104	8	12
5	OK	109	2100838	42	145
6	OK	109	2087731	43	145
7	OK	109	2092646	51	169
8	OK	109	2097971	45	152
9	OK	140	2088140	105	321
10	OK	109	2091417	110	334
11	OK	109	2099609	107	327
12	OK	156	2103705	461	1945
13	OK	125	2106572	560	2232
14	OK	93	2107392	388	1723
15	OK	125	2101248	408	1784
16	OK	125	2102476	1042	3677
17	OK	156	2107392	1043	3685
18	OK	125	2105344	1044	3676
19	OK	156	2277376	5587	24514
20	OK	156	2289664	6733	27938
21	OK	140	2267955	4737	21961
22	OK	140	2291712	5685	24800

23	OK	125	23437312	10383	38969
24	OK	140	23314432	10421	39061
25	OK	140	23461888	10420	39058
26	OK	218	34578432	65880	295389
27	OK	265	34598912	77550	330377
28	OK	234	34242560	57488	270214
29	OK	218	34639872	68090	301998
30	OK	281	35651584	103872	410190
31	OK	265	35414016	103940	410367
32	OK	265	35344384	103842	410123
33	OK	890	76845056	758839	3454252
34	OK	937	77377536	875802	3805103
35	OK	765	73273344	675241	3203455
36	OK	812	73154560	782803	3526114
37	OK	1000	76423168	1038992	4303249
38	OK	1031	76173312	1038702	4302564
39	OK	1000	76005376	1039245	4303714

Задача №2

Условие

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 java.io.*;

public class Week2_2 {

    private static BufferedReader bufferedReader;
    private static BufferedWriter bufferedWriter;
    private static long inversionsNumber = 0;

    // l1 < l2; r1 < r2
    public static void merge(int[] array, int l1, int r1, int l2, int r2) throws IOException {
        int i = 0;
        int size = r2 - l1;
```

```
        int destStart = l1;
        int[] tmp = new int[size];
        while (i < size) {
            while (l1 < r1 && (l2 >= r2 || array[l1] <= array[l2])) { // stable
                tmp[i++] = array[l1];
                l1++;
            }
            while (l2 < r2 && (l1 >= r1 || array[l2] < array[l1])) {
                tmp[i++] = array[l2];
                inversionsNumber += r1 - l1;
                l2++;
            }
        }

        for (int k = 0; k < size; k++) {
            array[destStart + k] = tmp[k];
        }
    }

    public static void mergeSort(int[] array, int l, int r) throws IOException {
        if (l == r - 1) {
            return;
        }
        mergeSort(array, l, l + (r - l) / 2);
        mergeSort(array, l + (r - l) / 2, r);
        merge(array, l, l + (r - l) / 2, l + (r - l) / 2, r);
    }

    public static void main(String[] args) throws IOException {
        bufferedReader = new BufferedReader(new FileReader("input.txt"));
        bufferedWriter = new BufferedWriter(new FileWriter("output.txt"));

        int arraySize = Integer.valueOf(bufferedReader.readLine());
        String[] arrayStr = bufferedReader.readLine().split(" ");
        int[] array = new int[arraySize];
        for (int i = 0; i < arraySize; i++) {
            array[i] = Integer.valueOf(arrayStr[i]);
        }

        mergeSort(array, 0, array.length);

        bufferedWriter.write(inversionsNumber + "\n");

        bufferedWriter.flush();
        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Результаты бенчмарка

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		390	49926144	1039245	11
1	OK	109	20856832	25	3
2	OK	109	20905984	6	2
3	OK	140	20955136	8	2

4	OK	93	20885504	8	2
5	OK	109	20975616	42	2
6	OK	125	20926464	43	3
7	OK	93	20971520	51	2
8	OK	125	20930560	45	3
9	OK	125	20910080	105	3
10	OK	109	20951040	110	3
11	OK	109	20930560	107	3
12	OK	125	20893696	461	2
13	OK	109	20963328	560	5
14	OK	109	20963328	388	2
15	OK	109	20885504	408	5
16	OK	109	20922368	1042	5
17	OK	109	20996096	1043	5
18	OK	109	20893696	1044	5
19	OK	156	21557248	5587	2
20	OK	125	21475328	6733	7
21	OK	125	21393408	4737	2
22	OK	125	21438464	5685	7
23	OK	125	22212608	10383	7
24	OK	125	22089728	10421	7
25	OK	140	22171648	10420	7
26	OK	171	25346048	65880	2
27	OK	156	25620480	77550	9
28	OK	187	24788992	57488	2
29	OK	156	25595904	68090	9
30	OK	171	25829376	103872	9
31	OK	156	25829376	103940	9
32	OK	171	25894912	103842	9
33	OK	296	48173056	758839	2
34	OK	281	48787456	875802	11
35	OK	265	48009216	675241	2
36	OK	328	48787456	782803	11
37	OK	343	49926144	1038992	11
38	OK	390	49717248	1038702	11

39	OK	375	49582080	1039245	11

Задача №3

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

```
var a : array [1..N] of integer;

procedure QSort(left, right : integer);
var i, j, key, buf : integer;
begin
   key := a[(left + right) div 2];
   i := left;
   j := right;
   repeat
      while a[i] < key do
         inc(i);
      while key < a[j] do
         dec(j);
      if i <= j then begin
         buf := a[i];
         a[i] := a[j];
         a[j] := buf;
         inc(i);
         dec(j);
      end;
   until i > j;
   if left < j then QSort(left, j);
   if i < right then QSort(i, right);
end;
begin
   ...
   QSort(1, N);
end.
```

Хотя QuickSort является очень быстрой сортировкой в среднем, существуют тесты, на которых она работает очень долго. Оценивать время работы алгоритма будем числом сравнений с элементами массива (то есть, суммарным числом сравнений в первом и втором while). Требуется написать программу, генерирующую тест, на котором быстрая сортировка сделает наибольшее число таких сравнений.

Решение

```
package week2;

import mooc.EdxIO;



public class Week2_3 {

   private static EdxIO edxIO;

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

```
    public static void main(String[] args) {
        edxIO = EdxIO.create();

        int n = edxIO.nextInt();

        int[] a = new int[n];

        for (int i = 0; i < a.length; i++) {
            a[i] = i + 1;

            int m = (i) / 2;

            int tmp = a[i];
            a[i] = a[m];
            a[m] = tmp;
        }

        int ind2 = 0;
        while (ind2 < a.length - 1 && a[ind2] != 2) {
            ind2++;
        }
        int ind1 = a.length - 1;
        while (ind1 > 0 && a[ind1] != 1) {
            ind1--;
        }
        swap(a, ind1, ind2);

        for (int i = 0; i < a.length; i++) {
            edxIO.print(a[i] + " ");
        }


        edxIO.close();
    }
}
```

Результаты бенчмарка

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		468	77754368	9	6888896
1	OK	109	21577728	3	6
2	OK	109	21504000	3	2
3	OK	125	21606400	3	4
4	OK	109	21618688	3	8
5	OK	109	21561344	3	10
6	OK	125	21561344	3	12
7	OK	109	21643264	3	14
8	OK	125	21581824	3	16
9	OK	109	21594112	3	18
10	OK	125	21639168	4	21

11	OK	93	21528576	4	36
12	OK	140	21528576	5	292
13	OK	93	21700608	6	3893
14	OK	109	23867392	7	48900
15	OK	109	23650304	7	48894
16	OK	296	41144320	8	756195
17	OK	328	56557568	8	1556239
18	OK	343	58286080	8	3151812
19	OK	468	77742080	8	6888888
20	OK	406	77754368	9	6888896

Задача №4

Условие

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`).

Формат выходного файла

В первой и единственной строке выходного файла выведите k_1-ое, (k_1+1)-ое, ..., k_2-ое в порядке неубывания числа в массиве a. Числа разделяйте одним пробелом.

Решение

```
package week2;

import mooc.EdxIO;

import java.io.IOException;
```

```
/**
 * Asc/desc enhancements are commented cause they are not required to pass :3
 * though some tests run faster with them (but most of them are actually slower =D)
 */
public class Week2_4 {

    private static EdxIO edxIO;

    private static void swap(int[] array, int i, int j) throws IOException {
        int tmp = array[i];
        array[i] = array[j];
        array[j] = tmp;
    }

    private static void searchKth(int[] array, int kIndex, int kEnd, int startIndex, int endIndex, int
absStart, int absEnd) throws IOException {
        if (endIndex < startIndex) {
            return;
        }

        int pivot = startIndex + (endIndex - startIndex) / 2; // startIndex <= pivot < endIndex
        int l = startIndex;
        int r = endIndex;
        while (l < r) {
            while (l < pivot) {
                if (array[l] > array[pivot]) {
                    swap(array, l, pivot);
                    pivot = l;
                } else {
                    l++;
                }
            }

            while (r > pivot) {
                if (array[r] < array[pivot]) {
                    swap(array, pivot, r);
                    pivot = r;
                } else {
                    r--;
                }
            }
        }

        if (pivot == kIndex) {
            edxIO.print(array[pivot]);
            edxIO.print(" ");
            if (kIndex == kEnd) {
                return;
            } else {
                searchKth(array, kIndex + 1, kEnd, pivot + 1, absEnd, absStart, absEnd);
            }
        } else if (pivot < kIndex) {
            absStart = pivot + 1;
            searchKth(array, kIndex, kEnd, pivot + 1, endIndex, absStart, absEnd);
        } else {
            if (pivot > kEnd) {
                absEnd = pivot - 1;
            }
            searchKth(array, kIndex, kEnd, absStart, pivot - 1, absStart, absEnd);
```

```
        }
    }

    public static void main(String[] args) throws IOException {
        edxIO = EdxIO.create();

        int n = edxIO.nextInt();
        int kStart = edxIO.nextInt() - 1;
        int kEnd = edxIO.nextInt() - 1;
        int[] a = new int[n];
        int A = edxIO.nextInt();
        int B = edxIO.nextInt();
        int C = edxIO.nextInt();
        a[0] = edxIO.nextInt();
        a[1] = edxIO.nextInt();


        for (int i = 2; i < n; i++) {
            a[i] = A * a[i - 2] + B * a[i - 1] + C;
        }

        searchKth(a, kStart, kEnd, 0, a.length - 1, 0, a.length - 1);

        edxIO.close();
    }
}
```

Результаты бенчмарка

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1218	183427072	54	2400
1	OK	125	21532672	18	6
2	OK	125	21626880	28	9
3	OK	109	21557248	32	4
4	OK	109	21577728	33	5
5	OK	125	21610496	32	10
6	OK	109	21540864	33	5
7	OK	125	21585920	32	19
8	OK	125	21557248	32	21
9	OK	109	21598208	25	300
10	OK	125	21647360	22	382
11	OK	109	21577728	23	477
12	OK	109	21557248	35	12
13	OK	109	21594112	38	11
14	OK	125	21606400	36	1074
15	OK	125	21585920	36	561

16	OK	109	21606400	37	220
17	OK	109	21721088	24	400
18	OK	109	21798912	28	1200
19	OK	109	21876736	29	1400
20	OK	109	21651456	37	12
21	OK	109	21655552	45	11
22	OK	125	21790720	38	2400
23	OK	125	21782528	39	2400
24	OK	125	21725184	44	2200
25	OK	109	21762048	43	2200
26	OK	125	21688320	41	676
27	OK	140	22732800	28	600
28	OK	140	22859776	31	1400
29	OK	156	22900736	32	1600
30	OK	156	22519808	37	12
31	OK	156	22700032	48	11
32	OK	156	22933504	40	2400
33	OK	140	22839296	40	2400
34	OK	171	22937600	47	2200
35	OK	140	22855680	46	2200
36	OK	156	22425600	45	200
37	OK	156	26447872	32	800
38	OK	187	26746880	34	1600
39	OK	187	26681344	35	1800
40	OK	171	26361856	38	12
41	OK	171	26791936	49	11
42	OK	171	26480640	40	2400
43	OK	218	26820608	40	2003
44	OK	171	26357760	49	2200
45	OK	218	26550272	47	2200
46	OK	203	26902528	48	560
47	OK	437	183275520	33	800
48	OK	500	183287808	39	2000
49	OK	578	183259136	40	2200
50	OK	718	183160832	40	12

51	OK	609	183050240	52	11
52	OK	656	183160832	42	2400
53	OK	703	183250944	42	2400
54	OK	781	183074816	54	2200
55	OK	812	183144448	54	2200
56	OK	906	183164928	52	1076
57	OK	781	183152640	53	2200
58	OK	890	183304192	52	2076
59	OK	671	183123968	54	2035
60	OK	671	183345152	53	1859
61	OK	953	183181312	51	2208
62	OK	1218	183189504	49	2189
63	OK	718	183128064	53	2057
64	OK	937	183263232	54	1991
65	OK	828	183177216	50	2004
66	OK	828	183128064	52	1793
67	OK	625	183427072	54	1930

Задача №5

Условие

«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?

Формат входного файла

В первой строчке содержатся числа n и k $(1 \le n, k \le 10^5)$ — число матрёшек и размах рук.

Во второй строчке содержится n целых чисел, которые по модулю не превосходят 10^9 — размеры матрёшек.

Формат выходного файла

Выведите «YES», если возможно отсортировать матрёшки по неубыванию размера, и «NO» в противном случае.

Решение

```
package week2;

import mooc.EdxIO;

import java.io.IOException;


public class Week2_5 {

    private static EdxIO edxIO;

    private static void swap(int[] array, int i, int j) throws IOException {
        int tmp = array[i];
        array[i] = array[j];
        array[j] = tmp;
    }

    private static boolean quickSort(int[][] a, int k, int startIndex, int endIndex, int span) throws
IOException {
        if (endIndex < startIndex) {
            return true;
        }

        int pivot = startIndex + (endIndex - startIndex) / 2; // startIndex <= pivot < endIndex
        int l = startIndex;
        int r = endIndex;
        while (l < r) {
            while (l < pivot) {
                if (a[k][l] > a[k][pivot]) {
                    swap(a[k], l, pivot);
                    pivot = l;
                } else {
                    l++;
                }
            }

            while (r > pivot) {
                if (a[k][r] < a[k][pivot]) {
                    swap(a[k], pivot, r);
                    pivot = r;
                } else {
                    r--;
                }
            }
        }

        // checking: prev subsequence is less or equal
        if (k != 0 && a[k - 1][pivot] > a[k][pivot]) {
            return false;
        }
        // checking: last subsequence > first
        if (span > 1 && k == span - 1 && a[0].length > pivot + 1 && a[0][pivot + 1] < a[k][pivot]) {
            return false;
        }

        boolean result = true;
        result = quickSort(a, k, startIndex, pivot - 1, span);
        if (!result) {
            return result;
        }
```

```
        return quickSort(a, k, pivot + 1, endIndex, span);

    }

    public static void printArray(int[][] array, int n, int span) {
        edxIO.println();
        int i = 0;
        int k = 0;
        while (i < n) {
            edxIO.print(array[k][i / span]+  " ");
            i++;
            k++;
            if (k == span) {
                k = 0;
            }
        }
    }

    public static void main(String[] args) throws IOException {
        edxIO = EdxIO.create();

        int n = edxIO.nextInt();
        int span = edxIO.nextInt();

        int[][] array = new int[span][n / span];
        for (int i = 0; i < n % span; i++) {
            array[i] = new int[n / span + 1];
        }

        int i = 0;
        int k = 0;
        while (i < n) {
            array[k][i / span] = edxIO.nextInt();
            i++;
            k++;
            if (k == span) {
                k = 0;
            }
        }

        for (k = 0; k < span; k++) {
            if (!quickSort(array, k, 0, array[k].length - 1, span)) {
                edxIO.print("NO");
//                printArray(array, n, span);
                edxIO.close();
                return;
            }
        }
        edxIO.print("YES");
//      printArray(array, n, span);
        edxIO.close();
    }
}
```

Результаты бенчмарка

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		328	28418048	1039313	3

1	OK	125	21602304	12	2
2	OK	156	21577728	16	3
3	OK	109	21557248	112	3
4	OK	93	21630976	111	2
5	OK	125	21504000	112	3
6	OK	109	21577728	112	2
7	OK	125	21610496	109	3
8	OK	140	21512192	112	2
9	OK	109	21594112	110	3
10	OK	125	21602304	111	2
11	OK	109	21577728	108	3
12	OK	156	22122496	11674	3
13	OK	125	22110208	11707	2
14	OK	125	21901312	11712	3
15	OK	171	21848064	11754	2
16	OK	109	21958656	11708	3
17	OK	125	21966848	11740	2
18	OK	109	21880832	11726	3
19	OK	109	21884928	11680	2
20	OK	125	21831680	11741	3
21	OK	140	23674880	128736	3
22	OK	109	23580672	128832	2
23	OK	109	23695360	128751	3
24	OK	109	23273472	128866	2
25	OK	109	23703552	128700	3
26	OK	125	23318528	128707	2
27	OK	125	23584768	128729	3
28	OK	125	23035904	128807	2
29	OK	125	23662592	128784	3
30	OK	234	25677824	1039313	3
31	OK	187	25604096	1038610	2
32	OK	328	25915392	1038875	3
33	OK	187	26836992	1038723	2
34	OK	187	28418048	1038749	3
35	OK	171	25915392	1038747	2

36	OK	250	25935872	1039043	3
37	OK	171	25464832	1039210	2
38	OK	203	27111424	1038967	3