Санкт-Петербургский Национальный Исследовательский
Университет Информационных Технологий, Механики и Оптики
Мегафакультет компьютерных технологий и управления

Дисциплина
«Алгоритмы и структуры данных»

Лабораторная работа №2

Выполнил:
Студент группы P3218
Масалкин Савелий Евгеньевич
Преподаватель:
Муромцев Дмитрий Ильич

Санкт-Петербург,
2018

1. Сортировка слиянием

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, $1 \le If \le Il \le n$). 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.

Исходный код (python):

```
def merge_array(array, buffer_array, first_array_begin, first_array_end,
second_array_begin, second_array_end):
    sorted_elements_count = 0
    begin = first_array_begin
    end = second_array_end

    while (first_array_begin <= first_array_end and second_array_begin
<= second_array_end):
        if (array[first_array_begin] < array[second_array_begin]):
            buffer_array[sorted_elements_count] = array[first_array_begin]
            first_array_begin += 1
        else:
            buffer_array[sorted_elements_count] =
            array[second_array_begin] second_array_begin += 1
        sorted_elements_count += 1

    for index in range(first_array_begin, first_array_end + 1):
        buffer_array[sorted_elements_count] = array[index]
        sorted_elements_count += 1
    for index in range(second_array_begin, second_array_end + 1):
        buffer_array[sorted_elements_count] = array[index]
        sorted_elements_count += 1
```

```
    array[begin:end + 1] = buffer_array[0:sorted_elements_count]

    file_output.write(f"{begin + 1} {end + 1} {array[begin]} {array[end]}\n")
    return

def sort_array(array, file_output):
    size = len(array)
    buffer_array = [0 for i in range(size)]

    merging_size = 1
    while merging_size < size:
        first_array_begin = 0
        while first_array_begin < size:
            second_array_end = first_array_begin + merging_size * 2 - 1

            first_array_end = (first_array_begin + second_array_end) // 2 -
(first_array_begin + second_array_end + 1) % 2
            second_array_begin = first_array_end + 1

            if (second_array_begin < size):
                if (second_array_end >= size):
                    second_array_end = size - 1
                merge_array(array, buffer_array, first_array_begin,
first_array_end, second_array_begin, second_array_end)

            first_array_begin += merging_size * 2
        merging_size *= 2
    return

with open("input.txt", "r") as file_input:
    size = int(file_input.readline())
    array = [int(x) for x in file_input.read().split()]

with open("output.txt", "w") as file_output:
    sort_array(array, file_output)
    for element in array:
        file_output.write(f"{element} ")
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.968	33280000	1039245	4403485
1	OK	0.078	18046976	25	103
2	OK	0.078	18042880	6	2
3	OK	0.093	18051072	8	13
4	OK	0.093	18038784	8	13
5	OK	0.078	18010112	42	153
6	OK	0.078	18038784	43	154
7	OK	0.093	18042880	51	176
8	OK	0.093	18063360	45	159
9	OK	0.078	18046976	105	330
10	OK	0.078	18001920	110	341
11	OK	0.093	18030592	107	334
12	OK	0.078	18001920	461	2037
13	OK	0.062	18034688	560	2325
14	OK	0.078	17997824	388	1817
15	OK	0.078	18046976	408	1871
16	OK	0.093	18042880	1042	3772
17	OK	0.062	18018304	1043	3779
18	OK	0.062	18034688	1044	3772
19	OK	0.171	20271104	5587	25513
20	OK	0.109	20176896	6733	28932
21	OK	0.093	19521536	4737	22965
22	OK	0.109	20275200	5685	25794
23	OK	0.140	20254720	10383	39982
24	OK	0.109	19259392	10421	40067
25	OK	0.125	20602880	10420	40050
26	OK	0.250	23490560	65880	305381
27	OK	0.203	23646208	77550	340355
28	OK	0.281	24461312	57488	280204

2. Число инверсий

Инверсией в последовательности чисел A называется такая ситуация, когда i<j, а Ai>Aj.

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

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

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

В выходной файл надо вывести число инверсий в массиве.

Исходный код (python):

```
def merge_array(array, buffer_array, first_array_begin, first_array_end,
second_array_begin, second_array_end):
    sorted_elements_count = 0
    begin = first_array_begin
    end = second_array_end

    inversions_count = 0

    while (first_array_begin <= first_array_end and second_array_begin
<= second_array_end):
        if (array[first_array_begin] <= array[second_array_begin]):
            buffer_array[sorted_elements_count] = array[first_array_begin]
            first_array_begin += 1
        else:
            buffer_array[sorted_elements_count] =
            array[second_array_begin] second_array_begin += 1
            inversions_count += first_array_end - first_array_begin + 1
        sorted_elements_count += 1

    for index in range(first_array_begin, first_array_end + 1):
        buffer_array[sorted_elements_count] = array[index]
        sorted_elements_count += 1

    for index in range(second_array_begin, second_array_end + 1):
        buffer_array[sorted_elements_count] = array[index]
        sorted_elements_count += 1

    array[begin:end + 1] = buffer_array[0:sorted_elements_count]

    return inversions_count

def sort_array(array):
    size = len(array)
    buffer_array = [0] * size

    inversions_count = 0
    sorting_size = 2
    while (sorting_size < size * 2):
        first_array_begin = 0
        while (first_array_begin < size):
            second_array_end = first_array_begin + sorting_size - 1
```

```
            first_array_end = (first_array_begin + second_array_end) // 2 -
(first_array_begin + second_array_end + 1) % 2
            second_array_begin = first_array_end + 1

            if (second_array_begin < size):
                if (second_array_end >= size):
                    second_array_end = size - 1
                inversions_count += merge_array(array, buffer_array,
first_array_begin, first_array_end, second_array_begin, second_array_end)

            first_array_begin += sorting_size
        sorting_size *= 2
    return inversions_count

with open("input.txt", "r") as file_input:
    size = int(file_input.readline())
    array = [int(x) for x in file_input.read().split()]

with open("output.txt", "w") as file_output:
    file_output.write(str(sort_array(array)))
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.328	33095680	1039245	10
1	OK	0.062	17956864	25	2
2	OK	0.093	18022400	6	1
3	OK	0.093	18022400	8	1
4	OK	0.062	17977344	8	1
5	OK	0.062	18010112	42	1
6	OK	0.062	18018304	43	2
7	OK	0.093	17977344	51	1
8	OK	0.062	18018304	45	2
9	OK	0.078	18034688	105	2
10	OK	0.062	17977344	110	2
11	OK	0.062	18022400	107	2
12	OK	0.062	18026496	461	1
13	OK	0.078	17977344	560	4
14	OK	0.078	18014208	388	1
15	OK	0.078	18026496	408	4
16	OK	0.062	17969152	1042	4
17	OK	0.078	18022400	1043	4
18	OK	0.062	18034688	1044	4
19	OK	0.078	19374080	5587	1

3. Анти-quick sort

Требуется написать программу, генерирующую тест, на котором быстрая сортировка сделает наибольшее число таких сравнений.

Формат входного файла

В первой строке находится единственное число n ($1 \le n \le 10^6$).

Формат выходного файла

Вывести перестановку чисел от 1 до n, на которой быстрая сортировка выполнит максимальное число сравнений. Если таких перестановок несколько, вывести любую из них.

Исходный код (python):

```
def add_element(array, n):
    array.append(n)
    middle = (n - 1) // 2
    array[n - 1], array[middle] = array[middle], array[n - 1]

with open("input.txt", "r") as file_input:
    size = int(file_input.readline())

array = [1, 2]
for ind in range(2, size):
    add_element(array, ind + 1)

with open("output.txt", "w") as file_output:
    for ind in range(0, min(size, len(array))):
        file_output.write(f"{array[ind]} ")
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.140	31956992	9	6888896
1	OK	0.062	17944576	3	6
2	OK	0.062	17887232	3	2
3	OK	0.078	17928192	3	4
4	OK	0.078	17903616	3	8
5	OK	0.062	17920000	3	10
6	OK	0.062	17973248	3	12
7	OK	0.078	17879040	3	14
8	OK	0.078	17948672	3	16
9	OK	0.062	17948672	3	18
10	OK	0.078	17903616	4	21

4. K-ая порядковая статистика

Дан массив из n элементов. Какие числа являются k1-ым, (k1+1)-ым, …, k2-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 k1 и k2, при этом $2 \le n \le 4 \cdot 10^7$, $1 \le k1 \le k2 \le n$, $k2 - k1 < 200$.

Во второй строке находятся числа A, B, C, a1, a2, по модулю не превосходящие 10^9. Вы должны получить элементы массива, начиная с третьего, по формуле: $ai = A \cdot ai-2 + B \cdot ai-1 + C$. Все вычисления должны производится в 32-битном знаковом типе, переполнения должны игнорироваться.

Формат выходного файла

В первой и единственной строке выходного файла выведите k1-ое, (k1+1)-ое, …, k2-ое в порядке неубывания числа в массиве a. Числа разделяйте одним пробелом.

Исходный код (C#):

```
using System;
using System.IO;

namespace ADS
{
    class Program
    {
        private static StreamReader streamReader;
        private static StreamWriter streamWriter;

        public static void Quicksort(int[] elements, int left, int right, int k1, int k2)
        {
            if (left >= right || left > k2 - 1 || right < k1 - 1)
                return;

            int i = left, j = right;
            int pivot = elements[(left + right) / 2];
            while (i <= j)
            {
                while (elements[i].CompareTo(pivot) < 0)
                    i++;
                while (elements[j].CompareTo(pivot) > 0)
                    j--;

                if (i <= j)
                {
                    int tmp = elements[i];
                    elements[i] = elements[j];
                    elements[j] = tmp;
                    i++;
                    j--;
                }
            }
            Quicksort(elements, left, j, k1, k2);
            Quicksort(elements, i, right, k1, k2);
        }
```

```
        static void Main(string[] args)
        {
            streamReader = new StreamReader("input.txt");
            streamWriter = new StreamWriter("output.txt");

            string[] str = streamReader.ReadLine().Split(' ');
            int n = Convert.ToInt32(str[0]); int k1 =
            Convert.ToInt32(str[1]);
            int k2 = Convert.ToInt32(str[2]);

            str = streamReader.ReadLine().Split(' ');
            int A = Convert.ToInt32(str[0]);
            int B = Convert.ToInt32(str[1]);
            int C = Convert.ToInt32(str[2]);

            int[] array = new int[n];
            array[0] = Convert.ToInt32(str[3]);
            array[1] = Convert.ToInt32(str[4]);

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

            Quicksort(array, 0, n - 1, k1, k2);

            for (int i = k1; i <= k2; i++)
            {
                streamWriter.Write("{0} ", array[i - 1]);
            }
            streamReader.Close();
            streamWriter.Close();
        }
    }
}
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.781	170463232	54	2400
1	OK	0.031	10178560	18	6
2	OK	0.031	10113024	28	9
3	OK	0.015	10129408	32	4
4	OK	0.031	10072064	33	5
5	OK	0.031	10121216	32	10
6	OK	0.015	10084352	33	5
7	OK	0.031	10117120	32	19
8	OK	0.031	10170368	32	21
9	OK	0.031	10096640	25	300
10	OK	0.031	10108928	22	382

5. Сортировка пугалом

«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.

Исходный код (python):

```
file = open("input.txt", "r")
                                firstLine = list(map(int, file.read().split()))
                                quantity = firstLine[0]
                                armSpan = firstLine[1]
                                array = firstLine[2:]

                                arrays = []
                                for i in range(armSpan):
                                    tmpArr = []
                                    for j in range (i, quantity, armSpan):
                                        tmpArr.append(array[j])
                                    tmpArr.sort()
                                    arrays.append(tmpArr)
                                result = "YES"
                                for i in range(1, quantity):
                                    if arrays[i % armSpan][i // armSpan] < arrays[(i - 1)
                                % armSpan][(i - 1) // armSpan]:
                                        result = "NO"
                                        break
                                file = open("output.txt", "w")
                                file.write(result)
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.296	33112064	1039313	3
1	OK	0.078	17833984	12	2
2	OK	0.062	17866752	16	3
3	OK	0.062	17891328	112	3
4	OK	0.125	17915904	111	2
5	OK	0.062	17846272	112	3
6	OK	0.062	17862656	112	2
7	OK	0.078	17879040	109	3
8	OK	0.062	17838080	112	2
9	OK	0.062	17895424	110	3
10	OK	0.078	17874944	111	2
11	OK	0.062	17891328	108	3
12	OK	0.078	19087360	11674	3
13	OK	0.062	19025920	11707	2
14	OK	0.078	19181568	11712	3
15	OK	0.078	19312640	11754	2
16	OK	0.078	19443712	11708	3
17	OK	0.078	19017728	11740	2
18	OK	0.062	19189760	11726	3
19	OK	0.062	19415040	11680	2
20	OK	0.062	19693568	11741	3
21	OK	0.093	21020672	128736	3
22	OK	0.093	20684800	128832	2
23	OK	0.093	20905984	128751	3
24	OK	0.109	21864448	128866	2
25	OK	0.109	21962752	128700	3
26	OK	0.078	20664320	128707	2
27	OK	0.093	21094400	128729	3
28	OK	0.140	21508096	128807	2
29	OK	0.093	21626880	128784	3
30	OK	0.234	33095680	1039313	3
31	OK	0.250	33083392	1038610	2
32	OK	0.250	33075200	1038875	3
33	OK	0.265	33099776	1038723	2
34	OK	0.250	32792576	1038749	3
35	OK	0.234	33103872	1038747	2
36	OK	0.234	32866304	1039043	3
37	OK	0.234	32854016	1039210	2
38	OK	0.296	33112064	1038967	3