Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа № 2

по дисциплине «Алгоритмы и структуры данных»

Openedu – неделя 2

Подготовил:

студент группы P3217

Бураков Илья Алексеевич

Преподаватели:

Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург, 2019

Сортировка слиянием

Условие

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.

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.

> **Исключением из этого правила, возможно, является PyPy** — по причине отсутствия буферизации вывода, при решения задачи на этом языке рекомендуется собрать все описания слияния в один буфер, а затем вывести его как одно целое. Обертка ввода-вывода для PyPy, расположенная на GitHub и доступная (при компиляции решения в тестирующей системе) как модуль `openedu_io`, делает это автоматически.

Корректность выданного Вами сценария сортировки слиянием проверяется специальной программой. Любая корректная сортировка слиянием, делящая подмассивы на две части (необязательно равных!), будет зачтена, если успеет завершиться, уложившись в ограничения.

Пример

input.txt	output.txt
10 1 8 2 1 4 7 3 2 3 6	1 2 1 8 3 4 1 2 1 4 1 8 5 6 4 7 1 6 1 8 7 8 2 3 9 10 3 6 7 10 2 6 1 10 1 8 1 1 2 2 3 3 4 6 7 8

Решение

openedu/week2/lab2_1.cpp

```
#include "edx-io.hpp"
#include <vector>

using namespace std;

void merge_sort(vector<int> &arr, int left, int right) {
	// left subarray index in inclusive
	// right subarray index is exclusive (!)

	// length of the subarray to sort
	int subarray_n = right - left;

	if (subarray_n == 1) {
```

```
                // already sorted, job done
                return;
        }

        // beginning of left and right subarrays
        int left_beginning = left;
        int right_beginning = left + subarray_n / 2;

        // recursively sort subarrays
        merge_sort(arr, left_beginning, right_beginning);
        merge_sort(arr, right_beginning, right);

        // merge sorted subarrays
        auto merged_arr = vector<int>(subarray_n);
        int i = left_beginning;
        int j = right_beginning;
        for (int k = 0; k < subarray_n;) {
                if (j == right || (i != right_beginning && arr[i] < arr[j])) {
                        // left subarray wins for current k
                        merged_arr[k++] = arr[i++];
                } else {
                        // right subarray wins for current k
                        merged_arr[k++] = arr[j++];
                }
        }

        // place merged_arr
        copy_n(merged_arr.begin(), subarray_n, arr.begin() + left);

        // print required info
        // (right - 1) -- inclusive
        // (right - 1) + 1 -- indexing from 1
        io << left + 1 << " " << right << " " << merged_arr.front() << " " <<
merged_arr.back() << "\n";
}

int main() {
        // read N
        int n;
        io >> n;

        // read input array
        auto arr = vector<int>(n);
        for (auto &e : arr) {
                io >> e;
        }

        // sorting
        merge_sort(arr, 0, n);

        // printing sorted array
        for (auto e : arr) {
                io << e << " ";
        }
        return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.187	3874816	1039245	4403713
1	OK	0.015	2220032	25	105
2	OK	0.015	2224128	6	2

3	OK	0.015	2236416	8	13
4	OK	0.000	2224128	8	13
5	OK	0.015	2224128	42	154
6	OK	0.000	2220032	43	154
7	OK	0.000	2236416	51	178
8	OK	0.000	2224128	45	161
9	OK	0.000	2236416	105	330
10	OK	0.000	2232320	110	343
11	OK	0.000	2232320	107	336
12	OK	0.000	2220032	461	2044
13	OK	0.015	2236416	560	2331
14	OK	0.000	2236416	388	1822
15	OK	0.015	2224128	408	1883
16	OK	0.000	2236416	1042	3776
17	OK	0.000	2224128	1043	3784
18	OK	0.000	2220032	1044	3775
19	OK	0.000	2232320	5587	25513
20	OK	0.000	2244608	6733	28937
21	OK	0.015	2228224	4737	22960
22	OK	0.000	2232320	5685	25799
23	OK	0.000	2232320	10383	39968
24	OK	0.015	2232320	10421	40060
25	OK	0.000	2232320	10420	40057
26	OK	0.015	2236416	65880	305388
27	OK	0.031	2236416	77550	340376
28	OK	0.015	2236416	57488	280213
29	OK	0.015	2236416	68090	311997
30	OK	0.015	2236416	103872	420189
31	OK	0.015	2220032	103940	420366
32	OK	0.015	2232320	103842	420122
33	OK	0.140	3596288	758839	3554251
34	OK	0.156	3710976	875802	3905102
35	OK	0.187	3506176	675241	3303454
36	OK	0.156	3620864	782803	3626113
37	OK	0.156	3874816	1038992	4403248
38	OK	0.171	3874816	1038702	4402563
39	OK	0.156	3874816	1039245	4403713

Число инверсий

Условие

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.

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.

Пример

input.txt	output.txt
10 1 8 2 1 4 7 3 2 3 6	17

Решение

openedu/week2/lab2_2.cpp

```
#include "edx-io.hpp"
#include <vector>

using namespace std;

long long count_inversions(vector<int> &arr, int left, int right) {
	// left subarray index in inclusive
	// right subarray index is exclusive (!)

	// length of the subarray to sort
	int subarray_n = right - left;

	if (subarray_n == 1) {
		// already sorted, job done
		return 0;
	}

	// beginning of left and right subarrays
	int left_beginning = left;
	int right_beginning = left + subarray_n / 2;

	// recursively sort subarrays
	long long result = 0;
	result += count_inversions(arr, left_beginning, right_beginning);
	result += count_inversions(arr, right_beginning, right);

	// merge sorted subarrays
	auto merged_arr = vector<int>(subarray_n);
	int i = left_beginning;
	int j = right_beginning;
```

```
	for (int k = 0; k < subarray_n;) {
		if (j == right || (i != right_beginning && arr[i] <= arr[j])) {
			// left subarray wins for current k
			merged_arr[k++] = arr[i++];
		} else {
			// right subarray wins for current k
			merged_arr[k++] = arr[j++];
			// Since right subarray won, all left subarray elements larger
than the element from
			// right subarray will form an inversion. Add their quantity to
result.
			result += right_beginning - i;
		}
	}

	// place merged_arr
	copy_n(merged_arr.begin(), subarray_n, arr.begin() + left);

	return result;
}

int main() {
	// read N
	int n;
	io >> n;

	// read input array
	auto arr = vector<int>(n);
	for (auto &e : arr) {
		io >> e;
	}

	// sorting
	io << count_inversions(arr, 0, n);

	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	3866624	1039245	10
1	OK	0.000	2220032	25	2
2	OK	0.000	2224128	6	1
3	OK	0.000	2220032	8	1
4	OK	0.000	2224128	8	1
5	OK	0.000	2220032	42	1
6	OK	0.000	2224128	43	2
7	OK	0.000	2236416	51	1
8	OK	0.015	2236416	45	2
9	OK	0.000	2224128	105	2
10	OK	0.000	2224128	110	2
11	OK	0.000	2224128	107	2
12	OK	0.000	2224128	461	1
13	OK	0.000	2236416	560	4
14	OK	0.000	2236416	388	1

15	OK	0.000	2236416	408	4
16	OK	0.015	2236416	1042	4
17	OK	0.000	2224128	1043	4
18	OK	0.000	2224128	1044	4
19	OK	0.000	2228224	5587	1
20	OK	0.015	2224128	6733	6
21	OK	0.015	2224128	4737	1
22	OK	0.000	2224128	5685	6
23	OK	0.000	2240512	10383	6
24	OK	0.000	2240512	10421	6
25	OK	0.000	2240512	10420	6
26	OK	0.000	2220032	65880	1
27	OK	0.000	2236416	77550	8
28	OK	0.000	2236416	57488	1
29	OK	0.000	2240512	68090	8
30	OK	0.000	2220032	103872	8
31	OK	0.031	2236416	103940	8
32	OK	0.015	2236416	103842	8
33	OK	0.031	3588096	758839	1
34	OK	0.015	3702784	875802	10
35	OK	0.031	3502080	675241	1
36	OK	0.031	3612672	782803	10
37	OK	0.031	3866624	1038992	10
38	OK	0.031	3866624	1038702	10
39	OK	0.031	3866624	1039245	10

Анти-quick sort

Условие

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.

```
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.

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.

Пример

input.txt	output.txt
3	1 3 2

Решение

openedu/week2/lab2_3.cpp

```
#include "edx-io.hpp"
#include <vector>

using namespace std;

void swap(vector<int> &arr, int a, int b) {
	int t = arr[a];
	arr[a] = arr[b];
	arr[b] = t;
}

int main() {
	// read N
	int n;
	io >> n;

	// read input array
	auto arr = vector<int>(n);
```

```
	for (int i = 0; i < n; i++) {
		arr[i] = i + 1;
	}

	// generate array
	for (int i = 2; i < n; i++) {
		swap(arr, i, i / 2);
	}

	// print arr
	//reverse(arr.begin(), arr.end());
	for (auto e : arr) {
		io << e << " ";
	}

	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.234	5824512	9	6888896
1	OK	0.000	2228224	3	6
2	OK	0.000	2228224	3	2
3	OK	0.000	2228224	3	4
4	OK	0.015	2240512	3	8
5	OK	0.000	2240512	3	10
6	OK	0.015	2228224	3	12
7	OK	0.000	2240512	3	14
8	OK	0.015	2240512	3	16
9	OK	0.015	2228224	3	18
10	OK	0.000	2228224	4	21
11	OK	0.000	2228224	4	36
12	OK	0.000	2228224	5	292
13	OK	0.000	2248704	6	3893
14	OK	0.000	2285568	7	48900
15	OK	0.015	2269184	7	48894
16	OK	0.031	2322432	8	756195
17	OK	0.062	2777088	8	1556239
18	OK	0.109	3690496	8	3151812
19	OK	0.234	5824512	8	6888888
20	OK	0.218	5820416	9	6888896

K-ая порядковая статистика

Условие

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`).

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.

Примеры

input.txt	output.txt
5 3 4 2 3 5 1 2	13 48
5 3 4 200000 300000 5 1 2	2 800005

Примечание

Во втором примере элементы массива a равны: [1, 2, 800005, −516268571, 1331571109].

Решение

openedu/week2/lab2_4.cpp

```
#include "edx-io.hpp"
#include <vector>

using namespace std;

int k1, k2;

int hoare_patrition(vector<int> &arr, int left, int right) {
```

```
	int pivot = arr[(left + right) / 2];
	int i = left - 1;
	int j = right + 1;
	while (true) {
		do i++; while (arr[i] < pivot);
		do j--; while (arr[j] > pivot);

		if (i >= j) {
			return j;
		}

		int t = arr[i];
		arr[i] = arr[j];
		arr[j] = t;
	}
}

void quick_sort(vector<int> &arr, int left, int right) {
	// if we don't need to sort that subarray, skip it!
	if (right < k1 || left > k2) return;

	if (left < right) {
		int middle = hoare_patrition(arr, left, right);
		quick_sort(arr, left, middle);
		quick_sort(arr, middle + 1, right);
	}
}

int main() {
	// read input
	int n, A, B, C;
	io >> n >> k1 >> k2;
	k1--;
	k2--;

	auto arr = vector<int>(n);
	io>> A >> B >> C >> arr[0] >> arr[1];

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

	// sort array
	quick_sort(arr, 0, n - 1);

	// print needed part
	for (int i = k1; i <= k2; i++) {
		io << arr[i]<< " ";
	}

	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.671	161816576	54	2400
1	OK	0.000	2224128	18	6
2	OK	0.000	2224128	28	9
3	OK	0.015	2224128	32	4
4	OK	0.000	2224128	33	5

5	OK	0.000	2224128	32	10
6	OK	0.000	2224128	33	5
7	OK	0.015	2236416	32	19
8	OK	0.000	2236416	32	21
9	OK	0.015	2220032	25	300
10	OK	0.000	2236416	22	382
11	OK	0.015	2224128	23	477
12	OK	0.015	2236416	35	12
13	OK	0.000	2224128	38	11
14	OK	0.015	2224128	36	1074
15	OK	0.000	2224128	36	561
16	OK	0.000	2224128	37	220
17	OK	0.000	2265088	24	400
18	OK	0.000	2265088	28	1200
19	OK	0.000	2265088	29	1400
20	OK	0.000	2265088	37	12
21	OK	0.015	2273280	45	11
22	OK	0.000	2273280	38	2400
23	OK	0.000	2265088	39	2400
24	OK	0.000	2265088	44	2200
25	OK	0.000	2260992	43	2200
26	OK	0.000	2265088	41	676
27	OK	0.000	2228224	28	600
28	OK	0.000	2236416	31	1400
29	OK	0.000	2215936	32	1600
30	OK	0.000	2220032	37	12
31	OK	0.000	2224128	48	11
32	OK	0.000	2220032	40	2400
33	OK	0.000	2228224	40	2400
34	OK	0.000	2236416	47	2200
35	OK	0.000	2224128	46	2200
36	OK	0.000	2240512	45	200
37	OK	0.015	5816320	32	800
38	OK	0.015	5816320	34	1600
39	OK	0.000	5816320	35	1800
40	OK	0.015	5816320	38	12
41	OK	0.015	5816320	49	11
42	OK	0.015	5812224	40	2400
43	OK	0.015	5816320	40	2003
44	OK	0.015	5816320	49	2200
45	OK	0.015	5816320	47	2200

46	OK	0.015	5816320	48	560
47	OK	0.343	161816576	33	800
48	OK	0.328	161816576	39	2000
49	OK	0.343	161816576	40	2200
50	OK	0.468	161816576	40	12
51	OK	0.359	161816576	52	11
52	OK	0.406	161816576	42	2400
53	OK	0.468	161816576	42	2400
54	OK	0.453	161812480	54	2200
55	OK	0.531	161816576	54	2200
56	OK	0.671	161816576	52	1076
57	OK	0.515	161816576	53	2200
58	OK	0.546	161816576	52	2076
59	OK	0.437	161816576	54	2035
60	OK	0.375	161816576	53	1859
61	OK	0.625	161816576	51	2208
62	OK	0.343	161816576	49	2189
63	OK	0.406	161816576	53	2057
64	OK	0.578	161816576	54	1991
65	OK	0.562	161816576	50	2004
66	OK	0.546	161812480	52	1793
67	OK	0.359	161816576	54	1930

Сортировка пугалом

Условие

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?

Формат входного файла

В первой строчке содержатся числа n и k $(1 \leq n, k \leq 10^5)$ — 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.

Примеры

input.txt	output.txt
3 2 2 1 3	NO
5 3 1 5 3 4 1	YES

Решение

openedu/week2/lab2_5.cpp

```
#include "edx-io.hpp"
#include <vector>
#include <algorithm>

using namespace std;

int n, k;

void sort_k_slice(vector<long long> &arr, int start) {
	auto slice = vector<long long>();
	slice.reserve(n / k);

	// extract a slice
	for (int i = start; i < n; i += k) {
		slice.push_back(arr[i]);
	}

	sort(slice.begin(), slice.end());

	// put it back
	for (int i = 0; i < slice.size(); i++) {
		arr[start + i * k] = slice[i];
	}
}

int main() {
	// read input
	io >> n >> k;

	// EDGE CASES
	// if step is 1, ANY array can be sorted
	// if n is 1, array is sorted by definition
	if (k == 1 || n == 1) {
		io << "YES";
		return 0;
	}

	// initializing array with input
```

```
	auto arr = vector<long long>(n);
	for (int i = 0; i < n; i++) {
		io >> arr[i];
	}

	// SEPARATELY sort slices of corresponding k (in pythonic terms - [i::k])
	for (int i = 0; i < k; i++) {
		sort_k_slice(arr, i);
	}

	// check if array is sorted.
	// if it's not, then it's impossible.
	bool is_pugalo_crying = false;
	for (int i = 1; i < n; i++) {
		if (arr[i] < arr[i - 1]) {
			is_pugalo_crying = true;
			break;
		}
	}

	// print needed part
	io << (is_pugalo_crying ? "NO" : "YES");

	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	4067328	1039313	3
1	OK	0.015	2232320	12	2
2	OK	0.000	2228224	16	3
3	OK	0.000	2228224	112	3
4	OK	0.000	2232320	111	2
5	OK	0.015	2232320	112	3
6	OK	0.015	2228224	112	2
7	OK	0.000	2232320	109	3
8	OK	0.000	2228224	112	2
9	OK	0.000	2244608	110	3
10	OK	0.015	2232320	111	2
11	OK	0.000	2232320	108	3
12	OK	0.000	2240512	11674	3
13	OK	0.000	2244608	11707	2
14	OK	0.000	2240512	11712	3
15	OK	0.015	2252800	11754	2
16	OK	0.000	2244608	11708	3
17	OK	0.000	2244608	11740	2
18	OK	0.000	2244608	11726	3
19	OK	0.000	2260992	11680	2
20	OK	0.000	2256896	11741	3
21	OK	0.015	2244608	128736	3

22	OK	0.015	2236416	128832	2
23	OK	0.000	2248704	128751	3
24	OK	0.000	2228224	128866	2
25	OK	0.015	2232320	128700	3
26	OK	0.000	2228224	128707	2
27	OK	0.000	2240512	128729	3
28	OK	0.000	2236416	128807	2
29	OK	0.000	2228224	128784	3
30	OK	0.000	2228224	1039313	3
31	OK	0.015	4067328	1038610	2
32	OK	0.015	4067328	1038875	3
33	OK	0.031	3657728	1038723	2
34	OK	0.031	3661824	1038749	3
35	OK	0.031	3842048	1038747	2
36	OK	0.015	4067328	1039043	3
37	OK	0.015	3661824	1039210	2
38	OK	0.015	3661824	1038967	3