Лабораторная работа № 3

по дисциплине «Алгоритмы и структуры данных»

Openedu – неделя 3

Подготовил:

студент группы P3217

Бураков Илья Алексеевич

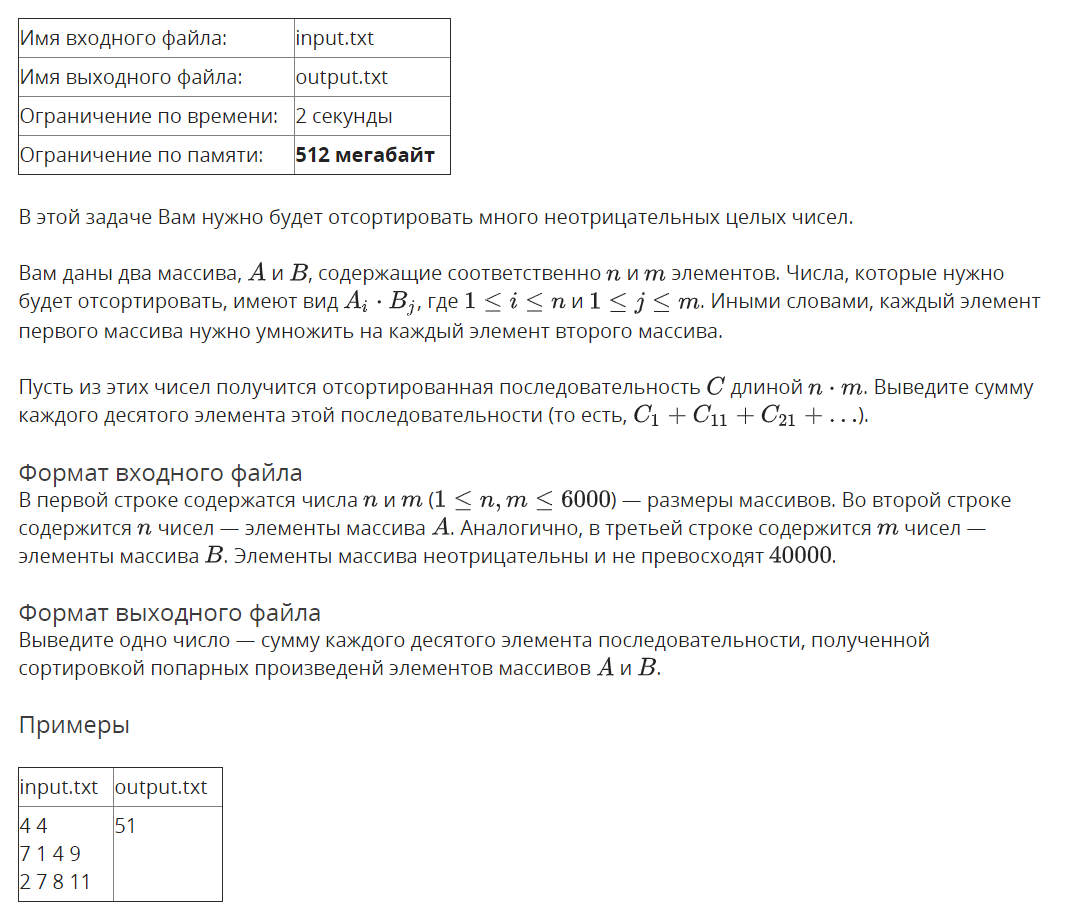
Преподаватели:

Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

## Сортировка целых чисел

## Условие



## Решение

openedu/week3/lab3\_1.cpp

#include "edx-io.hpp"

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

int n, m;

# define KTH\_BYTE(k, val) ((val) >> 8 \* k) & 0xFF

int main() {

// read array sizes

io >> n >> m;

// read arrays

auto a = vector<int>(n);

auto b = vector<int>(m);

for (auto &e : a) io >> e;

for (auto &e : b) io >> e;

// compose array to sort

// STL vectors result in TLE! how's that possible

auto c = new int[n \* m];

int max\_val = 0;

int next;

for (int i = 0; i < n; i++){

for (int j = 0; j < m; j++) {

next = a[i] \* b[j];

c[i\*m + j] = next;

if (next > max\_val) {

max\_val = next;

}

}

}

// i - radix sort phase

auto result = new int[n \* m];

auto counts = new int[256];

for (int byte\_n = 0; (1LL << (byte\_n \* 8)) <= max\_val; byte\_n++) {

memset(counts, 0, 256 \* sizeof(int));

// do element counting

for (int i = 0; i < n \* m; i++) {

counts[KTH\_BYTE(byte\_n, c[i])]++;

}

// calculate prepending elements count for each element

for (int i = 1; i < 256; i++) {

counts[i] += counts[i - 1];

}

// assembling sorted array

for (int i = n \* m - 1; i >= 0; i--) {

// calculate result position and set c[i] to it

result[--counts[KTH\_BYTE(byte\_n, c[i])]] = c[i];

}

memcpy(c, result, sizeof(int) \* n \* m);

}

// print order that we've got

long long sum = 0;

for (int i = 0; i < n \* m; i += 10) {

sum += c[i];

}

io << sum;

return 0;

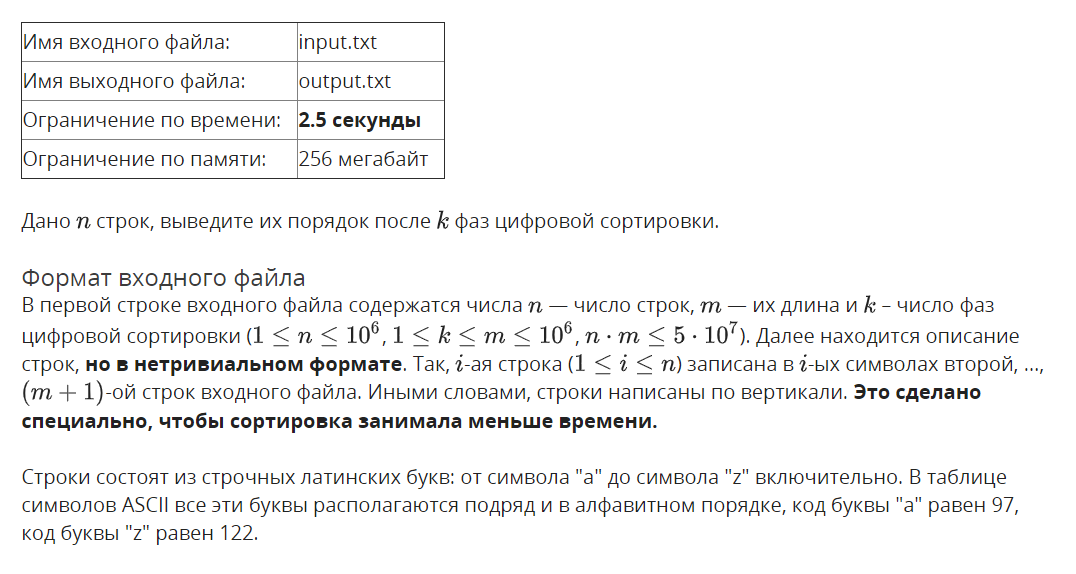
}

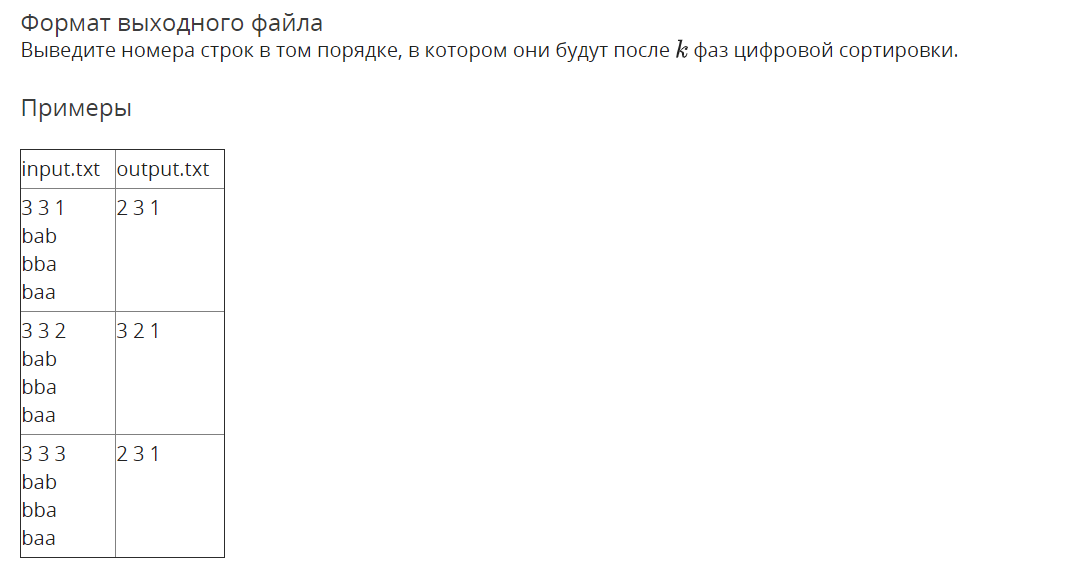
## Результаты

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 1.859 | 289931264 | 68699 | 16 |
| 1 | OK | 0.000 | 2224128 | 24 | 2 |
| 2 | OK | 0.000 | 2224128 | 34 | 1 |
| 3 | OK | 0.015 | 2224128 | 38 | 2 |
| 4 | OK | 0.000 | 2236416 | 106 | 10 |
| 5 | OK | 0.000 | 2224128 | 234 | 11 |
| 6 | OK | 0.000 | 2240512 | 698 | 11 |
| 7 | OK | 0.000 | 2228224 | 705 | 12 |
| 8 | OK | 0.000 | 2236416 | 586 | 12 |
| 9 | OK | 0.000 | 2220032 | 34325 | 12 |
| 10 | OK | 0.000 | 2285568 | 5769 | 12 |
| 11 | OK | 0.000 | 2285568 | 3498 | 12 |
| 12 | OK | 0.000 | 2220032 | 924 | 12 |
| 13 | OK | 0.000 | 2269184 | 3494 | 12 |
| 14 | OK | 0.000 | 2269184 | 5772 | 12 |
| 15 | OK | 0.000 | 2220032 | 34449 | 12 |
| 16 | OK | 0.000 | 2351104 | 34368 | 13 |
| 17 | OK | 0.000 | 2293760 | 4006 | 13 |
| 18 | OK | 0.015 | 2314240 | 2886 | 13 |
| 19 | OK | 0.015 | 2293760 | 4009 | 13 |
| 20 | OK | 0.000 | 2347008 | 34361 | 13 |
| 21 | OK | 0.031 | 6668288 | 34966 | 14 |
| 22 | OK | 0.031 | 6623232 | 9167 | 14 |
| 23 | OK | 0.031 | 6619136 | 9162 | 14 |
| 24 | OK | 0.031 | 6668288 | 34917 | 14 |
| 25 | OK | 0.296 | 49881088 | 39991 | 15 |
| 26 | OK | 0.296 | 51863552 | 28668 | 15 |
| 27 | OK | 0.296 | 49881088 | 40034 | 15 |
| 28 | OK | 0.906 | 145903616 | 51489 | 15 |
| 29 | OK | 0.875 | 145903616 | 51525 | 15 |
| 30 | OK | 1.859 | 289931264 | 68655 | 16 |
| 31 | OK | 1.718 | 289931264 | 68625 | 16 |
| 32 | OK | 1.718 | 289931264 | 68699 | 16 |

## Цифровая сортировка

## Условие





## Решение

openedu/week3/lab3\_2.cpp

#include "edx-io.hpp"

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

int n, m, k;

vector<pair<char, int>> counting\_sort(vector<pair<char, int>> &vec) {

// indices 97 - 122 (26 letters), shift = 97

auto counts = vector<int>(26, 0);

// do element counting

for (auto e : vec) {

counts[e.first - 'a']++;

}

// calculate prepending elements count for each element

for (int i = 1; i < counts.size(); i++) {

counts[i] += counts[i - 1];

}

// assembling sorted array

vector<pair<char, int>> result(vec.size());

for (int i = vec.size() - 1; i >= 0; i--) {

int result\_position = --counts[vec[i].first - 'a'];

result[result\_position] = vec[i];

}

return result;

}

int main() {

// read main input values

io >> n >> m >> k;

// read and store all m input strings in memory because they're in invalid order!

auto read\_chars = vector<string>(m);

for (auto &str: read\_chars) {

// read n chars - i-th chars of n strings

io >> str;

}

// will be sorted vector of pairs: strings' ith characters (i = 0..m-1) and their initial indices

auto sorted = vector<pair<char, int>>(n);

for (int i = 0; i < n; i++) {

// '-' is temporary: will be replaced by real char from input

sorted[i] = make\_pair('-', i);

}

// i - radix sort phase

for (int i = 0; i < k; i++) {

// put read chars in appropriate order to "sorted" vec

for (int j = 0; j < n; j++) {

sorted[j].first = read\_chars.back()[sorted[j].second];

}

read\_chars.pop\_back();

// stable counting sort by i-th chars

sorted = counting\_sort(sorted);

}

// print order that we've got

for (auto p : sorted) {

io << p.second + 1 << ' ';

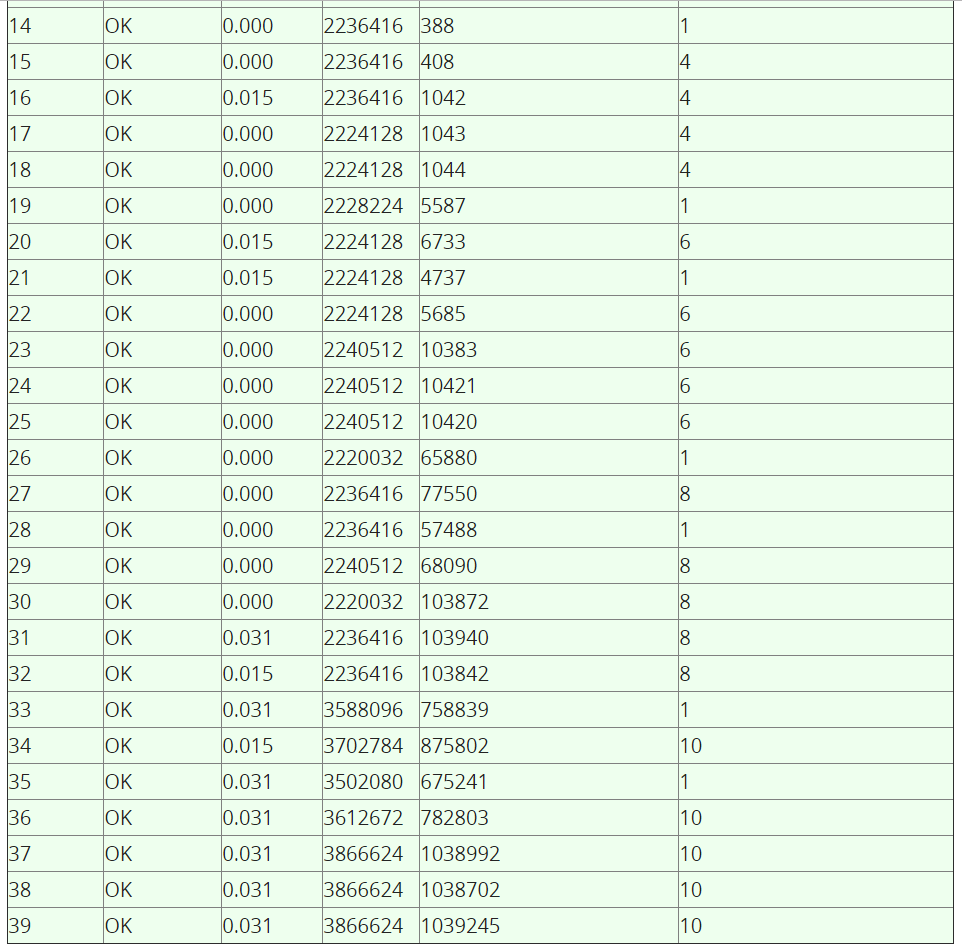
}

return 0;

}

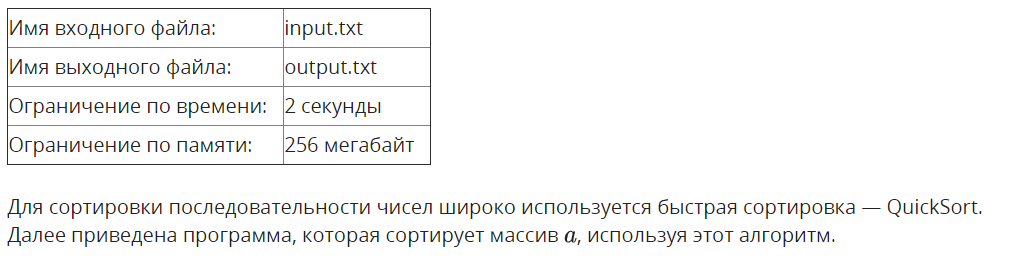
## Результаты





## Анти-quick sort

## Условие



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);

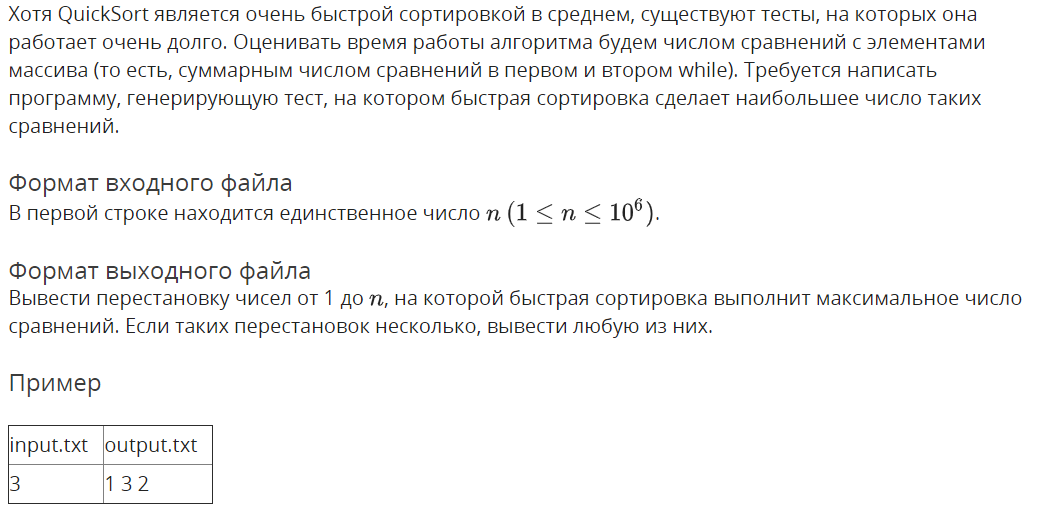
end;

begin

...

QSort(1, N);

end.



## 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) {

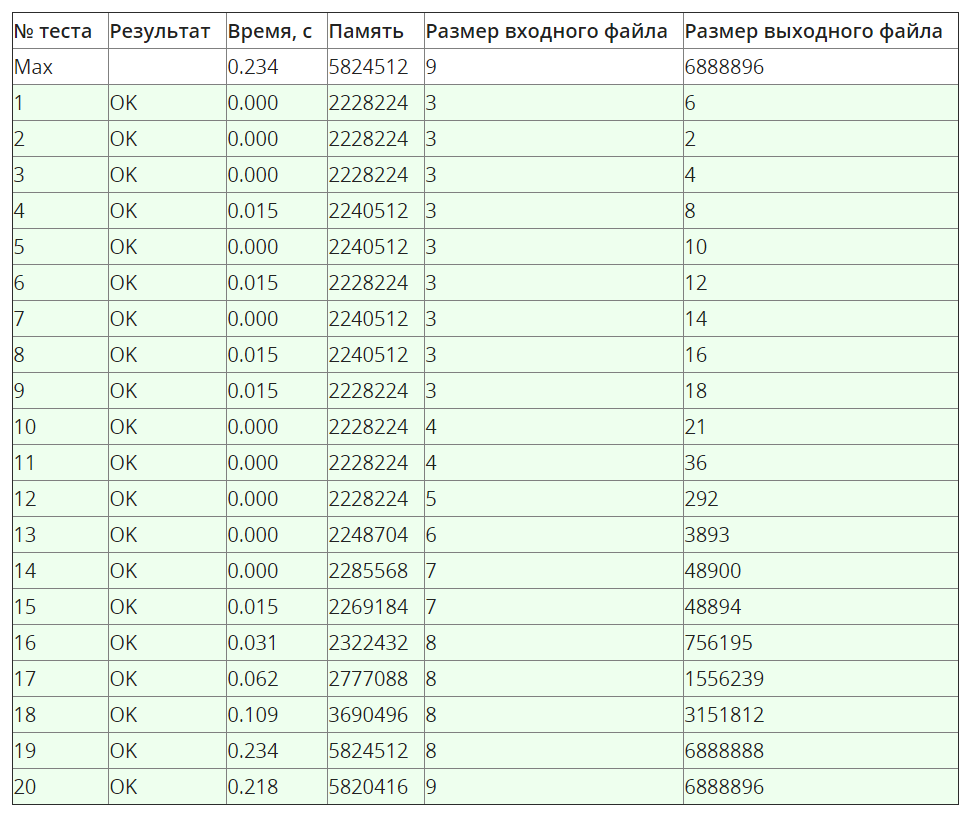
io << e << " ";

}

return 0;

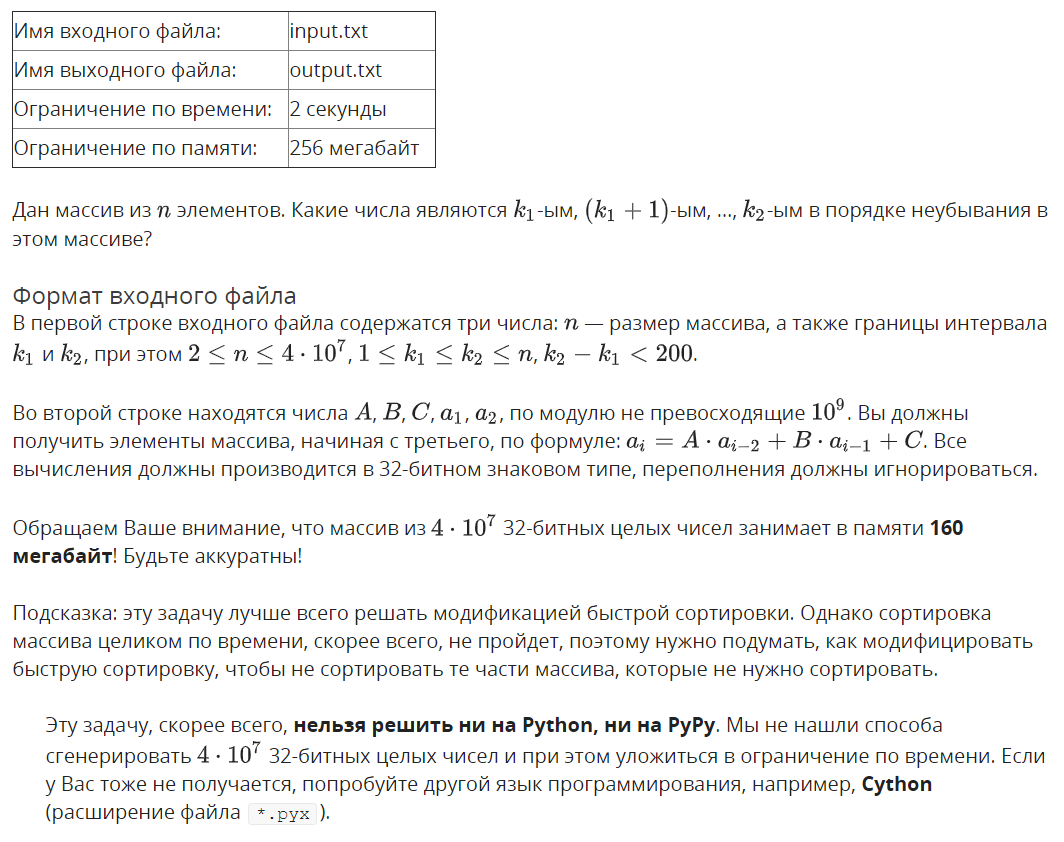
}

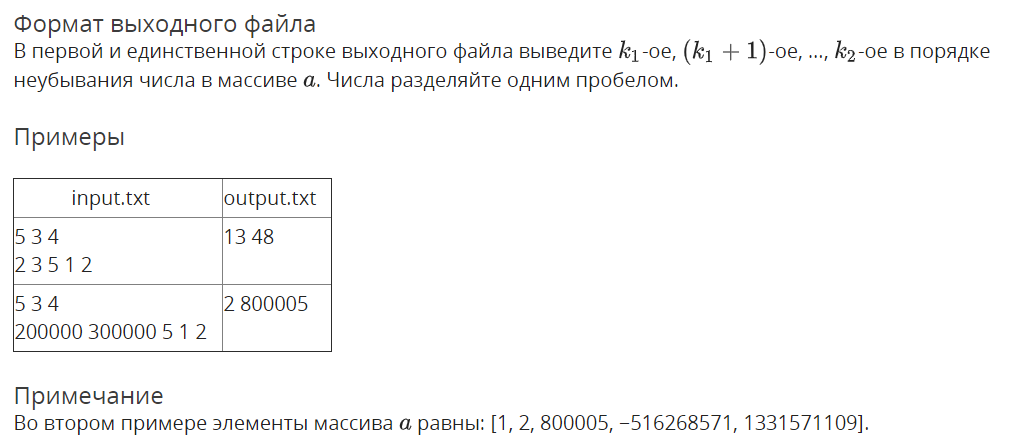
## Результаты



## K-ая порядковая статистика

## Условие





## 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++) {

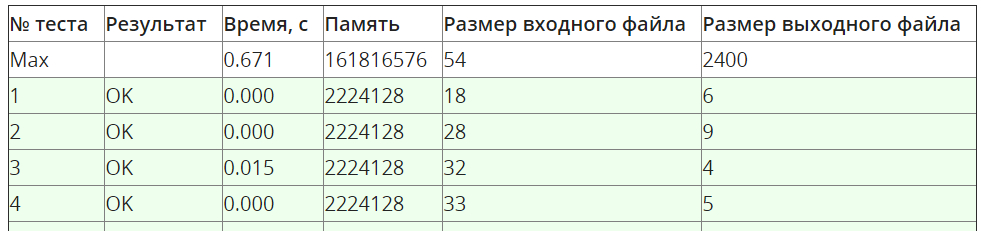
io << arr[i]<< " ";

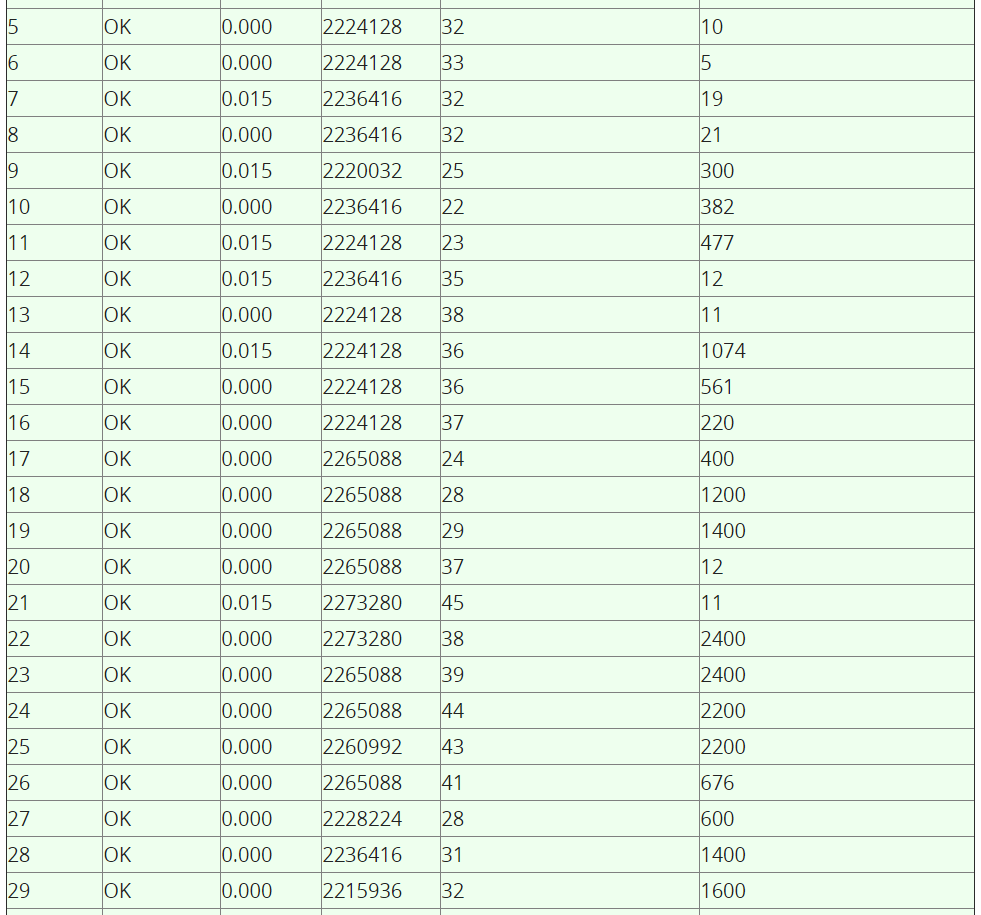
}

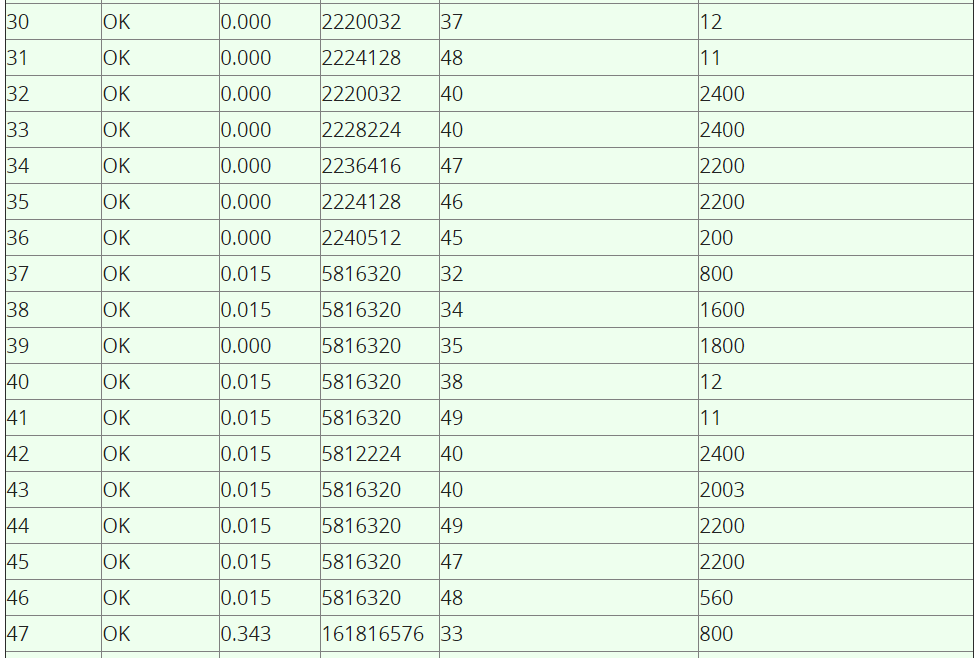
return 0;

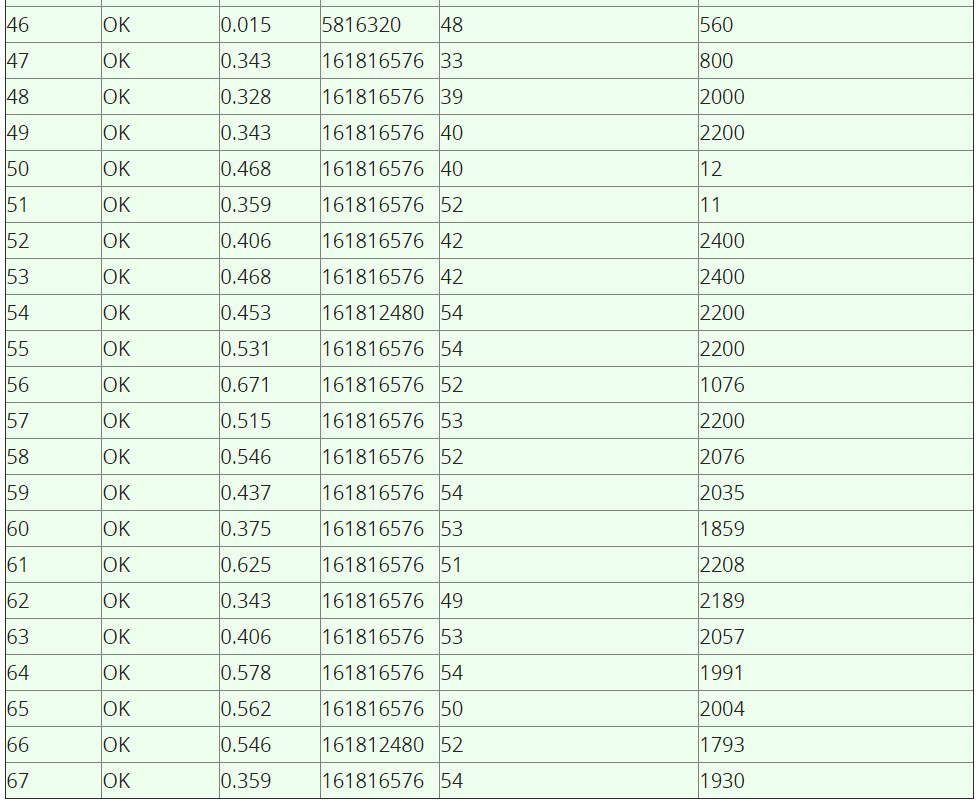
}

## Результаты



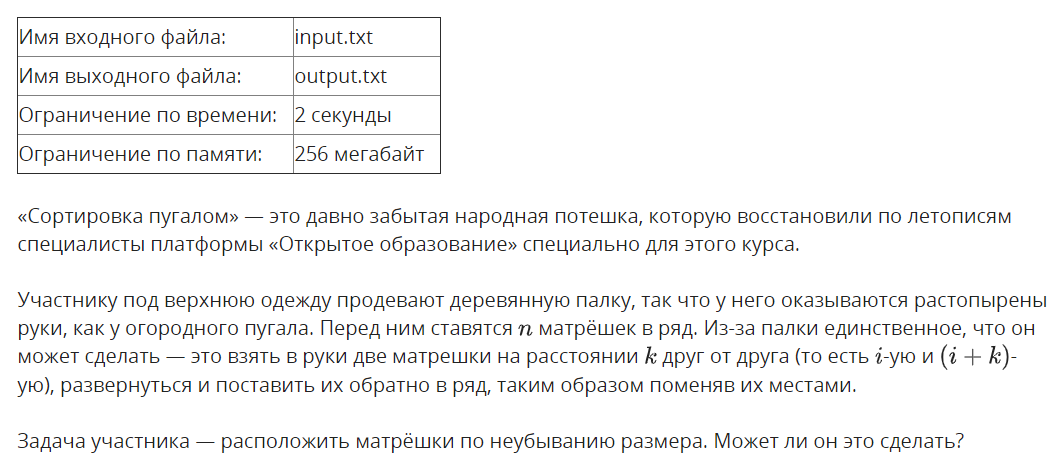


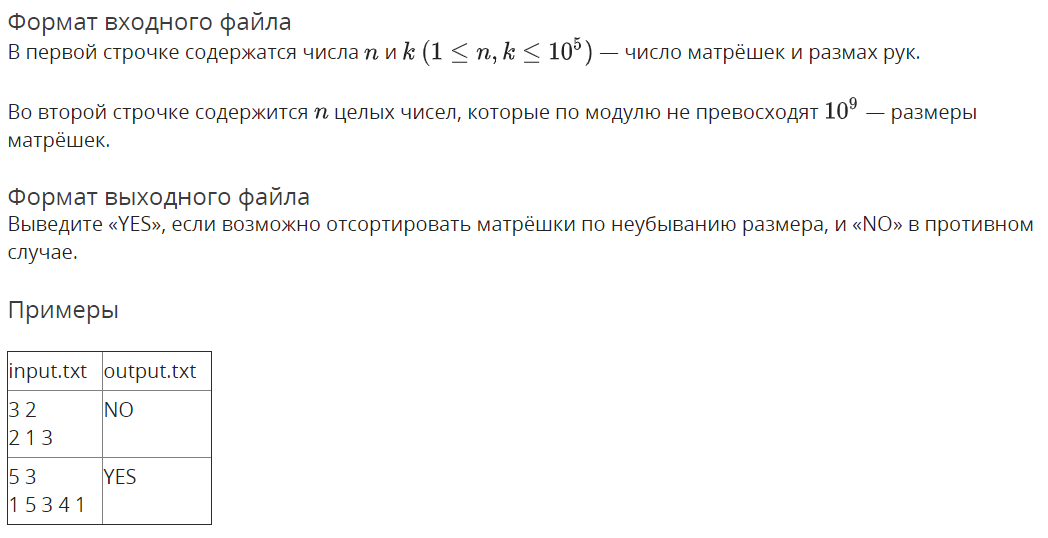




## Сортировка пугалом

## Условие





## 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

