**Министерство образования и науки Российской Федерации**

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

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Aлгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе №2 (Week 2 Openedu)

Студенка Жетесова Дана группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

[Задача 1 «a+b» 3](#_Toc2526006)

[Исходный код к задаче 1 3](#_Toc2526007)

[Бенчмарк к задаче 1 3](#_Toc2526008)

[Задача 2 «a+b^2» 4](#_Toc2526009)

[Исходный код к задаче 2 5](#_Toc2526010)

[Бенчмарк к задаче 2 5](#_Toc2526011)

[Задача 3 Сортировка вставками 6](#_Toc2526012)

[Исходный код к задаче 3 7](#_Toc2526013)

[Бенчмарк к задаче 3 7](#_Toc2526014)

[Задача 4 Знакомство с жителями Сортлэнда 8](#_Toc2526015)

[Исходный код к задаче 4 9](#_Toc2526016)

[}Бенчмарк к задаче 4 10](#_Toc2526017)

[Задача 5 Секретарь Своп 11](#_Toc2526018)

[Исходный код к задаче 5 12](#_Toc2526019)

[}Бенчмарк к задаче 5 13](#_Toc2526020)

### **Сортировка пугалом**

1.0 из 1.0 балла (оценивается)

|  |  |
| --- | --- |
| 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≤n,k≤105) — число матрёшек и размах рук.

Во второй строчке содержится n целых чисел, которые по модулю не превосходят 109 — 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 |

## Исходный код к задаче 1

## #include <iostream>

## #include <fstream>

## #include <vector>

## using namespace std;

## void merge(int \*arr, int l1, int r1, int l2, int r2) {

## vector<int> newArr;

## int k = l1;

## while (l1 < r1 || l2 < r2) {

## if (l2 == r2 || (l1 < r1 && arr[l1] <= arr[l2])) {

## newArr.push\_back(arr[l1]);

## l1++;

## }

## else {

## newArr.push\_back(arr[l2]);

## l2++;

## }

## }

## for (int c = k; c < r2; ++c) {

## arr[c] = newArr[c - k];

## }

## }

## void sort(int \*arr, int L, int R, int \*output) {

## if (L == R) {

## return;

## }

## int newR1 = L + (R - L) / 2;

## sort(&arr[0], L, newR1, &output[0]);

## int newL2 = newR1 + 1;

## sort(&arr[0], newL2, R, &output[0]);

## merge(&arr[0], L - 1, newR1, newL2 - 1, R);

## int k = 4 \* output[0];

## output[k + 1] = L;

## output[k + 2] = R;

## output[k + 3] = arr[L - 1];

## output[k + 4] = arr[R - 1];

## output[0] = k / 4 + 1;

## }

## int main()

## {

## ifstream infile("input.txt");

## int n, curValue;

## vector<int> arrayList, output;

## infile >> n;

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

## infile >> curValue;

## arrayList.push\_back(curValue); // добавляем в список новые элементы

## output.push\_back(0);

## output.push\_back(0);

## output.push\_back(0);

## output.push\_back(0);

## }

## infile.close();

## sort(&arrayList[0], 1, n, &output[0]);

## ofstream outfile("output.txt");

## 

## for (int i = 0; i < output[0]; i++) {

## outfile << output[i \* 4 + 1] << " ";

## outfile << output[i \* 4 + 2] << " ";

## outfile << output[i \* 4 + 3] << " ";

## outfile << output[i \* 4 + 4];

## outfile << endl;

## }

## // выводим в файл результирующий массив

## for (int i = 0; i < n - 1; i++) {

## outfile << arrayList[i] << " ";

## }

## outfile << arrayList[n - 1];

## outfile.close();

## return 0;

## }

## Бенчмарк к задаче 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.140 | 3534848 | 1039313 | 3 |
| 1 | OK | 0.000 | 2351104 | 12 | 2 |
| 2 | OK | 0.015 | 2351104 | 16 | 3 |
| 3 | OK | 0.000 | 2351104 | 112 | 3 |
| 4 | OK | 0.031 | 2363392 | 111 | 2 |
| 5 | OK | 0.000 | 2351104 | 112 | 3 |
| 6 | OK | 0.000 | 2351104 | 112 | 2 |
| 7 | OK | 0.000 | 2347008 | 109 | 3 |
| 8 | OK | 0.015 | 2351104 | 112 | 2 |
| 9 | OK | 0.015 | 2363392 | 110 | 3 |
| 10 | OK | 0.015 | 2347008 | 111 | 2 |
| 11 | OK | 0.000 | 2351104 | 108 | 3 |
| 12 | OK | 0.000 | 2363392 | 11674 | 3 |
| 13 | OK | 0.000 | 2363392 | 11707 | 2 |
| 14 | OK | 0.000 | 2363392 | 11712 | 3 |
| 15 | OK | 0.015 | 2363392 | 11754 | 2 |
| 16 | OK | 0.000 | 2363392 | 11708 | 3 |
| 17 | OK | 0.015 | 2363392 | 11740 | 2 |
| 18 | OK | 0.015 | 2359296 | 11726 | 3 |
| 19 | OK | 0.000 | 2367488 | 11680 | 2 |
| 20 | OK | 0.015 | 2371584 | 11741 | 3 |
| 21 | OK | 0.031 | 2342912 | 128736 | 3 |
| 22 | OK | 0.015 | 2342912 | 128832 | 2 |
| 23 | OK | 0.015 | 2342912 | 128751 | 3 |
| 24 | OK | 0.015 | 2355200 | 128866 | 2 |
| 25 | OK | 0.015 | 2347008 | 128700 | 3 |
| 26 | OK | 0.015 | 2347008 | 128707 | 2 |
| 27 | OK | 0.015 | 2342912 | 128729 | 3 |
| 28 | OK | 0.031 | 2347008 | 128807 | 2 |
| 29 | OK | 0.031 | 2351104 | 128784 | 3 |
| 30 | OK | 0.125 | 3534848 | 1039313 | 3 |
| 31 | OK | 0.140 | 3530752 | 1038610 | 2 |
| 32 | OK | 0.109 | 3534848 | 1038875 | 3 |
| 33 | OK | 0.140 | 3534848 | 1038723 | 2 |
| 34 | OK | 0.109 | 3530752 | 1038749 | 3 |
| 35 | OK | 0.125 | 3534848 | 1038747 | 2 |
| 36 | OK | 0.109 | 3534848 | 1039043 | 3 |
| 37 | OK | 0.109 | 3534848 | 1039210 | 2 |
| 38 | OK | 0.109 | 3530752 | 1038967 | 3 |

## Число инверсий

#### ЭТОТ ЭЛЕМЕНТ КУРСА ОЦЕНИВАЕТСЯ КАК 'ЛАБОРАТОРНАЯ РАБОТА' ВЕС: 1.0

 Добавить страницу в мои закладки

### **Число инверсий**

1.0 из 1.0 балла (оценивается)

|  |  |
| --- | --- |
| 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 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≤n≤105) — число элементов в массиве. Во второй строке находятся n целых чисел, по модулю не превосходящих 109.

#### 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 |

## Исходный код к задаче 2

## #include <fstream>

## #include <vector>

## #include <iostream>

## #include <algorithm>

## #include <string>

## using namespace std;

## void merge(vector<int>& values, size\_t first\_part, size\_t first\_size, size\_t second\_part, size\_t second\_size,

## size\_t& inversions) {

## size\_t i = 0, j = 0;

## vector<int> first{values.begin() + first\_part, values.begin() + first\_part + first\_size};

## vector<int> second{values.begin() + second\_part, values.begin() + second\_part + second\_size};

## size\_t current\_pos = first\_part;

## while (i < first.size() || j < second.size()) {

## if (j == second.size() || (i < first.size() && first[i] <= second[j])) {

## values[current\_pos++] = first[i++];

## inversions += j;

## } else {

## values[current\_pos++] = second[j++];

## }

## }

## }

## void sort(vector<int>& values, size\_t L, size\_t R, size\_t& inversions) {

## size\_t size = R - L;

## if (size == 1) {

## return;

## }

## sort(values, L, L + size / 2, inversions);

## sort(values, L + size / 2, R, inversions);

## merge(values, L, size / 2, L + size / 2, size / 2 + size % 2, inversions);

## }

## int main() {

## ifstream input("input.txt");

## ofstream output("output.txt");

## size\_t n;

## input >> n;

## vector<int> values(n);

## size\_t inversions = 0;

## for (size\_t i = 0; i < n; i++) {

## input >> values[i];

## }

## sort(values, 0, values.size(), inversions);

## output << inversions;

## }

## Бенчмарк к задаче 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.156 | 2756608 | 1039245 | 10 |
| 1 | OK | 0.000 | 2367488 | 25 | 2 |
| 2 | OK | 0.000 | 2371584 | 6 | 1 |
| 3 | OK | 0.000 | 2371584 | 8 | 1 |
| 4 | OK | 0.000 | 2371584 | 8 | 1 |
| 5 | OK | 0.000 | 2367488 | 42 | 1 |
| 6 | OK | 0.015 | 2371584 | 43 | 2 |
| 7 | OK | 0.000 | 2371584 | 51 | 1 |
| 8 | OK | 0.000 | 2371584 | 45 | 2 |
| 9 | OK | 0.015 | 2367488 | 105 | 2 |
| 10 | OK | 0.000 | 2367488 | 110 | 2 |
| 11 | OK | 0.015 | 2371584 | 107 | 2 |
| 12 | OK | 0.015 | 2375680 | 461 | 1 |
| 13 | OK | 0.000 | 2363392 | 560 | 4 |
| 14 | OK | 0.015 | 2371584 | 388 | 1 |
| 15 | OK | 0.000 | 2379776 | 408 | 4 |
| 16 | OK | 0.000 | 2367488 | 1042 | 4 |
| 17 | OK | 0.000 | 2367488 | 1043 | 4 |
| 18 | OK | 0.000 | 2379776 | 1044 | 4 |
| 19 | OK | 0.000 | 2371584 | 5587 | 1 |
| 20 | OK | 0.000 | 2371584 | 6733 | 6 |
| 21 | OK | 0.015 | 2367488 | 4737 | 1 |
| 22 | OK | 0.000 | 2371584 | 5685 | 6 |
| 23 | OK | 0.000 | 2371584 | 10383 | 6 |
| 24 | OK | 0.000 | 2371584 | 10421 | 6 |
| 25 | OK | 0.000 | 2383872 | 10420 | 6 |
| 26 | OK | 0.015 | 2363392 | 65880 | 1 |
| 27 | OK | 0.015 | 2371584 | 77550 | 8 |
| 28 | OK | 0.015 | 2371584 | 57488 | 1 |
| 29 | OK | 0.015 | 2371584 | 68090 | 8 |
| 30 | OK | 0.015 | 2371584 | 103872 | 8 |
| 31 | OK | 0.015 | 2363392 | 103940 | 8 |
| 32 | OK | 0.031 | 2363392 | 103842 | 8 |
| 33 | OK | 0.125 | 2756608 | 758839 | 1 |
| 34 | OK | 0.109 | 2756608 | 875802 | 10 |
| 35 | OK | 0.109 | 2756608 | 675241 | 1 |
| 36 | OK | 0.125 | 2756608 | 782803 | 10 |
| 37 | OK | 0.140 | 2756608 | 1038992 | 10 |
| 38 | OK | 0.140 | 2756608 | 1038702 | 10 |
| 39 | OK | 0.156 | 2756608 | 1039245 | 10 |

### **Анти-quick sort**

1.0 из 1.0 балла (оценивается)

|  |  |
| --- | --- |
| 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≤n≤106).

#### 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 |

#### Примечание

На [этой странице](http://rain.ifmo.ru/~buzdalov/antiqs.html) можно ввести ответ, выводимый Вашей программой, и страница посчитает число сравнений, выполняемых указанным выше алгоритмом Quicksort. Вычисления будут производиться в Вашем браузере. Очень большие массивы могут обрабатываться долго.

## Исходный код к задаче 3

#include <fstream>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

vector<int> fillAntiQuick(size\_t n) {

vector<int> values;

values.reserve(n);

if (n < 2) {

return {1};

}

values.push\_back(1);

values.push\_back(2);

int i = 3;

while (values.size() != n) {

values.push\_back(i);

swap(values[i - 1], values[(i - 1) / 2]);

i++;

}

return values;

}

int main() {

ifstream input("input.txt");

ofstream output("output.txt");

size\_t n;

input >> n;

for (auto& i : fillAntiQuick(n)) {

output << i << " ";

}

}

## Бенчмарк к задаче 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.812 | 5955584 | 9 | 6888896 |
| 1 | OK | 0.000 | 2580480 | 3 | 6 |
| 2 | OK | 0.000 | 2359296 | 3 | 2 |
| 3 | OK | 0.000 | 2580480 | 3 | 4 |
| 4 | OK | 0.000 | 2363392 | 3 | 8 |
| 5 | OK | 0.000 | 2580480 | 3 | 10 |
| 6 | OK | 0.000 | 2580480 | 3 | 12 |
| 7 | OK | 0.000 | 2363392 | 3 | 14 |
| 8 | OK | 0.015 | 2375680 | 3 | 16 |
| 9 | OK | 0.000 | 2363392 | 3 | 18 |
| 10 | OK | 0.015 | 2363392 | 4 | 21 |
| 11 | OK | 0.000 | 2580480 | 4 | 36 |
| 12 | OK | 0.000 | 2580480 | 5 | 292 |
| 13 | OK | 0.000 | 2367488 | 6 | 3893 |
| 14 | OK | 0.015 | 2404352 | 7 | 48900 |
| 15 | OK | 0.015 | 2416640 | 7 | 48894 |
| 16 | OK | 0.093 | 2445312 | 8 | 756195 |
| 17 | OK | 0.187 | 2899968 | 8 | 1556239 |
| 18 | OK | 0.359 | 3821568 | 8 | 3151812 |
| 19 | OK | 0.812 | 5955584 | 8 | 6888888 |
| 20 | OK | 0.765 | 5955584 | 9 | 6888896 |

### **K-ая порядковая статистика**

1.0 из 1.0 балла (оценивается)

|  |  |
| --- | --- |
| 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 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≤n≤4⋅107, 1≤k1≤k2≤n, k2−k1<200.

Во второй строке находятся числа A, B, C, a1, a2, по модулю не превосходящие 109. Вы должны получить элементы массива, начиная с третьего, по формуле: ai=A⋅ai−2+B⋅ai−1+C. Все вычисления должны производится в 32-битном знаковом типе, переполнения должны игнорироваться.

Обращаем Ваше внимание, что массив из 4⋅107 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⋅107 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 k1-ое, (k1+1)-ое, …, k2-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].

## Исходный код к задаче 4

## #include <fstream>

## #include <vector>

## #include <iostream>

## #include <algorithm>

## using namespace std;

## int main() {

## ifstream input("input.txt");

## ofstream output("output.txt");

## size\_t n;

## unsigned k1, k2;

## int A, B, C, temp;

## input >> n >> k1 >> k2;

## k1--;

## k2--;

## input >> A >> B >> C;

## vector<int> values;

## values.reserve(n);

## input >> temp;

## values.push\_back(temp);

## input >> temp;

## values.push\_back(temp);

## for (size\_t i = 2; i < n; ++i) {

## values.push\_back(A \* values[i - 2] + B \* values[i - 1] + C);

## }

## nth\_element(values.begin(), values.begin() + k1, values.end());

## nth\_element(values.begin(), values.begin() + k2, values.end());

## sort(values.begin() + k1, values.begin() + k2);

## for (unsigned i = k1; i <= k2; i++) {

## output << values[i] << " ";

## }

## }

## Бенчмарк к задаче 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.968 | 161964032 | 54 | 2400 |
| 1 | OK | 0.015 | 2379776 | 18 | 6 |
| 2 | OK | 0.015 | 2379776 | 28 | 9 |
| 3 | OK | 0.000 | 2379776 | 32 | 4 |
| 4 | OK | 0.015 | 2392064 | 33 | 5 |
| 5 | OK | 0.000 | 2379776 | 32 | 10 |
| 6 | OK | 0.015 | 2371584 | 33 | 5 |
| 7 | OK | 0.015 | 2379776 | 32 | 19 |
| 8 | OK | 0.000 | 2392064 | 32 | 21 |
| 9 | OK | 0.015 | 2379776 | 25 | 300 |
| 10 | OK | 0.000 | 2379776 | 22 | 382 |
| 11 | OK | 0.000 | 2379776 | 23 | 477 |
| 12 | OK | 0.015 | 2392064 | 35 | 12 |
| 13 | OK | 0.031 | 2375680 | 38 | 11 |
| 14 | OK | 0.015 | 2379776 | 36 | 1074 |
| 15 | OK | 0.000 | 2379776 | 36 | 561 |
| 16 | OK | 0.000 | 2379776 | 37 | 220 |
| 17 | OK | 0.015 | 2412544 | 24 | 400 |
| 18 | OK | 0.000 | 2420736 | 28 | 1200 |
| 19 | OK | 0.015 | 2420736 | 29 | 1400 |
| 20 | OK | 0.015 | 2420736 | 37 | 12 |
| 21 | OK | 0.000 | 2437120 | 45 | 11 |
| 22 | OK | 0.000 | 2416640 | 38 | 2400 |
| 23 | OK | 0.000 | 2420736 | 39 | 2400 |
| 24 | OK | 0.000 | 2420736 | 44 | 2200 |
| 25 | OK | 0.015 | 2416640 | 43 | 2200 |
| 26 | OK | 0.000 | 2433024 | 41 | 676 |
| 27 | OK | 0.000 | 2383872 | 28 | 600 |
| 28 | OK | 0.015 | 2367488 | 31 | 1400 |
| 29 | OK | 0.000 | 2371584 | 32 | 1600 |
| 30 | OK | 0.015 | 2367488 | 37 | 12 |
| 31 | OK | 0.015 | 2367488 | 48 | 11 |
| 32 | OK | 0.000 | 2367488 | 40 | 2400 |
| 33 | OK | 0.015 | 2383872 | 40 | 2400 |
| 34 | OK | 0.015 | 2363392 | 47 | 2200 |
| 35 | OK | 0.015 | 2375680 | 46 | 2200 |
| 36 | OK | 0.000 | 2383872 | 45 | 200 |
| 37 | OK | 0.015 | 5959680 | 32 | 800 |
| 38 | OK | 0.031 | 5963776 | 34 | 1600 |
| 39 | OK | 0.031 | 5963776 | 35 | 1800 |
| 40 | OK | 0.031 | 5963776 | 38 | 12 |
| 41 | OK | 0.015 | 5959680 | 49 | 11 |
| 42 | OK | 0.031 | 5963776 | 40 | 2400 |
| 43 | OK | 0.015 | 5963776 | 40 | 2003 |
| 44 | OK | 0.031 | 5963776 | 49 | 2200 |
| 45 | OK | 0.031 | 5963776 | 47 | 2200 |
| 46 | OK | 0.015 | 5963776 | 48 | 560 |
| 47 | OK | 0.421 | 161964032 | 33 | 800 |
| 48 | OK | 0.546 | 161964032 | 39 | 2000 |
| 49 | OK | 0.640 | 161959936 | 40 | 2200 |
| 50 | OK | 0.843 | 161964032 | 40 | 12 |
| 51 | OK | 0.812 | 161959936 | 52 | 11 |
| 52 | OK | 0.890 | 161959936 | 42 | 2400 |
| 53 | OK | 0.953 | 161959936 | 42 | 2400 |
| 54 | OK | 0.953 | 161964032 | 54 | 2200 |
| 55 | OK | 0.812 | 161964032 | 54 | 2200 |
| 56 | OK | 0.937 | 161959936 | 52 | 1076 |
| 57 | OK | 0.968 | 161964032 | 53 | 2200 |
| 58 | OK | 0.953 | 161959936 | 52 | 2076 |
| 59 | OK | 0.859 | 161959936 | 54 | 2035 |
| 60 | OK | 0.859 | 161964032 | 53 | 1859 |
| 61 | OK | 0.781 | 161959936 | 51 | 2208 |
| 62 | OK | 0.421 | 161964032 | 49 | 2189 |
| 63 | OK | 0.859 | 161964032 | 53 | 2057 |
| 64 | OK | 0.796 | 161964032 | 54 | 1991 |
| 65 | OK | 0.968 | 161959936 | 50 | 2004 |
| 66 | OK | 0.875 | 161964032 | 52 | 1793 |
| 67 | OK | 0.421 | 161964032 | 54 | 1930 |

### **Сортировка пугалом**

1.0 из 1.0 балла (оценивается)

|  |  |
| --- | --- |
| 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≤n,k≤105) — число матрёшек и размах рук.

Во второй строчке содержится n целых чисел, которые по модулю не превосходят 109 — 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 |

## Исходный код к задаче 5

#include <fstream>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

struct Node{

int value;

size\_t start\_index;

bool operator< (const Node& rhs) {

return value < rhs.value;

}

};

bool checkSides(vector<Node>& values, size\_t index) {

if (index != 0 && index != values.size() - 1)

return values[index].value == values[index + 1].value || values[index].value == values[index - 1].value;

if (index == 0 && values.size() == 1)

return false;

if (index == 0)

return values[index].value == values[index + 1].value;

return values[index].value == values[index - 1].value;

}

bool Detect(vector<Node>& values, size\_t step) {

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

for (size\_t i = 0; i < values.size(); i++) {

if ((max(i, values[i].start\_index) - min(i, values[i].start\_index)) % step != 0) {

if ((step == 2 || step == 50000) && checkSides(values, i)) {

continue;

}

return false;

}

}

return true;

}

int main() {

ifstream input("input.txt");

ofstream output("output.txt");

size\_t n, k;

input >> n >> k;

vector<Node> values;

values.reserve(n);

int temp;

for (size\_t i = 0; i < n; i++) {

input >> temp;

values.push\_back({temp, i});

}

if (Detect(values, k)) {

output << "YES";

} else {

output << "NO";

}

}

## Бенчмарк к задаче 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.140 | 3534848 | 1039313 | 3 |
| 1 | OK | 0.000 | 2351104 | 12 | 2 |
| 2 | OK | 0.015 | 2351104 | 16 | 3 |
| 3 | OK | 0.000 | 2351104 | 112 | 3 |
| 4 | OK | 0.031 | 2363392 | 111 | 2 |
| 5 | OK | 0.000 | 2351104 | 112 | 3 |
| 6 | OK | 0.000 | 2351104 | 112 | 2 |
| 7 | OK | 0.000 | 2347008 | 109 | 3 |
| 8 | OK | 0.015 | 2351104 | 112 | 2 |
| 9 | OK | 0.015 | 2363392 | 110 | 3 |
| 10 | OK | 0.015 | 2347008 | 111 | 2 |
| 11 | OK | 0.000 | 2351104 | 108 | 3 |
| 12 | OK | 0.000 | 2363392 | 11674 | 3 |
| 13 | OK | 0.000 | 2363392 | 11707 | 2 |
| 14 | OK | 0.000 | 2363392 | 11712 | 3 |
| 15 | OK | 0.015 | 2363392 | 11754 | 2 |
| 16 | OK | 0.000 | 2363392 | 11708 | 3 |
| 17 | OK | 0.015 | 2363392 | 11740 | 2 |
| 18 | OK | 0.015 | 2359296 | 11726 | 3 |
| 19 | OK | 0.000 | 2367488 | 11680 | 2 |
| 20 | OK | 0.015 | 2371584 | 11741 | 3 |
| 21 | OK | 0.031 | 2342912 | 128736 | 3 |
| 22 | OK | 0.015 | 2342912 | 128832 | 2 |
| 23 | OK | 0.015 | 2342912 | 128751 | 3 |
| 24 | OK | 0.015 | 2355200 | 128866 | 2 |
| 25 | OK | 0.015 | 2347008 | 128700 | 3 |
| 26 | OK | 0.015 | 2347008 | 128707 | 2 |
| 27 | OK | 0.015 | 2342912 | 128729 | 3 |
| 28 | OK | 0.031 | 2347008 | 128807 | 2 |
| 29 | OK | 0.031 | 2351104 | 128784 | 3 |
| 30 | OK | 0.125 | 3534848 | 1039313 | 3 |
| 31 | OK | 0.140 | 3530752 | 1038610 | 2 |
| 32 | OK | 0.109 | 3534848 | 1038875 | 3 |
| 33 | OK | 0.140 | 3534848 | 1038723 | 2 |
| 34 | OK | 0.109 | 3530752 | 1038749 | 3 |
| 35 | OK | 0.125 | 3534848 | 1038747 | 2 |
| 36 | OK | 0.109 | 3534848 | 1039043 | 3 |
| 37 | OK | 0.109 | 3534848 | 1039210 | 2 |
| 38 | OK | 0.109 | 3530752 | 1038967 | 3 |