**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 Кузенкова Елизавета группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

[Задача 1 Сортировка слиянием 3](#_Toc2644444)

[Исходный код к задаче 1 4](#_Toc2644445)

[Бенчмарк к задаче 1 5](#_Toc2644446)

[Задача 2 Число инверсий 6](#_Toc2644447)

[Исходный код к задаче 2 7](#_Toc2644448)

[Бенчмарк к задаче 2 8](#_Toc2644449)

[Задача 3 Анти-quick sort 9](#_Toc2644450)

[Исходный код к задаче 3 11](#_Toc2644451)

[Бенчмарк к задаче 3 11](#_Toc2644452)

[Задача 4 K-ая порядковая статистика 12](#_Toc2644453)

[Исходный код к задаче 4 13](#_Toc2644454)

[Бенчмарк к задаче 4 14](#_Toc2644455)

[Задача 5 Сортировка пугалом 16](#_Toc2644456)

[Исходный код к задаче 5 17](#_Toc2644457)

[Бенчмарк к задаче 5 19](#_Toc2644458)

# Задача 1 Сортировка слиянием

|  |  |
| --- | --- |
| 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  — число элементов в массиве. Во второй строке находятся целых чисел, по модулю не превосходящих.

#### 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:    , где  — индекс начала области слияния,  — индекс конца области слияния,  — значение первого элемента области слияния,  — значение последнего элемента области слияния.

Все индексы начинаются с единицы (то есть, ). **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.

#### 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 |

## Исходный код к задаче 1

class Lab2\_1

{

public void MergeSort<T>(T[] array, long startIndex, long endIndex, StreamWriter sw) where T : IComparable<T>

{

if (endIndex - startIndex == 0)

{

return;

}

else

{

long middleIndex = (endIndex + startIndex) / 2;

this.MergeSort(array, startIndex, middleIndex, sw);

this.MergeSort(array, middleIndex + 1, endIndex, sw);

this.Merge(array, startIndex, middleIndex, endIndex, sw);

}

}

public void Merge<T>(T[] array, long startIndex, long middleIndex, long endIndex, StreamWriter sw) where T : IComparable<T>

{

T[] result = new T[endIndex - startIndex + 1];

long li = 0, ri = 0;

while (li < middleIndex - startIndex + 1 && ri < endIndex - middleIndex)

{

if (array[startIndex + li].CompareTo(array[middleIndex + ri + 1]) <= 0)

{

result[ri + li] = array[startIndex + li++];

}

else

{

result[ri + li] = array[middleIndex + ++ri];

}

}

while (li < middleIndex - startIndex + 1)

{

result[ri + li] = array[startIndex + li++];

}

while (ri < endIndex - middleIndex)

{

result[ri + li] = array[middleIndex + ++ri];

}

for (int i = 0; i < result.Length; i++)

array[startIndex + i] = result[i];

sw.WriteLine("{0} {1} {2} {3}", startIndex + 1, endIndex + 1, array[startIndex], array[endIndex]);

}

private void DoWork(string[] args)

{

int[] array;

using (var file = new System.IO.StreamReader("input.txt"))

{

var count = int.Parse(file.ReadLine());

var input = file.ReadLine().Split(' ');

array = new int[count];

for (int i = 0; i < count; i++)

{

array[i] = int.Parse(input[i]);

}

}

using (var file = new System.IO.StreamWriter("output.txt"))

{

this.MergeSort(array, 0, array.Length - 1, file);

for (int i = 0; i < array.Length; i++)

{

file.Write($"{array[i]} ");

}

}

}

public static void Main(string[] args)

{

var app = new Lab2\_1();

app.DoWork(args);

}

}

## Бенчмарк к задаче 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.250 | 30621696 | 1039245 | 4403893 |
| 1 | OK | 0.046 | 11272192 | 25 | 104 |
| 2 | OK | 0.015 | 11153408 | 6 | 2 |
| 3 | OK | 0.031 | 11259904 | 8 | 13 |
| 4 | OK | 0.031 | 11182080 | 8 | 13 |
| 5 | OK | 0.031 | 11231232 | 42 | 154 |
| 6 | OK | 0.031 | 11206656 | 43 | 154 |
| 7 | OK | 0.031 | 11206656 | 51 | 178 |
| 8 | OK | 0.046 | 11227136 | 45 | 160 |
| 9 | OK | 0.031 | 11231232 | 105 | 331 |
| 10 | OK | 0.031 | 11264000 | 110 | 342 |
| 11 | OK | 0.031 | 11231232 | 107 | 335 |
| 12 | OK | 0.031 | 11268096 | 461 | 2040 |
| 13 | OK | 0.031 | 11325440 | 560 | 2331 |
| 14 | OK | 0.031 | 11300864 | 388 | 1820 |
| 15 | OK | 0.015 | 11276288 | 408 | 1878 |
| 16 | OK | 0.046 | 11309056 | 1042 | 3775 |
| 17 | OK | 0.015 | 11292672 | 1043 | 3785 |
| 18 | OK | 0.031 | 11321344 | 1044 | 3780 |
| 19 | OK | 0.031 | 11972608 | 5587 | 25511 |
| 20 | OK | 0.031 | 11960320 | 6733 | 28933 |
| 21 | OK | 0.031 | 11943936 | 4737 | 22957 |
| 22 | OK | 0.031 | 11988992 | 5685 | 25795 |
| 23 | OK | 0.031 | 12079104 | 10383 | 39969 |
| 24 | OK | 0.031 | 12099584 | 10421 | 40065 |
| 25 | OK | 0.031 | 12087296 | 10420 | 40048 |
| 26 | OK | 0.046 | 14028800 | 65880 | 305380 |
| 27 | OK | 0.062 | 14016512 | 77550 | 340369 |
| 28 | OK | 0.062 | 13963264 | 57488 | 280209 |
| 29 | OK | 0.046 | 13996032 | 68090 | 311996 |
| 30 | OK | 0.046 | 14049280 | 103872 | 420233 |
| 31 | OK | 0.062 | 14135296 | 103940 | 420400 |
| 32 | OK | 0.046 | 14061568 | 103842 | 420154 |
| 33 | OK | 0.218 | 30531584 | 758839 | 3554239 |
| 34 | OK | 0.234 | 30588928 | 875802 | 3905092 |
| 35 | OK | 0.234 | 29364224 | 675241 | 3303444 |
| 36 | OK | 0.234 | 30621696 | 782803 | 3626100 |
| 37 | OK | 0.250 | 30453760 | 1038992 | 4403365 |
| 38 | OK | 0.250 | 30400512 | 1038702 | 4402358 |
| 39 | OK | 0.250 | 30191616 | 1039245 | 4403893 |

# Задача 2 Число инверсий

|  |  |
| --- | --- |
| 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 , а .

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  — число элементов в массиве. Во второй строке находятся целых чисел, по модулю не превосходящих .

#### 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

class Lab2\_2

{

private long \_inversions = 0;

public void MergeSort<T>(T[] array, long start, long end) where T : IComparable<T>

{

if (end - start == 0)

{

return;

}

else

{

long middle = (end + start) / 2;

this.MergeSort(array, start, middle);

this.MergeSort(array, middle + 1, end);

this.Merge(array, start, middle, end);

}

}

public void Merge<T>(T[] array, long start, long middle, long end) where T : IComparable<T>

{

T[] result = new T[end - start + 1];

long li = 0, ri = 0;

while (li < middle - start + 1 && ri < end - middle)

{

if (array[start + li].CompareTo(array[middle + ri + 1]) <= 0)

{

result[ri + li] = array[start + li];

li++;

}

else

{

//bacause of subarrays sorted yet

\_inversions += middle - start - li + 1;

result[ri + li] = array[middle + ++ri];

}

}

while (li < middle - start + 1)

{

result[ri + li] = array[start + li];

li++;

}

while (ri < end - middle)

{

result[ri + li] = array[middle + ++ri];

//ri++;

}

for (int i = 0; i < result.Length; i++)

array[start + i] = result[i];

}

private void DoWork(string[] args)

{

int[] array;

using (var file = new System.IO.StreamReader("input.txt"))

{

var count = int.Parse(file.ReadLine());

var input = file.ReadLine().Split(' ');

array = new int[count];

for (int i = 0; i < count; i++)

{

array[i] = int.Parse(input[i]);

}

}

this.MergeSort(array, 0, array.Length - 1);

using (var file = new System.IO.StreamWriter("output.txt"))

{

file.Write(\_inversions);

}

}

public static void Main(string[] args)

{

var app = new Lab2\_2();

app.DoWork(args);

}

}

## Бенчмарк к задаче 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.109 | 27930624 | 1039245 | 10 |
| 1 | OK | 0.031 | 10330112 | 25 | 2 |
| 2 | OK | 0.015 | 10256384 | 6 | 1 |
| 3 | OK | 0.031 | 10280960 | 8 | 1 |
| 4 | OK | 0.015 | 10338304 | 8 | 1 |
| 5 | OK | 0.031 | 10321920 | 42 | 1 |
| 6 | OK | 0.031 | 10321920 | 43 | 2 |
| 7 | OK | 0.031 | 10326016 | 51 | 1 |
| 8 | OK | 0.031 | 10313728 | 45 | 2 |
| 9 | OK | 0.015 | 10366976 | 105 | 2 |
| 10 | OK | 0.015 | 10338304 | 110 | 2 |
| 11 | OK | 0.015 | 10362880 | 107 | 2 |
| 12 | OK | 0.031 | 10297344 | 461 | 1 |
| 13 | OK | 0.015 | 10420224 | 560 | 4 |
| 14 | OK | 0.015 | 10358784 | 388 | 1 |
| 15 | OK | 0.015 | 10297344 | 408 | 4 |
| 16 | OK | 0.031 | 10317824 | 1042 | 4 |
| 17 | OK | 0.015 | 10342400 | 1043 | 4 |
| 18 | OK | 0.031 | 10301440 | 1044 | 4 |
| 19 | OK | 0.046 | 10493952 | 5587 | 1 |
| 20 | OK | 0.031 | 10452992 | 6733 | 6 |
| 21 | OK | 0.031 | 10502144 | 4737 | 1 |
| 22 | OK | 0.031 | 10461184 | 5685 | 6 |
| 23 | OK | 0.015 | 10600448 | 10383 | 6 |
| 24 | OK | 0.031 | 10539008 | 10421 | 6 |
| 25 | OK | 0.015 | 10584064 | 10420 | 6 |
| 26 | OK | 0.031 | 11628544 | 65880 | 1 |
| 27 | OK | 0.031 | 11603968 | 77550 | 8 |
| 28 | OK | 0.031 | 11563008 | 57488 | 1 |
| 29 | OK | 0.031 | 11685888 | 68090 | 8 |
| 30 | OK | 0.031 | 11755520 | 103872 | 8 |
| 31 | OK | 0.031 | 11694080 | 103940 | 8 |
| 32 | OK | 0.031 | 11747328 | 103842 | 8 |
| 33 | OK | 0.062 | 27095040 | 758839 | 1 |
| 34 | OK | 0.078 | 27930624 | 875802 | 10 |
| 35 | OK | 0.062 | 25444352 | 675241 | 1 |
| 36 | OK | 0.078 | 27148288 | 782803 | 10 |
| 37 | OK | 0.093 | 27381760 | 1038992 | 10 |
| 38 | OK | 0.109 | 27385856 | 1038702 | 10 |
| 39 | OK | 0.093 | 27430912 | 1039245 | 10 |

# Задача 3 Анти-quick sort

|  |  |
| --- | --- |
| 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 .

#### 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

class Lab2\_3

{

private int[] MakeAntiQuickSortArray(int n)

{

int[] array = new int[n];

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

array[i] = i + 1;

for (int i = 2; i < n; i++)

{

int temp = array[i / 2];

array[i / 2] = array[i];

array[i] = temp;

}

return array;

}

private void DoWork(string[] args)

{

int n;

int[] array;

using (var file = new System.IO.StreamReader("input.txt"))

{

n = int.Parse(file.ReadLine());

}

array = this.MakeAntiQuickSortArray(n);

using (var file = new System.IO.StreamWriter("output.txt"))

{

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

{

file.Write($"{array[i]} ");

}

}

}

public static void Main(string[] args)

{

var app = new Lab2\_3();

app.DoWork(args);

}

}

## Бенчмарк к задаче 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.328 | 15360000 | 9 | 6888896 |
| 1 | OK | 0.015 | 10113024 | 3 | 6 |
| 2 | OK | 0.031 | 10153984 | 3 | 2 |
| 3 | OK | 0.031 | 10108928 | 3 | 4 |
| 4 | OK | 0.015 | 10108928 | 3 | 8 |
| 5 | OK | 0.015 | 10100736 | 3 | 10 |
| 6 | OK | 0.015 | 10141696 | 3 | 12 |
| 7 | OK | 0.031 | 10104832 | 3 | 14 |
| 8 | OK | 0.031 | 10149888 | 3 | 16 |
| 9 | OK | 0.031 | 10149888 | 3 | 18 |
| 10 | OK | 0.031 | 10129408 | 4 | 21 |
| 11 | OK | 0.015 | 10153984 | 4 | 36 |
| 12 | OK | 0.031 | 10129408 | 5 | 292 |
| 13 | OK | 0.031 | 10227712 | 6 | 3893 |
| 14 | OK | 0.015 | 11255808 | 7 | 48900 |
| 15 | OK | 0.031 | 11292672 | 7 | 48894 |
| 16 | OK | 0.062 | 11816960 | 8 | 756195 |
| 17 | OK | 0.093 | 12320768 | 8 | 1556239 |
| 18 | OK | 0.156 | 13176832 | 8 | 3151812 |
| 19 | OK | 0.312 | 15360000 | 8 | 6888888 |
| 20 | OK | 0.328 | 15360000 | 9 | 6888896 |

# Задача 4 K-ая порядковая статистика

|  |  |
| --- | --- |
| 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

Во второй строке находятся числа A, B, C, a1, a2, по модулю не превосходящие . Вы должны получить элементы массива, начиная с третьего, по формуле: . Все вычисления должны производиться в 32-битном знаковом типе, переполнения должны игнорироваться.

#### Формат выходного файла

В первой и единственной строке выходного файла выведите

-ое, (+1)-ое, …, -ое в порядке неубывания числа в массиве. 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 |

## Исходный код к задаче 4

class Lab2\_4

{

private int k1, k2;

private static void Swap(int[] array, int i, int j)

{

int tmp = array[i];

array[i] = array[j];

array[j] = tmp;

}

public void Quicksort(int[] elements, int left, int right)

{

while (true)

{

if (left > k2 || right < k1) 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) continue;

int tmp = elements[i];

elements[i] = elements[j];

elements[j] = tmp;

i++;

j--;

}

if (left < j)

{

this.Quicksort(elements, left, j);

}

if (i < right)

{

left = i;

this.Quicksort(elements, left, right);

}

break;

}

}

private void DoWork(string[] args)

{

int n, A, B, C;

int[] a;

using (var file = new System.IO.StreamReader("input.txt"))

{

var input = file.ReadLine().Split(' ').Select(int.Parse).ToArray();

n = input[0];

k1 = input[1] - 1;

k2 = input[2] - 1;

input = file.ReadLine().Split().Select(int.Parse).ToArray();

A = input[0];

B = input[1];

C = input[2];

a = new int[n];

a[0] = input[3];

a[1] = input[4];

}

for (var i = 2; i < n; i++)

{

a[i] = A \* a[i - 2] + B \* a[i - 1] + C;

}

this.Quicksort(a, 0, a.Length - 1);

using (var file = new System.IO.StreamWriter("output.txt"))

{

for (int i = k1; i <= k2; i++)

{

file.Write($"{a[i]} ");

}

}

}

public static void Main(string[] args)

{

var app = new Lab2\_4();

app.DoWork(args);

}

}

## Бенчмарк к задаче 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.828 | 171479040 | 54 | 2400 |
| 1 | OK | 0.031 | 11132928 | 18 | 6 |
| 2 | OK | 0.046 | 11149312 | 28 | 9 |
| 3 | OK | 0.031 | 11112448 | 32 | 4 |
| 4 | OK | 0.046 | 11096064 | 33 | 5 |
| 5 | OK | 0.031 | 11157504 | 32 | 10 |
| 6 | OK | 0.031 | 11112448 | 33 | 5 |
| 7 | OK | 0.031 | 11202560 | 32 | 19 |
| 8 | OK | 0.031 | 11128832 | 32 | 21 |
| 9 | OK | 0.031 | 11124736 | 25 | 300 |
| 10 | OK | 0.031 | 11128832 | 22 | 382 |
| 11 | OK | 0.015 | 11141120 | 23 | 477 |
| 12 | OK | 0.046 | 11141120 | 35 | 12 |
| 13 | OK | 0.031 | 11128832 | 38 | 11 |
| 14 | OK | 0.031 | 11177984 | 36 | 1074 |
| 15 | OK | 0.046 | 11169792 | 36 | 561 |
| 16 | OK | 0.046 | 11116544 | 37 | 220 |
| 17 | OK | 0.031 | 11194368 | 24 | 400 |
| 18 | OK | 0.031 | 11182080 | 28 | 1200 |
| 19 | OK | 0.031 | 11239424 | 29 | 1400 |
| 20 | OK | 0.031 | 11264000 | 37 | 12 |
| 21 | OK | 0.031 | 11194368 | 45 | 11 |
| 22 | OK | 0.031 | 11173888 | 38 | 2400 |
| 23 | OK | 0.046 | 11198464 | 39 | 2400 |
| 24 | OK | 0.031 | 11202560 | 44 | 2200 |
| 25 | OK | 0.031 | 11243520 | 43 | 2200 |
| 26 | OK | 0.031 | 11259904 | 41 | 676 |
| 27 | OK | 0.046 | 11554816 | 28 | 600 |
| 28 | OK | 0.031 | 11538432 | 31 | 1400 |
| 29 | OK | 0.046 | 11558912 | 32 | 1600 |
| 30 | OK | 0.015 | 11526144 | 37 | 12 |
| 31 | OK | 0.046 | 11526144 | 48 | 11 |
| 32 | OK | 0.031 | 11608064 | 40 | 2400 |
| 33 | OK | 0.031 | 11542528 | 40 | 2400 |
| 34 | OK | 0.031 | 11579392 | 47 | 2200 |
| 35 | OK | 0.031 | 11546624 | 46 | 2200 |
| 36 | OK | 0.031 | 11530240 | 45 | 200 |
| 37 | OK | 0.046 | 15179776 | 32 | 800 |
| 38 | OK | 0.031 | 15187968 | 34 | 1600 |
| 39 | OK | 0.046 | 15155200 | 35 | 1800 |
| 40 | OK | 0.031 | 15163392 | 38 | 12 |
| 41 | OK | 0.031 | 15142912 | 49 | 11 |
| 42 | OK | 0.046 | 15175680 | 40 | 2400 |
| 43 | OK | 0.031 | 15196160 | 40 | 2003 |
| 44 | OK | 0.031 | 15159296 | 49 | 2200 |
| 45 | OK | 0.046 | 15179776 | 47 | 2200 |
| 46 | OK | 0.031 | 15151104 | 48 | 560 |
| 47 | OK | 0.375 | 171429888 | 33 | 800 |
| 48 | OK | 0.390 | 171446272 | 39 | 2000 |
| 49 | OK | 0.437 | 171405312 | 40 | 2200 |
| 50 | OK | 0.578 | 171421696 | 40 | 12 |
| 51 | OK | 0.406 | 171417600 | 52 | 11 |
| 52 | OK | 0.515 | 171438080 | 42 | 2400 |
| 53 | OK | 0.546 | 171442176 | 42 | 2400 |
| 54 | OK | 0.546 | 171466752 | 54 | 2200 |
| 55 | OK | 0.640 | 171462656 | 54 | 2200 |
| 56 | OK | 0.828 | 171458560 | 52 | 1076 |
| 57 | OK | 0.593 | 171405312 | 53 | 2200 |
| 58 | OK | 0.656 | 171446272 | 52 | 2076 |
| 59 | OK | 0.515 | 171438080 | 54 | 2035 |
| 60 | OK | 0.468 | 171470848 | 53 | 1859 |
| 61 | OK | 0.750 | 171479040 | 51 | 2208 |
| 62 | OK | 0.390 | 171425792 | 49 | 2189 |
| 63 | OK | 0.500 | 171409408 | 53 | 2057 |
| 64 | OK | 0.718 | 171450368 | 54 | 1991 |
| 65 | OK | 0.640 | 171425792 | 50 | 2004 |
| 66 | OK | 0.671 | 171421696 | 52 | 1793 |
| 67 | OK | 0.390 | 171429888 | 54 | 1930 |

# Задача 5 Сортировка пугалом

|  |  |
| --- | --- |
| 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

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 ()— число матрёшек и размах рук.

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

— 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 |

верно

## Исходный код к задаче 5

class Lab2\_5

{

public static bool ScarecrowSort(long[] array, int minStep)

{

List<List<long>> arrays = new List<List<long>>();

for (int i = 0; i < minStep; i++)

{

arrays.Add(new List<long>());

for (int j = i; j < array.Length; j += minStep)

arrays[i].Add(array[j]);

MergeSort(arrays[i], 0, arrays[i].Count - 1);

}

array[0] = arrays[0][0];

for (int i = 1; i < array.Length; i++)

{

int arrayIndex = i % minStep;

int index = i / minStep;

if (array[i - 1] > arrays[arrayIndex][index])

return false;

else

array[i] = arrays[arrayIndex][index];

}

return true;

}

public static void MergeSort(List<long> array, int startIndex, int endIndex)

{

if (endIndex - startIndex == 0) return;

int middleIndex = (endIndex + startIndex) / 2;

MergeSort(array, startIndex, middleIndex);

MergeSort(array, middleIndex + 1, endIndex);

Merge(array, startIndex, middleIndex, endIndex);

}

public static void Merge(List<long> array, int startIndex, int middleIndex, int endIndex)

{

long[] result = new long[endIndex - startIndex + 1];

int li = 0, ri = 0;

while (li < middleIndex - startIndex + 1 && ri < endIndex - middleIndex)

if (array[startIndex + li].CompareTo(array[middleIndex + ri + 1]) <= 0)

result[ri + li] = array[startIndex + li++];

else

result[ri + li] = array[middleIndex + ++ri];

while (li < middleIndex - startIndex + 1)

result[ri + li] = array[startIndex + li++];

while (ri < endIndex - middleIndex)

result[ri + li] = array[middleIndex + ++ri];

for (int i = 0; i < result.Length; i++)

array[startIndex + i] = result[i];

}

private void DoWork(string[] args)

{

bool canBeSort;

using (var file = new System.IO.StreamReader("input.txt"))

{

var arg = file.ReadLine().Split(' ').Select(x => int.Parse(x)).ToArray();

var inputArray = file.ReadLine().Split(' ').Select(x => long.Parse(x)).ToArray();

canBeSort = true;

if (arg[1] != 1)

canBeSort = ScarecrowSort(inputArray, arg[1]);

}

using (var file = new System.IO.StreamWriter("output.txt"))

{

if (canBeSort)

file.WriteLine("YES");

else

file.WriteLine("NO");

}

}

public static void Main(string[] args)

{

var app = new Lab2\_5();

app.DoWork(args);

}

}

## Бенчмарк к задаче 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.125 | 35991552 | 1039313 | 5 |
| 1 | OK | 0.046 | 11653120 | 12 | 4 |
| 2 | OK | 0.031 | 11608064 | 16 | 5 |
| 3 | OK | 0.031 | 11329536 | 112 | 5 |
| 4 | OK | 0.031 | 11419648 | 111 | 4 |
| 5 | OK | 0.031 | 11448320 | 112 | 5 |
| 6 | OK | 0.031 | 11403264 | 112 | 4 |
| 7 | OK | 0.031 | 11476992 | 109 | 5 |
| 8 | OK | 0.031 | 11440128 | 112 | 4 |
| 9 | OK | 0.031 | 11399168 | 110 | 5 |
| 10 | OK | 0.031 | 11419648 | 111 | 4 |
| 11 | OK | 0.046 | 11448320 | 108 | 5 |
| 12 | OK | 0.046 | 11337728 | 11674 | 5 |
| 13 | OK | 0.031 | 11788288 | 11707 | 4 |
| 14 | OK | 0.031 | 11780096 | 11712 | 5 |
| 15 | OK | 0.031 | 11837440 | 11754 | 4 |
| 16 | OK | 0.031 | 11767808 | 11708 | 5 |
| 17 | OK | 0.031 | 11837440 | 11740 | 4 |
| 18 | OK | 0.031 | 11841536 | 11726 | 5 |
| 19 | OK | 0.031 | 11722752 | 11680 | 4 |
| 20 | OK | 0.031 | 11792384 | 11741 | 5 |
| 21 | OK | 0.046 | 12742656 | 128736 | 5 |
| 22 | OK | 0.046 | 15106048 | 128832 | 4 |
| 23 | OK | 0.031 | 15085568 | 128751 | 5 |
| 24 | OK | 0.046 | 14696448 | 128866 | 4 |
| 25 | OK | 0.031 | 14651392 | 128700 | 5 |
| 26 | OK | 0.031 | 15032320 | 128707 | 4 |
| 27 | OK | 0.046 | 14970880 | 128729 | 5 |
| 28 | OK | 0.031 | 14221312 | 128807 | 4 |
| 29 | OK | 0.046 | 14233600 | 128784 | 5 |
| 30 | OK | 0.062 | 23154688 | 1039313 | 5 |
| 31 | OK | 0.109 | 33030144 | 1038610 | 4 |
| 32 | OK | 0.109 | 33079296 | 1038875 | 5 |
| 33 | OK | 0.093 | 35987456 | 1038723 | 4 |
| 34 | OK | 0.093 | 35991552 | 1038749 | 5 |
| 35 | OK | 0.109 | 31129600 | 1038747 | 4 |
| 36 | OK | 0.109 | 33021952 | 1039043 | 5 |
| 37 | OK | 0.093 | 32067584 | 1039210 | 4 |
| 38 | OK | 0.125 | 32047104 | 1038967 | 5 |