**Министерство образования и науки Российской Федерации**

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

Факультет программной инженерии и компьютерной техники

Кафедра информатики и прикладной математики   
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Алгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе №1

Студент Кочарян С.А. группы P3218

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

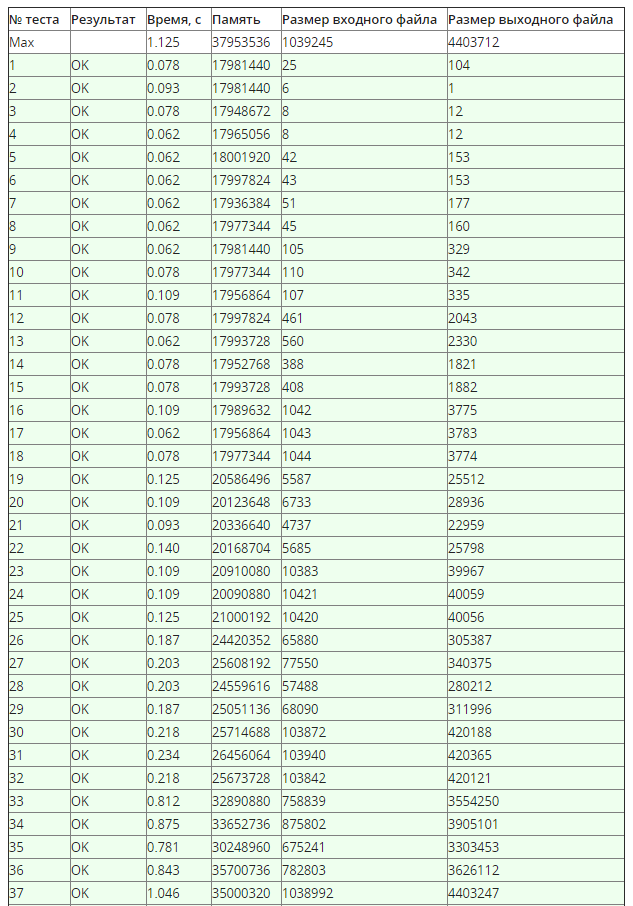
2019

## Сортировка слиянием

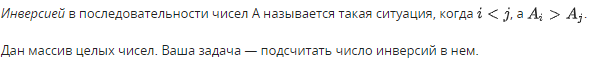
Дан массив целых чисел. Ваша задача — отсортировать его в порядке неубывания с помощью сортировки слиянием.

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.

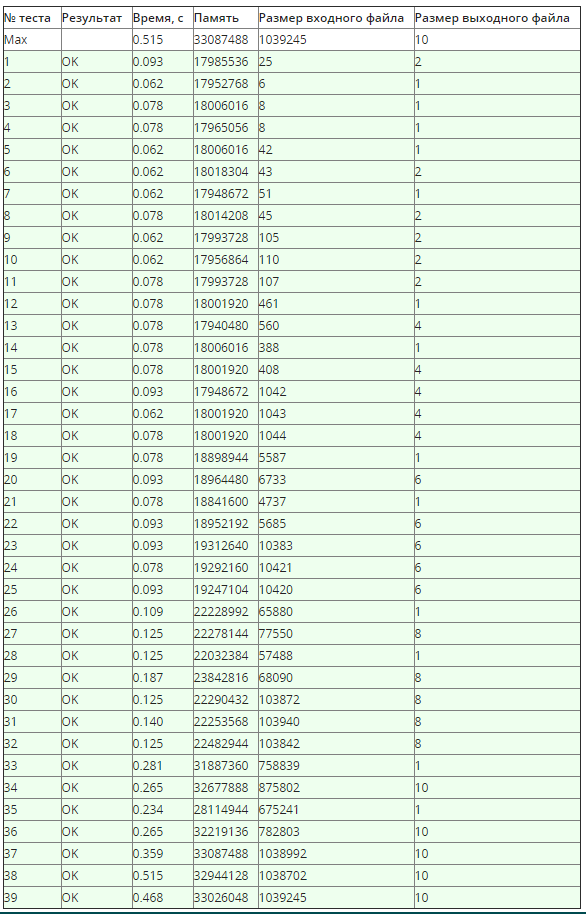
def mergeSort(array, leftBorder, rightBorder, file):  
 if len(array) > 1:  
 middle = len(array) // 2  
 leftPart = array[:middle]  
 rightPart = array[middle:]  
 mergeSort(leftPart, leftBorder, leftBorder + middle - 1, file)  
 mergeSort(rightPart, leftBorder + middle, rightBorder, file)  
 leftIndex = 0  
 rightIndex = 0  
 position = 0  
 while (leftIndex < len(leftPart) and rightIndex < len(rightPart)):  
 if (leftPart[leftIndex] < rightPart[rightIndex]):  
 array[position] = leftPart[leftIndex]  
 leftIndex += 1  
 else:  
 array[position] = rightPart[rightIndex]  
 rightIndex += 1  
 position += 1  
 while (leftIndex < len(leftPart)):  
 array[position] = leftPart[leftIndex]  
 position += 1  
 leftIndex += 1  
 while (rightIndex < len(rightPart)):  
 array[position] = rightPart[rightIndex]  
 position += 1  
 rightIndex += 1  
 file.write(f"{leftBorder + 1} {rightBorder + 1} {array[0]} {array[-1]}\n")  
  
  
  
file = open("input.txt", "r")  
quantity = int(file.readline())  
numbers = list(map(int, file.read().split()))  
file = open("output.txt", "w")  
mergeSort(numbers, 0, quantity - 1, file)  
file.write(' '.join(map(str, numbers)))



## Число инверсий



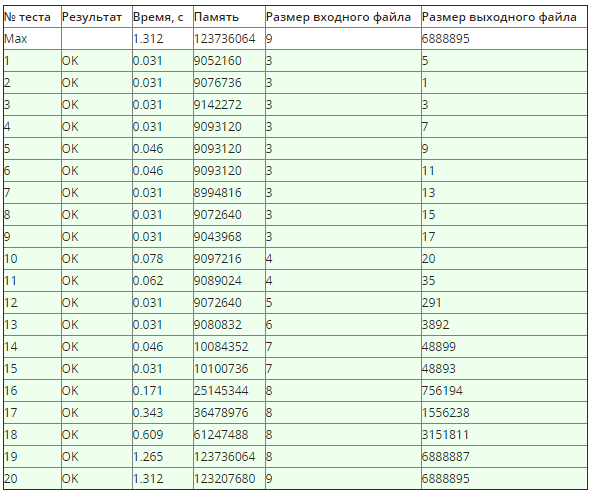
def mergeSort(array):  
 if len(array) > 1:  
 middle = len(array) // 2  
 leftPart = array[:middle]  
 rightPart = array[middle:]  
 inversion = mergeSort(leftPart) + mergeSort(rightPart)  
 leftIndex = 0  
 rightIndex = 0  
 position = 0  
 leftLen = len(leftPart)  
 rightLen = len(rightPart)  
 while leftIndex < leftLen and rightIndex < rightLen:  
 if leftPart[leftIndex] <= rightPart[rightIndex]:  
 array[position] = leftPart[leftIndex]  
 leftIndex += 1  
 else:  
 inversion += leftLen - leftIndex  
 array[position] = rightPart[rightIndex]  
 rightIndex += 1  
 position += 1  
 while leftIndex < leftLen:  
 array[position] = leftPart[leftIndex]  
 position += 1  
 leftIndex += 1  
 while rightIndex < rightLen:  
 array[position] = rightPart[rightIndex]  
 position += 1  
 rightIndex += 1  
 return inversion  
 return 0  
  
  
file = open("input.txt", "r")  
quantity = int(file.readline())  
numbers = list(map(int, file.read().split()))  
file = open("output.txt", "w")  
file.write(f"{mergeSort(numbers)}")



## Анти-quick sort

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.

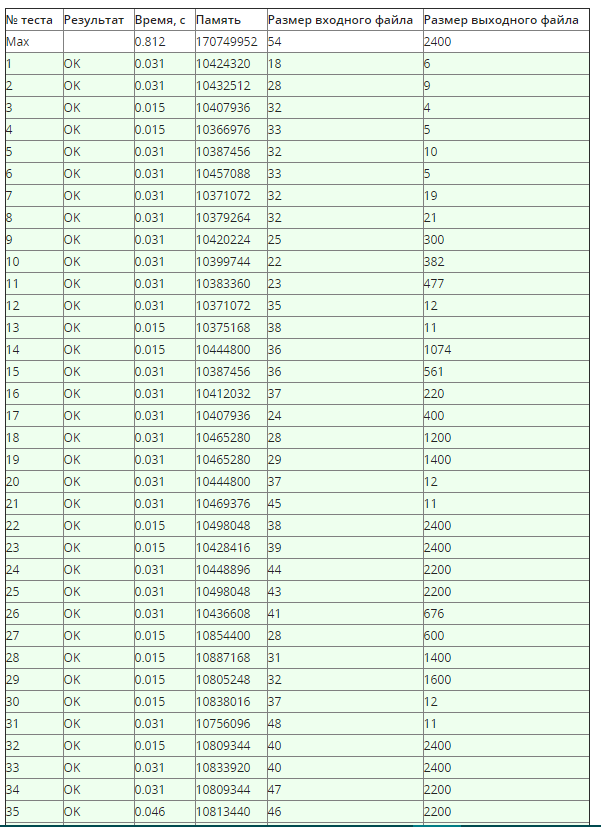
quantity = int(open("input.txt", "r").readline())  
array = [0 for i in range(quantity)]  
for i in range(quantity):  
 array[i] = i + 1  
 if (i > 1):  
 array[i], array[i // 2] = array[i // 2], array[i]  
open("output.txt", "w").write(' '.join(map(str, array)))



## K-ая порядковая статистика



using System;  
using System.Collections.Generic;  
using System.IO;  
using System.Linq;  
using System.Text;  
  
namespace \_4  
{  
 class Program  
 {  
 public static void Quicksort(int[] elements, int left, int right, int k1, int k2)  
 {  
 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)  
 {  
 int tmp = elements[i];  
 elements[i] = elements[j];  
 elements[j] = tmp;  
  
 i++;  
 j--;  
 }  
 }  
  
 if (left < j)  
 {  
 Quicksort(elements, left, j, k1, k2);  
 }  
  
 if (i < right)  
 {  
 Quicksort(elements, i, right, k1, k2);  
 }  
 }  
  
  
 static void Main(string[] args)  
 {  
 int n, k1, k2;  
 int A, B, C;  
 int[] a;  
 using (StreamReader sr = new StreamReader("input.txt"))  
 {  
 string[] FirstLine = sr.ReadLine().Split();  
 n = int.Parse(FirstLine[0]);  
 k1 = int.Parse(FirstLine[1]) - 1;  
 k2 = int.Parse(FirstLine[2]) - 1;  
 string[] SecondLine = sr.ReadLine().Split();  
 A = int.Parse(SecondLine[0]);  
 B = int.Parse(SecondLine[1]);  
 C = int.Parse(SecondLine[2]);  
 a = new int[n];  
 a[0] = (int.Parse(SecondLine[3]));  
 a[1] = (int.Parse(SecondLine[4]));  
 }  
 for (int i = 2; i < n; i++)  
 {  
 a[i] = (A \* a[i - 2] + B \* a[i - 1] + C);  
 }  
  
  
 using (StreamWriter sw = new StreamWriter("output.txt"))  
 {  
 Quicksort(a, 0, n - 1, k1, k2);  
 for (int i = k1; i <= k2; i++)  
 {  
 sw.Write($"{a[i]} ");  
 }  
 }  
 }  
 }  
}



## 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?

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)

