Санкт-Петербургский национальный исследовательский университет

Информационных технологий, механики и оптики

Отчет по решению задач третей недели   
По курсу «Алгоритмы и структуры данных»

на Openedu

Выполнил: Сыроватский Павел Валентинович

Группа P3218

Санкт-Петербург

2019

**Task 1**

**Сортировка целых чисел**

2.0 из 2.0 баллов (оценивается)

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | **512 мегабайт** |

В этой задаче Вам нужно будет отсортировать много неотрицательных целых чисел.

Вам даны два массива, A и B, содержащие соответственно n и m элементов. Числа, которые нужно будет отсортировать, имеют вид Ai⋅Bj, где 1≤i≤n и 1≤j≤m. Иными словами, каждый элемент первого массива нужно умножить на каждый элемент второго массива.

Пусть из этих чисел получится отсортированная последовательность C длиной n⋅m. Выведите сумму каждого десятого элемента этой последовательности (то есть, C1+C11+C21+…).

**Формат входного файла**

В первой строке содержатся числа n и m (1≤n,m≤6000) — размеры массивов. Во второй строке содержится n чисел — элементы массива A. Аналогично, в третьей строке содержится m чисел — элементы массива B. Элементы массива неотрицательны и не превосходят 40000.

**Формат выходного файла**

Выведите одно число — сумму каждого десятого элемента последовательности, полученной сортировкой попарных произведенй элементов массивов A и B.

**Примеры**

|  |  |
| --- | --- |
| input.txt | output.txt |
| 4 4 7 1 4 9 2 7 8 11 | 51 |

**Пояснение к примеру**

Неотсортированная последовательность C выглядит следующим образом:

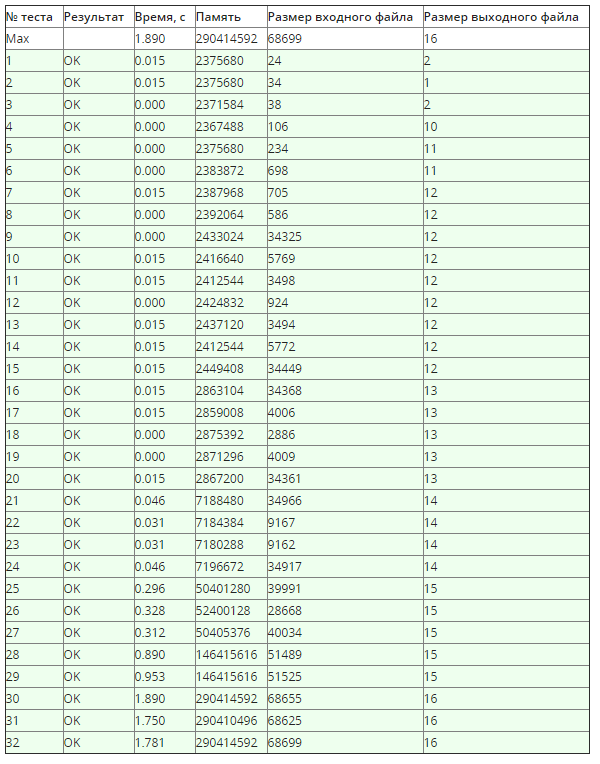
[14, 2, 8, 18, 49, 7, 28, 63, 56, 8, 32, 72, 77, 11, 44, 99].

Отсортировав ее, получим:

[**2**, 7, 8, 8, 11, 14, 18, 28, 32, 44, **49**, 56, 63, 72, 77, 99].

Жирным выделены первый и одиннадцатый элементы последовательности, при этом двадцать первого элемента в ней нет. Их сумма — 51 — и будет ответом.

Результат выполнения первой задачи



#include <fstream>

using namespace std;

int main() {

//Объявляем переменные

int n, m;

long pos;

ifstream input("input.txt");

input >> n >> m;

long \*A = new long[n];

long \*B = new long[n];

long \*C = new long[m \* n];

//Заполняем массивы и находим максимум

long maxA = 0;

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

input >> A[i];

//Находим максимальный элемент в первом массиве

if (A[i] > maxA) {

maxA = A[i];

}

}

//Заполняем второй массив и сразу вычесляем результирующий

long maxB = 0;

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

input >> B[i];

//Заполнсяем массив С

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

pos = (i \* n) + j;

C[pos] = B[i] \* A[j];

}

//Находим максимальный элемент во втором массиве

if (B[i] > maxB) {

maxB = B[i];

}

}

input.close();

//Подхотовка к сортировке

long maxNum = maxA \* maxB;

int digit = 1;

while (maxNum >> digit > 0) {

digit++;

}

digit--;

//Цифровая сортировка

int size = n \* m;

long \*out = new long[size];

long \*count = new long[256];

//

for (int Byte = 0; Byte <= digit; Byte += 8) {

for (int i = 0; i < 256; i++) {

count[i] = 0;

}

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

count[(C[i] >> Byte) & 255]++;

}

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

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

}

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

out[count[(C[i] >> Byte) & 255] - 1] = C[i];

count[(C[i] >> Byte) & 255]--;

}

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

C[i] = out[i];

}

}

//Вывод каждого десятого

long long sum = 0;

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

sum += C[i];

}

ofstream output("output.txt");

output << sum;

output.close();

return 0;

}

**Цифровая сортировка**

**ЭТОТ ЭЛЕМЕНТ КУРСА ОЦЕНИВАЕТСЯ КАК 'ЛАБОРАТОРНАЯ РАБОТА'  
ВЕС: 3.0**

 Добавить страницу в мои закладки

Цифровая сортировка

3.0 из 3.0 баллов (оценивается)

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | **2.5 секунды** |
| Ограничение по памяти: | 256 мегабайт |

Дано n строк, выведите их порядок после k фаз цифровой сортировки.

**Формат входного файла**

В первой строке входного файла содержатся числа n — число строк, m — их длина и k – число фаз цифровой сортировки (1≤n≤106, 1≤k≤m≤106, n⋅m≤5⋅107). Далее находится описание строк, **но в нетривиальном формате**. Так, i-ая строка (1≤i≤n) записана в i-ых символах второй, …, (m+1)-ой строк входного файла. Иными словами, строки написаны по вертикали. **Это сделано специально, чтобы сортировка занимала меньше времени.**

Строки состоят из строчных латинских букв: от символа "a" до символа "z" включительно. В таблице символов ASCII все эти буквы располагаются подряд и в алфавитном порядке, код буквы "a" равен 97, код буквы "z" равен 122.

**Формат выходного файла**

Выведите номера строк в том порядке, в котором они будут после k фаз цифровой сортировки.

**Примеры**

|  |  |
| --- | --- |
| input.txt | output.txt |
| 3 3 1 bab bba baa | 2 3 1 |
| 3 3 2 bab bba baa | 3 2 1 |
| 3 3 3 bab bba baa | 2 3 1 |

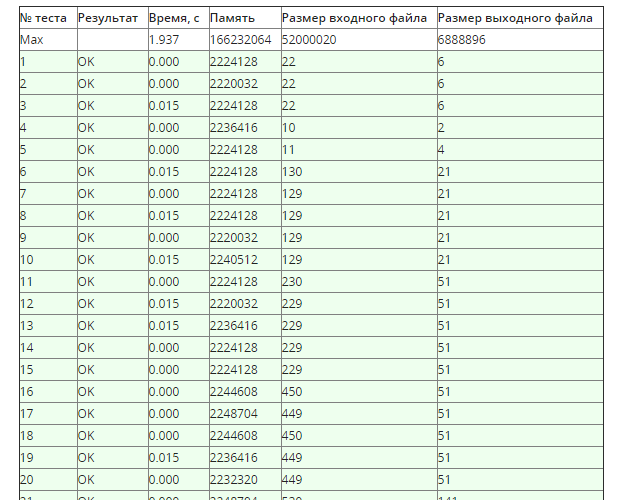
**Примечание 1**

Во всех примерах входных данных даны следующие строки:

* «bbb», имеющая индекс 1;
* «aba», имеющая индекс 2;
* «baa», имеющая индекс 3.

Разберем первый пример. Первая фаза цифровой сортировки отсортирует строки по последнему символу, таким образом, первой строкой окажется «aba» (индекс 2), затем «baa» (индекс 3), затем «bbb» (индекс 1). Таким образом, ответ равен «2 3 1».

Результат решения задачи



#include <fstream>

#include <string>

#include "edx-io.hpp"

using namespace std;

int main() {

long n, m, k;

io >> n >> m >> k;

string \*A = new std::string[m];

for (long i = 0; i < m; i++) {

io >> A[i];

}

//переменные для сохранения позиций по тз

long \*Pos = new long[n];

long \*PostPos = new long[n];

long \*PositionChooser[2] = { Pos, PostPos };

int Controller = 0;

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

Pos[i] = i;

}

int \*count = new int[123];

//Цифровая сортировка

for (int j = m - 1; m - j <= k; j--) {

for (int i = 97; i < 123; i++) {

count[i] = 0;

}

//Сортировка подсчетом

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

//Cчетчик для каждой буквы

count[A[j][PositionChooser[Controller][i]]]++;

}

for (int i = 98; i < 124; i++) {

//Каждый преведущий элемент суммируется с преведущим

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

}

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

PositionChooser[1 - Controller][--count[A[j][PositionChooser[Controller][i]]]] = PositionChooser[Controller][i];

}

Controller = ~Controller & 1;

}

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

io << PositionChooser[Controller][i] + 1 << ' ';

}

return 0;