ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА № 3

ПО ДИСЦИПЛИНЕ «Алгоритмы и структуры данных»

Выполнил: Мещеряков Владимир Алевтинович

Группа: P3218

Преподаватель: Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2019 г.

Week 3

1) **Сортировка целых чисел**

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 $A_i \cdot B_j$, где $1 \le i \le n$ и $1 \le j \le m$. Иными словами, каждый элемент первого массива нужно умножить на каждый элемент второго массива.
Пусть из этих чисел получится отсортированная последовательность C длиной $n \cdot m$. Выведите сумму каждого десятого элемента этой последовательности (то есть, $C_1 + C_{11} + C_{21} + \ldots$).

Формат входного файла

В первой строке содержатся числа n и m ($1 \le n, m \le 6000$) — 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

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.781	290435072	68699	16
1	OK	0.000	2588672	24	2
2	OK	0.000	2592768	34	1
3	OK	0.000	2600960	38	2
4	OK	0.015	2588672	106	10
5	OK	0.015	2596864	234	11
6	OK	0.000	2621440	698	11
7	OK	0.000	2613248	705	12
8	OK	0.000	2613248	586	12
9	OK	0.000	2666496	34325	12
10	OK	0.000	2637824	5769	12
11	OK	0.000	2637824	3498	12
12	OK	0.015	2662400	924	12
13	OK	0.000	2420736	3494	12
14	OK	0.000	2428928	5772	12
15	OK	0.000	2449408	34449	12
16	OK	0.015	2871296	34368	13
17	OK	0.015	2850816	4006	13
18	OK	0.015	2887680	2886	13
19	OK	0.015	2863104	4009	13
20	OK	0.015	2875392	34361	13
21	OK	0.046	7200768	34966	14
22	OK	0.031	7192576	9167	14
23	OK	0.031	7192576	9162	14
24	OK	0.031	7213056	34917	14
25	OK	0.312	50417664	39991	15
26	OK	0.312	52412416	28668	15
27	OK	0.359	50417664	40034	15
28	OK	0.906	146419712	51489	15
29	OK	0.890	146423808	51525	15
30	OK	1.765	290435072	68655	16
31	OK	1.781	290435072	68625	16
32	OK	1.750	290435072	68699	16

```cpp
#include <fstream>
using namespace std;

//Цифровая сортировка, разряд = 1 байт
void radixSort(long *arr, int size, int dig) {
	long *output = new long[size];
	long *count = new long[256];
	//Сортируем побайтово начиная с правого байта
	for (int pow = 0; pow <= dig; pow += 8) {
		//Сортировка подсчётом

		for (int i = 0; i < 256; i++) {
			count[i] = 0;
		}
		// Для каждого элемента (байта) считаем количество в исходном массиве
		for (int i = 0; i < size; i++) {
			count[(arr[i] >> pow) & 255]++;
		}
		//Увеличиваем каждый следующий элемент вспомогательного массива на предыдущий,
		//в итоге под каждым элементом (байтом) хранится его позиция в отсортированном массиве
		for (int i = 1; i < 256; i++) {
			count[i] += count[i - 1];
		}
		//Заполняем новый массив проходя вспомоготельный с конца, при этом уменьшая количество
равных элементов
		//чтобы корректно записывать одинаковые элементы
		for (int i = size - 1; i >= 0; i--) {
			output[count[(arr[i] >> pow) & 255] - 1] = arr[i];
			count[(arr[i] >> pow) & 255]--;
		}

		for (int i = 0; i < size; i++) {
			arr[i] = output[i];
		}
	}
}

int main() {
	int n, m;

	ifstream input("input.txt");
	input >> n >> m;

	long *A = new long[n];
	long *B = new long[m];
	long *C = new long[n * m];

	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];
		}
	}
	long maxB = 0;
	long pos;

	for (int i = 0; i < m; i++) {
		input >> B[i];
		//Заполнсяем массив C
		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();

	long maxValue = maxA * maxB;
	int dig = 1;
	//Считаем количество битов в максимальном числе
	while (maxValue >> dig > 0) {
		dig++;
	}
	dig--;

	radixSort(C, n*m, dig);



	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;
}
```

2) Цифровая сортировка

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 ($1 \le n \le 106$, $1 \le k \le m \le 106$, $n \cdot m \le 5 \cdot 107$). Далее находится описание строк, **но в нетривиальном формате**. Так, i-ая строка ($1 \le i \le n$) 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

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.828	166232064	52000020	6888896
1	OK	0.000	2236416	22	6
2	OK	0.000	2240512	22	6
3	OK	0.000	2236416	22	6
4	OK	0.000	2224128	10	2
5	OK	0.000	2224128	11	4
6	OK	0.000	2224128	130	21
7	OK	0.000	2236416	129	21
8	OK	0.000	2224128	129	21
9	OK	0.031	2220032	129	21
10	OK	0.015	2220032	129	21
11	OK	0.000	2220032	230	51
12	OK	0.000	2224128	229	51
13	OK	0.000	2236416	229	51
14	OK	0.000	2220032	229	51
15	OK	0.000	2224128	229	51
16	OK	0.000	2236416	450	51
17	OK	0.000	2236416	449	51
18	OK	0.015	2252800	450	51
19	OK	0.015	2236416	449	51
20	OK	0.015	2252800	449	51

191	OK	1.828	109875200	50000115	6888896
192	OK	0.281	109879296	50000114	6888896
193	OK	1.765	109879296	50000115	6888896
194	OK	1.125	109875200	50000115	6888896
195	OK	1.484	109879296	50000115	6888896
196	OK	0.578	108417024	50200019	1892
197	OK	0.109	108421120	50200014	1892
198	OK	0.562	108421120	50200018	1892
199	OK	0.375	108421120	50200018	1892
200	OK	0.453	108421120	50200018	1892
201	OK	0.781	102715392	50001016	588895
202	OK	0.109	102715392	50001014	588895
203	OK	0.750	102719488	50001016	588895
204	OK	0.406	102719488	50001016	588895
205	OK	0.187	102715392	50001015	588895
206	OK	0.609	102346752	50002017	288894
207	OK	0.109	102346752	50002014	288894
208	OK	0.609	102346752	50002016	288894
209	OK	0.359	102342656	50002016	288894
210	OK	0.609	102346752	50002016	288894
211	OK	1.312	105877504	50000216	3388895
212	OK	0.187	105877504	50000214	3388895
213	OK	1.187	105877504	50000215	3388895
214	OK	1.000	105877504	50000215	3388895
215	OK	0.781	105877504	50000215	3388895
216	OK	0.765	166232064	52000020	141
217	OK	0.234	166227968	52000014	141
218	OK	0.750	166227968	52000019	141
219	OK	0.703	166232064	52000019	141
220	OK	0.281	166232064	52000018	141
221	OK	0.562	103751680	50010017	48894
222	OK	0.093	103755776	50010014	48894
223	OK	0.562	103747584	50010017	48894
224	OK	0.250	103747584	50010017	48894
225	OK	0.203	103751680	50010017	48894
226	OK	0.562	103882752	50020018	23893
227	OK	0.109	103882752	50020014	23893
228	OK	0.562	103882752	50020017	23893
229	OK	0.484	103882752	50020017	23893
230	OK	0.484	103882752	50020017	23893

```cpp
#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];
	}

	long *Pos = new long[n];
	long *Pos2 = new long[n];
	//Создаём два массива позиций букв для хранения текущих и предыдущих
	long *pointer[2] = { Pos, Pos2 };
	//Переключатель между ними
	int pIn = 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--) {
		//Сортировка подсчётом
		for (int i = 97; i < 123; i++) {
			count[i] = 0;
		}

		for (int i = 0; i < n; i++) {
			//j - строка, второй индекс - позиция в исходной строке
			count[A[j][pointer[pIn][i]]]++;
		}

		for (int i = 98; i < 124; i++) {
			count[i] += count[i - 1];
		}

		for (int i = n - 1; i >= 0; i--) {
			//Во второй массив на место позиции отсортированной буквы ставится её старая
позиция
			pointer[1 - pIn][--count[A[j][pointer[pIn][i]]]] = pointer[pIn][i];
		}
		//Переключаем массивы
		pIn = ~pIn & 1;

	}

	for (long i = 0; i < n; i++) {
		io << pointer[pIn][i] + 1 << ' ';
	}
	return 0;
}
```