Санкт-Петербургский Научный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Задачи третьей недели

по курсу «Алгоритмы и структуры данных»

на Openedu

Выполнил: студент группы P3218

Артамонов Александр Владимирович

Санкт-Петербург, 2019г

Задача 1. Сортировка целых чисел

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 произведений элементов массивов A и B.

Примеры

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

Код программы (C++)

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

//Цифровая сортировка, в качестве разряда выбран 1 байт
void radix_sort(long *array, int size, int digit) {
	long *output = new long[size];
	long *count = new long[256];
	//Сортируем побайтово начиная с правого байта
	for (int pow = 0; pow <= digit; pow += 8) {
		//Сортировка подсчётом
		//1. Заполняем вспомогательный массив 0
		for (int i = 0; i < 256; i++) {
			count[i] = 0;
		}
		//2. Для каждого элемента (байта) считаем количество в исходном массиве
		for (int i = 0; i < size; i++) {
			count[(array[i] >> pow) & 255]++;
		}
		//3. 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];
		}
		//4. 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[(array[i] >> pow) & 255] - 1] = array[i];
			count[(array[i] >> pow) & 255]--;
		}

		for (int i = 0; i < size; i++) {
			array[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 maxNum = maxA * maxB;
		int digit = 1;
		//Считаем количество битов в максимальном числе
		while (maxNum >> digit > 0) {
			digit++;
		}
		digit--;

		radix_sort(C, n*m, digit);



		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;
}
```

Бенчмарк (задача 1)

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.796	290439168	68699	16
1	OK	0.015	2375680	24	2
2	OK	0.000	2371584	34	1
3	OK	0.000	2400256	38	2
4	OK	0.000	2367488	106	10
5	OK	0.000	2375680	234	11
6	OK	0.015	2383872	698	11
7	OK	0.015	2408448	705	12
8	OK	0.000	2392064	586	12
9	OK	0.000	2445312	34325	12
10	OK	0.015	2420736	5769	12
11	OK	0.015	2424832	3498	12
12	OK	0.000	2424832	924	12
13	OK	0.000	2424832	3494	12
14	OK	0.015	2420736	5772	12
15	OK	0.015	2445312	34449	12
16	OK	0.015	2887680	34368	13
17	OK	0.000	2863104	4006	13
18	OK	0.000	2887680	2886	13
19	OK	0.000	2850816	4009	13
20	OK	0.015	2875392	34361	13
21	OK	0.031	7213056	34966	14
22	OK	0.031	7192576	9167	14
23	OK	0.031	7192576	9162	14
24	OK	0.031	7204864	34917	14
25	OK	0.296	50409472	39991	15
26	OK	0.328	52408320	28668	15
27	OK	0.312	50413568	40034	15
28	OK	0.875	146419712	51489	15
29	OK	0.937	146419712	51525	15
30	OK	1.796	290439168	68655	16
31	OK	1.781	290435072	68625	16
32	OK	1.765	290435072	68699	16

Задача 2. Цифровая сортировка

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 ($1 \le n \le 10^6$, $1 \le k \le m \le 10^6$, $n \cdot m \le 5 \cdot 10^7$). 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

Код программы (C++)

```
#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;
}
```

Бенчмарк (задача 2)

191	OK	1.843	109879296	50000115	6888896
192	OK	0.296	109875200	50000114	6888896
193	OK	1.828	109879296	50000115	6888896
194	OK	1.140	109879296	50000115	6888896
195	OK	1.593	109879296	50000115	6888896
196	OK	0.578	108421120	50200019	1892
197	OK	0.109	108421120	50200014	1892
198	OK	0.562	108421120	50200018	1892
199	OK	0.375	108417024	50200018	1892
200	OK	0.484	108421120	50200018	1892
201	OK	0.765	102715392	50001016	588895
202	OK	0.125	102719488	50001014	588895
203	OK	0.734	102719488	50001016	588895
204	OK	0.390	102719488	50001016	588895
205	OK	0.187	102719488	50001015	588895
206	OK	0.609	102350848	50002017	288894
207	OK	0.109	102346752	50002014	288894
208	OK	0.609	102346752	50002016	288894
209	OK	0.359	102346752	50002016	288894
210	OK	0.609	102346752	50002016	288894
211	OK	1.218	105877504	50000216	3388895
212	OK	0.203	105877504	50000214	3388895
213	OK	1.203	105877504	50000215	3388895
214	OK	1.046	105877504	50000215	3388895
215	OK	0.812	105877504	50000215	3388895
216	OK	0.781	166232064	52000020	141
217	OK	0.234	166232064	52000014	141
218	OK	0.734	166227968	52000019	141
219	OK	0.703	166232064	52000019	141
220	OK	0.250	166227968	52000018	141
221	OK	0.578	103751680	50010017	48894
222	OK	0.109	103751680	50010014	48894
223	OK	0.593	103747584	50010017	48894
224	OK	0.250	103755776	50010017	48894
225	OK	0.203	103755776	50010017	48894
226	OK	0.546	103882752	50020018	23893
227	OK	0.093	103878656	50020014	23893
228	OK	0.562	103882752	50020017	23893
229	OK	0.500	103882752	50020017	23893
230	OK	0.484	103878656	50020017	23893