Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа № 3

по дисциплине «Алгоритмы и структуры данных»

Openedu – неделя 3

Подготовил:

студент группы P3217

Бураков Илья Алексеевич

Преподаватели:

Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург, 2019

Сортировка целых чисел

Условие

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 произведений элементов массивов A и B.

Примеры

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

Решение

openedu/week3/lab3_1.cpp

```
#include "edx-io.hpp"
#include <vector>
#include <string>
#include <algorithm>

using namespace std;

int n, m;

# define KTH_BYTE(k, val) ((val) >> 8 * k) & 0xFF

int main() {
	// read array sizes
	io >> n >> m;

	// read arrays
	auto a = vector<int>(n);
	auto b = vector<int>(m);
```

```
	for (auto &e : a) io >> e;
	for (auto &e : b) io >> e;

	// compose array to sort
	// STL vectors result in TLE! how's that possible
	auto c = new int[n * m];

	int max_val = 0;
	int next;
	for (int i = 0; i < n; i++){
		for (int j = 0; j < m; j++) {
			next = a[i] * b[j];
			c[i*m + j] = next;

			if (next > max_val) {
				max_val = next;
			}
		}
	}

	// i - radix sort phase
	auto result = new int[n * m];
	auto counts = new int[256];
	for (int byte_n = 0; (1LL << (byte_n * 8)) <= max_val; byte_n++) {
		memset(counts, 0, 256 * sizeof(int));

		// do element counting
		for (int i = 0; i < n * m; i++) {
			counts[KTH_BYTE(byte_n, c[i])]++;
		}

		// calculate prepending elements count for each element
		for (int i = 1; i < 256; i++) {
			counts[i] += counts[i - 1];
		}

		// assembling sorted array
		for (int i = n * m - 1; i >= 0; i--) {
			// calculate result position and set c[i] to it
			result[--counts[KTH_BYTE(byte_n, c[i])]] = c[i];
		}

		memcpy(c, result, sizeof(int) * n * m);
	}

	// print order that we've got
	long long sum = 0;
	for (int i = 0; i < n * m; i += 10) {
		sum += c[i];
	}

	io << sum;
	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.687	290295808	68699	16
1	OK	0.000	2224128	24	2
2	OK	0.015	2236416	34	1
3	OK	0.000	2220032	38	2

4	OK	0.015	2220032	106	10
5	OK	0.015	2244608	234	11
6	OK	0.015	2240512	698	11
7	OK	0.031	2252800	705	12
8	OK	0.031	2244608	586	12
9	OK	0.015	2297856	34325	12
10	OK	0.015	2273280	5769	12
11	OK	0.000	2281472	3498	12
12	OK	0.000	2293760	924	12
13	OK	0.000	2269184	3494	12
14	OK	0.015	2273280	5772	12
15	OK	0.015	2297856	34449	12
16	OK	0.000	2727936	34368	13
17	OK	0.000	2707456	4006	13
18	OK	0.000	2723840	2886	13
19	OK	0.015	2719744	4009	13
20	OK	0.015	2744320	34361	13
21	OK	0.031	7057408	34966	14
22	OK	0.031	7045120	9167	14
23	OK	0.031	7045120	9162	14
24	OK	0.031	7061504	34917	14
25	OK	0.296	50270208	39991	15
26	OK	0.296	52260864	28668	15
27	OK	0.281	50266112	40034	15
28	OK	0.843	146280448	51489	15
29	OK	0.859	146280448	51525	15
30	OK	1.671	290287616	68655	16
31	OK	1.687	290295808	68625	16
32	OK	1.687	290291712	68699	16

Цифровая сортировка

Условие

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

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

Решение

openedu/week3/lab3_2.cpp

```
#include "edx-io.hpp"
#include <vector>
#include <string>
#include <algorithm>

using namespace std;

int n, m, k;

vector<pair<char, int>> counting_sort(vector<pair<char, int>> &vec) {
	// indices 97 - 122 (26 letters), shift = 97
	auto counts = vector<int>(26, 0);

	// do element counting
```

```
	for (auto e : vec) {
		counts[e.first - 'a']++;
	}

	// calculate prepending elements count for each element
	for (int i = 1; i < counts.size(); i++) {
		counts[i] += counts[i - 1];
	}

	// assembling sorted array
	vector<pair<char, int>> result(vec.size());
	for (int i = vec.size() - 1; i >= 0; i--) {
		int result_position = --counts[vec[i].first - 'a'];
		result[result_position] = vec[i];
	}

	return result;
}

int main() {
	// read main input values
	io >> n >> m >> k;

	// read and store all m input strings in memory because they're in invalid
order!
	auto read_chars = vector<string>(m);
	for (auto &str: read_chars) {
		// read n chars - i-th chars of n strings
		io >> str;
	}

	// will be sorted vector of pairs: strings' ith characters (i = 0..m-1) and
their initial indices
	auto sorted = vector<pair<char, int>>(n);

	for (int i = 0; i < n; i++) {
		// '-' is temporary: will be replaced by real char from input
		sorted[i] = make_pair('-', i);
	}

	// i - radix sort phase
	for (int i = 0; i < k; i++) {
		// put read chars in appropriate order to "sorted" vec
		for (int j = 0; j < n; j++) {
			sorted[j].first = read_chars.back()[sorted[j].second];
		}
		read_chars.pop_back();

		// stable counting sort by i-th chars
		sorted = counting_sort(sorted);
	}

	// print order that we've got
	for (auto p : sorted) {
		io << p.second + 1 << ' ';
	}

	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.406	166260736	52000020	6888896

1	OK	0.015	2248704	22	6
2	OK	0.000	2248704	22	6
3	OK	0.015	2236416	22	6
4	OK	0.000	2236416	10	2
5	OK	0.000	2232320	11	4
6	OK	0.000	2248704	130	21
7	OK	0.015	2228224	129	21
8	OK	0.000	2236416	129	21
9	OK	0.000	2236416	129	21
10	OK	0.015	2236416	129	21
11	OK	0.000	2248704	230	51
12	OK	0.000	2236416	229	51
13	OK	0.000	2232320	229	51
14	OK	0.015	2236416	229	51
15	OK	0.000	2236416	229	51
16	OK	0.000	2236416	450	51
17	OK	0.000	2252800	449	51
18	OK	0.000	2232320	450	51
19	OK	0.000	2236416	449	51
20	OK	0.000	2248704	449	51
21	OK	0.000	2240512	530	141
22	OK	0.015	2260992	529	141
23	OK	0.000	2256896	529	141
24	OK	0.015	2240512	529	141
25	OK	0.000	2256896	529	141
26	OK	0.000	2244608	1212	21
27	OK	0.000	2236416	1210	21
28	OK	0.000	2236416	1211	21
29	OK	0.000	2236416	1211	21
30	OK	0.000	2256896	1211	21
31	OK	0.000	2232320	2031	692
32	OK	0.015	2240512	2030	692
33	OK	0.000	2236416	2030	692
34	OK	0.015	2236416	2030	692
35	OK	0.015	2248704	2030	692
36	OK	0.015	2248704	2610	141
37	OK	0.000	2248704	2609	141
38	OK	0.000	2260992	2610	141
39	OK	0.000	2260992	2610	141
40	OK	0.000	2248704	2609	141
41	OK	0.000	2256896	4051	692
42	OK	0.000	2260992	4050	692

43	OK	0.000	2240512	4051	692
44	OK	0.015	2244608	4051	692
45	OK	0.015	2256896	4051	692
46	OK	0.015	2252800	6012	21
47	OK	0.000	2252800	6010	21
48	OK	0.000	2265088	6012	21
49	OK	0.000	2252800	6012	21
50	OK	0.000	2252800	6010	21
51	OK	0.000	2269184	10213	292
52	OK	0.000	2260992	10211	292
53	OK	0.000	2252800	10212	292
54	OK	0.015	2260992	10212	292
55	OK	0.000	2273280	10212	292
56	OK	0.000	2277376	20052	3893
57	OK	0.015	2273280	20051	3893
58	OK	0.000	2285568	20052	3893
59	OK	0.000	2289664	20052	3893
60	OK	0.015	2293760	20051	3893
61	OK	0.000	2318336	26012	141
62	OK	0.000	2318336	26010	141
63	OK	0.000	2318336	26012	141
64	OK	0.000	2314240	26011	141
65	OK	0.000	2334720	26012	141
66	OK	0.000	2285568	40413	692
67	OK	0.000	2310144	40411	692
68	OK	0.000	2289664	40413	692
69	OK	0.000	2293760	40412	692
70	OK	0.000	2289664	40413	692
71	OK	0.000	2318336	52014	141
72	OK	0.000	2318336	52011	141
73	OK	0.015	2322432	52013	141
74	OK	0.015	2318336	52013	141
75	OK	0.000	2330624	52013	141
76	OK	0.000	2355200	102015	292
77	OK	0.000	2355200	102012	292
78	OK	0.000	2367488	102014	292
79	OK	0.000	2351104	102014	292
80	OK	0.000	2363392	102014	292
81	OK	0.000	2543616	200033	108894
82	OK	0.015	2551808	200032	108894
83	OK	0.015	2543616	200032	108894
84	OK	0.000	2547712	200032	108894

85	OK	0.015	2543616	200032	108894
86	OK	0.015	2445312	250112	23893
87	OK	0.000	2461696	250111	23893
88	OK	0.015	2457600	250112	23893
89	OK	0.015	2441216	250111	23893
90	OK	0.015	2457600	250112	23893
91	OK	0.015	2940928	400053	108894
92	OK	0.000	2945024	400052	108894
93	OK	0.015	2945024	400053	108894
94	OK	0.000	2945024	400053	108894
95	OK	0.015	2945024	400053	108894
96	OK	0.000	2891776	501014	3893
97	OK	0.000	2891776	501012	3893
98	OK	0.015	2891776	501014	3893
99	OK	0.000	2891776	501014	3893
100	OK	0.000	2891776	501013	3893
101	OK	0.031	3985408	1000414	23893
102	OK	0.000	3985408	1000412	23893
103	OK	0.015	3993600	1000414	23893
104	OK	0.015	3981312	1000413	23893
105	OK	0.015	3989504	1000414	23893
106	OK	0.093	10629120	2400018	21
107	OK	0.000	10625024	2400013	21
108	OK	0.093	10633216	2400018	21
109	OK	0.078	10633216	2400018	21
110	OK	0.093	10629120	2400018	21
111	OK	0.062	7639040	2500113	288894
112	OK	0.031	7639040	2500112	288894
113	OK	0.062	7639040	2500113	288894
114	OK	0.031	7639040	2500112	288894
115	OK	0.046	7634944	2500113	288894
116	OK	0.046	10326016	4004016	8893
117	OK	0.015	10334208	4004013	8893
118	OK	0.062	10326016	4004016	8893
119	OK	0.031	10330112	4004015	8893
120	OK	0.062	10330112	4004016	8893
121	OK	0.093	12652544	5000215	288894
122	OK	0.031	12652544	5000213	288894
123	OK	0.109	12652544	5000214	288894
124	OK	0.062	12652544	5000214	288894
125	OK	0.093	12652544	5000214	288894
126	OK	0.203	23457792	10000216	588895

127	OK	0.046	23453696	10000214	588895
128	OK	0.203	23457792	10000215	588895
129	OK	0.062	23457792	10000215	588895
130	OK	0.078	23457792	10000215	588895
131	OK	0.421	45064192	20000216	1288895
132	OK	0.078	45064192	20000214	1288895
133	OK	0.390	45064192	20000215	1288895
134	OK	0.281	45064192	20000215	1288895
135	OK	0.312	45064192	20000215	1288895
136	OK	0.406	52699136	25001015	288894
137	OK	0.078	52707328	25001013	288894
138	OK	0.390	52695040	25001015	288894
139	OK	0.250	52699136	25001015	288894
140	OK	0.187	52699136	25001015	288894
141	OK	0.578	84078592	26000018	141
142	OK	0.156	84070400	26000013	141
143	OK	0.609	84078592	26000018	141
144	OK	0.562	84078592	26000018	141
145	OK	0.421	84078592	26000018	141
146	OK	0.359	55164928	25100017	1892
147	OK	0.062	55099392	25100013	1892
148	OK	0.359	55164928	25100017	1892
149	OK	0.234	55164928	25100017	1892
150	OK	0.125	55164928	25100016	1892
151	OK	0.343	53051392	25010016	23893
152	OK	0.046	53055488	25010013	23893
153	OK	0.359	53051392	25010016	23893
154	OK	0.093	53055488	25010015	23893
155	OK	0.312	53051392	25010016	23893
156	OK	0.640	59867136	25000114	3388895
157	OK	0.156	59863040	25000113	3388895
158	OK	0.609	59867136	25000114	3388895
159	OK	0.296	59867136	25000114	3388895
160	OK	0.171	59867136	25000113	3388895
161	OK	0.546	84254720	40040018	8893
162	OK	0.078	84258816	40040014	8893
163	OK	0.546	84254720	40040018	8893
164	OK	0.265	84254720	40040017	8893
165	OK	0.437	84267008	40040018	8893
166	OK	0.640	93782016	40400019	692
167	OK	0.125	93761536	40400014	692
168	OK	0.671	93782016	40400019	692

169	OK	0.312	93782016	40400018	692
170	OK	0.140	93761536	40400016	692
171	OK	0.562	82370560	40004017	108894
172	OK	0.093	82370560	40004014	108894
173	OK	0.578	82370560	40004017	108894
174	OK	0.265	82366464	40004016	108894
175	OK	0.359	82370560	40004017	108894
176	OK	0.812	85082112	40000416	1288895
177	OK	0.125	85082112	40000414	1288895
178	OK	0.890	85082112	40000416	1288895
179	OK	0.218	85082112	40000415	1288895
180	OK	0.140	85082112	40000414	1288895
181	OK	0.921	133156864	51000019	292
182	OK	0.218	133152768	51000014	292
183	OK	0.953	133156864	51000019	292
184	OK	0.265	133160960	51000018	292
185	OK	0.703	133156864	51000019	292
186	OK	0.703	105156608	50100018	3893
187	OK	0.125	105050112	50100014	3893
188	OK	0.687	105156608	50100018	3893
189	OK	0.265	105156608	50100018	3893
190	OK	0.437	105156608	50100018	3893
191	OK	1.406	117891072	50000115	6888896
192	OK	0.312	117891072	50000114	6888896
193	OK	1.328	117891072	50000115	6888896
194	OK	0.890	117891072	50000115	6888896
195	OK	1.187	117891072	50000115	6888896
196	OK	0.750	108507136	50200019	1892
197	OK	0.125	108445696	50200014	1892
198	OK	0.734	108503040	50200018	1892
199	OK	0.453	108507136	50200018	1892
200	OK	0.578	108507136	50200018	1892
201	OK	0.828	103518208	50001016	588895
202	OK	0.109	103530496	50001014	588895
203	OK	0.828	103518208	50001016	588895
204	OK	0.531	103518208	50001016	588895
205	OK	0.187	103518208	50001015	588895
206	OK	0.765	102756352	50002017	288894
207	OK	0.109	102764544	50002014	288894
208	OK	0.734	102756352	50002016	288894
209	OK	0.437	102756352	50002016	288894
210	OK	0.734	102756352	50002016	288894

211	OK	1.156	109879296	50000216	3388895
212	OK	0.187	109879296	50000214	3388895
213	OK	1.140	109879296	50000215	3388895
214	OK	0.968	109879296	50000215	3388895
215	OK	0.750	109879296	50000215	3388895
216	OK	1.203	166256640	52000020	141
217	OK	0.296	166252544	52000014	141
218	OK	1.234	166256640	52000019	141
219	OK	1.078	166248448	52000019	141
220	OK	0.343	166260736	52000018	141
221	OK	0.703	103845888	50010017	48894
222	OK	0.109	103854080	50010014	48894
223	OK	0.718	103849984	50010017	48894
224	OK	0.296	103849984	50010017	48894
225	OK	0.250	103845888	50010017	48894
226	OK	0.687	103981056	50020018	23893
227	OK	0.109	103948288	50020014	23893
228	OK	0.671	103981056	50020017	23893
229	OK	0.609	103989248	50020017	23893
230	OK	0.609	103989248	50020017	23893