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

Лабораторная работа №3

по дисциплине

«Алгоритмы и структуры данных»

Выполнил: Студент группы P3217

Сергачев Данила Дмитриевич

Преподаватели:

Романов Алексей Андреевич и

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2018 г

Задание №1

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.

Решение

```
using System;
using System.IO;
using System.Linq;

namespace Task1
{
    public class Program
    {
        private const int w = 32;
        private const int d = 8;

        static void Main(string[] args)
        {
            Solve();
        }

        private static void Solve()
        {
            int[] mainArr;
            using (var input = new StreamReader("input.txt"))
            {
                int[] arr1;

                int mArrItem;

                var argArr = input.ReadLine().Split(' ').Select(t => int.Parse(t)).ToArray();

                arr1 = input.
                    ReadLine().
                    Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).
                    Select(t => int.Parse(t)).
                    ToArray();

                GC.Collect();

                mainArr = new int[argArr[0] * argArr[1]];

                var t2 = input.ReadLine().Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

                for (int i = 0, counter = 0; i < argArr[0] * argArr[1]; i+= argArr[0], counter++)
                {
                    mArrItem = int.Parse(t2[counter]);
```

```
                    for(int j = 0; j < argArr[0]; j++)
                    {
                        mainArr[i + j] = mArrItem * arr1[j];
                    }
                }
                t2 = null;
                arr1 = null;

                GC.Collect();
            }

            using (var output = new StreamWriter("output.txt"))
            {
                output.WriteLine(RadixSort(mainArr));
            }

        }

        private static long RadixSort(int[] a)
        {
            int[] b = null;

            long sum = 0;
            for (int p = 0; p < w / d; p++)
            {
                int[] c = new int[1 << d];

                b = new int[a.Length];
                for (int i = 0; i < a.Length; i++)
                    c[(a[i] >> d * p) & ((1 << d) - 1)]++;
                for (int i = 1; i < 1 << d; i++)
                    c[i] += c[i - 1];
                for (int i = a.Length - 1; i >= 0; i--)
                {
                    b[--c[(a[i] >> d * p) & ((1 << d) - 1)]] = a[i];
                }

                a = b;

                GC.Collect();
            }

            for (int i = 0; i < a.Length; i += 10)
            {
                sum += b[i];
            }

            return sum;
        }
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.796	300257280	68699	18
1	OK	0.031	11128832	24	4
2	OK	0.015	11059200	34	3
3	OK	0.015	11104256	38	4
4	OK	0.046	11091968	106	12
5	OK	0.046	11104256	234	13
6	OK	0.015	11116544	698	13
7	OK	0.015	11177984	705	14
8	OK	0.031	11116544	586	14
9	OK	0.031	11636736	34325	14
10	OK	0.031	11386880	5769	14
11	OK	0.031	11288576	3498	14
12	OK	0.031	11231232	924	14
13	OK	0.031	11243520	3494	14
14	OK	0.031	11194368	5772	14
15	OK	0.031	11608064	34449	14
16	OK	0.031	12128256	34368	15
17	OK	0.031	11673600	4006	15

Задание №2

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.**

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.

Решение

```
using System;
using System.IO;
using System.Linq;
using System.Text;

namespace Task2
{
    public class Task2
    {
        private static int[] countingArr = new int[130];
        private static int[] resIndexs;

        static void Main(string[] args)
        {
            Solve();
        }

        private static void Solve()
        {
            using (var output = new StreamWriter("output.txt"))
            {
                string[] stdin = File.ReadAllLines("input.txt");

                int[] parameters = stdin[0].Split(' ').Select(x => int.Parse(x)).ToArray();

                resIndexs = RadixSort(stdin, Enumerable.Range(0, parameters[0] + 1).ToArray(), parameters[0],
parameters[1], parameters[2]);

                StringBuilder st = new StringBuilder();
                for (int i = 1; i < resIndexs.Length; i++)
                {
                    st.Append(resIndexs[i]);
                    st.Append(" ");
                }

                output.Write(st);
            }
        }

        private static int[] RadixSort(string[] array, int[] indexes, int n, int m, int k)
        {
            for (int i = 0; i < k; i++)
            {
                resIndexs = new int[indexes.Length];
                indexes = SortCounting(array, indexes, m - i);
```

```
                resIndexs = null;

                if (i % 100000 == 0)
                    GC.Collect();
            }

            return indexes;
        }

        private static int[] SortCounting(string[] array, int[] indexes, int phase)
        {
            for (int i = 0; i < countingArr.Length; i++)
            {
                countingArr[i] = 0;
            }

            for (int i = 0; i < array[phase].Length; i++)
                countingArr[array[phase][i]]++;
            for (int i = 98; i <= 125; i++)
                countingArr[i] += countingArr[i - 1];

            int charValue, countedIndex;

            for (int i = array[phase].Length; i > 0; i--)
            {
                charValue = array[phase][indexes[i] - 1];
                countedIndex = countingArr[charValue];
                resIndexs[countedIndex] = indexes[i];
                countingArr[charValue]--;
            }

            return resIndexs;
        }
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.578	Превышение	51000019	3388895
1	OK	0.046	11165696	22	6
2	OK	0.046	11128832	22	6
3	OK	0.031	11149312	22	6
4	OK	0.031	11194368	10	2
5	OK	0.015	11210752	11	4
6	OK	0.031	11132928	130	21
7	OK	0.046	11149312	129	21
8	OK	0.031	11157504	129	21
9	OK	0.046	11182080	129	21
10	OK	0.046	11153408	129	21