Санкт-Петербургский Национальный Исследовательский
Университет Информационных Технологий, Механики и Оптики
Мегафакультет компьютерных технологий и управления

Дисциплина
«Алгоритмы и структуры данных»

Лабораторная работа №3

Выполнил:
Студент группы P3218
Рябов Сергей Витальевич
Преподаватель:
Муромцев Дмитрий Ильич

Санкт-Петербург,
2018

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 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 произведений элементов массивов A и B.

Исходный код (C#):

```
using System;
using System.IO;

namespace ADS.Week3
{
    public class Task1
    {
        private static int GetRadixValue(int number, int radixSize, int radixNumber)
        {
            return (number >> (radixSize * radixNumber)) & ((1 << radixSize) - 1);
        }
        private static void RadixSort(int[] array, int radixSize = 8)
        {
            int radixCount = sizeof(int) * 8 / radixSize;
            int[] count = new int[1 << radixSize];
            int[] buffer = new int[array.Length];

            for (int i = 0; i < radixCount; i++)
            {
                Array.Clear(count, 0, count.Length);
                for (int j = 0; j < array.Length; j++)
                {
                    count[GetRadixValue(array[j], radixSize, i)]++;
                }
                for (int j = 1; j < count.Length; j++)
                {
                    count[j] += count[j - 1];
                }
                for (int j = array.Length - 1; j >= 0; j--)
                {
                    int radixValue = GetRadixValue(array[j], radixSize, i);
                    count[radixValue]--;
                    buffer[count[radixValue]] = array[j];
                }
                Array.Copy(buffer, array, array.Length);
            }
        }
```

```
        private static void main(string[] args)
        {
            int n, m;
            int[] array1, array2;
            using (StreamReader streamReader = new StreamReader("input.txt"))
            {
                string[] str = streamReader.ReadLine().Split(' ');
                n = int.Parse(str[0]);
                m = int.Parse(str[1]);

                array1 = new int[n];
                str = streamReader.ReadLine().Split(' ');
                for (int i = 0; i < n; i++)
                {
                    array1[i] = int.Parse(str[i]);
                }
                array2 = new int[m];
                str = streamReader.ReadLine().Split(' ');
                for (int i = 0; i < m; i++)
                {
                    array2[i] = int.Parse(str[i]);
                }
            }

            int[] array = new int[n * m];
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < m; j++)
                {
                    array[i * m + j] = array1[i] * array2[j];
                }
            }

            RadixSort(array);

            long sum = 0;
            for (int i = 0; i < array.Length; i += 10)
            {
                sum += array[i];
            }
            using(StreamWriter streamWriter = new StreamWriter("output.txt"))
            {
                streamWriter.Write(sum);
            }
        }
    }
}
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.796	299839488	68699	16
1	OK	0.078	10162176	24	2
2	OK	0.031	10121216	34	1
3	OK	0.031	10149888	38	2
4	OK	0.031	10153984	106	10
5	OK	0.015	10149888	234	11
6	OK	0.031	10145792	698	11
7	OK	0.046	10174464	705	12
8	OK	0.031	10186752	586	12
9	OK	0.015	10670080	34325	12
10	OK	0.046	10305536	5769	12
11	OK	0.031	10260480	3498	12
12	OK	0.031	10203136	924	12
13	OK	0.015	10264576	3494	12
14	OK	0.031	10289152	5772	12
15	OK	0.015	10665984	34449	12
16	OK	0.031	11112448	34368	13
17	OK	0.031	10715136	4006	13
18	OK	0.031	10661888	2886	13
19	OK	0.031	10702848	4009	13
20	OK	0.031	11116544	34361	13
21	OK	0.062	15486976	34966	14

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.

Исходный код (C#):

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

namespace ADS.Week3
{
    public class Task2
    {
        private static int[] RadixSort(string[] strings, int n, int m, int k)
        {
            int[] count = new int['z' - 'a' + 1];
            int[] indicies = new int[n];
            int[] newIndicies = new int[n];

            for (int i = 0; i < n; i++)
            {
                indicies[i] = i;
            }

            for (int i = m - 1; i >= m - k ; i--)
            {
                Array.Clear(count, 0, count.Length);
                for (int j = 0; j < n; j++)
                {
                    count[strings[i][j] - 'a']++;
                }
                for (int j = 1; j < count.Length; j++)
                {
                    count[j] += count[j - 1];
                }
                for (int j = n - 1; j >= 0; j--)
                {
                    int radixValue = strings[i][indicies[j]] - 'a';
                    count[radixValue]--;
                    newIndicies[count[radixValue]] = indicies[j];
                }
                int[] buffer = newIndicies;
                newIndicies = indicies;
                indicies = buffer;
            }
            return indicies;
        }
        private static void Main(string[] args)
        {
            int n, m, k;
            string[] strings;
            using (StreamReader streamReader = new StreamReader("input.txt"))
            {
                string[] str = streamReader.ReadLine().Split(' ');
                n = int.Parse(str[0]);
                m = int.Parse(str[1]);
                k = int.Parse(str[2]);

                strings = new string[m];
                for (int i = 0; i < m; i++)
                {
                    strings[i] = streamReader.ReadLine();
                }
            }

            int[] result = RadixSort(strings, n, m, k);
```

```
            using (StreamWriter streamWriter = new StreamWriter("output.txt"))
            {
                for (int i = 0; i < result.Length; i++)
                {
                    streamWriter.Write("{0} ", result[i] + 1);
                }
            }
        }
    }
}
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.859	173326336	52000020	6888896
1	OK	0.015	10125312	22	6
2	OK	0.031	10207232	22	6
3	OK	0.031	10186752	22	6
4	OK	0.031	10137600	10	2
5	OK	0.062	10145792	11	4
6	OK	0.031	10162176	130	21
7	OK	0.031	10166272	129	21
8	OK	0.046	10178560	129	21
9	OK	0.031	10108928	129	21
10	OK	0.031	10145792	129	21
11	OK	0.031	10125312	230	51
12	OK	0.031	10190848	229	51
13	OK	0.015	10104832	229	51
14	OK	0.031	10113024	229	51
15	OK	0.031	10108928	229	51
16	OK	0.031	10072064	450	51
17	OK	0.031	10145792	449	51
18	OK	0.031	10162176	450	51
19	OK	0.031	10096640	449	51
20	OK	0.015	10145792	449	51