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

Лабораторная работа №10

по дисциплине

«Алгоритмы и структуры данных»

Выполнил: Студент группы P3217

Сергачев Данила Дмитриевич

Преподаватели:

Романов Алексей Андреевич и

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2019 г

Задание №1

Префикс-функция

1.0 из 1.0 балла (оценивается)

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Постройте префикс-функцию для всех непустых префиксов заданной строки s.

Решение

```
using System;
using System.IO;

namespace Week10
{
    class Task1
    {
        static void Mzin1(string[] args)
        {
            using (StreamWriter sw = new StreamWriter("output.txt"))
            {
                string[] inputString = File.ReadAllLines("input.txt");
                int[] prefixFunction = getPrefixFunction(inputString[0]);

                for (int i = 0; i < prefixFunction.Length; i++)
                {
                    sw.Write(prefixFunction[i] + " ");
                }
            }
        }

        private static int[] getPrefixFunction(String text)
        {
            var prefixFunction = new int[text.Length];

            for (int i = 1; i < text.Length; i++)
            {
                int j = prefixFunction[i - 1]; //текущая длина префикса, который мы хотим
продолжить

                while (j > 0 && text[i] != text[j]) //пока мы не можем продолжить текущий
префикс
                {
                    j = prefixFunction[j - 1]; //уменьшаем его длину до следующей
возможной
                }

                //Теперь j - максимальная длина префикса, который мы можем продолжить,
                //или 0, если такового не существует.
                if (text[i] == text[j])
                {
```

```
                    prefixFunction[i] = j + 1;
                }
                else
                {
                    prefixFunction[i] = j;
                }
            }
            return prefixFunction;
        }
    }
}
```

Результат работы

Верное решение!
Результаты работы Вашего решения

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.250	23359488	1000002	6888890
1	OK	0.031	10035200	8	12
2	OK	0.031	9994240	9	14
3	OK	0.031	10092544	3	2
4	OK	0.031	10084352	4	4
5	OK	0.031	10002432	4	4

Задание №2

Z-функция

1.0 из 1.0 балла (оценивается)

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Постройте Z-функцию для заданной строки s.

Решение

```
using System;
using System.IO;

namespace Week10
{
    class Task2
    {
        static void Main2(string[] args)
        {
            using (StreamWriter sw = new StreamWriter("output.txt"))
            {
                string[] inputString = File.ReadAllLines("input.txt");
                int[] zFunction = getZFunction(inputString[0]);

                for (int i = 1; i < zFunction.Length; i++)
                {
                    sw.Write(zFunction[i] + " ");
                }
            }
        }

        private static int[] getZFunction(String text)
        {
            var lenght = text.Length;
            var zFunction = new int[lenght];

            for(int i = 1, l = 0, r = 0; i  < lenght; i++)
            {
                if(i <= r)
                {
                    zFunction[i] = Math.Min(r - i + 1, zFunction[i - l]);
                }
                while(i+zFunction[i] < lenght && text[zFunction[i]] == text[i+
zFunction[i]]) { zFunction[i]++; }
                if(i + zFunction[i] - 1 > r)
                {
                    l = i;
                    r = i + zFunction[i] - 1;
                }
            }

            return zFunction;
        }
    }
}
```

Результат работы

Верное решение!
Результаты работы Вашего решения

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.234	23306240	1000002	6888888
1	OK	0.031	9965568	8	10
2	OK	0.031	10067968	9	12
3	OK	0.031	9977856	4	2

Задание №3

Декомпозиция строки

ЭТОТ ЭЛЕМЕНТ КУРСА ОЦЕНИВАЕТСЯ КАК 'ЛАБОРАТОРНАЯ РАБОТА'
ВЕС: 2.0

Добавить страницу в мои закладки

Декомпозиция строки

2.0 из 2.0 баллов (оценивается)

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Строка `ABCABCDEDEDEF` содержит подстроку `ABC`, повторяющуюся два раза подряд, и подстроку `DE`, повторяющуюся три раза подряд. Таким образом, ее можно записать как `ABC*2+DE*3+F`, что занимает меньше места, чем исходная запись той же строки.

Ваша задача - построить наиболее экономное представление данной строки s в виде, продемонстрированном выше, а именно, подобрать такие $s_1, a_1, \ldots, s_k, a_k$, где s_i - строки, а a_i - числа, чтобы $s = s_1 \cdot a_1 + \ldots + s_k \cdot a_k$. Под операцией умножения строки на целое положительное число подразумевается конкатенация одной или нескольких копий строки, число которых равно числовому множителю, то есть, `ABC*2=ABCABC`. При этом требуется минимизировать общую длину итогового описания, в котором компоненты разделяются знаком `+`, а умножение строки на число записывается как умножаемая строка и множитель, разделенные знаком `*`. Если же множитель равен единице, его, вместе со знаком `*`, допускается не указывать.

Решение

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

namespace Week10
{
    public class Task3
    {
        private static string[] _input;
        private static int _currentLineIndex;
        private const long E15 = 1000000000000000;

        private static string ReadLine()
        {
            return _input[_currentLineIndex++];
        }

        public static void Main(string[] args)
        {
            _input = File.ReadAllLines("input.txt");

            string s = ReadLine();
            int[] d = new int[s.Length + 1];
            int[] from = new int[s.Length + 1];
            int[] length = new int[s.Length + 1];
```

```
for (int i = 0; i < s.Length + 1; i++)
{
    d[i] = int.MaxValue;
}
d[0] = 0;

for (int i = 0; i < s.Length; i++)
{
    int[] p = new int[s.Length + 1];
    p[1] = 0;

    if (d[i + 1] > d[i] + 1)
    {
        d[i + 1] = d[i] + 1;
        from[i + 1] = i;
        length[i + 1] = 1;
    }

    int k = 0;
    for (int j = 2; i + j - 1 < s.Length; j++)
    {
        while (s[i + j - 1] != s[i + k] && k > 0)
        {
            k = p[k];
        }

        if (s[i + j - 1] == s[i + k])
        {
            k++;
        }

        p[j] = k;

        if (j % (j - p[j]) == 0)
        {
            if (d[i + j] > d[i] + (j - p[j]))
            {
                d[i + j] = d[i] + (j - p[j]);
                from[i + j] = i;
                length[i + j] = j - p[j];
            }
        }
    }
}

StringBuilder sb = new StringBuilder();
string[] parts = new string[s.Length];
int[] partsCount = new int[s.Length];
int al = 0;

for (int i = s.Length; i > 0;)
{
    parts[al] = s.Substring(from[i], length[i]);
    partsCount[al] = (i - from[i]) / length[i];
    al++;
    i = from[i];
}

int maxAl = al - 1;
bool canAppendPrev = true;


for (al--; al >= 0; al--)
{
```

```
                bool optimized = (parts[al].Length > 2 || partsCount[al] != 2) &&
(parts[al].Length != 1 || partsCount[al] != 3);
                if (optimized)
                {
                    bool oldCanAppend = canAppendPrev;
                    canAppendPrev = partsCount[al] <= 1;
                    if ((al != maxAl) && (!oldCanAppend || !canAppendPrev))
                    {
                        sb.Append("+");
                    }
                    sb.Append(parts[al]);
                    if (!canAppendPrev)
                    {
                        sb.Append("*");
                        sb.Append(partsCount[al]);
                    }
                }
                else
                {
                    string o = parts[al] + parts[al];
                    if (partsCount[al] == 3)
                    {
                        o += parts[al];
                    }

                    if (!canAppendPrev)
                    {
                        sb.Append("+");
                    }
                    sb.Append(o);

                    canAppendPrev = true;
                }
            }

            File.WriteAllText("output.txt", sb.ToString());
        }
    }
}
```

Результат работы

Верное решение!
Результаты работы Вашего решения

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.265	11984896	5002	5000
1	OK	0.031	10100736	15	12
2	OK	0.031	10051584	7	5
3	OK	0.031	10100736	3	1
4	OK	0.015	10072064	4	2
5	OK	0.031	10067968	5	3
6	OK	0.046	10108928	6	3
7	OK	0.015	10088448	7	3