Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Алгоритмы и структуры данных»

ОТЧЁТ

по лабораторной работе №9 (Week 9 Openedu)

Студент Луговских Савелий Михайлович P3218

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

Задача 1 Наивный поиск подстроки в строке 3
Исходный код к задаче 1 3
Бенчмарк к задаче 1 4
Задача 2. Карта 7
Исходный код к задаче 2 8
Бенчмарк к задаче 2 9

Задача 1 Наивный поиск подстроки в строке

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

Даны строки p и t. Требуется найти все вхождения строки p в строку t в качестве подстроки.

Формат входного файла

Первая строка входного файла содержит p, вторая — t ($1 \leq p, t \leq 10^4$). Строки состоят из букв латинского алфавита.

Формат выходного файла

В первой строке выведите число вхождений строки p в строку t. Во второй строке выведите в возрастающем порядке номера символов строки t, с которых начинаются вхождения p. Символы нумеруются с единицы.

Примеры

input.txt	output.txt
aba abaCaba	2 1 5

Исходный код к задаче 1

```
class Lab9_1
{
    public static void Main(string[] args)
    {
        var app = new Lab9_1();
        app.DoWork(args);
    }

    //Префикс-функция для КМП
    public static int[] PrefFunc(string x)
    {
        //Инициализация массива-результата длиной X
        int[] res = new int[x.Length];
        int i = 0;
        int j = -1;
        res[0] = -1;
        //Вычисление префикс-функции
        while (i < x.Length - 1)
        {
            while ((j >= 0) && (x[j] != x[i]))
                j = res[j];
            i++;
            j++;
```

```
            if (x[i] == x[j])
                res[i] = res[j];
            else
                res[i] = j;
        }
        return res;//Возвращение префикс-функции
    }

    //Функция поиска алгоритмом КМП
    public static List<int> KMP(string x, string s)
    {
        List<int> numbers = new List<int>(); //Объявление строки с номерами позиций
        if (x.Length > s.Length) return numbers; //Возвращает 0 поиск если образец больше
исходной строки
                                                 //Вызов префикс-функции
        int[] d = PrefFunc(x);
        int i = 0, j;
        var prev = -1;
        while (i < s.Length)
        {
            var t = i;

            for (j = 0; (i < s.Length) && (j < x.Length); i++, j++)
                while ((j >= 0) && (x[j] != s[i]))
                    j = d[j];

            if (j == x.Length)
            {
                var pos = i - j;
                if (pos != prev)
                {
                    prev = pos;
                    numbers.Add(pos);
                }
            }

            i = t;
            i++;
        }

        return numbers; //Возвращение результата поиска
    }

    private void DoWork(string[] args)
    {
        using (StreamWriter sw = new StreamWriter("output.txt"))
        {
            string[] stdin = File.ReadAllLines("input.txt");
            var numbers = KMP(stdin[0], stdin[1]);
            var count = numbers.Count;
            sw.WriteLine(count);
            for (int i = 0; i < count; i++)
            {
                sw.Write(numbers[i] + 1 + " ");
            }
        }
    }
}
```

Бенчмарк к задаче 1

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.187	11354112	20003	48890

1	OK	0.015	10260480	14	7
2	OK	0.031	10227712	6	5
3	OK	0.031	10231808	6	3
4	OK	0.031	10272768	7	7
5	OK	0.031	10166272	7	3
6	OK	0.031	10272768	9	7
7	OK	0.015	10178560	10	5
8	OK	0.046	10203136	3004	3
9	OK	0.031	10190848	3028	7
10	OK	0.015	10207232	2656	429
11	OK	0.015	10395648	2005	8895
12	OK	0.031	10235904	4003	7
13	OK	0.015	10297344	3004	3
14	OK	0.015	10256384	2252	1850
15	OK	0.031	10215424	2021	186
16	OK	0.031	10399744	2008	8884
17	OK	0.031	10358784	3004	3904
18	OK	0.031	10272768	2670	3
19	OK	0.031	10182656	3028	7
20	OK	0.031	10244096	2404	691
21	OK	0.031	10407936	2005	8899
22	OK	0.031	10186752	4003	7
23	OK	0.031	10252288	2670	3
24	OK	0.015	10317824	2252	1886
25	OK	0.031	10248192	2022	190
26	OK	0.015	10383360	2008	8884
27	OK	0.015	10293248	3004	3904
28	OK	0.046	10240000	5337	3
29	OK	0.046	10223616	5028	8
30	OK	0.031	10285056	4372	648
31	OK	0.031	10641408	4005	18899
32	OK	0.046	10235904	8003	7
33	OK	0.078	10321920	5337	3
34	OK	0.046	10301440	4804	3480
35	OK	0.031	10190848	4015	789
36	OK	0.031	10645504	4008	18864
37	OK	0.031	10502144	6004	8904
38	OK	0.046	10235904	5337	3
39	OK	0.031	10326016	5028	8

40	OK	0.031	10219520	4477	786
41	OK	0.015	10739712	4005	18894
42	OK	0.046	10235904	8003	7
43	OK	0.062	10207232	5337	3
44	OK	0.031	10334208	4572	3974
45	OK	0.031	10215424	4015	397
46	OK	0.015	10694656	4008	18884
47	OK	0.031	10428416	6004	8904
48	OK	0.062	10219520	9004	3
49	OK	0.062	10252288	7028	13
50	OK	0.031	10264576	7179	660
51	OK	0.015	10993664	6005	28899
52	OK	0.062	10272768	12003	7
53	OK	0.078	10252288	8004	3
54	OK	0.031	10469376	6752	5678
55	OK	0.046	10215424	6015	1204
56	OK	0.015	10903552	6008	28884
57	OK	0.062	10559488	9004	13904
58	OK	0.062	10334208	9004	3
59	OK	0.046	10219520	7028	8
60	OK	0.031	10203136	6470	506
61	OK	0.015	10977280	6005	28899
62	OK	0.062	10272768	12003	7
63	OK	0.093	10264576	8004	3
64	OK	0.046	10317824	8004	4480
65	OK	0.015	10252288	6016	608
66	OK	0.031	10891264	6008	28884
67	OK	0.046	10665984	9004	13904
68	OK	0.093	10280960	12004	3
69	OK	0.062	10235904	9028	13
70	OK	0.046	10244096	9920	439
71	OK	0.015	11128832	8005	38899
72	OK	0.078	10305536	16003	7
73	OK	0.125	10244096	12004	3
74	OK	0.031	10383360	8728	8376
75	OK	0.015	10305536	8017	1623
76	OK	0.031	11132928	8008	38844
77	OK	0.062	10735616	12004	18904
78	OK	0.078	10231808	12004	3

79	OK	0.078	10313728	9028	17
80	OK	0.062	10280960	10660	350
81	OK	0.031	11239424	8005	38899
82	OK	0.078	10293248	16003	7
83	OK	0.125	10272768	10670	3
84	OK	0.062	10407936	10004	6769
85	OK	0.015	10203136	8022	812
86	OK	0.031	11141120	8008	38884
87	OK	0.078	10723328	12004	18904
88	OK	0.171	10326016	15004	3
89	OK	0.062	10293248	11028	17
90	OK	0.046	10240000	10925	665
91	OK	0.031	11354112	10005	48885
92	OK	0.125	10321920	20003	7
93	OK	0.171	10301440	13337	3
94	OK	0.062	10457088	12504	8256
95	OK	0.031	10256384	10020	1022
96	OK	0.031	11284480	10008	48884
97	OK	0.093	10891264	15004	23904
98	OK	0.109	10293248	15004	3
99	OK	0.062	10219520	11028	17
100	OK	0.046	10338304	11004	498
101	OK	0.031	11321344	10005	48890
102	OK	0.125	10346496	20003	7
103	OK	0.187	10252288	13337	3
104	OK	0.046	10555392	10912	10926
105	OK	0.031	10313728	10015	2042
106	OK	0.031	11288576	10008	48884
107	OK	0.093	10924032	15004	23904

Задача 2. Карта

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

Даже самый последний матрос знает, что мы едем искать сокровища. Не нравится мне всё это!

Капитан Смоллетт

В далеком 1744 году во время долгого плавания в руки капитана Александра Смоллетта попала древняя карта с указанием местонахождения сокровищ. Однако расшифровать ее содержание было не так уж и просто.

Команда Александра Смоллетта догадалась, что сокровища находятся на x шагов восточнее красного креста, однако определить значение числа она не смогла. По возвращению на материк Александр Смоллетт решил обратиться за помощью в расшифровке послания к знакомому мудрецу. Мудрец поведал, что данное послание таит за собой некоторое число. Для вычисления этого числа необходимо было удалить все пробелы между словами, а потом посчитать количество способов вычеркнуть все буквы кроме трех так, чтобы полученное слово из трех букв одинаково читалось слева направо и справа налево.

Александр Смоллетт догадывался, что число, зашифрованное в послании, и есть число x. Однако, вычислить это число у него не получилось.

После смерти капитана карта была безнадежно утеряна до тех пор, пока не оказалась в ваших руках. Вы уже знаете все секреты, осталось только вычислить число x.

Формат входного файла

В единственной строке входного файла дано послание, написанное на карте. Длина послания не превышает $3 * 10^5$. Гарантируется, что послание может содержать только строчные буквы английского алфавита и пробелы. Также гарантируется, что послание не пусто. Послание не может начинаться с пробела или заканчиваться им.

Формат выходного файла

Выведите одно число x — число способов вычеркнуть из послания все буквы кроме трех так, чтобы оставшееся слово одинаково читалось слева направо и справа налево.

Примеры

input.txt	output.txt
treasure	8
you will never find the treasure	146

Исходный код к задаче 2

```
class Lab9_2
{
    public static void Main(string[] args)
    {
        var app = new Lab9_2();
```

```
            app.DoWork(args);
        }

        class S
        {
            public int left, right;

            public ulong Muls { get => (ulong)this.left * (ulong)this.right; }

            public S(int left, int right)
            {
                this.left = left;
                this.right = right;
            }
        }

        private void DoWork(string[] args)
        {
            using (StreamWriter sw = new StreamWriter("output.txt"))
            {
                string[] stdin = File.ReadAllLines("input.txt");
                foreach (var line in stdin)
                {
                    var text = Regex.Replace(line, @"\s", string.Empty);
                    if (text.Length < 3)
                    {
                        sw.WriteLine("0");
                        continue;
                    }

                    var right = text.GroupBy(c => c).ToDictionary(g => g.Key, g => new S(0,
g.Count()));
                    var left = new Dictionary<char, S>();

                    var cc = text[0];
                    var entry = right[cc];
                    left[cc] = entry;
                    entry.left++;
                    entry.right--;

                    var count = text.Length - 1;
                    ulong s = 0;
                    for (int i = 1; i < count; i++)
                    {
                        cc = text[i];
                        entry = right[cc];
                        entry.right--;

                        foreach (var kv in left)
                            s += kv.Value.Muls;

                        entry.left++;
                        left[cc] = entry;
                    }

                    sw.WriteLine(s);
                }
            }
        }
}
```

Бенчмарк к задаче 2

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла

Max		0.218	17068032	300002	18
1	OK	0.046	12845056	10	3
2	OK	0.046	12865536	34	5
3	OK	0.062	12820480	5	3
4	OK	0.046	12840960	6	3
5	OK	0.046	12836864	7	3
6	OK	0.031	12836864	9	4
7	OK	0.046	12845056	7	3
8	OK	0.046	12869632	7	3
9	OK	0.046	12865536	13	4
10	OK	0.031	12849152	202	8
11	OK	0.046	12832768	202	8
12	OK	0.046	12857344	202	8
13	OK	0.046	12873728	202	8
14	OK	0.031	12894208	202	7
15	OK	0.046	12886016	202	7
16	OK	0.046	12869632	202	7
17	OK	0.031	12804096	202	9
18	OK	0.062	12886016	5002	13
19	OK	0.046	12873728	5002	13
20	OK	0.046	12877824	5002	13
21	OK	0.046	12918784	5002	13
22	OK	0.046	12886016	5002	13
23	OK	0.046	12910592	5002	13
24	OK	0.031	12935168	5002	13
25	OK	0.031	12984320	5002	13
26	OK	0.046	12910592	5002	13
27	OK	0.046	13012992	5002	13
28	OK	0.046	12910592	5002	11
29	OK	0.046	12906496	5002	11
30	OK	0.046	12926976	5002	11
31	OK	0.046	12992512	5002	11
32	OK	0.046	12980224	5002	11
33	OK	0.093	15736832	300002	18
34	OK	0.109	15667200	300002	18
35	OK	0.109	15708160	300002	18
36	OK	0.093	15699968	300002	18
37	OK	0.125	15712256	300002	18
38	OK	0.109	15667200	300002	18

39	OK	0.171	15138816	300002	17
40	OK	0.187	15044608	300002	17
41	OK	0.187	15134720	300002	17
42	OK	0.171	15097856	300002	17
43	OK	0.218	17068032	300002	17
44	OK	0.171	16986112	300002	17
45	OK	0.171	17002496	300002	17
46	OK	0.156	16949248	300002	17
47	OK	0.171	16965632	300002	17
48	OK	0.156	15069184	300002	17
49	OK	0.171	15126528	300002	17
50	OK	0.171	15069184	300002	17
51	OK	0.156	15036416	300002	17
52	OK	0.187	15093760	300002	17