دانشکده مهندسی کامپیوتر

طراحی و تحلیل الگوریتم‌ها

تمرین ۷*

اساتید حل تمرین: نگار زین‌العابدین، محمد مصطفی رستم‌خانی
تهیه و تنظیم مستند: مریم سادات هاشمی

استاد درس: سید صالح اعتمادی

نیم‌سال دوم ۱۴۰۰-۱۳۹۹

تلگرام	@N_zeyn @mohammadmostafarostamkhani
نام شاخه	fb_A7
نام پروژه/پوشه/پول ریکوست	A7
مهلت تحویل	۱۴۰۰/۲/۲۵

*تشکر ویژه از اساتید حل‌تمرین مریم سادات هاشمی، بنفشه کریمیان، مهسا سادات رضوی، امیر خاکپور، سهیل رستگار و علی آل‌یاسین که در نیم‌سال دوم سال تحصیلی ۹۷-۹۸ نسخه اول این مجموعه تمرین‌ها را تهیه فرمودند.

توضیحات کلی تمرین

۱. ابتدا مانند تمرین های قبل، یک پروژه به نام A7 بسازید.

۲. کلاس هر سوال را به پروژه‌ی خود اضافه کنید و در قسمت مربوطه کد خود را بنویسید. هر کلاس شامل دو متد اصلی است:

- متد اول: تابع `Solve` است که شما باید الگوریتم خود را برای حل سوال در این متد پیاده سازی کنید.
- متد دوم: تابع `Process` است که مانند تمرین های قبلی در `TestCommon` پیاده سازی شده است. بنابراین با خیال راحت سوال را حل کنید و نگران تابع `Process` نباشید! زیرا تمامی پیاده سازی ها برای شما انجام شده است و نیازی نیست که شما کدی برای آن بزنید.

۳. اگر برای حل سوالی نیاز به تابع های کمکی دارید؛ می توانید در کلاس مربوط به همان سوال تابع تان را اضافه کنید.

اکنون که پیاده سازی شما به پایان رسیده است، نوبت به تست برنامه می رسد. مراحل زیر را انجام دهید.

۱. یک `UnitTest` برای پروژه‌ی خود بسازید.

۲. فولدر `TestData` که در ضمیمه همین فایل قرار دارد را به پروژه‌ی تست خود اضافه کنید.

۳. فایل `GradedTests.cs` را به پروژه‌ی تستی که ساخته اید اضافه کنید.

توجه:
برای اینکه تست شما از بهینه سازی کامپایلر دات نت حداکثر بهره را ببرد زمان تست ها را روی بیلد **Release** امتحان کنید، درغیر اینصورت ممکن است تست های شما در زمان داده شده پاس نشوند.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using A7;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TestCommon;

namespace A7.Tests
{
    [DeploymentItem("TestData", "A7_TestData")]
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q1FindAllOccur()
        {
            Assert.Inconclusive("Not Solved");
            RunTest(new Q1FindAllOccur("TD1"));
        }

        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q2CunstructSuffixArray()
        {
            Assert.Inconclusive("Not Solved");
            RunTest(new Q2CunstructSuffixArray("TD2"));
        }

        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q3PatternMatchingSuffixArray()
        {
            Assert.Inconclusive("Not Solved");
            RunTest(new Q3PatternMatchingSuffixArray("TD3"));
        }

        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("A7", p.Process, p.TestDataName, p.Verifier, VerifyResultWitho
                excludedTestCases: p.ExcludedTestCases);
        }
    }
}
```

۱ پیدا کردن تمام تکرارهای یک الگو در یک رشته [۱]

در این سوال به دنبال پیدا کردن تعداد تکرارهای یک الگو به عنوان زیررشته در رشته‌ی ورودی هستیم. توجه کنید که این زیررشته‌ها می‌توانند با یکدیگر همپوشانی داشته باشند. برای مثال الگوی ATA سه بار در رشته‌ی CGATATATCCATAG تکرار شده است.

در این سوال باید الگوریتمی بنویسید که در یک رشته ورودی تعداد تکرارهای الگوهای داده شده را پیدا کند. در خط اول فایل ورودی رشته متنی که باید الگو را در آن جستجو کنیم وجود دارد و در خط دوم الگویی که به دنبال آن هستیم و باید آن را در رشته متنی جستجو کنیم وجود دارد.

در خروجی باید در هر خط ایندکس‌هایی از متن (با فرض شروع ایندکس گذاری از صفر) که زیر رشته از آنجا شروع شده را برگردانید. در صورتی که الگو در رشته متنی وجود نداشت ۱- را به عنوان خروجی برگردانید.

ورودی نمونه	خروجی نمونه
GT TACG	-1

در این مثال طول الگو از رشته بیشتر است پس الگو در رشته وجود ندارد.

ورودی نمونه	خروجی نمونه
ATATA ATA	0 2

ورودی نمونه	خروجی نمونه
GATATATGCATATACTT ATAT	1 3 9

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TestCommon;

namespace A7
{
    public class Q1FindAllOccur : Processor
    {
        public Q1FindAllOccur(string testDataName) : base(testDataName)
        {
                        this.VerifyResultWithoutOrder = true;
        }

```

[۱] Find All Occurrences of a Pattern in a String

```
        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<String, String, long[]>)Solve, "\n");

        protected virtual long[] Solve(string text, string pattern)
        {
            // write your code here
            throw new NotImplementedException();
        }
    }
}
```

۲ تشکیل Suffix Array برای یک رشته ی طولانی [۲]

هدف این سوال بدست آوردن Suffix Array است اما این بار رشته مورد نظر طول بزرگی دارد. بنابراین الگوریتمی با پیچیدگی زمانی درجه دو نمی‌تواند برای این سوال مناسب باشد و نیازمندید الگوریتمی با پیچیدگی زمانی تقریبا خطی برای این سوال پیاده‌سازی کنید.

در فایل ورودی یک رشته وجود دارد که با حروف T G C A ساخته شده است و با نماد $ پایان می‌یابد.

خروجی Suffix Array برای رشته است که شامل لیستی از اعداد صحیح نشان‌دهنده‌ی ایندکس شروع پسوندهای مرتب شده‌ی رشته هستند.

ورودی نمونه	خروجی نمونه
AAA$	3 2 1 0

Sorted suffixes:
3 $
2 A$
1 AA$
0 AAA$

شکل ۱: نمونه اول

ورودی نمونه	خروجی نمونه
GAC$	3 1 2 0

Sorted suffixes:
3 $
1 AC$
2 C$
0 GAC$

شکل ۲: نمونه دوم

[۲] Construct the Suffix Array of a Long String

ورودی نمونه	خروجی نمونه
GAGAGAGA$	8 7 5 3 1 6 4 2 0

Sorted suffixes:

```
8   $
7   A$
5   AGA$
3   AGAGA$
1   AGAGAGA$
6   GA$
4   GAGA$
2   GAGAGA$
0   GAGAGAGA$
```

شکل ۳: نمونه سوم

ورودی نمونه	خروجی نمونه
AACGATAGCGGTAGA$	15 14 0 1 12 6 4 2 8 13 3 7 9 10 11 5

Sorted suffixes:

```
15   $
14   A$
0    AACGATAGCGGTAGA$
1    ACGATAGCGGTAGA$
12   AGA$
6    AGCGGTAGA$
4    ATAGCGGTAGA$
2    CGATAGCGGTAGA$
8    CGGTAGA$
13   GA$
3    GATAGCGGTAGA$
7    GCGGTAGA$
9    GGTAGA$
10   GTAGA$
11   TAGA$
5    TAGCGGTAGA$
```

شکل ۴: نمونه چهارم

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using TestCommon;

namespace A7
{
    public class Q2CunstructSuffixArray : Processor
    {
        public Q2CunstructSuffixArray(string testDataName) : base(testDataName)
        {
        }

        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<String, long[]>)Solve);

        protected virtual long[] Solve(string text)
        {
            // write your code here
            throw new NotImplementedException();
        }
    }
}
```

۳ تطبیق الگوها با استفاده از Suffix Array [۳]

در این سوال باید الگوریتم تطبیق چندگانه الگوها را با استفاده از Suffix Array پیاده‌سازی کنید.

در این سوال باید تمامی تکرارهای الگوهای داده شده در رشته ورودی را بدست آورید. خط اول از فایل ورودی رشته مورد نظر است که با حروف A T C G ساخته شده است و طول آن بین ۱ تا ۱۰۰۰۰۰ کاراکتر است. در خط بعدی یک عدد صحیح وجود دارد که نشان‌دهنده تعداد الگوها برای یافتن در رشته است. در هر یک از n خط بعدی الگوهای مورد نظر برای پیدا کردن در رشته آمده است.

در فایل خروجی در هر خط تمامی ایندکس هایی از متن که هر یک از الگوها در آنجا به عنوان زیررشته وجود دارند را برگردانید. اگر در یک ایندکس بیش از یک الگو یافت شد تنها یک بار آن را در خروجی چاپ کنید و اگر جوابی نیافتید ۱- را به عنوان خروجی برگردانید.

ورودی نمونه	خروجی نمونه
AAA 1 A	0 1 2

ورودی نمونه	خروجی نمونه
ATA 3 C G C	-1

[۳] Pattern Matching with the Suffix Array

ورودی نمونه	خروجی نمونه
ATATATA 3 ATA C TATAT	4 2 0 1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TestCommon;

namespace A7
{
    public class Q3PatternMatchingSuffixArray : Processor
    {
        public Q3PatternMatchingSuffixArray(string testDataName) : base(testDataName)
        {
            this.VerifyResultWithoutOrder = true;
        }

        public override string Process(string inStr) =>
        TestTools.Process(inStr, (Func<String, long, string[], long[]>)Solve, "\n");

        protected virtual long[] Solve(string text, long n, string[] patterns)
        {
            // write your code here
            throw new NotImplementedException();
        }
    }
}
```