دانشکده مهندسی کامپیوتر

تحلیل و طراحی الگوریتم

امتحان عملی اول

بابک بهکام کیا
محمد خاوری
محمدجواد مهدی تبار
سید صالح اعتمادی

نیم‌سال دوم ۱۴۰۱-۱۴۰۰

ایمیل/تیمز	babak_behkamkia@comp.iust.ac.ir mohammad_khavari@comp.iust.ac.ir m_mehditabar@comp.iust.ac.ir
نام شاخه	fb_E1
نام پروژه/پوشه/پول ریکوست	E1
مهلت تحویل	۲۳ اردیبهشت ساعت ۱۳

توضیحات کلی امتحان

۱. ابتدا مانند تمرین های قبل، یک پروژه به نام E1 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.

سایر نکات

۱. تمام url هایی که موقع امتحان استفاده کردهاید را در فایل کد خود به صورت کامنت اضافه کنید.

۲. استفاده از اسلایدهای درس و کدهایی که «خود شما» برای «این درس» نوشته و در گیت موجود دارید مجاز است. استفاده از هرگونه کد دیگر که یا توسط شما نوشته نشده یا در برای این درس نوشته نشده یا در گیت شما قبلا موجود نبوده مجاز نمیباشد.

۳. استفاده از هر گونه ویدیو مجاز نمیباشد.

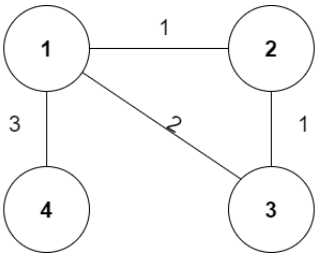
۴. تصویر صفحه نمایش و وبکم شما در کل مدت امتحان بدون وقفه باید توسط نرمافزار FlashBackExpress (یا نرمافزار مشابه) ضبط شده و پس از فشردهسازی برای استاد درس ارسال شود.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TestCommon;

namespace E1.Tests
{

    [DeploymentItem("TestData", "E1_TestData")]
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(600)]
        public void SolveTest_Q1SecondMST()
        {
            RunTest(new Q1SecondMST("TD1"));
        }

        [TestMethod(), Timeout(250)]
        public void SolveTest_Q2SubStrings()
        {
            RunTest(new Q2SubStrings("TD2"));
        }

        [TestMethod(), Timeout(450)]
        public void SolveTest_Q3FindAllOccur()
        {
            RunTest(new Q3LeastLengthString("TD3"));
        }
        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("E1", p.Process, p.TestDataName, p.Verifier,
                VerifyResultWithoutOrder: p.VerifyResultWithoutOrder,
                excludedTestCases: p.ExcludedTestCases);
        }

    }
}
```

۱ ساخت جاده‌ها برای اتصال شهرها [۱]

مایک که دنبال پیدا کردن کمترین هزینه برای وصل کردن جاده‌های شهر به یکدیگر بود . اکنون با وصل کردن جاده‌های مختلف به نتایج متفاوتی دست می‌یابد . او به این فکر افتاد که اگر یک کدام از این جاده‌ها دیگر وجود نداشته‌باشند آیا تاثیری در نتیجه کمترین هزینه او دارد یا خیر؟ به همین دلیل او سعی کرد که یک هزینه جایگزین کمترین هزینه پیدا کند . در واقع او سعی داری دومین کمترین هزینه برای وصل کردن جاده‌های شهر به یکدیگر پیدا کند و او از شما می‌خواهد در این کار به اوکمک کنید .
باید به این نکته هم توجه داشت امکان دارد دومین کمترین هزینه وجود نداشته‌باشد در این صورت باید مقدار -1 را خروجی دهید.

ورودی نمونه	خروجی نمونه
4 1 2 1 4 1 3 2 3 1 1 3 2	6

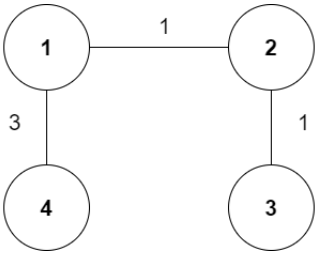


شکل ۱: نمونه اول

کمترین هزینه وصل کردن تمام راس‌ها ۵ می‌باشد و دومین هزینه آن برابر ۶ می‌باشد که با حذف کردن یال ۱ به ۲ این به دست‌می‌آید

ورودی نمونه	خروجی نمونه
4 1 2 1 4 1 3 2 3 1	-1

[۱] second minimum spanning tree



شکل ۲: نمونه دوم

چون به ازای حذف کردن هر کدام از یال دیگر نمی‌توان تمام شهر ها را به هم متصل کرد پس خروجی برابر -1 می‌باشد.

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


namespace E1
{
    public class Q1SecondMST : Processor
    {
        public Q1SecondMST(string testDataName) : base(testDataName)
        {
        }

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


        public long Solve(long nodeCount, long[][] edges)
        {
                throw new NotImplementedException();
        }

    }
}
```

راهنمایی: می‌توانید با استفاده از الگوریتم $kruksal$ ابتدا minimum spanning tree را پیدا کنید سپس یک spanning tree جدید پیدا کنید که تنها در یک یال به minimum spanning tree تفاوت داشته‌باشد و بعد آن $minimum$ باشد.

۲ Unique Substrings

در این سوال به شما یک رشته ژنوم و طول آن رشته داده خواهد شد. وظیفه شما این است که تعداد تمام substring های یکتا موجود در این رشته را پیدا کنید

ورودی نمونه	خروجی نمونه
ACACA	9

A, C, AC, CA, ACA, CAC, ACAC, CACA, ACACA

توجه:

این سوال را باید با suffix array حل کنید، در غیر اینصورت حتی اگر تمام تست ها را pass بکنید هم نمره کامل نخواهید گرفت.

```
using System;
using System.Collections.Generic;
using TestCommon;

namespace E1
{
    public class Q2UniqueSubStrings : Processor
    {
        public Q2UniqueSubStrings(string testDataName) : base(testDataName) { }

        public override string Process(string inStr) =>
        E1Processors.ProcessQ2UniqueSubStrings(inStr, Solve);

        public virtual long Solve(long n, String text)
        {
            // write your code here
            throw new NotImplementedException();
        }

    }
}
```

۳ مسئله کوتاه‌ترین ابر رشته [۲]

در این مسئله یک رشته‌ی ورودی t داریم. می‌خواهیم کوتاه ترین رشته‌ای را تولید کنیم که شامل n بار تکرار رشته‌ی s باشد. به عبارتی می‌خواهیم کوتاه‌ترین رشته‌ی S را بیابیم به صورتی که شامل n زیررشته‌ی t باشد.

ورودی: در خط اول عدد n و در خط بعدی رشته‌ی t

خروجی: طول رشته‌ی S (کوتاه ترین اَبَررشته که شامل n زیررشته‌ی t باشد)

مثال ۱)

ورودی نمونه	خروجی نمونه
2 t	2

مثال ۲)

ورودی نمونه	خروجی نمونه
3 gorego	14

توضیح : goregoregorego

مثال ۳)

ورودی نمونه	خروجی نمونه
2 ababab	8

توضیح : abababab

مثال ۴)

ورودی نمونه	خروجی نمونه
4 aagctgaagct	29

توضیح : aagctgaagctgaagctgaagct

[۲] shortest superstring

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

namespace E1
{
    public class Q3LeastLengthString : Processor
    {
        public Q3LeastLengthString(string testDataName) : base(testDataName)
        {
                        this.VerifyResultWithoutOrder = true;
        }

        public override string Process(string inStr) =>
        E1Processors.ProcessQ3FindAllOccur(inStr, Solve);

        public long Solve(string text, long k)
        {
            throw new NotImplementedException();
        }
    }
}
```