دانشکده مهندسی کامپیوتر

طراحی و تحلیل الگوریتم‌ها

تمرین ۲*

زهرا حسینی
سهراب نمازی
سید صالح اعتمادی

نیم‌سال دوم ۰۰-۹۹

تلگرام	@arrhhaz @Sohlaub
نام شاخه	fb_A2
نام پروژه/پوشه/پول ریکوست	A2
مهلت تحویل	۹۹/۱۲/۲۳

*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 A2 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 درکلاس 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.

در این سری از تمرین، علاوه برفایل ها با پسوند txt که تست کیس های سوالات بودند و شما از آن ها برای تست کدتان استفاده می کردید، فایل هایی با پسوند webgraphviz نیز در پوشه TestData وجود دارد. شما با استفاده از این فایل ها می توانید گراف های کوچکتر از ۱۰۰ گره را به صورت تصویری سایت زیر مشاهده کنید.

http://www.webgraphviz.com

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

namespace A2.Tests
{
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q1ShortestPath()
        {
            RunTest(new Q1ShortestPath("TD1"));
        }

        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q2BipartiteGraph()
        {
            RunTest(new Q2BipartiteGraph("TD2"));
        }

        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("A2", p.Process, p.TestDataName, p.Verifier,
                VerifyResultWithoutOrder: p.VerifyResultWithoutOrder,
                excludedTestCases: p.ExcludedTestCases);
        }
    }
}
```

۱ محاسبه حداقل تعداد قسمت های پرواز

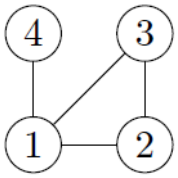
فرض کنید که شما می خواهید حداقل تعداد بخش های پرواز لازم برای رفتن از یک شهر به شهر دیگر را محاسبه کنید. برای این منظور یک گراف غیر چهت دار در نظر بگیرید که: رأس ها شهرها را نشان می دهند، و اگر پروازی بین دو شهر وجود داشته باشد، یک یال بین راس های متناظر با این شهر ها قرار داده می شود. سپس، کافی است که کوتاه ترین مسیر از شهر مورد نظر به سوی شهر دیگر را در این گراف غیر جهت دار پیدا کنید.

بنابراین شما باید الگوریتمی بنویسید که کوتاه ترین مسیر موجود بین راس u و راس v را در گراف غیر جهت دار با n راس و m یال را محاسبه کند. (یعنی حداقل تعداد لبه ها در مسیر از u به v).

خط اول فایل ورودی، تعداد راس های گراف را نشان می دهد و هر یک از خطوط بعدی شامل دو عدد می باشد که بیانگر وجود یک یال بین این دو عدد(راس) است. در خط آخر هم دو عدد وجود دارد که باید کوتاه ترین مسیر بین عدد(راس) اول و عدد(راس)دوم را بدست آورید. اگر کوتاه ترین مسیر بین دو راس مورد نظر وجود داشته باشد، در فایل خروجی طول مسیر قرار می گیرد ودر غیر این صورت عدد ۱- درج می شود.

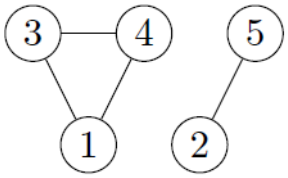
نمونه ۱

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



نمونه ۲

ورودی نمونه	خروجی نمونه
5 5 2 1 3 3 4 1 4 3 5	-1




```
using System;
using TestCommon;

namespace A2
{
    public class Q1ShortestPath : Processor
    {
        public Q1ShortestPath(string testDataName) : base(testDataName) { }

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

        public long Solve(long NodeCount, long[][] edges,
                          long StartNode,  long EndNode)
        {
            throw new NotImplementedException();
        }
    }
}
```

۲ بررسی دو طرفه بودن یک گراف

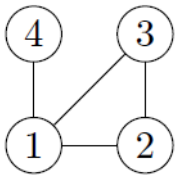
یک گراف غیر جهت دار دو طرفه نامیده می شود اگر بتوان رأس های آن را به دو بخش تقسیم کرد به طوری که هر لبه گراف به رأس ها از قسمت های مختلف متصل شود. به زبان ساده تر، یک گراف دو طرفه است، اگر رأس های آن را بتوان با دو رنگ (به عنوان مثال سیاه و سفید) رنگی کرد به طوری که نقاط انتهایی هر لبه رنگ های متفاوتی داشته باشند. گراف های دو طرفه به طور طبیعی در برنامه های کاربردی بوجود می آیند که در آن گراف برای مدل سازی ارتباط بین اشیاء دو نوع مختلف (مثلا پسر و دختر، یا دانش آموز و خوابگاه) استفاده می شود.

در این سوال، یک گراف غیر جهت دار با n راس و m یال داده می شود و شما باید برنامه ای بنویسید که مشخص کند آیا این گراف دو طرفه است یا خیر.

خط اول فایل ورودی شامل یک عدد است که نشان دهنده ی تعداد راس های گراف غیر جهت دار است و در هر یک از خطوط بعدی دو عدد وجود دارد که بیانگر وجود یک یال یا لبه بین عدد(راس) اول و عدد(راس) دوم است. اگر گراف داده شده دو طرفه باشد، خروجی یک و در غیر این صورت صفر خواهد بود.

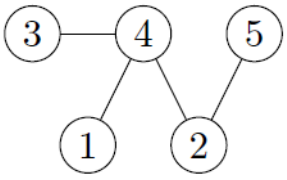
نمونه ۱

ورودی نمونه	خروجی نمونه
4 1 2 4 1 2 3 3 1 1 4	0



نمونه ۲

ورودی نمونه	خروجی نمونه
5 5 2 4 2 3 4 1 4	1



```
using System;
using TestCommon;

namespace A2
{
    public class Q2BipartiteGraph : Processor
    {
        public Q2BipartiteGraph(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();
        }
    }
}
```