دانشکده مهندسی کامپیوتر

طراحی و تحلیل الگوریتم‌ها

امتحان عملی اول

تهیه و تنظیم مستند: مبین داریوش همدانی

استاد درس: سید صالح اعتمادی

نیم‌سال دوم ۱۴۰۰-۱۳۹۹

تلگرام	@Nibom
نام شاخه	fb_E1
نام پروژه/پوشه/پول ریکوست	E1
مهلت ارسال	۲۶ فروردین ساعت ۱ ب.ظ.

توضیحات کلی تمرین

امتحان عملی شما ۲ سوال دارد. این سوالات به ترتیب سختی از آسان به سخت مرتب شده اند.

۱. ابتدا مانند تمرین‌های 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.

۳. اگر برای حل سوالی نیاز به تابع‌های کمکی دارید؛ می‌توانید در کلاس مربوط به همان سوال تابع‌تان را اضافه کنید.

اکنون که پیاده‌سازی شما به پایان رسیده است، نوبت به تست برنامه می‌رسد. 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 ساخته‌اید اضافه کنید.

۴. بعد از درست کردن پروژه ها و اضافه کردن فایلها قبل از اینکه هیچ کدی بزنید لازم است که فایل ها را به گیت add/commit/push کنید و در visualstudio.com برای بردن این شاخه به مستر یک پول ریکوست به نام E۱ درست کنید. در صورت صحیح انجام ندادن این بخش، نمره امتحان را از دست می‌دهید.

۵. در انتهای زمان آزمون باید شاخه امتحان شما روی شاخه master پروژه مرج شده باشد.

به موارد زیر توجه کنید!

۱. جستجو در اینترنت و استفاده از منابع موجود در آن برای پاسخگویی به سوالات مجاز نیست. تنها منابع قابل استفاده در طول امتحان اسلایدهای درس و کدهایی است که خود شما برای تمرین‌های درس زده‌اید و در ریپازیتوری شما موجود است. استفاده از این کدها بلا مانع است.

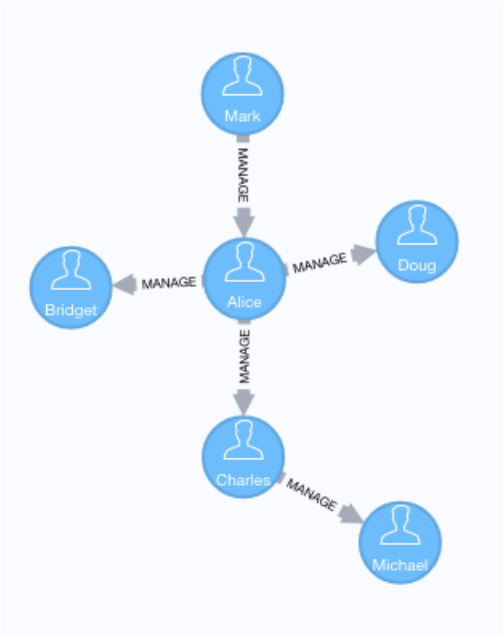
۲. اگر سوالی داشتید می‌توانید از استاد درس یا استاد حل تمرین در محیط Teams بپرسید.

۳. صدا و صفحه نمایش شما باید از طریق نرم افزار Flashback recorder به طور کامل از ابتدا تا انتهای امتحان ضبط و ذخیره شود.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using TestCommon;
using E1;

namespace E1.Tests
{
    [DeploymentItem("TestData", "E1_TestData")]
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(10000)]
        public void SolveTest_Q1BetweennessTest()
        {
            RunTest(new Q1Betweenness("TD1"));

        }

        [TestMethod(), Timeout(5000)]
        [DeploymentItem("TestData", "E1_TestData")]
        public void SolveTest_Q2RoadReconstruction()
        {
            RunTest(new Q2RoadReconstruction("TD2"));
        }


        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("E1", p.Process, p.TestDataName, p.Verifier,
                VerifyResultWithoutOrder: p.VerifyResultWithoutOrder,
                excludedTestCases: p.ExcludedTestCases);
        }
    }
}
```

۱ Betweenness Centrality (آسان - ۶۰ نمره)

در نظریه گراف، یک مفهوم به نام Betweenness Centrality وجود دارد که به معنای اندازه گیری مرکزیت در گراف، بر پایه کوتاهترین مسیر است. برای هر جفت رأس در یک گراف متصل، حداقل یک مسیر کوتاه بین رأس ها وجود دارد به طوری که یا تعداد یال هایی که مسیر از طریق آن عبور می کند (برای گراف های بدون وزن) و یا مجموع وزن های یال ها (برای گراف های وزن دار) به حداقل برسد. Betweenness Centrality برای هر گره برابر است با تعداد کوتاه ترین مسیرهایی که از آن گره عبور می کند.
برای مثال به گراف زیر توجه کنید.



Alice مهمترین ارتباط در این گراف است. اگر Alice حذف شود، تمام ارتباطات در گراف قطع می شود. این باعث می شود Alice مهم باشد.
در گراف بالا اگر بخواهیم betweenness Centrality را برای همه ی راس های این گراف مشخص کنیم، ابتدا باید کوتاه ترین مسیر موجود بین هر دو گره مجزا را پیدا کنیم که برای گراف بالا به صورت زیر خواهد بود:

1. Mark -> Alice
2. Mark -> Alice -> Charlrs
3. Mark -> Alice -> Doug
4. Mark -> Alice -> Bridget
5. Mark -> Alice -> Charlrs -> Micheal
6. Alice -> Charlrs
7. Alice -> Charlrs -> Micheal

حال باید ببنیم برای هر گره، چه تعداد از مسیرهای بالا از آن گره عبور کرده است تا Betweenness Centrality هر گره از گراف بدست آید.همانطور که پیدا ست ۴ مسیر ۲، ۳، ۴ و ۵ از گره Alice عبور کرده است. دو مسیر ۵ و ۷ هم از گره Charles عبور کرده است و از باقی گره ها مسیری عبور نکرده است. **دقت کنید که گره های آغازی و پایانی مسیر به عنوان گره هایی که مسیر از آن ها عبور می کند، در نظر گرفته نشده است.** پس به صورت خلاصه Betweenness Centrality هر گره را می توان در جدول زیر نمایش داد:

Name	Weight Centrality
Alice	4
Charles	2
Bridget	0
Michael	0
Doug	0
Mark	0

Table 1: Betweenness Centrality in Graph

مشاهده می کنید که Alice واسطه اصلی این شبکه است و بعد از آن Charles . دیگران هیچ تاثیری ندارند، زیرا همه کوتاه ترین مسیرها بین جفت افراد از طریق Alice یا Charles گذر می کنند.

در این سوال به شما یک گراف جهت دار بدون وزن داده می شود که در خط اول فایل ورودی عدد n قرار دارد که نشان دهنده ی تعداد گره های گراف است و در هر یک از خطوط بعدی دو گره u و v وجود دارد که نشان دهنده ی یک یال از گره u به گره v است. شما باید برنامه ای بنویسید که Betweenness Centrality را برای هر یک از گره های این گراف بدست بیاورد و به ترتیب برای گره ۱ تا n در فایل خروجی نمایش دهد.

توجه:چنانچه بیش از یک کوتاه‌ترین مسیر وجود داشته باشد، مسیری باید در نظر گرفته شود، که از نود با شماره بیشتر عبور می‌کند.

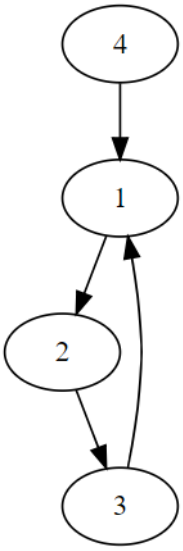
نمونه ۱

ورودی:

```
4
1 2
4 1
2 3
3 1
```

خروجی:

```
3 2 1 0
```



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

namespace E1
{
    public class Q1Betweenness : Processor
    {
        public Q1Betweenness(string testDataName) : base(testDataName)
        {
            //this.ExcludeTestCaseRangeInclusive(2, 50);
        }

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


        public long[] Solve(long NodeCount, long[][] edges)
        {
            return new long[] { };
        }
    }
}
```

۲ Road Reconstruction (سخت - ۴۰ نمره)

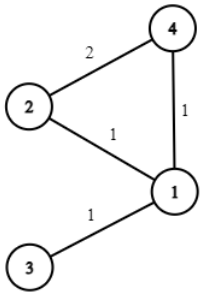
زمانی در کشور Neverland تعداد n شهر وجود داشت که با n جاده دو طرفه وزن دار به هم متصل شده بودند، به طوری که از هر شهری می‌توانستیم به شهر دیگر سفر کنیم. متاسفانه در اثر حمله قوم هوک به این سرزمین تمامی این جاده ها در حال حاضر از بین رفته اند و تنها اطلاعاتی که از شبکه راه ها باقی مانده است جدولی به شکل یک ماتریس $n * n$ است که طول کوتاهترین مسیر بین هر دو شهر در آن مشخص شده است. از شما خواسته شده که با استفاده از این ماتریس شبکه راه های کشور را با احداث n جاده جدید بازسازی کنید به طوری که طول کوتاهترین مسیر بین هر دو شهر مختلف، مطابق با جدول موجود باشد.
دقت کنید که ممکن است چندین جواب برای یک جدول وجود داشته باشد. در این صورت یکی از جواب ها را به دلخواه برگردانید.
ورودی:

```
4
0 1 1 1
1 0 2 2
1 2 0 2
1 2 2 0
```

خروجی:

```
2 1 1
3 1 1
4 1 1
```

```
2 4 2
```



شکل ۱: گراف خروجی نمونه

۱.۲ راهنمایی:

سعی کنید از ویژگی Cut Property که در درس با آن آشنا شدیم و الگوریتم های مرتبط با آن برای حل این سوال استفاده کنید.

Cut property

Let $X \subseteq E$ be a part of a MST of $G(V, E)$, $S \subseteq V$ be such that no edge of X crosses between S and $V - S$, and $e \in E$ be a lightest edge across this partition. Then $X + \{e\}$ is a part of some MST.

شکل ۲: Cut Property

```
using System;
using System.Linq;
using TestCommon;

namespace E1
{
    public class Q2RoadReconstruction : Processor
    {
        public Q2RoadReconstruction(string testDataName) : base(testDataName)
        {
        }

        public override Action<string, string> Verifier => RoadReconstructionVerifier.Verify;

        public override string Process(string inStr) {
            long count;
```

```
            long[][] data;
            TestTools.ParseGraph(inStr, out count, out data);
            return string.Join("\n", Solve(count, data).Select(edge => string.Join(" ", edge)));
        }

        // returns n different edges in the form of {u, v, weight}
        public long[][] Solve(long n, long[][] distance)
        {
            return null;
        }
    }
}
```