دانشکده مهندسی کامپیوتر

طراحی و تحلیل الگوریتم‌ها

امتحان عملی ۲

اساتید حل تمرین: بابک بهکام کیا، محمدجواد مهدی‌تبار
استاد درس: سید صالح اعتمادی

نیم‌سال دوم ۱۴۰۰-۱۴۰۱

نام شاخه	fb_E2
نام پروژه/پوشه/پول ریکوست	E2
مدت امتحان	۴ ساعت

توضیحات کلی امتحان

۱. شیوه ساخت پروژه/شاخه/گیت/... مانند تمرین‌ها و امتحان عملی قبل است.

۲. استفاده از شبکه/اینترنت فقط و فقط برای درست کردن PullRequest مجاز است. هر گونه استفاده دیگر حتی جستجوی syntax جایز نیست. چنانچه اشکالی دارید با استاد/حل‌تمرین در میان بگذارید.

۳. 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 برای این درس نوشته نشده یا در گیت شما قبلا موجود نبوده مجاز نمی‌باشد.

۴. استفاده از هر گونه ویدیو مجاز نمی‌باشد.

۵. تصویر صفحه نمایش شما در کل مدت امتحان بدون وقفه باید توسط نرم‌افزار FlashBackExpress (یا نرم‌افزار مشابه) ضبط شده و پس از امتحان در فلش دیسک به استاد تحویل داده شود.

توجه:
خروجی سوال دوم شما مانند تمرین ۱۰ توسط یک SAT-Solver راست‌آزمایی می‌شود. لازم است مراحل پیش‌نیاز تمرین ۱۰ انجام شده باشند. همچنین برای پردازش داده‌های تست فایل `TestCommon.cs` بروز شده و در اختیار شما قرار گرفته است. لازم است این فایل بروز رسانی شود.

```
using TestCommon;

namespace E2.Tests;

[DeploymentItem("TestData", "E2_TestData")]
[TestClass()]
public class GradedTests
{
    [TestMethod(), Timeout(20000)]
    public void SolveTest_Q1LatinSquareSAT()
    {
        Assert.Inconclusive();
        RunTest(new Q1LatinSquareSAT("TD1"));
    }

    [TestMethod(), Timeout(7000)]
    public void SolveTest_Q2MaxflowVertexCapacity()
    {
        Assert.Inconclusive();
        RunTest(new Q2MaxflowVertexCapacity("TD2"));
    }

    public static void RunTest(Processor p)
    {
        TestTools.RunLocalTest("E2", p.Process, p.TestDataName, p.Verifier, VerifyResultWithoutOr
            excludedTestCases: p.ExcludedTestCases);
    }

}
```

۱ سوال اول `Q1LatinSquareSAT`

در این سوال لازم است معادل `CNF` امکان حل مساله مربع لاتین را با فرمت تمرین دهم برگردانید. به این شکل که خط اول شامل دو عدد می‌باشد. اولی تعداد عبارت‌ها، c، و دومی تعداد متغیرها، v، می‌باشد. در خطوط بعدی، هر عبادت در یک سطر گذاشته می‌شود. هر عبارت شامل یک یا بیشتر عدد بین 1 و v یا بین $-v$ و -1 می‌باشد. عدد منفی n معادل نقیض x_n یا $\bar{x}_n$ و اعداد مثبت برابر خود متغیر x هستند. متغیرها/اعداد یک عبارت با کارکتر فاصله از هم جدا می‌شوند و عبارات با خط جدید ($\backslash n$) از هم جدا می‌شوند.

۱.۱ معرفی مربع لاتین (`Latin Square`)

مربع لاتین [۱] با اندازه n، یک ماتریس مربعی با ابعاد n است که هر سطر و هر ستون آن شامل اعداد 0 تا $n-1$ می‌باشد. لازم است هر سطر و هر ستون تمام اعداد را داشته باشند و هیچ عدد تکراری در سطر یا ستون نداشته باشد. برای نمونه جدول ۱ را ببینید.

[۱] برای مشخص کردن پیاده‌سازی در امتحان این تعریف با تعریف دقیق و کلی مربع لاتین مقداری تفاوت دارد.

۳	۲	۱	۰
۲	۳	۰	۱
۱	۰	۳	۲
۰	۱	۲	۳

جدول ۱: مربع لاتین با ابعاد ۴

۲.۱ مساله مربع لاتین

مساله مربع لاتین پیدا کردن امکان کامل کردن یک ماتریس مربعی نیمه پر باابعاد $n \times n$ و مقادیر 0 تا $n-1$ است بطوریکه ماتریس نهایی یک مربع لاتین باشد. به عنوان مثال جدول ۲ را ببینید.

0	2	1
1		0
	1	2

2	0	
1		0
	1	

جدول ۲: مساله مربع لاتین: مساله سمت راست قابل حل است ولی سمت چپ قابل حل نیست.

۳.۱ پیاده‌سازی

برای این مساله اول متد `Solve` عبارت `CNF` معادل را باید برگرداند که توسط `SatVerifier` حل شده و با جواب داده تست مقایسه می‌شود.
کتابخانه `TestCommon` داده تست را پردازش کرده و ماتریس ورودی را به صورت یک آرایه دوبعدی از نوع `int?` به متد `Solve` به عنوان پارامتر ورودی پاس می‌دهد. متغیر `int dim` نیز بعد ماتریس را مشخص می‌کند.

```
۱ public string Solve(int dim, int?[,] square)
۲ {
۳     throw new NotImplementedException();
۴ }
```

اگر با نوع داده‌ای `int?` آشنا نیستید، `int?` مخفف است برای `Nullable<int>`
که دو مشخصه مورد استفاده دارد: `.HasValue` و `.Value` برای خانه‌های خالی ماتریس `square.HasValue` برابر `false` می‌باشد. مثلا اگر خانه `[2,3]` مقدار نداشته باشد، `square[2,3].HasValue` برابر `false` خواهد بود و درصورتی که مقدار داشته باشد، مقدار آن در `.Value` است.
با توجه به اینکه فایل‌های تست توسط `TestCommon` پردازش می‌شود. نیازی به دانستن فرمت فایل‌های ورودی نیست. فقط جهت اطلاع: ماتریس‌های نیمه پر به عنوان ورودی مساله مربع لاتین در فایل‌های تست به این صورت ذخیره شده‌اند که در خط اول بعد ماتریس و در خطوط بعدی محل‌های خالی کاراکتر . و بقیه ماتریس با اعداد متناظر پر شده است.

```
5
3 4 1 . .
. . . 0 .
. . 2 . .
. 2 . 4 1
. 0 3 . 4
```

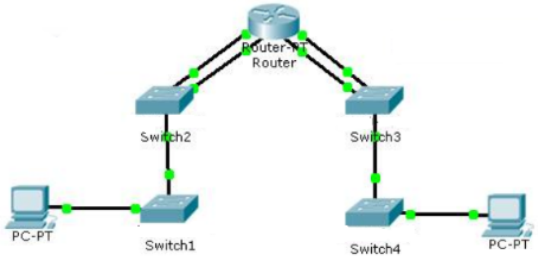
۴.۱ محدویت‌ها

برای دریافت نمره کامل لازم است در ۲۰ ثانیه تمام ۵۴ تست کیس، پاس بشوند.

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

namespace E2
{
    public class Q1LatinSquareSAT : Processor
    {
        public Q1LatinSquareSAT(string testDataName) : base(testDataName)
        {
        }

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

        public override Action<string, string> Verifier =>
            TestTools.SatVerifier;


        public virtual string Solve(int dim, int?[,] square)
        {
            throw new NotImplementedException();
        }
    }
}
```

۲ جریان بیشینه با محدودیت ظرفیت در گره‌ها

مساله پیدا کردن جریان بیشینه [۲] را با در نظر گرفتن ظرفیت برای یال‌های شبکه قبلا دیده و حل کرده‌ایم. در مساله‌های واقعی معمولا گره‌ها هم داری ظرفیت می‌باشند. مثلا در تقاطع خیابان‌ها بستگی به وجود چراغ خطر یا نه، خود تقاطع یک ظرفیت دارد که ممکن است از ظرفیت خیابان‌های ورودی به تقاطع کمتر بوده و باعث ایجاد ترافیک شود.
همچنین در شبکه های کامپیوتری برای اتصال به اینترنت و ارسال یا دریافت داده در بستر آن ، داده های ما از یک سری گره‌ها به نام router و لینک های بین آن ها عبور می کنند . شکل زیر تصویر کوچکی است از مسیری که داده ها از آن عبور می کنند .



[۲] MaxFlow

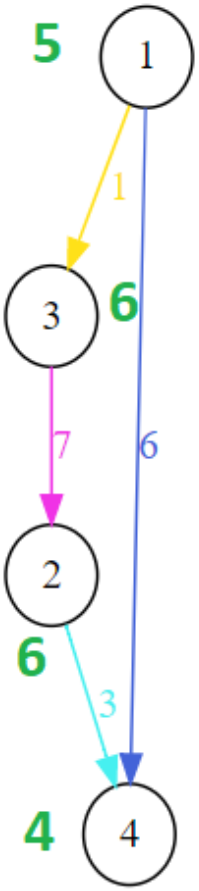
می دانیم هر کدام از گره‌ها یا router/switch هایی که در مسیر قرار دارند در هر ثانیه حداکثر مقدار مشخصی داده را می توانند از خود عبور دهند . همچنین لینک‌ها یا یال‌های مابین switch / router نیز ظرفیت یا Bandwidth مشخص دارند .(حداکثر بیت قابل عبور در هر ثانیه از آن مشخص است). در این سوال مشخصات شبکه (به صورت یک گراف جهت دار) به شما داده می شود که یال‌ها و گره‌ها هر کدام وزن مخصوص به خود را دارند که در واقع نشان دهنده ماکزیمم بیت دیتایی است که قادر به عبور آن در واحد زمان هستند . شما باید برنامه ای بنویسید که به ازای هر دو گره در شبکه حداکثر جریان بین دو گره را محاسبه کند. دقت کنید که ظرفیت گره مبدا و مقصد نیز باید در نظر گرفته شوند.

فرمت ورودی: خط اول به ترتیب تعداد گره‌ها v و تعداد یال‌ها e را مشخص می‌کند. سپس e خط بعدی یال‌ها را بصورت گره مبدا، گره مقصد و ظرفیت مشخص می‌کند. در خط بعد ظرفیت v گره با فاصله از هم جدا شده‌اند. نهایتا در خط آخر گره مبدا و مقصد مشخص شده‌اند. البته فایل‌های ورودی برای شما پردازش شده و بصورت متغیرهای ورودی به متد Solve شما پاس داده می‌شوند. برای راحتی شما برای شبکه‌های با تعداد نود پایین یک فایل با پسوند webgraphviz نیز گذاشته شده که می‌توانید از سایت `http://www.webgraphviz.com` برای نمایش شبکه استفاده کنید. استفاده از این سایت مجاز می‌باشد.

فرمت خروجی: یک عدد می‌باشد که نشان‌دهنده حداکثر ظرفیت بین گره مبدا و مقصد می‌باشد.

ورودی نمونه	خروجی نمونه
4 4 3 2 7 2 4 3 1 3 1 1 4 6 5 6 6 4 1 4	4

با اندکی دقت در شکل زیر می‌توان دریافت که به علت محدودیت ظرفیت گره مقصد (گره چهار) حداکثر ظرفیت شبکه بین گره ۱ و ۴ معادل ۴ می‌باشد.




```
using TestCommon;

namespace E2;

public class Q2MaxflowVertexCapacity : Processor
{
    public Q2MaxflowVertexCapacity(string testDataName) : base(testDataName)
    {
    }
    public override string Process(string inStr) =>
        TestTools.Process(inStr, (Func<long, long, long[][], long[], long, long, long>)Solve);

    public virtual long Solve(long nodeCount, long edgeCount, long[][] edges, long[] nodeWeight
        , long startNode, long endNode)
    {
        throw new NotImplementedException();
    }
}
```