دانشکده مهندسی کامپیوتر

طراحی و تحلیل الگوریتم‌ها

تمرین ۸*

اساتید حل تمرین: غزاله محمودی، محمد جواد میرشکاری
تهیه و تنظیم مستند: مریم سادات هاشمی

استاد درس: سید صالح اعتمادی

نیم‌سال دوم ۹۹-۹۸

تلگرام	@mj_haghighi @ghazale_mahmoodi
نام شاخه	fb_A8
نام پروژه/پوشه/پول ریکوست	A8
مهلت تحویل	۱۳۹۹/۰۲/۲۰

*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 A8 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 A8;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TestCommon;

namespace A8.Tests
{
    [DeploymentItem("TestData", "A8_TestData")]
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q1Evaquating()
        {
            RunTest(new Q1Evaquating("TD1"));
        }

        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q2Airlines()
        {
            RunTest(new Q2Airlines("TD2"));
        }

        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q3Stocks()
        {
            RunTest(new Q3Stocks("TD3"));
        }

        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("A8", p.Process, p.TestDataName, p.Verifier, VerifyResultWitho
                excludedTestCases: p.ExcludedTestCases);
        }

    }
}
```

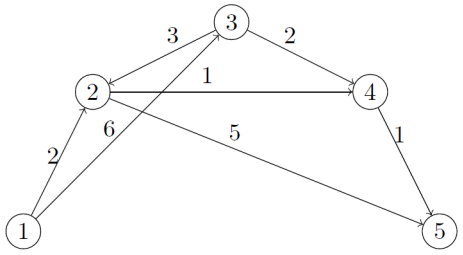
۱ تخلیه مردم [۱]

در این سوال شما باید مقدار Max Flow برای گرافی که در ورودی دریافت می‌کنید را محاسبه کنید. و خلاصه داستان سوال به این صورت است که در اثر وقوع یک تورنادو سکنه شهر اول باید به سمت پایتخت تخلیه بشوند.

در خط اول ورودی شما دو عدد n و m را دریافت می‌کنید که عدد اول نشان‌دهنده تعداد راس‌های گراف است و عدد دوم تعداد یال‌های گراف را توضیح می‌دهد. در هر یک از m خط بعدی شما به ترتیب سه عدد u و v و c را دریافت می‌کنید که هر خط نشان‌دهنده یک یال جهت‌دار است که از راس u به راس v با ظرفیت c می‌رود. شما باید مقدار Max Flow را از شهر ۱ به شهر n حساب کنید.

توجه کنید که امکان دارد چند یال از راس u به v داشته باشیم و همچنین امکان دارد که هم یالی از u به v داشته باشیم و در یالی جداگانه از راس v به u را هم داشته باشیم. برای مشاهده‌ی محدودیت‌های ورودی می‌توانید به منبع انگلیسی سوال مراجعه کنید.

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

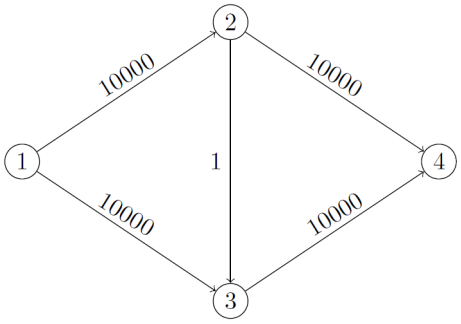


شکل ۱: نمونه اول

در مثال بالا جریان دو واحد از مسیر ۱-۲-۵ ، سه واحد از مسیر ۱-۳-۲-۵ و یک واحد از مسیر ۱-۳-۴-۵ می‌توانیم داشته باشیم.

ورودی نمونه	خروجی نمونه
4 5 1 2 10000 1 3 10000 2 3 1 3 4 10000 2 4 10000	20000

[۱] Evacuating People



شکل ۲: نمونه دوم

ر مثال بالا جریان ۱۰۰۰۰ واحد از مسیر ۱-۲-۴ و ۱۰۰۰۰ واحد از مسیر ۱-۳-۴ می‌توانیم داشته باشیم. توجه کنید که در مثال بالا اگر تنها الگوریتم Ford-Fulcerson را پیاده‌سازی کنید کافی نیست و لازم است آن را ارتقا بدید.

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

namespace A8
{
    public class Q1Evaquating : Processor
    {
        public Q1Evaquating(string testDataName) : base(testDataName) { }

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

        public virtual long Solve(long nodeCount, long edgeCount, long[][] edges)
        {
            // write your code here
            throw new NotImplementedException();
        }
    }
}
```

۲ اختصاص خدمه هواپیمایی به پروازها [۲]

در این سوال شما وظیفه دارید که maximum-matching را در گرافی دو بخشی که در ورودی به شما داده می شود؛ پیدا کنید.

در خط اول ورودی دو عدد n ، m را دریافت می‌کنید که به ترتیب تعداد راس‌های در هر یک از بخش‌های گراف ما هستند. سپس در n خط بعدی در هر یک از خطوط m عدد دریافت می‌کنید که اگر درایه امj خط امi برابر یک بود نشان می‌دهد که راس i از بخش اول به راس j از بخش دوم یال دارد. درواقع شما ماتریس مجاورت این گراف دو بخشی را دریافت می‌کنید.

[۲] Assigning Airline Crews to Flights

در تنها خط خروجی شما باید n خط چاپ شود که عدد امi نشان می‌دهد که راس i از بخش اول به کدام راس match شده‌است. در صورتی که راس i در maximum-matching شما نبود؛ عدد امi خروجی شما باید ۱- باشد. توجه کنید که این سوال برای هر تست ممکن است چند جواب داشته باشد و شما کافیست که تنها یکی از جواب‌هارا چاپ کنید.

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

در مثال بالا راس سوم از بخش اول به هیچ راسی از بخش دوم match نشده است.

ورودی نمونه	خروجی نمونه
2 2 1 1 1 0	2 1

در مثال بالا یک matching کامل رخ داده است ، یعنی هیچ راسی نیست که match نشده باشد.

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

namespace A8
{
    public class Q2Airlines : Processor
    {
        public Q2Airlines(string testDataName) : base(testDataName) { }

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

        public virtual long[] Solve(long flightCount, long crewCount, long[][] info)
        {
            // write your code here
            throw new NotImplementedException();
        }
    }
}
```

۳ نمودار سهام [۳]

وقتی که شما وارد این فصل (Algorithms Advanced) می‌شوید باید انتظار سوال‌های سخت رو هم داشته باشید. این سوال سخت‌ترین تمرین این بخش هست که در آن شما با پیاده‌سازی یک الگوریتم باید حداقل تعداد صفحات مختصات را پیدا کنید که به وسیله آن‌ها بتوان همه‌ی نمودارهایی که در ورودی تحویل داده می‌شود را نمایش داد به صورتی که هیچ دو نموداری که در یک صفحه قرار می‌گیرند با یکدیگر تداخل نداشته‌باشند.

در خط اول ورودی شما به ترتیب اعداد n و k را دریافت می‌کنید . در n خط بعدی شما در هر خط k عدد دریافت می‌کنید که این اعداد نشان‌دهنده قیمت n سهام در k نقطه زمانی است . با اعداد هر خط یک نمودار خطی تشکیل می‌دهیم و می‌خواهیم این نمودارها را در صفحاتی کنار هم قرار بدیم که در یک صفحه هیچ دو نموداری با یکدیگر برخورد نداشته باشند. شما باید تعداد صفحات لازم را پیدا کنید و عدد آن را در تنها خط خروجی چاپ کنید.

ورودی نمونه	خروجی نمونه
3 3 5 5 5 4 4 6 4 5 4	3

نمودارهای اول و دوم با هم تداخل ندارند پس می‌توانند در یک صفحه قرار بگیرند ولی نمودار سوم که با هردو نمودار قبلی برخورد دارد باید در یک صفحه جداگانه قرار بگیرد.

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

در این مثال برای هر نمودار احتیاج به یک صفحه جدا داریم . نمودار اول در نقطه ۲ با نمودار سوم و بین نقاط۲و ۳ با نمودار دوم برخورد دارد. نمودار دوم و سوم هم در نقطه‌ی اول و بین نقاط ۲ و ۳ با هم برخورد دارند.

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

namespace A8
{
    public class Q3Stocks : Processor
    {
        public Q3Stocks(string testDataName) : base(testDataName) { }

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

```

[۳] Stock Charts

```
        public virtual long Solve(long stockCount, long pointCount, long[][] matrix)
        {
            // write your code here
            throw new NotImplementedException();
        }
    }
}
```