دانشکده مهندسی کامپیوتر

طراحی و تحلیل الگوریتم‌ها

تمرین ۹*

اساتید حل تمرین: مطهّره میرزایی، محمّدجواد پیرهادی
تهیه و تنظیم مستند: مریم سادات هاشمی

استاد درس: سید صالح اعتمادی

نیم‌سال دوم ۹۹-۹۸

تلگرام	@MohammadJavad_Pirhadi @mirzaei2114
نام شاخه	fb_A9
نام پروژه/پوشه/پول ریکوست	A9
مهلت تحویل	۱۳۹۹/۲/۲۷

*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 A9 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 A9;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TestCommon;

namespace A9.Tests
{
    [DeploymentItem("TestData", "A9_TestData")]
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(200)]
        public void SolveTest_Q1InferEnergyValues()
        {
            //Assert.Inconclusive("A9.Q1 Not Solved");
            RunTest(new Q1InferEnergyValues("TD1"));
        }

        [TestMethod(), Timeout(200)]
        public void SolveTest_Q2OptimalDiet()
        {
            //Assert.Inconclusive("A9.Q2 Not Solved");
            RunTest(new Q2OptimalDiet("TD2"));
        }

        [TestMethod(), Timeout(200)]
        public void SolveTest_Q3OnlineAdAllocation()
        {
            //Assert.Inconclusive("A9.Q3 Not Solved");
            RunTest(new Q3OnlineAdAllocation("TD3"));
        }

        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("A9", p.Process, p.TestDataName, p.Verifier,
                VerifyResultWithoutOrder: p.VerifyResultWithoutOrder,
                excludedTestCases: p.ExcludedTestCases);
        }

    }
}
```

۱ میزان انرژی مواد سازنده [۱]

شما در این مسئله برای بدست آوردن میزان انرژی موادغذایی تشکیل‌دهنده‌ی منوی یک رستوران (با استفاده از لیست مواد تشکیل‌دهنده و میزان کالری هر کدام) الگوریتم Elimination Gaussian را پیاده‌سازی می‌کنید.

شما منوی یک رستوران را در اختیار دارید که در آن برای هر غذا لیست مواد تشکیل‌دهنده و تخمینی از میزان کالری آن غذا مشخص شده‌اند. شما باید میزان کالری مواد تشکیل‌دهنده را برای تخمین میزان کالری غذای مورد علاقه‌ی خود بدست آورید.

فرمت ورودی: در خط اول تعداد غذاهای هر منو (تعداد مواد تشکیل‌دهنده برابر تعداد غذاهای در منو است) و در خطوط بعدی E و $a_1, \ldots, a_n$ را در یافت می‌کنید که برای هر غذا (هر خط) میزان ماده‌ی i ام و E تخمینی از کالری کلی این غذا است. اگر ماده‌ای در غذایی استفاده نشده باشد مقدار آن 0 دریافت می‌شود ولی توجه داشته باشید که کد شما باید برای مقادیر منفی هم کار کند.

فرمت خروجی: برای هر کدام از مواد تشکیل‌دهنده کالری تخمین‌زده‌ی خود را رند کنید. به این صورت که اگر اعشار شما از ۰/۲۵ کمتر بود اعشار را برداشته و یا از ۰/۷۵ بزرگتر مساوی بود به بالا رند کنید (اعشار را برداشته و در صورت مثبت بودن جواب یک عدد به جواب اضافه و در غیر این‌صورت یک عدد از آن کم کنید) در غیر این صورت اعشار را ۰/۵ قرار دهید.

توجه:

لطفا عملیات رند کردن آخرین مرحله انجام شود.

[۱] Infer Energy Values of Ingredients

ورودی نمونه	خروجی نمونه
4 1 0 0 0 1 0 1 0 0 5 0 0 1 0 4 0 0 0 1 3	1 5 4 3

ورودی نمونه	خروجی نمونه
2 1 1 3 2 3 7	2 1

ورودی نمونه	خروجی نمونه
5 5 -5 -1 1 -2 -1	0 0.5

دقت کنید که جواب دقیق برابر با ۰/۴ و ۰/۲ بوده که به ۰ و ۰/۵ رند شده است.

```
using System;
using TestCommon;

namespace A9
{
    public class Q1InferEnergyValues : Processor
    {
        public Q1InferEnergyValues(string testDataName) : base(testDataName)
        {
        }

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

        public double[] Solve(long MATRIX_SIZE, double[,] matrix)
        {
            // Comment the line below and write your code here
            throw new NotImplementedException();
        }
    }
}
```

۲ مسئله‌ی رژیم بهینه [۲]

در این مسئله یک الگوریتم برای حل Programming Linear با تعداد نامساوری‌های کم برای حل مسئله‌ی رژیم بهینه پیاده سازی می‌کنید. شما می‌خواهید رژیم خود را بهینه کنید به این معنا که علاوه بر رعایت تمام ضوابط پیشنهاد داده شده توسط متخصص تغذیه، می‌خواهید بیشترین لذت را از غذای خود ببرید. شما ضوابط برای هر غذا و تخمینی از میزان علاقه خود به غذای مورد نظر را دارید. مثالی ضوابط می‌تواند "جمع میزان مصرف روزانه کره و پنیر باید از ۲ واحد کمتر باشد" باشد. هدف این مسئله افزایش لذت وعده غذایی خود علاوه بر رعایت ضوابط است. میزان لذت شما از هر وعده غذایی برابر جمع میزان علاقه شما از خوردن هر غذا و میزان پیشنهادی الگوریتم شما برای آن غذا است. ضوابط داده شده را می‌توان به فرم یک سری نامساوی‌های خطی نوشت. برای مثال ضابطه‌ی "جمع میزان مصرف روزانه کره و پنیر باید از ۲ واحد کمتر باشد" را می‌توان به صورت $amount_1 + amount_2 < 2$ نوشت. برای این مسئله در نظر داشته باشید که $amount_i >= 0$ است. برای رسیدن به هدف مسئله باید بیشترین میزان ممکن برای جمع $amount_i \times pleasure_i$ را بیابید. برای این کار باید توجه کنید که در ورودی حداکثر ۸ نامساوی داده می‌شود و جواب بهینه همواره در یکی از یال‌های چند ضلعی حاصل از نامساوی است. در نظر داشته باشید که تعداد کل نامساوی‌ها برابر مجموع تعداد نامساوی‌ها(n) و تعداد متغیرها(m) است (هر متغیر باید بزرگ‌تر از ۰ باشد که خود یک نامساوی تلقی می‌شود) و m نامساوی از $n + m$ نامساوی تبدیل به معادله تساوی می‌شود. برای حل باید m نامساوی را به عنوان معادله تساوی حل کنید و جواب را چک کنید که برای بقیه نامساوی‌ها جواب دهد و در بین این جواب‌ها مقدار بهینه را انتخاب کنید.

فرمت ورودی: در خط اول تعداد ضوابط و تعداد غذاها (که با n و m به ترتیب نشان داده می‌شوند) و در n خط بعدی ماتریس A (که با ابعاد $m \times n$ است) را دریافت می‌کنید به صورتی که خط iام $a_{i1}, \ldots, a_{im}$ را شامل می‌شود. پس از دریافت ماتریس A در خط بعدی وکتور B به طول n را دریافت می‌کنید به طوری که $Ax < B$ است (دقت کنید که x میزان مصرفی از هر غذا و به طول m است). برای مثال اگر فرض کنیم ۲ نوع غذای f_1 و f_2 داریم ($m == 2$) اگر در خط iام از ماتریکس ۱ و ۳ را به عنوان ورودی بگیریم و مقدارiام وکتور B برابر ۵ باشد یعنی $1 \times amount_{f1} + 3 \times amount_{f2} < 5$ باید باشد. در خط آخر از ورودی وکتور V که نمایانگر میزان علاقه به هر غذا است را به عنوان ورودی می‌گیرید. برای مثال اگر خط آخر ورودی به ترتیب ۲ و -1 باشد یعنی برای رسیدن به هدف مسئله باید

$$amount_{f1} \times 2 + -1 \times amount_{f2}$$

را بیشینه کنید. در نظر داشته باشید که میزان علاقه به یک غذا می‌تواند منفی نیز باشد.

فرمت خروجی: اگر هیچ رژیمی با این شرایط یافت نمی‌شد solution No و اگر تعداد زیادی رژیم با این محدودیت‌ها بود Infinity را به خروجی دهید. در غیر این‌صورت solution Bounded را در خط اول و وکتوری از میزان پیشنهادی برای مصرف هر غذا در این رژیم پس از رند شدن به روش توضیح داده را در خط دوم بنویسید.

[۲] Optimal Diet Problem

ورودی نمونه	خروجی نمونه
3 2 -1 1 1 0 0 1 -1 2 2 -1 2	Bounded solution 0 2

ورودی نمونه	خروجی نمونه
2 2 1 1 -1 -1 1 -2 1 1	No Solution

ورودی نمونه	خروجی نمونه
1 3 0 0 1 3 1 1 1	Infinity

```
using System;
using TestCommon;

namespace A9
{
    public class Q2OptimalDiet : Processor
    {
        public Q2OptimalDiet(string testDataName) : base(testDataName)
        {
        }

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

        public string Solve(int N, int M, double[,] matrix1)
        {
            // Comment the line below and write your code here
            throw new NotImplementedException();
        }
    }
}
```

۳ جاگذاری تبلیغات آنلاین [۳]

یکی از بیزینس‌های سودآور در دنیا تبلیغات آنلاین است. گوگل و فیسبوک سالانه بیلیون‌ها دلار سود از این راه بدست می‌آورند و غالب بر ۹۰ درصد سود خود را از تبلیغات بدست می‌آورند. در این مسئله شما به یک سیستم آنلاین تبلیغات مثل AdSense Google یا Yandex در جایگذاری تبلیغات با پیاده‌سازی الگوریتم Simplex کمک می‌کنید تا علاوه بر پر کردن محل تبلیغات، نیازمندی‌های متقاضیان تبلیغ را پوشش دهید. شما n مشتری دارید که هرکدام می‌خواهند تبلیغ‌شان توسط تعدادی کاربر که در قرارداد ذکر شده، دیده شود. شبکه‌ی تبلیغات آنلاین شما m جا برای تبلیغات دارد. شما میزان پولی که هر مشتری برای هر کاربر که تبلیغش را ببیند پرداخت می‌کند، تعداد حداقل کاربرانی که می‌خواهد تبلیغش را ببیند و همچنین تعداد کاربرانی که هر کدام از m جای مخصوص تبلیغات را می‌بینند در اختیار دارید. شما می‌توانید تبلیغات متفاوت را در طول یک ماه در یک جا نمایش بدهید یا یک جا را فقط به یک تبلیغ خاص اختصاص دهید. هدف بیشینه کردن سود که برابر جمع مبلغی است که هر مشتری برای تعداد کاربرهایی که تبلیغش را دیده پرداخت می‌کند. اگر فرض کنیم x_{ij} تعداد کاربرانی است که تبلیغ امi را در جای امj دیده‌اند؛ در این صورت می‌توان تمام شروط را به صورت یک سری نامساوی خطی نوشت. برای مثال $\sum x_{ij} = S_j$ برابر تعداد کل کاربرانی است که تبلیغ i ام را می‌بینند و اگر مشتری i ام بخواهد تبلیغاتش توسط حداقل U_i کاربر دیدهشوند؛ آنگاه نامساوری $\sum x_{ij}U_i$ می‌بایست درست باشد. دقت کنید که $x_{ij}0$ است. اگر میزان مبلغی که مشتریiام برای هر کاربر که تبلیغش را در جای امj می‌بیند، با cij نشان دهیم، هدف مسئله بیشینه کردن $\sum\sum c_{ij}x_{ij}$ است که یک مسئله‌ی programming linear ولی با تعداد متغیرهای بیشتر است.

فرمت ورودی: شما مسئله را ساده شده به صورت یک مسئله ی programming linear دریافت می‌کنید. در خط اول شما به ترتیب p و q را که نشان‌دهنده‌ی تعداد نامساوی‌ها و تعداد متغیرهاست را دریافت می‌کنید. در p خط بعد ماتریس A (ابعاد $p \times q$ است) را دریافت می‌کنید به صورتی که خط iام $a_{i1}, a_{i2}, ..., a_{iq}$ را شامل می‌شود. پس از دریافت ماتریس A در خط بعدی وکتور B به طول q را دریافت می‌کنید به طوری که $Ax < B$ است. در خط آخر شما وکتور C را دریافت می‌کنید که هدف مسئله بیشینه کردن $\sum z_i x_j$ با رعایت ضوابط است.

فرمت خروجی: اگر هیچ رژیمی با این شرایط یافت نمی‌شد solution No و اگر تعداد زیادی رژیم با این محدودیت ها بود Infinity را به خروجی دهید. در غیر اینصورت solution Bounded را در خط اول و وکتوری از میزان پیشنهادی برای مصرف هر غذا در این رژیم پس از رند شدن به روش توضیح داده شده را در خط دوم بنویسید. (دقت کنید که $x = x_{ij}$ همان تعداد کاربرانی است که تبلیغiام در جای امj می‌بینند ولی در این سوال با $x_1, x_2, , x_q$ نماش داده می‌شوند).

[۳] Online Advertisement Allocation

ورودی نمونه	خروجی نمونه
3 2 -1 1 1 0 0 1 -1 2 2 -1 2	Bounded Solution 0 2

ورودی نمونه	خروجی نمونه
2 2 1 1 -1 -1 1 -2 1 1	No Solution

ورودی نمونه	خروجی نمونه
1 3 0 0 1 3 1 1 1	Infinity

```
using System;
using TestCommon;

namespace A9
{
    public class Q3OnlineAdAllocation : Processor
    {

        public Q3OnlineAdAllocation(string testDataName) : base(testDataName)
        {
        }

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

        public string Solve(int c, int v, double[,] matrix1)
        {
            // Comment the line below and write your code here
            throw new NotImplementedException();
        }
    }
}
```