دانشکده مهندسی کامپیوتر

طراحی و تحلیل الگوریتم‌ها

تمرین ۱۰*

اساتید حل تمرین: مهدی مقدمی، مهسا قادران
تهیه و تنظیم مستند: مریم سادات هاشمی

استاد درس: سید صالح اعتمادی

نیم‌سال دوم ۹۹-۹۸

تلگرام	@mahdimoqaddami @mahsa_noname
نام شاخه	fb_A10
نام پروژه/پوشه/پول ریکوست	A10
مهلت تحویل	۱۳۹۹/۳/۵

*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.

توجه:
خروجی این مجموعه تمرین شما توسط یک **SAT-Solver** راست‌آزمایی می‌شود. برای این منظور لازم است فایل جدید `TestCommon.cs` جایگزین شود. همچنین لازم است دستور زیر در پوشه پروژه `TestCommon` اجرا شود تا بسته نرم‌افزاری `OR-Tools` برای این پروژه دانلود و نصب شود.

```
dotnet add package Kingdom.OrTools.ConstraintSolver.Core
```

توضیحات کلی تمرین

۱. ابتدا مانند تمرین های قبل، یک پروژه به نام `A10` 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 A10;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TestCommon;

namespace A10.Tests
{
    [DeploymentItem("TestData", "A10_TestData")]
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q1FrequencyAssignment()
        {
            Assert.Inconclusive("A10.Q1 Not Solved");
            RunTest(new Q1FrequencyAssignment("TD1"));
        }

        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q2CleaningApartment()
        {
            Assert.Inconclusive("A10.Q2 Not Solved");
            RunTest(new Q2CleaningApartment("TD2"));
        }

        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q3AdBudgetAllocation()
        {
            Assert.Inconclusive("A10.Q3 Not Solved");
            RunTest(new Q3AdBudgetAllocation("TD3"));
        }

        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("A10", p.Process, p.TestDataName, p.Verifier,
                VerifyResultWithoutOrder: p.VerifyResultWithoutOrder,
                excludedTestCases: p.ExcludedTestCases);
        }
    }
}
```

۱ قراردادن فرکانس برای برج‌های شبکه جی اس ام [۱]

در این مسئله شما یاد می‌گیرید که مسئله‌ی دادن فرکانس به شبکه gsm را به مسئله‌ی رنگ کردن یک گراف با ۳ رنگ کاهش دهید. سپس الگوریتمی طراحی می‌کنید که این مسئله را به یک SAT تبدیل می‌کند. شبکه‌های gsm برای برقراری ارتباط بین تلفن‌های همراه هستند که شامل فرستادن اطلاعات محیطی که با فرکانس‌های مختلف کار می‌کند؛ می‌شود. در هر شش ضلعی به طور معمول یک برج شبکه قرار می‌گیرد که به آن cell می‌گویند. (به همین خاطر به تلفن همراه phone cell می‌گویند) هر تلفن همراه از بین برج‌های موجود برجی با بیشترین قدرت را انتخاب می‌کند. برای این منظور برج‌های نزدیک و در همسایگی یکدیگر باید از فرکانس‌های مختلف استفاده کنند. شما باید روی مسئله‌ی دادن فرکانس به برج‌های مختلف کار کنید و ۳ فرکانس مختلف در دست دارید. همسایگی‌ها برای شما مشخص است و شما باید امکان دادن فرکانس‌های مختلف برای این شبکه با ۳ فرکانس را بررسی کنید. در واقع این مسئله با مسئله‌ی معروف رنگ‌آمیزی گراف برابر است و شما یک گراف به عنوان ورودی دریافت می‌کنید. رنگ‌ها همان فرکانس‌ها، نودها همان برج‌ها و یال‌ها همان همسایگی برج‌ها باهم هستند. این مسئله یک مسئله‌ی complete np است و راه حل بهینه برای حل آن را نمی‌دانیم. پس شما باید آن را به یک SAT کاهش دهید که اغلب با برنامه‌هایی به اسم هاSAT-solver قابل حل هستند.

فرمت ورودی: خط اول ورودی دو عدد n یا تعداد نودها و m یا تعداد یال‌ها را دارد. شماره‌گذاری نودها از ۱ تا n است و m خط بعدی شماره‌ی نودهایی که بهم متصل اند رو به صورت دو به دو دارد. (هر یال نمی‌تواند به خودش متصل شود)

فرمت خروجی: شما باید یک فرمول بولین را به فرم CNF با فرمتی که معرفی می‌کنیم به خروجی دهید. اگر مسئله قابل حل است و جوابی وجود دارد که هرنود گراف یک رنگ داشته باشد و هر دو یال کنار هم یک رنگ نباشند، فرمول شما باید قابل حل باشد در غیر این‌صورت فرمول شما باید غیرقابل حل باشد. تعداد متغیرهای فرمول شما نباید بیشتر از ۳۰۰۰ و تعداد نامساوی‌ها نباید بیشتر از ۵۰۰۰ باشد.

در خط اول خروجی دو عدد c تعداد نامساوی‌ها و v تعداد متغیرها را قراردهید. برای متغیرهای خود از ۱ تا v یک ایندکس قراردهید. دقت کنید که متغیرها به فرم باینری اند. فرمت هر خط فرمول به صورت x۱ or x۲ or x۳ x۷ or است. در c خط بعدی می‌بایست ابتدا ایندکس متغیرهای فرمول را قرار دهید سپس یک ۰ در انتها قرار دهید که بتوان پایان خط را تشخیص داد. برای مثال آورده شده باید ۱ ۲ ۳ ۷ ۰ در خروجی قرار گیرند. not هر متغیر را با یک منفی قبل از ایندکس آن نشان دهید. دقت داشته باشید که هر عدد به جز عدد آخر باید عددی غیر صفر بین v- تا v باشد و ۱<C<۵۰۰۰ و ۱<=V<=۳۰۰۰ است. اگر تعداد فرمول‌هایی که می‌توانند امکان پذیر بودن مسئله را نشان دهند بیشتر از ۱ بودند هرکدام از آن‌ها را می‌توانید به خروجی دهید خروجی ها با یک SAT-solver چک می‌شوند.

راهنمایی: برای هر نود سه متغیر بولین قراردهید که نشان‌دهنده‌ی هر رنگ باشند و در صورت یک شدن آن رنگ به آن نود داده می‌شود. حال باید به گونه‌ای ضوابطی مانند هر نود تنها یک رنگ باید داشته باشد و نودهای همسایه می‌بایست رنگ‌های متفاوت داشته باشند را با این متغیرها به فرم CNF بنویسید.

در مثال زیر دقت کنید که گراف ورودی قابل رنگ آمیزی با ۳ رنگ به طوری که همسایه‌ها رنگ‌های متفاوتی داشته باشند است پس هدف این است که در خروجی یک فرمول بدهید که زمانی که به یک Solver SAT داده می‌شود حل شود و جوابی داشته باشد. مهم این است که برای گراف‌های ممکن خروجی در Solver SAT حل شود و برای ناممکن‌ها حل نشود و فرمول‌های ممکن یکتا نیستند. خروجی فرمول زیر x یا x not است که نمونه‌ی ساده‌ای از فرمول‌های قابل حل است.

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

[۱] Assign Frequencies to the Cells of a GSM Network

مثال زیر غیر قابل‌حل است پس فرمولی غیرقابل حل باید به خروجی بدهیم که x و x not به خروجی داده شده است که نمونه‌ی ساده‌ای از فرمول‌های غیرقابل حل است.

ورودی نمونه	خروجی نمونه
4 6 1 2 1 3 1 4 2 3 2 4 3 4	2 1 1 0 -1 0

```
using System;
using TestCommon;

namespace A10
{
    public class Q1FrequencyAssignment : Processor
    {
        public Q1FrequencyAssignment(string testDataName) : base(testDataName) { }

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


        public String[] Solve(int V, int E, long[,] matrix)
        {
            // write your code here
            throw new NotImplementedException();
        }

        public override Action<string, string> Verifier { get; set; } =
            TestTools.SatVerifier;

    }
}
```

۲ تمیز کردن آپارتمان [۲]

در این مسئله شما بررسی می‌کنید که آیا امکان‌پذیر است که یک آپارتمان را بعد از یک مهمانی به گونه‌ای تمیز کنید که اثری از مهمانی در آن باقی نماند. شما باید این مسئله را به مسئله‌ی مسیر همیلتونی تبدیل و یک الگوریتم برای تبدیل آن به SAT پیاده کنید. شما یک مهمانی در خانه داشتید و می‌خواهید خانه را تمیز کنید به طوری که اثری از مهمانی باقی نماند. برای این منظور شما بعد از تمیز کردن یک اتاق نمی‌توانید دوباره از آن عبور کنید به دلیل این‌که ممکن است اثری از مهمانی باقی بگذارید. پس شما باید از یک اتاق به اتاق دیگری که در همسایگی اتاق قبلی است بروید، همه‌ی اتاق‌ها را تمیز کنید و از هر اتاق فقط یک بار عبور کنید. این مسئله می‌تواند به مسئله‌ی معروف مسیر همیلتونی تبدیل شود، یعنی با دادن یک گراف امکان یافتن راهی که از همه‌ی نودها فقط یکبار می‌گذرد بیابید. اتاق‌ها همان

[۲] Cleaning the Apartment

نودها و همسایگی آن‌ها همان یال‌ها هستند. مسیر همیلتونی یک مسئله‌ی complete np است و راه‌حل بهینه‌ای برای آن نداریم. شما می‌توانید با کاهش آن به یک SAT به کمک یک SAT-solver ان را بهینه حل کنید.

فرمت ورودی: خط اول ورودی دو عدد n یا تعداد اتاق‌ها و m یا تعداد اتاق‌های متصل بهم را دارد. شماره‌گذاری اتاق‌ها از ۱ تا n است و m خط بعدی شماره‌ی اتاق‌هایی که بهم متصل‌اند رو به صورت دو به دو دارد. درها دو طرفه‌اند یعنی از هر دو اتاق در همسایگی یکدیگر می‌توان به دیگری رفت. بدیهی است که هر در هر اتاق را به خودش وصل نمی‌کند. دقت کنید که هر در ممکن است چندبار به ورودی داده شود و یا در بین دو اتاق u و v بصورت u v و یا v u نمایش داده شود.

فرمت خروجی: شما باید یک فرمول بولین را به فرم CNF با فرمتی که معرفی می‌کنیم به خروجی دهید. اگر مسئله قابل‌حل است و بتوان از یک یال شروع کرد و از هر یال حتما یک بار بدون گذشتن از یال‌های تکراری گذشت، فرمول شما باید قابل حل باشد در غیر این‌صورت فرمول شما باید غیرقابل حل باشد. دقت کنید که تعداد متغیرها باید از ۱۲۰۰۰۰ تا کمتر باشد.

در خط اول خروجی دو عدد c تعداد نامساوی‌ها و v تعداد متغیرها را قراردهید. برای متغیرهای خود از ۱ تا v یک ایندکس قرار دهید. دقت کنید که متغیرها به فرم باینری‌اند. فرمت هر خط فرمول به صورت x۱ or x۲ or x۳ است. در c خط بعدی می‌بایست ابتدا ایندکس متغیرهای فرمول را قرار دهید سپس یک ۰ در انتها قرار دهید که بتوان پایان خط را تشخیص داد.

برای مثال اورده شده باید ۱ ۲ ۳ ۷ ۰ در خروجی قرار گیرند. not هر متغیر را با یک منفی قبل از ایندکس آن نشان دهید. دقت داشته باشید که هر عدد به جز عدد آخر باید عددی غیر صفر بین v- تا v باشد. اگر تعداد فرمول‌هایی که می‌توانند امکان‌پذیر بودن مسئله را نشان دهند بیشتر از ۱ بودند؛ هر کدام از آن‌ها را می‌توانید به خروجی دهید خروجی‌ها با یک SAT-solver چک می‌شوند.

راهنمایی: به ازای هر نود یا اتاق به تعداد اتاق‌ها متغیر تعریف کنید که این متغیرها نشان‌دهنده‌ی ترتیب آن‌ها در مسیر باشد. برای مثال اگر ۳ اتاق داشتته باشیم و برای اتاق اول داشته باشیم ۰۱۰ یعنی این اتاق دومین اتاقی است که آن را تمیز می‌کنیم. حال باید ضوابطی نظیر: تمیز کردن حتمی هر اتاق، خالی نبودن یک ترتیب در مسیر (برای مثال دومین اتاقی که تمیز می‌کنیم باید وجود داشته باشد)، هر اتاق تنها یک بار در مسیر باشد، هیچ دو اتاقی در یک ترتیب مسیر قرار نگیرند(برای مثال اتاق اول در مسیر تنها یک اتاق می‌تواند باشد نه چندتا) و همسایگی (داشتن در) دو اتاق متوالی در مسیر را به فرم CNF بنویسید.

ورودی نمونه	خروجی نمونه
5 4 1 2 2 3 3 5 5 4	1 1 1 -1 0

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

```
using System;
using TestCommon;

namespace A10
```

```
{
    public class Q2CleaningApartment : Processor
    {
        public Q2CleaningApartment(string testDataName) : base(testDataName)
        {
        }

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

        public override Action<string, string> Verifier { get; set; } =
            TestTools.SatVerifier;

        public String[] Solve(int V, int E, long[,] matrix)
        {
            // write your code here
            throw new NotImplementedException();
        }

    }
}
```

۳ جاگذاری بودجه‌ی تبلیغات [۳]

در تمرین قبلی شما روی تبلیغات آنلاین کار کردید. در این مسئله شما برای یک شرکت بزرگ که برای پیشرفت از تبلیغات استفاده می‌کند کار می‌کنید. شما باید تعیین کنید که آیا امکان‌پذیر است که بودجه‌ی تبلیغات را به همراه تمام ضوابط رعایت کنید یا خیر. شما یاد می‌گیرد که این مسئله را به یک programming Linear Integer تبدیل کنید و سپس الگوریتمی طراحی می‌کنید که این مسئله را به SAT کاهش دهد.

شرکت شما زیرمجموعه‌های زیادی دارد که روی تبلیغات تلوزیونی، رادیویی، اینترنتی و غیره کار می‌کنند. همه‌ی آن‌ها برنامه‌های تبلیغاتی خود را ریخته‌اند و شما برای پوشش همه‌ی این برنامه‌ها بودجه ندارید. شما باید برنامه‌ی بودجه را فردا تحویل دهید و زمان کافی برای مطالعه‌ی پیشنهاد همه‌ی زیرمجموعه‌ها ندارید. شما تصمیم می‌گیرید که یک پیشنهاد را به طور کامل رد و یا قبول کنید ولی با ضوابط بسیاری روبه‌رو می‌شوید. برای مثال بودجه‌ی کل تبلیغات شما محدود است و یک سری قراردادها با بعضی شرکت‌های تبلیغاتی دارید که شمارا مجبور به صرف یک میزان هزینه‌ی حداقلی برای آن نوع تبلیغات می‌کند و صرف نظر از آن جریمه سنگینی دارد. ضوابط دیگری نیز همانند اجبار برای گذاشتن حداقل ۱۰ درصد بودجه یا یک ملیون در ماه برای تبلیغات تلوزیونی به منظور به یاد موندن برند شما برای مردم نیز می‌توانند تعریف شوند. تمام این ضوابط و شروط را می‌توان به فرم programming linear integer بازنویسی کرد. Xi می‌تواند قبول شدن یا نشدن پیشنهاد دپارتمان i درنظر گرفت و شرایط را بازنویسی کرد. برای مثال قابل پوشش دادن هزینه تمام پیشنهادهای پذیرفته‌شده با بودجه‌ای که داریم را می‌توان به صورت ... نوشت. شما ضوابط را به صورت programming linear integer دریافت می‌کنید و می‌بایست آن را به SAT کاهش دهید. دقت کنید که در این مسئله حداکثر ۳ متغیر با کوئفیشنت غیر ۰ در هر نامساوی وجود دارد.

فرمت ورودی: در خط ورودی دو عدد n تعداد نامساوی‌ها و m تعداد متغیرها را دریافت می‌کنید. در n خط بعدی توضیحات ماتریس کوئفیشنت نامساوی‌ها، A که nxm بعدی است را دریافت می‌کنید. هر خط m عدد صحیح دارد که حداکثر ۳ عدد از آن‌ها غیر ۰ است. خط اخر n عدد صحیح که وکتور b را تشکیل می‌دهند را دارد. برای سیستم نامساوی Ax<=b باید بررسی کنید که آیا وکتور ایx وجود دارد که این سیستم را پوشش دهد یا خیر.

فرمت خروجی: شما باید یک فرمول بولین را به فرم CNF با فرمتی که معرفی می‌کنیم به خروجی دهید. اگر مسئله قابل حل است و جوابی برای آن وجود دارد که تمام ضوابط را پوشش می‌دهد، فرمول شما باید قابل حل باشد

[۳] Advertisement Budget Allocation

در غیر این صورت فرمول شما باید غیرقابل حل باشد. تعداد متغیرهای فرمول شما نباید بیشتر از ۳۰۰۰ و تعداد نامساوی‌ها نباید بیشتر از ۵۰۰۰ باشد.

در خط اول خروجی دو عدد c تعداد نامساوی‌ها و v تعداد متغیرها را قراردهید. برای متغیرهای خود از ۱ تا v یک ایندکس قراردهید. دقت کنید که متغیرها به فرم باینری‌اند. فرمت هر خط فرمول بصورت x۱ or x۲ or x۳ or x۷ است. در c خط بعدی می‌بایست ابتدا ایندکس متغیرهای فرمول را قرار دهید سپس یک ۰ در انتها قرار دهید که بتوان پایان خط را تشخیص داد.

برای مثال آورده شده باید ۱ ۲ ۳ ۷ ۰ در خروجی قرار گیرند. not هر متغیر را با یک منفی قبل از ایندکس آن نشان دهید. دقت داشته باشید که هر عدد به جز عدد آخر باید عددی غیر صفر بین v- تا v باشد و ۱<C<۵۰۰۰ و ۱<=V<=۳۰۰۰ است. اگر تعداد فرمول‌هایی که می‌توانند امکان‌پذیر بودن مسئله را نشان‌دهند بیشتر از ۱ بودند هرکدام از آن‌ها را می‌توانید به خروجی دهید. خروجی‌ها با یک SAT-solver چک می‌شوند.

ورودی نمونه	خروجی نمونه
2 3 5 2 3 -1 -1 -1 6 -2	1 1 -1 1 0

ورودی نمونه	خروجی نمونه
3 3 1 0 0 0 1 0 0 0 1 1 1 1	1 1 1 -1 0

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

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

```
using System;
using TestCommon;

namespace A10
{
```

```
    public class Q3AdBudgetAllocation : Processor
    {
        public Q3AdBudgetAllocation(string testDataName) : base(testDataName) { }

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

        public override Action<string, string> Verifier { get; set; } =
            TestTools.SatVerifier;

        public string[] Solve(long eqCount, long varCount, long[][] A, long[] b)
        {
            // write your code here
            throw new NotImplementedException();
        }
    }
}
```