



دانشکده مهندسی کامپیوتر

طراحی و تحلیل الگوریتم‌ها

امتحان عملی ۲

اساتید حل تمرین: محمد جواد میرشکاری، غزاله محمودی
استاد درس: سید صالح اعتمادی

نیم‌سال دوم ۹۸-۹۹

@mj_haghighi @ghazale_mahmoodi	تلگرام
fb_E2	نام شاخه
E2	نام پروژه/پوشه/پول ریکوست
۴ ساعت	مدت امتحان

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 استفاده از نرم‌افزار Teams و درست کردن PullRequest مجاز است. هرگونه 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 در برای این درس نوشته نشده یا در گیت شما قبلاً موجود نبوده مجاز نمی‌باشد.
۴. استفاده از هرگونه ویدیو مجاز نمی‌باشد.
۵. انتظار می‌رود که تمام دانشجویانی که تمرین‌ها را خودشان حل کرده‌اند بتوانند هر دو سوال را در زمان مشخص شده حل کنند. ولی چنانچه هیچ ایده‌ای برای حل مساله ندارید برای سوال ۱ استاد درس می‌تواند به ازای کسر ۲۵٪ نمره یک راهنمایی بکند. برای سوال ۲ نیز می‌تواند ۲ راهنمایی بکند که به ازای هر کدام از راهنمایی‌ها ۲۵٪ نمره کسر خواهد شد.
۶. تصویر صفحه نمایش و وب‌کم شما در کل مدت امتحان بدون وقفه باید توسط نرم‌افزار FlashBackExpress (یا نرم‌افزار مشابه) ضبط شده و پس از فشرده‌سازی برای استاد درس ارسال 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.

```

1 using Microsoft.VisualStudio.TestTools.UnitTesting;
2 using System;
3 using TestCommon;
4
5 namespace E2.Tests
6 {
7     [DeploymentItem("TestData")]
8     [TestClass()]
9     public class GradedTests
10    {
11        [TestMethod(), Timeout(5000)]
12        public void SolveTest_Q1MaxflowVertexCapacity()
13        {
14            //Assert.Inconclusive("E2.Q1 Not Solved");
15            RunTest(new Q1MaxflowVertexCapacity("TD1"));
16        }
17
18        [TestMethod(), Timeout(1500)]
19        public void SolveTest_Q2BoardGame()
20        {
21            //Assert.Inconclusive("E2.Q2 Not Solved");
22            RunTest(new Q2BoardGame("TD2"));
23        }
24
25        public static void RunTest(Processor p)
26        {
27            TestTools.RunLocalTest("E2", p.Process, p.TestDataName, p.Verifier,
28                VerifyResultWithoutOrder: p.VerifyResultWithoutOrder,
29                excludedTestCases: p.ExcludedTestCases);
30        }
31    }
32 }

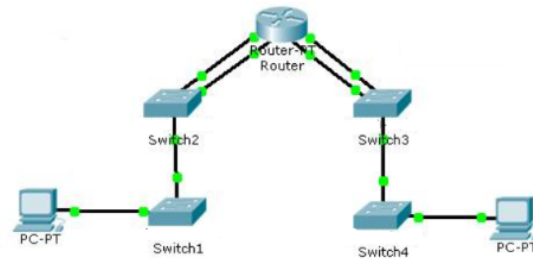
```

۱ جریان بیشینه با محدودیت ظرفیت در گره‌ها

مساله پیدا کردن جریان بیشینه^۱ 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 شبکه‌های کامپیوتری برای اتصال به اینترنت و ارسال یا دریافت داده در بستر آن، داده‌های ما از یک سری گره‌ها به نام router و لینک‌های بین آن‌ها عبور می‌کنند. شکل زیر تصویر کوچکی است از مسیری که داده‌ها از آن عبور می‌کنند.

می‌دانیم هر کدام از گره‌ها یا router/switch‌هایی که در مسیر قرار دارند در هر ثانیه حداکثر مقدار مشخصی داده را می‌توانند 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 می‌شود 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 برنامه‌ای 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.

^۱MaxFlow

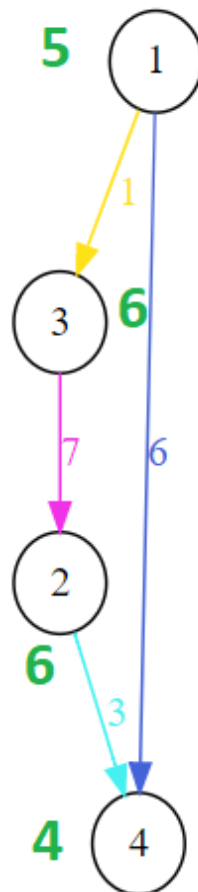


فرمت ورودی: خط اول به ترتیب تعداد گره‌ها v و تعداد یال‌ها e را مشخص می‌کند. سپس e خط بعدی یال‌ها را بصورت گره مبدا، گره مقصد و ظرفیت مشخص می‌کند. در خط بعد ظرفیت v گره با فاصله از هم جدا شده‌اند. نهایتاً در خط آخر گره مبدا و مقصد مشخص شده‌اند. البته فایل‌های ورودی برای شما پردازش شده و بصورت متغیرهای 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.

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



```

1 using System;
2 using TestCommon;
3
4 namespace E2
5 {
6     public class Q1MaxflowVertexCapacity : Processor
7     {
8         public Q1MaxflowVertexCapacity(string testDataName) : base(testDataName)
9         {}
10
11         public override string Process(string inStr) =>
12             TestTools.Process(inStr, (Func<long, long, long[][], long[] , long, long, long>)Solve)
13
14         public virtual long Solve(long nodeCount,
15             long edgeCount, long[][] edges, long[] nodeWeight,
16             long startNode , long endNode)
17         {
18             // write your code here
19             throw new NotImplementedException();
20         }
21     }
22 }

```

۲ بازی تخته‌ای^۲

در این سوال تخته‌ی بازی در اختیار شما قرار می‌گیرد که هر خانه آن یا خالی است و یا یک مهره قرمز یا آبی در آن قرار دارد. بازی از این قرار است که شما تنها مجاز هستید مهره ها را از داخل هر خانه بردارید و آن خانه را خالی کنید و با این کار باید سعی کنید به شرایط زیر در بازی برسید:



۱. در هر سطر از تخته حداقل یک مهره باقی بماند.

۲. در هر ستون از تخته حداقل یک مهره باقی بماند.

۳. مهره های درون هر ستون هم‌رنگ باشند.

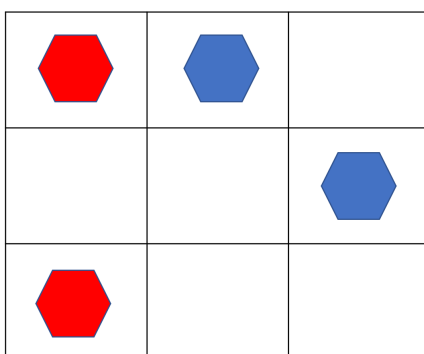
فرمت ورودی: در خط اول ابعاد تخته به شما داده می‌شود، تخته بازی مربعی است مثلاً در مثال بالا ابعاد تخته 2×2 است. در خط‌های بعدی هر سطر تخته به ترتیب داده شده است. عدد (۱ خالی) نشان‌دهنده‌ی خالی بودن آن خانه از تخته است، همچنین عدد (۲ آبی) نشان‌دهنده‌ی مهره آبی و عدد (۳ قرمز) نشان‌دهنده‌ی مهره قرمز در آن خانه از تخته است.

خروجی نمونه	ورودی نمونه
UNSATISFIABLE	2 1 3 1 3

شکل ۱: نمونه اول

خروجی نمونه	ورودی نمونه
SATISFIABLE	3 3 2 1 1 1 2 3 1 1



شکل ۲: نمونه دوم

فرمت خروجی: شما باید با داشتن تخته به این سوال پاسخ دهید که آیا این تخته قابل حل است یا خیر. برای این کار شما باید یک فرمول بولین را به فرم CNF با فرمتی که در طول ترم معرفی شد به خروجی دهید. اگر مسئله قابل حل باشد فرمول شما هم باید قابل حل باشد در غیر این صورت فرمول شما باید غیرقابل حل باشد.

یادآوری فرمت CNF: در خط اول خروجی دو عدد c تعداد نامساوی‌ها و v تعداد متغیرها را قرار دهید. برای متغیرهای خود از ۱ تا v یک ایندکس قرار دهید. دقت کنید که متغیرها به فرم باینری اند. فرمت هر خط فرمول به صورت $x_1 \vee x_2 \vee x_3 \vee x_7$ 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. نقیض یا 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 و $v - 1$ باشد و $1 < C < 5000$ و $1 \leq V \leq 3000$ است. برای جزئیات بیشتر می‌توانید به مستند تمرین ۱۰ مراجعه کنید.

```

۱ public class Q2BoardGame : Processor
۲ {
۳     public Q2BoardGame(string testDataName) : base(testDataName) { }
۴
۵     public override string Process(string inStr) =>
۶         TestTools.Process(inStr, (Func<int, long[,], string[]>)Solve);
۷
۸
۹     public string[] Solve(int BoardSize, long[,], Board)
۱۰    {
۱۱        // write your code here
۱۲        throw new NotImplementedException();
۱۳    }
۱۴    public override Action<string, string> Verifier { get; set; } =
۱۵        TestTools.SatVerifier;
۱۶ }

```