دانشکده مهندسی کامپیوتر

ساختمان داده

تمرین ۱۲*

زهرا حسینی
سهراب نمازی
سید صالح اعتمادی

نیم‌سال اول ۰۰-۹۹

ایمیل/تیمز	hosseini_zahra@comp.iust.ac.ir sohrab_namazi@comp.iust.ac.ir
نام شاخه	fb_A12
نام پروژه/پوشه/پول ریکوست	A12
مهلت تحویل	۹۹/۱۰/۱۳

*تشکر ویژه از خانم مریم سادات هاشمی که در نیم‌سال اول سال تحصیلی ۹۷-۹۸ نسخه اول این مجموعه تمرین‌ها را تهیه فرمودند. همچنین از اساتید حل‌تمرین نیم‌سال اول سال تحصیلی ۹۹-۹۸ سارا کدیری، محمد مهدی عبداللهپور، مهدی مقدمی، مهسا قادران، علیرضا مرادی، پریسا یل‌سوار، غزاله محمودی و محمدجواد میرشکاری که مستند این مجموعه تمرین‌ها را بهبود بخشیدند، متشکرم.

توضیحات کلی تمرین

۱. ابتدا مانند تمرین های قبل، یک پروژه به نام `A12` 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.

در این سری از تمرین، علاوه برفایل ها با پسوند txt که تست کیس های سوالات بودند و شما از آن ها برای تست کدتان استفاده می کردید، فایل هایی با پسوند webgraphviz نیز در پوشه TestDat وجود دارد. شما با استفاده از این فایل ها می توانید گراف های کوچکتر از ۱۰۰ گره را به صورت تصویری در این سایت مشاهده کنید.

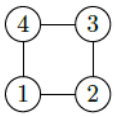
```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TestCommon;

namespace A12.Tests
{
    [DeploymentItem("TestData")]
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(100)]
        public void SolveTest_Q1MazeExit()
        {
            RunTest(new Q1MazeExit("TD1"));
        }

        [TestMethod(), Timeout(4000)]
        public void SolveTest_Q2AddExitToMaze()
        {
            RunTest(new Q2AddExitToMaze("TD2"));
        }

        [TestMethod(), Timeout(300)]
        public void SolveTest_Q3Acyclic()
        {
            RunTest(new Q3Acyclic("TD3"));
        }

        [TestMethod(), Timeout(10000)]
        public void SolveTest_Q4OrderOfCourse()
        {
            RunTest(new Q4OrderOfCourse("TD4"));
        }

        [TestMethod(), Timeout(500)]
        public void SolveTest_Q5StronglyConnected()
        {
            RunTest(new Q5StronglyConnected("TD5"));
        }

        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("A12", p.Process, p.TestDataName, p.Verifier,
                        VerifyResultWithoutOrder: p.VerifyResultWithoutOrder,
                    excludedTestCases: p.ExcludedTestCases);
        }
    }
}
```

۱ پیدا کردن راه خروجی [۱]

Maze یک شبکه مستطیل شکل از سلول ها و خانه هاست. در این سوال شما باید بررسی کنید که با شروع از یک نقطهی خاص از maze آیا مسیری برای خروج از آن وجود دارد یا خیر. برای این منظور، می توانید maze را یک گراف غیر جهت دار در نظر بگیرید که خانه های maze رأس این گراف هستند. اگر دو خانه با هم مجاور باشند و هیچ دیواری بین آن ها نباشند، با یک یال غیر جهت دار به هم متصل می شوند. بنابراین، برای بررسی این که آیا بین دو خانه داده شده در maze مسیری وجود دارد یا خیر، کافی است، چک کنیم که بین دو رأس متناظر در گراف مسیر وجود دارد یا خیر.

خط اول فایل ورودی، تعداد گره های گراف(یعنی n) را مشخص می کند. هر یک از n خط بعدی، شامل دو گره است که بدین معنی است که این دو گره توسط یک یال بهم متصل شدند. در خط آخر هم گره های u و v قرار دارد که الگوریتم شما باید چک کند آیا در گراف داده شده مسیری از گرهی u به گرهی v وجود دارد یا خیر. اگر مسیری وجود داشت خروجی الگوریتم یک می باشد و در غیر این صورت صفر.

ورودی نمونه	خروجی نمونه
4 1 2 3 2 4 3 1 4 1 4	1



همانطور که در شکل بالا معلوم است دو مسیر از گره ۱ به گره ۴ وجود دارد: ۱-۴ و ۱-۲-۳-۴.

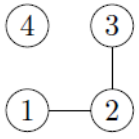
[۱] Finding an Exit from a Maze

```cs
using System;
using System.Collections.Generic;
using TestCommon;

namespace A12
{
    public class Q1MazeExit : Processor
    {
        public Q1MazeExit(string testDataName) : base(testDataName) { }

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

        public long Solve(long nodeCount, long[][] edges,
                long StartNode, long EndNode)
        {
            throw new NotImplementedException();
        }
     }
}
```

۲ اضافه کردن راه خروج [۲]

فرضیات سوال قبل را در نظر بگیرید. در این سوال شما باید مطمئن شوید که در یک maze هیچ منطقه‌ی مرده ای وجود ندارد. یعنی حداقل یک مسیر خروج از هر خانه وجود دارد که ما را به بیرون از Maze می رساند. برای این کار، اجزای متصل در گراف غیر جهت دار را پیدا کنید و اطمینان حاصل کنید که در هر جز متصل، یک راه خروج به بیرون از maze وجود دارد. در این سوال یک گراف با n گره به شما داده می شود و شما باید تعداد اجزای متصل این گراف را پیدا کنید.
خط اول فایل ورودی، تعداد گره های گراف(یعنی n) را مشخص می کند. هر یک از n خط بعدی، شامل دو گره است که بدین معنی است که این دو گره توسط یک یال بهم متصل شدند. در فایل خروجی هم یک عدد که نشان دهندهی تعداد اجزای متصل گراف است، می باشد.

ورودی نمونه	خروجی نمونه
4 1 2 3 2	2



همان طور که در شکل بالا مشخص است، تعداد اجزای متصل در این گراف دو تا می باشد.

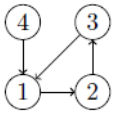
[۲] Adding Exists to a Maze

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

namespace A12
{
    public class Q2AddExitToMaze : Processor
    {
        public Q2AddExitToMaze(string testDataName) : base(testDataName) { }

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

        public long Solve(long nodeCount, long[][] edges)
        {
            throw new NotImplementedException();
        }
    }
}
```

۳ بررسی چارت درسی رشته کامپیوتر [۳]

فرض کنید که برنامه درسی رشته علوم کامپیوتر را به شما داده اند که لیستی از درس های این رشته است که پیش نیاز هر درس را هم مشخص می کند. وظیفهی شما این است که در این لیست چک کنید که وابستگی درس ها بهم ایجاد دور یا cycle نمی کند. بدین منظور، یک گراف جهت دار در نظر بگیرید که گره های این گراف درس ها هستند و یال ها وابستگی درس ها به یکدیگر را نشان می دهند. یعنی اگر درس u پیش نیاز درس v باشد، از راس u به راس v یک یال جهتدار در گراف متناظر وجود دارد. پس از ساخت چنین گرافی شما کافی است چک کنید که در این گراف دور وجود نداشته باشد.
خط اول ورودی، تعداد گره های گراف را مشخص می کند و در هر یک از خط های بعدی، دو راس وجود دارد که نشان دهندهی وجود یک یال جهتدار از راس اول به راس دوم می باشد. اگر در گراف دور وجود داشته باشد، خروجی ۱ و در غیر این صورت صفر می باشد.

ورودی نمونه	خروجی نمونه
4 1 2 4 1 2 3 3 1	1



[۳] Checking Consistency of CS Curriculum

همانطور که در شکل بالا معلوم است، دور ۳۱۲۳ در گراف وجود دارد. بنابراین خروجی یک خواهد بود.

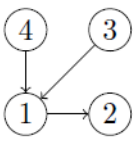
```
using System;
using System.Collections.Generic;
using TestCommon;

namespace A12
{
    public class Q3Acyclic : Processor
    {
        public Q3Acyclic(string testDataName) : base(testDataName) { }

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

        public long Solve(long nodeCount, long[][] edges)
        {
            throw new NotImplementedException();
        }
    }
}
```

۴ تعیین یک برنامه درسی معتبر [۴]

حال که مطمئن شدید که در برنامه‌ی درسی علوم کامپیوتر، وابستگی های چرخه ای وجود ندارد، وظیفه‌ی شما این است که ترتیبی را برای اخذ درس ها، با رعایت پیش نیاز ها پیشنهاد دهید که تمامی درس ها در این ترتیب وجود داشته باشد. برای این کار کافی است همان گراف جهتدار سوال قبل را در نظر بگیرید و یکی از توپولوژی های این گراف را پیدا کنید.
خط اول ورودی، تعداد گره های گراف را مشخص می کند و در هر یک از خط های بعدی، دو راس وجود دارد که نشان دهندهی وجود یک یال جهتدار از راس اول به راس دوم می باشد. تمامی گراف های تست کیس ها یک گراف جهت دار بدون دور هستند(DAG). در فایل خروجی هم، توپولوژی گراف داده شده، می باشد.

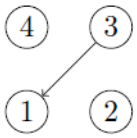
برای این سوال جواب ارائه شده فقط یک راه حل معتبر است. کد شما با متدی که داخل کلاس Q4 پیاده سازی شده راست آزمایی می شود. لطفا این کد را تغییر ندهید

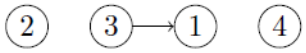
ورودی نمونه	خروجی نمونه
4 1 2 4 1 3 1	4 3 1 2



[۴] Determining an Order of Courses

ورودی نمونه	خروجی نمونه
4 3 1	2 3 1 4





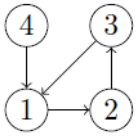

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using TestCommon;

namespace A12
{
    public class Q4OrderOfCourse: Processor
    {
        public Q4OrderOfCourse(string testDataName) : base(testDataName) { }

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

        public long[] Solve(long nodeCount, long[][] edges)
        {
            throw new NotImplementedException();
        }

        public override Action<string, string>
                Verifier { get; set; } = TopSortVerifier;

        public static void TopSortVerifier(string inFileName, string strResult)
        {
            long[] topOrder = strResult.Split(TestTools.IgnoreChars)
                .Select(x => long.Parse(x)).ToArray();

            long count;
            long[][] edges;
            TestTools.ParseGraph(File.ReadAllText(inFileName),
                        out count, out edges);

// Build an array for looking up the position of each node in topological order
// for example if topological order is 2 3 4 1, topOrderPositions[2] = 0,
// because 2 is first in topological order.
            long[] topOrderPositions = new long[count];
            for (int i = 0; i < topOrder.Length; i++)
                topOrderPositions[topOrder[i] - 1] = i;
// Top Order nodes is 1 based (not zero based).

```

```
// Make sure all direct depedencies (edges) of the graph are met:
//   For all directed edges u -> v, u appears before v in the list
            foreach (var edge in edges)
                if (topOrderPositions[edge[0] - 1] >= topOrderPositions[edge[1] - 1])
                    throw new InvalidDataException(
                        $"{Path.GetFileName(inFileName)}: " +
                        $"Edge dependency violoation: {edge[0]}->{edge[1]}");

        }
    }
}
```

۵ بررسی دسترسی به تمام چهارراه‌های یک شهر [۵]

فرض کنید که اداره پلیس یک شهر همه خیابان ها را یک طرفه کرده است. شما باید بررسی کنید که آیا هنوز هم می‌توان به صورت قانونی از هر تقاطع به هر تقاطع دیگر رفت و رانندگی کرد. بدین منظور، یک گراف جهت دار ایجاد کنید که رأسها تقاطع باشند و هر زمان که یک خیابان (یک طرفه) از u تا v در شهر وجود داشته باشد یک یال بین راس های u و v باشد. سپس، کافی است که بررسی کنید که آیا تمام رأس های گراف در یک مولفه قویا همبند[۶] هستند یا خیر.
خط اول ورودی، تعداد گره‌های گراف را مشخص می کند و در هر یک از خط های بعدی، دو راس وجود دارد که نشان دهنده‌وجود یک یال جهتدار از راس اول به راس دوم می باشد. در فایل خروجی هم تعداد مولفه‌های قویا همبند در گراف داده شده است.

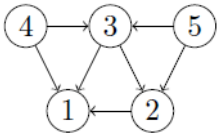
ورودی نمونه	خروجی نمونه
4 1 2 4 1 2 3 3 1	2



همانطور که در شکل بالا مشاهده می کنید، این گراف دارای دو مولفه قویا همبند می باشد: (۱و۲و۳) - (۴)

[۵] City a in Intersections of Reachability Checking
[۶] Component Connected Strongly

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



همانطور که در شکل بالا مشاهده می کنید، این گراف دارای ۵ مولفه قویا همبند می باشد: (۱) - (۲) - (۳) - (۴) - (۵)

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

namespace A12
{
    public class Q5StronglyConnected: Processor
    {
        public Q5StronglyConnected(string testDataName) : base(testDataName) { }

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

        public long Solve(long nodeCount, long[][] edges)
        {
            throw new NotImplementedException();
        }
    }
}
```