دانشکده مهندسی کامپیوتر

ساختمان داده

تمرین ۵*

مبین داریوش همدانی
بابک بهکام
سید صالح اعتمادی

نیم‌سال اول ۱۴۰۱-۱۴۰۰

ایمیل/تیمز	m_dariushhamedani@comp.iust.ac.ir babak_behkamkia@comp.iust.ac.ir
نام شاخه	fb_A5
نام پروژه/پوشه/پول ریکوست	A5
مهلت تحویل	۱۴۰۰/۸/۸

*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 A5 بسازید. همچنین پروژه تست متناظر آن را ساخته و مطابق راهنمای تمرین یک فایل‌ها را در پوشه متناظر اضافه کرده و تنظیمات مربوط به کپی کردن TestData به پوشه خروجی را در تنظیمات پروژه تست قرار دهید. دقت کنید که پروژه TestCommon فقط یکبار باید در ریشه گیت موجود باشد و نباید در هر تمرین مجدد کپی شود. برای روش ارجاع به این پروژه به تمرین شماره یک مراجعه کنید.

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.

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 TestCommon;

namespace A5.Tests
{
    [DeploymentItem("TestData")]
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q1BinarySearch()
        {
            RunTest(new Q1BinarySearch("TD1"));
        }
        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q2MajorityElement()
        {
            RunTest(new Q2MajorityElement("TD2"));
        }
        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q3ImprovingQuickSort()
        {
            RunTest(new Q3ImprovingQuickSort("TD3"));
        }
        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q4NumberOfInversions()
        {
            RunTest(new Q4NumberOfInversions("TD4"));
        }
        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q5OrganizingLottery()
        {
            RunTest(new Q5OrganizingLottery("TD5"));
        }
        [TestMethod(), Timeout(1000)]
        public void SolveTest_Q6ClosestPoints()
        {
            RunTest(new Q6ClosestPoints ("TD6"));
        }

        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("A5", p.Process, p.TestDataName, p.Verifier, VerifyResultWitho
                excludedTestCases: p.ExcludedTestCases);
        }
    }
}
```

۱ Binary Search

فرض کنید که دو آرایه ی a و b به طول n در اختیار دارید. شما باید بررسی کنید که آیا هر یک از عنصر های آرایه ی b در آرایه a موجود است یا خیر. اگر در آرایه ی a باشد شما باید Index آن را در آرایه ی a به عنوان خروجی برگردانید و اگر در آرایه ی a نباشد عدد ۱- را برگردانید. دقت کنید که آرایه ی a به صورت صعودی است و همه ی عناصر آن از یکدیگر متمایز هستند. به مثال زیر توجه کنید:

$a : 1, 5, 8, 12, 13$
$b : 8, 1, 23, 1, 11$

در مثال بالا شما می بینید که ۸ و ۱ در آرایه ی a وجود دارند و Index آن ها به ترتیب ۲ و ۰ می باشد ولی ۱۱ و ۲۳ در a وجود ندارد. بنابراین خروجی به صورت زیر خواهد بود:

$2, 0, -1, 0, -1$

- محدودیت زمانی : ۱۰۰۰ میلی ثانیه

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

namespace A5
{
    public class Q1BinarySearch : Processor
    {
        public Q1BinarySearch(string testDataName) : base(testDataName)
        { }

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


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

۲ Majority Element

فرض کنید که دنباله ای از اعداد به صورت $a_1, a_2, ..., a_n$ داریم. شما باید یک الگوریتم
divide and conquer
بنویسید که چک کند آیا عنصری در دنباله وجود دارد که بیش از $n/2$ بار تکرار شده باشد یا خیر. اگر چنین عنصری وجود داشته باشد، به آن majority element می گوییم و شما در این حالت بایدد عدد یک را برگردانید در غیر این صورت عدد صفر را برگردانید.
در testcase های این سوال خط اول تعداد المان های دنباله و خطوط بعدی المان های آرایه هستند. و خروجی هم عدد یک یا صفر است. به مثال زیر توجه کنید:

5
2, 3, 9, 2, 2
output: 1

در این دنباله عدد ۲، majority element است. راهنمایی: باید الگوریتمی با پیچیدگی زمانی $O(nlogn)$ طراحی کنید. در نظر بگیرید که اگر یک دنباله یک majority element داشته باشد، این عدد، majority element یک از دو نیمه اول یا دوم دنباله نیز هست. پس برای حل مسئله، باید دنباله را به دو نیمه تقسیم کنید و بعد از بررسی آن ها به صورت بازگشتی، نتایج را جمع بندی کنید.

- محدودیت زمانی : ۱۰۰۰ میلی ثانیه

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

namespace A5
{
    public class Q2MajorityElement:Processor
    {

        public Q2MajorityElement(string testDataName) : base(testDataName)
        { }

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


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

۳ Improving Quick Sort

در کلاس درس شما پیاده سازی الگوریتم Quick Sort را دیدید. اکنون شما باید این الگوریتم را به گونه ای تغییر دهید که برای آرایه هایی که تعداد المان های مساوی زیادی دارند هم، سریع عمل کند. برای راهنمایی شما باید به جای ۲ partition بندی، ۳ partition بندی داشته باشید.
testcase های این تمرین هم به این شکل است که در فایل ورودی خط اول تعدا المان های آرایه و در خطوط بعدی هر یک از المان های آرایه قرار دارد.و در فایل خروجی هم آرایه ی مرتب شده می باشد.

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

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

namespace A5
{
    public class Q3ImprovingQuickSort:Processor
    {
        public Q3ImprovingQuickSort(string testDataName) : base(testDataName)
        { }

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

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

۴ Number of Inversions

inversion در یک دنباله از اعداد مانند $a_0, a_1, ..., a_n$ یعنی به ازای $0 <= i < j < n$ رابطه ی $a_i > a_j$ برقرار باشد. شما در این سوال باید تعداد inversion های آرایه ی ورودی را پیداکنید. تعداد Inversion یک آرایه مشخص می کند که چقدر یک آرایه مرتب شده است. یعنی در واقع در یک آرایه ی نزولی تعداد inversion ها صفر می باشد.

testcase های این تمرین هم به این شکل است که در فایل ورودی، خط اول تعدا المان های آرایه و خطوط بعدی المان های آرایه می باشد و در فایل خروجی یک عدد می باشد که برابر با تعداد inversion های آرایه است.

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

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


namespace A5
{
    public class Q4NumberOfInversions:Processor
    {

        public Q4NumberOfInversions(string testDataName) : base(testDataName)
        { }

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

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

۵ Organizing a Lottery

فرض کنید سازماندهی یک لاتاری آنلاین به شما واگذار شده است. برای شرکت در مسابقه، هر شرکت کننده یک عدد صحیح را انتخاب می کند. سپس شما به صورت تصادفی چندین بازه را تعریف می کنید. امتیاز هر شرکت کننده در لاتاری متناسب با تعداد بازه هایی است که شامل عدد مورد نظر شرکت کننده است منهای تعداد محدوده هایی است که آن را شامل نمی شود. شما نیاز به یک الگوریتم کارآمد برای محاسبه امتیاز هر شرکت کننده دارید. Naive Algorithm برای حل این سوال این است که برای همه شرکت کنندگان همه بازه ها را اسکن کنید. اما در این قرعه کشی هزاران نفر از شرکت کنندگان و هزاران محدوده وجود دارد. به همین دلیل شما به یک الگوریتم سریع نیاز دارید.

صورت مسئله به بیان ریاضی به صورت زیر است:

فرض کنید دنباله ای به طول n از اعداد در اختیار دارید که هر کدام از المان های این دنباله یک نقطه هستند و دنباله ی دیگری هم به طول m داریم که شامل بازه ای از اعداد یا segment می باشد. شما باید یک الگوریتمی بنویسید که برای هر نقطه، تعداد segment هایی که شامل آن نقطه می شود را خروجی بدهد.

فرمت testcase های این سوال به این صورت است که در فایل ورودی، خط اول شامل دنباله نقطه هاست و در هر یک از خطوط بعدی، عدد اول شروع بازه و عدد دوم پایان بازه است. فایل خروجی هم تعداد segment هایی که شامل یک نقطه می شود؛ را نشان می دهد.

- محدودیت زمانی : ۱۰۰۰ میلی ثانیه

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

namespace A5
{
    public class Q5OrganizingLottery:Processor
    {
        public Q5OrganizingLottery(string testDataName) : base(testDataName)
        {}
        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long[], long[], long[], long[]>)Solve);

        public virtual long[] Solve(long[] points, long[] startSegments, long[] endSegment)
        {
            //write your code here
            throw new NotImplementedException();
        }
    }
}
```

۶ Closest Points

فرض کنید n تا نقطه داریم. شما باید در این مجموعه دو نقطه را پیدا کنید که نزدیک ترین فاصله را از یکدیگر دارند.
این مسئله یکی از کاربردی ترین مسائل در حوزه ی گرافیک، بینایی کامپیوتر و کنترل ترافیک است.
فرمت testcase های این سوال به این صورت است که در فایل ورودی در خط اول تعداد نقطه ها و در خطوط بعدی به ترتیب عدد اول،x نقطه و عدد دوم y نقطه است. فایل خروجی هم شامل یک عدد است که همان نزدیک ترین فاصله است.

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

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

namespace A5
{
    public class Q6ClosestPoints : Processor
    {
        public Q6ClosestPoints(string testDataName) : base(testDataName)
        { }
        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long[], long[], long[], long[]>)Solve);

        public virtual long[] Solve(long[] points, long[] startSegments, long[] endSegment)
        {
            //write your code here
            throw new NotImplementedException();
        }
    }
}
```