تمرین ۱۱ درس ساختمان داده

مریم سادات هاشمی
سید صالح اعتمادی
دانشگاه علم و صنعت ۹۸_۹۷

لطفا به نکات زیر توجه کنید:

- مهلت ارسال این تمرین شنبه ۱ دی ماه ساعت ۱۱:۵۹ ب.ظ است.
- این تمرین شامل سوال های برنامه نویسی می باشد، بنابراین توجه کنید که حتماً موارد خواسته شده را رعایت کنید. .
- نام شاخه، پوشه و پول ریکوست همگی دقیقا "A۱۱" باشد.
- در صورتی که به اطلاعات بیشتری نیاز دارید می توانید با ایدی تلگرام @maryam_sadat_hashemi در ارتباط باشید.
- اگر در حل تمرین شماره ی ۱۱ مشکلی داشتید،لطفا به این لینک مراجعه کنید و زمانی را برای رفع اشکال تنظیم کنید.

موفق باشید.

توضیحات کلی تمرین

تمرین این هفته ی شما، ۵ سوال دارد که باید به همه ی این سوال ها پاسخ دهید. برای حل این سری از تمرین ها مراحل زیر را انجام دهید:

۱. ابتدا مانند تمرین های قبل، یک پروژه به نام A۱۱ بسازید.

۲. کلاس هر سوال را به پروژه ی 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 پروژه ی خود بسازید.

۲. فولدر TestData که در ضمیمه همین فایل قرار دارد را به پروژه ی تست خود اضافه کنید.

۳. فایل GradedTests.cs را به پروژه ی تستی که ساخته اید اضافه کنید. توجه کنید که مانند تمرین های قبل، لازم نیست که برای هر سوال TestMethod بنویسید. تمامی آنچه که برای تست هر سوالتان نیاز دارید از قبل در این فایل برای شما پیاده سازی شده است.

```
namespace A11.Tests
{
    [TestClass()]
    – references | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
    public class GradedTests
    {
        [TestMethod(), Timeout(2000)]
        [DeploymentItem("TestData", "A11_TestData")]
        – references | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public void SolveTest()
        {
            Processor[] problems = new Processor[] {
                new BinaryTreeTraversals("TD1"),
                new IsItBST("TD2"),
                new IsItBSTHard("TD3"),
                new SetWithRageSums("TD4"),
                new Rope("TD5")
            };

            foreach (var p in problems)
            {
                TestTools.RunLocalTest("A11", p.Process, p.TestDataName);
            }
        }
    }
}
```

دقت کنید که TestCommon تغییر یافته است. بنابراین شما باید نسخه ی جدید آن را با دستورgit Pull دریافت کنید .

۱ Binary tree traversals

در این سوال به شما یک درخت دودویی داده می شود که باید پیمایش های in-order و pre-order و post-order برای این درخت را در خروجی نمایش دهید.

برای حل این سوال می توانید از توضیحات داخل ویدئو های درس استفاده کنید. اما دقت کنید که پیمایش های درخت را نباید بازگشتی پیاده سازی کنید. در این صورت زمان اجرای برنامه ی شما زیاد خواهد شد یا با ارور StackOverFlow مواجه خواهید شد.

گره های درخت دودویی از ایندکس ۰ تا $n-1$ شماره گذاری شده اند و گره با ایندکس ۰ ریشه ی درخت می باشد. همچنین شماره ی هر خط از فایل ورودی همان ایندکس گره می باشد. یعنی در خط اول از فایل ورودی گره با ایندکس صفر است که همان ریشه درخت نیز می باشد. خط دوم گره با ایندکس یک و خط سوم گره با ایندکس دو و الی آخر.

فرمت هر خط از وروردی به صورت زیر است:

key_i , $left_i$, $right_i$

که در واقع بیان کننده ی این است که$left_i$ و $right_i$ فرزند های چپ و راست گره key_i هستند. اگر گره ی key_i فرزند چپ یا راست نداشته باشد، $left_i$ یا $right_i$ برای این گره برابر ۱ـ خواهد بود. در غیر این صورت ایندکس فرزند های چپ و راست خواهد بود. تمامی ورودی ها یک درخت دودویی می باشند.

خروجی سه خط می باشد که فرمت آن به صورت زیر می باشد:

- خط اول پیمایش in-order
- خط دوم پیمایش pre-order
- خط سوم پیمایش post-order

ورودی:

```
4 1 2
2 3 4
5 -1 -1
1 -1 -1
3 -1 -1
```

خروجی:

```
1 2 3 4 5
4 2 1 3 5
1 3 2 5 4
```

```
using System;
using TestCommon;

namespace A11
{
    1 reference | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
    public class BinaryTreeTraversals : Processor
    {
        0 references | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public BinaryTreeTraversals(string testDataName) : base(testDataName) { }

        1 reference | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long[][], long[][]>)Solve);


        1 reference | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public long[][] Solve(long[][] nodes)
        {
            return new long[][] { };
        }
    }
}
```

۲ Is it a binary search tree؟

فرض کنید ساختار داده ای درخت جستجوی دودویی را از کتابخانه های زبان های برنامه نویسی مختلف، به شما داده اند و از شما خواستند که بررسی کنید، آیا این ساختار داده به درستی پیاده سازی شده است یا خیر. در حال حاضر یک برنامه وجود دارد که با عملیات های قرار دادن(Insert)، حذف(remove)، جستجو(search) اعداد صحیح را در این ساختار داده قرار می دهد و یک درخت جستجوی دودویی را می سازد و وضعیت های داخلی این درخت را خروجی می دهد. حالا شما باید تست کنید که آیا درخت ساخته شده یک درخت جستجو دودویی است یا خیر . تعریف درخت جستجوی دودویی به صورت زیر است:

به ازای هر گره از درخت که مقدار آن x باشد، در این صورت به ازای هر گره ای که در زیر درخت سمت چپ وجود دارد، مقدارش باید کمتر از x باشد و به ازای هر گره ای که در زیر درخت سمت راست وجود دارد مقدارش باید بزرگتر از x باشد. بنابراین شما باید چک کنید که این شرط ها برای درخت جستجوی دودویی داده شده برقرار است یا خیر.

دقت کنید که اگر این شروط را برای هر گره و گره های زیر درختش به صورت بازگشتی چک کنید، بسیار کند خواهد بود و مدت زمان زیادی طول خواهد کشید. بنابراین شما باید یک الگوریتم سریعتر برای حل این سوال پیدا کنید.

تضمین می شود که داده های ورودی همگی یک درخت دودویی هستند یعنی ورودی یک درخت است و هر گره حداکثر دو فرزند دارد.

مانند سوال قبل، گره های درخت دودویی از ایندکس ۰ تا $n-1$ شماره گذاری شده اند و گره با ایندکس ۰ ریشه ی درخت می باشد. همچنین شماره ی هر خط از فایل ورودی همان ایندکس گره می باشد. یعنی در خط اول از فایل ورودی گره با ایندکس صفر است که همان ریشه درخت نیز می باشد. خط دوم گره با ایندکس یک و خط سوم گره با ایندکس دو و الی آخر.

فرمت هر خط از وروردی به صورت زیر می باشد:

key_i , $left_i$, $right_i$

که در واقع بیان کننده ی این است که$left_i$ و $right_i$ فرزند های چپ و راست گره key_i هستند. اگر گره ی key_i فرزند چپ یا راست نداشته باشد، $left_i$ یا $right_i$ برای این گره برابر ۱ـ خواهد بود. در غیر این صورت ایندکس فرزند های چپ و راست خواهد بود. تمامی ورودی ها یک درخت دودویی می باشند.

ورودی:

```
2 1 2
1 -1 -1
3 -1 -1
```

خروجی:

```
True
```

```
using System;
using TestCommon;

namespace A11
{
    1 reference | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
    public class IsItBST : Processor
    {
        0 references | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public IsItBST(string testDataName) : base(testDataName) { }

        5 references | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long[][], bool>)Solve);


        1 reference | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public bool Solve(long[][] nodes)
        {
            return false;
        }
    }
}
```

۳ Is it a binary search tree؟ Hard version

در این سوال شما باید همان سوال قبلی را حل کنید اما در حالت کلی تر. یعنی در درخت جستجوی دودویی ممکن است گره هایی با مقدار های یکسان وجود داشته باشد. بنابراین تعریف درخت جستجوی دودویی به صورت زیر تغییر خواهد یافت:

به ازای هر گره از درخت که مقدار آن x باشد، در این صورت به ازای هر گره ای که در زیر درخت سمت چپ وجود دارد، مقدارش باید کمتر از x باشد و به ازای هر گره ای که در زیر درخت سمت راست وجود دارد مقدارش باید بزرگتر از x یا مساوی آن باشد. یعنی مقدار های مساوی همیشه در زیردرخت راست قرار می گیرند. بنابراین شما باید چک کنید که این شرط ها برای درخت جستجوی دودویی داده شده برقرار است یا خیر.

تضمین می شود که داده های ورودی همگی یک درخت دودویی هستند یعنی ورودی یک درخت است و هر گره حداکثر دو فرزند دارد. گره های درخت دودویی از ایندکس ۰ تا $n-1$ شماره گذاری شده اند و گره با ایندکس ۰ ریشه ی درخت می باشد. همچنین شماره ی هر خط از فایل ورودی همان ایندکس گره می باشد. یعنی در خط اول از فایل ورودی گره با ایندکس صفر است که همان ریشه درخت نیز می باشد. خط دوم گره با ایندکس یک و خط سوم گره با ایندکس دو و الی آخر.

فرمت هر خط از وروردی به صورت زیر می باشد:

key_i , $left_i$, $right_i$

که در واقع بیان کننده ی این است که$left_i$ و $right_i$ فرزند های چپ و راست گره key_i هستند. اگر گره ی key_i فرزند چپ یا راست نداشته باشد، $left_i$ یا $right_i$ برای این گره برابر ۱ـ خواهد بود. در غیر این صورت ایندکس فرزند های چپ و راست خواهد بود. تمامی ورودی ها یک درخت دودویی می باشند.

اگر درخت داده شده یک درخت جستجوی دودویی باشد خروجی True و در غیر این صورت False می باشد.

برای حل این سوال کافی است الگوریتم سوال قبل را اندکی تغییر دهید :)

ورودی:

```
2 1 2
2 -1 -1
3 -1 -1
```

خروجی:

```
0
```

```
using System;
using TestCommon;

namespace A11
{
    1 reference | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
    public class IsItBSTHard : Processor
    {
        0 references | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public IsItBSTHard(string testDataName) : base(testDataName) { }

        5 references | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long[][], bool>)Solve);


        1 reference | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public bool Solve(long[][] nodes)
        {
            return false;
        }
    }
}
```

۴ Set with range sums

در این سوال شما باید یک ساختار داده ای را پیاده سازی کنید که مجموعه ای از اعداد صحیح S را با عملیات مجاز زیر ذخیره کند:

- add(i) : این بدان معنی است که عدد صحیح i را به مجموعه ی S اضافه کنید. اگر در مجموعه از قبل وجود دارد، مجموعه نباید تغییر کند.
- del(i) : این بدان معنی است که عدد صحیح i را از مجموعه ی S حذف کنید. اگر در مجموعه وجود ندارد، اتفاق خاصی نمی افتد.
- find(i) : این بدان معنی است که بررسی کنید آیا عدد صحیح i در مجموعه ی S وجود دارد یا خیر.
- sum(l،r) : این بدان معنی است که مجموع همه ی المان های v را بدست آورید و خروجی دهید که $l <= v <= r$

فرض کنید که در ابتدا مجموعه ی S خالی است. همچنین اگرعملیات sum انجام شده باشد،x برابر با نتیجه آخرین عملیات sum است. در غیر این صورت برابر صفر خواهد بود. فرمت ورودی به صورت زیر است:

- "+ i" means add((i + x) mod M)
- "- i" means del((i + x) mod M)
- "? i" means find((i + x) mod M)
- "s l r" means sum((l + x) mod M, (r + x) mod M)

در خروجی شما باید نتیجه ی عملیات find و sum را پس بدهید. برای عملیات find اگر عدد $(i + x)modM$ را در مجموعه ی S پیدا کردید باید رشته ی Found و در غیر این صورت Not found را برگردانید. همچنین برای عملیات sum شما باید مجموع مقدار های v را برگردانید که $(l + x)modM <= v <= (r + x)modM$.

ورودی:

```
? 1
+ 1
? 1
+ 2
s 1 2
+ 1000000000
? 1000000000
- 1000000000
? 1000000000
s 999999999 1000000000
- 2
? 2
- 0
+ 9
s 0 9
```

خروجی:

```
Not found
Found
3
Found
Not found
1
Not found
10
```

به نمونه ی بالا توجه کنید. در ۵ درخواست اول مقدار $x = 0$ می باشدو در ۵ درخواست دوم مقدار $x = 5$ و در ۵ درخواست سوم وروردی مقدار $x = 1$ می باشد. بنابراین عملیات های واقعی ورودی به صورت زیر خواهند بود:

$\textbf{find}(1)$
$\textbf{add}(1)$
$\textbf{find}(1)$
$\textbf{add}(2)$
$\textbf{sum}(1,2) \to 3$
$\textbf{add}(2)$
$\textbf{find}(2) \to \text{Found}$
$\textbf{del}(2)$
$\textbf{find}(2) \to \text{Not found}$
$\textbf{sum}(1,2) \to 1$
$\textbf{del}(3)$
$\textbf{find}(3) \to \text{Not found}$
$\textbf{del}(1)$
$\textbf{add}(10)$
$\textbf{sum}(1,10) \to 10$

پیاده سازی ساده و ابتدایی این سوال برای شما انجام شده است. اما این پیاده سازی صرفا بار آموزشی دارد و در عمل کند است و زمان زیادی طول خواهد کشید تا برنامه اجرا شود. برای حل این سوال شما باید یک splay tree را پیاده سازی کنید. برای اینکه بخواهید مواردی را بر روی splay tree به صورت تصویری تست کنید، می توانید از این لینک استفاده کنید.

```
using System;
using TestCommon;

namespace A11
{
    public class SetWithRageSums : Processor
    {
        public SetWithRageSums(string testDataName) : base(testDataName) { }

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


        public string[] Solve(string[] nodes)
        {
            return new string[] { };
        }
    }
}
```

Rope ۵

در این سوال شما باید ساختار داده rope را پیاده سازی کنید که می تواند یک رشته را ذخیره کند و بخشی از این رشته (زیر رشته) را به طور موثری برش دهد و آن را در جای دیگری از رشته قرار دهد.

این یک سوال بسیار پیشرفته است، سخت تر از تمام سوالات پیشرفته ای که تا به حال در تمرین ها داشتیم بنابراین هر چه زودتر دست به کار شوید و این سوال را حل کنید.

فرض کنید که رشته s را به شما داده اند و شما باید n تا query را بر روی این رشته پردازش کنید. هر query توسط سه عدد صحیح توصیف می شود: i ، j و k و به معنای آین است که زیر رشته $s[i...j]$ را از رشته S برش دهید و سپس آن را پس از k امین حرف در رشته باقی مانده درج کنید. اگر k برابر صفر باشد، زیر رشته در ابتدا رشته قرار داده می شود.

در خط اول ورودی رشته ی s قرار دارد و در n خط بعدی query هایی که باید بر روی رشته ی s انجام بدهید، قرار دارد. هر query با سه عدد توصیف می شود که در بالا توضیح داده شد.). پس از این که تمام query ها بر روی رشته ی s انجام شد، باید نتیجه را در خروجی چاپ نمایید.

ورودی:

```
hlelowrold
1 1 2
6 6 7
```

خروجی:

```
helloworld
```

پیاده سازی ساده و ابتدایی این سوال نیز برای شما انجام شده است ولی بسیار کند است و زمان زیادی طول می کشد. برای حل این سوال نیز شما باید از splay tree استفاده کنید. فکر کنید که چه چیزی باید به عنوان key در درخت ذخیره شود.

```
using System;
using TestCommon;

namespace A11
{
    1 reference | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
    public class Rope : Processor
    {
        0 references | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public Rope(string testDataName) : base(testDataName) { }

        5 references | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<string, long[][], string>)Solve);


        1 reference | Sauleh Eetemadi, 2 days ago | 1 author, 1 change
        public string Solve(string text, long[][] nodes)
        {
            return null;
        }
    }
}
```