تمرین ۱۴ 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 ۲۴ آذر 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 "A۱۰" 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 ۱۰ مشکلی داشته اید،لطفا به `https://calendly.com/hashemi-maryam-sadat` مراجعه کنید و زمانی را برای رفع اشکال تنظیم کنید.

موفق باشید.

توضیحات کلی تمرین

تمرین این هفته ی شما، ۳ 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 A۱۰ 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.

توجه کنید که سوال ۱ و ۲ دارای متد های دیگری غیر از دو متد بالاست که حتما شما باید بدنه ی این متد ها را بنویسید.

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.

```
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
{
    [TestClass()]
    – references | Sauleh Eetemadi, 8 days ago | 1 author, 1 change
    public class GradedTests
    {
        [TestMethod(), Timeout(1000)]
        [DeploymentItem("TestData", "A10_TestData")]
        – references | Sauleh Eetemadi, 8 days ago | 1 author, 1 change
        public void SolveTest()
        {
            Processor[] problems = new Processor[] {
                new PhoneBook("TD1"),
                new HashingWithChain("TD2"),
                new RabinKarp("TD3")
            };

            foreach (var p in problems)
            {
                TestTools.RunLocalTest("A10", 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 Phone book

در این سوال شما باید یک دفترچه تلفن ساده را پیاده سازی کنید. برنامه ی شما باید بتواند درخواست های کاربر را پردازش کند. این درخواست ها به صورت زیر است:

- add number name

  این بدان معنی است که کاربر یک فرد را با نام و شماره تلفنی که وارد کرده است را به دفترچه اضافه می کند. اگر فردی با چنین شماره ای در دفترچه تلفن وجود دارد، شما باید نام متناظر را بازنویسی کند.

- del number

  این بدان معنی است که باید یک فرد با شماره تلفن ورودی را از دفترچه تلفن پاک کنید. اگر چنین شخصی وجود نداشته باشد، این درخواست را نادیده بگیرید.

- find number

  این بدان معنی است که کاربر به دنبال فردی با شماره تلفن ورودی است. شما باید نام شخص متناظر با شماره ی تلفن را برگردانید.اگر چنین شخصی در دفترچه تلفن وجود نداشت، عبارت not found را برگردانید.

در هر خط از فایل ورودی یکی از درخواست های بالا وجود دارد. در هر خط از خروجی هم پاسخ به درخواست find قرار دارد.

لطفا نمونه های ورودی و خروجی سوال را از داخل داکیومنت اصلی مطالعه فرمایید.

توجه کنید که برای راحتی شما کلاس contact را پیاده سازی کردیم. این کلاس دو ویژگی name و number دارد. شما می توانید در صورت نیاز از این کلاس استفاده کنید و یا آن را تغییر دهید.

```
public class Contact
{
    public string Name;
    public int Number;

    public Contact(string name, int number)
    {
        Name = name;
        Number = number;
    }
}
```

در این سوال شما باید تابع های Add و Delete و Find که در فایل PhoneBook.cs قرار دارد را پیاده سازی و کامل کنید.

در این سوال پیاده سازی ساده و ابتدایی برای شما انجام شده است که اگر تست کنید خواهید دید که نسبت به الگوریتم اصلی که شما باید پیاده سازی کنید بسیار کند است و بیش از اندازه طول می کشد.

```
public void Add(string name, int number)
{
    for(int i=0; i<PhoneBookList.Count; i++)
    {
        if (PhoneBookList[i].Number == number)
        {
            PhoneBookList[i].Name = name;
            return;
        }
    }
    PhoneBookList.Add(new Contact(name, number));
}
```

```
public string Find(int number)
{
    for (int i = 0; i < PhoneBookList.Count; i++)
    {
        if (PhoneBookList[i].Number == number)
            return PhoneBookList[i].Name;
    }
    return "not found";
}
```

```
public void Delete(int number)
{
    for (int i = 0; i < PhoneBookList.Count; i++)
    {
        if (PhoneBookList[i].Number == number)
        {
            PhoneBookList.RemoveAt(i);
            return;
        }
    }
}
```

```
namespace A10
{
    public class Contact
    {
        public string Name;
        public int Number;

        public Contact(string name, int number)
        {
            Name = name;
            Number = number;
        }
    }

    public class PhoneBook : Processor
    {
        public PhoneBook(string testDataName) : base(testDataName) { }

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

        protected List<Contact> PhoneBookList;

        public string[] Solve(string [] commands)...

        public void Add(string name, int number)...

        public string Find(int number)...

        public void Delete(int number)...
    }
}
```

۲ Hashing with chains

در این سوال شما باید یک جدول hash را به صورت زنجیره ای پیاده سازی کنید. Chaining یکی از رایج ترین روش های اجرای جدول های hash است. از چنین جدول hash ای می توان برای پیاده سازی یک دفترچه تلفن بر روی تلفن همراه یا ذخیره جدول رمز عبور کامپیوتر و سرویس وب استفاده کرد.

فرض کنید که تعداد bucket ها m باشد بنابراین برای پیاده سازی جدول hash به صورت زنجیره ای از تابع چند جمله ای زیر به عنوان تابع hash استفاده کنید.

$$h(S) = \left(\sum_{i=0}^{|S|-1} S[i]x^i \bmod p \right) \bmod m$$

که در این تابع $s[i]$ کد ASCII ،المان i رشته یs است و

$$\mathrm{p} = 1\ 000\ 000\ 007\ ,\ \mathrm{x} = 263$$

برنامه شما باید دستور های زیر را پشتیبانی کند:

- add string

  این بدان معنی است که رشته را به جدول وارد کنید. اگر قبلا چنین رشته ای در جدول hash وجود دارد،این درخواست را نادیده بگیرید.

- del string

  این بدان معنی است که رشته را از جدول حذف کنید. اگر چنین رشته ای در جدول hash وجود ندارد، پس درخواست را نادیده بگیرید.

- find string

  این بدان معنی است که بسته به اینکه آیا جدول شامل رشته هست یا نه، خروجی "yes" یا "no" را برگردانید .

- check i

  این بدان معنی است که محتوای i امین لیست در جدول را خروجی بدهید. با استفاده از space المان ها از هم جدا می شوند. اگر لیست i ام خالی باشد، یک خط خالی را در خروجی نمایش دهید.

هنگام وارد کردن یک رشته جدید به زنجیره hash باید آن را در ابتدای زنجیره وارد کنید.
خط اول ورودی تعداد Bucket هاست و هر یک از خطوط بعدی یکی از درخواست های بالا می باشد. هر یک از خطوط خروجی جواب درخواست های check و find به ترتیبی که این دستور ها در ورودی آمده است، می باشد.

لطفا نمونه های ورودی و خروجی سوال را از داخل داکیومنت اصلی مطالعه فرمایید.

در این سوال شما باید تابع های Add و Delete و Find و check که در فایل Hashing-WithChain.cs قرار دارد را پیاده سازی و کامل کنید. دقت کنید برخی از مواردی که برای حل این سوال نیاز دارید برای شما پیاده سازی شده است.

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

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


        public string[] Solve(long bucketCount, string[] commands)...

        public const long BigPrimeNumber = 1000000007;
        public const long ChosenX = 263;

        public static long PolyHash(
            string str, int start, int count,
            long p = BigPrimeNumber, long x = ChosenX)
        {
            long hash = 0;
            return hash;
        }

        public void Add(string str)...

        public string Find(string str)...

        public void Delete(string str)...

        public string Check(int i)...
    }
}
```

۳ Find pattern in text

در این سوال شما باید الگوریتم Rabin-Karp را برای جستجوی یک الگو در یک متن پیاده سازی کنید. در فایل ورودی دو رشته وجود دارد که رشته ی اول الگو و رشته ی دوم متن می باشد. در خروجی هم باید مکان هایی از متن که در آن الگو داده شده اتفاق افتاده است را برگردانید. یعنی:

ورودی:

```
aba
abacaba
```

همان طور که ملاحظه می کنید الگو ی aba در متن داده شده، یک بار در مکان صفر اتفاق افتاده است و بار دیگر در مکان ۴. بنابراین خروجی به صورت زیر خواهد بود:

خروجی:

```
0 4
```

لطفا ادامه ی نمونه های ورودی و خروجی سوال را از داخل داکیومنت اصلی مطالعه فرمایید.

همانطور که در شکل زیر مشاهده می کنید، در این سوال پیاده سازی ساده و ابتدایی برای شما انجام شده است. اگر این برنامه را تست کنید خواهید دید که نسبت به الگوریتم اصلی که شما باید پیاده سازی کنید بسیار کند است و بیش از اندازه طول می کشد.

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

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

        public long[] Solve(string pattern, string text)
        {
            List<long> occurrences = new List<long>();
            int startIdx = 0;
            int foundIdx = 0;
            while ((foundIdx = text.IndexOf(pattern, startIdx)) >= startIdx)
            {
                startIdx = foundIdx + 1;
                occurrences.Add(foundIdx);
            }
            return occurrences.ToArray();
        }


        public static long[] PreComputeHashes(
            string T,
            int P,
            long p,
            long x)
        {
            return null;
        }
    }
}
```