دانشکده مهندسی کامپیوتر

ساختمان داده

تمرین ۹*

مبین داریوش همدانی
بابک بهکام کیا
سید صالح اعتمادی

نیم‌سال اول ۱۴۰۱-۱۴۰۰

ایمیل/تیمز	m_dariushhamedani@comp.iust.ac.ir babak_behkamkia@comp.iust.ac.ir
نام شاخه	fb_A9
نام پروژه/پوشه/پول ریکوست	A9
مهلت تحویل	۱۴۰۰/۹/۱۳

*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 A9 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.

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

namespace A9.Tests
{
    [DeploymentItem("TestData")]
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(1500)]
        public void SolveTest_Q1ConvertIntoHeap()
        {
            RunTest(new Q1ConvertIntoHeap("TD1"));
        }

        [TestMethod(), Timeout(2000)]
        public void SolveTest_Q2MergingTables()
        {
            RunTest(new Q2MergingTables("TD2"));
        }

        [TestMethod(), Timeout(2500)]
        public void SolveTest_Q3ParallelProcessing()
        {
            RunTest(new Q4ParallelProcessing("TD3"));
        }


        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("A9", p.Process, p.TestDataName, p.Verifier,
                VerifyResultWithoutOrder: p.VerifyResultWithoutOrder,
                excludedTestCases: p.ExcludedTestCases);
        }
    }
}
```

۱ Convert array into heap

در این سوال شما باید یک آرایه از اعداد صحیح را به یک heap تبدیل کنید. این کار یک مرحله مهم از الگوریتم مرتب سازی HeapSort است. این الگوریتم تضمین می کند که در بدترین حالت، زمان اجرا $nlog(n)$ است در صورتی که در الگوریتم QuickSort زمان اجرای متوسط $nlog(n)$ است. QuickSort معمولا در عمل استفاده می شود، زیرا به طور معمول سریعتر است اما HeapSort برای مرتب سازی خارجی مورد استفاده قرار می گیرد یعنی زمانی که شما نیاز به مرتب کردن فایل هایی دارید که در حافظه کامپیوتر شما به صورت یک پارچه جا نمی شود.

وظیفه شما در این سوال این است که آرایه ای از اعداد صحیح داده شده را به یک heap تبدیل کنید. شما این کار را با اعمال تعداد معینی swap بر روی آرایه انجام می دهید. swap یک عملیات است که عناصر a_i و a_j از آرایه a را با هم جابه جا می کند. همان طور که در کلاس دیدید شما بایستی آرایه را با استفاده از $o(n)$تا swap به heap تبدیل کنید. توجه داشته باشید که شما باید از min-heap به جای max-heap دراین سوال استفاده کنید.

خط اول ورودی، یک آرایه از اعداد صحیح می باشد. در خط اول خروجی، تعداد swap های لازم برای تبدیل آرایه ی ورودی به heap می باشد و هر یک از خط های بعدی، شامل ایندکس هایی از آرایه که با هم swap شده اند می باشد. دقت کنید که ایندکس آرایه از۰ شروع می شود. همچنین هر المان از آرایه متمایز از دیگر المان های آرایه می باشد.

فرض کنید i یک شمارنده برای ایندکس های آرایه باشد و swap های لازم را بر روی آرایه برای تبدیل به heap انجام داده باشید.اگر شرط های زیر برقرار باشد؛ یعنی آرایه تبدیل به heap شده است:

1. If $2i + 1 \leq n - 1$, then $a_i < a_{2i+1}$.
2. If $2i + 2 \leq n - 1$, then $a_i < a_{2i+2}$.

ورودی نمونه	خروجی نمونه
5 5 4 3 2 1	3 1 4 0 1 1 3

ورودی نمونه	خروجی نمونه
5 1 2 3 4 5	0

۲ Merging tables

فرض کنید که n تا جدول در یک پایگاه داده ذخیره شده است. جداول از ۱ تا n شماره گذاری می شوند. تعداد ستون ها در همه جداول برابر است. هر جدول شامل چندین ردیف با داده های واقعی است یا یک لینک به جدول دیگری دارد. در ابتدا تمام جداول حاوی داده ها هستند، و جدول i دارای r_i ردیف است. شما باید m تا از عملیات های زیر را انجام دهید:

- جدول $destination_i$ را در نظر بگیرید. برای رسیدن به داده ها مسیر لینک ها را پیمایش کنید. به این معنا که:

$$\text{while } destination_i \text{ contains a symbolic link instead of real data do}$$
$$destination_i \leftarrow \texttt{symlink}(destination_i)$$

- جدول شماره $source_i$ را در نظر بگیرید و مسیر لینک ها از این جدول را به همان شیوه ای که برای جدول $destination_i$ انجام دادید؛ پیمایش کنید.
- حالا، با انجام دو عملیات بالا مطمئن هستیم که دو جدول $destination_i$ و $source_i$ داده های واقعی دارند. اگر $destination_i != source_i$ تمام سطرها را از جدول $source_i$ به جدول $destination_i$ کپی کنید، سپس جدول $source_i$ را پاک کنید و به جای داده های واقعی نماد لینک به $destination_i$ را به آن اضافه کنید.
- حداکثر سایز را در میان n تا جدول چاپ کنید (به خاطر داشته باشید که سایز جدول همان تعداد ردیف ها در جدول است). اگر جدول فقط شامل نماد لینک باشد، سایز آن ۰ است.

خط اول ورودی حاوی n تا عدد است که با فاصله از هم جدا شده اند. هر یک از این اعداد سایز جدول را مشخص می کنند. یعنی عدد اول سایز جدول ۱ و عدد دوم سایز جدول ۲ و الی آخر(توجه داشته باشید که شماره گذاری جدول ها از یک شروع می شود).سپس در هر یک از خطوط بعدی دو عدد وجود داردکه توصیف ادغام جدول ها را نشان می دهند. عدد اول جدول $destination_i$ و عدد دوم $source_i$ می باشد.
در خروجی،هر خط بیان کننده ی بزرگترین سایز همه ی جدول ها برای هر خط از ورودی که یک توصیف ادغام را بیان کرده است، می باشد.

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

۳ Parallel processing

در این سوال شما باید یک برنامه را شبیه سازی کنید که لیستی از job ها را از ورودی بگیرد و آن ها را به صورت موازی پردازش کند. سیستم های عاملی مانند لینوکس، MacOS یا ویندوز همه برنامه های ویژه ای را دارند که Schedulers نامیده می شوند که دقیقا همین کار را برای برنامه های رایانه شما انجام می دهند.

فرض کنید شما یک برنامه دارید که به صورت موازی در آمده است و از n تا thread مستقل برای پردازش لیستی از m تا Job استفاده می کند. thread ها، job ها را به ترتیبی که در ورودی داده می شوند؛ پردازش می کنند. اگر یک thread بیکار شود، بلافاصله job بعدی را از لیست می گیرد و شروع به پردازش آن می کند. توجه کنید که اگر یک thread پردازش یک job را آغاز کرده باشد، تا زمانی که پردازش آن job را تمام نکند، وقفه(Interrupt) ایجاد نمی کند یا آن را متوقف(stop) نخواهد کرد. اگر چندین thread به صورت همزمان از لیست یک job را بخواهند بگیرند، thread با شاخص(index) کوچکتر، کار را انجام می دهد. برای هر job شما دقیقا می دانید که چه مدت زمانی را هر thread لازم دارد تا این job را پردازش کند و این مدت زمان برای همه thread ها مشابه است.

تصور کنید که لیستی از job ها را به شما داده اند. در ادامه شما باید برای هر job از این لیست تعیین کنید که کدام یک از thread ها آن job را پردازش می کند و چه زمانی thread شروع به پردازش می کند.

خط اول ورودی شامل عدد صحیح n است که همان تعداد thread ها است. خط دوم شامل زمان لازم برای پردازش هر job است که بر اساس ثانیه می باشد. ترتیب زمان ها مطابق با ترتیب thread در لیست است. ایندکس thread ها از ۰ شروع می شود.

در هر خط از خروجی دو عدد وجود دارد که عدد اول ایندکس thread است که در حال انجام پردازش یک job است و عدد دوم زمان شروع انجام پردازش است. بنابراین تعداد خطوط خروجی برابر با تعداد job ها در لیست است.

ورودی نمونه	خروجی نمونه
2 5 1 2 3 4 5	0 0 1 0 0 1 1 2 0 4