امتحان عملی اول

درس طراحی و تحلیل الگوریتم

مریم سادات هاشمی
سهیل رستگار
مهسا سادات رضوی
امیر خاکپور
سید صالح اعتمادی

دانشگاه علم و صنعت ۹۸ـ۹۷

لطفا به نکات زیر توجه کنید:

- مهلت ارسال کدها برای این امتحان دوشنبه ۲ اردیبهشت ساعت ... ب.ظ است.
- این امتحان شامل سوال های برنامه نویسی است، بنابراین توجه کنید که حتماً موارد خواسته شده را رعایت کنید. .
- نام شاخه، پوشه و پول ریکوست همگی دقیقا "" باشد.

موفق باشید.

توضیحات کلی امتحان عملی

امتحان عملی شما،۲ سوال دارد که باید به هردوی 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 "Exam۱" 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.

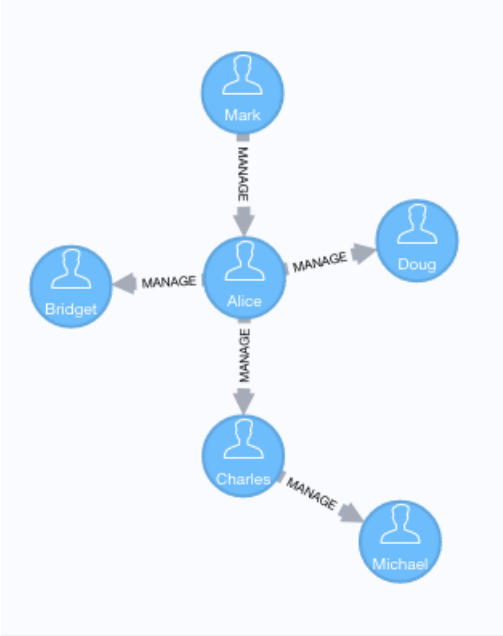
۲. فولدر TestData که در ضمیمه همین فایل قرار دارد را به پروژه ی تست خود اضافه کنید.

۳. فایل GradedTests.cs را به پروژه ی تستی که ساخته اید اضافه کنید. توجه کنید که بر خلاف تمرین‌های قبل برای هر سوال یک تست در نظر گرفته شده تا Timeout هر تست جداگانه محاسبه شود و در صورت حل نکردن هر تست بتوانید برای آن تست جداگانه Assert.Inconclusive بنویسید.

۱ Centrality Betweenness

در نظریه گراف، یک مفهوم به نام Betweenness Centrality وجود دارد که به معنای اندازه گیری مرکزیت در گراف، بر پایه کوتاهترین مسیر است. برای هر جفت رأس در یک گراف متصل، حداقل یک مسیر کوتاه بین رأس ها وجود دارد به طوری که یا تعداد یال هایی که مسیر از طریق آن عبور می کند (برای گراف های بدون وزن) و یا مجموع وزن های یال ها (برای گراف های وزن دار) به حداقل برسد. Betweenness Centrality برای هر گره برابر است با تعداد کوتاه ترین مسیرهایی که از آن گره عبور می کند.

برای مثال به گراف زیر توجه کنید.



Alice مهمترین ارتباط در این گراف است. اگر Alice حذف شود، تمام ارتباطات در گراف قطع می شود. این باعث می شود Alice مهم باشد.

در گراف بالا اگر بخواهیم betweenness Centrality را برای همه ی راس های این گراف مشخص کنیم، ابتدا باید کوتاه ترین مسیر موجود بین هر دو گره مجزا را پیدا کنیم که برای گراف بالا به صورت زیر خواهد بود:

1. Mark -> Alice
2. Mark -> Alice -> Charlrs

3. Mark -> Alice -> Doug

4. Mark -> Alice -> Bridget

5. Mark -> Alice -> Charlrs -> Micheal

6. Alice -> Charlrs

7. Alice -> Charlrs -> Micheal

حال باید ببنیم برای هر گره، چه تعداد از مسیرهای بالا از آن گره عبور کرده است تا Betweenness Centrality هر گره از گراف بدست آید.همانطور که پیدا ست ۴ مسیر ۲، ۳، ۴ و ۵ از گره Alice عبور کرده است. دو مسیر ۵ و ۷ هم از گره Charles عبور کرده است و از باقی گره ها مسیری عبور نکرده است. **دقت کنید که گره های آغازی و پایانی مسیر به عنوان گره هایی که مسیر از آن ها عبور می کند، در نظر گرفته نشده است**. پس به صورت خلاصه Betweenness Centrality هر گره را می توان در جدول زیر نمایش داد:

Name	Weight Centrality
Alice	4
Charles	2
Bridget	0
Michael	0
Doug	0
Mark	0

Table 1: Betweenness Centrality in Graph

مشاهده می کنید که Alice واسطه اصلی این شبکه است و بعد از آن Charles . دیگران هیچ تاثیری ندارند، زیرا همه کوتاه ترین مسیرها بین جفت افراد از طریق Alice یا Charles گذر می کنند.

در این سوال به شما یک گراف جهت دار بدون وزن داده می شود که در خط اول فایل ورودی عدد n قرار دارد که نشان دهنده ی تعداد گره های گراف است و در هر یک از خطوط بعدی دو گره u و v وجود دارد که نشان دهنده ی یک یال از گره u به گره v است. شما

باید برنامه ای بنویسید که Betweenness Centrality را برای هر یک از گره های این گراف بدست بیاورد و به ترتیب برای گره ۱ تا n در فایل خروجی نمایش دهد.

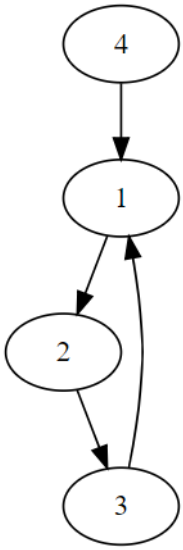
نمونه ۱

ورودی:

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

خروجی:

```
3 2 1 0
```



۲ رمزگشایی با استفاده از Pattern Matching

در این سوال متنی رمز شده به شما داده شده است که باید با پیدا کردن کلید رمزگذاری متن اصلی را پیدا کنید. الگوریتم رمزگذاری استفاده شده برای کد کردن متن به این صورت عمل می کند که کد اسکی هر کاراکتر از متن اصلی را با یک کلید مشخص که عددی صحیح است جمع کرده و باقی مانده عدد بدست آمده بر تفاضل بیشترین کد اسکی با کمترین کد اسکی کاراکتر معادل در متن رمز شده خواهد بود. برای مثال اگر کلید رمزگذاری برابر با ۱۰ و تفاضل بیشترین کد اسکی با کمترین کد اسکی کاراکترهای معادل در متن برابر ۱۲۸ باشد حرف a با کد اسکی ۹۷ به حرف k با کد اسکی ۱۰۷ تبدیل می شود.

$$97 + 10 \pmod{128} = 107$$

برای پیدا کردن کلید صحیح باید کلمات متن رمزگشایی شده با کلمات موجود در دیکشنری ای که به شما داده شده است تطبیق داده شوند. در صورت تطبیق کلمات متن بدست آمده با دیکشنری کلید بدست آمده از الگوریتم شما صحیح بوده و متن بدرستی رمزگشایی می شود. ورودی الگوریتم یک فایل متنی رمز شده به همراه دیکشنری ای از کلمات است و در فایل خروجی هش کد متن رمزگشایی شده برگردانده می شود.