

تمرین یازدهم درس تحلیل و طراحی الگوریتم

مریم سادات هاشمی
علی آل یاسین
بنفشه کریمیان
سید صالح اعتمادی

دانشگاه علم و صنعت - نیمسال دوم ۹۸-۹۷

لطفاً به نکات زیر توجه کنید:

- مهلت ارسال این تمرین سه‌شنبه ۲۱ خرداد ماه ساعت ۱۱:۵۹ ب.ظ است.
- این تمرین شامل سوال‌های برنامه‌نویسی می‌باشد، بنابراین توجه کنید که حتماً موارد خواسته شده را رعایت کنید.
- نام شاخه، پوشه و پول ریکوست همگی دقیقاً "A11" باشد.
- در صورتی که به اطلاعات بیشتری نیاز دارید می‌توانید با ایدی تلگرام @maryam_sadat_hashemi در ارتباط باشید.
- اگر در حل تمرین شماره ۴ مشکلی داشتید، لطفاً به این [لینک](#) مراجعه کنید و زمانی را برای رفع اشکال تنظیم کنید.

موفق باشید.

توضیحات کلی تمرین

تمرین این هفته ی شما، ۴ 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 A11 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 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. برای این تمرین مانند A8، A9 و A7 برای هر سوال تست جداگانه با زمان جداگانه در نظر گرفته شده. بعد از حل کردن هر تمرین Assert.Inconclusive را از ابتدای تست حذف کرده و آن را اجرا کنید.

چنانچه سوالی را حل نکردید از Assert.Inconclusive استفاده کنید.

۴. بدلیل اینکه برای سوالهای ۱، ۳ و ۴ جواب نهایی یکتا نیست، برای هر کدام متد راست‌آزمای جداگانه در TestCommon پیاده‌سازی شده است. برای دسترسی به این متدها لازم است که ابتدا شاخه جدید A11 TestCommon را در مستر merge کنید، سپس شاخه A11 را از روی شاخه مستر درست کنید.

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

۱ Integrated Circuit Design

VLSI یا *VeryLarge – ScaleIntegration* فرآیند ایجاد یک مدار یکپارچه با ترکیب هزاران ترانزیستور در یک تراشه واحد است. شما می خواهید یک لایه از یک مدار یکپارچه را طراحی کنید. شما دقیقاً می دانید کدام ماژول ها در این لایه استفاده می شود و کدام یک از آن ها باید توسط سیم ها متصل شود. سیم ها همه در یک لایه قرار می گیرند، اما آن ها نمی توانند با یکدیگر تقاطع داشته باشند. همچنین، هر سیم تنها می تواند تنها در یکی از دو جهت چپ یا راست خم شود. اگر دو ماژول را با یک سیم وصل کنید، انتخاب جهت خم شدن سیم، موقعیت سیم را تعیین می کند. شما باید موقعیت هر سیم را به گونه ای تنظیم کنید که هیچ دو سیمی با یکدیگر متقاطع نباشد.

این مسئله را می توان به مسئله ی SAT-2 کاهش داد (یک مورد خاص از مسئله ی SAT که در آن هر گزاره شامل دقیقاً ۲ متغیر است). هر سیم i را با متغیر باینری x_i نشان می دهیم که مقدار ۱ یعنی سیم به سمت راست خم شده است و مقدار ۰ یعنی سیم به سمت چپ خم شده است. همچنین، برخی از جفت سیم ها در برخی مواقع تقاطع دارند. به عنوان مثال، اگر سیم ۱ به سمت چپ خم شود و سیم ۲ به سمت راست خم شود، آن ها با یکدیگر متقاطع می شوند. ما می خواهیم فرمولی بنویسیم که تنها در صورتی که هیچ دو سیمی با یکدیگر متقاطع نباشند، satisfied شود.

در این مورد، عبارت $(x_1 \text{ OR } \bar{x}_2)$ را به فرمول اضافه می کنیم که تضمین می کند که x_1 (اولین سیم به سمت راست خم شده) درست است یا \bar{x}_2 (سیم دوم به سمت چپ خم شده) درست است. بنابراین زمانی که فرمول satisfied است، حالتی که سیم ۱ به سمت چپ خم شده و سیم ۲ به سمت راست خم شده، هرگز اتفاق نمی افتد.

پس ما برای هر دو جفت سیم و موقعیت آن ها، یک گزاره به فرمول اضافه خواهیم کرد؛ اگر این سیم ها در هنگام قرار گرفتن در این موقعیت ها متقاطع شوند. البته، اگر برخی از جفت سیم ها در هر جفت موقعیت احتمالی متقاطع باشند، در این صورت نمی توانیم یک مدار طراحی کنیم. وظیفه شما این است که مشخص کنید، آیا ممکن است یک مدار طراحی کنیم و اگر بله، جهت خم شدن برای هر سیم را تعیین کنید.

ورودی یک فرمول CNF-2 است. خط اول شامل دو عدد صحیح V و C است که به ترتیب تعداد متغیرها و تعداد گزاره ها در فرمول است. هر خط بعدی حاوی دو عدد صحیح غیر صفر i و j است که نشان دهنده ی یک جمله در فرم CNF است. اگر $i > 0$ باشد یعنی x_i و در غیر این صورت اگر $i < 0$ باشد، یعنی \bar{x}_i و همین طور برای j است. به عنوان مثال یک خط "۳ ۲" یک گزاره $(x_3 \text{ OR } x_2)$ را نشان می دهد و خط "۴ - ۱" $(\bar{x}_4 \text{ OR } x_1)$ ، خط

” ۱-۳ ” $(\overline{x_3} \text{ OR } \overline{x_1})$ و خط ” ۰۲ ” نمی تواند رخ دهد، زیرا i و j باید غیر صفر باشند.
 اگر فرمول ۲-CNF در ورودی unsatisfied باشد، فقط کلمه ” UNSATISFIABLE ” را
 خروجی می دهد. اگر فرمول ۲-CNF در ورودی satisfied باشد، کلمه ” SATISFIABLE ” را
 در خط اول و انتساب متناظر متغیرها در خط دوم نمایش داده می شود. برای هر x_i ، خروجی
 i اگر $x_i = 1$ یا $-i$ اگر $x_i = 0$. به عنوان مثال، اگر یک فرمول توسط انتساب $x_1 = 0$ ،
 $x_2 = 1$ و $x_3 = 0$ راضی باشد در این صورت خروجی به صورت ۳-۲-۱ خواهد بود.

نمونه ۱

ورودی:

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

خروجی:

```
SATISFIABLE
1 2 -3
```

نمونه ۲

ورودی:

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

خروجی:

UNSATISFIABLE

۲ Plan a Fun Party

شما در حال برنامه ریزی یک مهمانی برای اعضای یک شرکت هستید. شما می خواهید باحال ترین افراد را دعوت کنید و برای هر یک از آن ها یک فاکتور شوخ طبعی تعریف کرده اید. هرچه فاکتور شوخ طبعی بیشتر باشد، شخص باحال تر است. شما می خواهید مجموع فاکتور شوخ طبعی همه افراد دعوت شده را به حداکثر برسانید. با این حال، شما نمی توانید همه را دعوت کنید، زیرا اگر رئیس مستقیم افراد دعوت شده نیز دعوت شود، ناخوشایند خواهد بود. بنابراین شما باید پیدا کنید که با دعوت کردن چه کسانی از اعضای شرکت مجموع فاکتور شوخ طبعی حداکثر می شود.

خط اول شامل یک عدد صحیح n است که نشان دهنده ی تعداد افراد در شرکت است. خط بعدی شامل n شماره f_i فاکتور شوخ طبعی هر یک از افراد شرکت است. هر خط بعدی ساختار وابستگی را توصیف می کند. هر کس به جز مدیر عامل این شرکت دقیقاً یک رئیس مستقیم دارد. هیچ چرخه ای وجود ندارد: هیچ کس نمی تواند یک رئیس یک رئیس ... از رئیس خود باشد. هر یک از $n - 1$ خط دارای دو عدد صحیح u و v است و شما می دانید که u رئیس v است یا بالعکس (شما واقعا نمی خواهید بدانید که کدام یک رئیس است، اما شما می توانید تنها یکی از آن ها یا هیچکدام از آنها را دعوت کنید).
در خروجی یک عدد به عنوان حداکثر مجموع فاکتور شوخ طبعی برمی گردانید.

نمونه ۱

ورودی:

1
1000

خروجی:

1000

نمونه ۲

ورودی:

2
1 2
1 2

خروجی:

2

نمونه ۳

ورودی:

5
1 5 3 7 5
5 4
2 3
4 2
1 2

خروجی:

11

School Bus ۲

یک اتوبوس مدرسه باید از صبح زود از انبار شروع کند، همه دانش آموزان را از خانه هایشان با یک نظم خاص بردارد، همه آن ها را به مدرسه برساند و به انبار بازگردد. شما زمان لازم برای رفتن از انبار به هر خانه، از هر خانه به هر خانه دیگر، از هر خانه به مدرسه و از مدرسه به انبار را می دانید. شما باید ترتیبی را تعریف کنید که کمترین زمان ممکن برای طی کردن این مسیرها نیاز باشد.

این نمونه ای از یک مسئله کامل NP -complete کلاسیک به نام *Traveling Salesman Problem* است. در این صورت می توانید شما این مسئله را یک گراف در نظر بگیرید که رأس های آن خانه ها، مدرسه و انبار است و وزن یال ها زمان لازم برای رفتن از یک رأس به یک رأس دیگر است. بعضی از رأس ها ممکن است با هیچ رأسی در ارتباط نباشد. خط اول شامل دو عدد صحیح n و m است که به ترتیب نشان دهنده ی تعداد رأس ها و تعداد یال ها در گراف است. رأس ها از ۱ تا n شماره گذاری شده اند. هر یک از m خط بعدی حاوی سه عدد صحیح u ، v و t است که نشان دهنده ی این است که از رأس u به رأس v میتوان در زمان t رفت و بالعکس. از یک رأس به خودش نیز یالی وجود ندارد. اگر بتوان از یک رأس شروع کرد به طوری که از تمامی رأس های گراف عبور کنیم و هر رأس را تنها یک بار ملاقات کنیم، دو خط در خروجی باید برگردانید. خط اول کمترین زمان ممکن برای طی کردن چنین مسیری می باشد و خط دوم ترتیب ملاقات رأس ها در این مدت زمان است. اگر چنین مسیری در گراف وجود نداشته باشد، کفایت عدد ۱- را در خروجی برگردانید.

توجه داشته باشید که اگر یک گراف تنها شامل دو رأس باشد، می توان با یالی که بین این دو رأس وجود دارد از یک رأس به رأس دیگر رفت و دوباره از همان یال این مسیر را برگشت. پس در این حالت مسیر مورد نظر سوال در گراف وجود دارد.

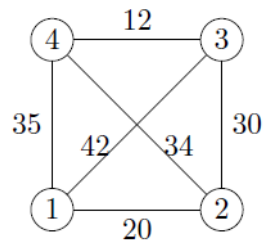
نمونه ۱

ورودی:

4 6
1 2 20
1 3 42
1 4 35
2 3 30
2 4 34
3 4 12

خروجی:

97
1 4 3 2



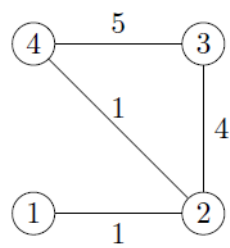
نمونه ۲

ورودی:

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

خروجی:

-1



Reschedule the Exams ۴

مسئول جدید آموزش دانشکده ی علوم کامپیوتر برنامه ای برای امتحان درس CS-۱۰۱ آماده کرده است. در واقع برای هر دانشجو یک تاریخ امتحان مشخص کرده است. با این حال، این برنامه ریزی دو مشکل اساسی دارد. مشکل اول این است که تاریخ امتحان برخی از دانشجویان که با هم دوست صمیمی هستند، در یک روز افتاده است و مشکل دوم این است که هیچ یک از دانشجویان در تاریخی که برای آن ها تعیین شده است، نمیتوانند در امتحان شرکت کنند. استادان سه تاریخ متفاوت برای این امتحان در نظر گرفته اند و این تاریخ ها قابل تغییر نیستند. تنها چیزی که می توان تغییر داد اختصاص تاریخ امتحان به هر دانشجو است.

شما مطمئناً می دانید که هر دانشجو نمی تواند در تاریخی که هم اکنون برای او برنامه ریزی شده است، شرکت کند ولی قطعاً در دو تاریخ دیگر می تواند در امتحان شرکت کند. همچنین، باید اطمینان حاصل کنید که هیچ دو دوست صمیمی تاریخ امتحان یکسانی نداشته باشند. شما باید تعیین کنید که آیا این امکان وجود دارد که برنامه ای برای امتحانات دانشجویان در نظر گرفت که این شرایط را داشته باشد یا خیر.

این مسئله را می توان به مسئله رنگ کردن گراف با سه رنگ کاهش داد. در این مسئله، یک گراف به شما داده می شود و هر رأس را می توان با یکی از ۳ رنگ ممکن، رنگ کرد. شما باید به هر رأس یک رنگ اختصاص دهید به طوری که متفاوت از رنگ راس های متصل به آن باشد. در اینجا رنگ ها متناظر با تاریخ امتحان هستند و راس ها متناظر با دانشجویان و یال ها متناظر با رابطه ی دوستی دانشجویان است.

خط اول شامل دو عدد صحیح n و m است که به ترتیب نشان دهنده ی تعداد رأس ها و تعداد لبه های گراف است. رأس ها از ۱ تا n شماره گذاری شده اند. خط بعدی شامل یک رشته به طول n است که فقط شامل حروف R ، G و B است که نماینده رنگ های فعلی هستند. برای هر موقعیت i در رشته، اگر R باشد یعنی رأس i قرمز رنگ است؛ اگر G باشد یعنی رأس i سبز رنگ است؛ اگر B باشد یعنی رأس i آبی رنگ است. این تخصیص رنگ فعلی است و هر یک از آنها باید تغییر کند. هر یک از m خط بعدی حاوی دو عدد صحیح u و v است که نشان دهنده ی آن است که رأس u و v توسط یک یال بهم متصل می شوند (ممکن است که $v = u$).

اگر این مسئله را نمی توان حل کرد کافی است در خروجی "Impossible" را برگردانید و اگر قابل حل است، رشته جدیدی که رنگ هر راس را نشان می دهد را در خروجی نمایش دهید.

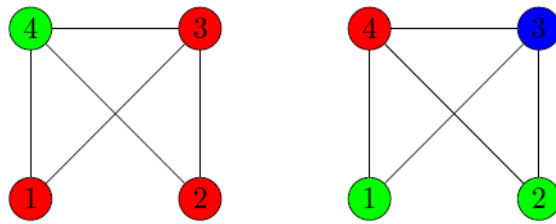
نمونه ۱

ورودی:

4 5
RRRG
1 3
1 4
3 4
2 4
2 3

خروجی:

GGBR



نمونه ۲

ورودی:

4 5

RGRR

1 3

1 4

3 4

2 4

2 3

خروجی:

Impossible

