به نام خدا

درس تحلیل و طراحی الگوریتم ها

تمرین چهارم

دانشکده مهندسی کامپیوتر

دانشگاه علم و صنعت ایران

استاد مرضیه ملکی مجد

نیم سال دوم ۱۴۰۲-۱۴۰۱

مهلت ارسال :

۲۷ / ۱ / ۱۴۰۲

مبحث :

گراف مقدماتی

مسئول تمارین :

آریا شهسوار

فهرست

آداب نامه تمرینات

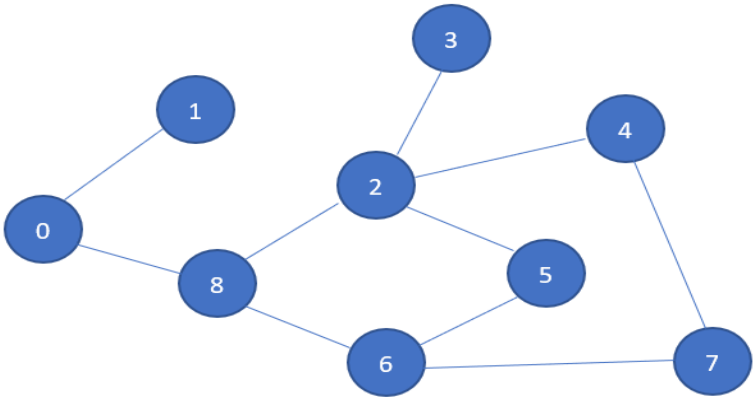
- پاسخ تمامی سوالات تنها به زبان های C# و C++ قابل قبول می باشد
- علیرغم اعتماد کامل تیم تی ای به شما دانشجویان عزیز، تمامی کد های شما با سایر دانشجویان بصورت خودکار و توسط برنامه مقایسه خواهند شد . همچنین در طول ترم ، از تمامی پاسخ های شما ارائه گرفته خواهد شد و نحوه کار تمامی بخش های هر سوال از شما پرسیده خواهد شد، لذا از کپی نمودن کد دوستانتان خودداری کنید و تمامی پاسخ ها، کد خودتان باشد . همچنین از آنجایی که مشورت و هم فکری با سایر دوستان بسیار کار پسندیده و مفیدی است – برخلاف کپی کردن کد (: – در صورت هم فکری با دانشجوی دانشجویان ، نام وی را بصورت کامنت شده در ابتدای کد خود بنویسید.
- برای ارسال تمارین در طول ترم، در مجموع ۱۵ روز می توانید تاخیر داشته باشید و در صورتی که جمع تاخیر دانشجویی بیشتر از ۱۵ روز شود، تمرین وی قابل قبول نخواهد بود لذا تلاش کنید تمرینات را در زمان مقرر در سامانه آپلود کنید.
- در تمامی تمرینات سعی شده است که سوالات ساده تر در ابتدا و سوالات دشوار تر در انتهای فایل قرار گیرند (از ساده به دشوار مرتب شده اند) .
- در صورت وجود هرگونه سوال در مورد تمرینات ، سعی کنید تا جایی که امکان دارد سوال خود را در گروه بپرسید چرا که شاید سوال شما، سوال دوستتان نیز باشد و دوستانتان نیز بتوانند از پاسخ سوال شما بهره ببرند.

نکات تمرین سری چهارم

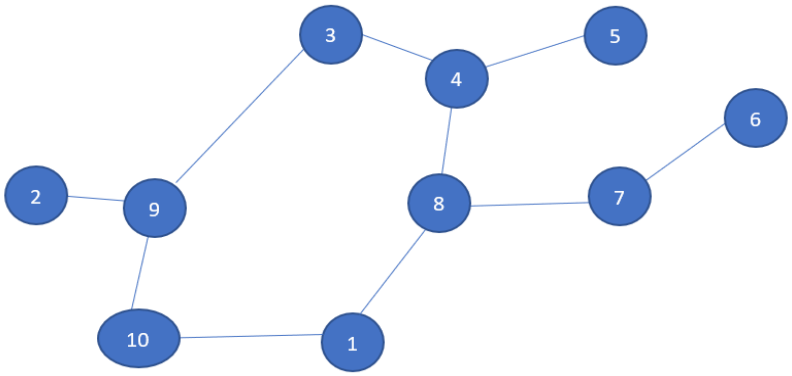
- سوالات را در سامانه کوئرا و در قسمت تمرین سری چهارم آپلود نمایید .
- سوالات دوم و سوم تمرین شامل دو بخش می باشند که برای هر بخش به طور مجزا در کوئرا محل مناسب برای آپلود پاسخ ایجاد شده است .
- از آنجایی که هر سوال توسط یک تی ای طرح شده است ، تنها تی ای طراح آن سوال می تواند شما را بصورت دقیق راهنمایی کند به همین منظور طراح هر سوال در زیر نوشته شده است تا در صورت ابهام و پرسش در مورد هر سوال، در صورتی که نیاز به پرسش سوال بصورت انفرادی در پیوی هست ، به تی ای مربوطه مراجعه بفرمایید
  - سوال ۱ . آقای شهرابی - آقای شهسوار
  - سوال ۲ . آقای عباسپور - آقای شهسوار
  - سوال ۳ . آقای عباسپور - آقای شهسوار
  - سوال ۴ . آقای شهرابی - آقای شهسوار
  - سوال ۵ . خانم بخشنده - آقای شهسوار
  - سوال ۶ . خانم بخشنده - آقای شهسوار
  - سوال ۷ . آقای احمدی - آقای شهسوار
  - سوال ۸ . آقای احمدی - آقای شهسوار

تمرین ۱ . سوال تئوری

الف) برای گراف زیر الگوریتم DFS را پیاده سازی کنید . دقت کنید که لازم است مرحله به مرحله الگوریتم را توضیح دهید و محتوای ساختمان داده ای که در حل سوال از آن استفاده می‌کنید را مرحله به مرحله مشخص کنید . عملیات را از راس شماره صفر آغاز کنید .



ب) این بار برای گراف زیر الگوریتم BFS را پیاده سازی کنید . تمامی نکاتی که در بخش قبل گفته شده بود ، برای این بخش نیز باید رعایت شوند . عملیات را از راس شماره ۱ آغاز کنید .



ج) پیچیدگی زمانی و پیچیدگی حافظه دو الگوریتم بالا را بیان کنید . همچنین تفاوت این الگوریتم‌ها از نظر ساختمان داده برای پیاده‌سازی و دلیل آن (برای مثال اگر از صف استفاده می‌شود ، چرا از پشته استفاده نمی‌شود و ...) بیان کنید . اگر بدانیم راس نهایی که دنبال آن می‌گردیم به راس شروع نزدیک‌تر است ، استفاده از کدام روش را پیشنهاد می‌کنید؟ در حالت برعکس چه طور (یعنی بدانیم راس نهایی در فاصله نسبتا زیادی نسبت به راس اولیه قرار دارد) ؟ دلیل خود را بیان کنید .

تمرین ۲ . پیاده سازی DFS

#DFS

الگوریتم Depth-First Search (DFS) یکی از پرکاربردترین الگوریتم های گراف می باشد و برای پیمایش (traverse) و یا جستجو (search) در ساختمان های داده گراف و یا درخت استفاده میشود .

بخش ۱ . Recursive DFS

در این بخش می خواهیم الگوریتم DFS را به صورت بازگشتی (recursive) پیاده سازی کنیم. به این صورت که به شما یک گراف با v راس (شماره راس ها از 1 تا v میباشد) و e یال داده میشود و شما باید با شروع از راس s تا جای ممکن گراف را پیمایش کنید و در خروجی راس های پیمایش شده را به ترتیبی که پیمایش کردید چاپ کنید. (دقت کنید که برای همسایه های هر راس، از چپ ترین شروع و به راست ترین بروید)

نکته ۱ : کد شما بررسی می شود و در صورتی که این الگوریتم را به صورت بازگشتی پیاده سازی نکرده باشید، نمره ای به شما برای این بخش تعلق نخواهد گرفت.

نکته ۲ : تضمین میشود که بین دو رأس حداکثر یک یال وجود دارد . (به بیان دیگر تضمین می شود که گراف ساده می باشد)

نکته ۳ : ترتیب خروجی پیمایش همسایه های یک راس بر اساس ترتیب ورودی یال های آن راس باشد . (برای فهم بهتر این موضوع به نمونه های ذکر شده توجه کنید)

ورودی :

در خط اول دو عدد v و e با فاصله از یکدیگر وارد میشوند که به ترتیب تعداد رئوس و تعداد یال های گراف می باشند .

در خط دوم عدد s می آید که بیانگر راس شروع DFS می باشد (به بیان دیگر الگوریتم DFS شما ابتدا بر روی این راس صدا زده میشود و root درخت DFS شما میباشد)

سپس در e خط بعدی در هر خط دو عدد v_i و v_j به شما داده میشود . ($1 \leq i, j \leq v$) در واقع هر خط بیانگر یک یال میان دو راس v_i و v_j می باشد .

خروجی :

در خروجی باید با توجه به ترتیب پیمایش راس ها شماره هر راس را در خط های جداگانه چاپ کنید .

Example 1 :

Input:

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

Output:

```
1
2
3
4
```

بخش ۲ . Iterative DFS

در گراف هایی که عمق درخت DFS بسیار زیاد میشود فراخوانی متعدد تابع DFS باعث پر شدن stack و ارور stack overflow میشود.

در این بخش می خواهیم این روش را بهبود ببخشیم و به جای استفاده از call stack با استفاده از ساختمان داده پشته - stack - این الگوریتم را پیاده سازی کنیم. مشابه قسمت قبل ، به شما یک گراف با v راس (شماره راس ها از ۱ تا v میباشد) و e یال داده میشود و شما باید با شروع از راس s تا جای ممکن گراف را پیمایش کنید و در خروجی راس های پیمایش شده را به ترتیبی که پیمایش کردید چاپ کنید.

نکته ۱ : کد شما بررسی می شود و در صورتی که این الگوریتم را با استفاده از ساختمان داده stack پیاده سازی نکرده باشید ، نمره ای به شما برای این بخش تعلق نخواهد گرفت.

نکته ۲ : تضمین می شود که گراف ساده می باشد .

نکته ۳ : ترتیب خروجی پیمایش همسایه های یک راس بر اساس ترتیب ورودی یال های آن راس باشد . (برای فهم بهتر این موضوع به نمونه های ذکر شده توجه کنید)

ورودی :

در خط اول دو عدد v و e با فاصله از یکدیگر وارد میشوند که به ترتیب تعداد رئوس و تعداد یال های گراف می باشند .

در خط دوم عدد s می آید که بیانگر راس شروع DFS می باشد (به بیان دیگر الگوریتم DFS شما ابتدا بر روی این راس صدا زده میشود و root درخت DFS شما میباشد)

سپس در e خط بعدی در هر خط دو عدد v_i و v_j به شما داده میشود . ($1 \leq i, j \leq v$) در واقع هر خط بیانگر یک یال میان دو راس v_i و v_j می باشد .

خروجی :

در خروجی باید با توجه به ترتیب پیمایش راس ها شماره هر راس را در خط های جداگانه چاپ کنید .

تمرین۳ . پیاده سازی BFS

#BFS

در سوال قبل با الگوریتم DFS که یکی از معروف ترین الگوریتم های پیمایش گراف است آشنا شدید . در این سوال با یکی دیگر از معروف ترین این الگوریتم ها به نام الگوریتم Breadth-First Search (BFS) آشنا خواهید شد .

بخش ۱ . Recursive BFS

در این بخش می خواهیم الگوریتم BFS را به صورت بازگشتی (recursive) پیاده سازی کنیم. به این صورت که به شما یک گراف با v راس (شماره راس ها از ۱ تا v میباشد) و e یال داده میشود و شما باید با شروع از راس s تا جای ممکن گراف را پیمایش کنید و در خروجی راس های پیمایش شده را به ترتیبی که پیمایش کردید چاپ کنید .

نکته ۱ : کد شما بررسی می شود و در صورتی که این الگوریتم را به صورت بازگشتی پیاده سازی نکرده باشید، نمره ای به شما برای این بخش تعلق نخواهد گرفت.

نکته ۲ : تضمین میشود که بین دو رأس حداکثر یک یال وجود دارد . (به بیان دیگر تضمین می شود که گراف ساده می باشد)

نکته ۳ : ترتیب خروجی پیمایش همسایه های یک راس بر اساس ترتیب ورودی یال های آن راس باشد . (برای فهم بهتر این موضوع به نمونه های ذکر شده توجه کنید)

ورودی :

در خط اول سه عدد v ، e و s با فاصله از یکدیگر وارد میشوند که به ترتیب تعداد رئوس ، تعداد یال های گراف و شماره راس شروع می باشند . (منظور از s این است که الگوریتم BFS شما ابتدا بر روی این راس صدا زده میشود و root درخت BFS شما میباشد)

سپس در e خط بعدی در هر خط دو عدد v_i و v_j به شما داده میشود . ($1 \leq i, j \leq v$) در واقع هر خط بیانگر یک یال میان دو راس v_i و v_j می باشد .

خروجی :

در خروجی باید با توجه به ترتیب پیمایش راس ها شماره هر راس را در خط های جداگانه چاپ کنید .

Example 1 :

Input:

```
10 10 1
1 8
1 10
8 4
8 7
10 9
4 3
4 5
7 6
9 2
9 3
```

Output:

```
1
8
10
4
7
9
3
5
6
2
```

بخش ۲ . iterative BFS

مشابه آنچه در الگوریتم DFS داشتیم فراخوانی متعدد تابع BFS نیز میتواند باعث پر شدن stack و ارور stack overflow میشود .

در این بخش می خواهیم این روش را بهبود ببخشیم و به جای استفاده از call stack با استفاده از ساختمان داده صف - queue - این الگوریتم را پیاده سازی کنیم. مشابه قسمت قبل، به شما یک گراف با v راس (شماره راس ها از ۱ تا v میباشد) و e یال داده میشود و شما باید با شروع از راس s تا جای ممکن گراف را پیمایش کنید و در خروجی راس های پیمایش شده را به ترتیبی که پیمایش کردید چاپ کنید .

نکته ۱ : کد شما بررسی می شود و در صورتی که این الگوریتم را با استفاده از ساختمان داده queue پیاده سازی نکرده باشید ، نمره ای به شما برای این بخش تعلق نخواهد گرفت.

نکته ۲ : تضمین میشود که بین دو رأس حداکثر یک یال وجود دارد . (به بیان دیگر تضمین می شود که گراف ساده می باشد)

نکته ۳ : ترتیب خروجی پیمایش همسایه های یک راس بر اساس ترتیب ورودی یال های آن راس باشد . (برای فهم بهتر این موضوع به نمونه های ذکر شده توجه کنید)

ورودی :

در خط اول دو عدد v و e با فاصله از یکدیگر وارد میشوند که به ترتیب تعداد رئوس و تعداد یال های گراف می باشند .

در خط دوم عدد s می آید که بیانگر راس شروع BFS می باشد (به بیان دیگر الگوریتم BFS شما ابتدا بر روی این راس صدا زده میشود و root درخت BFS شما میباشد)

سپس در e خط بعدی در هر خط دو عدد v_i و v_j به شما داده میشود . ($1 \leq i, j \leq v$) در واقع هر خط بیانگر یک یال میان دو راس v_i و v_j می باشد .

خروجی : در خروجی باید با توجه به ترتیب پیمایش راس ها شماره هر راس را در خط های جداگانه چاپ کنید .

تمرین۴ . عید دیدنی

#Reachability

در کشور زامپایا موعد سال نو فرا رسیده و همگی به دید و بازدید یکدیگر می‌روند. علی در شهر A و دوست او رضا در شهر B قرار دارد. علی میخواهد بداند آیا مسیری وجود دارد که او بتواند برای ملاقات دوستش که در شهر B قرار دارد برود یا نه.

این مسئله را میتوان به یکی از مسائل گراف تبدیل و مدل‌سازی کرد. به این صورت که کل کشور را یک گراف با v رأس و e یال در نظر میگیریم و هر شهر را یک رأس؛ همچنین هر مسیر بین دو شهر را میتوان یک یال در گراف در نظر گرفت. حال میتوان مسئله را به این صورت تبدیل کرد که آیا در گراف G حداقل یک مسیر بین دو رأس A و B وجود دارد یا نه؟

ورودی :

در خط اول دو عدد v و e با فاصله از هم آمده اند که به ترتیب بیانگر تعداد شهر ها - تعداد رئوس - و تعداد مسیر (جاده) های موجود در کشور - تعداد یال ها - در کشور زامپایا می باشند .

در خط دوم راس A و B با فاصله هم از هم آمده اند که به ترتیب بیانگر شهر علی و شهر رضا می باشند .

سپس در v خط بعدی در هر خط همسایه های هر رأس آمده است (که معادل همسایه های یک شهر می باشند).

به این صورت که در خط i، عدد اول، k، مشخص کننده تعداد همسایه های شهر (رأس) i می‌باشد و سپس k عدد بعدی همسایه های شهر (رأس) i می‌باشند.

خروجی :

در صورتی که حداقل یک مسیر بین دو شهر (رأس) A و B وجود داشته باشد، در خروجی کلمه YES و در غیر این صورت کلمه NO را چاپ کنید.

Example 1 :

Input:

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

Output:

```
YES
```

تمرین۵ . تعیین برنامه درسی

#Topological_sort

فرض کنید که برنامه درسی رشته مهندسی کامپیوتر را به شما داده اند که لیستی از دروس این رشته است و پیش نیاز هر درس را نیز مشخص میکند .
وظیفه شما این است که ترتیبی را برای اخذ درس ها ، با رعایت پیش نیاز ها پیشنهاد دهید که تمامی درس ها در این ترتیب وجود داشته باشند .
بدین منظور ، یک گراف جهت دار را در نظر بگیرید که راس های این گراف معادل درس های رشته و یال ها وابستگی درس ها به یکدیگر را نشان میدهد به این معنا که اگر درس u پیش نیاز درس v باشد ، از راس u به راس v یک یال جهت دار در گراف متناظر وجود دارد . در نهایت برای یافتن ترتیبی برای اخذ درس ها ، با رعایت پیش نیاز ها ، برای این گراف توپولوژی آن را پیدا کنید

ورودی :

در خط اول ورودی ، تعداد یال های گراف e و در خط دوم تعداد راس (گره) های گراف v بیان میشوند . سپس در e خط بعدی ، در هر خط دو عدد با فاصله از یکدیگر می آیند که نشان‌دهنده ی وجود یک یال جهت دار از راس با شماره عدد اول به راس با شماره عدد دوم می باشد .
نکته ۱ : تضمین میشود که گراف ورودی یک گراف جهت دار بدون دور می باشد (DAG)

خروجی :

در خروجی ، یکی از توپولوژی های مرتب گراف داده شده را چاپ کنید . (یک ترتیب اخذ درسها را چاپ کنید)

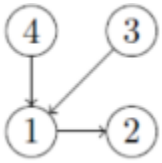
Example 1 :

Input:

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

Output:

```
4 3 1 2
```



تمرین۶ . تولد سایین

#Connected_component

سایین و چند نفر از دوستان سایین به تولد سایین دعوت شده اند . از آنجایی که برخی دوستان سرکار سایین هستند و برخی از دوستان دانشگاه سایین و.... بعضی از دوستان سایین همدیگر رو نمیشناسند و برخی دیگران را میشناسند .

نکته : دقت شود که این رابطه دو طرفه در نظر گرفته میشود به این معنا که اگر فرد A ، فرد B را بشناسد ، فرد B نیز او را می شناسد .

از آنجایی که سایین عاشق مسابقه بین دوستانش هست تصمیم گرفت که دوستانش به گروه هایی تقسیم شوند و مسابقه دهند . شرط تیم شدن افراد در این مسابقه این است که به ازای هر فرد در گروه بتوان یک مسیر آشنایی به تمام اعضای گروه پیدا کرد)

از آنجایی که مسابقه خرج دارد و سایین باید به ازای هر تیم هزینه کند از آریا خواست که حداقل تعداد تیم ها برای این که هر تیم شرایطش را داشته باشد حساب کند .

این مسئله را میتوان به یکی از مسائل گراف تبدیل و مدل‌سازی کرد به این صورت که هر کدام از دوستان سایین را یک node در گراف در نظر گرفته و هر رابطه آشنایی بین دو فرد را یک یال میان آن دو نفر در نظر بگیرید . حال کمترین تعداد گروه هایی که میتوان با شرط مذکور ایجاد کرد معادل تعداد اجزای متصل در گراف می باشد .

ورودی :

در خط اول ورودی ، تعداد یال های گراف e و در خط دوم تعداد دوستان سایین v بیان میشوند . سپس در e خط بعدی ، در هر خط دو عدد با فاصله از یکدیگر می آیند که نشان‌دهنده ی وجود یک یال جهت دار از راس با شماره عدد اول به راس با شماره عدد دوم می باشد .

خروجی :

در خروجی ، تعداد اجزای متصل در گراف را چاپ کنید .

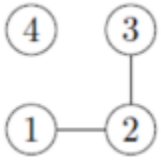
Example 1 :

Input:

```
2

4

1 2

3 2
```

Output:

```
2
```



تمرین۷ . وسواس فکری

#Singly_connected_graph

تولد ناجی است و از آنجا که ناجی علاقه زیادی به ریاضیات دارد کیا یک گراف جهت دار به این مناسبت به او هدیه داده است . از آن جایی که ناجی مبتلا به اختلال OCD (وسواس فکری) است ، تنها در صورتی که بین هر دو راس گراف حداکثر یک مسیر ساده وجود داشته باشد ، آن را به عنوان هدیه می پذیرد . به همین دلیل کیا از شما میخواهد تا به او کمک کنید که آیا گرافی که در نظر دارد این ویژگی را دارد یا خیر .

ورودی :

در خط اول ورودی دو عدد v و e با فاصله از یکدیگر داده میشوند که به ترتیب بیانگر تعداد راس ها و یال های گراف می باشند . سپس در e خط بعدی ، در هر خط دو عدد با فاصله از یکدیگر می آیند که نشان‌دهنده ی وجود یک یال جهت دار از راس با شماره عدد اول به راس با شماره عدد دوم می باشد .

خروجی :

در تنها خط خروجی در صورتی شرط مورد نظر در گراف داده شده صادق باشد عبارت YES و در غیر این صورت عبارت NO چاپ شود .

Example 1 :

Input:

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

Output:

```
YES
```

Example 2 :

Input:

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

Output:

```
YES
```

Example 3 :

Input:

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

Output:

```
NO
```

تمرین۸ . اکیپ های دانشجویی

#Strongly_connected_component

علی بسیار به جمع آوری شایعات و اطلاعات از دانشجویان علاقه مند است . حال او میخواهد تعداد اکیپ های ورودی ۱۴۰۱ را بدست آورد . برای اینکار او یک گراف طراحی کرده است که راس های آن دانشجویان هستند و اگر دانشجوی x اقدام به دست دادن با دانشجوی y کند یک یال جهت دار از راس متناظر x به راس متناظر y میکشد. علی بر این باور است که مولفه های قویا همبند این گراف تشکیل اکیپ های دانشجویی را میدهند . مولفه قویا همبند در گراف به زیر مجموعه ای از راس های گراف میگویند که از هر راس آن به بقیه راس های آن حداقل یک مسیر وجود داشته باشد .

ورودی :

در خط اول ورودی دو عدد v و e با فاصله از یکدیگر داده میشوند که به ترتیب بیانگر تعداد راس ها و یال های گراف می باشند . سپس در e خط بعدی ، در هر خط دو عدد با فاصله از یکدیگر می آیند که نشان‌دهنده ی وجود یک یال جهت دار از راس با شماره عدد اول به راس با شماره عدد دوم می باشد .

خروجی :

در تنها خط خروجی یک عدد که بیانگر تعداد اکیپ های دانشجویی (تعداد مولفه های قویا همبند گراف متناظر) است را چاپ کنید .

Example 1 :

Input:

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

Output:

```
2
```

Example 2 :

Input:

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

Output:

```
4
```

Example 3 :

Input:

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

Output:

```
3
```