**پروژه‌ی پایانی – درس طراحی الگوریتم**
**دکتر باقری**
**ترم زمستان ۱۳۹۷–۱۳۹۶**
**دانشکده‌ی مهندسی کامپیوتر و فناوری اطلاعات – دانشگاه صنعتی امیرکبیر**
**زمان تحویل: ۲۳ خرداد ۱۳۹۷**

دانشگاه صنعتی امیرکبیر
( پلی تکنیک تهران )

پروژه‌ی پایانی به دو دسته تقسیم می‌شود:

۱. الگوریتم‌های تشخیص اجتماعات (که مناسب گروه‌های دو نفره می‌باشد)
   - این الگوریتم‌ها با شماره‌های ۱ تا ۷ مشخص شده‌اند.
۲. الگوریتم‌های رنگ‌آمیزی گراف (که مناسب گروه‌های تک نفره می‌باشد)
   - این الگوریتم‌ها با شماره‌های ۸ تا ۳۱ مشخص شده‌اند.

با توجه به اسامی ایمیل شده به تدریس‌یار درس (آقای نادری) این پروژه‌ها برای افراد داوطلب انجام پروژه تعریف شده است.

نکات قابل توجه:

۱) تنها زبانِ قابل قبول برای انجام پروژه‌ها، زبان **جاوا** می‌باشد.
۲) تحویل پروژه‌ها به صورت حضوری می‌باشد و زمان آن متعاقباً اعلام خواهد شد.
۳) فایل پروژه ها به همراه گزارش کامل انجام پروژه، باید تا ساعت ۲۳:۵۹ روز پنجشنبه ۲۳ خرداد در مودل بارگذاری شود.
   - فایل بارگذاری شده باید یک فایل زیپ با عنوان شماره‌ی دانشجویی اعضا باشد.
   - اسم (اسامی) افراد گروه را در یک فایل با نام Info.txt در داخل فایل زیپ نهایی قرار دهید. (اسم فایل نهایی شماره دانشجویی‌تان باشد).
   - از قید توضیحات طولانی درباره‌ی کدها در گزارش کار خودداری کنید.
   - روند انجام کار و الگوریتم استفاده شده را به اختصار در گزارش توضیح دهید.
   - نتایج خروجی الگوریتم مورد نظر خود را در گزارش قید کنید.
   - کدها را کامنت گذاری کنید.
۴) تحویل پروژه‌ها صرفاً به صورت ("درست و کامل" و یا "نادرست") می‌باشد. در واقع پروژه‌هایی که عملکرد صحیحی داشته باشند و از لحاظ زمان اجرا نیز منطقی باشند، نمره‌ی کامل دریافت خواهند نمود. همچنین کدهایی که ناقص هستند، به درستی اجرا نمی‌شوند، جواب صحیح تولید نمی‌کنند و یا سرعت اجرای آن‌ها قابل قبول نمی‌باشند؛ متأسفانه نمره‌ای دریافت نخواهند نمود.
۵) برای پیاده‌سازی پروژه‌ها می‌توانید از تمام امکانات، کتابخانه‌ها و ساختمان داده‌های زبان جاوا استفاده کنید.
۶) اسامی افراد و پروژه‌هایی که باید انجام دهند در ادامه آمده است. توضیح اینکه به دلیل اینکه تعداد افراد متقاضیِ انجام پروژه‌ی دو نفره زیاد بود، مجبور به جدا کردن اعضای بعضی از گروه‌ها و تبدیل آن‌ها به گروه‌های تک نفره شدیم (که این کار توسط یک تابع رندوم انجام شد). طبعاً گروه‌هایی که اعضای آن‌ها از هم جدا شده‌اند، موظف به انجام پروژه‌های تک نفره و به صورت جدا جدا می‌باشند.
۷) توضیحاتِ مربوط به پروژه‌ها به صورت مجزا و به اختصار در ادامه آمده است. ضمناً در صورت وجود هرگونه سؤال و ابهام از طریق ایمیل با تدریس‌یاران درس در ارتباط باشید:
   - Sabergholami72@gmail.com پروژه‌های رنگ‌آمیزی گراف (پروژه‌های شماره‌ی ۸ تا ۳۱)
   - Hamid.sh.j@gmail.com پروژه‌های تشخیص اجتماعات (پروژه‌های شماره‌ی ۱ تا ۷)

| اسامی گروه‌های دو نفره | شماره‌ی پروژه |
|---|---|
| سروش برمکی-سارا اصغری | ۱ |
| علی رضا موذنی - علی ارجمند بیدگلی | ۲ |
| احسان سوری-حسین نظری ابر | ۳ |
| محمدمهدی عبدالله پور-آریا وارسته نژاد | ۴ |
| محمد مهدی قنبری-سید امیر محمد جلیلی | ۵ |
| محمد رحمدل-محمد مهدی حیدری | ۶ |
| زهرا ناصری-مریم سادات معصومی | ۷ |
| پارسا فرین نیا-سروش اسماعیلیان | ۱ |
| فاطمه کشوری-زهرامتشکرآرانی | ۲ |
| سجاد پیشواییان-محمدمهدی نفر | ۳ |
| محمد سامی-محمدرضا صمدی | ۴ |
| محمدحسین خجسته-مهران تقیان | ۵ |
| مسعود غیاثی-سینا ترکاشوند | ۶ |
| زینب خالوندی-نجمه محمدباقری | ۷ |
| روزبه قاسمی-سپهر عسگریان | ۱ |
| اشکان میرزاحسینی-عیسی کرامتی | ۲ |
| علی رضا بختیاری-بهنام امین آزاد | ۳ |
| امیرحسین ژاله محرابی-مهشید شیری | ۴ |
| کوثر بهنیا-سیده زهرا رشته احمدی | ۵ |
| زهرا گنجی-فاطمه شیرازی | ۶ |
| سید سعید صفایی-پارسا کاوه زاده | ۷ |

| نام | پروژه | نام | پروژه |
|---|---|---|---|
| نوید شهسواری | ۸ | امیرحسین قندهاری | ۸ |
| امیرمحمد نظری | ۹ | مسعود قاسمی | ۹ |
| ارمغان سرور | ۱۰ | نازنین تقوی | ۱۰ |
| محمدمحسن محمدی | ۱۱ | الهه رنجبری | ۱۱ |
| امید حاجی عبدلپور | ۱۲ | پرند ضیایی فر | ۱۲ |
| نیما داوری | ۱۳ | شهرزاد حاجی امین شیرازی | ۱۳ |
| محسن متقیان | ۱۴ | مهدی قنبری | ۱۴ |
| امیر آخوند مهدی | ۱۵ | آراد اشرفی | ۱۵ |
| نازنین اختریان | ۱۶ | آذین اله خانی | ۱۶ |
| مهسان اکبری امین | ۱۷ | مهسا بیکران | ۱۷ |
| مریم محمدی اردهالی | ۱۸ | عرفان علیزاده نوحی | ۱۸ |
| پگاه زاهدی | ۱۹ | مصطفی معصومی | ۱۹ |
| علی شفیعی | ۲۰ | سوده نیلفروشان | ۲۰ |
| فاطمه غلام زاده | ۲۱ | سید امیرعلی سجادی | ۲۱ |
| رعنا شمیم نسب | ۲۲ | رادین احمدی راد | ۲۲ |

| | | | |
|---|---|---|---|
| سحر ابراهیمی | ۲۳ | امیرحسین کامرانی | ۲۳ |
| دلارا فرقانی | ۲۴ | احسان فتوحی | ۲۴ |
| حافظ بهرآسمانی | ۲۵ | علی رضا کاووسی | ۲۵ |
| محمد خلجی | ۲۶ | محمد معین حاج ابراهیمی | ۲۶ |
| کیمیا جوانشیر | ۲۷ | پرهام رحیمی | ۲۷ |
| علی یزدانی | ۲۸ | مهرداد شیخ جابری | ۲۸ |
| مهدیس صفری | ۲۹ | بیتا رحیم‌خانی | ۲۹ |
| پارسا اسکندرنژاد | ۳۰ | مینافریدی | ۳۰ |
| مارال رسولی جابری | ۳۱ | امیرمحمد پیرحسینلو | ۳۱ |
| ابولفضل طاهری | ۸ | حیدر سودانی | ۸ |
| محمد حسن مجاب | ۹ | | |

با توجه به شماره‌ی پروژه‌ی خود، می‌توانید توضیحات کلی درباره‌ی آن را در ادامه مشاهده کنید. (اگر شماره‌ی پروژه‌ی شما بین ۱ تا ۷ می‌باشد توضیحات مربوط به «یافتن اجتماعات» را مطالعه کنید و اگر بین ۸ تا ۳۱ است توضیحات مربوط به «رنگ‌آمیزی گراف» را مشاهده کنید).

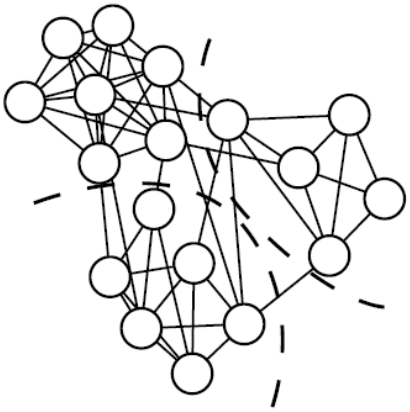
# توضیحات مربوط به پروژه‌های یافتن اجتماعات

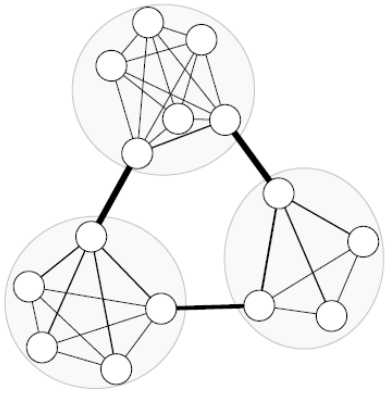
### تعریف صورت پروژه:

در این پروژه قصد داریم تا الگوریتم‌های مختلف تشخیص اجتماعات را در زبان Java پیاده‌سازی نماییم. تمام این الگوریتم‌های بر اساس الگوریتم LPA می‌باشند و سعی نموده‌اند که با بهبود این الگوریتم، کارایی را افزایش دهند. در ادامه توضیح مختصری از تشخیص اجتماعات و الگوریتم LPA ارایه می‌گردد.

### تشخیص اجتماعات:

در اجتماعات، تعامل میان گره‌های داخل یک اجتماع با یکدیگر بیش‌تر از گره‌های خارج از اجتماع است. بصورت انتزاعی، یک تعریف خاص از اجتماعات در شبکه‌های اجتماعی می‌توان داشت که، اجتماع متشکل از افرادی است که شباهت‌هایی را به اشتراک می‌گذارند و یا دارای علاقه‌مندی‌ها یا منافع مشترکی هستند و یا به‌واسطه یک رابط خاص در دنیای واقعی به هم مرتبط شده‌اند. در سطح شبکه ممکن است راس‌ها به چندین دسته مختلف تقسیم شده، در حالی که هر دسته یک اجتماع تشکیل می‌دهد. تعداد حالات مختلف برای دسته‌بندی راس‌ها از درجه‌ی نمایی می‌باشد و طبق تحقیقات انجام شده پیچیدگی یافتن بهترین دسته‌بندی به نحوه‌ی که اجتماعات تابع هدف، بیشترین تراکم یال در داخل اجتماعات و خلوترین تراکم یال بین اجتماعات، را بیشینه نمایند از دسته مسایل ان‌پی سخت محسوب می‌شود.





### تعاریف مشابه با اجتماعات

در نهایت می‌توان هشت تعریف زیر که مشابه با مفهوم اجتماعات هستند را در نظر گرفت:

- زیرگراف کامل: زیرگروه‌هایی که تمامی اعضای آن به هم متصل هستند.

- زیرگراف کامل از درجه n: بزرگترین زیرگرافی که فاصله هر زوج مرتب از یال‌های آن، از n بیش‌تر نیست.
- کاپلکس: بزرگترین زیرگرافی که هر گره از آن، با تمام گره‌های دیگر در زیرگراف همجوار است، به جز حداکثر k گره از آن.
- گردایه ال اس: زیرگرافی که درجه داخلی رئوس آن از درجه خارجی بیش‌تر است.
- گردایه لاندا: زیرگرافی که هر دو گره از آن اتصال قوی‌تری نسبت به اتصال یک گره از درون زیرگراف و یک گره خارج از زیرگراف دارند.
- انجمن‌هایی بر اساس میزان سازگاری یا میزان کیفیت
- انجمن‌هایی که بر اساس الگوریتم‌های ماژولاریتی مشخص می‌شوند.

خوشه‌ها: انجمن‌هایی که بر اساس روش‌های خوشه‌بندی مشخص می‌شوند

## الگوریتم تکثیر برچسپ

در سال ۲۰۰۷ رقوان[۱] الگوریتمی برپایه‌ی تصادف ارایه داد که با سرعت بسیار زیاد می‌تواند اجتماعات را در شبکه‌های پیچیده تشخیص دهد. در این روش در ابتدا تمام راس‌ها یک برچسپ مجزا دارند و در هر بار اجرا هر راس برچسپی که بیشترین فراوانی را در همسایه‌هایش دارد، انتخاب نموده و به عنوان برچسپ جدید خود جایگزین می‌نماید. این رویه تا زمانی که تمام راس‌های شبکه برچسپشان با پرتکرارترین برچسپ همسایه‌اش یکی باشد ادامه می‌دهد. در نهایت برچسپ‌های یکسان نشان دهنده‌ی یک اجتماع می‌باشند. قدم‌های زیر رویه‌ی این الگوریتم را با جزییات بیان می‌نماید:

۱. هر راس یک برچسپ مجزا می‌گیرد.
۲. راس‌ها بصورت تصادفی در لیست X قرار می‌گیرند
۳. بترتیب، راس‌های لیست X برچسپی که بیشترین تکرار را در همسایه‌اش دارد بعنوان برچسپ جدید خود انتخاب می‌کند.
۴. اگر تمام راس‌ها برچسپی که بیشترین تکرار را در همسایه‌اش دارد را نداشته باشند برو به قدم ۲
۵. راس‌های که برچسپ مشابه دارند یک اجتماع در نظر گرفته می‌شوند.

پیچیدگی قدم‌های اول، دوم و پنجم از مرتبه‌ی O(n) است ولی پیچیدگی قدم‌های دوم و سوم از مرتبه‌ی O(m) است زیرا، هر راس باید برچسپ همسایه‌های خود را بررسی نماید از سوی متوسط درجه برابر است با m/n پس حاصل ضرب تعداد راس‌ها در متوسط درجه‌ی ریوس پیچیدگی برابر با O(m) دارد. چون پیچیدگی مرتبه‌ای O(n) از O(m) کمتر است پس پیچیدگی کل الگوریتم برابر با O(m) است. از طرفی تعداد دور حلقه موجود در قدم چهارم از مرتبه‌ی ثابت می‌باشد و طبق ادعای نویسنده در ۹۵٪ مواقع در کمتر از ۵ دور جواب مناسب را پیدا می‌نماید. در گراف‌های خلوت متوسط درجه عددی ثابت است و معمولا گراف‌های اجتماعپذیر خلوت هستند پس، می‌توان این نتیجه را گرفت که پیچیدگی این الگوریتم در گراف‌های خلوت از مرتیه‌ی O(n) است که نشان دهنده‌ی خطی بودن و سریع بودن این الگوریتم است.

پیچیدگی فضایی این الگوریتم نیز خطی است زیرا، یک ارایه به اندازه O(n) جهت ذخیره نمودن لیست تصادفی راس‌ها در قدم دوم نیاز است. از سوی، شبکه را می‌توان بصورت لیست مجاورتی ذخیره نمود که نیاز به فضایی به اندازه‌ی O(m) دارد. در نهایت کل فضایی مورد نیاز از مرتبه‌ی O(m) است که در ماتریس‌های خلوت به مرتبه‌ی O(n) می‌رسد پس مرتبه‌ی فضایی این الگوریتم نیز نزدیک به خطی است.

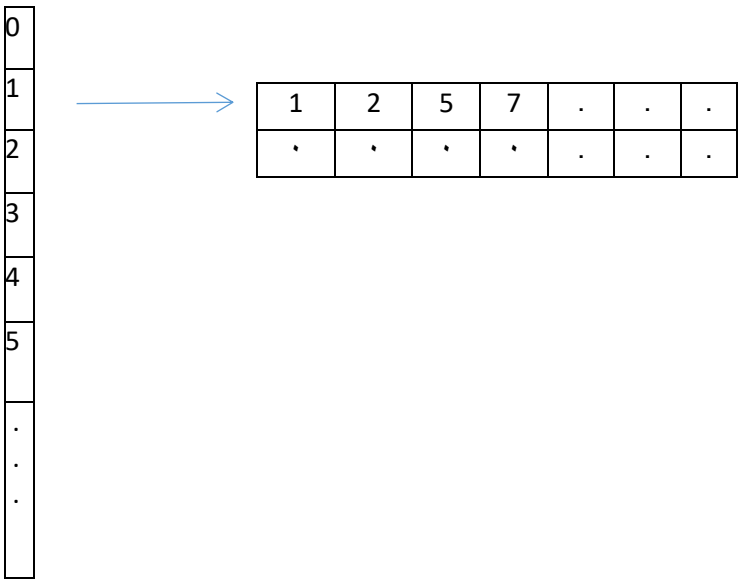
## فرمت ورودی

[۱] Raghvan

فایلی که از دیسک خوانده می‌شود به فرمت CSV است و هر راس با یک عدد طبیعی نشان داده می‌شود؛ توجه شود که شروع اندیس ها از عدد یک می‌باشد. تمام فایل باید یکجا خوانده شده و در رم بارگذاری شود. جهت ذخیره‌ی گراف در حافظه، از لیست مجاورت باید استفاده شود. کد زیر در زبان java می‌باشد که یک فایل متنی به فرمت CSV را خوانده و در ساختمان داده‌ی مدنظر ما ذخیره می‌نماید.

```
public static Vector graph = new Vector<Vector>();
int n= nodes' number
for (int i = 0; i < n; i++)
    graph.add(new Vector<int[]>());
BufferedReader br = new BufferedReader(new FileReader("network.txt"));
String line = br.readLine();
while (line != null) {
    String[] parts = line.split(" ");
    int source = Integer.parseInt(parts[0]);
    int destination = Integer.parseInt(parts[1]);
    ((Vector) (graph.get(source))).add(new int[]{destination,0});
    ((Vector) (graph.get(destination))).add(new int[]{source,0});
    line = br.readLine();
}
br.close();
```

توجه شود که ما فرض می‌کنیم که تعداد راس‌ها را می‌دانیم و ساختمان داده‌ی ما یک vectorی است که هر عضو آن به یک vectorی اشاره می‌نماید که حاوی راس‌های همسایه می‌باشند. شما قادر هستید که در صورت نیاز، اگر الگوریتم شما نیاز داشت تا اطلاعاتی اضافه‌تر برای همسایه‌ها (به عبارتی یال‌ها) نگه دارد، هر عضو vector افقی را یک آرایه تعریف کنید بطول k که شامل راس همسایه و سایر امتیازات اضافی باشند.  بعنوان مثال کد بالا یک عدد نیز برای هر همسایه در نظر گرفته است که صفر است.




برای ذخیره سازی اجتماعات هر راس نیز می‌تواند یک آرایه داشته باشید که هر عنصر آن بیانگر اجتماع آن راس می‌باشد.

### نحوی پیاده‌سازی:

برای پیاده‌سازی الگوریتم‌های باید مقاله‌ی مربوط به الگوریتم را مطالعه نمایید؛ البته، تنها بخش‌های مربوط به تعریف الگوریتم باید مطالعه شود و سایر بخش‌ها مانند مقدمه، کارهای مرتبط و آزمایشات در طول پیاده‌سازی استفاده نخواهند شد. برای پیاده‌سازی آزاد هستید که از تمام امکانات زبان Java استفاده نمایید و از کتابخانه‌ها و ساختمان داده‌های موجود استفاده نمایید. تاکید می‌شود که ساختمان داده‌ی استفاده شده حتما با ساختمان داده‌ی اشاره شده در بالا یکسان باشد. کد اجرایی شما باید به بهترین نحو ممکن پیاده‌سازی شود و زمان اجرای آن با زمان ارایه شده در مقاله یکسان باشد و پیاده‌سازی شما کاستی نداشته باشد.

نمره‌دهی بصورت صفر/یک است؛ یعنی اگر کد شما بدرستی اجرا شود و زمان اجرای آن نیز معقول باشد، آنگاه نمره‌ی کامل را خواهد گرفت. کدهای که ناقص هستند یا بدرستی اجرا نمی‌شوند یا سرعت اجرای آنها قابل قبول نباشند، هیچ نمره‌ای کسب نخواهند نمود.

در صورت وجود هر گونه ابهام در مقالات، مفاهیم، فرمول‌ها و تعاریف می‌توانید سوال خود را به میل زیر ارسال نمایید.

حمید شهریوری جوقان - hamid.sh.j@gmail.com

| No. | Tittle |
|---|---|
| **1** | GA-LP: A genetic algorithm based on label propagation to detect communities in directed networks. |
| **2** | Detecting communities in social networks using label propagation with information entropy. |
| **3** | LabelRank: A Stabilized Label Propagation Algorithm for Community Detection in Networks. |
| **4** | Detecting community structure using label propagation with weighted coherent neighborhood propinquity. |
| **5** | Community Detection Using A Neighborhood Strength Driven Label Propagation Algorithm. |
| **6** | Layered Label Propagation: A MultiResolution Coordinate-Free Ordering for Compressing Social Networks Categories and Subject Descriptors. |
| **7** | Advanced modularity-specialized label propagation algorithm for detecting communities in networks. |

# توضیحات مربوط به پروژه‌های رنگ‌آمیزی گراف

**تعریف صورت پروژه:**

در این پروژه قصد داریم تا الگوریتم‌های مختلف رنگ‌آمیزی گراف را با زبان Java پیاده‌سازی کنیم. در واقع این مسأله به این صورت تعریف می‌شود:

**ورودی) یک گراف G(V,E) که به صورت CSV است.**

**خروجی) برای هر رأس v که متعلق به V است؛ رنگ رأس (به صورت یک عدد بین 1 تا C)**

این الگوریتم‌ها در واقع یک زیر مجموعه‌ای از مسأله‌ی معروف "رنگ‌آمیزی گراف" می‌باشند. در ادامه توضیحات بیشتری درباره‌ی این موضوع داده می‌شود.

**رنگ‌آمیزی گراف:**

رنگ آمیزی گراف[۲] یکی از مسائل معروف دنیای گراف‌ها می‌باشد. این مسأله یک زیرمجموعه از مسأله‌ای بزرگتر به نام Graph Labeling است. رنگ آمیزی گراف به ۳ صورت قابل بیان است:

۱. رنگ آمیزی رئوس
۲. رنگ آمیزی یال‌ها
۳. رنگ آمیزی رئوس و یال‌ها

**رنگ‌آمیزی رأس:**

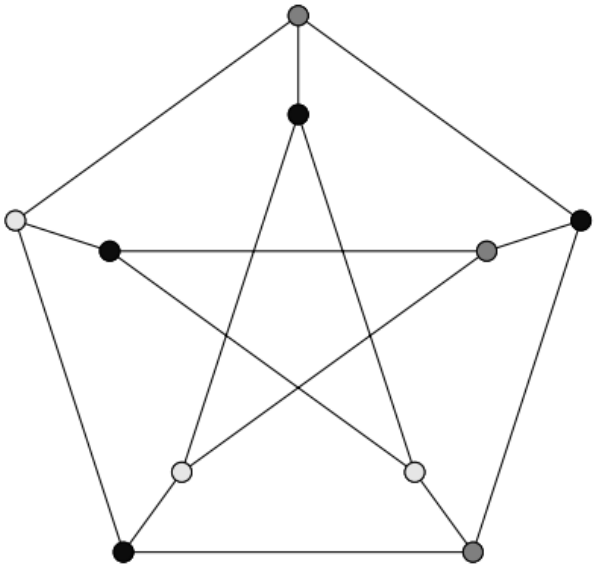
یک رنگ‌آمیزی صحیح رأس برای یک گراف G=(V,E) به این صورت است که به تمام رئوس یک رنگ نسبت داده شود به صورتی که هیچ دو رأس همسایه‌ای رنگ یکسانی نداشته باشند (به دو رأس در صورتی همسایه گفته می‌شود که بین آن دو رأس یک یال باشد).

گرافِ $G = (V, E)$ را به عنوانِ یک گرافِ بدونِ جهت در نظر بگیرید که نمایانگرِ یک شبکه است. به صورتی که $V$ مجموعه‌ی رئوس و $E$ مجموعه‌ی یال‌هاست. برای مشخص کردن رنگ‌های مجموعه‌ی رئوس، از یک مجموعه‌ی رنگ به نام $C$ استفاده می‌کنیم. همچنین برای تقسیم رئوس به دسته رنگ‌های مختلف، از تابع $\varphi$ استفاده می‌شود: $\varphi: V(G) \to C$ به صورتی‌که $\varphi(x) \neq \varphi(y)$ برای تمام $xy$ های متعلق به مجموعه‌ی یال‌های $E$ (در این مسأله فرض می‌شود که گراف بدونِ طوقه می‌باشد). اگر اندازه‌ی $C$ برابر $k$ باشد، $\varphi$ یک رنگ‌آمیزیِ k ام[۳] می‌باشد. ضمناً اگر k متناهی باشد، مجموعه‌ی C به صورت روبرو در نظر گرفته می‌شود: $C = \{1, 2, 3, \dots, k\}$. برای تمام $i \in C$، مجموعه‌ی $\varphi^{-1}(i)$ کلاس رنگیِ $i$ ام است. در واقع تمامِ رئوسی که رنگ $i$ ام به آن‌ها نسبت داده شده است، در مجموعه‌ی رنگیِ $i$ ام قرار می‌گیرند. بنابراین هر کلاس رنگی، یک مجموعه‌ی مستقل از رئوس را تشکیل می‌دهد (اشتراکی بین آن‌ها نیست). به عبارتی دیگر بین هیچ جفت رأسی که در یک مجموعه نباشند، یالی وجود ندارد. ضمناً کمترین مقدار k برای یک گراف، "عدد کروماتیک رنگی" نامیده می‌شود.

در شکل زیر یک رنگ‌آمیزیِ صحیح رأس مشاهده می‌شود:

---

[2] Graph Coloring
[3] K-coloring



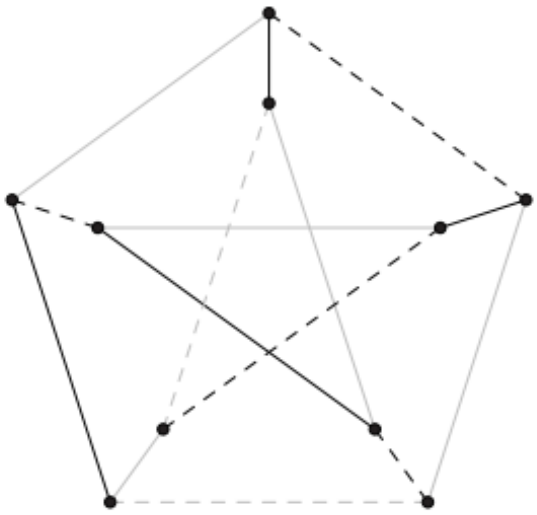
همان‌طور که مشاهده می‌شود، هیچ دو رأس مجاوری در گراف بالا؛ رنگ یکسانی ندارند. بنابراین این رنگ‌آمیزی یک رنگ‌آمیزی صحیح است.

## رنگ آمیزی یال:

نوع دیگر و محبوب رنگ‌آمیزی گراف، رنگ‌آمیزی یال می‌باشد. مانندِ رنگ‌آمیزی رأس، رنگ‌آمیزی یال برای گراف G=(V,E) به معنی ارائه‌ی یک تبدیل است که به هر یال گراف یک رنگ نسبت می‌دهد با این شرط که یال‌هایی که در یک رأس مشترکند، رنگ یکسانی نخواهند داشت. به صورت ریاضی داریم:

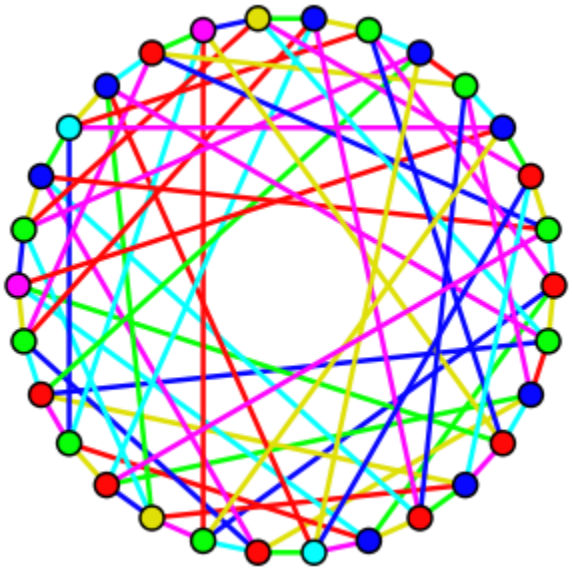
$$f: E(G) \to N \text{ such that } \forall \left(e_i, e_j \in E(G)\right), i \neq j \;\; e_i, e_j \text{ are adjacent } \Rightarrow f(i) \neq f(j)$$

یک مثال از رنگ آمیزی یال در شکل زیر قابل مشاهده است:



## رنگ آمیزی رأس و یال:

رنگ‌آمیزی رأس و یال (Total) به این معنی است که یال‌ها و رأس‌های گراف به صورتی رنگ شوند که هیچ یالی و هیچ رأسی در رنگی مشترک نباشند. همچنین رنگ رئوسِ دو طرف یک یال نیز یکسان نباشد.



در شکل بالا یک گراف که به صورت Total رنگ‌آمیزی شده است مشاهده می‌شود.

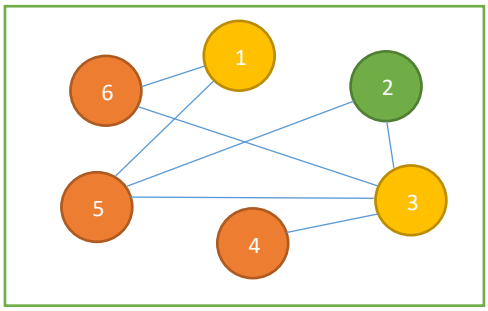
### نحوه‌ی پیاده سازی:

در این پروژه تعدادی از الگوریتم‌های رنگ‌آمیزی گراف باید پیاده سازی شوند. برنامه‌ی شما ابتدا یک فایل ورودی که با فرمت CSV است را می‌خواند (این فایل همان گراف G=(V,E) است). سپس الگوریتم مربوطه به خودتان را برای رنگ‌آمیزی گراف اجرا می‌کند. و در نهایت مشخصات رأس‌های رنگ‌شده‌ی گراف را در یک فایل خروجی می‌نویسد. (توضیح اینکه برای نوشتن فایل خروجی ذکر شماره‌ی رأس به علاوه‌ی شماره‌ی رنگ برای تمامی رئوس کفایت می‌کند). توضیحات مربوط به هر الگوریتم در لینک ویکیپدیای مربوط به آن وجود دارد. مطالعه‌ی این توضیحات برای درک الگوریتم کافیست اما شما می‌توانید از منابع بی‌شماری که در اینترنت در رابطه با آن الگوریتم وجود دارد نیز استفاده کنید.

در زیر یک فایل ورودی نمونه (با نامِ in.txt) و گراف رنگ‌شده‌ی معادل آن را مشاهده می‌کنید:

| | |
|---|---|
| 1 | 6 |
| 6 | 3 |
| 1 | 5 |
| 5 | 2 |
| 5 | 3 |
| 3 | 4 |

گراف رنگ‌شده‌ی بدون جهت معادل



خروجی مورد نظر برای این مثال به صورت زیر خواهد بود (این خروجی را در فایل out.txt ذخیره کنید):

| | |
|---|---|
| 1 | 1 |
| 2 | 3 |
| 3 | 1 |
| 4 | 2 |
| 5 | 2 |
| 6 | 2 |

به این معنی که رأس شماره‌ی ۱ دارای رنگ 1 (نارنجی)، رأس شماره‌ی ۲ دارای رنگ 3 (سبز) و... می‌باشد.

دقت کنید که در مثال بالا رنگ آمیزی از جنس رأس و دقیق (Exact) می‌باشد. لزومی ندارد که الگوریتمِ شما گراف بالا را به این صورت رنگ کند.

## قسمت‌های اختیاری:

- رنگ‌زدن و نمایش گرافیکی گراف دارای نمره‌ی اضافه می‌باشد.
- در صورت پیشنهاد یک روش برای مسأله‌ی "بیشینه‌سازی انتشار" در گراف؛ با استفاده از رنگ‌آمیزی گراف نمره‌ی اضافی به شما تعلق خواهد گرفت. در صورتِ ارائه روش، راه حل خود را توضیح دهید و برای یک گراف فرضی راه حل را توصیف کنید. (توضیحات اضافه برای بیشینه سازی انتشار در لینک زیر قابل مشاهده است:
  (http://homepage.cs.uiowa.edu/~sriram/196/spring12/lectureNotes/Lecture18.pdf
  - دقت کنید که نیازی به پیاده‌سازی روش خود با کد وجود ندارد. و صرفاً توضیح روش پیشنهادی و همچنین ارائه‌ی یک مثال برای آن کفایت می‌کند.

(توضیح اینکه با انجام ندادن این ۲ بخش، هیچ نمره‌ای از شما کسر نخواهد شد.)

لیست پروژه‌ها و لینک ویکیپدیای مربوط به آن‌ها در ادامه آمده است. در صورت وجود هرگونه سؤال یا ابهامی با ایمیل زیر در ارتباط باشید:

صابر غلامی - Sabergholami72@gmail.com

| No. | Algorithm's name | Description | Link |
|---|---|---|---|
| **8** | *Acyclic coloring* | Every 2-chromatic subgraph is acyclic. | https://en.wikipedia.org/wiki/Acyclic_coloring |
| **9** | *B-coloring* | A coloring of the vertices where each color class contains a vertex that has a neighbor in all other color classes. | https://en.wikipedia.org/wiki/B-coloring |
| **10** | *Co coloring* | An improper vertex coloring where every color class induces an independent set or a clique. | https://en.wikipedia.org/wiki/Cocoloring |
| **11** | *Complete coloring* | Every pair of colors appears on at least one edge. | https://en.wikipedia.org/wiki/Complete_coloring |
| **12** | *Defective coloring* | An improper vertex coloring where every color class induces a | https://en.wikipedia.org/wiki/Defective_coloring |

| | | | |
|---|---|---|---|
| | | bounded degree subgraph. | |
| **13** | *Equitable coloring* | The sizes of color classes differ by at most one. | https://en.wikipedia.org/wiki/Equitable_coloring |
| **14** | *Exact coloring* | Every pair of colors appears on exactly one edge. | https://en.wikipedia.org/wiki/Exact_coloring |
| **15** | *Hamiltonian coloring* | Uses the length of the longest path between two vertices, also known as the detour distance. | https://en.wikipedia.org/wiki/Hamiltonian_coloring |
| **16** | *Oriented coloring* | Takes into account orientation of edges of the graph. | https://en.wikipedia.org/wiki/Oriented_coloring |
| **17** | *Radio coloring* | Sum of the distance between the vertices and the difference of their colors is greater than k+1, where k is a positive integer. | https://en.wikipedia.org/wiki/Radio_coloring |
| **18** | *Rank coloring* | If two vertices have the same color *i*, then every path between them contain a vertex with color greater than *i.* | https://en.wikipedia.org/wiki/Cycle_rank |
| **19** | *Sub coloring* | An improper vertex coloring where every color class induces a union of cliques. | https://en.wikipedia.org/wiki/Subcoloring |
| **20** | *Sum coloring* | The criterion of minimalization is the sum of colors. | https://en.wikipedia.org/wiki/Sum_coloring |
| **21** | *T-coloring* | Absolute value of the difference between two colors of adjacent vertices must not belong to fixed set *T.* | https://en.wikipedia.org/wiki/T-coloring |
| **22** | *Total coloring* | Vertices and edges are colored. | https://en.wikipedia.org/wiki/Total_coloring |
| **23** | *L(h,k)-coloring* | Difference of colors at adjacent vertices is at least *h* and difference of colors of vertices at a distance two is at least *k.* | https://en.wikipedia.org/wiki/L(h,_k)-coloring |
| **24** | *Weak coloring* | An improper vertex coloring where every non-isolated node has at least one neighbor with a different color. | https://en.wikipedia.org/wiki/Weak_coloring |
| **25** | *Strong coloring* | Every color appears in every partition of equal size exactly once. | https://en.wikipedia.org/wiki/Strong_coloring |
| **26** | *Greedy coloring* | Color a graph with aspect of greedy algorithms. | - |
| **27** | *Backtrack coloring* | Color a graph with aspect of backtrack algorithms. | - |
| **28** | *Dynamic coloring* | Color a graph with aspect of Dynamic programming. | - |
| **29** | *Divide and conquer coloring* | Color a graph with aspect of divide and conquer algorithms. | - |

| | | | |
|---|---|---|---|
| **30** | *Parallel coloring* | Color a graph with parallel aspect. | - |
| **31** | *Min-color coloring* | Color a graph with minimum number of colors. | - |