معرفی

سید ناصر رضوی n.razavi@tabrizu.ac.ir
۱۳۹۵

مرور درس

- این درس درباره چیست؟
  - حل مسئله و برنامه‌نویسی به همراه مثال‌های کاربردی.
  - الگوریتم. روشی به منظور حل مسئله.
  - ساختمان داده. روشی به منظور ذخیره اطلاعات.

اهداف درس

- تبدیل یک برنامه‌نویس مبتدی به یک برنامه‌نویس خوب
  - آشنایی با ساختمان داده‌های متداول
  - آشنایی با روش‌های مختلف پیاده‌سازی
  - مقایسه کارایی روش‌های مختلف پیاده‌سازی
  - بررسی برخی از کاربردها

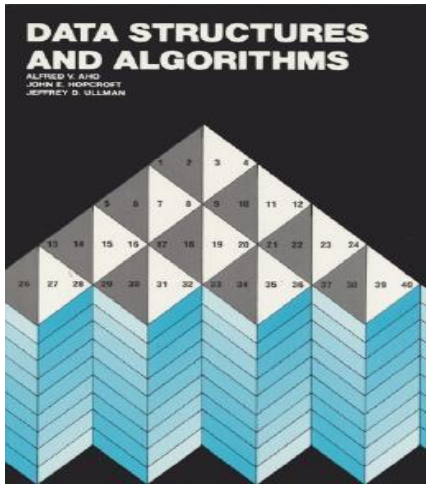
"به نظر من، تفاوت میان یک برنامه‌نویس بد و خوب در این است که برنامه‌نویسان بد بیشتر به کد اهمیت می‌دهند در حالی که برای برنامه‌نویس‌های خوب ساختمان داده‌ها اهمیت بیشتری دارد"

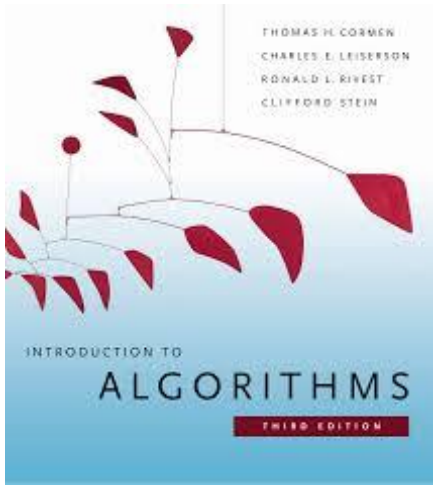
لاینوس تروالدز (مبدع لینوکس)

معرفی سرفصل

- معرفی.
- تحلیل الگوریتم‌ها.
- ساختمان داده‌های پایه‌ای.
  - لیست، پشته، صف، کیسه
- درخت.
  - درخت دودویی، درخت جستجوی دودویی، درخت جستجوی دودویی متوازن
- گراف.
  - بازنمایی گراف، الگوریتم‌های ابتدایی گراف (جستجوی عمقی و سطحی)
  - یافتن کوتاه‌ترین مسیرها، محاسبه‌ی درخت پوشای کمینه
- مرتب‌سازی.
  - درخت هرمی، مرتب‌سازی هرمی، مرتب‌سازی ادغامی

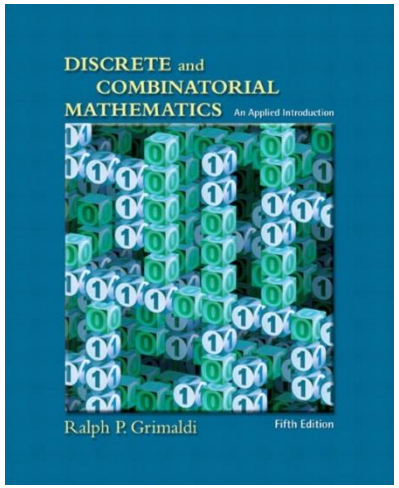
فهرست منابع و مراجع



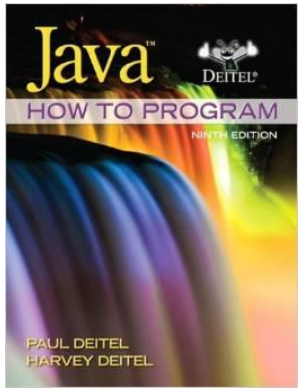


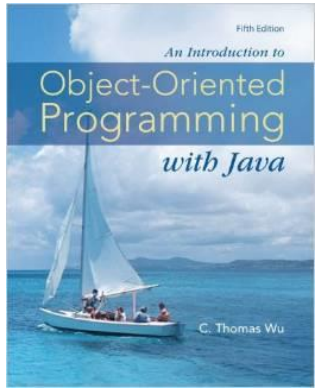
- ساختمان‌داده‌ها و الگوریتم‌ها.
  - اِیهو، هاپکرافت، آلمن
- داده‌ساختارها و مبانی الگوریتم‌ها.
  - محمد قدسی، انتشارات فاطمی
- مقدمه‌ای بر الگوریتم‌ها.
  - کورمِن، لایسرسون، ریوست، اشتاین (ویراست سوم)
- الگوریتم‌ها.
  - رابرت سژویک، کوین وین (ویراست چهارم)

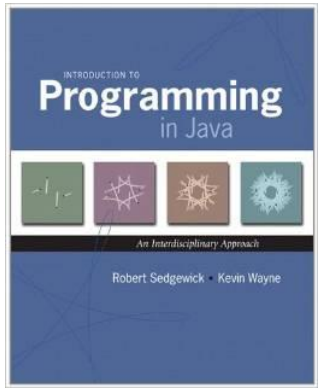
پیش‌نیازها



- ریاضیات گسسته و ترکیباتی.
  - ریاضیات گسسته و ترکیباتی، گریمالدی (ویراست پنجم)
- برنامه‌سازی مقدماتی و پیشرفته.
  - چگونه در جاوا برنامه بنویسیم، دیتل و دیتل (ویراست نهم)
  - معرفی برنامه نویسی شی‌گرا با جاوا، توماس وو (ویراست پنجم)
  - آشنایی با برنامه‌نویسی در جاوا: یک رویکرد میان رشته‌ای، سژویک و وین



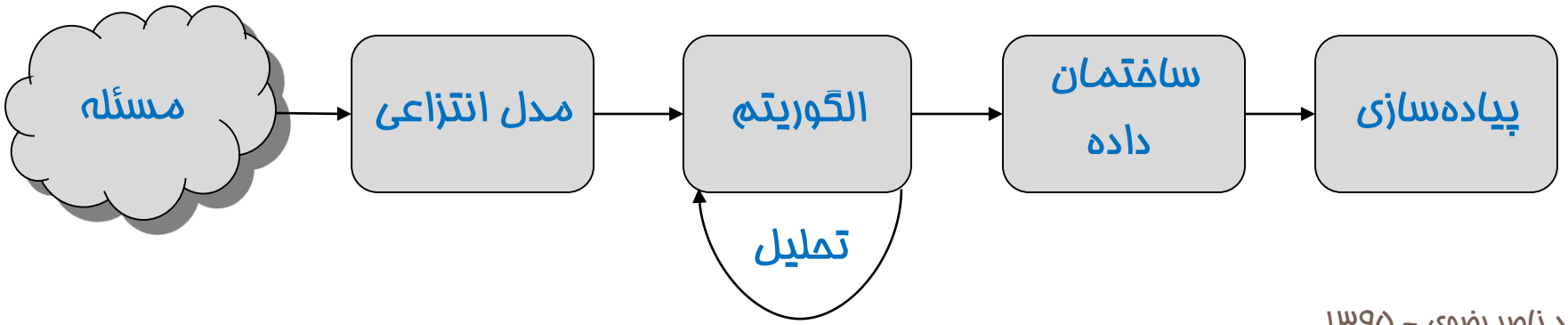




مراحل حل مسئله

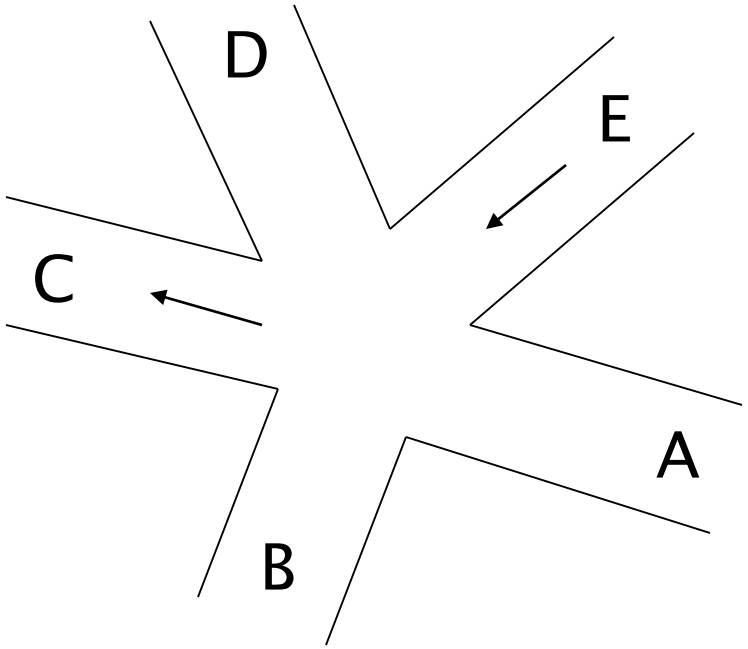
- مراحل حل مسئله.

۱. ایجاد یک مدل انتزاعی از مسئله -- مدل
۲. طراحی الگوریتم برای حل مدل -- الگوریتم
۳. تحلیل الگوریتم برای حل کارا -- الگوریتم صحیح و کارا
۴. تعریف ساختمان داده‌های لازم -- ساختمان داده‌ها
۵. پیاده‌سازی الگوریتم ارائه شده -- برنامه قابل اجرا روی ماشین
۶. ...



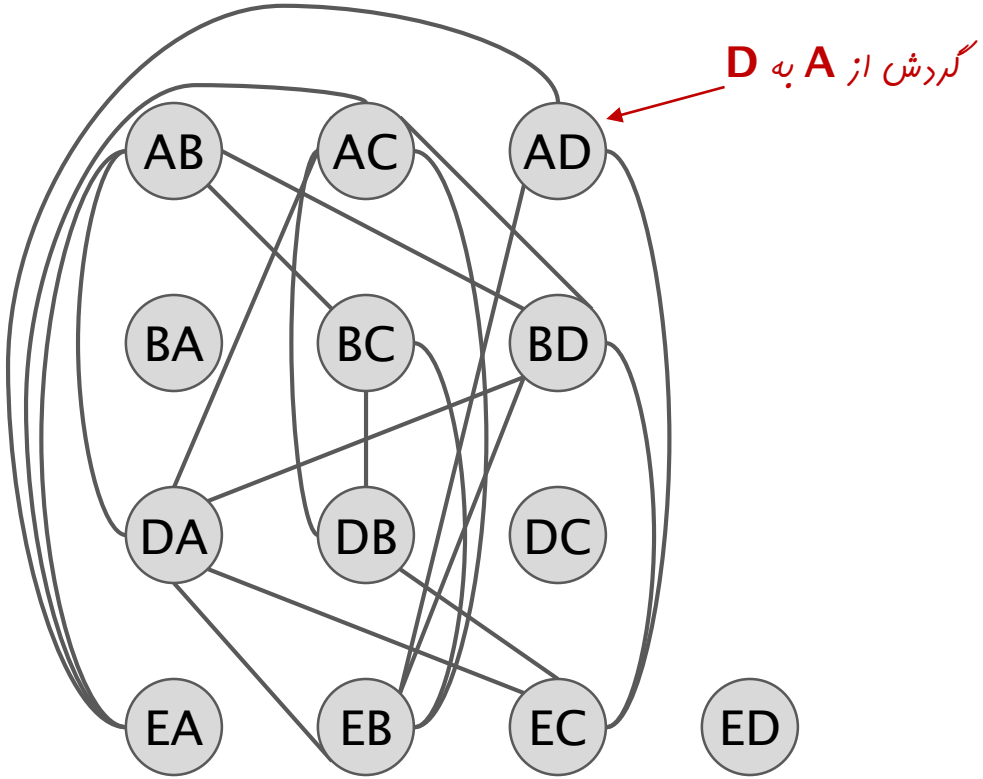
مثال

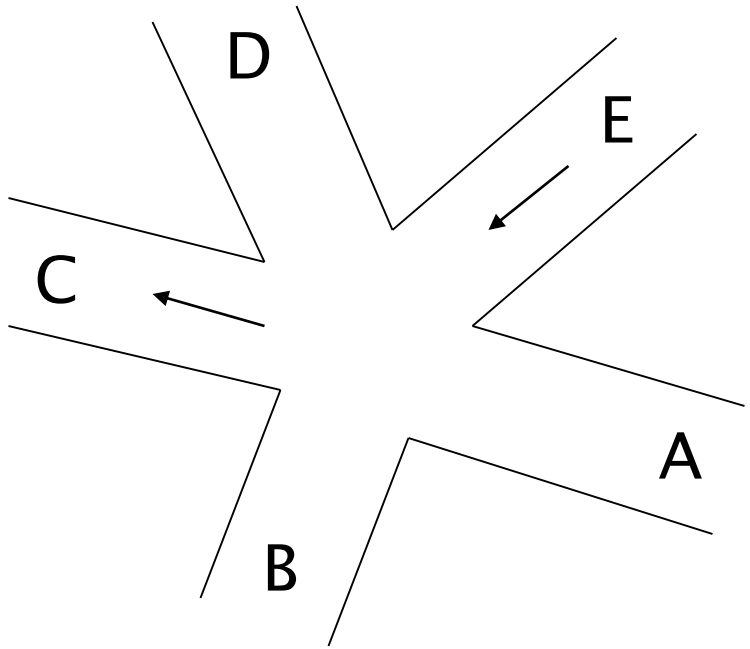
- مسئله. برنامه‌ریزی چراغ راهنمایی تقاطع زیر با استفاده از حداقل تعداد رنگ‌ها.



مدل‌سازی

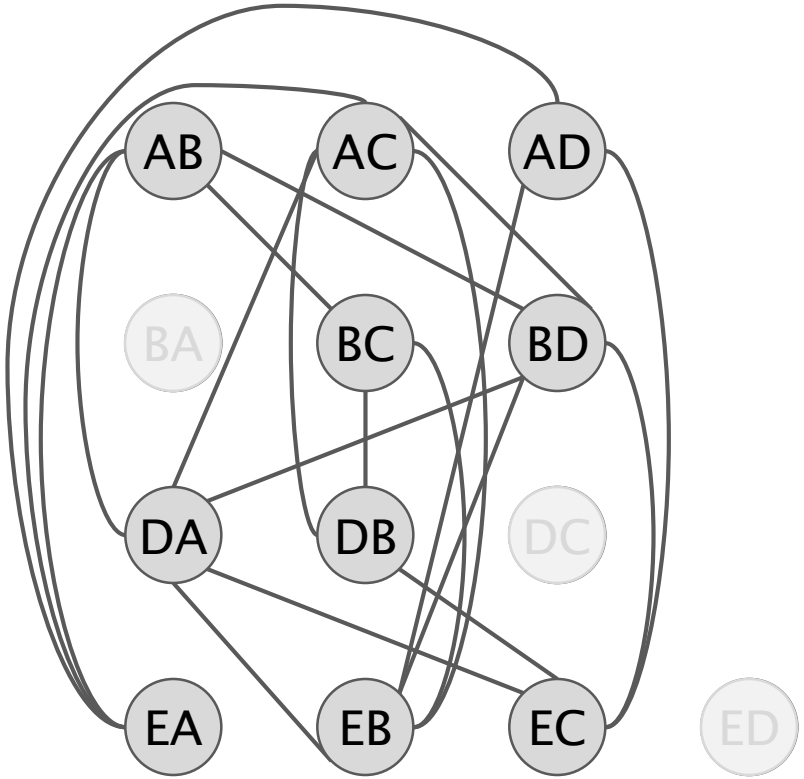
- مدل‌سازی گردش‌ها.
  - رسم یک یال بین هر دو گردشی که همزمان میسر نیستند.





۱) مدل‌سازی: گراف

- مسئله‌ی رنگ‌آمیزی گراف.
  - حل این مسئله به حل مسئله‌ی رنگ‌آمیزی گراف می‌انجامد، به طوری که هیچ دو رأس مجاوری همرنگ نباشند.



- توجه.
  - رئوس BA، DC و ED نیاز به رنگ‌آمیزی ندارند. [گردش به راست]

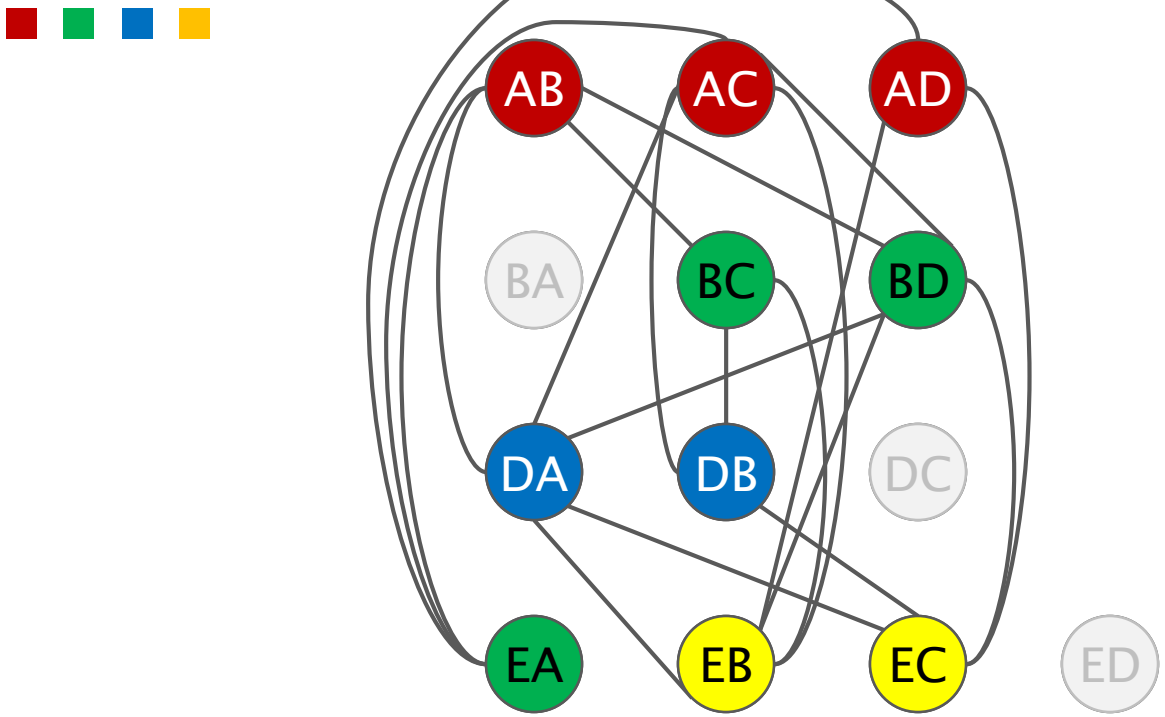
- مسایل «ان‌پی کامل».
  - مسئله‌ی رنگ‌آمیزی گراف یک مسئله‌ی ان‌پی کامل است!
  - یعنی، تا کنون راه‌حل کارایی برای آن پیدا نشده و از طرفی عدم وجود راه‌حل کارا اثبات نشده است.

۲) طراحی الگوریتم: روش حریصانه

- روش حریصانه. یک راه‌حل سریع که لزوما پاسخ بهینه نمی‌دهد.
- یک الگوریتم حریصانه به منظور رنگ‌آمیزی گراف.
  - در ابتدا سعی می‌کنیم بیشترین تعداد رئوس ممکن را با رنگ اول رنگ کنیم. سپس بیشترین تعداد رئوس رنگ نشده‌ی ممکن را با رنگ دوم رنگ می‌کنیم و این کار را تا زمان رنگ شدن تمامی رئوس ادامه می‌دهیم.
- الگوریتم رنگ‌آمیزی رئوس با استفاده از یک رنگ جدید.
  - یک رأس بدون رنگ را انتخاب کن و آن را با رنگ جدید رنگ بزن.
  - لیست رئوس رنگ نشده را پیمایش کن. به ازای هر رأس رنگ نشده، تعیین کن که آیا این رأس با یکی از رئوسی که تاکنون با رنگ جدید رنگ شده یالی دارد یا خیر. اگر چنین یالی وجود نداشت، این رأس را با رنگ جدید رنگ بزن.

۲) طراحی الگوریتم: روش حریصانه

- رنگ‌آمیزی گراف مربوط به مسئله‌ی چراغ راهنمایی.



یک راه‌حل برای مسئله

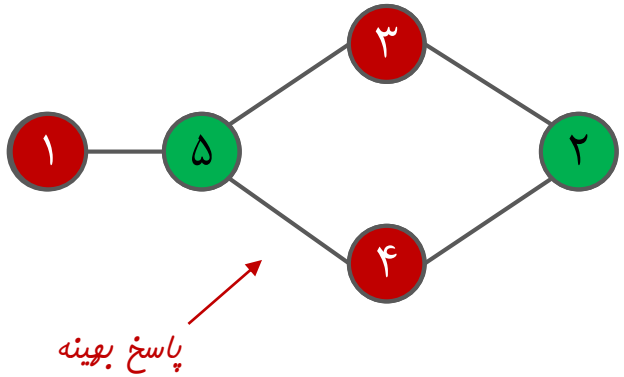
- رنگ‌آمیزی گراف مربوط به مسئله‌ی چراغ راهنمایی.

color	*turns*	*extras*
	AB, AC, AD	BA, DC, ED
	BC, BD, EA	BA, DC, ED
	DA, DB	AD, BA, DC, ED
	EB, EC	BA, DC, EA, ED

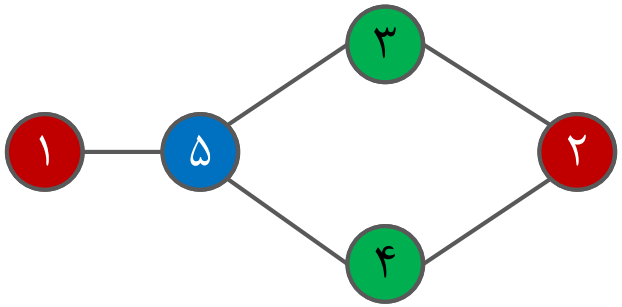
۲) طراحی الگوریتم: روش حریصانه

- یک مثال نقض برای الگوریتم حریصانه.

رنگ‌آمیزی با ۲ رنگ

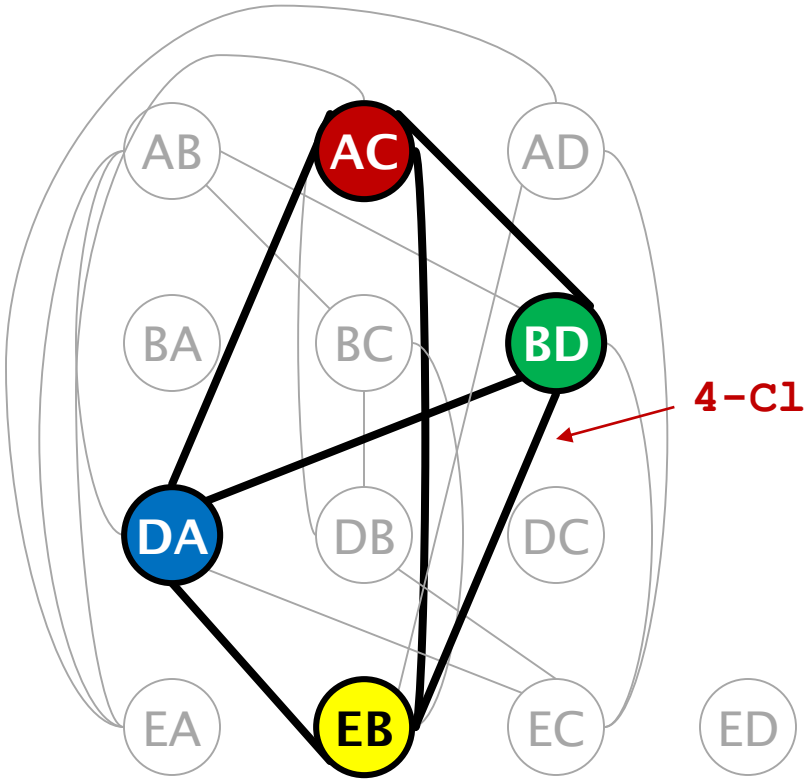


رنگ‌آمیزی با ۳ رنگ



ارزیابی راه‌حل مسئله

- **k-clique**.
  - مجموعه‌ای از **k** رأس به طوری که هر زوج از این رئوس به وسیله‌ی یک یال به هم متصل باشند.
  - بدیهی است تعداد رنگ‌های لازم برای رنگ‌آمیزی یک **k-clique** برابر **k** است؟!



- این گراف شامل یک **4-Clique** است و بنابراین حداقل تعداد رنگ‌های لازم برابر ۴ است و در نتیجه راه‌حل یافته شده بهینه است.

الگوریتم حریصانه: پیاده‌سازی در جاوا

```
public void greedy(Graph G, SET<Integer> newColor) {
   newColor = new SET<Integer>();
   for (int v = 0; v < G.V(); v++) {
      if (!G.isColored(v)) {
         boolean found = false;
         for (int w : newColor)
            if (G.hasEdge(v, w) {found = true; break;}
         if (!found) {
            G.color(v);
            newColor.add(v);
         }
      }
   }
}
```

انواع داده‌ای انتزاعی (ADT)

public class	Graph	
	Graph(int V)	ایجاد یک گراف تهی با V رأس
boolean	hasEdge(int v, int w)	بررسی وجود یال v- w
boolean	isColored(int v)	آیا رأس v رنگ شده است؟
void	color(int v)	رنگ زدن رأس v
int	V()	برگرداندن تعداد رئوس

public class	SET	
	SET( )	ایجاد یک مجموعه تهی
void	add(int v)	افزودن عنصر v به مجموعه

نوع داده‌ای، ساختمان داده، ADT

- نوع داده‌ای. یک مجموعه از مقادیر به همراه عملیات قابل انجام بر روی آنها.
  - مثال: نوع داده‌ای بولی
    - مقادیر **`true`** و **`false`**
    - عملیات: ترکیب عطفی، ترکیب فصلی و ...
- نوع داده‌ای انتزاعی. یک مدل ریاضی، به همراه عملیات تعریف شده روی مدل.
  - مثال: صف، پشته، درخت و گراف و ...
- ساختمان داده. روشی به منظور سازماندهی اطلاعات در حافظه.
  - مثال: آرایه، رکورد، فایل

انواع داده‌ای انتزاعی (ADT)

- نوع داده‌ی انتزاعی. یک مدل ریاضی به همراه مجموعه‌ای از عملیات تعریف شده بر روی آن.
  - مثال: مجموعه‌ای از اعداد صحیح به همراه عملیات اجتماع، اشتراک و تفاضل.
  - مثال: یک گراف به همراه مجموعه عملیات قابل انجام بر روی آن مانند ایجاد گراف، دسترسی به رئوس گراف، بررسی وجود یال بین دو رأس دلخواه و ...

- مزایای **ADT**
  - سادگی در تغییر پیاده‌سازی آن
  - خارج از بخشی که عملیات ADT تعریف شده است، می‌توان با ADT به عنوان یک نوع داده‌ای اولیه رفتار نمود؛ یعنی توجهی به پیاده‌سازی آن نداریم.

خلاصه‌ی مراحل برنامه‌نویسی

الگوریتم
- مدل ریاضی

شبه کد
- نوع داده انتزاعی

برنامه
- ساختمان داده