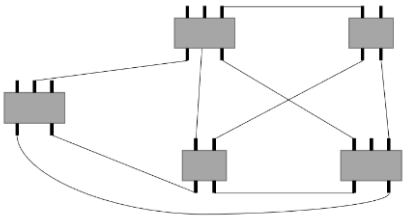
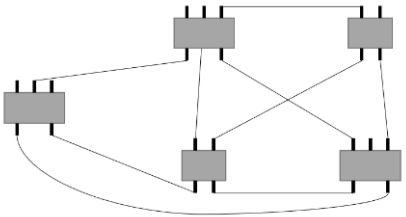
第10章 平面

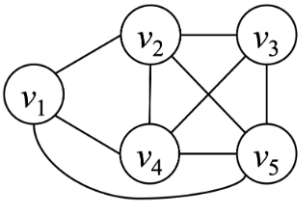
程龚

电路版图设计问题

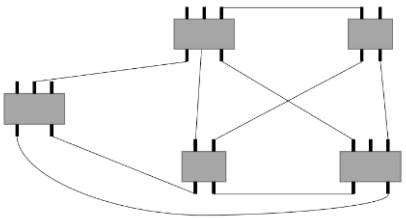


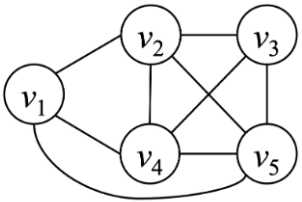
电路版图设计问题

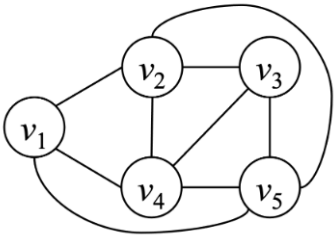




电路版图设计问题







本次课的主要内容

10.1 可平面图

10.2 面的染色

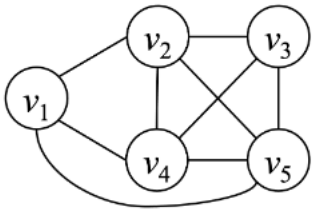
本次课的主要内容

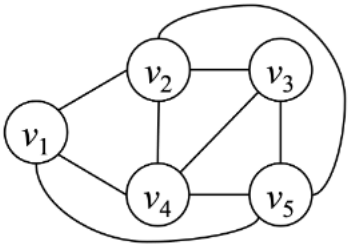
10.1 可平面图

10.2 面的染色

可平面图

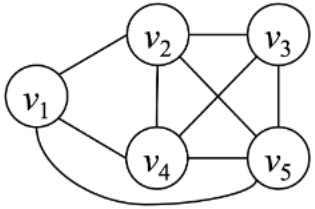
- 在平面上的**画法**
  - 单射函数dr:
    将顶点$v \in V$映射到平面上的坐标点$dr(v)$
    将边$(u, v) \in E$映射到平面上的$dr(u)$-$dr(v)$曲线

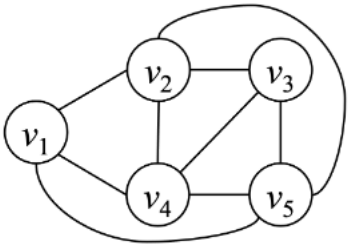




可平面图

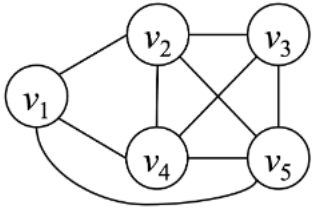
- 在平面上的画法
  - 单射函数dr:
    将顶点$v \in V$映射到平面上的坐标点$dr(v)$
    将边$(u, v) \in E$映射到平面上的$dr(u)$-$dr(v)$曲线
- **可平面图**
  - 任意两条边映射到的平面曲线不交叉
    （没有除端点外的公共坐标点）
  - **平面嵌入**：画法
    平面图：映射到平面上的结果

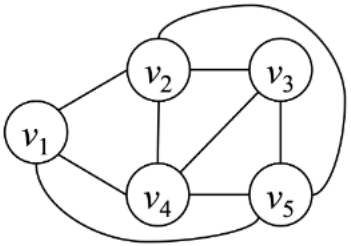




可平面图

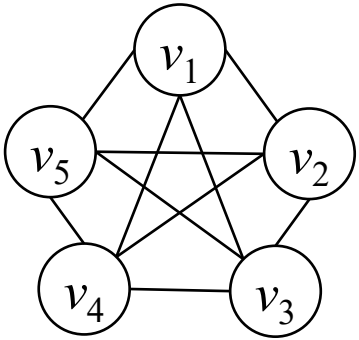
- 在平面上的画法
  - 单射函数dr:
    将顶点$v \in V$映射到平面上的坐标点$dr(v)$
    将边$(u, v) \in E$映射到平面上的$dr(u)$-$dr(v)$曲线
- 可平面图
  - 任意两条边映射到的平面曲线不交叉
    （没有除端点外的公共坐标点）
  - 平面嵌入：画法
    平面图：映射到平面上的结果
- **不可平面图**





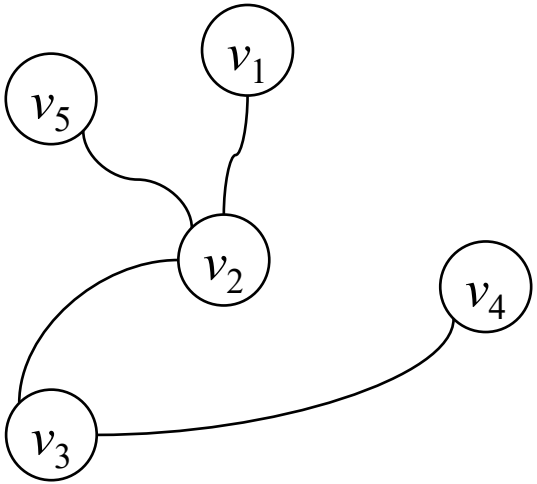
可平面图

- 完全图K_1, K_2, K_3, K_4, K_5是可平面图吗?



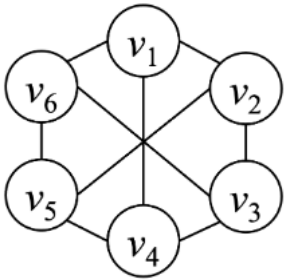
可平面图

- 完全图K_1, K_2, K_3, K_4, K_5是可平面图吗?
- 树是可平面图吗?



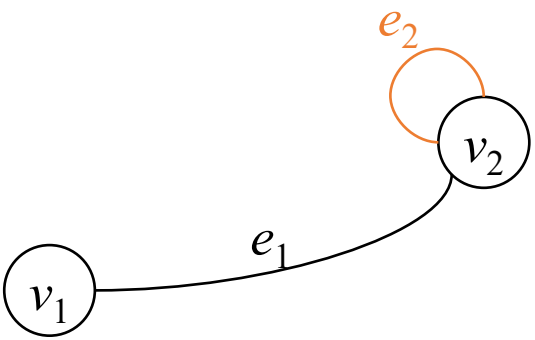
可平面图

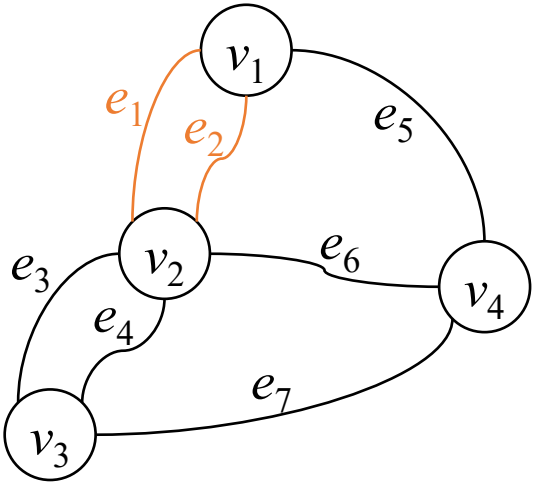
- 完全图K_1, K_2, K_3, K_4, K_5是可平面图吗?
- 树是可平面图吗?
- 完全二分图$K_{1,1}$, $K_{2,2}$, $K_{3,3}$是可平面图吗?



可平面图

- 完全图K_1, K_2, K_3, K_4, K_5是可平面图吗?
- 树是可平面图吗?
- 完全二分图$K_{1,1}$, $K_{2,2}$, $K_{3,3}$是可平面图吗?
- 自环和重边影响图的可平面性吗?





可平面图

- 完全图K_1, K_2, K_3, K_4, K_5是可平面图吗?
- 树是可平面图吗?
- 完全二分图$K_{1,1}$, $K_{2,2}$, $K_{3,3}$是可平面图吗?
- 自环和重边影响图的可平面性吗?
- 可平面图的子图是可平面图吗?

可平面图

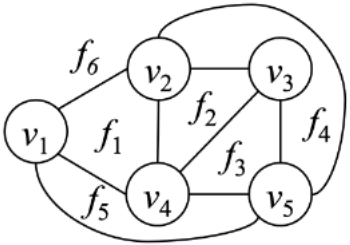
- 完全图K_1, K_2, K_3, K_4, K_5是可平面图吗?
- 树是可平面图吗?
- 完全二分图$K_{1,1}$, $K_{2,2}$, $K_{3,3}$是可平面图吗?
- 自环和重边影响图的可平面性吗?
- 可平面图的子图是可平面图吗?
- 完全图K_6和完全二分图$K_{4,4}$是可平面图吗?

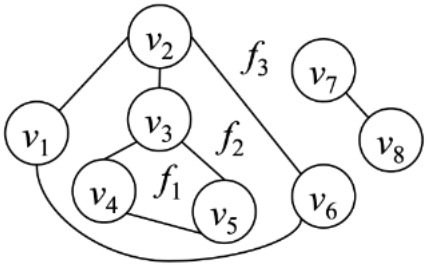
可平面图

- 完全图K_1, K_2, K_3, K_4, K_5是可平面图吗?
- 树是可平面图吗?
- 完全二分图$K_{1,1}$, $K_{2,2}$, $K_{3,3}$是可平面图吗?
- 自环和重边影响图的可平面性吗?
- 可平面图的子图是可平面图吗?
- 完全图K_6和完全二分图$K_{4,4}$是可平面图吗?
- 若图G的所有连通分支都是可平面图，则G是可平面图吗?

可平面图

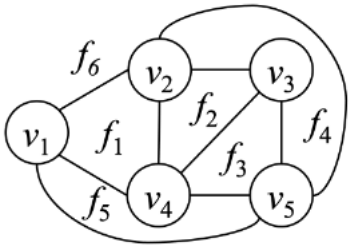
- **面**
  - 平面图将平面分隔出的极大相连区域（不含平面图自身）

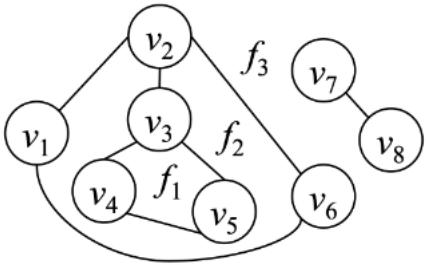




可平面图

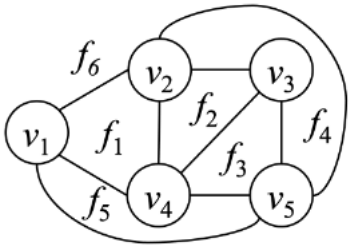
- 面
  - 平面图将平面分隔出的极大相连区域（不含平面图自身）
- **无限面（外部面）**
  - 面积无限的面

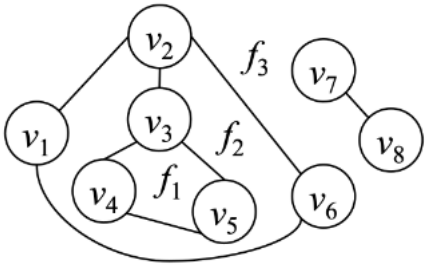




可平面图

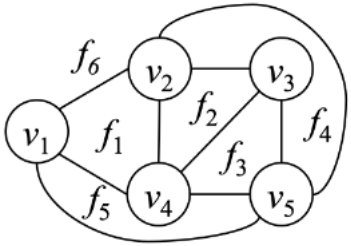
- 面
  - 平面图将平面分隔出的极大相连区域（不含平面图自身）
- 无限面（外部面）
  - 面积无限的面
- **有限面（内部面）**
  - 面积有限的面

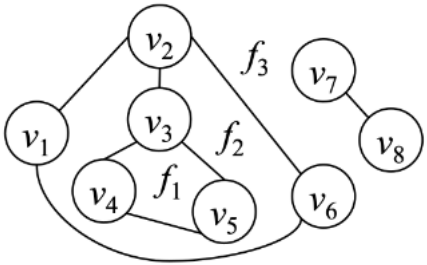




可平面图

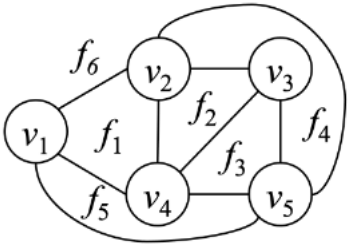
- 面
  - 平面图将平面分隔出的极大相连区域（不含平面图自身）
- 无限面（外部面）
  - 面积无限的面
- 有限面（内部面）
  - 面积有限的面
- **面数**
  - 面的数量，记作$\varphi(H)$

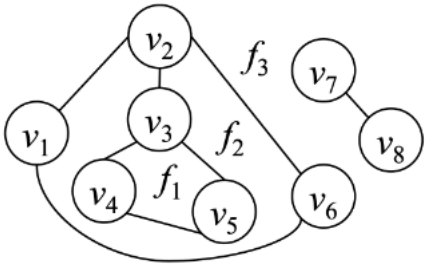




可平面图

- 面
  - 平面图将平面分隔出的极大相连区域（不含平面图自身）
- 无限面（外部面）
  - 面积无限的面
- 有限面（内部面）
  - 面积有限的面
- 面数
  - 面的数量，记作$\varphi(H)$
- 平面图可以有多少个无限面?





可平面图

- 对于任意一个连通图G的平面图H: $v(G) - \varepsilon(G) + \varphi(H) = 2$

可平面图

- 对于任意一个连通图G的平面图H：$v(G)-\varepsilon(G)+\varphi(H)=2$
  - 采用数学归纳法，对$v(G)$归纳
  - $v(G)=1$时：
  - 假设$v(G)=k$时成立，则$v(G)=k+1$时：

可平面图

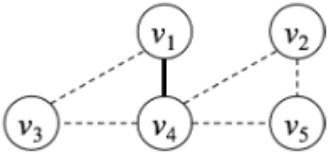
- 对于任意一个连通图G的平面图H：$v(G)-\varepsilon(G)+\varphi(H)=2$
  - 采用数学归纳法，对$v(G)$归纳
  - $v(G)=1$时：

  - 假设$v(G)=k$时成立，则$v(G)=k+1$时：

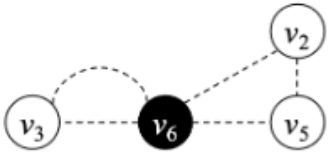
可平面图

- 对于任意一个连通图G的平面图H：$v(G) - \varepsilon(G) + \varphi(H) = 2$
  - 采用数学归纳法，对$v(G)$归纳
  - $v(G) = 1$时：

  - 假设$v(G) = k$时成立，则$v(G) = k + 1$时：
    收缩一条非自环边



(a)



(b)

可平面图

- 对于任意一个有w个连通分支的图G的平面图H:
  $v(G) - \varepsilon(G) + \varphi(H) = w + 1$

可平面图

- 对于任意一个有w个连通分支的图G的平面图H:
  $v(G) - \varepsilon(G) + \varphi(H) = w + 1$

$$
\begin{aligned}
w \cdot 2 &= \sum_{i=1}^{w} \nu(G_i) - \epsilon(G_i) + \phi(H_i) \\
&= \sum_{i=1}^{w} \nu(G_i) - \sum_{i=1}^{w} \epsilon(G_i) + \sum_{i=1}^{w} \phi(H_i)
\end{aligned}
$$

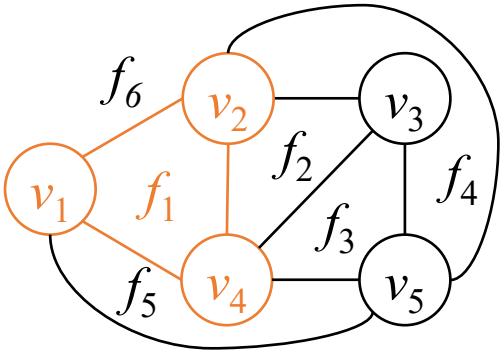
可平面图

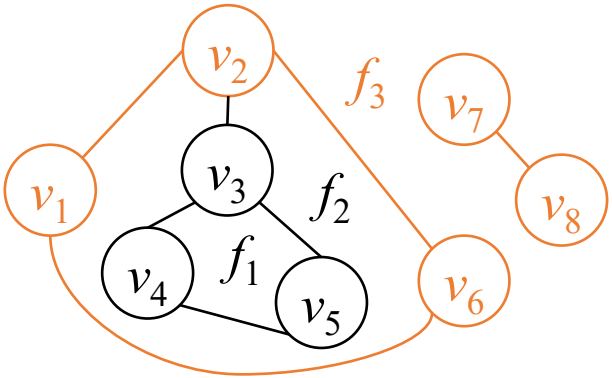
- 对于任意一个有w个连通分支的图G的平面图H:
  $v(G) - \varepsilon(G) + \varphi(H) = w + 1$

$$
\begin{aligned}
w \cdot 2 &= \sum_{i=1}^{w} \nu(G_i) - \epsilon(G_i) + \phi(H_i) \\
&= \sum_{i=1}^{w} \nu(G_i) - \sum_{i=1}^{w} \epsilon(G_i) + \sum_{i=1}^{w} \phi(H_i) \\
&= \nu(G) - \epsilon(G) + (\phi(H) + (w-1)),
\end{aligned}
$$

可平面图

- 面的**边界**
  - 在平面上相邻的坐标点对应的顶点和边形成的子图

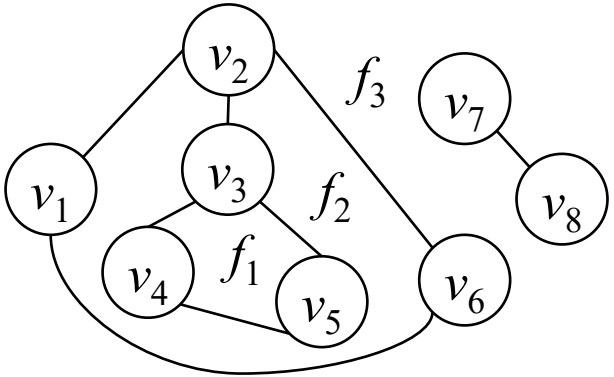




可平面图

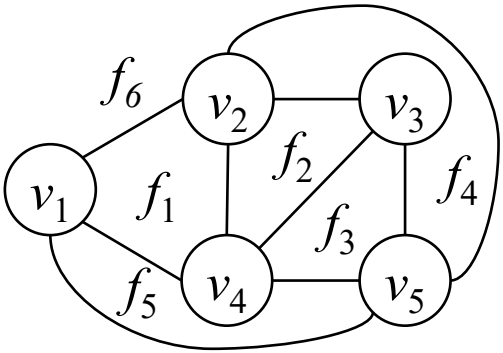
- 什么样的边在两个面的边界中？
  什么样的边只在一个面的边界中？

随堂小测



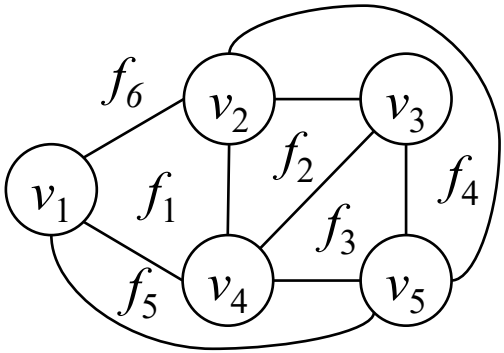
可平面图

- 什么样的边在两个面的边界中？
  什么样的边只在一个面的边界中？
- 平面图的两个不同的面的边界可以完全相同吗？



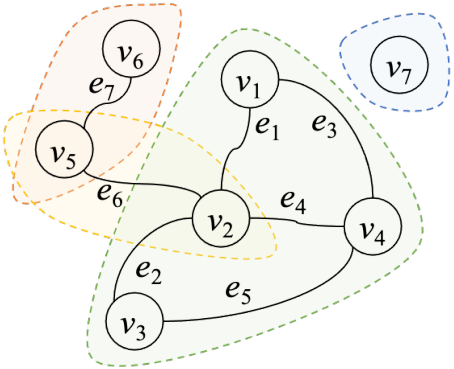
可平面图

- 什么样的边在两个面的边界中？
  什么样的边只在一个面的边界中？
- 平面图的两个不同的面的边界可以完全相同吗？
- 对于任意一个图的平面图（面数至少为2）的无限面，
  你能找到该图的另一个平面图的有限面，
  使两个面的边界是相同的子图吗？反之能吗？



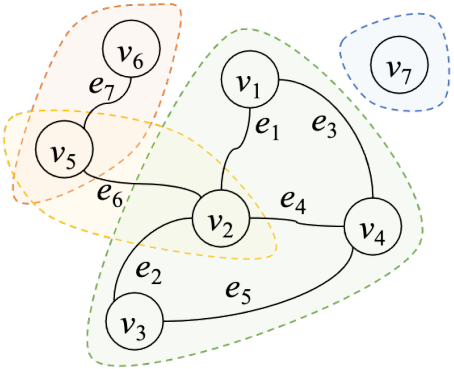
可平面图

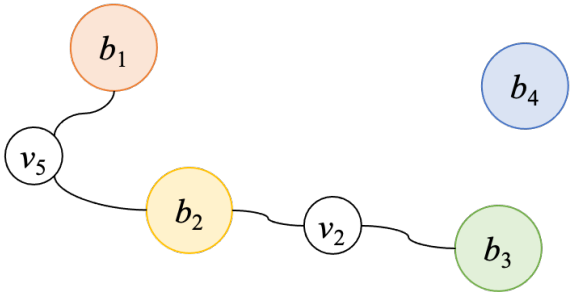
- 什么样的边在两个面的边界中？
  什么样的边只在一个面的边界中？
- 平面图的两个不同的面的边界可以完全相同吗？
- 对于任意一个图的平面图（面数至少为2）的无限面，
  你能找到该图的另一个平面图的有限面，
  使两个面的边界是相同的子图吗？反之能吗？
- 若图G的所有块都是可平面图，则G是可平面图吗？



可平面图

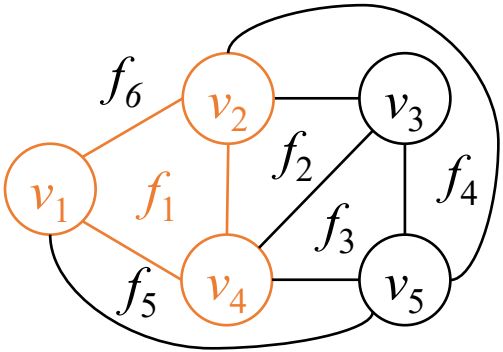
- 什么样的边在两个面的边界中？
  什么样的边只在一个面的边界中？
- 平面图的两个不同的面的边界可以完全相同吗？
- 对于任意一个图的平面图（面数至少为2）的无限面，
  你能找到该图的另一个平面图的有限面，
  使两个面的边界是相同的子图吗？反之能吗？
- 若图G的所有块都是可平面图，则G是可平面图吗？

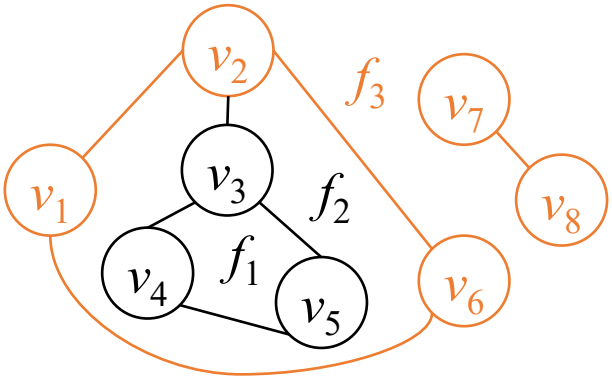




可平面图

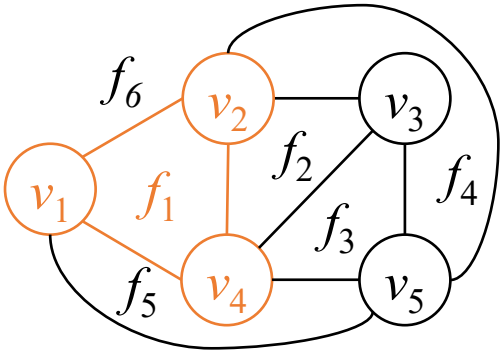
- 面的**长度**
  - 从平面分隔出面的闭路线的长度的和，记作$l(f)$

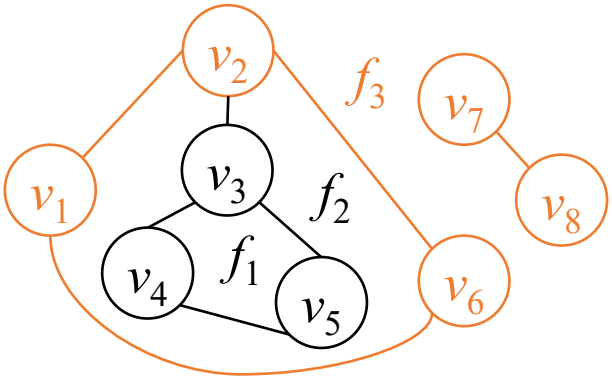




可平面图

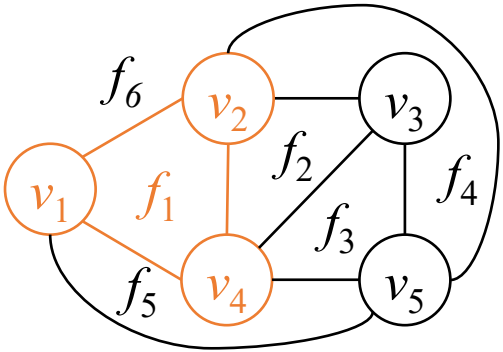
- 面的长度
  - 从平面分隔出面的闭路线的长度的和，记作$l(f)$
- 面的长度和面的边界的边数相等吗？

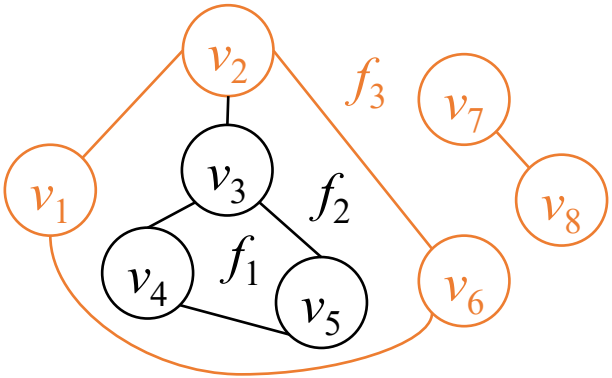




可平面图

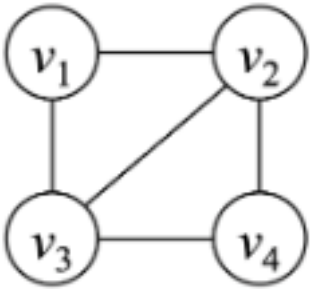
- 对于任意一个图G的平面图H,
  H的所有面的长度的和等于G的边数的 2 倍: $\sum_{i=1}^{\phi(H)} l(f_i) = 2 \cdot \epsilon(G)$

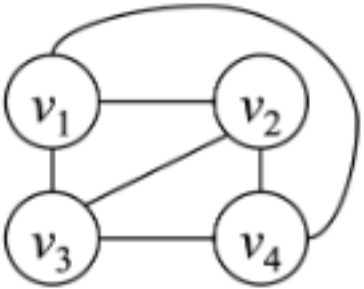




可平面图

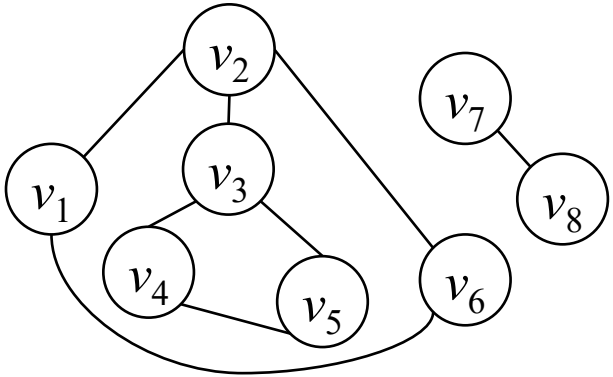
- **极大可平面图**
  - 简单可平面图，且不是任何简单可平面图的生成真子图





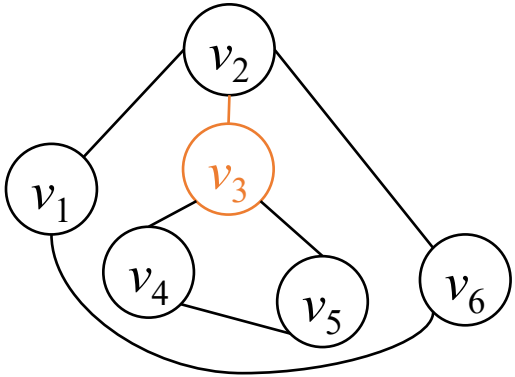
可平面图

- 极大可平面图连通吗?



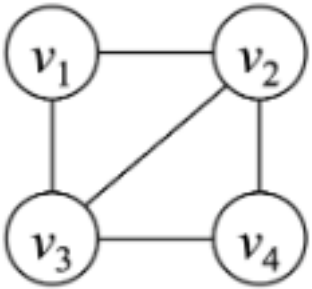
可平面图

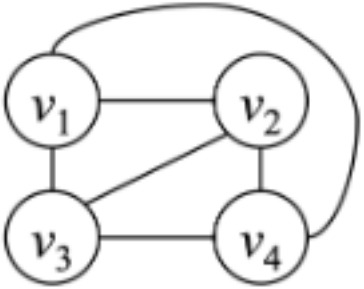
- 极大可平面图连通吗?
- 阶至少为3的极大可平面图可以有割点或割边吗?



可平面图

- 极大可平面图连通吗?
- 阶至少为3的极大可平面图可以有割点或割边吗?
- 阶至少为3的极大可平面图的平面图的每个面的边界有什么特征?





可平面图

- 对于任意一个阶为n（$n \geq 3$）的极大可平面图G：$\varepsilon(G) = 3n-6$

可平面图

- 对于任意一个阶为n（$n \geq 3$）的极大可平面图G：$\varepsilon(G) = 3n-6$
  - G的平面图H的每个面的长度均为3
    则$2 \cdot \varepsilon(G) = 3 \cdot \varphi(H)$，即$\varphi(H) = 2/3 \cdot \varepsilon(G)$
  - 代入连通图的欧拉公式：$n - \varepsilon(G) + \varphi(H) = 2$

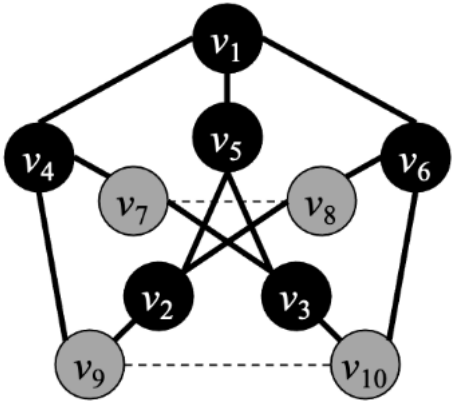
可平面图

- 对于任意一个阶为n（$n \geq 3$）的极大可平面图G：$\varepsilon(G) = 3n-6$
  - G的平面图H的每个面的长度均为3
    则$2 \cdot \varepsilon(G) = 3 \cdot \varphi(H)$，即$\varphi(H) = 2/3 \cdot \varepsilon(G)$
  - 代入连通图的欧拉公式：$n - \varepsilon(G) + \varphi(H) = 2$

- 对于任意一个阶为n（$n \geq 3$）的简单可平面图G：$\varepsilon(G) \leq 3n-6$

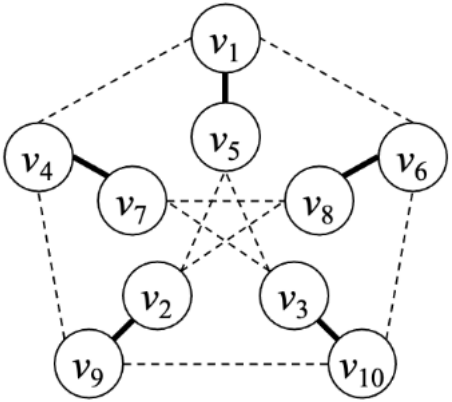
可平面图

- 可平面图的充要条件：
  - G不含这样的子图：可通过对K_5或$K_{3,3}$进行若干次边剖分得到



可平面图

- 可平面图的充要条件：
  - G不含这样的子图：可通过对K_5或$K_{3,3}$进行若干次边剖分得到
  - G不含这样的子图：可通过若干次边收缩得到K_5或$K_{3,3}$



可平面图

- Kazimierz Kuratowski，1896年出生于波兰
- Klaus Wagner，1910年出生于德国

https://upload.wikimedia.org/wikipedia/commons/1/18/Kazimierz_Kuratowski.jpg
https://en.wikipedia.org/wiki/File:Wagner_and_Harary.jpg

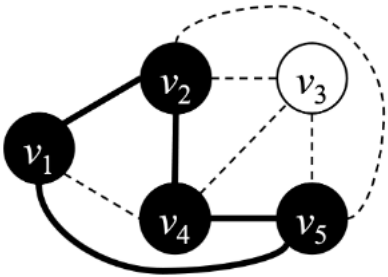
可平面图

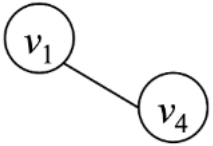
- 如何判定一个图是否为可平面图?

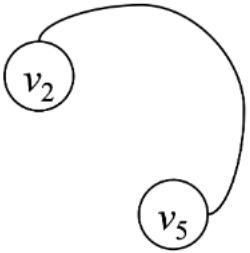
可平面图

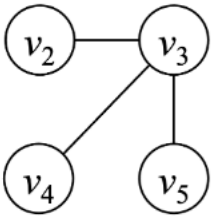
- 对于图G的子图$H = <V_H, E_H>$，若G的子图B恰由
  - 不在E_H中但端点在V_H中的一条边
  - 或$G - V_H$的一个连通分支以及端点分别在该连通分支和V_H中的所有边

  组成，则
  - B称作G的***H*片段**
  - B和H的公共顶点称作B的**固定点**



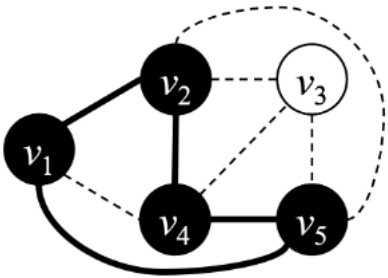


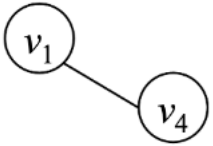


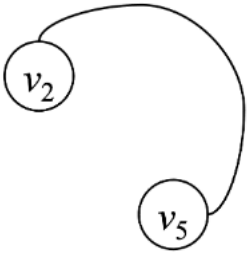


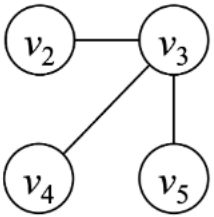
可平面图

- H片段连通吗?



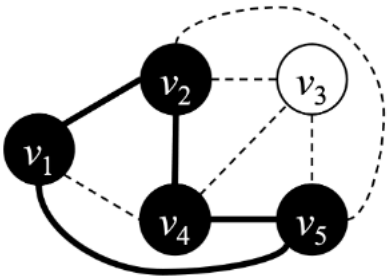


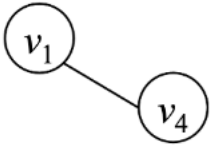


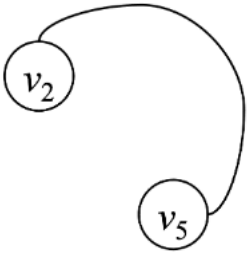


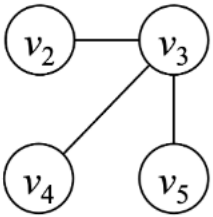
可平面图

- H片段连通吗?
- 对于连通图G，子图H和所有H片段的并是什么?



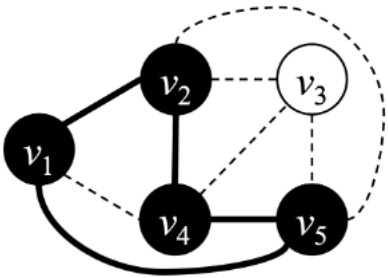


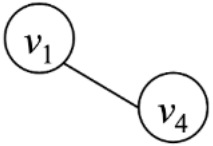


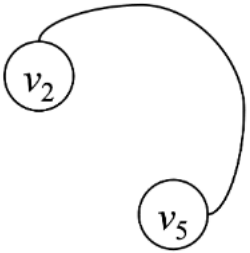


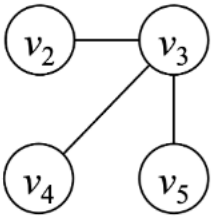
可平面图

- H片段连通吗?
- 对于连通图G，子图H和所有H片段的并是什么?
- 子图H和H片段的边集相交吗？两个H片段的边集相交吗?









可平面图

- DMP算法
  - 逐步尝试构造图的平面嵌入，每步尝试将
    当前已映射到平面上的子图的一个片段中的一条路映射到平面上

算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

$H = \langle V_H, E_H \rangle \leftarrow G$ 中任意一个圈;
将 H 映射到平面上;
while $H \neq G$ **do**
  foreach $B_i \in G$ 的所有 H 片段 **do**
    $B_i.\mathrm{F} \leftarrow \emptyset$;
    foreach $f_j \in H$ 的平面图的所有面 **do**
      if f_j 的边界含 B_i 的所有固定点 **then**
        $B_i.\mathrm{F} \leftarrow B_i.\mathrm{F} \cup \{f_j\}$;
    if $B_i.\mathrm{F} = \emptyset$ **then**
      输出 (G 是不可平面图);
    else if $|B_i.\mathrm{F}| = 1$ **then**
      $B \leftarrow B_i$;
  if $B =$ null **then**
    $B \leftarrow G$ 的任意一个 H 片段;
  $P \leftarrow B$ 中任意两个固定点间的一条路;
  $f \leftarrow B.\mathrm{F}$ 中任意一个面;
  将 P 映射到平面上的 f 内;
  $V_H \leftarrow V_H \cup P$经过的顶点的集合;
  $E_H \leftarrow E_H \cup P$经过的边的集合;
输出 (G 是可平面图);

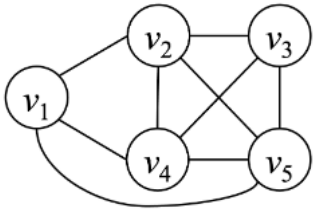
可平面图

- 对于2连通图G，从G中任意一个圈开始

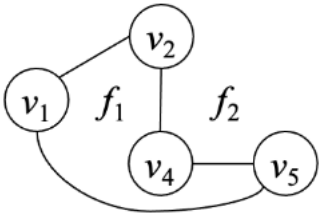
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```



G



H

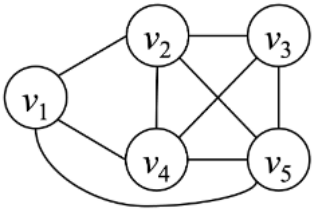
可平面图

- 每轮while循环尝试将
  当前已映射到平面上的子图H的一个片段B中的一条路P映射到平面上

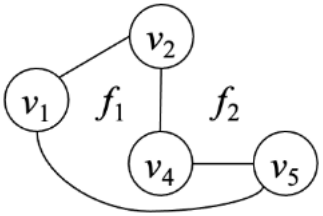
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

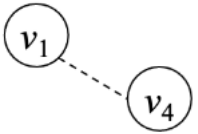
1 $H = \langle V_H, E_H \rangle \leftarrow G$ 中任意一个圈;
2 将 H 映射到平面上;
3 **while** $H \neq G$ **do**
4 **foreach** $B_i \in G$ 的所有 H 片段 **do**
5 B_i.F $\leftarrow \emptyset$;
6 **foreach** $f_j \in H$ 的平面图的所有面 **do**
7 **if** f_j 的边界含 B_i 的所有固定点 **then**
8 B_i.F $\leftarrow B_i$.F $\cup \{f_j\}$;
9 **if** B_i.F $= \emptyset$ **then**
10 输出 (G 是不可平面图);
11 **else if** $|B_i$.F$| = 1$ **then**
12 $B \leftarrow B_i$;
13 **if** $B =$ null **then**
14 $B \leftarrow G$ 的任意一个 H 片段;
15 $P \leftarrow B$ 中任意两个固定点间的一条路;
16 $f \leftarrow B$.F 中任意一个面;
17 将 P 映射到平面上的 f 内;
18 $V_H \leftarrow V_H \cup P$经过的顶点的集合;
19 $E_H \leftarrow E_H \cup P$经过的边的集合;
20 输出 (G 是可平面图);



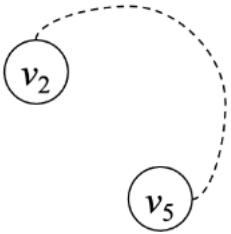
G



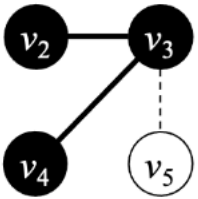
H



B_1



B_2



B_3

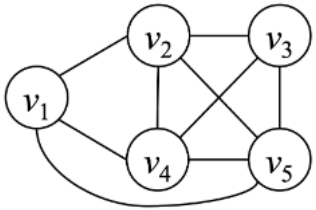
可平面图

- 直至G全部映射到平面上，则G为可平面图

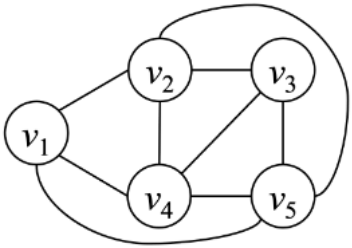
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```



G



H

可平面图

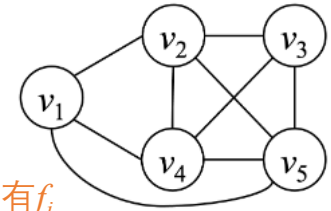
- 对于每个H片段B_i，仅当面f_j的边界含B_i的所有固定点时，B_i才有可能映射到平面上的f_j内

算法 10.1: DMP 算法

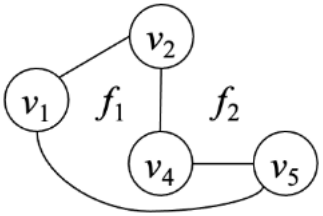
输入: 2 连通图 $G = \langle V, E \rangle$

```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```

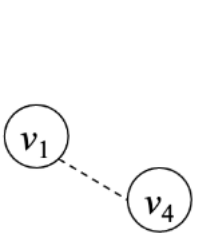
B_i.F：满足条件的所有f_j



G

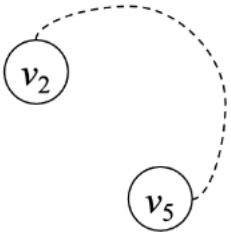


H



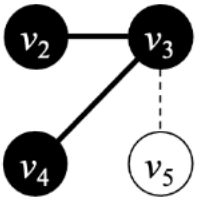
B_1

$B_1.\mathrm{F} \leftarrow \{f_1, f_2\}$



B_2

$B_2.\mathrm{F} \leftarrow \{f_1, f_2\}$



B_3

$B_3.\mathrm{F} \leftarrow \{f_1, f_2\}$

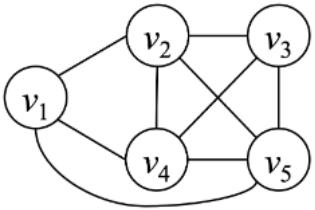
可平面图

- 若任意一个B_i的F属性值为空集，则G为不可平面图

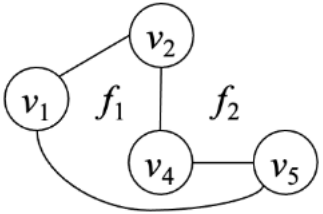
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

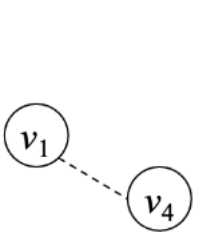
```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```



G

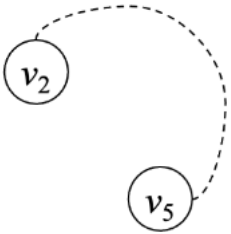


H



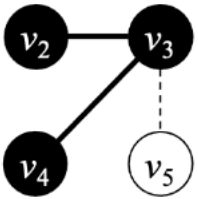
B_1

$B_1.\mathrm{F} \leftarrow \{f_1, f_2\}$



B_2

$B_2.\mathrm{F} \leftarrow \{f_1, f_2\}$



B_3

$B_3.\mathrm{F} \leftarrow \{f_1, f_2\}$

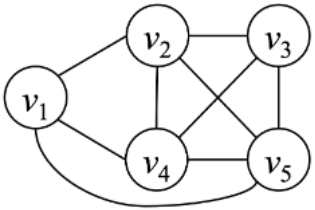
可平面图

- 否则，优先选择仅能映射到一个面内的H片段作为B

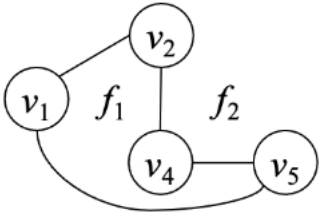
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

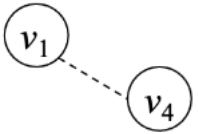
```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```



G

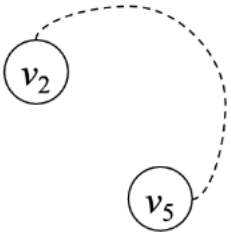


H



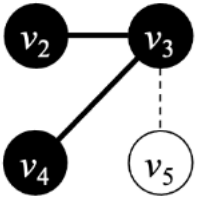
B_1

$B_1.\mathrm{F} \leftarrow \{f_1, f_2\}$



B_2

$B_2.\mathrm{F} \leftarrow \{f_1, f_2\}$



B_3

$B_3.\mathrm{F} \leftarrow \{f_1, f_2\}$

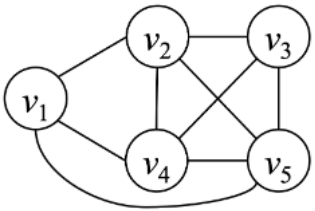
可平面图

- 否则，优先选择仅能映射到一个面内的H片段作为B

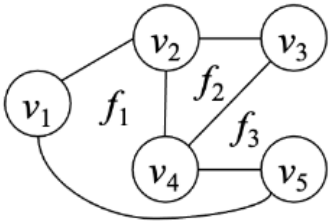
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

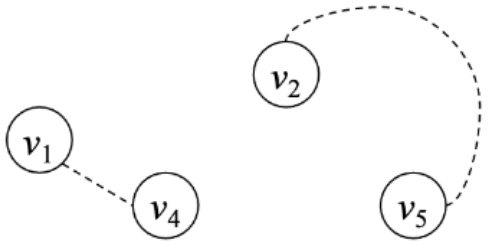
```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```



G



H



B_1 B_2 B_3

$B_1.\mathrm{F} \leftarrow \{f_1, f_3\}$ $B_2.\mathrm{F} \leftarrow \{f_1, f_3\}$ $B_3.\mathrm{F} \leftarrow \{f_3\}$

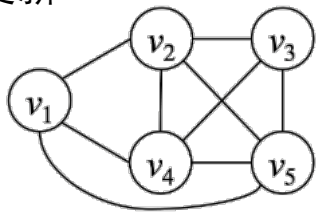
可平面图

- 从B中选择任意两个固定点间的一条路P
  从B的F属性值中选择任意一个面f
  将P映射到平面上的f内并更新H

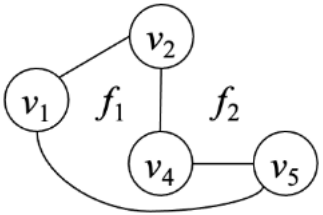
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

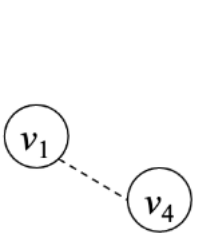
1 $H = \langle V_H, E_H \rangle \leftarrow G$ 中任意一个圈;
2 将 H 映射到平面上;
3 **while** $H \neq G$ **do**
4 **foreach** $B_i \in G$ 的所有 H 片段 **do**
5 $B_i.\mathrm{F} \leftarrow \emptyset$;
6 **foreach** $f_j \in H$ 的平面图的所有面 **do**
7 **if** f_j 的边界含 B_i 的所有固定点 **then**
8 $B_i.\mathrm{F} \leftarrow B_i.\mathrm{F} \cup \{f_j\}$;
9 **if** $B_i.\mathrm{F} = \emptyset$ **then**
10 输出 (G 是不可平面图);
11 **else if** $|B_i.\mathrm{F}| = 1$ **then**
12 $B \leftarrow B_i$;
13 **if** $B =$ null **then**
14 $B \leftarrow G$ 的任意一个 H 片段;
15 $P \leftarrow B$ 中任意两个固定点间的一条路;
16 $f \leftarrow B.\mathrm{F}$ 中任意一个面;
17 将 P 映射到平面上的 f 内;
18 $V_H \leftarrow V_H \cup P$经过的顶点的集合;
19 $E_H \leftarrow E_H \cup P$经过的边的集合;
20 输出 (G 是可平面图);



G

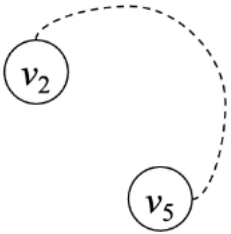


H



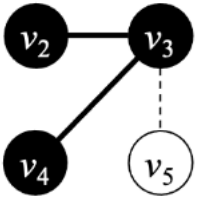
B_1

$B_1.\mathrm{F} \leftarrow \{f_1, f_2\}$



B_2

$B_2.\mathrm{F} \leftarrow \{f_1, f_2\}$

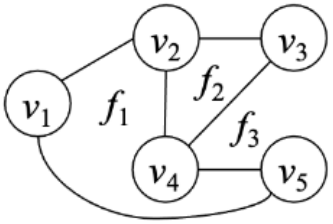


B_3

$B_3.\mathrm{F} \leftarrow \{f_1, f_2\}$

可平面图

- 从B中选择任意两个固定点间的一条路P
  从B的F属性值中选择任意一个面f
  将P映射到平面上的f内并更新H

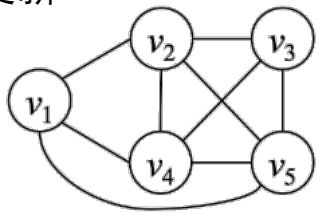


H

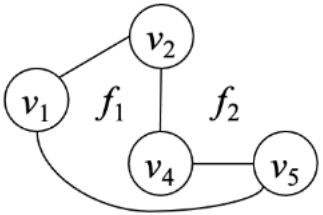
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

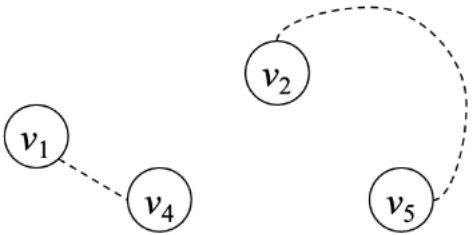
1 $H = \langle V_H, E_H \rangle \leftarrow G$ 中任意一个圈;
2 将 H 映射到平面上;
3 **while** $H \neq G$ **do**
4 **foreach** $B_i \in G$ 的所有 H 片段 **do**
5 $B_i.\mathrm{F} \leftarrow \emptyset$;
6 **foreach** $f_j \in H$ 的平面图的所有面 **do**
7 **if** f_j 的边界含 B_i 的所有固定点 **then**
8 $B_i.\mathrm{F} \leftarrow B_i.\mathrm{F} \cup \{f_j\}$;
9 **if** $B_i.\mathrm{F} = \emptyset$ **then**
10 输出 (G 是不可平面图);
11 **else if** $|B_i.\mathrm{F}| = 1$ **then**
12 $B \leftarrow B_i$;
13 **if** $B =$ null **then**
14 $B \leftarrow G$ 的任意一个 H 片段;
15 $P \leftarrow B$ 中任意两个固定点间的一条路;
16 $f \leftarrow B.\mathrm{F}$ 中任意一个面;
17 将 P 映射到平面上的 f 内;
18 $V_H \leftarrow V_H \cup P$经过的顶点的集合;
19 $E_H \leftarrow E_H \cup P$经过的边的集合;
20 输出 (G 是可平面图);



G



H

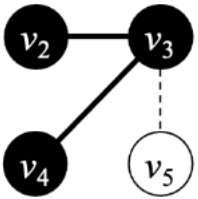


B_1

$B_1.\mathrm{F} \leftarrow \{f_1, f_2\}$

B_2

$B_2.\mathrm{F} \leftarrow \{f_1, f_2\}$



B_3

$B_3.\mathrm{F} \leftarrow \{f_1, f_2\}$

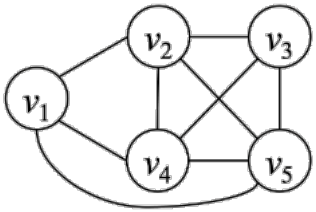
可平面图

- 算法开始

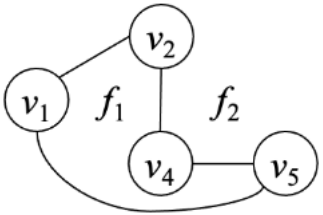
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```



G



H

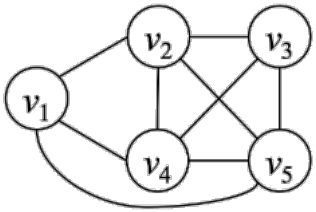
可平面图

- 第1轮while循环

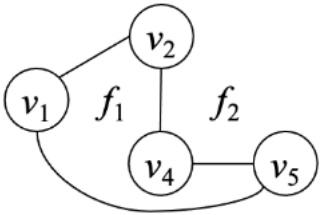
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

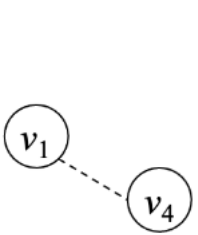
```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```



G

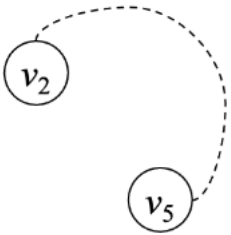


H



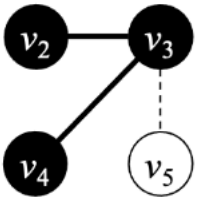
B_1

$B_1.\mathrm{F} \leftarrow \{f_1, f_2\}$



B_2

$B_2.\mathrm{F} \leftarrow \{f_1, f_2\}$



B_3

$B_3.\mathrm{F} \leftarrow \{f_1, f_2\}$

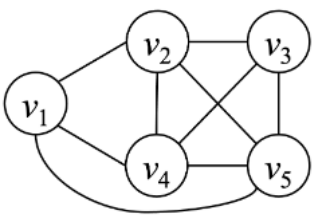
可平面图

- 第2轮while循环

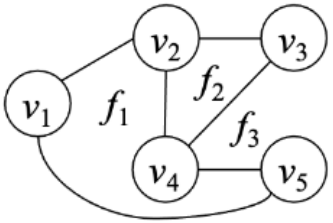
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

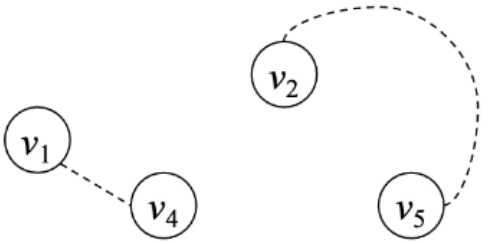
```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```



G



H



B_1 B_2 B_3

$B_1.\mathrm{F} \leftarrow \{f_1, f_3\}$ $B_2.\mathrm{F} \leftarrow \{f_1, f_3\}$ $B_3.\mathrm{F} \leftarrow \{f_3\}$

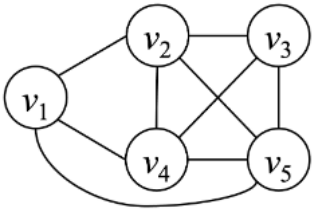
可平面图

- 第3轮while循环

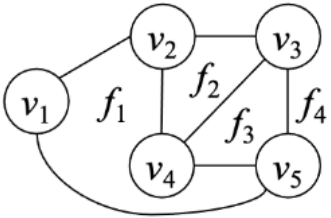
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

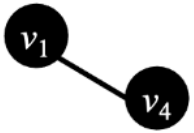
```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```



G

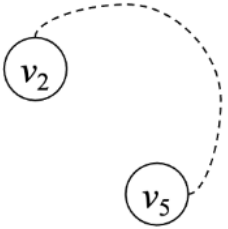


H



B_1

$B_1.\mathrm{F} \leftarrow \{f_1\}$



B_2

$B_2.\mathrm{F} \leftarrow \{f_1, f_4\}$

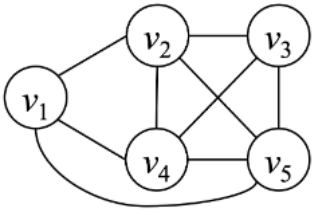
可平面图

- 第4轮while循环

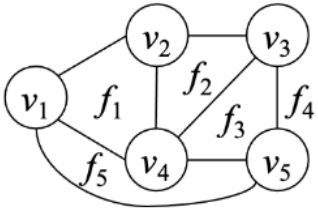
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

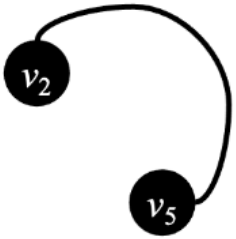
```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪ {f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```



G



H



B_1

$B_1.\mathrm{F} \leftarrow \{f_4\}$

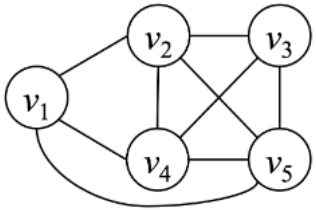
可平面图

- 第5轮while循环

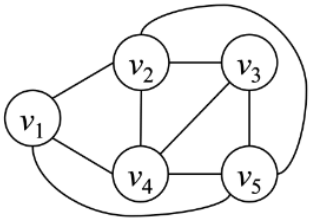
算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

1 $H = \langle V_H, E_H \rangle \leftarrow G$ 中任意一个圈;
2 将 H 映射到平面上;
3 **while** $H \neq G$ **do**
4 **foreach** $B_i \in G$ 的所有 H 片段 **do**
5 $B_i.\mathrm{F} \leftarrow \emptyset$;
6 **foreach** $f_j \in H$ 的平面图的所有面 **do**
7 **if** f_j 的边界含 B_i 的所有固定点 **then**
8 $B_i.\mathrm{F} \leftarrow B_i.\mathrm{F} \cup \{f_j\}$;
9 **if** $B_i.\mathrm{F} = \emptyset$ **then**
10 输出 (G 是不可平面图);
11 **else if** $|B_i.\mathrm{F}| = 1$ **then**
12 $B \leftarrow B_i$;
13 **if** $B =$ null **then**
14 $B \leftarrow G$ 的任意一个 H 片段;
15 $P \leftarrow B$ 中任意两个固定点间的一条路;
16 $f \leftarrow B.\mathrm{F}$ 中任意一个面;
17 将 P 映射到平面上的 f 内;
18 $V_H \leftarrow V_H \cup P$经过的顶点的集合;
19 $E_H \leftarrow E_H \cup P$经过的边的集合;
20 输出 (G 是可平面图);



G



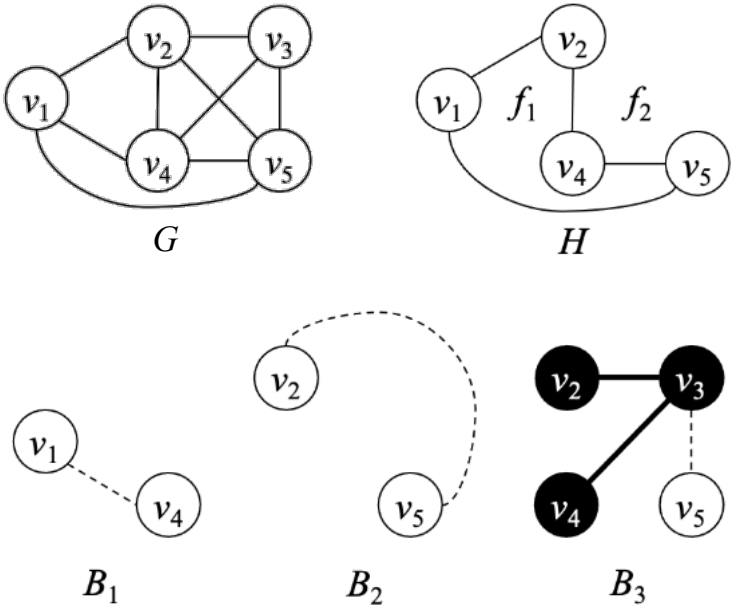
H

可平面图

- 对于非2连通图，如何利用DMP算法判定是否为可平面图?

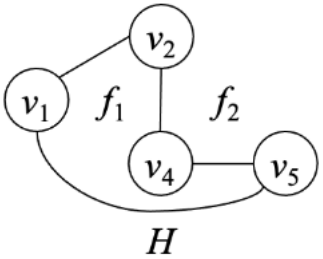
可平面图

- 对于非2连通图，如何利用DMP算法判定是否为可平面图？
- 为什么H片段一定有至少2个固定点？
  - 如果没有固定点？
  - 如果只有1个固定点？

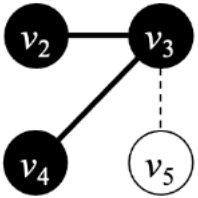


可平面图

- 对于非2连通图，如何利用DMP算法判定是否为可平面图?
- 为什么H片段一定有至少2个固定点?
- 为什么从B中只选择一条路映射到平面上?



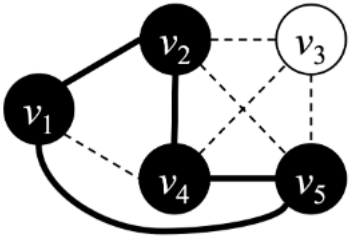
H



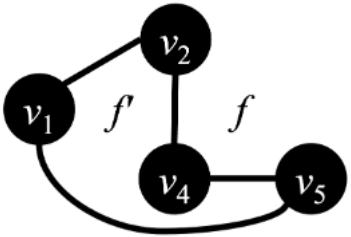
B_3

可平面图

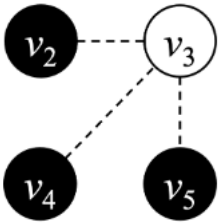
- 正确性：对于可平面图G，DMP算法每轮while循环条件判定前，子图H的平面图可以扩展为G的平面图



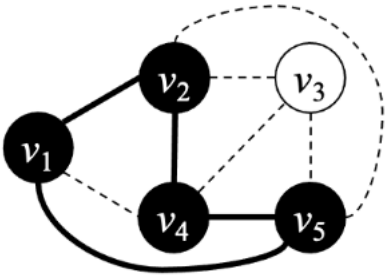
G



H



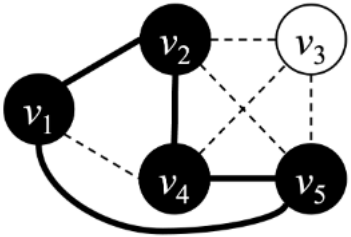
B



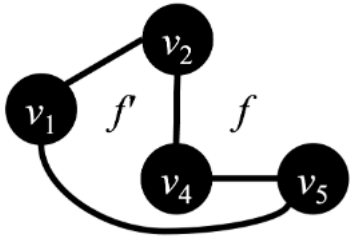
H_1

可平面图

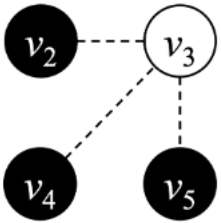
- 正确性：对于可平面图G，DMP算法每轮while循环条件判定前，子图H的平面图可以扩展为G的平面图



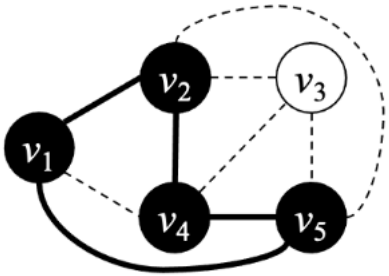
G



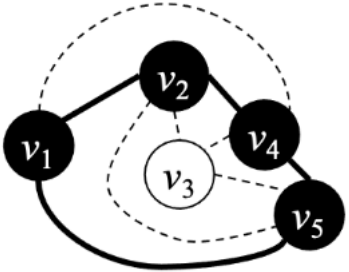
H



B



H_1



H_2

可平面图

- 时间复杂度：$O(n^2)$
  - while循环的轮数：$O(n)$
  - 每轮while循环的时间复杂度：$O(n)$

算法 10.1: DMP 算法

输入: 2 连通图 $G = \langle V, E \rangle$

```
1  H = ⟨V_H, E_H⟩ ← G 中任意一个圈;
2  将 H 映射到平面上;
3  while H ≠ G do
4      foreach B_i ∈ G 的所有 H 片段 do
5          B_i.F ← ∅;
6          foreach f_j ∈ H 的平面图的所有面 do
7              if f_j 的边界含 B_i 的所有固定点 then
8                  B_i.F ← B_i.F ∪{f_j};
9          if B_i.F = ∅ then
10             输出 (G 是不可平面图);
11         else if |B_i.F| = 1 then
12             B ← B_i;
13     if B = null then
14         B ← G 的任意一个 H 片段;
15     P ← B 中任意两个固定点间的一条路;
16     f ← B.F 中任意一个面;
17     将 P 映射到平面上的 f 内;
18     V_H ← V_H ∪ P经过的顶点的集合;
19     E_H ← E_H ∪ P经过的边的集合;
20 输出 (G 是可平面图);
```

本次课的主要内容

10.1 可平面图

10.2 面的染色

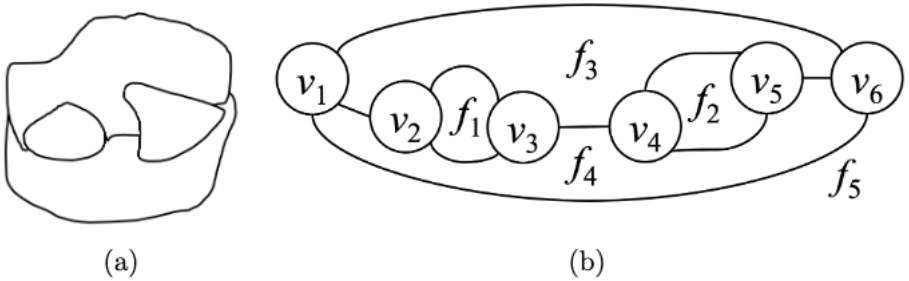
面的染色

- 地图染色问题

(a)

面的染色

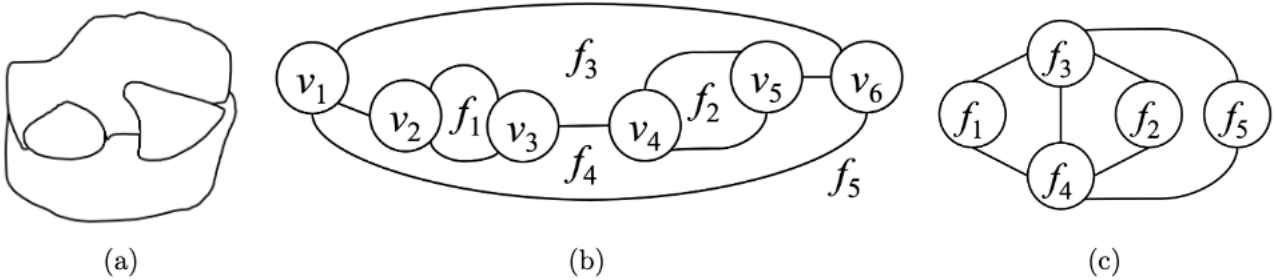
- 地图染色问题



(a) (b)

面的染色

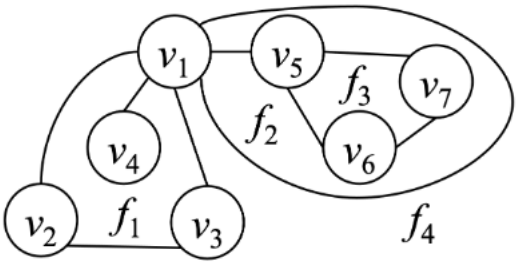
- 地图染色问题



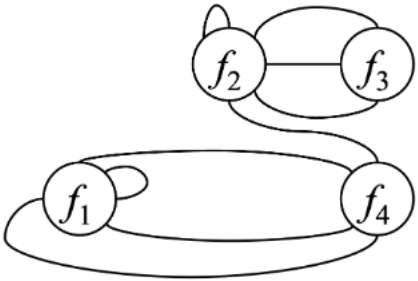
(a) (b) (c)

面的染色

- 可平面图$G = <V_G, E_G>$的平面图H的**对偶图**，记作H^*
  - 顶点集：H的所有面的集合
  - 对于每条边$e \in E_G$:
    若e在H的两个面f_i, f_j的边界中，则对应H^*中的一条边(f_i, f_j)
    若e只在H的一个面f_i的边界中，则对应H^*中的一条自环(f_i, f_i)



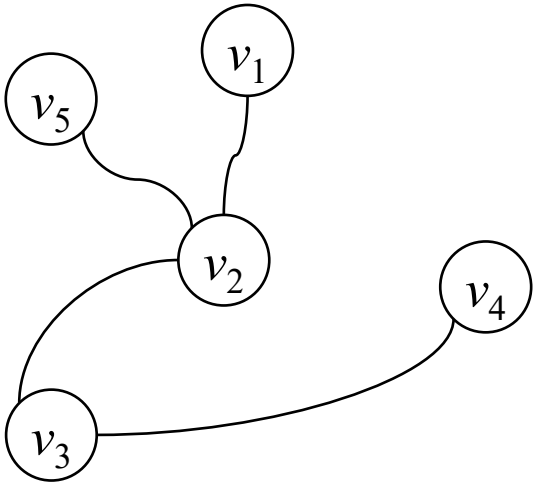
(a)



(b)

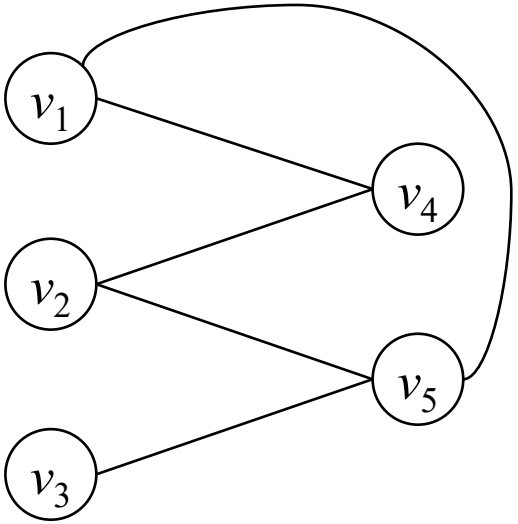
面的染色

- 树的平面图的对偶图有什么特征?



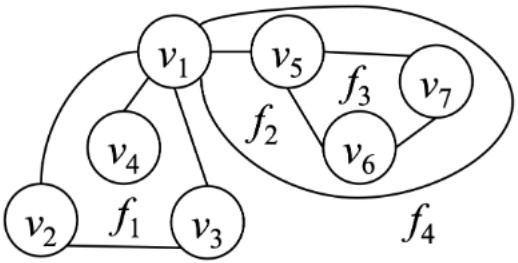
面的染色

- 树的平面图的对偶图有什么特征?
- 二分图的平面图的对偶图有什么特征?

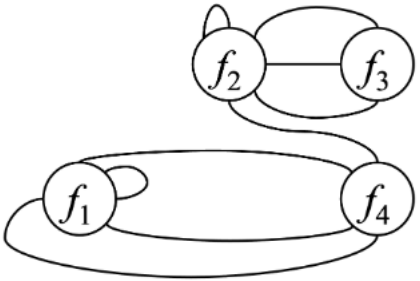


面的染色

- 对偶图连通吗?



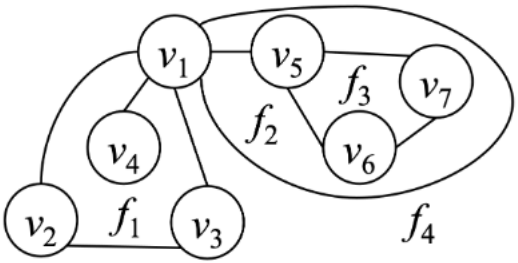
(a)



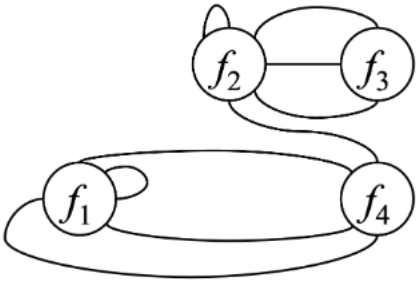
(b)

面的染色

- 对偶图连通吗?
- 自环对应的对偶图中的边有什么特征?



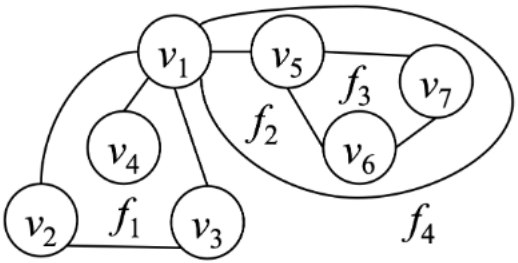
(a)



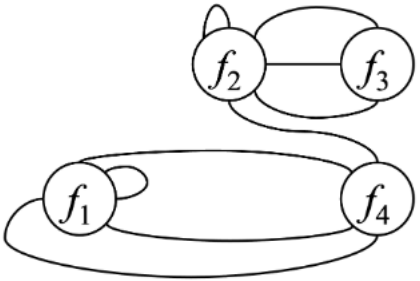
(b)

面的染色

- 对偶图连通吗?
- 自环对应的对偶图中的边有什么特征?
- 一个圈经过的所有边对应的对偶图中的边的集合有什么特征?



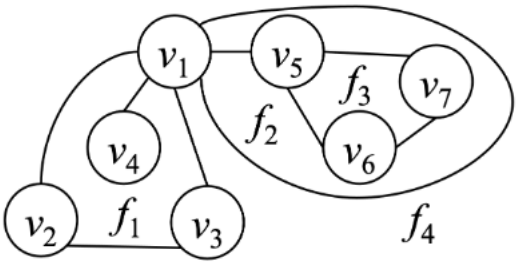
(a)



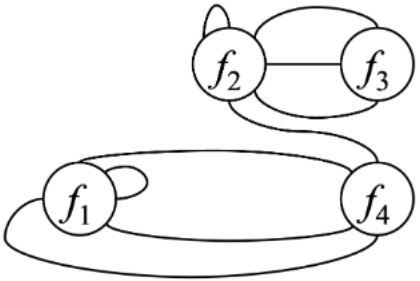
(b)

面的染色

- 对偶图是可平面图吗?



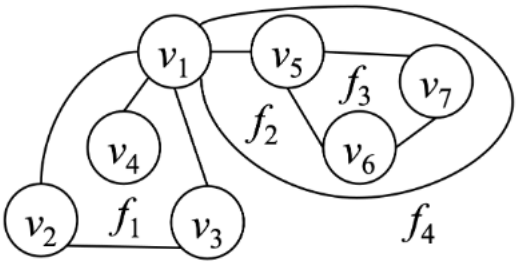
(a)



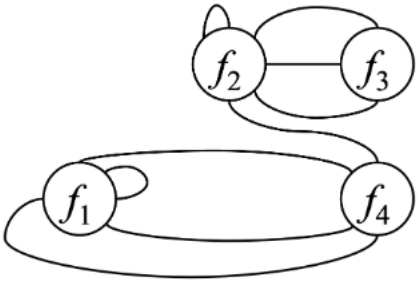
(b)

面的染色

- 对偶图是可平面图吗?
- 同一个平面图的不同对偶图同构吗?



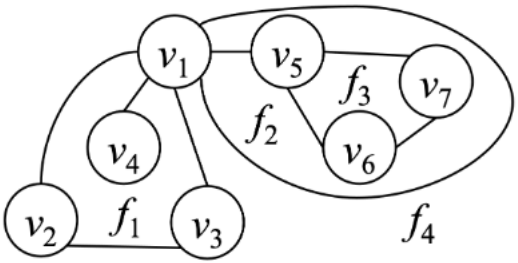
(a)



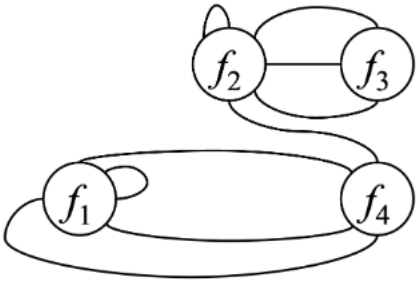
(b)

面的染色

- 对偶图是可平面图吗?
- 同一个平面图的不同对偶图同构吗?
- 同一个图的不同平面图的对偶图同构吗?



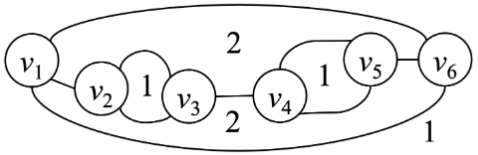
(a)



(b)

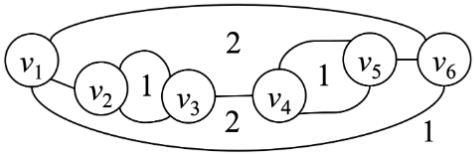
面的染色

- ***k*面染色**
  - 函数$fc : F \rightarrow \{1, \ldots, k\}$，值域代表$k$种色

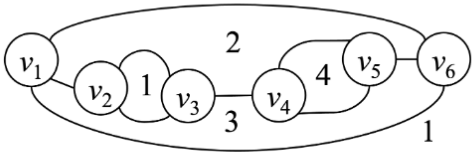


面的染色

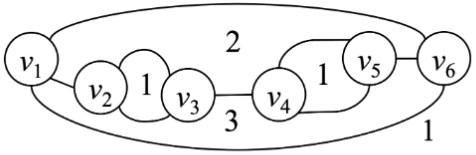
- k面染色
  - 函数$fc : F \rightarrow \{1, \ldots, k\}$，值域代表$k$种色
- 正常k面染色
  - 相邻（边界含公共边）的面的色都不同



(a)



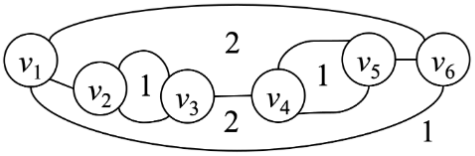
(b)



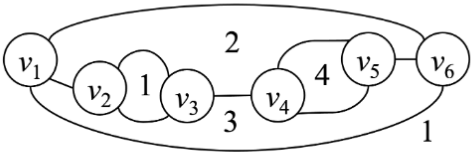
(c)

面的染色

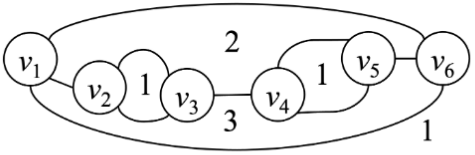
- ***k*面色可染**
  - 存在正常k面染色



(a)



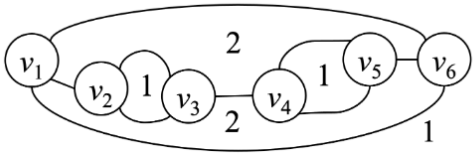
(b)



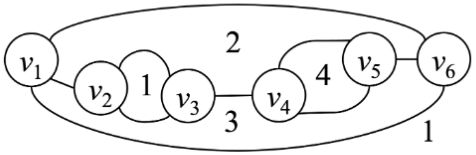
(c)

面的染色

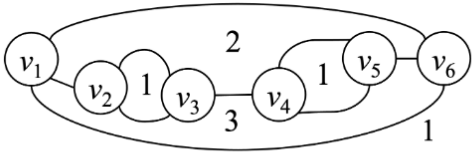
- k面色可染
  - 存在正常k面染色
- **面色数**
  - 使H是k面色可染的最小k值



(a)



(b)



(c)

面的染色

- 五色定理
  - 平面图是5面色可染
  - 对于任意一个简单可平面图G: $\chi(G) \leq 5$

面的染色

- Percy John Heawood，1861年出生于英国

https://en.wikipedia.org/wiki/File:Percy_J_Heawood.jpg

面的染色

- 对于任意一个简单可平面图G：$\chi(G) \leq 5$
  - 采用数学归纳法，对$v(G)$归纳

面的染色

- 对于任意一个简单可平面图G：$\chi(G) \leq 5$
  - 采用数学归纳法，对$v(G)$归纳
  - $v(G) \leq 5$时：$\chi(G) \leq 5$成立

面的染色

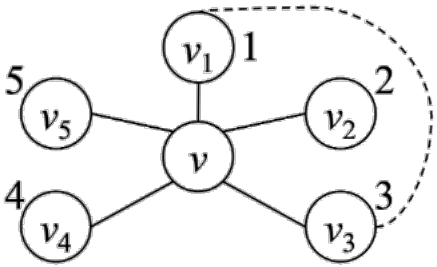
- 对于任意一个简单可平面图G：$\chi(G) \leq 5$
  - 采用数学归纳法，对$v(G)$归纳
  - $v(G) \leq 5$时：$\chi(G) \leq 5$成立
  - 假设$v(G) = k$时成立，则$v(G) = k + 1$时：
    存在顶点v满足$d(v) \leq 5$（作业），由归纳假设，$\chi(G - v) \leq 5$
    将$G - v$的正常$\chi(G - v)$点染色扩展为G的正常k'点染色且$k' \leq 5$

面的染色

- 对于任意一个简单可平面图G：$\chi(G) \leq 5$
  - 采用数学归纳法，对$v(G)$归纳
  - $v(G) \leq 5$时：$\chi(G) \leq 5$成立
  - 假设$v(G) = k$时成立，则$v(G) = k + 1$时：
    存在顶点v满足$d(v) \leq 5$（作业），由归纳假设，$\chi(G - v) \leq 5$
    将$G - v$的正常$\chi(G - v)$点染色扩展为G的正常k'点染色且$k' \leq 5$
  - 若$d(v) \leq 4$，或$d(v) = 5$且v有邻点的色相同，如何对v染色？

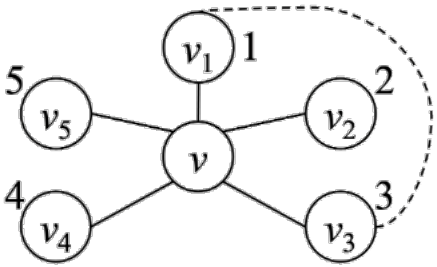
面的染色

- 对于任意一个简单可平面图G：$\chi(G) \leq 5$
  - 采用数学归纳法，对$v(G)$归纳
  - $v(G) \leq 5$时：$\chi(G) \leq 5$成立
  - 假设$v(G) = k$时成立，则$v(G) = k + 1$时：
    存在顶点v满足$d(v) \leq 5$（作业），由归纳假设，$\chi(G - v) \leq 5$
    将$G - v$的正常$\chi(G - v)$点染色扩展为G的正常k'点染色且$k' \leq 5$
  - 若$d(v) \leq 4$，或$d(v) = 5$且v有邻点的色相同，如何对v染色？
  - 若$d(v) = 5$且v的所有邻点的色互不相同，如何对v染色？



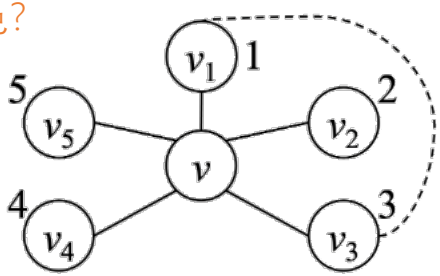
面的染色

- 对于任意一个简单可平面图G：$\chi(G) \leq 5$
  - 采用数学归纳法，对$v(G)$归纳
  - $v(G) \leq 5$时：$\chi(G) \leq 5$成立
  - 假设$v(G) = k$时成立，则$v(G) = k + 1$时：
    存在顶点v满足$d(v) \leq 5$（作业），由归纳假设，$\chi(G - v) \leq 5$
    将$G - v$的正常$\chi(G - v)$点染色扩展为G的正常k'点染色且$k' \leq 5$
  - 若$d(v) \leq 4$，或$d(v) = 5$且v有邻点的色相同，如何对v染色？
  - 若$d(v) = 5$且v的所有邻点的色互不相同：
    $G_{1,3}$：$G - v$中色为1或3的顶点子集的点导出子图
    若v_1和v_3在$G_{1,3}$的不同连通分支中，如何对v染色？



面的染色

- 对于任意一个简单可平面图G：$\chi(G) \le 5$
  - 采用数学归纳法，对$v(G)$归纳
  - $v(G) \le 5$时：$\chi(G) \le 5$成立
  - 假设$v(G) = k$时成立，则$v(G) = k + 1$时：
    存在顶点v满足$d(v) \le 5$（作业），由归纳假设，$\chi(G - v) \le 5$
    将$G - v$的正常$\chi(G - v)$点染色扩展为G的正常k'点染色且$k' \le 5$
  - 若$d(v) \le 4$，或$d(v) = 5$且v有邻点的色相同，如何对v染色？
  - 若$d(v) = 5$且v的所有邻点的色互不相同：
    $G_{1,3}$：$G - v$中色为1或3的顶点子集的点导出子图
    若v_1和v_3在$G_{1,3}$的不同连通分支中，如何对v染色？
    若v_1和v_3在$G_{1,3}$的同一个连通分支中，如何对v染色？

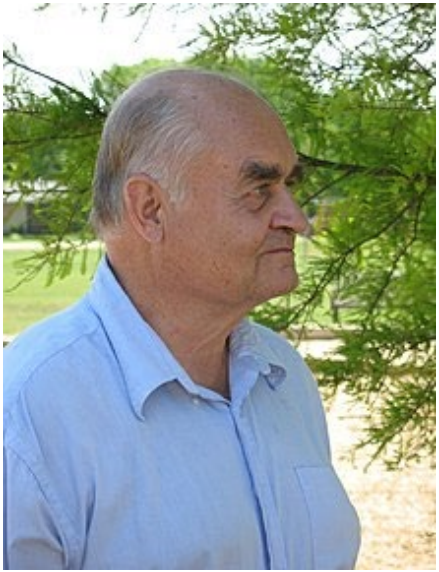


面的染色

- 四色定理
  - 平面图是4面色可染
  - 对于任意一个简单可平面图G: $\chi(G) \leq 4$

面的染色

- Kenneth Ira Appel，1936 年出生于美国
- Wolfgang Haken，1928年出生于德国



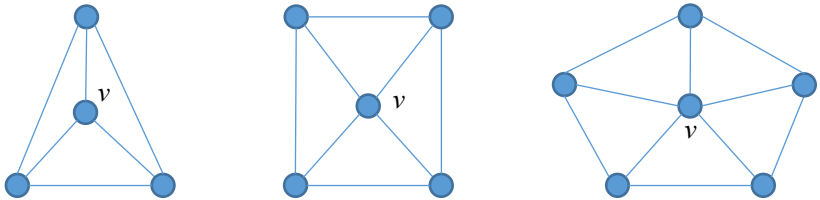
https://math.illinois.edu/sites/default/files/inline-images/ken-appel-150.jpg
https://upload.wikimedia.org/wikipedia/commons/thumb/1/1e/Wolfgang_Haken_2008.jpg/440px-Wolfgang_Haken_2008.jpg

面的染色

- 四色定理是计算机辅助证明的第一个重要定理

面的染色

- 四色定理是计算机辅助证明的第一个重要定理
  - 1879年
    英国数学家肯普（Alfred Kempe）给出了一个证明，
    尽管该证明存在错误，但逐步发展成为一种主要的证明思路：
    寻找由若干个可约构形组成的不可避免集，从而证明不存在（极小）反例
    - 极小反例：阶最小的反例
      不失一般性，假设其为极大可平面图（否则，增加边直至极大）
    - 构形：每个内部面的长度都为3的简单平面图
    - 可约构形：极小反例不可能含的构形（若含，则可被约减为一个更小反例）
    - 不可避免集：任何一个极小反例含其中至少一个构形



面的染色

- 四色定理是计算机辅助证明的第一个重要定理
  - 1879年
    英国数学家肯普（Alfred Kempe）给出了一个证明，
    尽管该证明存在错误，但逐步发展成为一种主要的证明思路：
    寻找由若干个可约构形组成的不可避免集，从而证明不存在（极小）反例
  - 1890年
    希伍德指出了肯普证明中的错误，并证明了五色定理

面的染色

- 四色定理是计算机辅助证明的第一个重要定理
  - 1879年
    英国数学家肯普（Alfred Kempe）给出了一个证明，
    尽管该证明存在错误，但逐步发展成为一种主要的证明思路：
    寻找由若干个可约构形组成的不可避免集，从而证明不存在（极小）反例
  - 1890年
    希伍德指出了肯普证明中的错误，并证明了五色定理
  - 20世纪60至70年代
    德国数学家希什（Heinrich Heesch）提出了一种构造不可避免集的方法，
    并设计了算法在计算机的辅助下验证构形的可约性
    然而，他没能得到足够的经费支持来完成这项计算任务

面的染色

- 四色定理是计算机辅助证明的第一个重要定理
  - 1879年
    英国数学家肯普（Alfred Kempe）给出了一个证明，
    尽管该证明存在错误，但逐步发展成为一种主要的证明思路：
    寻找由若干个可约构形组成的不可避免集，从而证明不存在（极小）反例
  - 1890年
    希伍德指出了肯普证明中的错误，并证明了五色定理
  - 20世纪60至70年代
    德国数学家希什（Heinrich Heesch）提出了一种构造不可避免集的方法，
    并设计了算法在计算机的辅助下验证构形的可约性
    然而，他没能得到足够的经费支持来完成这项计算任务
  - 1976年
    阿佩尔和哈肯宣布在计算机的辅助下找到了由1936个可约构形组成的不可避免集，从而证明了四色定理，相关论文于1977年发表
    1989年
    完成了错误修订

面的染色

- 四色定理是计算机辅助证明的第一个重要定理
  - 1996年
    罗伯逊（Neil Robertson）、桑德斯（Daniel Sanders）、西蒙（Paul Seymour）、托马斯（Robin Thomas）提出了一个真正的计算机可验证的证明，使用的633种构型的可约性和不可避免性都可由计算机检查

面的染色

- 四色定理是计算机辅助证明的第一个重要定理
  - 1996年
    罗伯逊（Neil Robertson）、桑德斯（Daniel Sanders）、西蒙（Paul Seymour）、托马斯（Robin Thomas）提出了一个真正的计算机可验证的证明，使用的633种构型的可约性和不可避免性都可由计算机检查
  - 2005年
    维尔纳（Benjamin Werner）和贡蒂埃（Georges Gonthier）用定理证明工具Coq形式化了这个证明，人们只需要相信Coq的正确性就足够了

书面作业

- 练习10.1、10.2、10.3、10.4