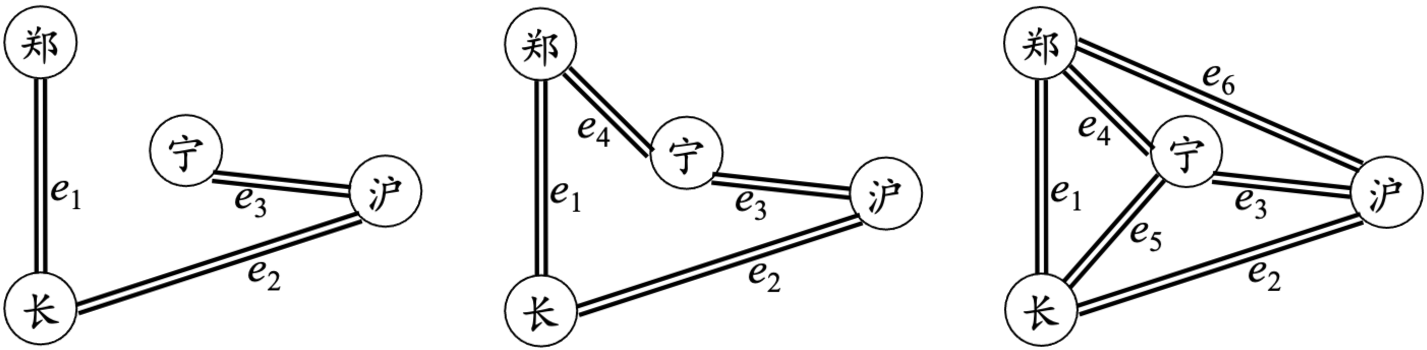
第4章 连通度

程龚

连通的强度



本次课的主要内容

4.1 块

4.2 割集和连通度

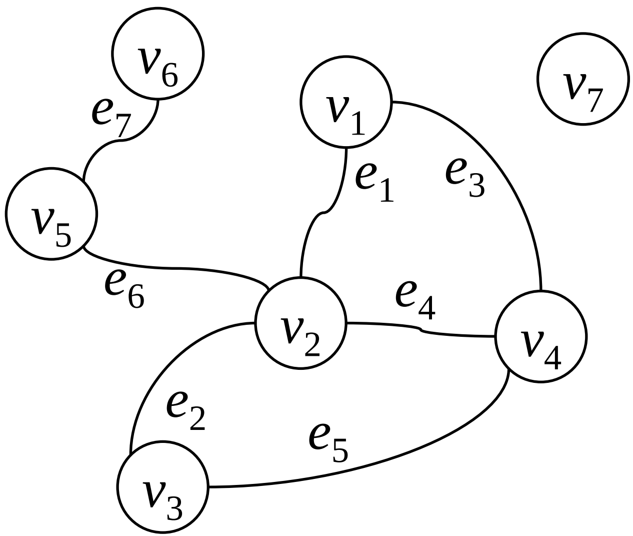
本次课的主要内容

4.1 块

4.2 割集和连通度

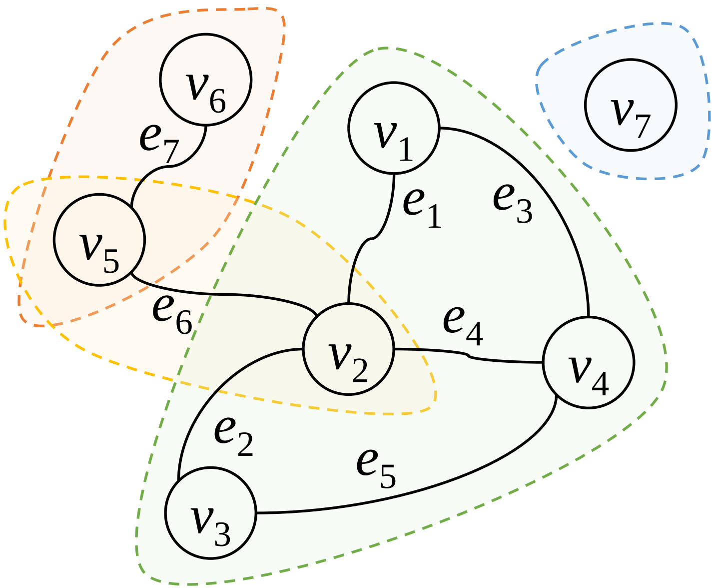
块

- **块**：极大的没有割点的连通子图



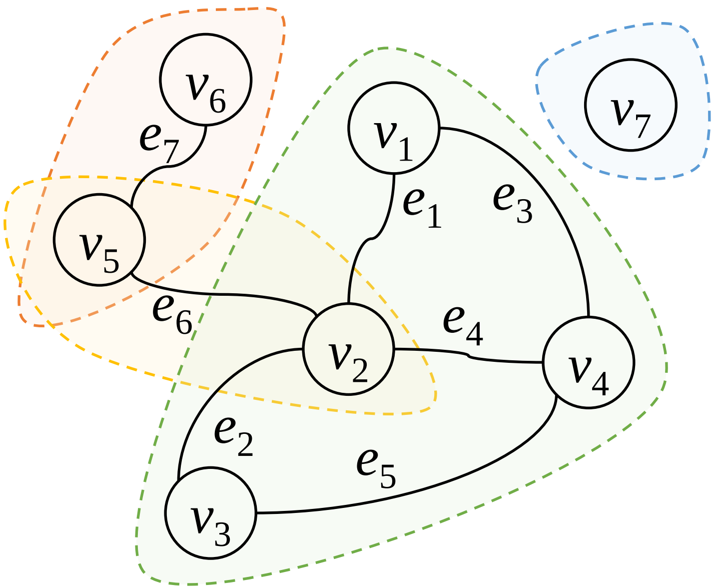
块

- **块**：极大的没有割点的连通子图



块

- **块**：极大的没有割点的连通子图
  - 若G只含1个块，即G连通且没有割点，则G自身称作一个块



块

- 完全图是块吗？

块

- 完全图是块吗?
- 树是块吗?

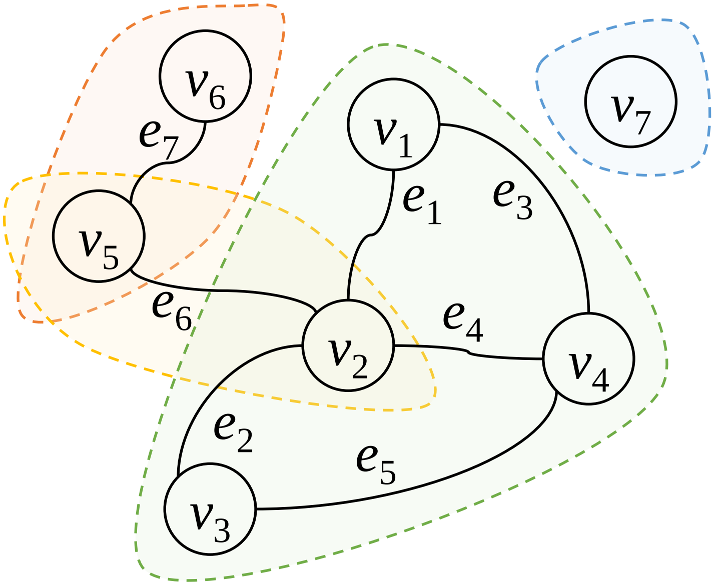
块

- 完全图是块吗?
- 树是块吗?
- 欧拉图和哈密尔顿图是块吗?

块

- 完全图是块吗？
- 树是块吗？
- 欧拉图和哈密尔顿图是块吗？

- 若图的块只含一个顶点，则这种顶点有什么特征？

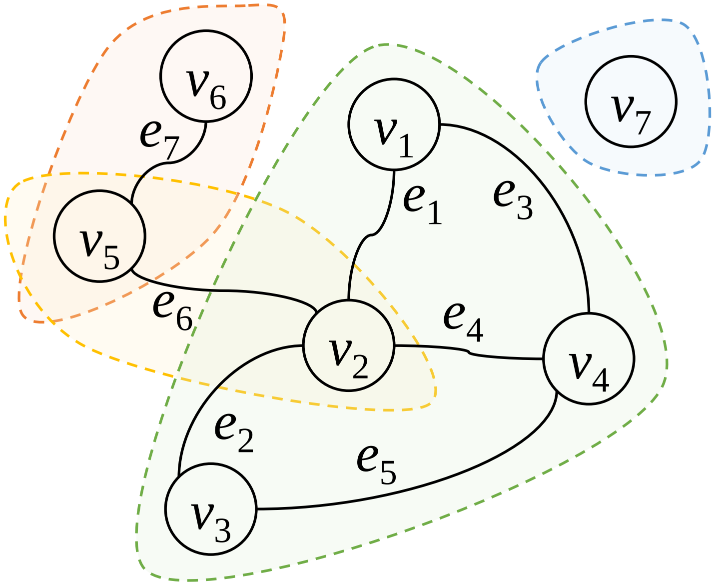


块

- 完全图是块吗？
- 树是块吗？
- 欧拉图和哈密尔顿图是块吗？

- 若图的块只含一个顶点，
  则这种顶点有什么特征？
- 若图的块只含一条边，
  则这种边有什么特征？ (iff)

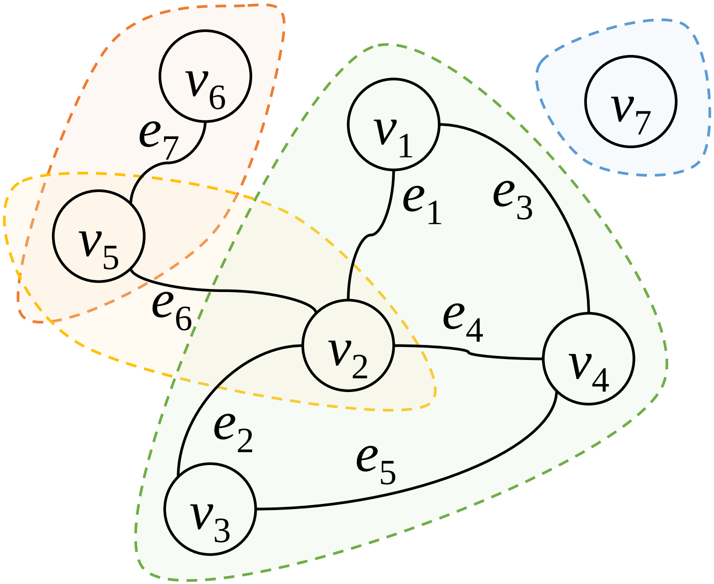
随堂小测



块

- 完全图是块吗？
- 树是块吗？
- 欧拉图和哈密尔顿图是块吗？

- 若图的块只含一个顶点，
  则这种顶点有什么特征？
- 若图的块只含一条边，
  则这种边有什么特征？ (iff)
  - 非割边为什么不行？
  - 割边为什么满足极大性？

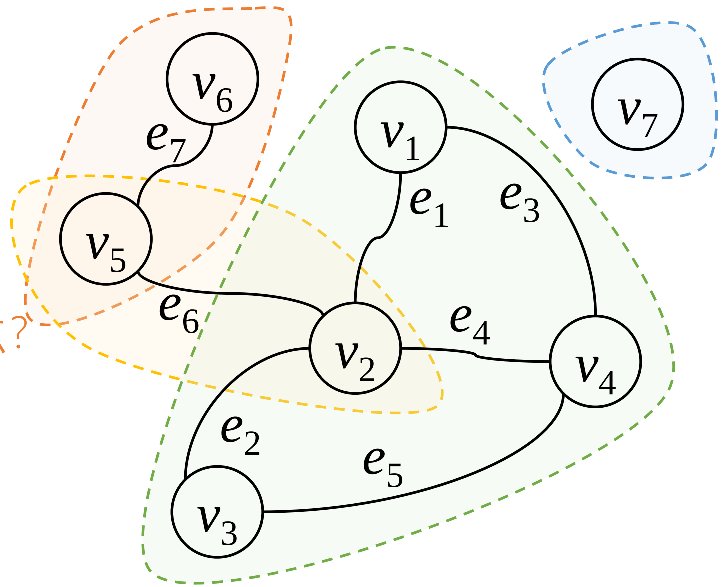


块

- 完全图是块吗？
- 树是块吗？
- 欧拉图和哈密尔顿图是块吗？

- 若图的块只含一个顶点，
  则这种顶点有什么特征？
- 若图的块只含一条边，
  则这种边有什么特征？ （iff）

- 两个块至多含多少个公共顶点？
  这种顶点有什么特征？

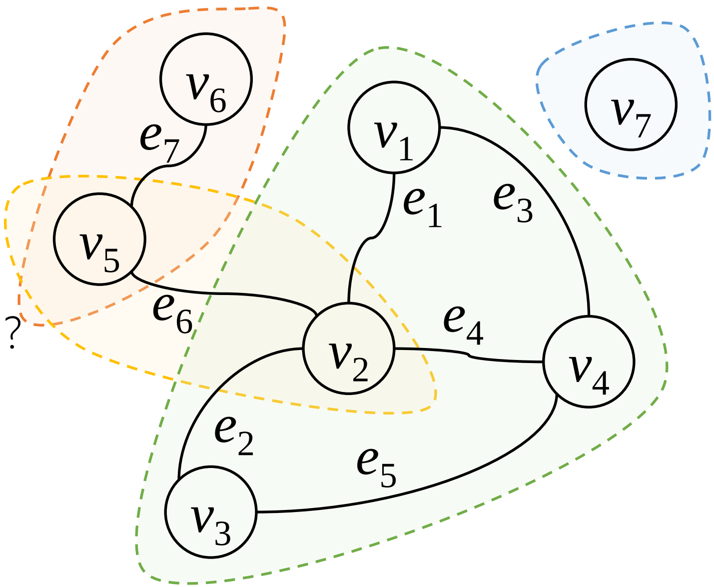


块

- 完全图是块吗?
- 树是块吗?
- 欧拉图和哈密尔顿图是块吗?

- 若图的块只含一个顶点,
  则这种顶点有什么特征?
- 若图的块只含一条边,
  则这种边有什么特征? (iff)

- 两个块至多含多少个公共顶点?
  这种顶点有什么特征?
- 两个块至多含多少条公共边?

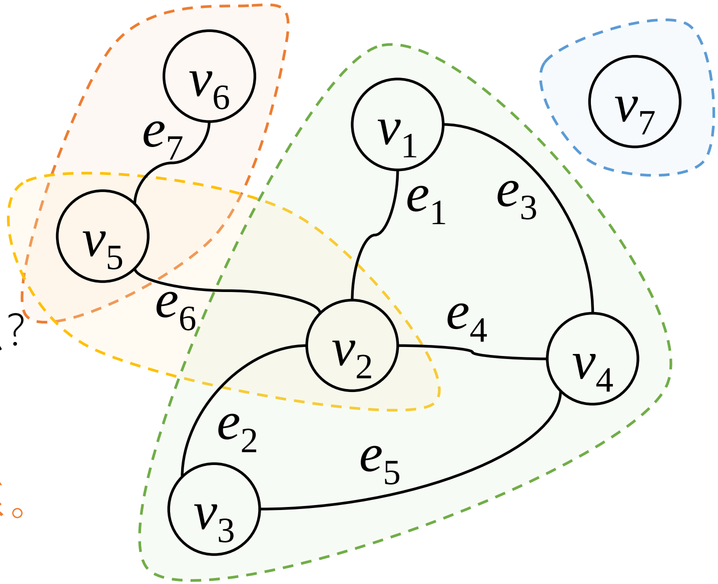


块

- 完全图是块吗?
- 树是块吗?
- 欧拉图和哈密尔顿图是块吗?

- 若图的块只含一个顶点,
  则这种顶点有什么特征?
- 若图的块只含一条边,
  则这种边有什么特征? (iff)

- 两个块至多含多少个公共顶点?
  这种顶点有什么特征?
- 两个块至多含多少条公共边?
- 块为边集定义了一种等价关系。
  - 划分是什么?



块

- 对于阶至少为3的连通图G，以下是块的等价定义
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。
  3. 对于任意两个顶点$u, v \in V$，G含圈经过u和v。
  4. 对于任意一个顶点$v \in V$和任意一条边$e \in E$，G含圈经过v和e 。
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e 。
  7. 对于任意三个顶点$u, v, w \in V$，G含u-v路经过w。
  8. 对于任意三个顶点$u, v, w \in V$，G含u-v路不经过w 。

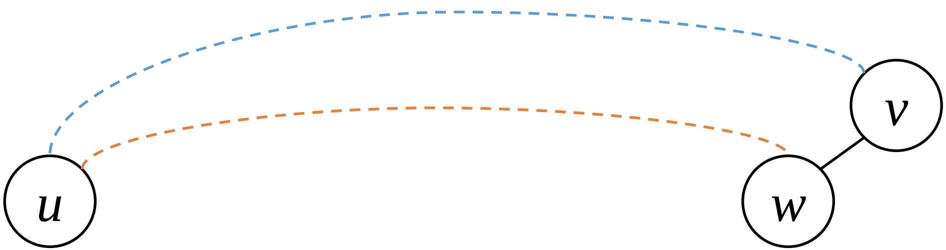
块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

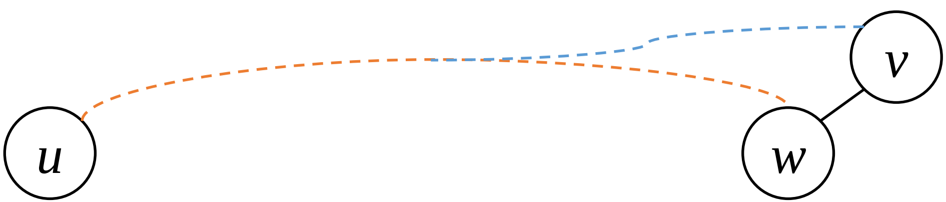
- 删去v的邻点w，仍存在其它的u-v路，与经过w的u-v路组成圈。这样证明存在什么问题？



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

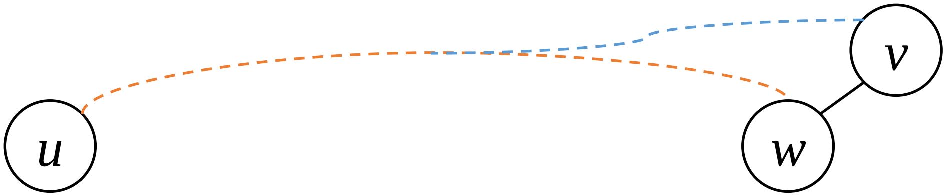
- 删去v的邻点w，仍存在其它的u-v路，与经过w的u-v路组成圈。这样证明存在什么问题？



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

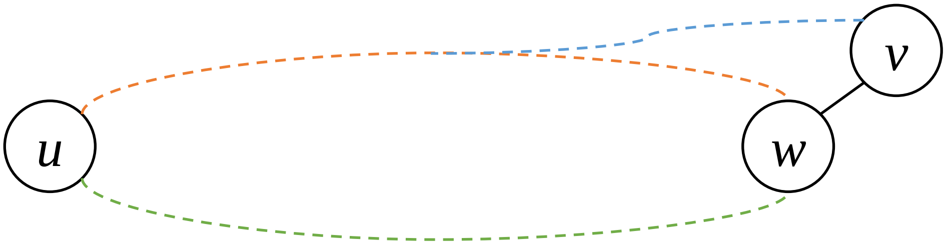
- 删去v的邻点w，仍存在其它的u-v路，与经过w的u-v路组成圈。这样证明存在什么问题？如何解决这个问题？



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

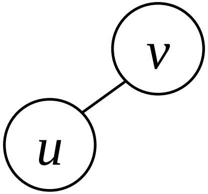
- 删去v的邻点w，仍存在其它的u-v路，与经过w的u-v路组成圈。这样证明存在什么问题？如何解决这个问题？



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

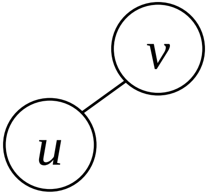
- 采用数学归纳法，对两个顶点间的距离归纳。
  - 当$dist(u, v) = 1$时，为什么成立？



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

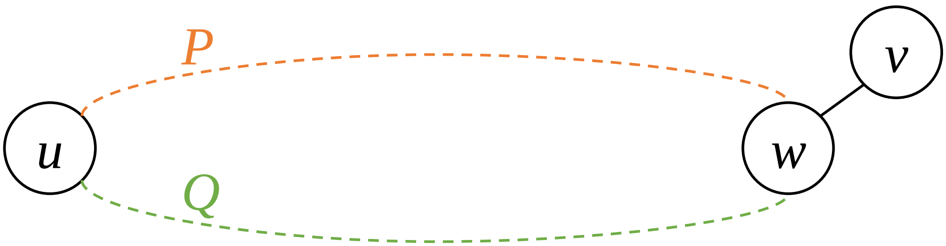
- 采用数学归纳法，对两个顶点间的距离归纳。
  - 当$dist(u, v) = 1$时，为什么成立？
    - G没有割点 → 没有割边 → u和v共圈



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

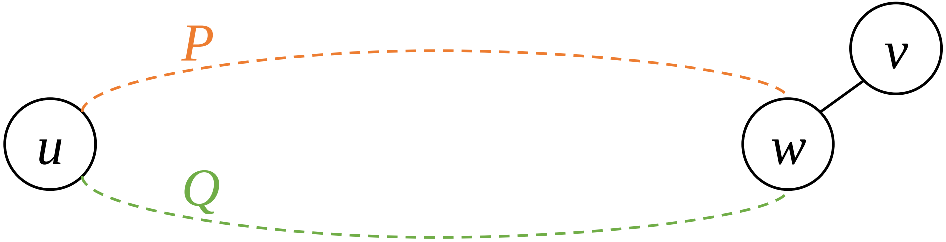
- 采用数学归纳法，对两个顶点间的距离归纳。
  - 当$dist(u, v) = 1$时，为什么成立？
    - G没有割点 → 没有割边 → u和v共圈
  - 假设$dist(u, v) = k$时成立，则$dist(u, v) = k + 1$时
    - $dist(u, w) = k$ → 共圈



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

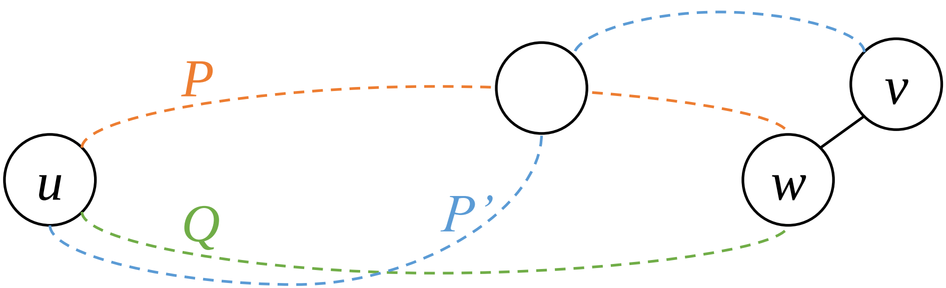
- 采用数学归纳法，对两个顶点间的距离归纳。
  - 当$dist(u, v) = 1$时，为什么成立？
    - G没有割点 → 没有割边 → u和v共圈
  - 假设$dist(u, v) = k$时成立，则$dist(u, v) = k + 1$时
    - $dist(u, w) = k$ → 共圈
    - 若这个圈经过v？



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

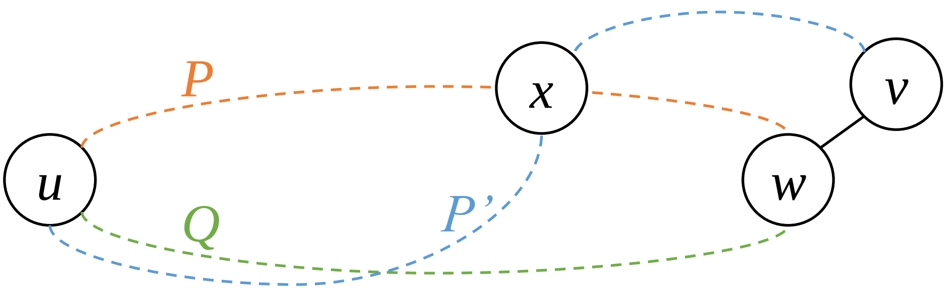
- 采用数学归纳法，对两个顶点间的距离归纳。
  - 当$dist(u, v) = 1$时，为什么成立？
    - G没有割点 → 没有割边 → u和v共圈
  - 假设$dist(u, v) = k$时成立，则$dist(u, v) = k + 1$时
    - $dist(u, w) = k$ → 共圈
    - 不经过w的u-v路与上述圈有公共顶点怎么办？



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

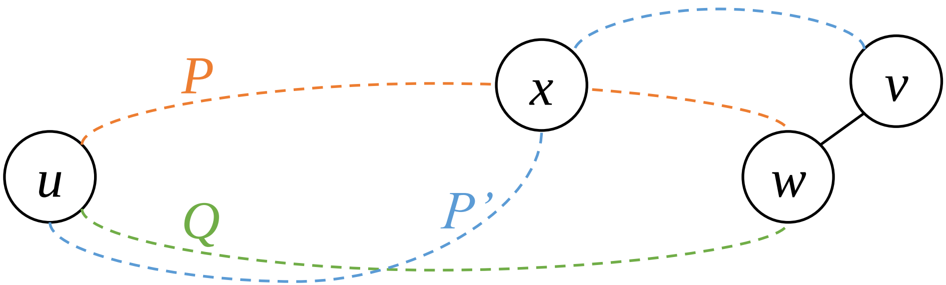
- 采用数学归纳法，对两个顶点间的距离归纳。
  - 当$dist(u, v) = 1$时，为什么成立？
    - G没有割点 → 没有割边 → u和v共圈
  - 假设$dist(u, v) = k$时成立，则$dist(u, v) = k + 1$时
    - $dist(u, w) = k$ → 共圈
    - x：不经过w的u-v路与上述圈的最后一个公共顶点



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

- 采用数学归纳法，对两个顶点间的距离归纳。
  - 当$dist(u, v) = 1$时，为什么成立？
    - G没有割点 → 没有割边 → u和v共圈
  - 假设$dist(u, v) = k$时成立，则$dist(u, v) = k + 1$时
    - $dist(u, w) = k$ → 共圈
    - x：不经过w的u-v路与上述圈的最后一个公共顶点
    - 两条无公共内顶点的u-v路：P中的u-x路 拼接 P'中的x-v路，Q 拼接 (w, v)



块

- 对于阶至少为3的连通图G
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。

- 你能自己证明吗？

块

- 对于阶至少为3的连通图G
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。
  3. 对于任意两个顶点$u, v \in V$，G含圈经过u和v。

- 你能自己证明吗?

块

- 对于阶至少为3的连通图G
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。
  3. 对于任意两个顶点$u, v \in V$，G含圈经过u和v。

- 你能自己证明吗？

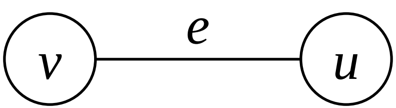
块

- 对于阶至少为3的连通图G
  3. 对于任意两个顶点$u, v \in V$，G含圈经过u和v。
  4. 对于任意一个顶点$v \in V$和任意一条边$e \in E$，G含圈经过v和e 。

块

- 对于阶至少为3的连通图G
  3. 对于任意两个顶点$u, v \in V$，G含圈经过u和v。
  4. 对于任意一个顶点$v \in V$和任意一条边$e \in E$，G含圈经过v和e 。

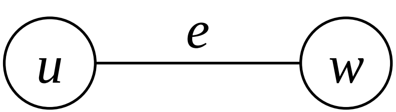
- 若v是e的端点？

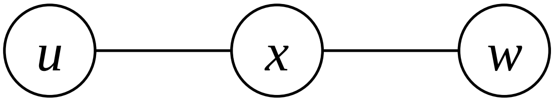


块

- 对于阶至少为3的连通图G
  3. 对于任意两个顶点$u, v \in V$，G含圈经过u和v。
  4. 对于任意一个顶点$v \in V$和任意一条边$e \in E$，G含圈经过v和e 。

- 若v是e的端点？
- 若v不是e的端点
  - 对e剖分，得到G' → G'也是块 → v和x共圈 → v和e共圈





块

- 对于阶至少为3的连通图G
  3. 对于任意两个顶点$u, v \in V$，G含圈经过u和v。
  4. 对于任意一个顶点$v \in V$和任意一条边$e \in E$，G含圈经过v和e 。

- 你能自己证明吗?

块

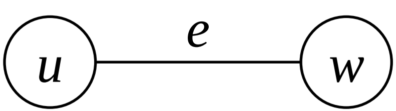
- 对于阶至少为3的连通图G
  4. 对于任意一个顶点$v \in V$和任意一条边$e \in E$，G含圈经过v和e 。
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。

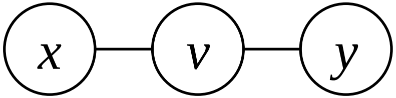
- 你能自己证明吗?

块

- 对于阶至少为3的连通图G
  4. 对于任意一个顶点$v \in V$和任意一条边$e \in E$，G含圈经过v和e 。
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。

- 你能自己证明吗？
  - 对f剖分





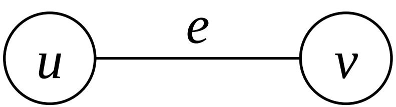
块

- 对于阶至少为3的连通图G
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e。

块

- 对于阶至少为3的连通图G
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e 。

- 若u和v是e的端点？



块

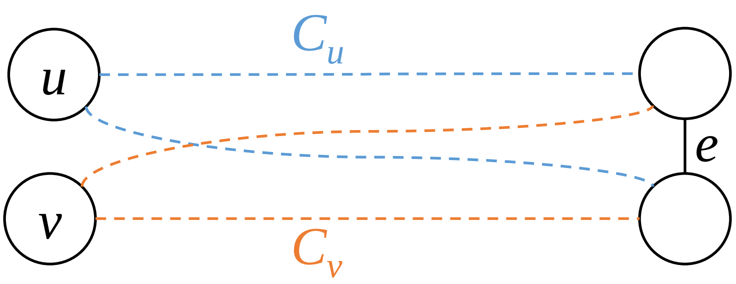
- 对于阶至少为3的连通图G
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e。

- 若u和v是e的端点？
- 若u是e的端点而v不是？

块

- 对于阶至少为3的连通图G
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e 。

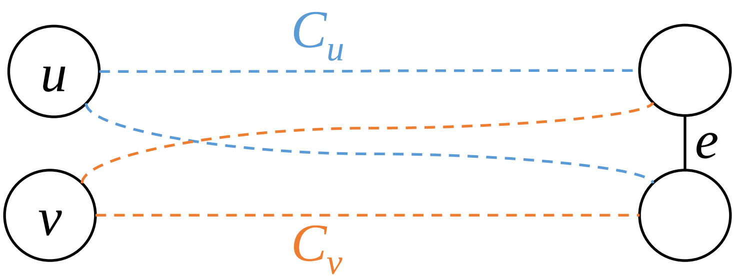
- 若u和v是e的端点？
- 若u是e的端点而v不是？
- 若u和v不是e的端点



块

- 对于阶至少为3的连通图G
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e。

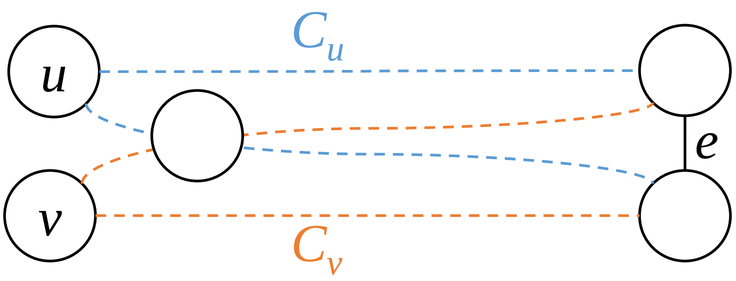
- 若u和v是e的端点？
- 若u是e的端点而v不是？
- 若u和v不是e的端点
  - 若C_u经过v或C_v经过u？



块

- 对于阶至少为3的连通图G
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e。

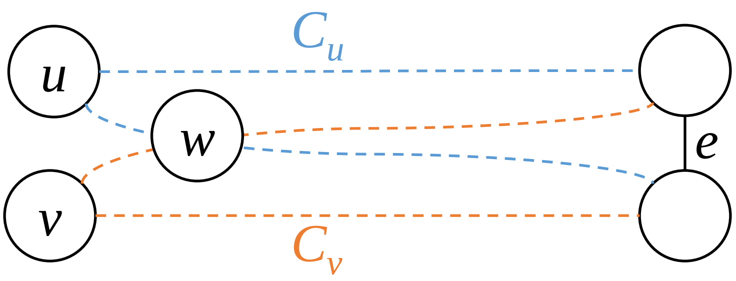
- 若u和v是e的端点？
- 若u是e的端点而v不是？
- 若u和v不是e的端点
  - C_u和C_v有公共顶点怎么办？



块

- 对于阶至少为3的连通图G
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e。

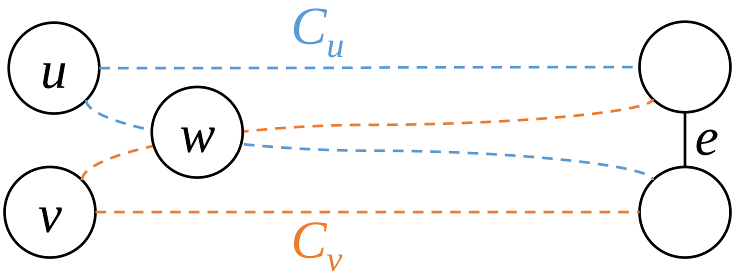
- 若u和v是e的端点？
- 若u是e的端点而v不是？
- 若u和v不是e的端点
  - w：C_u和C_v的公共顶点中，距离u最近的一个



块

- 对于阶至少为3的连通图G
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e。

- 若u和v是e的端点？
- 若u是e的端点而v不是？
- 若u和v不是e的端点
  - w：C_u和C_v的公共顶点中，距离u最近的一个
  - 经过e的u-v路：C_u中内顶点不被C_v经过的u-w路 拼接 C_v中经过e的w-v路



块

- 对于阶至少为3的连通图G
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e。
  7. 对于任意三个顶点$u, v, w \in V$，G含u-v路经过w。

- 你能自己证明吗?

块

- 对于阶至少为3的连通图G
  7. 对于任意三个顶点$u, v, w \in V$，G含u-v路经过w。
  8. 对于任意三个顶点$u, v, w \in V$，G含u-v路不经过w 。

- 你能自己证明吗?

块

- 对于阶至少为3的连通图G
  8. 对于任意三个顶点$u, v, w \in V$，G含u-v路不经过w。
  1. 图G是块。

- 你能自己证明吗?

块

- 对于阶至少为3的连通图G，以下是块的等价定义
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。
  3. 对于任意两个顶点$u, v \in V$，G含圈经过u和v。
  4. 对于任意一个顶点$v \in V$和任意一条边$e \in E$，G含圈经过v和e 。
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e 。
  7. 对于任意三个顶点$u, v, w \in V$，G含u-v路经过w。
  8. 对于任意三个顶点$u, v, w \in V$，G含u-v路不经过w 。

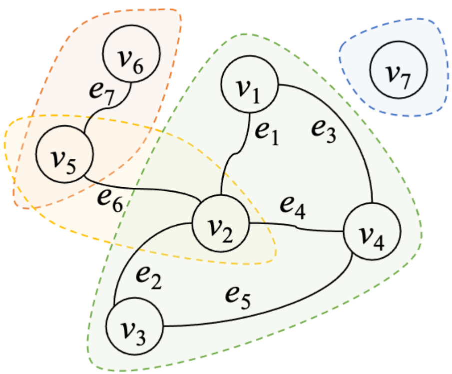
块

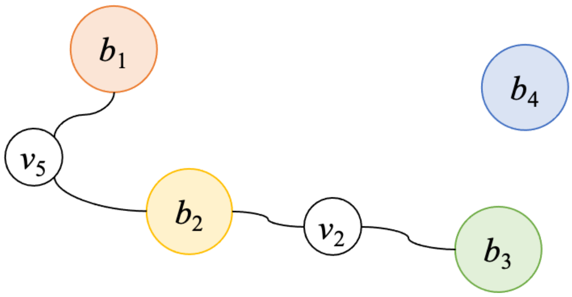
- 对于阶至少为3的连通图G，以下是块的等价定义
  1. G没有割点。
  2. 对于任意两个顶点$u, v \in V$，G含两条无公共内顶点的u-v路。
  3. 对于任意两个顶点$u, v \in V$，G含圈经过u和v。
  4. 对于任意一个顶点$v \in V$和任意一条边$e \in E$，G含圈经过v和e 。
  5. 对于任意两条边$e, f \in E$，G含圈经过e和f。
  6. 对于任意两个顶点$u, v \in V$和任意一条边$e \in E$，G含u-v路经过e 。
  7. 对于任意三个顶点$u, v, w \in V$，G含u-v路经过w。
  8. 对于任意三个顶点$u, v, w \in V$，G含u-v路不经过w 。

- 块为边集定义了一种等价关系，这种等价关系的内涵是什么？

块

- **块-割点图**
  - 二分图

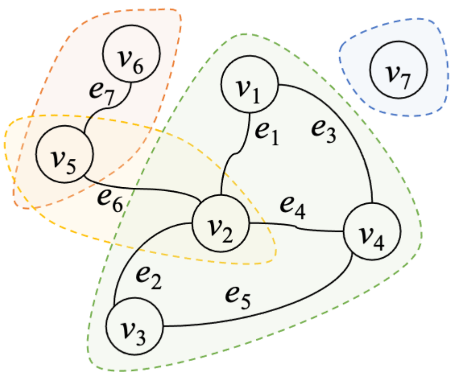


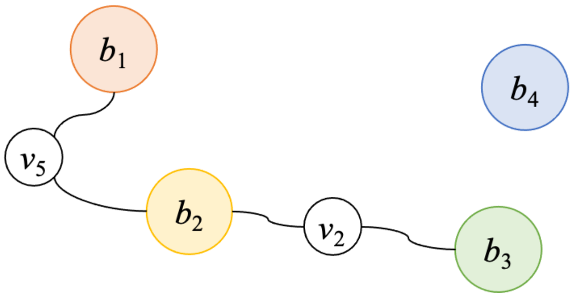


块

- 块-割点图
  - 二分图

- 块-割点图含圈吗？

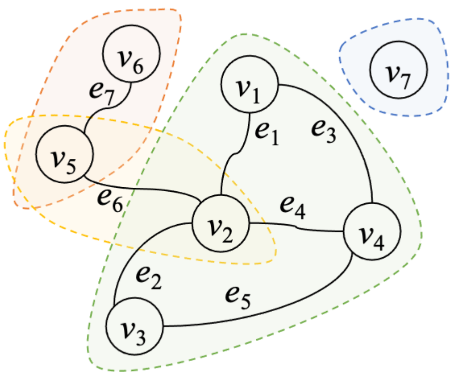


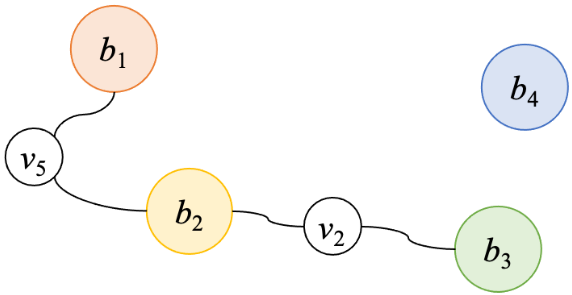


块

- 块-割点图
  - 二分图

- 块-割点图含圈吗？
- 对于图G的块-割点图H，H的叶顶点有可能是G的割点吗？

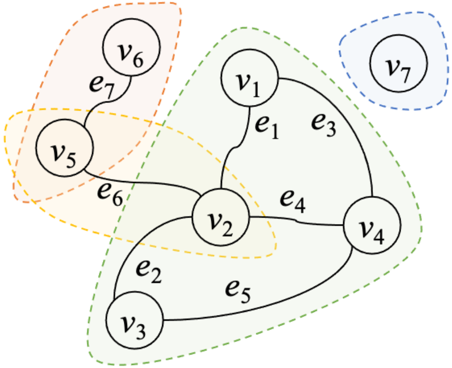


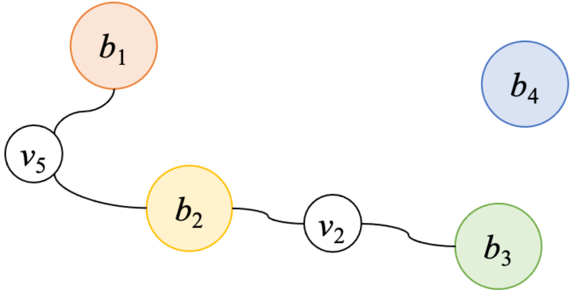


块

- 块-割点图
  - 二分图

- 块-割点图含圈吗？
- 对于图G的块-割点图H，H的叶顶点有可能是G的割点吗？
- 对于图$G = <V, E>$和顶点$v \in V$，v是G的割点当且仅当G的至少2个块含v。



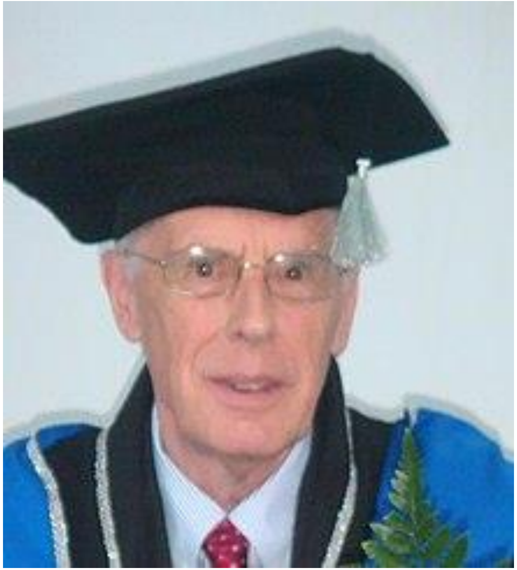


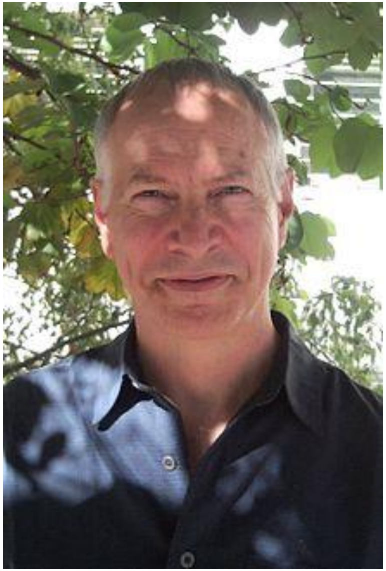
块

- 如何找出图中所有块?

块

- John Hopcroft，1939年出生于美国，1986年获图灵奖
- Robert Tarjan，1948年出生于美国，1986年获图灵奖





https://en.wikipedia.org/wiki/John_Hopcroft
https://en.wikipedia.org/wiki/Robert_Tarjan

块

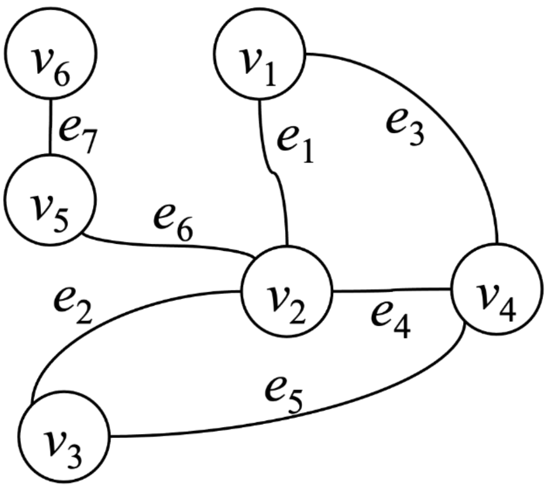
- 修改DFSCV算法：将发现的每条边入栈

算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
  if v.visited = false **then**
    入栈（S, (u, v)）;
    v.parent $\leftarrow u$;
    u.children $\leftarrow u$.children + 1;
    DFSBlk(G, v);
    u.low $\leftarrow \min\{u$.low, v.low$\}$;
    if (u.parent = null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
      输出（以下边组成一个块）;
      do
        $(x, y) \leftarrow$ 出栈（S）;
        输出（(x, y)）;
      while $(x, y) \neq (u, v)$;
  else if $v \neq u$.parent **then**
    if u.d $> v$.d **then**
      入栈（S, (u, v)）;
    u.low $\leftarrow \min\{u$.low, v.d$\}$;



块

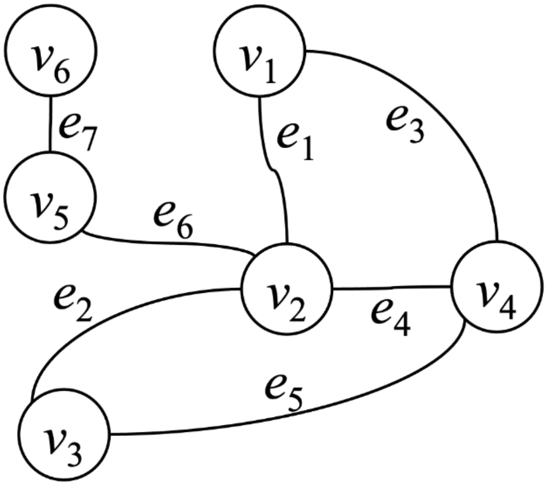
- 修改DFSCV算法：割点判定改为出栈一个块的边集

算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E\rangle$，顶点 u

初值: *time* 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow$ $time$;
u.low $\leftarrow$ u.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
    if v.visited = false **then**
        入栈（S, (u, v)）;
        v.parent $\leftarrow$ u;
        u.children $\leftarrow$ u.children + 1;
        `DFSBlk`(G, v);
        u.low $\leftarrow$ $\min\{u$.low, v.low$\}$;
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq$ u.d) **then**
            输出（以下边组成一个块）;
            do
                $(x, y) \leftarrow$ 出栈（S）;
                输出（(x, y)）;
            while $(x, y) \neq (u, v)$;
    else if $v \neq u$.parent **then**
        if u.d $>$ v.d **then**
            入栈（S, (u, v)）;
        u.low $\leftarrow$ $\min\{u$.low, v.d$\}$;



块

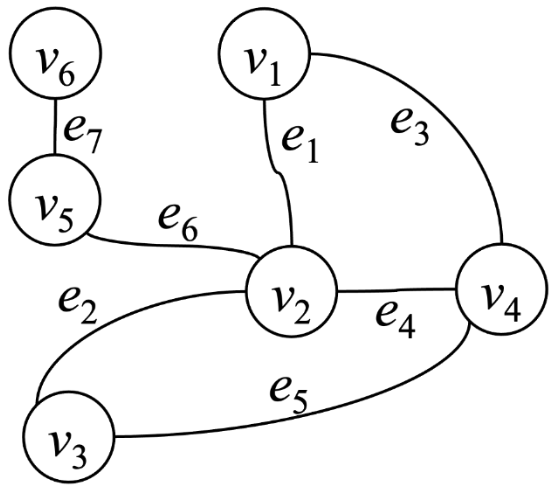
- 修改DFSCV算法：若结束时栈非空，出栈这个块的边集

算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
 if v.visited = false **then**
  入栈（S, (u, v)）;
  v.parent $\leftarrow u$;
  u.children $\leftarrow u$.children + 1;
  DFSBlk(G, v);
  u.low $\leftarrow$ min{u.low, v.low};
  if (u.parent = null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
   输出（以下边组成一个块）;
   do
    $(x, y) \leftarrow$ 出栈（S）;
    输出（(x, y)）;
   while $(x, y) \neq (u, v)$;
 else if $v \neq u$.parent **then**
  if u.d > v.d **then**
   入栈（S, (u, v)）;
  u.low $\leftarrow$ min{u.low, v.d};



块

- DFSBlk算法

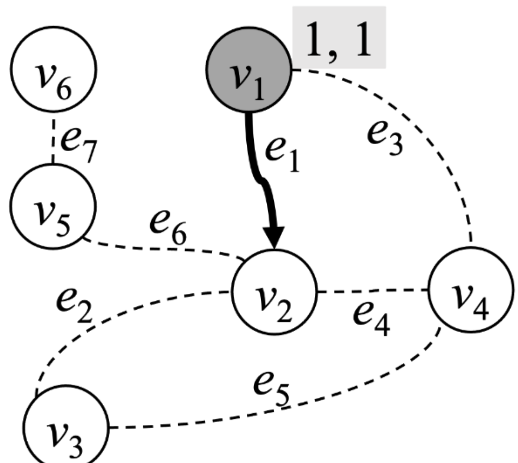
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
  if v.visited = false **then**
    入栈（S, (u, v)）;
    v.parent $\leftarrow u$;
    u.children $\leftarrow u$.children + 1;
    `DFSBlk`(G, v);
    u.low $\leftarrow \min\{u$.low, v.low$\}$;
    if (u.parent = null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
      输出（以下边组成一个块）;
      do
        $(x, y) \leftarrow$ 出栈（S）;
        输出（(x, y)）;
      while $(x, y) \neq (u, v)$;
  else if $v \neq u$.parent **then**
    if u.d > v.d **then**
      入栈（S, (u, v)）;
    u.low $\leftarrow \min\{u$.low, v.d$\}$;

e_1



块

- DFSBlk算法

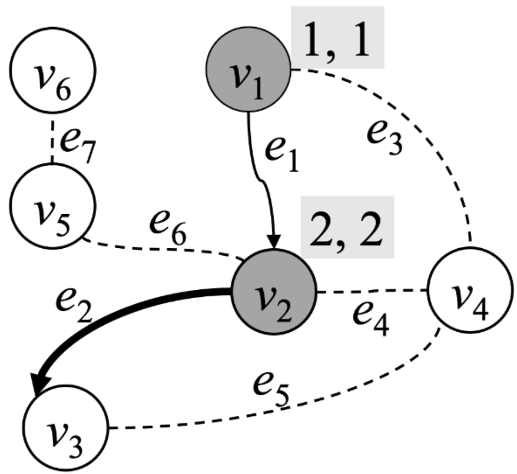
算法 4.1: DFSBlk

输入: 非平凡连通图 $G=\langle V,E\rangle$，顶点 u

初值: *time* 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
foreach (u, v) ∈ E do
    if v.visited = false then
        入栈 (S, (u, v));
        v.parent ← u;
        u.children ← u.children + 1;
        DFSBlk(G, v);
        u.low ← min{u.low, v.low};
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent ≠ null
         且 v.low ≥ u.d) then
            输出 (以下边组成一个块);
            do
                (x, y) ← 出栈 (S);
                输出 ((x, y));
            while (x, y) ≠ (u, v);
    else if v ≠ u.parent then
        if u.d > v.d then
            入栈 (S, (u, v));
        u.low ← min{u.low, v.d};
```

$e_1 \quad e_2$



块

- DFSBlk算法

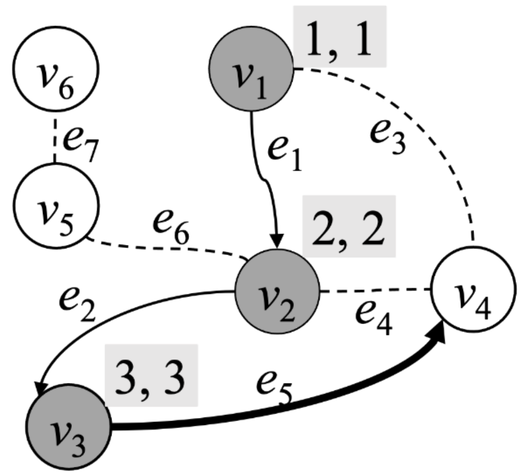
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
  if v.visited = false **then**
    入栈（S, (u, v)）;
    v.parent $\leftarrow u$;
    u.children $\leftarrow u$.children + 1;
    DFSBlk(G, v);
    u.low $\leftarrow$ min{u.low, v.low};
    if (u.parent = null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
      输出（以下边组成一个块）;
      do
        $(x, y) \leftarrow$ 出栈（S）;
        输出（(x, y)）;
      while $(x, y) \neq (u, v)$;
  else if $v \neq u$.parent **then**
    if u.d > v.d **then**
      入栈（S, (u, v)）;
    u.low $\leftarrow$ min{u.low, v.d};

e_1 e_2 e_5



块

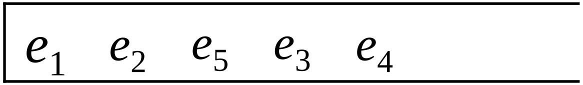
- DFSBlk算法

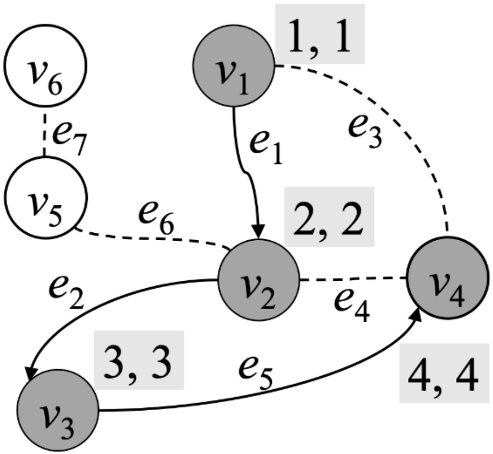
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
foreach (u, v) ∈ E do
    if v.visited = false then
        入栈 (S, (u, v));
        v.parent ← u;
        u.children ← u.children + 1;
        DFSBlk(G, v);
        u.low ← min{u.low, v.low};
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent ≠ null
         且 v.low ≥ u.d) then
            输出 (以下边组成一个块);
            do
                (x, y) ← 出栈 (S);
                输出 ((x, y));
            while (x, y) ≠ (u, v);
    else if v ≠ u.parent then
        if u.d > v.d then
            入栈 (S, (u, v));
        u.low ← min{u.low, v.d};
```





块

- DFSBlk算法

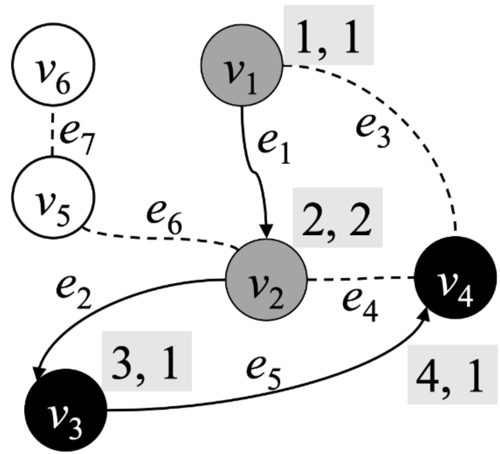
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
  if v.visited $=$ false **then**
    入栈 (S, (u, v));
    v.parent $\leftarrow u$;
    u.children $\leftarrow u$.children $+ 1$;
    DFSBlk(G, v);
    u.low $\leftarrow \min\{u$.low, v.low$\}$;
    if (u.parent $=$ null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
      输出 (以下边组成一个块);
      do
        $(x, y) \leftarrow$ 出栈 (S);
        输出 ((x, y));
      while $(x, y) \neq (u, v)$;
  else if $v \neq u$.parent **then**
    if u.d $>$ v.d **then**
      入栈 (S, (u, v));
    u.low $\leftarrow \min\{u$.low, v.d$\}$;

e_1	e_2	e_5	e_3	e_4



块

- DFSBlk算法

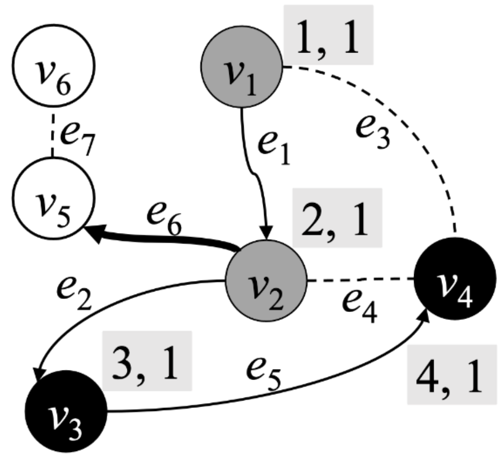
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
    if v.visited = false **then**
        入栈（S, (u, v)）;
        v.parent $\leftarrow u$;
        u.children $\leftarrow u$.children + 1;
        `DFSBlk`(G, v);
        u.low $\leftarrow \min\{u$.low, v.low$\}$;
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
            输出（以下边组成一个块）;
            do
                $(x, y) \leftarrow$ 出栈（S）;
                输出（(x, y)）;
            while $(x, y) \neq (u, v)$;
    else if $v \neq u$.parent **then**
        if u.d $> v$.d **then**
            入栈（S, (u, v)）;
        u.low $\leftarrow \min\{u$.low, v.d$\}$;

e_1 e_2 e_5 e_3 e_4 e_6



块

- DFSBlk算法

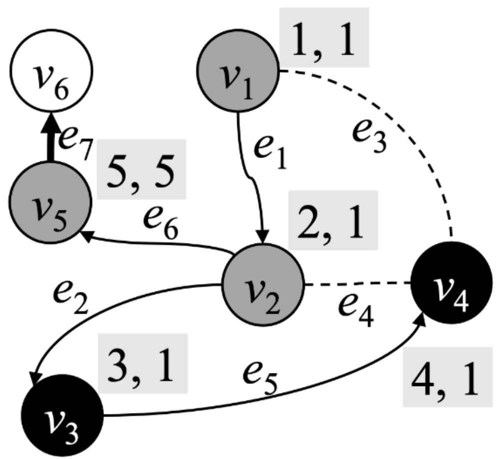
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E\rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
foreach (u, v) ∈ E do
    if v.visited = false then
        入栈 (S, (u, v));
        v.parent ← u;
        u.children ← u.children + 1;
        DFSBlk(G, v);
        u.low ← min{u.low, v.low};
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent ≠ null
          且 v.low ≥ u.d) then
            输出 (以下边组成一个块);
            do
                (x, y) ← 出栈 (S);
                输出 ((x, y));
            while (x, y) ≠ (u, v);
    else if v ≠ u.parent then
        if u.d > v.d then
            入栈 (S, (u, v));
        u.low ← min{u.low, v.d};
```

e_1	e_2	e_5	e_3	e_4	e_6	e_7



块

- DFSBlk算法

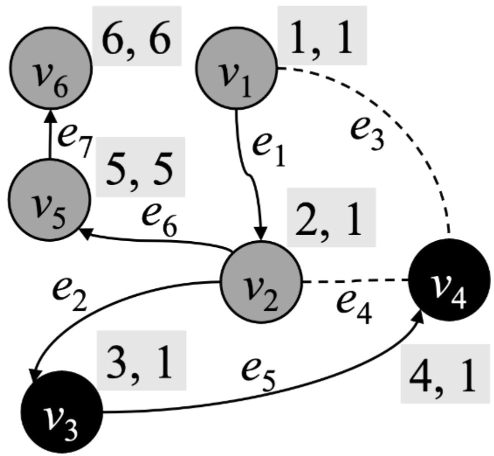
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
foreach (u, v) ∈ E do
    if v.visited = false then
        入栈 (S, (u, v));
        v.parent ← u;
        u.children ← u.children + 1;
        DFSBlk(G, v);
        u.low ← min{u.low, v.low};
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent ≠ null
         且 v.low ≥ u.d) then
            输出 (以下边组成一个块);
            do
                (x, y) ← 出栈 (S);
                输出 ((x, y));
            while (x, y) ≠ (u, v);
    else if v ≠ u.parent then
        if u.d > v.d then
            入栈 (S, (u, v));
        u.low ← min{u.low, v.d};
```

$e_1 \quad e_2 \quad e_5 \quad e_3 \quad e_4 \quad e_6 \quad e_7$



块

- DFSBlk算法

算法 4.1: DFSBlk

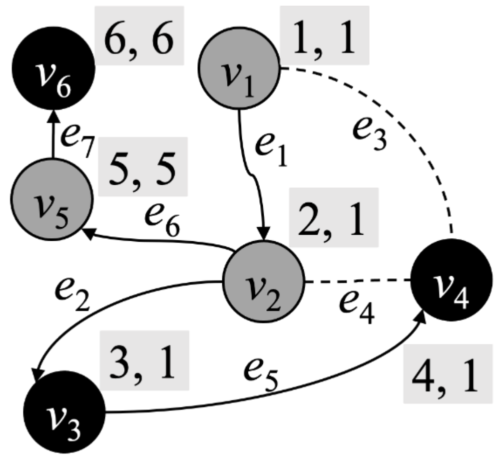
输入: 非平凡连通图 $G=\langle V,E\rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
foreach (u, v) ∈ E do
    if v.visited = false then
        入栈 (S, (u, v));
        v.parent ← u;
        u.children ← u.children + 1;
        DFSBlk(G, v);
        u.low ← min{u.low, v.low};
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent ≠ null
          且 v.low ≥ u.d) then
            输出 (以下边组成一个块);
            do
                (x, y) ← 出栈 (S);
                输出 ((x, y));
            while (x, y) ≠ (u, v);
    else if v ≠ u.parent then
        if u.d > v.d then
            入栈 (S, (u, v));
        u.low ← min{u.low, v.d};
```

以下边组成一个块：e_7

e_1 e_2 e_5 e_3 e_4 e_6



块

- DFSBlk算法

算法 4.1: DFSBlk

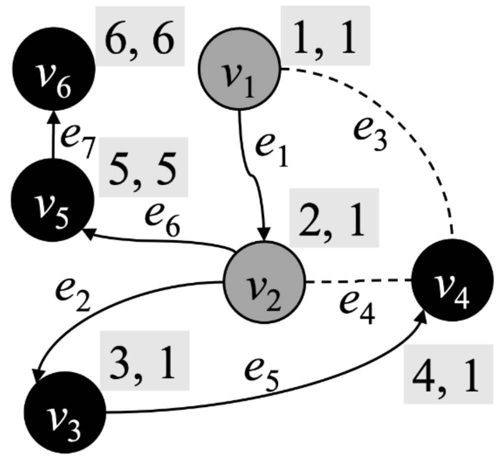
输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
  if v.visited = false **then**
    入栈（S, (u, v)）;
    v.parent $\leftarrow u$;
    u.children $\leftarrow u$.children + 1;
    DFSBlk(G, v);
    u.low $\leftarrow \min\{u.\text{low}, v.\text{low}\}$;
    if (u.parent = null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
      输出（以下边组成一个块）;
      do
        $(x, y) \leftarrow$ 出栈（S）;
        输出（(x, y)）;
      while $(x, y) \neq (u, v)$;
  else if $v \neq u$.parent **then**
    if u.d > v.d **then**
      入栈（S, (u, v)）;
    u.low $\leftarrow \min\{u.\text{low}, v.\text{d}\}$;

以下边组成一个块：e_6

e_1	e_2	e_5	e_3	e_4



块

- DFSBlk算法

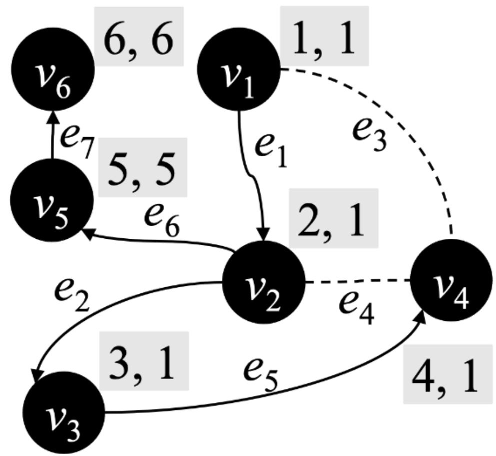
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E\rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
foreach (u, v) ∈ E do
    if v.visited = false then
        入栈 (S, (u,v));
        v.parent ← u;
        u.children ← u.children + 1;
        DFSBlk(G, v);
        u.low ← min{u.low, v.low};
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent ≠ null
          且 v.low ≥ u.d) then
            输出 (以下边组成一个块);
            do
                (x, y) ← 出栈 (S);
                输出 ((x, y));
            while (x, y) ≠ (u, v);
    else if v ≠ u.parent then
        if u.d > v.d then
            入栈 (S, (u,v));
        u.low ← min{u.low, v.d};
```

e_1 e_2 e_5 e_3 e_4



块

- DFSBlk算法

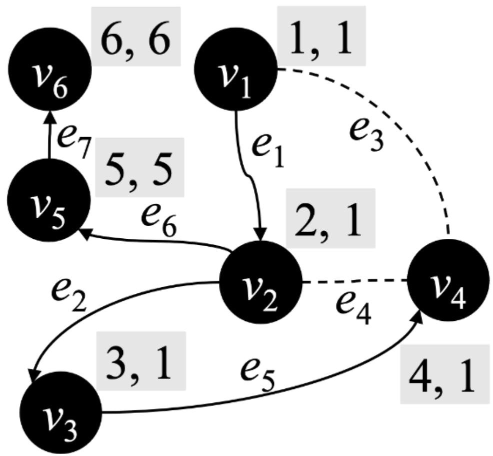
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
foreach (u, v) ∈ E do
    if v.visited = false then
        入栈 (S, (u, v));
        v.parent ← u;
        u.children ← u.children + 1;
        DFSBlk(G, v);
        u.low ← min{u.low, v.low};
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent ≠ null
         且 v.low ≥ u.d) then
            输出 (以下边组成一个块);
            do
                (x, y) ← 出栈 (S);
                输出 ((x, y));
            while (x, y) ≠ (u, v);
    else if v ≠ u.parent then
        if u.d > v.d then
            入栈 (S, (u, v));
        u.low ← min{u.low, v.d};
```

以下边组成一个块：$e_4\, e_3\, e_5\, e_2\, e_1$



块

- 从u发现(u, v)，到完成对v的递归调用并判定u为割点

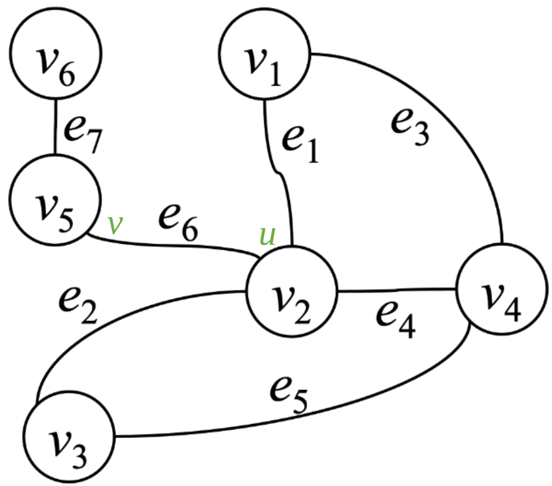
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: *time* 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
foreach (u, v) ∈ E do
    if v.visited = false then
        入栈 (S, (u, v));
        v.parent ← u;
        u.children ← u.children + 1;
        DFSBlk(G, v);
        u.low ← min{u.low, v.low};
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent ≠ null
         且 v.low ≥ u.d) then
            输出 (以下边组成一个块);
            do
                (x, y) ← 出栈 (S);
                输出 ((x, y));
            while (x, y) ≠ (u, v);
    else if v ≠ u.parent then
        if u.d > v.d then
            入栈 (S, (u, v));
        u.low ← min{u.low, v.d};
```

这个时间段内，发现了哪些边？



块

- 从u发现(u, v)，到完成对v的递归调用并判定u为割点

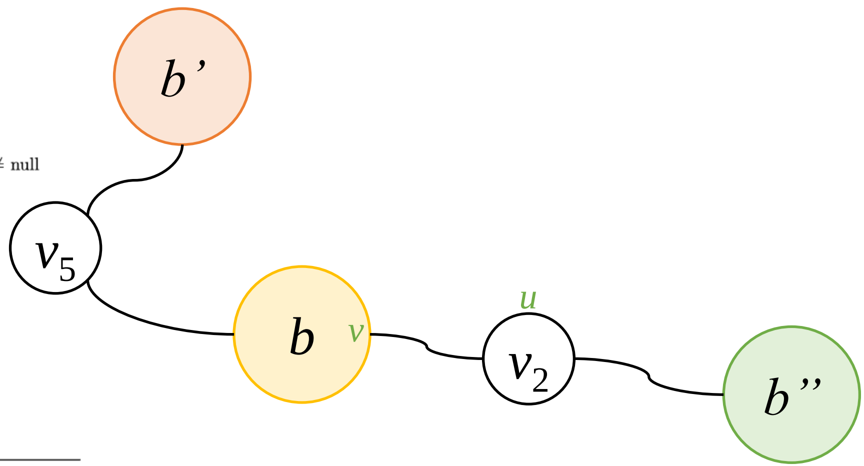
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E\rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
foreach (u, v) ∈ E do
    if v.visited = false then
        入栈 (S, (u, v));
        v.parent ← u;
        u.children ← u.children + 1;
        DFSBlk(G, v);
        u.low ← min{u.low, v.low};
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent ≠ null
          且 v.low ≥ u.d) then
            输出 (以下边组成一个块);
            do
                (x, y) ← 出栈 (S);
                输出 ((x, y));
            while (x, y) ≠ (u, v);
    else if v ≠ u.parent then
        if u.d > v.d then
            入栈 (S, (u, v));
        u.low ← min{u.low, v.d};
```

这个时间段内，发现了哪些边？



块

- 从u发现(u, v)，到完成对v的递归调用并判定u为割点

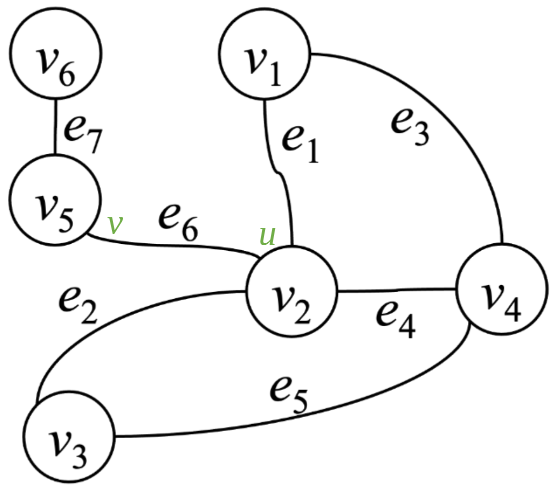
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: *time* 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
 if v.visited = false **then**
  入栈 (S, (u, v));
  v.parent $\leftarrow u$;
  u.children $\leftarrow u$.children + 1;
  `DFSBlk`(G, v);
  u.low $\leftarrow \min\{u$.low, v.low$\}$;
  if (u.parent = null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
   输出 (以下边组成一个块);
   do
    $(x, y) \leftarrow$ 出栈 (S);
    输出 ((x, y));
   while $(x, y) \neq (u, v)$;
 else if $v \neq u$.parent **then**
  if u.d $> v$.d **then**
   入栈 (S, (u, v));
  u.low $\leftarrow \min\{u$.low, v.d$\}$;

这个时间段结束时，输出哪些边？



块

- 从u发现(u, v)，到完成对v的递归调用并判定u为割点

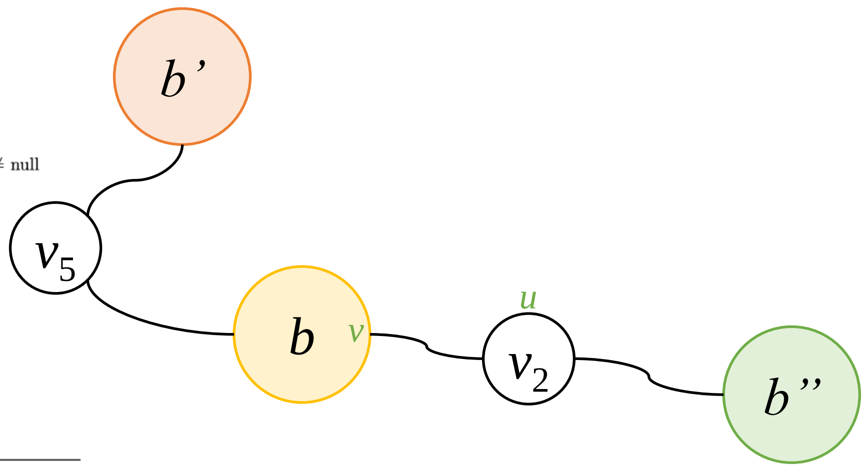
算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E\rangle$，顶点 u

初值: *time* 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
  if v.visited = false **then**
    入栈（S, (u, v)）;
    v.parent $\leftarrow u$;
    u.children $\leftarrow u$.children + 1;
    DFSBlk(G, v);
    u.low $\leftarrow$ min{u.low, v.low};
    if (u.parent = null 且 u.children $\geq$ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
      输出（以下边组成一个块）;
      do
        $(x, y) \leftarrow$ 出栈（S）;
        输出（(x, y)）;
      while $(x, y) \neq (u, v)$;
  else if $v \neq u$.parent **then**
    if u.d > v.d **then**
      入栈（S, (u, v)）;
    u.low $\leftarrow$ min{u.low, v.d};

这个时间段结束时，输出哪些边？



块

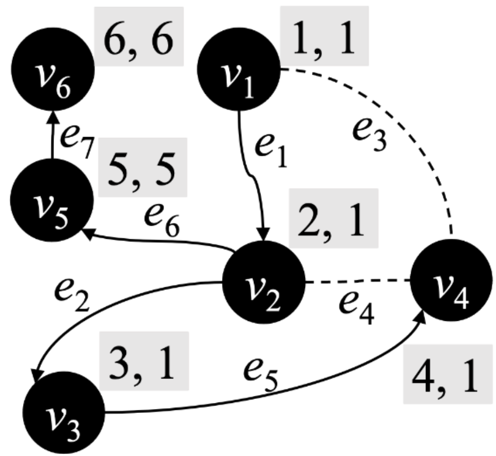
- 算法结束时，何时栈空，何时栈非空？

算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
    if v.visited = false **then**
        入栈（S, (u, v));
        v.parent $\leftarrow u$;
        u.children $\leftarrow u$.children + 1;
        DFSBlk(G, v);
        u.low $\leftarrow$ min{u.low, v.low};
        if (u.parent = null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
            输出（以下边组成一个块）;
            do
                $(x, y) \leftarrow$ 出栈（S）;
                输出（(x, y)）;
            while $(x, y) \neq (u, v)$;
    else if $v \neq u$.parent **then**
        if u.d $>$ v.d **then**
            入栈（S, (u, v));
        u.low $\leftarrow$ min{u.low, v.d};



块

- 时间复杂度：$O(n + m)$

算法 4.1: DFSBlk

输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

初值: $time$ 初值为 0；V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0；S 初值为空栈

$time \leftarrow time + 1$;
u.d $\leftarrow time$;
u.low $\leftarrow u$.d;
u.visited $\leftarrow$ true;
foreach $(u, v) \in E$ **do**
    if v.visited $=$ false **then**
        入栈（S, (u, v)）;
        v.parent $\leftarrow u$;
        u.children $\leftarrow u$.children $+ 1$;
        `DFSBlk`(G, v);
        u.low $\leftarrow$ min{u.low, v.low};
        if (u.parent $=$ null 且 u.children ≥ 2) 或 (u.parent $\neq$ null 且 v.low $\geq u$.d) **then**
            输出（以下边组成一个块）;
            do
                $(x, y) \leftarrow$ 出栈（S）;
                输出（(x, y)）;
            while $(x, y) \neq (u, v)$;
    else if $v \neq u$.parent **then**
        if u.d $>$ v.d **then**
            入栈（S, (u, v)）;
        u.low $\leftarrow$ min{u.low, v.d};

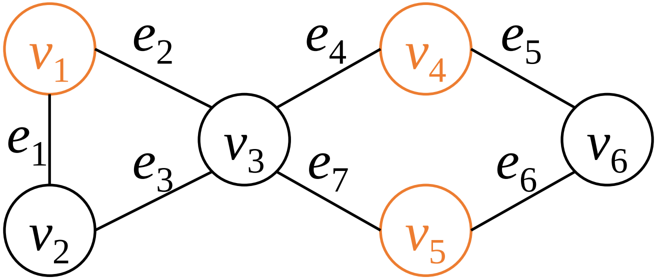
本次课的主要内容

4.1 块

4.2 割集和连通度

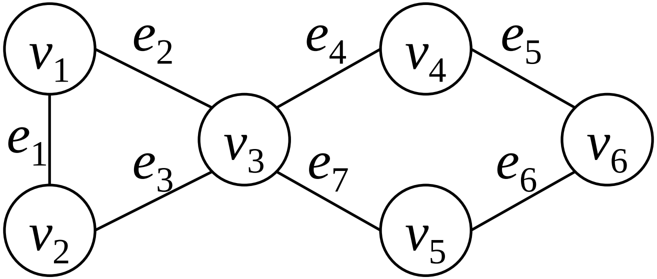
割集和连通度

- **点割集**（分离集）
  - 对于图$G = <V, E>$和顶点子集$S \subseteq V$，$G - S$的连通分支数量大于G的连通分支数量



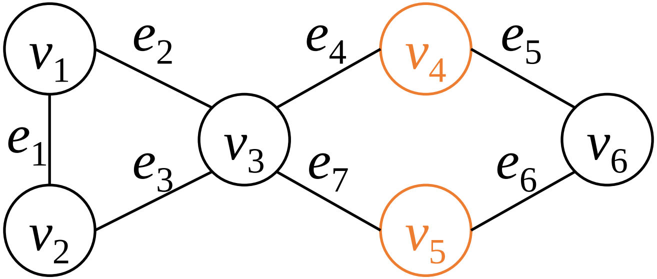
割集和连通度

- 点割集（分离集）
  - 对于图$G = <V, E>$和顶点子集$S \subseteq V$,
    $G - S$的连通分支数量大于G的连通分支数量
- 点割集和割点的关系？



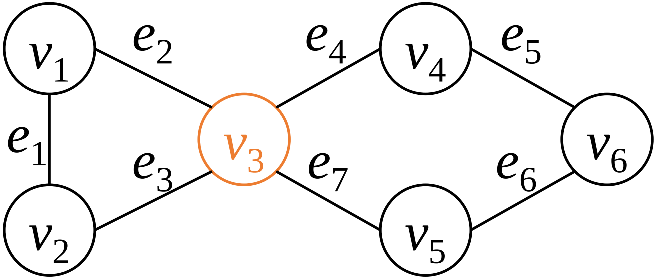
割集和连通度

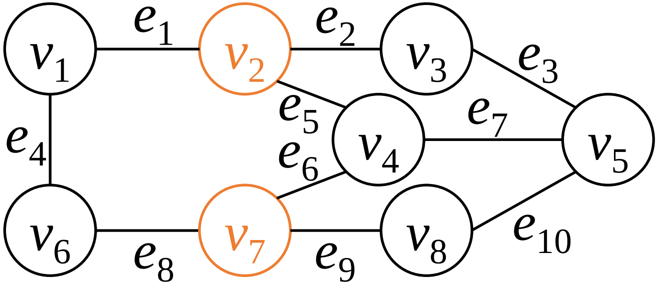
- 点割集（分离集）
  - 对于图$G = <V, E>$和顶点子集$S \subseteq V$, $G - S$的连通分支数量大于G的连通分支数量
- 点割集和割点的关系?
- **极小点割集**



割集和连通度

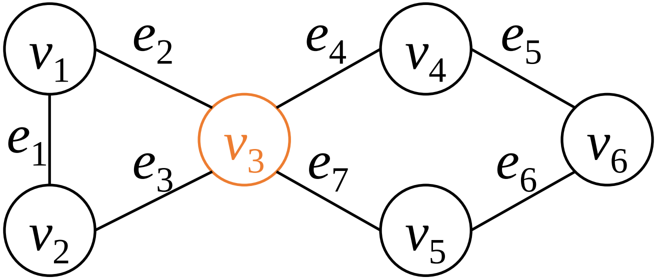
- 点割集（分离集）
  - 对于图$G = <V, E>$和顶点子集$S \subseteq V$，$G - S$的连通分支数量大于G的连通分支数量
- 点割集和割点的关系?
- 极小点割集
- **最小点割集**

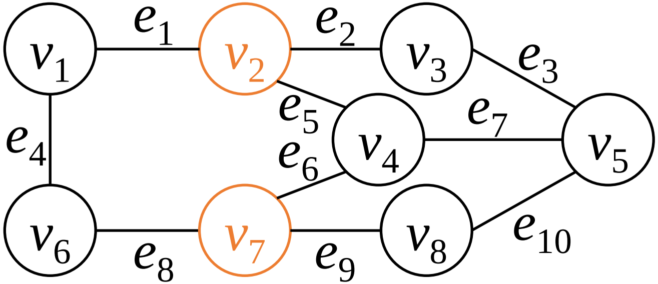




割集和连通度

- **点连通度**（连通度）
  - 为使图G不连通或成为平凡图，至少需要从G中删除的顶点数量，记作$\kappa(G)$

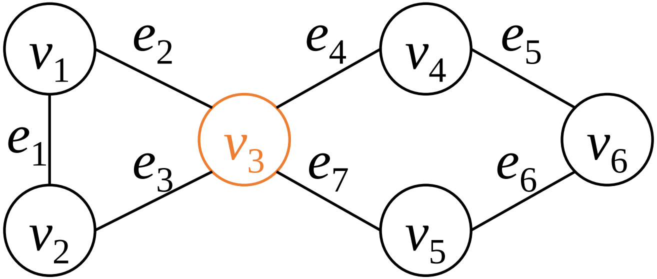


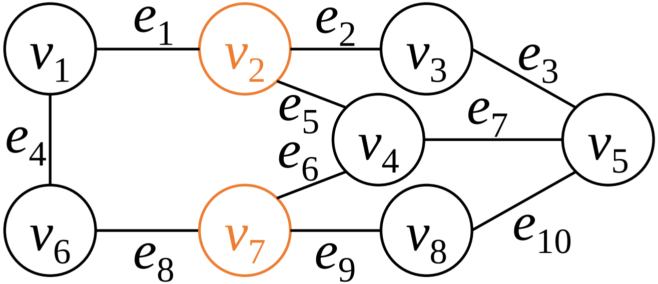


割集和连通度

- 点连通度（连通度）
  - 为使图G不连通或成为平凡图，至少需要从G中删除的顶点数量，记作$\kappa(G)$

- 若连通图G有点割集，则G的最小点割集和$\kappa(G)$有什么关系？

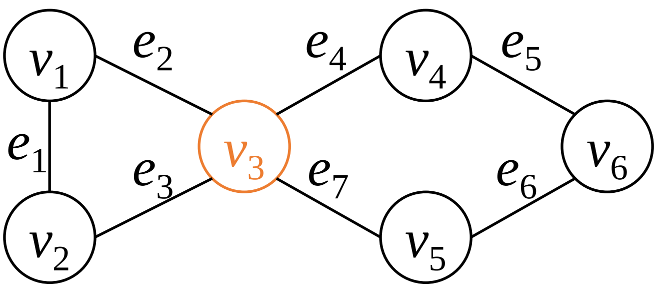


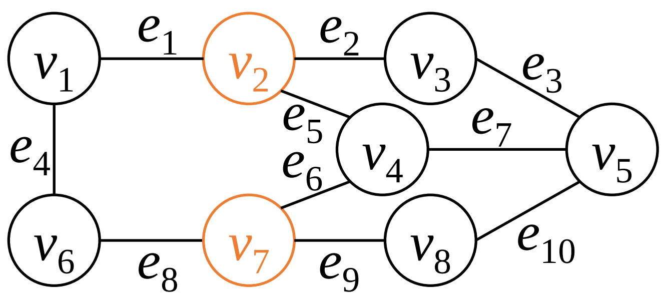


割集和连通度

- 点连通度（连通度）
  - 为使图G不连通或成为平凡图，至少需要从G中删除的顶点数量，记作$\kappa(G)$

- 若连通图G有点割集，则G的最小点割集和$\kappa(G)$有什么关系？
- 完全图K_n的连通度是多少？

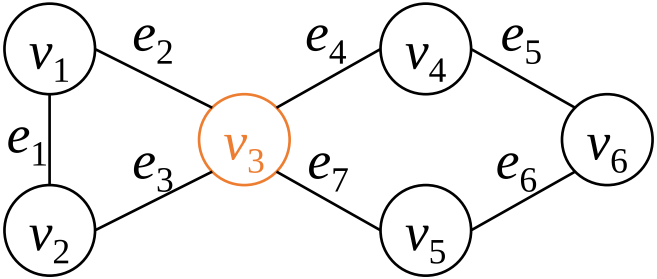


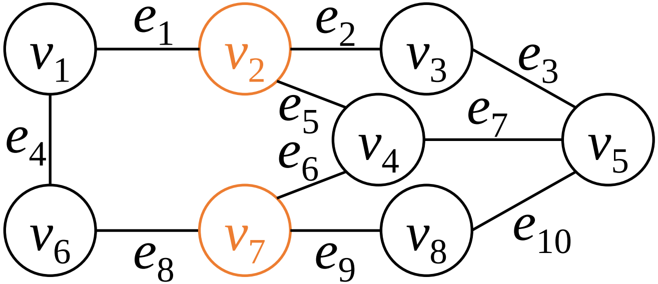


割集和连通度

- 点连通度（连通度）
  - 为使图G不连通或成为平凡图，至少需要从G中删除的顶点数量，记作$\kappa(G)$

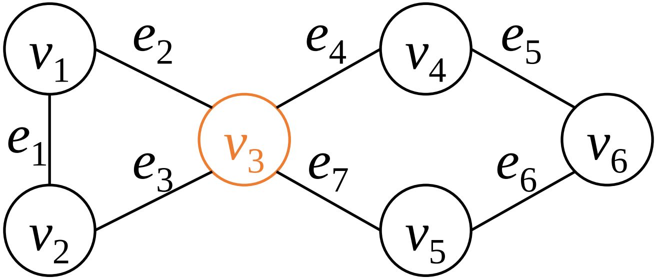
- 若连通图G有点割集，则G的最小点割集和$\kappa(G)$有什么关系？
- 完全图K_n的连通度是多少？
- 不连通图的连通度是多少？

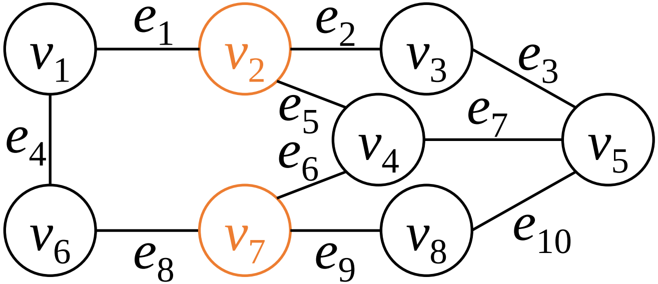




割集和连通度

- ***k*点连通图**（k连通图）
  - $k \leq \kappa(G)$

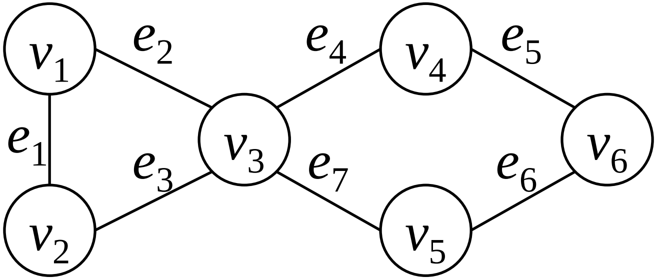


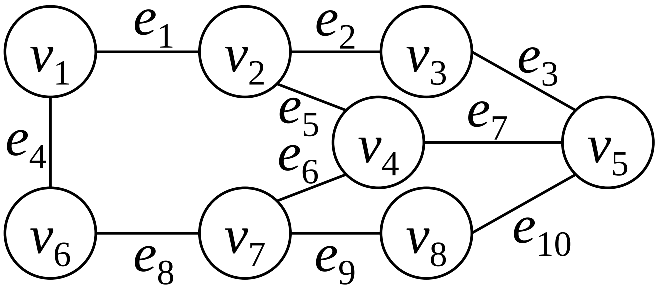


割集和连通度

- k点连通图（k连通图）
  - $k \le \kappa(G)$

- 1连通图一定是连通图吗？反之呢？

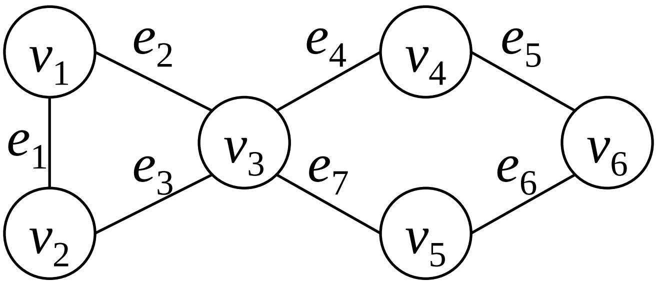


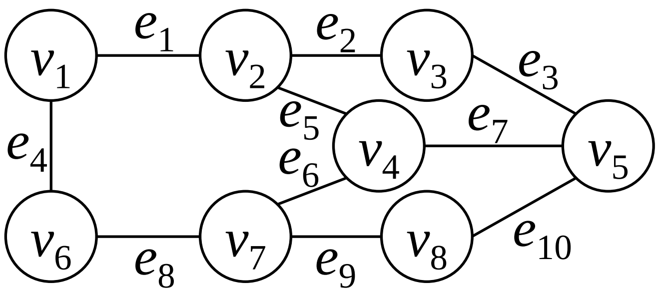


割集和连通度

- k点连通图（k连通图）
  - $k \le \kappa(G)$

- 1连通图一定是连通图吗？反之呢？
- 2连通图一定是块吗？反之呢？

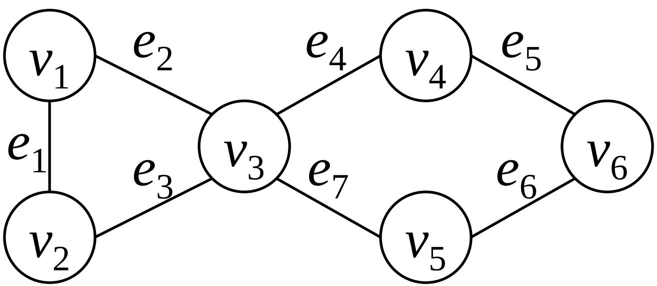


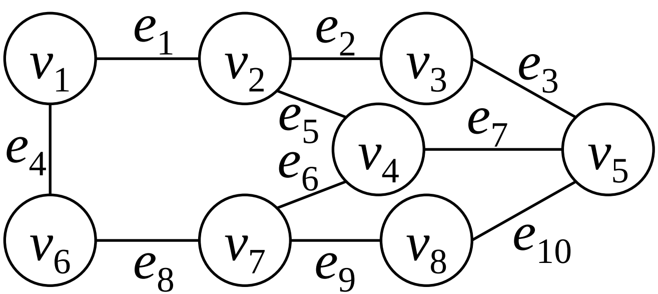


割集和连通度

- k点连通图（k连通图）
  - $k \le \kappa(G)$

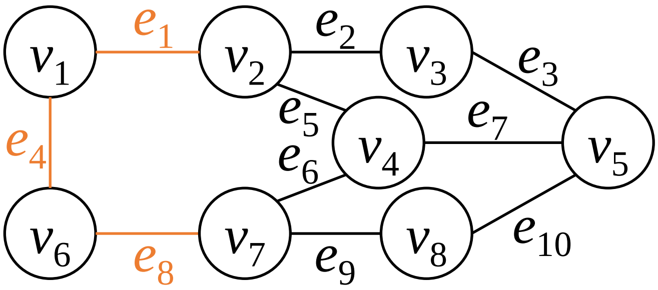
- 1连通图一定是连通图吗？反之呢？
- 2连通图一定是块吗？反之呢？
- 从k连通图中删除任意$k-1$个顶点，剩余图一定连通吗？





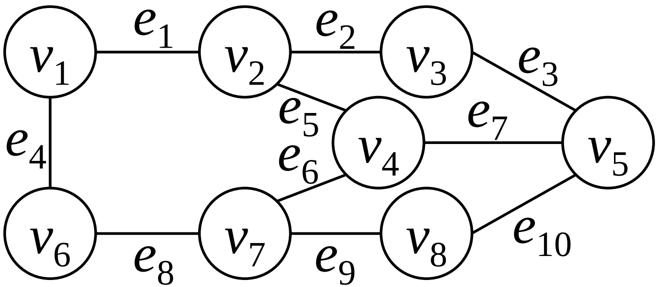
割集和连通度

- **边割集**
  - 对于图$G = \langle V, E\rangle$和边子集$S' \subseteq E$，
    $G - S'$的连通分支数量大于G的连通分支数量



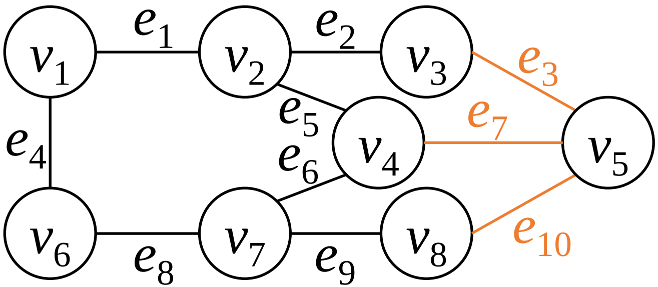
割集和连通度

- 边割集
  - 对于图$G = <V, E>$和边子集$S' \subseteq E$，$G - S'$的连通分支数量大于G的连通分支数量
- 边割集和割边的关系？



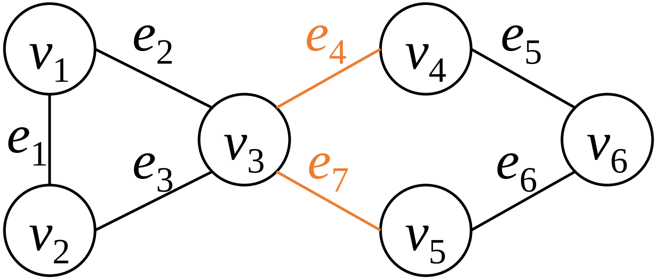
割集和连通度

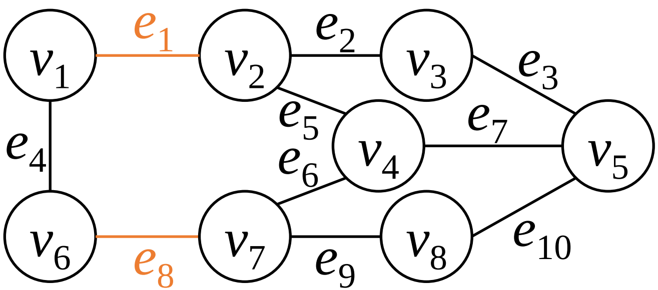
- 边割集
  - 对于图G = <V, E>和边子集$S' \subseteq E$,
    $G - S'$的连通分支数量大于G的连通分支数量
- 边割集和割边的关系?
- **极小边割集**



割集和连通度

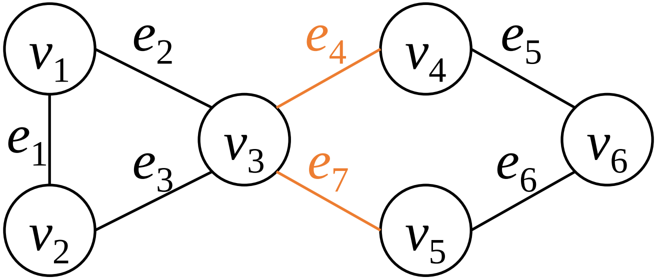
- 边割集
  - 对于图$G = <V, E>$和边子集$S' \subseteq E$,
    $G - S'$的连通分支数量大于G的连通分支数量
- 边割集和割边的关系?
- 极小边割集
- **最小边割集**

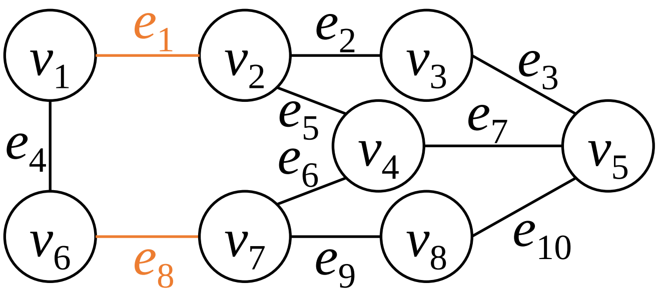




割集和连通度

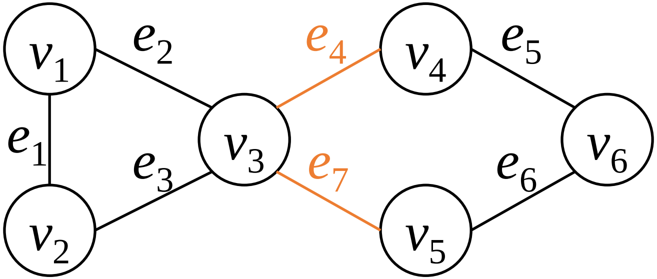
- **边连通度**
  - 为使图G不连通或成为平凡图，至少需要从G中删除的边的数量，记作$\kappa'(G)$

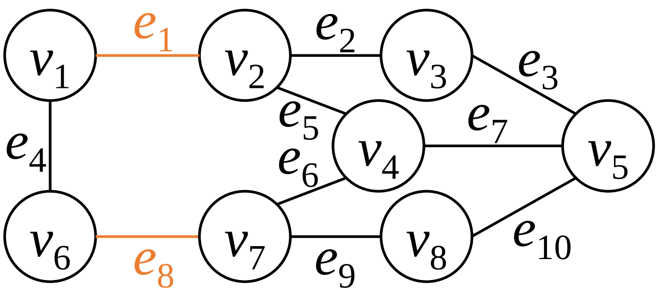




割集和连通度

- 边连通度
  - 为使图G不连通或成为平凡图，至少需要从G中删除的边的数量，记作$\kappa'(G)$
- 若连通图G有边割集，则G的最小边割集和$\kappa'(G)$有什么关系？

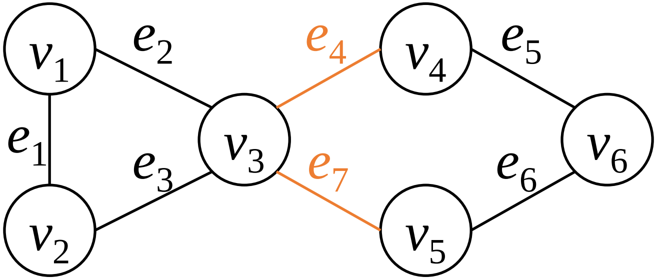


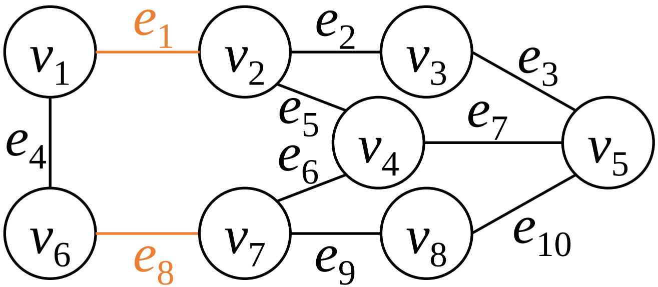


割集和连通度

- 边连通度
  - 为使图G不连通或成为平凡图，至少需要从G中删除的边的数量，记作$\kappa'(G)$

- 若连通图G有边割集，则G的最小边割集和$\kappa'(G)$有什么关系？
- 完全图K_n的边连通度是多少？

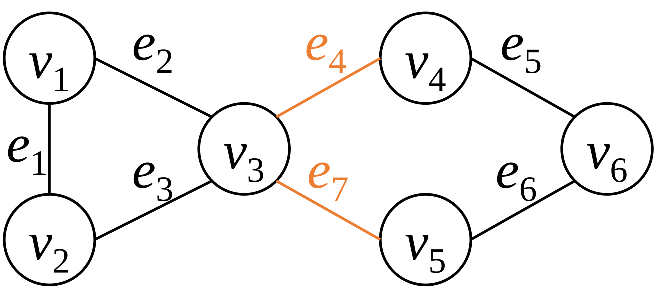


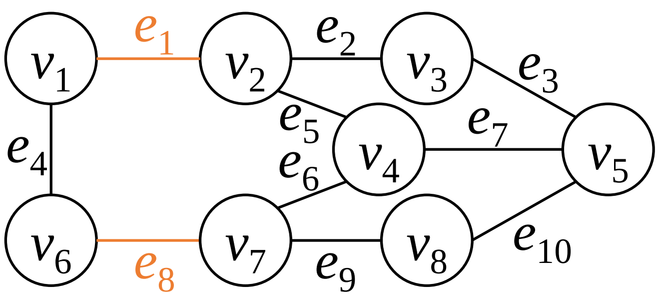


割集和连通度

- 边连通度
  - 为使图G不连通或成为平凡图，至少需要从G中删除的边的数量，记作$\kappa'(G)$

- 若连通图G有边割集，则G的最小边割集和$\kappa'(G)$有什么关系？
- 完全图K_n的边连通度是多少？
  - 为什么$\kappa'(G) \le n-1$？

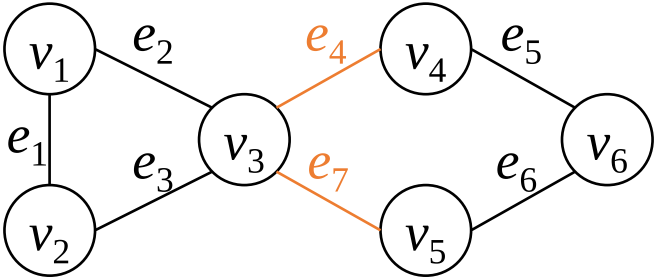


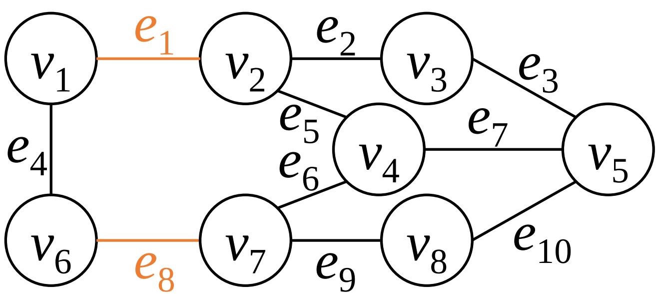


割集和连通度

- 边连通度
  - 为使图G不连通或成为平凡图，至少需要从G中删除的边的数量，记作$\kappa'(G)$

- 若连通图G有边割集，则G的最小边割集和$\kappa'(G)$有什么关系？
- 完全图K_n的边连通度是多少？
  - 为什么$\kappa'(G) \le n-1$？
  - 为什么$\kappa'(G) \ge n-1$？

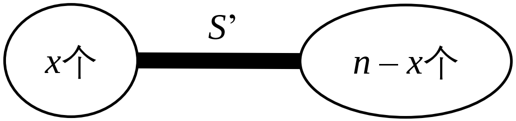




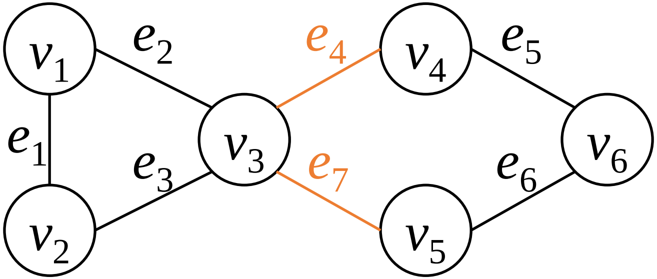
割集和连通度

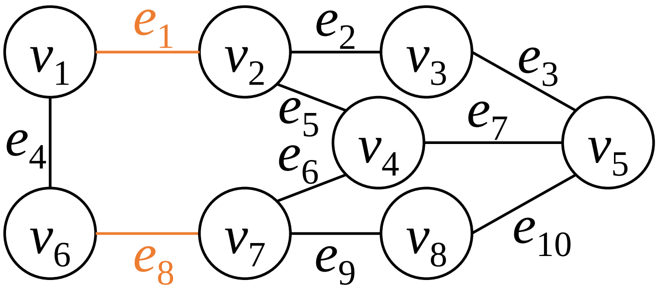
- 边连通度
  - 为使图G不连通或成为平凡图，至少需要从G中删除的边的数量，记作$\kappa'(G)$

- 若连通图G有边割集，则G的最小边割集和$\kappa'(G)$有什么关系？
- 完全图K_n的边连通度是多少？
  - 为什么$\kappa'(G) \le n-1$?
  - 为什么$\kappa'(G) \ge n-1$?



$|S'| = x(n\text{-}x) < n\text{-}1 \rightarrow (x\text{-}1)(n\text{-}x\text{-}1) < 0$，矛盾

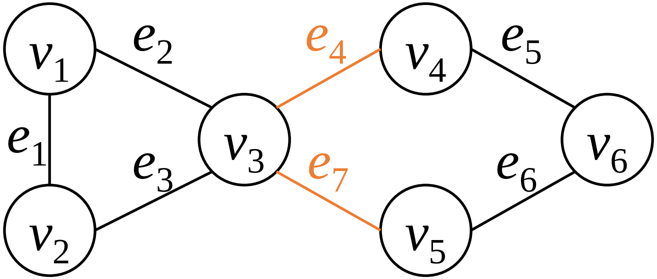


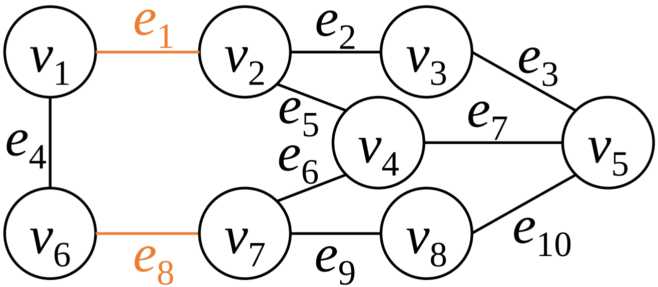


割集和连通度

- 边连通度
  - 为使图G不连通或成为平凡图，至少需要从G中删除的边的数量，记作$\kappa'(G)$

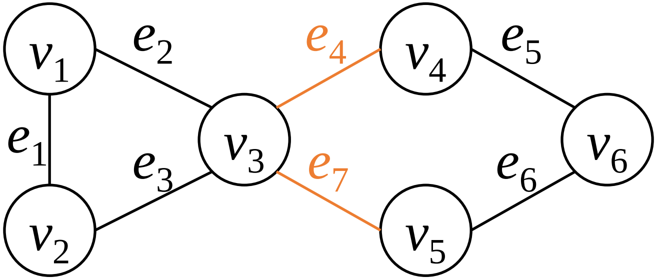
- 若连通图G有边割集，则G的最小边割集和$\kappa'(G)$有什么关系？
- 完全图K_n的边连通度是多少？
- 不连通图的边连通度是多少？

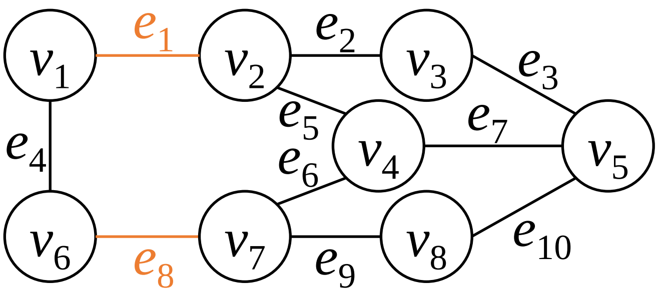




割集和连通度

- ***k*边连通图**
  - $k \leq \kappa'(G)$

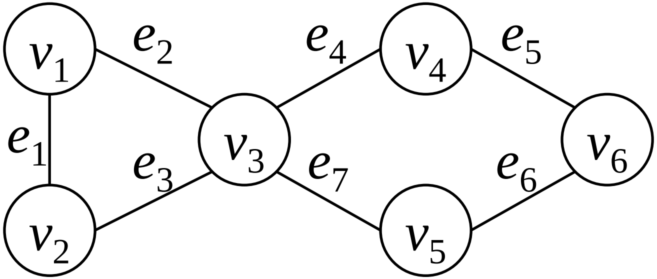


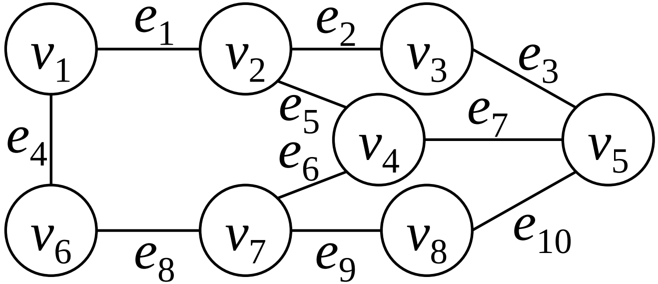


割集和连通度

- k边连通图
  - $k \le \kappa'(G)$

- 1边连通图一定是连通图吗？反之呢？

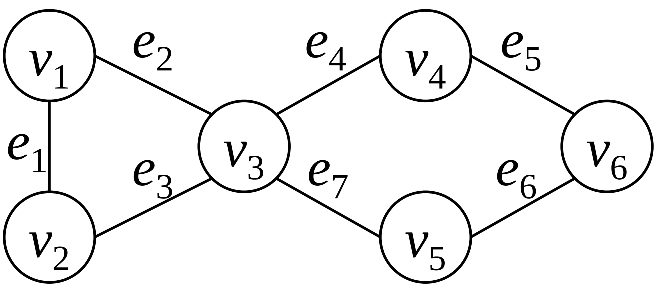


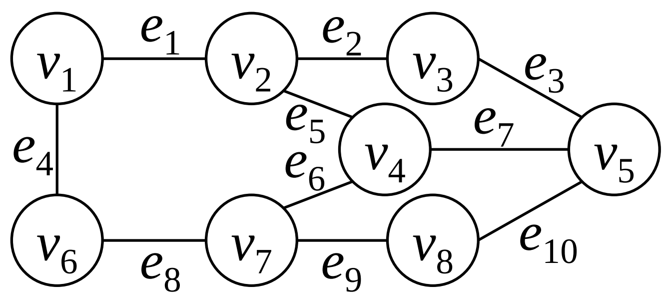


割集和连通度

- k边连通图
  - $k \le \kappa'(G)$

- 1边连通图一定是连通图吗？反之呢？
- 2边连通图一定是块吗？反之呢？

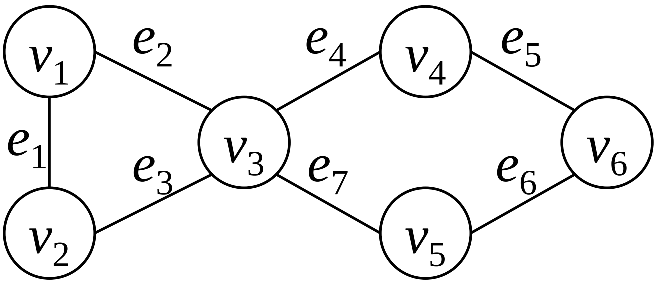


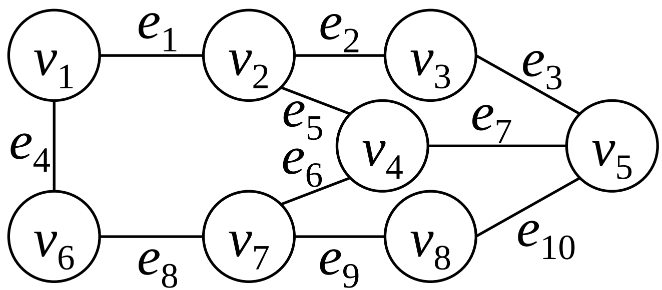


割集和连通度

- k边连通图
  - $k \leq \kappa'(G)$

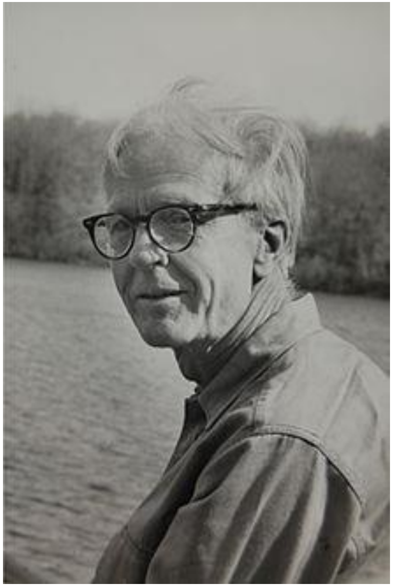
- 1边连通图一定是连通图吗？反之呢？
- 2边连通图一定是块吗？反之呢？
- 从k边连通图中删除任意$k-1$条边，剩余图一定连通吗？





割集和连通度

- Hassler Whitney，1907年出生于美国，1982年获得沃尔夫数学奖



奇点理论奠基人之一
拟阵理论奠基人之一
……

https://en.wikipedia.org/wiki/Hassler_Whitney

割集和连通度

- 对于任意一个图G：$\kappa(G) \le \kappa'(G) \le \delta(G)$

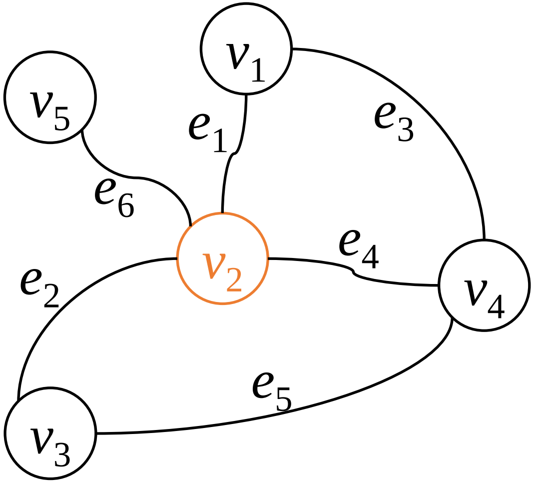
割集和连通度

- 决定图的点（边）连通度的本质要素是什么？
  向图中增加边，图的点连通度和边连通度一定提高吗？

割集和连通度

- 决定图的点（边）连通度的本质要素是什么？
  向图中增加边，图的点连通度和边连通度一定提高吗？

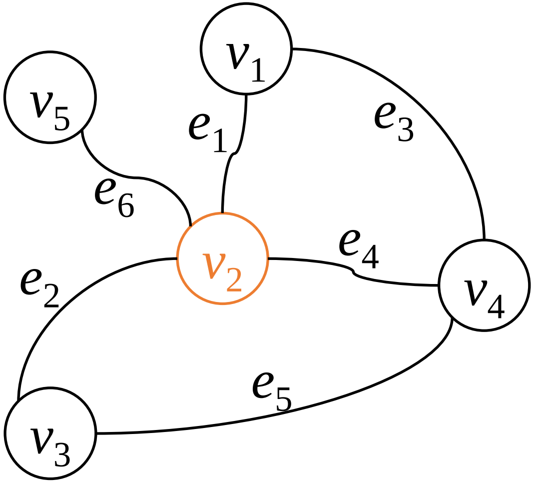
- 对于有割点的连通图，删除割点后不连通的本质原因是什么？



割集和连通度

- 决定图的点（边）连通度的本质要素是什么？
  向图中增加边，图的点连通度和边连通度一定提高吗？

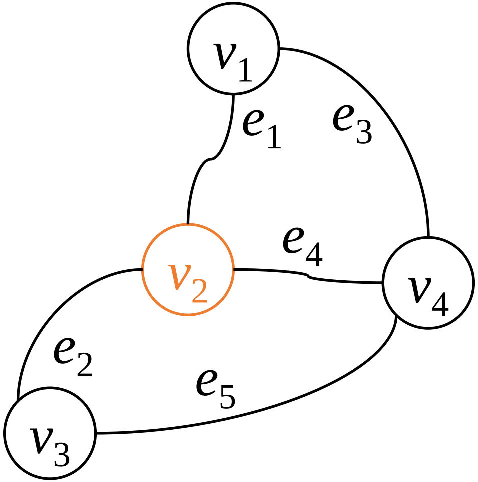
- 对于有割点的连通图，删除割点后不连通的本质原因是什么？
  - 割点是某两个顶点间的所有路的公共内顶点



割集和连通度

- 决定图的点（边）连通度的本质要素是什么？
  向图中增加边，图的点连通度和边连通度一定提高吗？

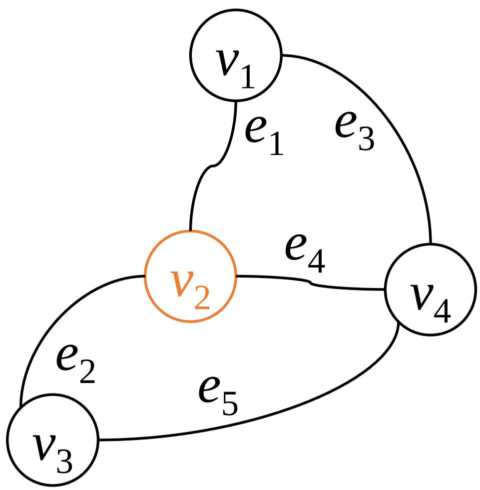
- 对于块，删除任意顶点后仍连通的本质原因是什么？



割集和连通度

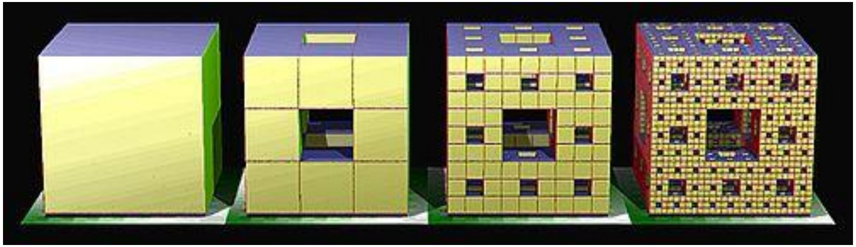
- 决定图的点（边）连通度的本质要素是什么？
  向图中增加边，图的点连通度和边连通度一定提高吗？

- 对于块，删除任意顶点后仍连通的本质原因是什么？
  - 任意两个顶点间存在至少2条无公共内顶点的路



割集和连通度

- Karl Menger，1902年出生于奥匈帝国



https://en.wikipedia.org/wiki/Karl_Menger
https://en.wikipedia.org/wiki/Menger_sponge

割集和连通度

- 门格尔定理：对于图$G = \langle V, E \rangle$和两个不相邻的顶点$u, v \in V$，使u和v不连通至少需要从G中删除的顶点数量等于G中两两无公共内顶点的u-v路的最大数量。

- 推论：非平凡图G是k连通图当且仅当G中任意两个顶点间存在至少k条两两无公共内顶点的路。

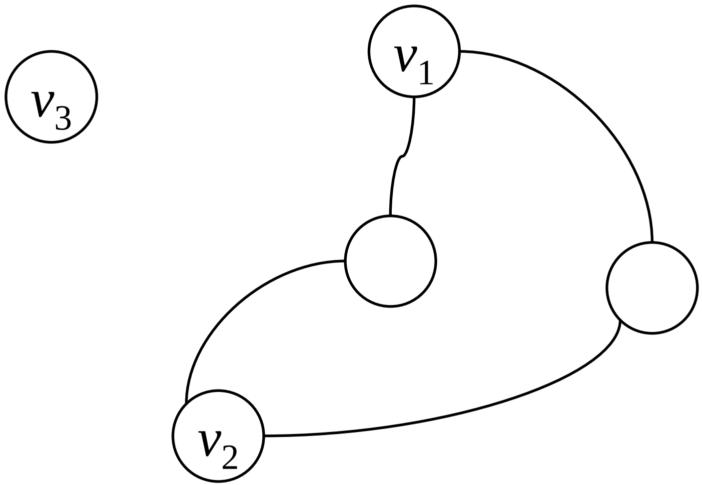
- 上述定理和推论面向的是点连通度，面向边连通度的结论是类似的。

割集和连通度

- 对于k连通图G = <V, E>和任意k个顶点$v_1, \ldots, v_k \in V$（$k \geq 2$），G含圈经过$v_1, \ldots, v_k$。

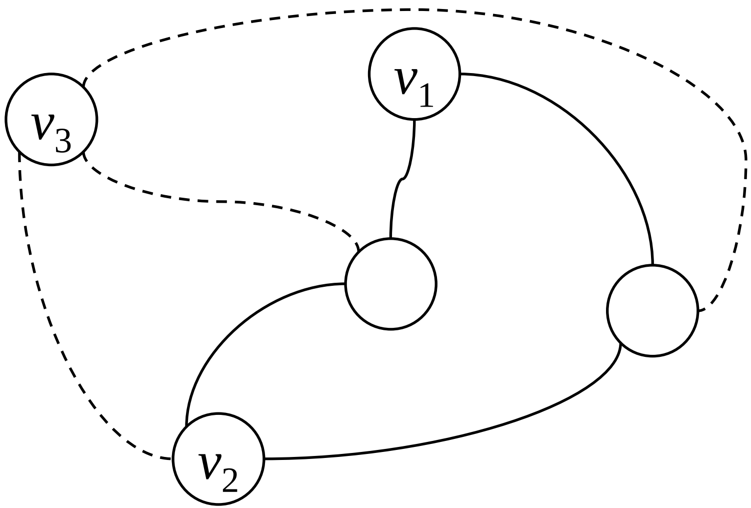
割集和连通度

- 对于k连通图$G = \langle V, E \rangle$和任意k个顶点$v_1, \ldots, v_k \in V$（$k \geq 2$），G含圈经过$v_1, \ldots, v_k$。



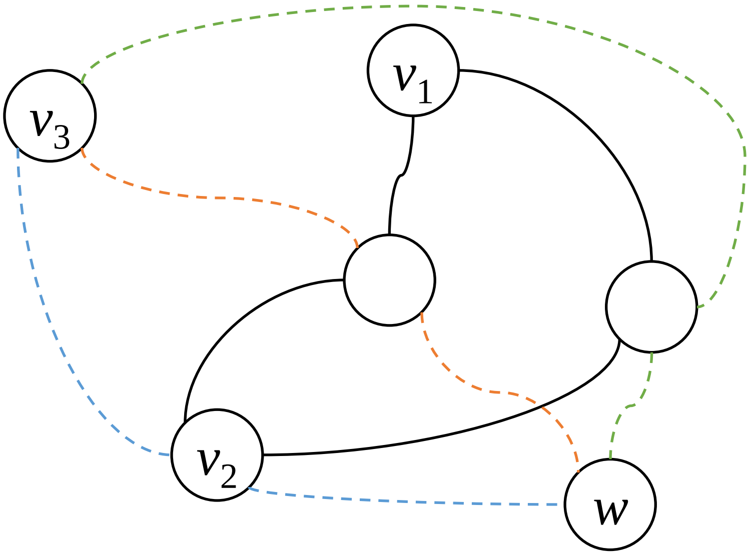
割集和连通度

- 对于k连通图G = <V, E>和任意k个顶点$v_1, \ldots, v_k \in V$（$k \geq 2$），G含圈经过$v_1, \ldots, v_k$。



割集和连通度

- 对于k连通图G = <V, E>和任意k个顶点$v_1, \ldots, v_k \in V$（$k \geq 2$），G含圈经过$v_1, \ldots, v_k$。



割集和连通度

- 如何计算图的点（边）连通度?

割集和连通度

- 如何计算图的点（边）连通度?

- 根据：非平凡图G是k（边）连通图当且仅当G中任意两个顶点间存在至少k条两两无公共内顶点（公共边）的路。
- 问题转化：如何计算任意两个顶点间无公共内顶点（公共边）的路的最大数量?
- 进一步转化：最大流问题

书面作业

- 练习4.1
- 练习4.4、4.5、4.6、4.7