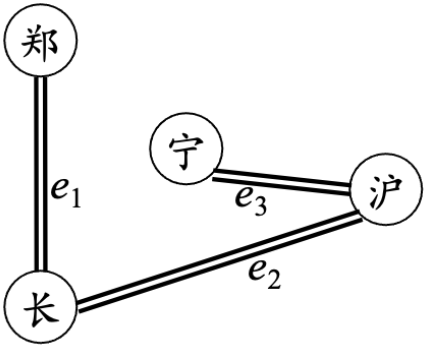
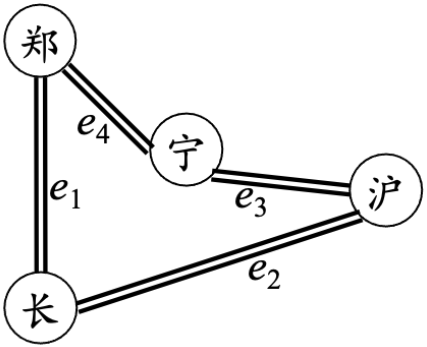
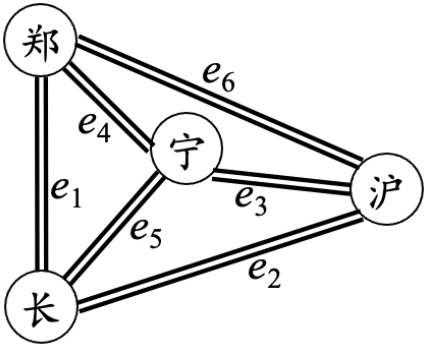
3-4 连通度

程龚

连通度有什么用？







你能解释这些术语吗？

■ 块

思考题 4.2 完全图是块吗？

思考题 4.3 树是块吗？

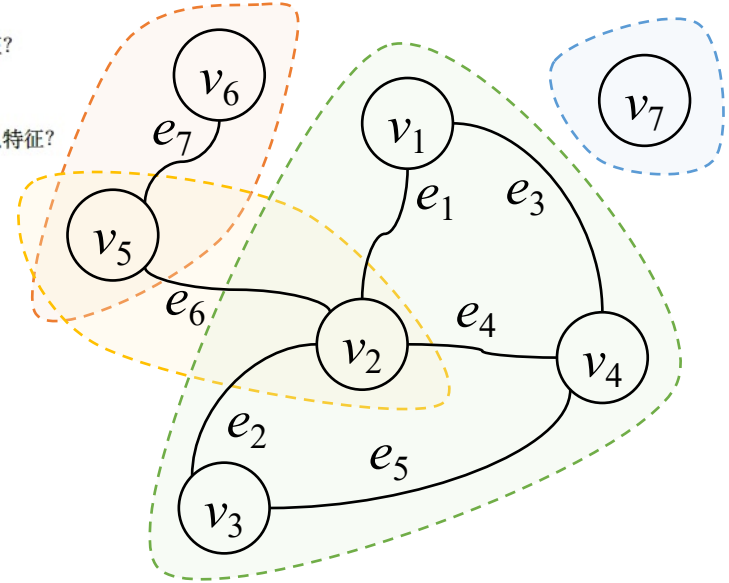
思考题 4.4 欧拉图和哈密尔顿图是块吗？

思考题 4.5 若图的块只含一个顶点，则这种顶点有什么特征？

思考题 4.6 若图的块只含一条边，则这种边有什么特征？

思考题 4.7 两个块至多含多少个公共顶点？这种顶点有什么特征？

思考题 4.8 两个块至多含多少条公共边？



请证明下述定理

定理 4.1 块为边集定义了一种等价关系。

- 证明：定义了一种划分

块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，以下是块的等价定义：

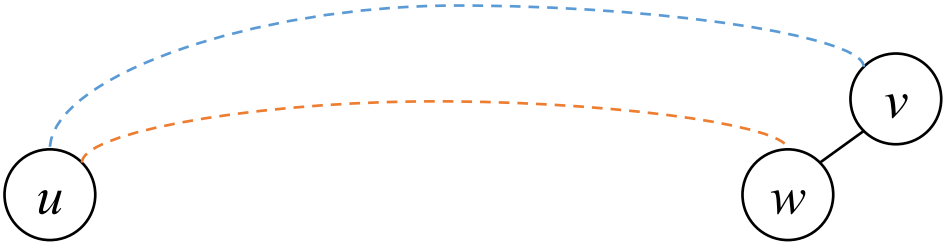
1. G 没有割点。
2. 对于任意两个顶点 $u, v \in V$，G 含两条无公共内顶点的 u-v 路。
3. 对于任意两个顶点 $u, v \in V$，G 含圈经过 u 和 v。
4. 对于任意一个顶点 $v \in V$ 和任意一条边 $e \in E$，G 含圈经过 v 和 e。
5. 对于任意两条边 $e, f \in E$，G 含圈经过 e 和 f。
6. 对于任意两个顶点 $u, v \in V$ 和任意一条边 $e \in E$，G 含 u-v 路经过 e。
7. 对于任意三个顶点 $u, v, w \in V$，G 含 u-v 路经过 w。
8. 对于任意三个顶点 $u, v, w \in V$，G 含 u-v 路不经过 w。

块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，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。

- 删去v的邻点w（不是割点），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？

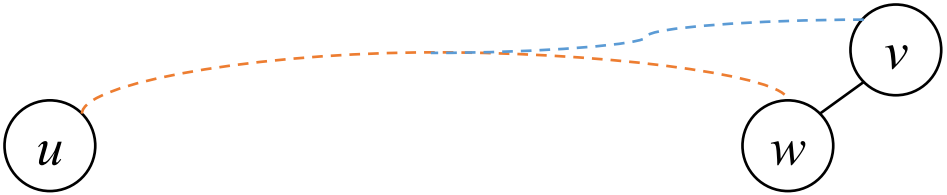


块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，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。

- 删去v的邻点w（不是割点），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？

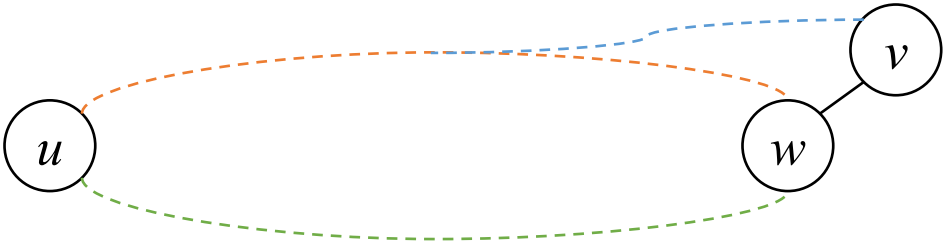


块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，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。

- 删去v的邻点w（不是割点），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？
- 所以想到对两个顶点间的距离归纳。

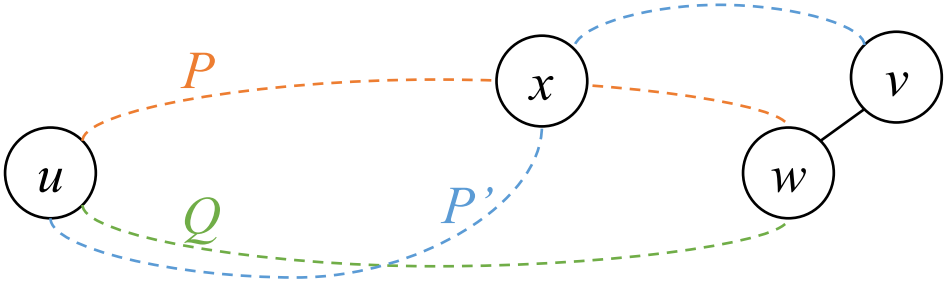


块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，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。

- 删去v的邻点w（不是割点），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？
- 所以想到对两个顶点间的距离归纳。
  - x：不经过w的u-v路与两条u-w路的最后一个公共顶点



块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，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。

- 反证法

块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，以下是块的等价定义：

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。

块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，以下是块的等价定义：

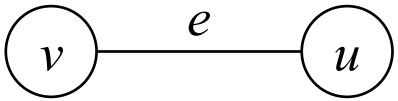
2. 对于任意两个顶点 $u, v \in V$，G 含两条无公共内顶点的 u-v 路。
3. 对于任意两个顶点 $u, v \in V$，G 含圈经过 u 和 v。

块的等价定义

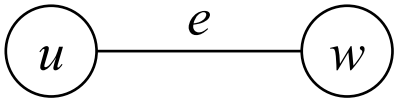
定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，以下是块的等价定义：

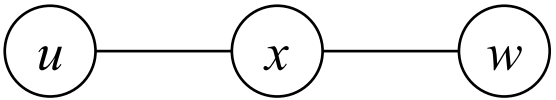
3. 对于任意两个顶点 $u, v \in V$，G 含圈经过 u 和 v。
4. 对于任意一个顶点 $v \in V$ 和任意一条边 $e \in E$，G 含圈经过 v 和 e。

- 若v是e的端点



- 若v不是e的端点：对e剖分





块的等价定义

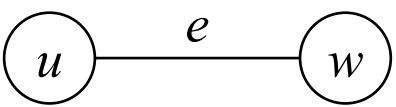
定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，以下是块的等价定义：

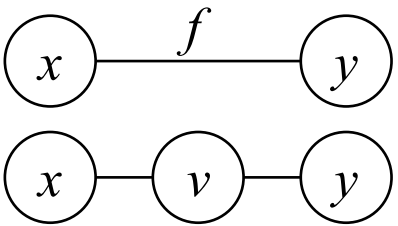
3. 对于任意两个顶点 $u, v \in V$，G 含圈经过 u 和 v。
4. 对于任意一个顶点 $v \in V$ 和任意一条边 $e \in E$，G 含圈经过 v 和 e。

块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，以下是块的等价定义：

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。



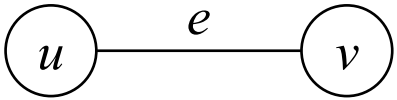


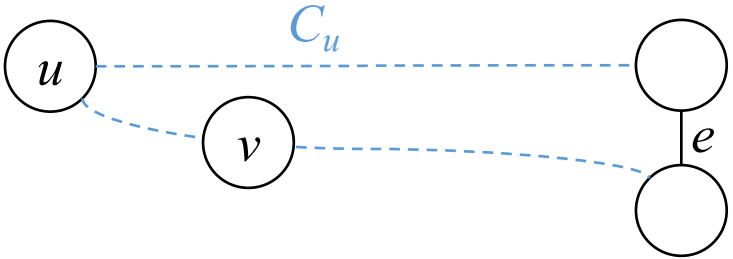
块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，以下是块的等价定义：

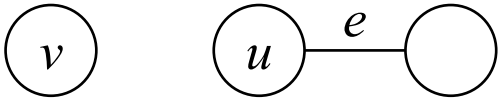
5. 对于任意两条边 $e, f \in E$，G 含圈经过 e 和 f。
6. 对于任意两个顶点 $u, v \in V$ 和任意一条边 $e \in E$，G 含 u-v 路经过 e。

- 若u和v是e的端点

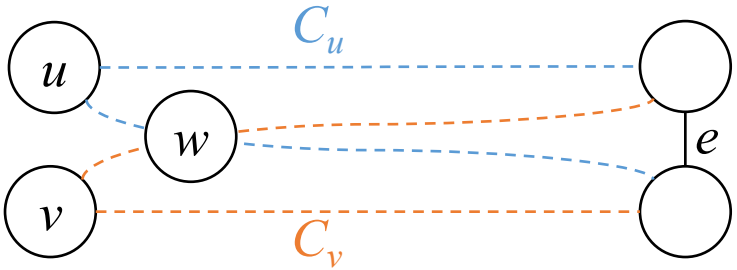




- 若u是e的端点而v不是



- 若u和v不是e的端点
  - 若C_u经过v或C_v经过u
  - 若C_u不经过v且C_v不经过u



块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，以下是块的等价定义：

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。

块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，以下是块的等价定义：

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。

块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，以下是块的等价定义：

1. G 没有割点。

8. 对于任意三个顶点 $u, v, w \in V$，G 含 u-v 路不经过 w。

- 反证法

块的等价定义

定理 4.2 对于阶至少为 3 的连通图 $G = \langle V, E \rangle$，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。

定理 4.1 块为边集定义了一种等价关系。

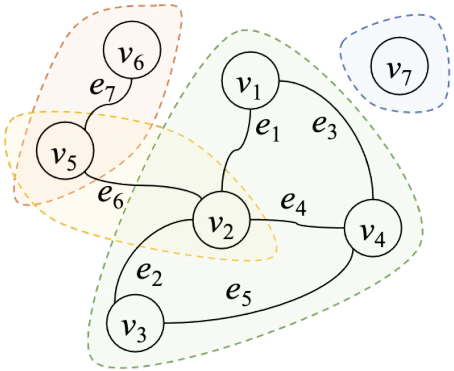
- 这种等价关系的内涵是什么？

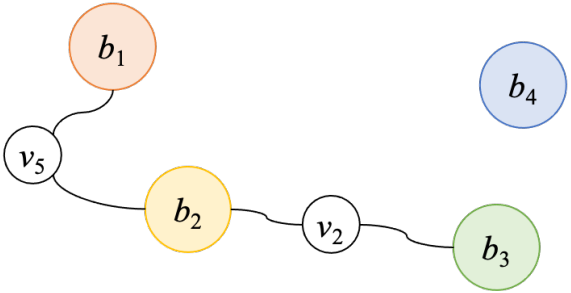
你能解释这些术语吗？

- 块-割点图

思考题 4.12 块-割点图含圈吗？

思考题 4.13 对于图 G 的块-割点图 H，H 的叶顶点有可能是 G 的割点吗？

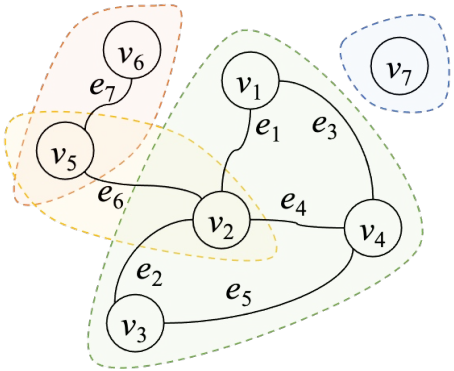


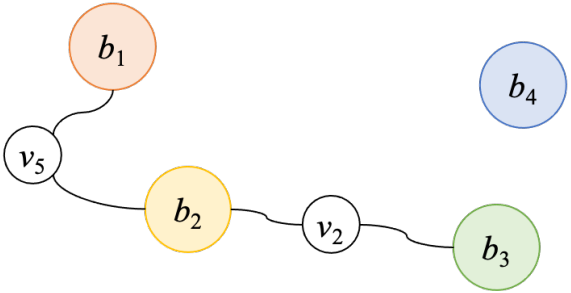


请证明下述定理

定理 4.3 对于图 $G = \langle V, E \rangle$ 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。

- 必要性：反证法
- 充分性





你理解这个算法了吗？

- 这个算法和DFSCV算法的主要区别是什么？

算法 4.1: DFSBlk

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

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

```
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};
```

算法 2.2: DFSCV

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

初值: 变量 *time* 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false，parent 初值为 null，children 初值为 0，isCutVertex 初值为 false

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
foreach (u, v) ∈ E do
    if v.visited = false then
        v.parent ← u;
        u.children ← u.children + 1;
        DFSCV(G, v);
        u.low ← min{u.low, v.low};
        if u.parent = null 且 u.children ≥ 2 then
            u.isCutVertex ← true;
        else if u.parent ≠ null 且 v.low ≥ u.d then
            u.isCutVertex ← true;
    else if v ≠ u.parent then
        u.low ← min{u.low, v.d};
```

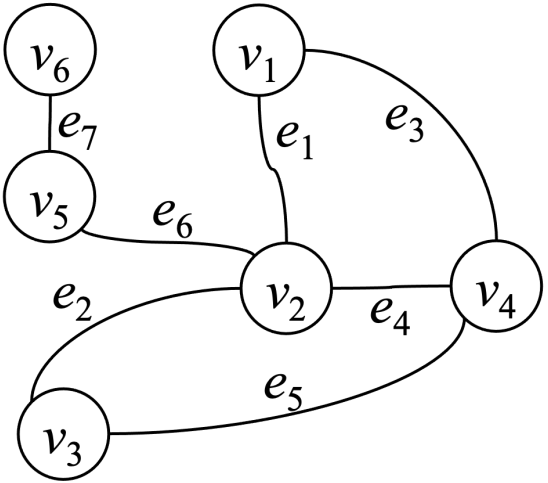
你理解这个算法了吗？

算法 4.1: DFSBlk

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

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

```
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};
```



你理解这个算法了吗？

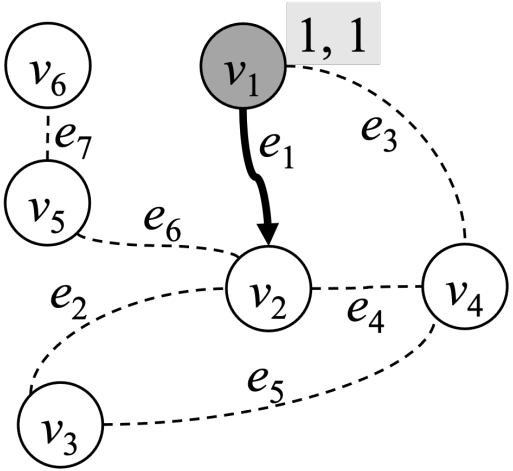
算法 4.1: DFSBlk

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

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

```
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



你理解这个算法了吗？

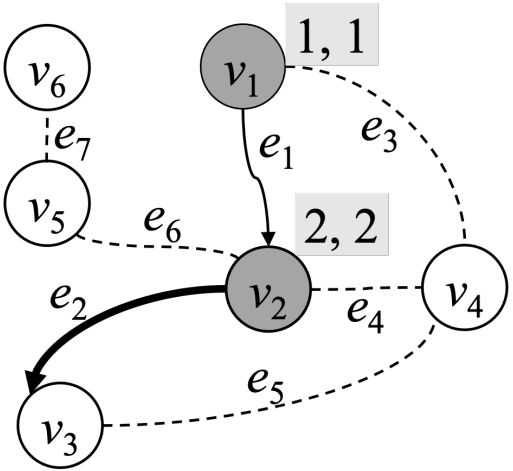
算法 4.1: DFSBlk

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

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

```
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};
```

$e_1 \quad e_2$



你理解这个算法了吗？

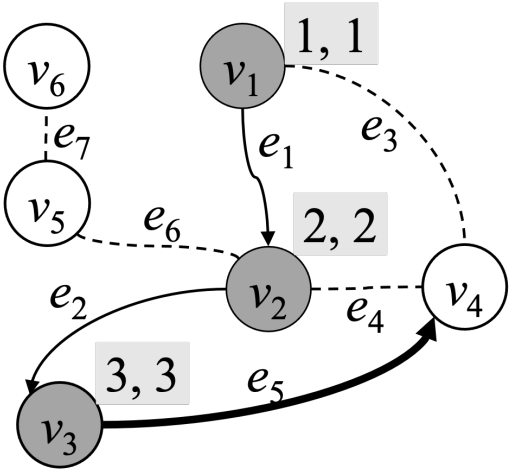
算法 4.1: DFSBlk

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

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

```
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};
```

$e_1 \quad e_2 \quad e_5$



你理解这个算法了吗？

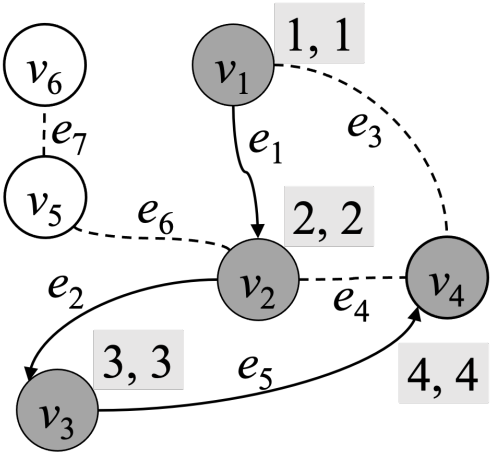
算法 4.1: DFSBlk

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

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

```
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



你理解这个算法了吗？

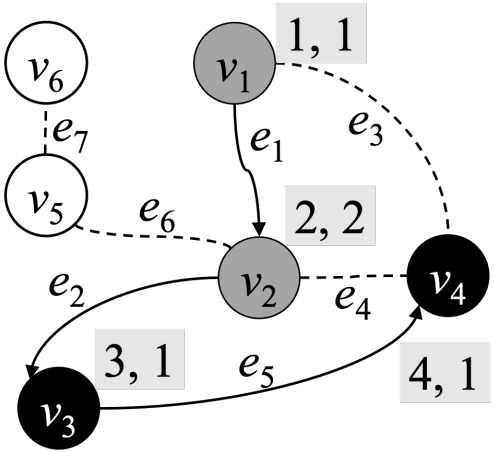
算法 4.1: DFSBlk

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

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

```
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};
```

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



你理解这个算法了吗？

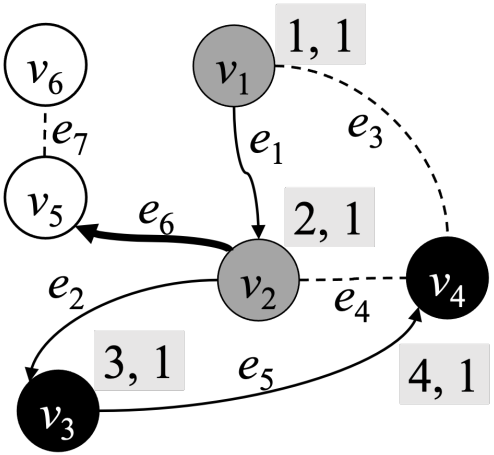
算法 4.1: DFSBlk

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

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

```
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



你理解这个算法了吗？

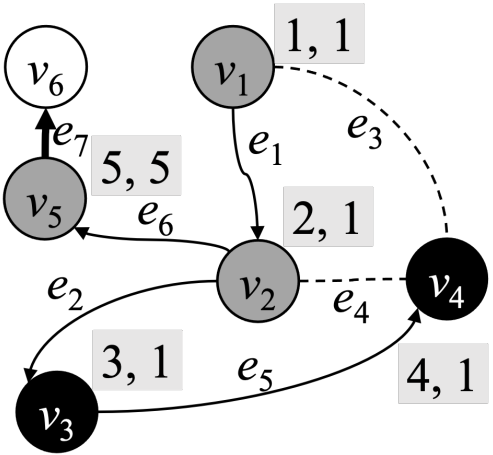
算法 4.1: DFSBlk

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

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

```
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



你理解这个算法了吗？

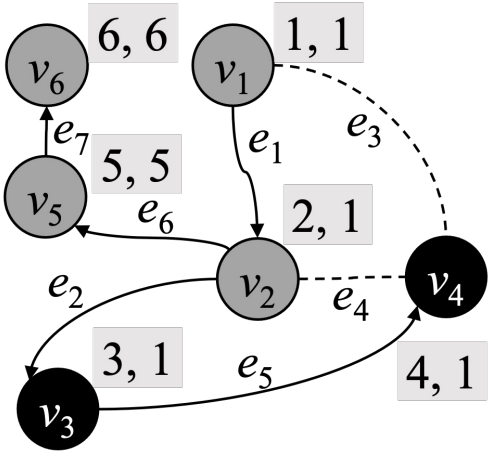
算法 4.1: DFSBlk

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

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

```
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



你理解这个算法了吗？

算法 4.1: DFSBlk

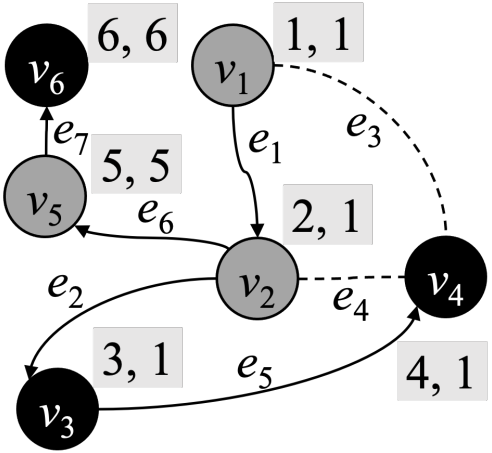
输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

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

```
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



你理解这个算法了吗？

算法 4.1: DFSBlk

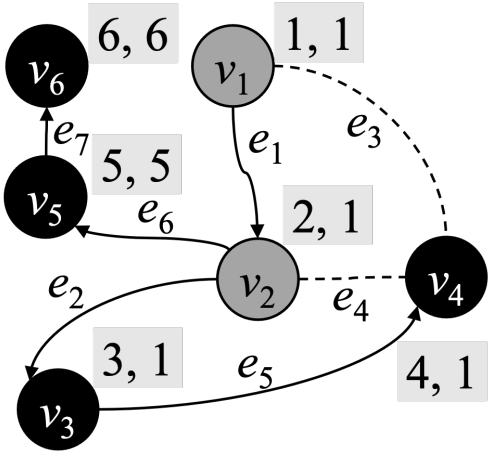
输入: 非平凡连通图 $G = \langle V, E \rangle$，顶点 u

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

```
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_6

e_1	e_2	e_5	e_3	e_4



你理解这个算法了吗？

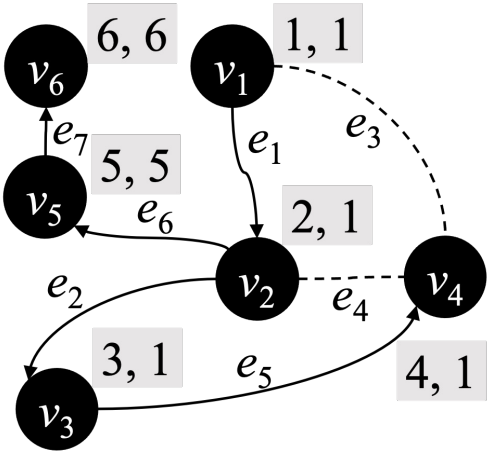
算法 4.1: DFSBlk

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

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

```
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



你理解这个算法了吗？

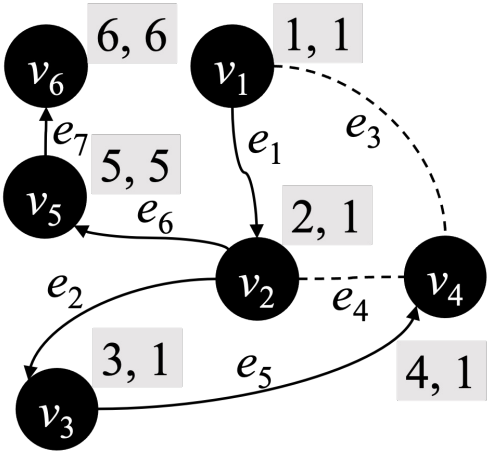
算法 4.1: DFSBlk

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

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

```
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：$e_4\, e_3\, e_5\, e_2\, e_1$



你理解这个算法了吗？

- 什么条件下，算法运行结束时S不为空？

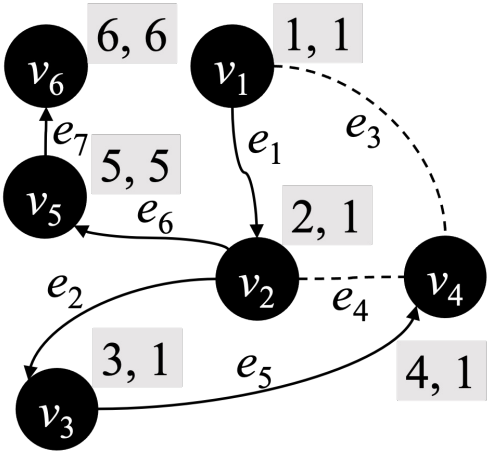
算法 4.1: DFSBlk

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

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

```
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：$e_4\, e_3\, e_5\, e_2\, e_1$



请证明下述定理

- 从u发现(u, v)，到完成对v的递归调用并判定u为割点

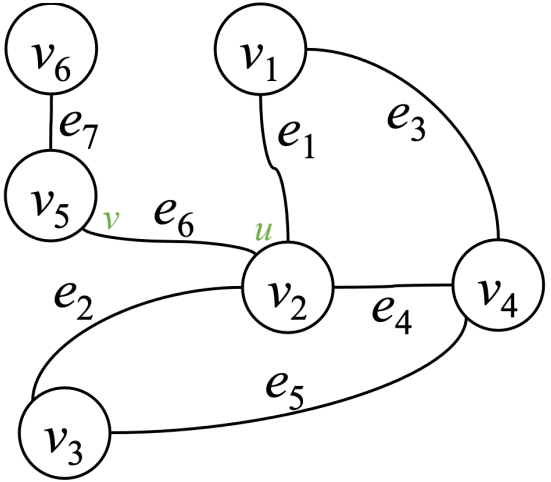
定理 4.4 上述时间段内发现的边恰组成集合 B_T 中的顶点表示的所有块。

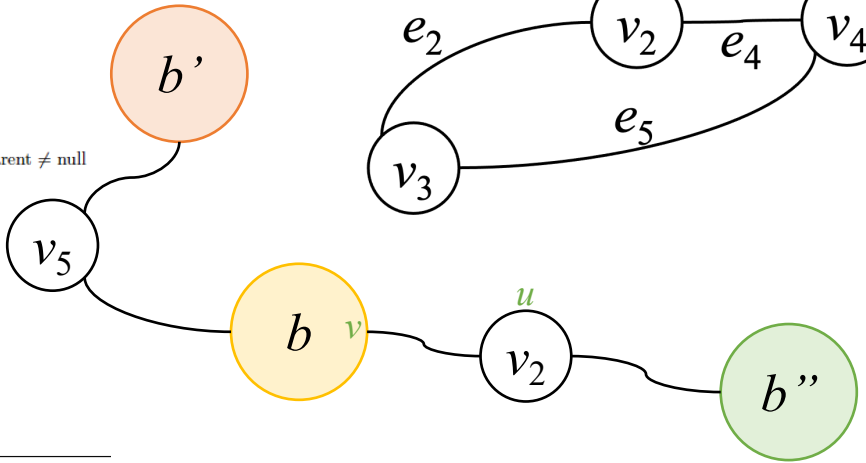
算法 4.1: DFSBlk

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

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

```
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};
```





请证明下述定理

- 从u发现(u, v)，到完成对v的递归调用并判定u为割点

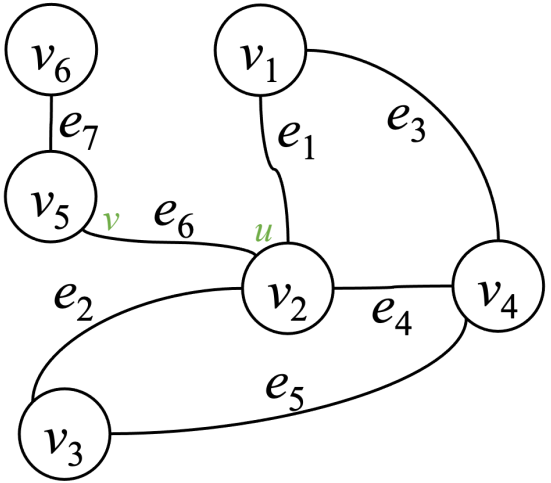
定理 4.5 在上述时间段的结束时刻，边 (u, v) 在栈 S 中，且从栈顶到 (u, v) 恰存储了顶点 b 表示的块的边集。

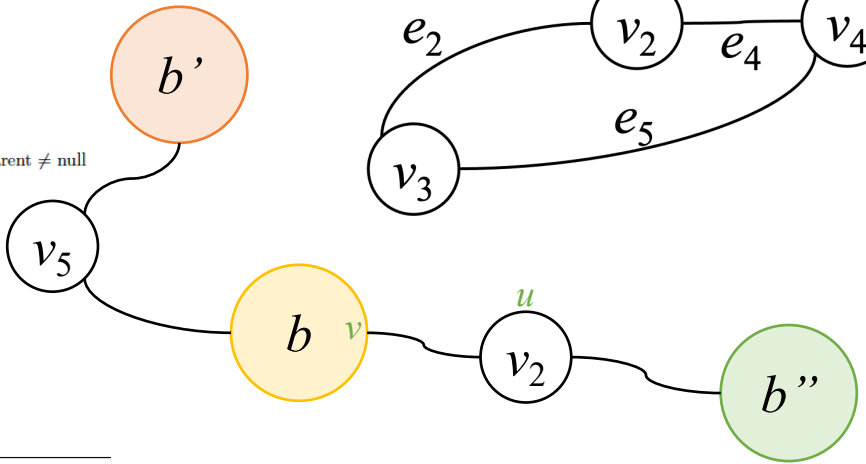
算法 4.1: DFSBlk

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

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

```
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};
```





你能解释这个算法的时间复杂度吗？

- 对于阶为n、边数为m的图，DFSBlk算法的时间复杂度为$O(n+m)$

算法 4.1: DFSBlk

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

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

```
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};
```

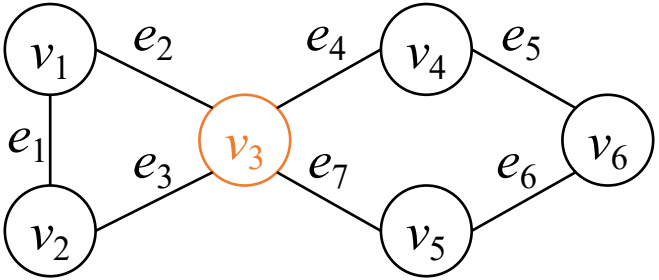
你能解释这些术语吗？

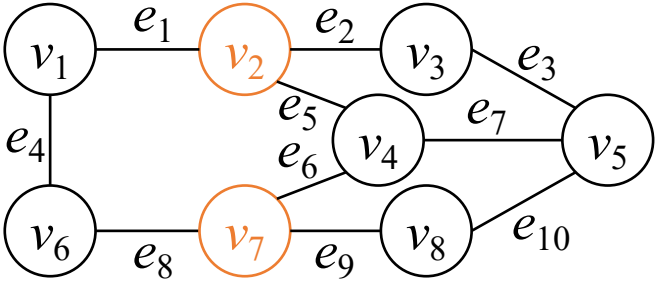
- 点割集（分离集）
- 极小点割集、最小点割集

思考题 4.18 每个图都有最小点割集吗？若有，则唯一吗？

思考题 4.19 完全图的最小点割集是什么？

思考题 4.20 树的最小点割集是什么？





你能解释这些术语吗？

- 点连通度（连通度）
- k点连通图（k连通图）

思考题 4.22 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？

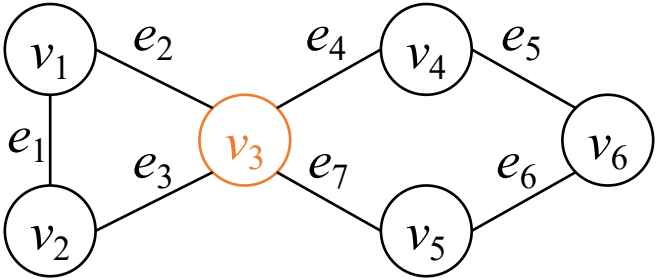
思考题 4.23 完全图 K_n 的连通度是多少？

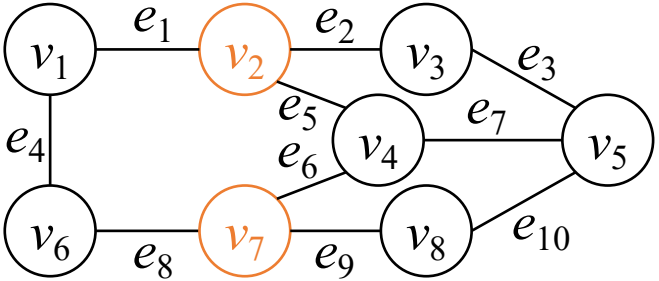
思考题 4.24 不连通图的连通度是多少？

思考题 4.25 ($^\diamond$) 1 连通图一定是连通图吗？反之呢？

思考题 4.26 2 连通图一定是块吗？反之呢？

思考题 4.27 从 k 连通图中删除任意 $k-1$ 个顶点，剩余图一定连通吗？





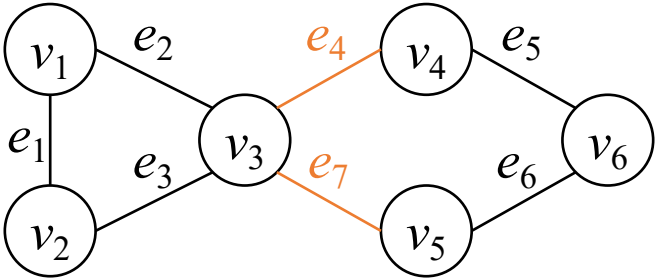
你能解释这些术语吗？

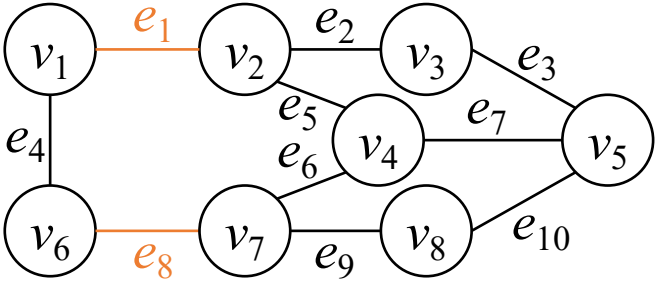
- 边割集
- 极小边割集、最小边割集

思考题 4.29 每个图都有最小边割集吗？若有，则唯一吗？

思考题 4.30 完全图有边割集吗？

思考题 4.31 树的最小边割集是什么？





你能解释这些术语吗？

- 边连通度
- k边连通图

思考题 4.33 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？

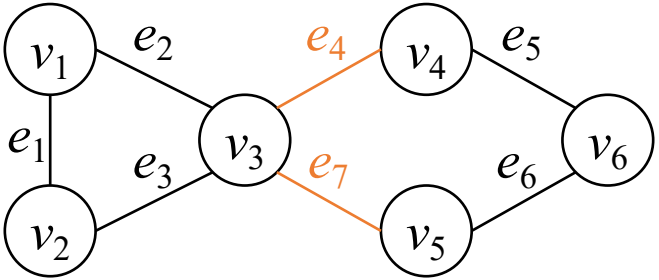
思考题 4.34 完全图 K_n 的边连通度是多少？

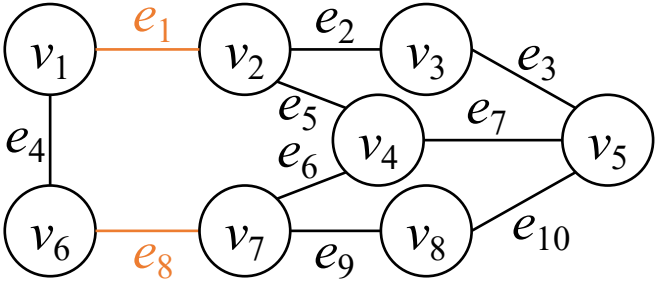
思考题 4.35 不连通图的边连通度是多少？

思考题 4.36 1-边连通图一定是连通图吗？反之呢？

思考题 4.37 2-边连通图一定是块吗？反之呢？

思考题 4.38 从 k 边连通图中删除任意 $k-1$ 条边，剩余图一定连通吗？





你能解释这些术语吗？

- 边连通度
- k边连通图

思考题 4.33 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？

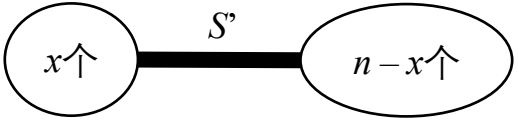
思考题 4.34 完全图 K_n 的边连通度是多少？

思考题 4.35 不连通图的边连通度是多少？

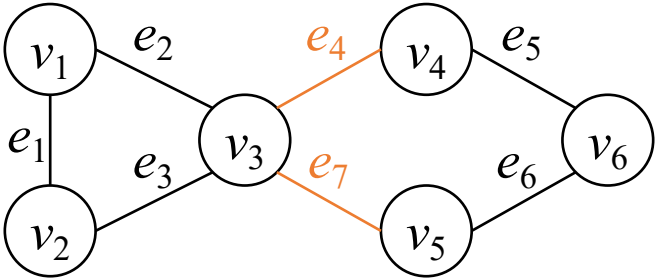
思考题 4.36 1-边连通图一定是连通图吗？反之呢？

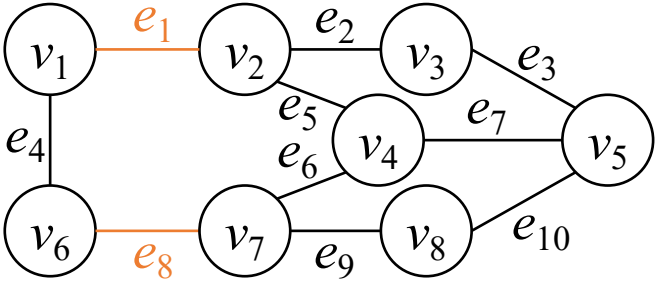
思考题 4.37 2-边连通图一定是块吗？反之呢？

思考题 4.38 从 k 边连通图中删除任意 $k-1$ 条边，剩余图一定连通吗？



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





请证明下述定理

定理 4.6 对于任意一个图 G:

$$\kappa(G) \le \kappa'(G) \le \delta(G).$$

证明. 若图 G 不连通或是平凡图，则 $\kappa(G) = \kappa'(G) = 0 \le \delta(G)$，得证。

若图 G 是完全图 K_n，则 $\kappa(G) = \kappa'(G) = \delta(G) = n-1$，得证。

若图 G 是连通的非完全图，则 G 有点割集和边割集，且 $\delta(G) < \nu(G) - 1$。

先证 $\kappa'(G) \le \delta(G)$。对于图 G 中度最小的顶点 v，$d(v) = \delta(G)$，v 关联的边的集合是 G 的边割集，因此，$\kappa'(G) \le \delta(G)$。

再证 $\kappa(G) \le \kappa'(G)$。对于图 G 的最小边割集 S'，图 $G - S'$ 恰含 2 个连通分支 G_u 和 G_v。接下来证明：G_u 含顶点 u，G_v 含顶点 v，满足 u 和 v 在 G 中不相邻。采用反证法，假设 G_u 中任意一个顶点 u 和 G_v 中任意一个顶点 v 相邻，则

$$\begin{aligned} |S'| &= \nu(G_u) \cdot \nu(G_v) \\ &= \nu(G_u) \cdot (\nu(G) - \nu(G_u)) \\ &\ge \nu(G) - 1. \end{aligned} \tag{B.11}$$

与 $|S'| = \kappa'(G) < \nu(G) - 1$ 矛盾。

请证明下述定理

定理 4.6 对于任意一个图 G:

$$\kappa(G) \le \kappa'(G) \le \delta(G).$$

既然顶点 u 和 v 不相邻，对于最小边割集 S' 中的每条边 e，取 e 的一个端点，如图 B.10所示：若 u 是 e 的端点，则取 e 的另一个端点，该端点必不是 v；若 u 不是 e 的端点，则取 e 在连通分支 G_u 中的端点。这样取得的至多 $|S'|$ 个顶点的集合记作 S，u 和 v 不在 S 中，且在图 $G-S$ 中不连通，因此，S 是图 G 的点割集，$\kappa(G) \le |S| \le |S'| = \kappa'(G)$。

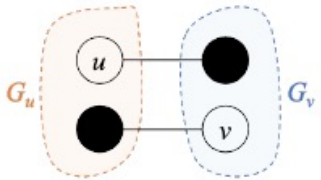
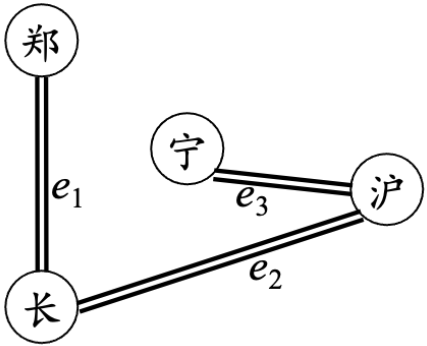


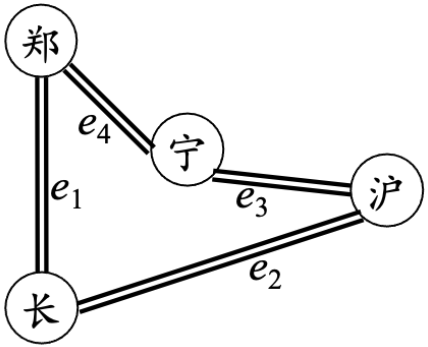
图 B.10 思考题 4.39证明的示意图

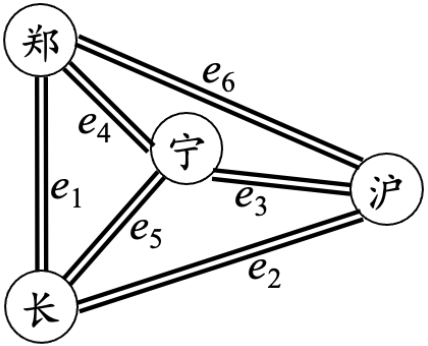
实线—最小边割集 S' 黑色顶点—点割集 S

决定图的点（边）连通度的本质要素是什么？

思考题 4.40 向图中增加边，图的点连通度和边连通度一定提高吗？

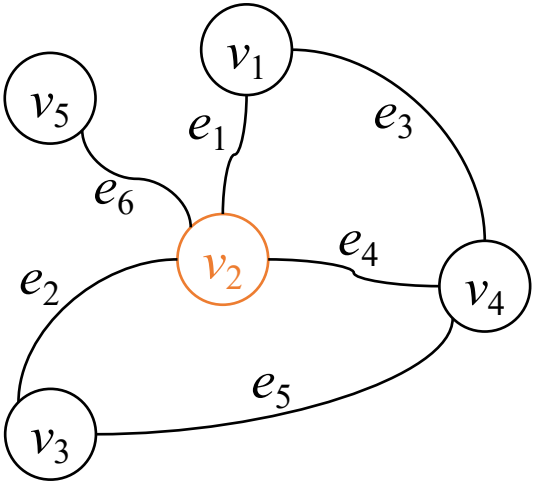






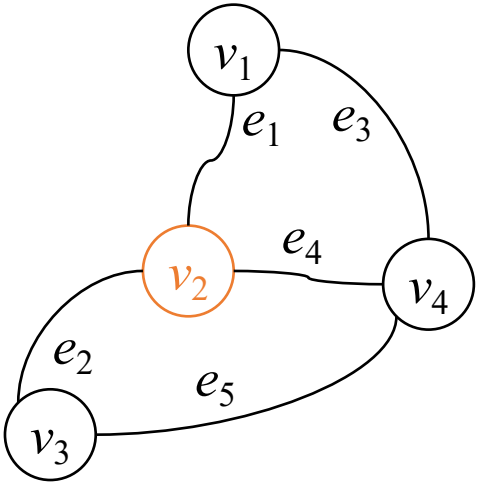
决定图的点（边）连通度的本质要素是什么？

- 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？



决定图的点（边）连通度的本质要素是什么？

- 对于块，删除任意顶点后仍连通的本质原因是什么？



决定图的点（边）连通度的本质要素是什么？

定理 4.7 (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 4.1 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。

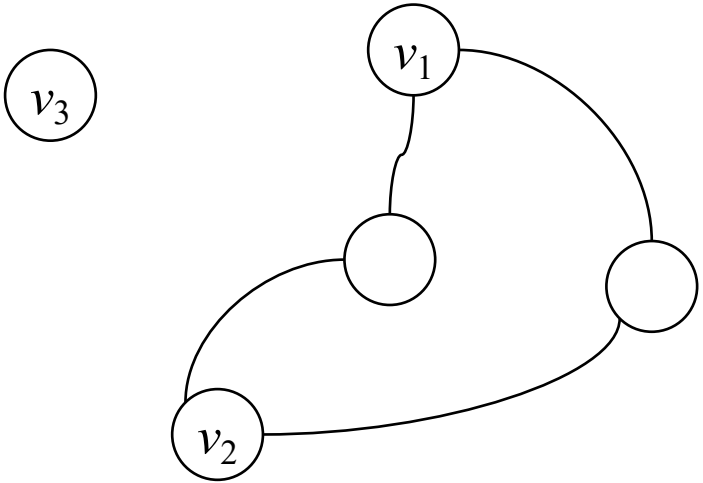
定理 4.8 对于图 $G = \langle V, E \rangle$ 和两个顶点 $u, v \in V$，使 u 和 v 不连通至少需要从 G 中删除的边的数量等于 G 中两两无公共边的 u-v 路的最大数量。

推论 4.2 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 条两两无公共边的路。

请证明下述定理

定理 4.9 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 $v_1, \ldots, v_k$。

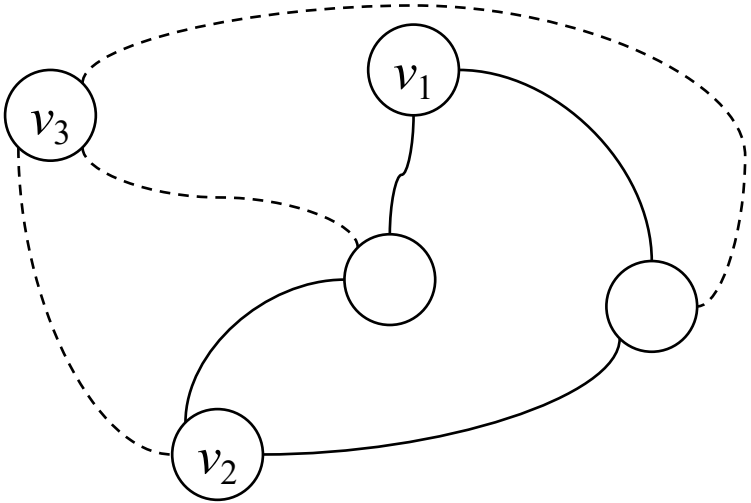
- 归纳法



请证明下述定理

定理 4.9 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 $v_1, \ldots, v_k$。

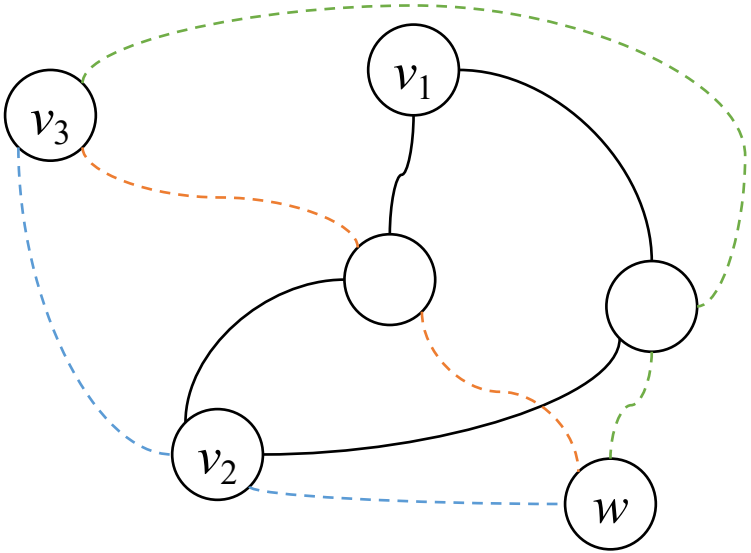
- 归纳法



请证明下述定理

定理 4.9 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 $v_1, \ldots, v_k$。

- 归纳法



OT

- 请调研并阐述面向点连通度或边连通度的门格尔定理的至少2种证明方法。