3-11有向图和伪多项式时间算法

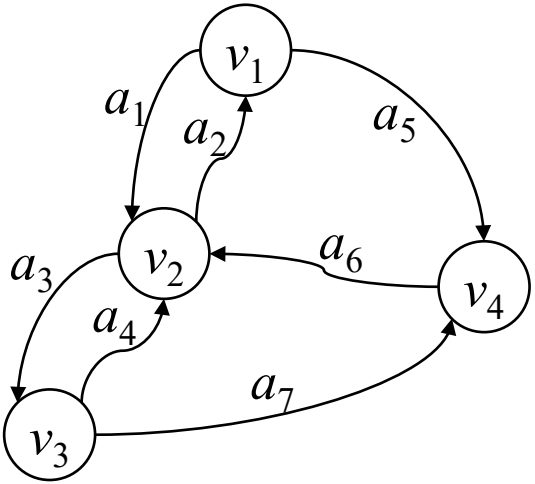
程龚

你能解释这些概念吗？

- 有向图
  - 有向边/弧
- 尾、头、端点
- 关联、入弧、出弧
- （顶点）相邻、入邻点、出邻点
- （边）相邻
- 重弧/平行弧、自环、简单有向图
- 反向弧
- 阶、弧数

思考题 7.1 阶为 n 的简单有向图的弧数的上界是多少?

- 完全有向图



你能解释这些概念吗？

- 入度、出度、度

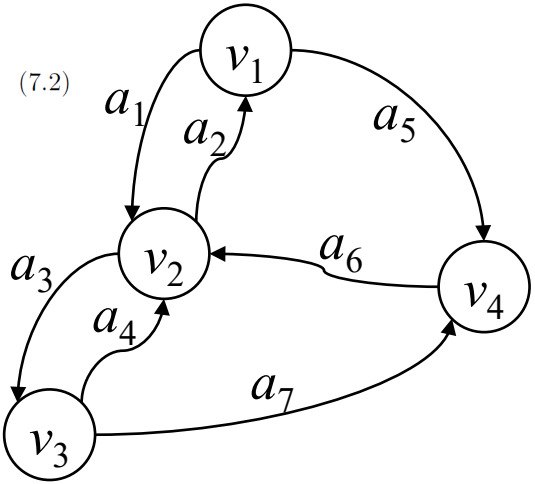
定理 7.1 对于任意一个有向图 $G = \langle V, A \rangle$，顶点集 V 中所有顶点的入度和与出度和都等于 G 的弧数：

$$\sum_{v \in V} d^-(v) = \sum_{v \in V} d^+(v) = \epsilon(G)\,. \tag{7.1}$$

推论 7.1 对于任意一个有向图 $G = \langle V, A \rangle$，顶点集 V 中所有顶点的度的和等于 G 的弧数的 2 倍：

$$\sum_{v \in V} d(v) = 2 \cdot \epsilon(G)\,. \tag{7.2}$$

- 入度序列、最大入度、最小入度
- 出度序列、最大出度、最小出度

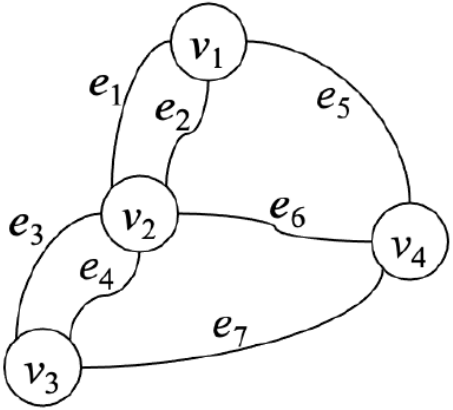


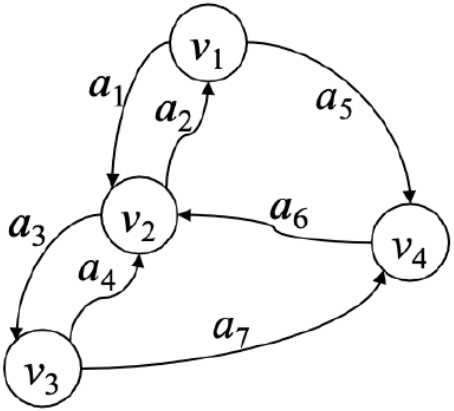
你能解释这些概念吗？

- 底图、定向

思考题 7.4 有向图的底图唯一吗？图的定向唯一吗？

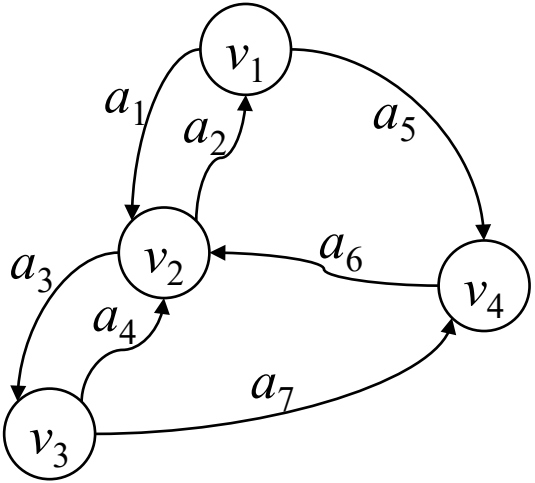
思考题 7.5 阶为 n、边数为 m 的图的定向有多少种？





你能解释这些概念吗？

- 邻接矩阵
  - 第i列元素之和的含义？
  - 第i行元素之和的含义？
- 关联矩阵
  - 第i行元素的绝对值之和的含义？
  - 第j列元素的绝对值之和的含义？
- 邻接表



$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 \end{pmatrix}$$

顶点	出邻点列表
v_1	v_2, v_4
v_2	v_1, v_3
v_3	v_2, v_4
v_4	v_2

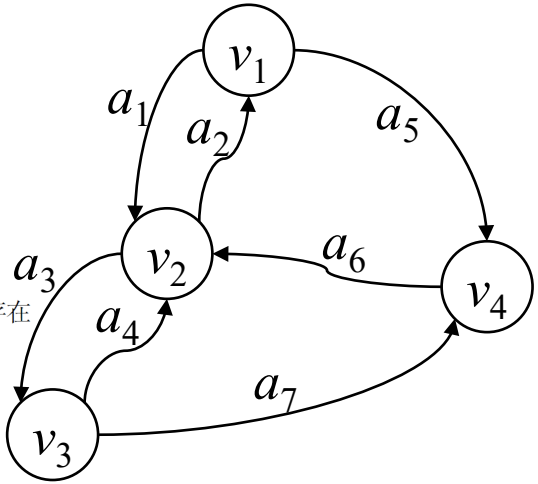
你能解释这些概念吗？

- 有向路线
  - 起点、终点、长度
- 有向(踪)迹
- 有向路(径)

思考题 7.6 若有向图中存在 u-v 有向路线，则一定存在 u-v 有向迹吗？若有向图中存在 u-v 有向迹，则一定存在 u-v 有向路吗？

- 有向闭路线
- 有向闭迹/回路
- 有向圈

思考题 7.8 若图中存在有向闭路线，则一定存在有向闭迹吗？若图中存在有向闭迹，则一定存在有向圈吗？

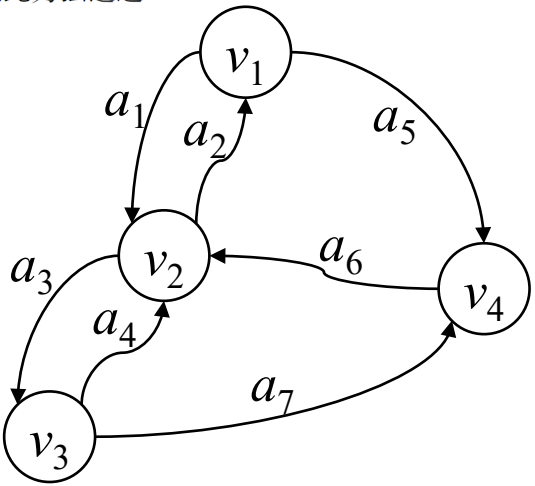


你能解释这些概念吗？

- 弱连通(图)
- 可达、强连通(图)

定理 7.2 对于有向图 $G = \langle V, A \rangle$，G 强连通当且仅当 G 含一条有向闭路线经过顶点集 V 中所有顶点。

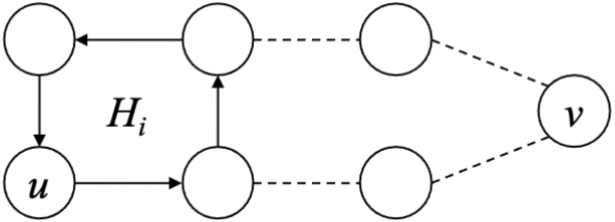
思考题 7.10 强连通图的每条弧都在某个有向圈中吗？你能就此为强连通给出另一个充要条件吗？



请证明下述定理

定理 7.3 (罗宾斯定理) 连通图 G 有强连通定向当且仅当 G 没有割边。

- 必要性
  - 反证法
- 充分性
  - 构造法

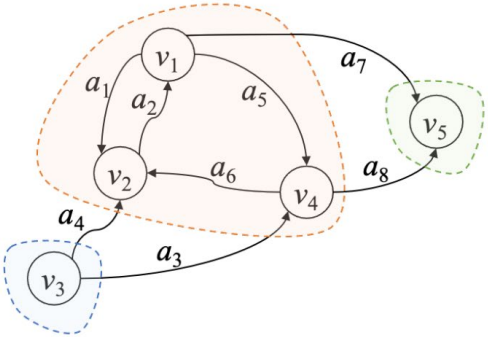


你能解释这些概念吗？

- 弱连通分支
- 强连通分支

思考题 7.13 弱连通图有多少个弱连通分支？强连通图有多少个强连通分支？

思考题 7.14 一个顶点可以出现在图的两个强连通分支中吗？一条弧可以出现在图的两个强连通分支中吗？



你能解释这些概念吗？

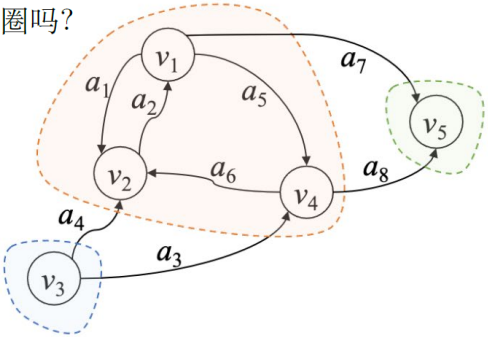
- 弱连通分支
- 强连通分支

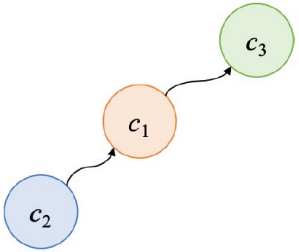
思考题 7.13 弱连通图有多少个弱连通分支？强连通图有多少个强连通分支？

思考题 7.14 一个顶点可以出现在图的两个强连通分支中吗？一条弧可以出现在图的两个强连通分支中吗？

- 浓缩

思考题 7.15 浓缩含圈吗？





你理解这个算法了吗？

算法 7.1: 计算强连通定向的算法

输入: 无割边的连通图 $G = \langle V, E \rangle$

初值: 弧集 A 初值为 $\emptyset$

$s \leftarrow V$ 中任意一个顶点;

DFS(G, s);

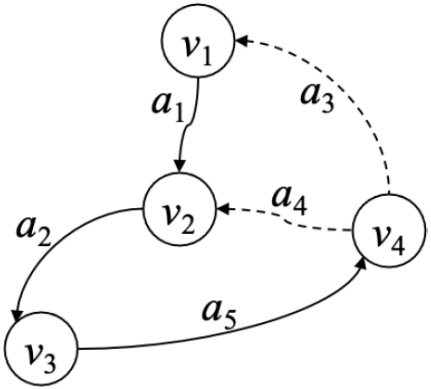
foreach 树边 $e \in E$ **do**

  $A \leftarrow A \cup \{\langle e\text{关联的父顶点}, e\text{关联的子顶点}\rangle\}$;

foreach 后向边 $e \in E$ **do**

  $A \leftarrow A \cup \{\langle e\text{关联的子顶点}, e\text{关联的父顶点}\rangle\}$;

输出（有向图 $\langle V, A \rangle$）;



你理解这个算法了吗？

引理 7.1 算法输出的定向中，从顶点 s 可达其它每个顶点。

引理 7.2 算法输出的定向中，顶点 s 从其它每个顶点都可达。

定理 2.5 若顶点 u 不是根顶点，则 u 是割点当且仅当 u 有子顶点 v 满足：不存在这样一条后向边，其一个端点是 v 或其后代顶点，另一个端点是 u 的祖先顶点。

算法 7.1: 计算强连通定向的算法

输入: 无割边的连通图 $G = \langle V, E \rangle$

初值: 弧集 A 初值为 $\emptyset$

$s \leftarrow V$ 中任意一个顶点;

DFS(G, s);

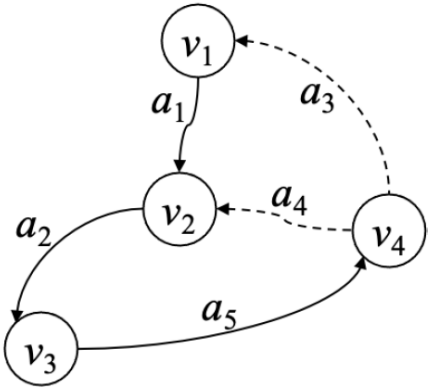
foreach 树边 $e \in E$ **do**

  $A \leftarrow A \cup \{\langle e$关联的父顶点，$e$关联的子顶点$\rangle\}$;

foreach 后向边 $e \in E$ **do**

  $A \leftarrow A \cup \{\langle e$关联的子顶点，$e$关联的父顶点$\rangle\}$;

输出（有向图 $\langle V, A \rangle$）;



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，该算法的时间复杂度为$O(n+m)$。

算法 7.1: 计算强连通定向的算法

输入: 无割边的连通图 $G = \langle V, E \rangle$

初值: 弧集 A 初值为 $\emptyset$

$s \leftarrow V$ 中任意一个顶点;

DFS(G, s);

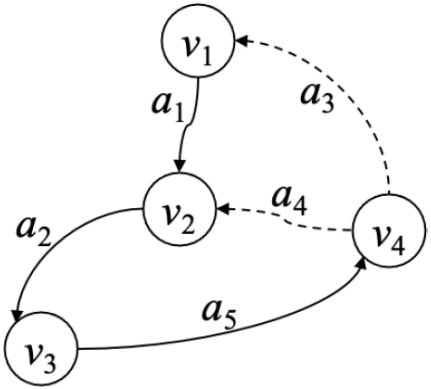
foreach 树边 $e \in E$ **do**

  $A \leftarrow A \cup \{\langle e$关联的父顶点，$e$关联的子顶点$\rangle\}$;

foreach 后向边 $e \in E$ **do**

  $A \leftarrow A \cup \{\langle e$关联的子顶点，$e$关联的父顶点$\rangle\}$;

输出（有向图 $\langle V, A \rangle$);



你理解这个算法了吗？

- S的含义？

算法 7.2: 塔尔真强连通分支算法

输入: 有向图 $G = \langle V, A \rangle$

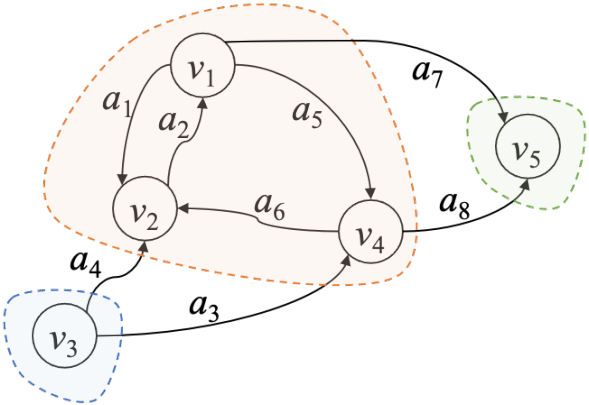
初值: 变量 $time$ 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false；栈 S 初值为空

```
foreach s ∈ V do
    if s.visited = false then
        DFSSCC(G, s);
```

算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈 (S, u);
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出 (以下顶点组成一个强连通分支);
    do
        v ← 出栈 (S);
        输出 (v);
    while v ≠ u;
```



你理解这个算法了吗？

算法 7.2: 塔尔真强连通分支算法

输入: 有向图 $G = \langle V, A \rangle$

初值: 变量 $time$ 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false；栈 S 初值为空

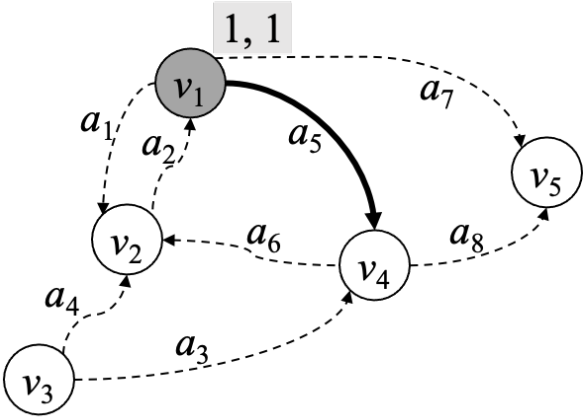
```
foreach s ∈ V do
    if s.visited = false then
        DFSSCC(G, s);
```

算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈 (S, u);
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出 (以下顶点组成一个强连通分支);
    do
        v ← 出栈 (S);
        输出 (v);
    while v ≠ u;
```

v_1



你理解这个算法了吗？

算法 7.2: 塔尔真强连通分支算法

输入: 有向图 $G = \langle V, A \rangle$

初值: 变量 $time$ 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false；栈 S 初值为空

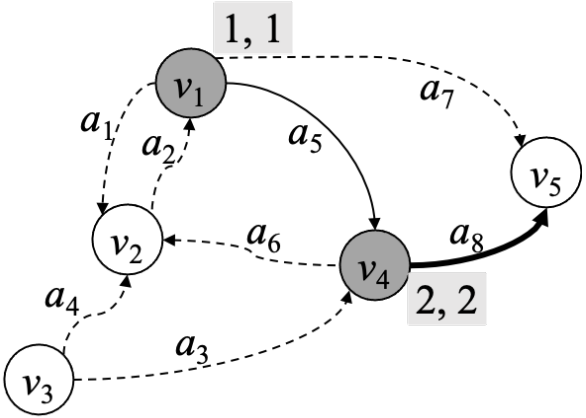
```
foreach s ∈ V do
    if s.visited = false then
        DFSSCC(G, s);
```

算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈 (S, u);
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出 (以下顶点组成一个强连通分支);
    do
        v ← 出栈 (S);
        输出 (v);
    while v ≠ u;
```

$v_1 \quad v_4$



你理解这个算法了吗？

算法 7.2: 塔尔真强连通分支算法

输入: 有向图 $G = \langle V, A \rangle$

初值: 变量 $time$ 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false；栈 S 初值为空

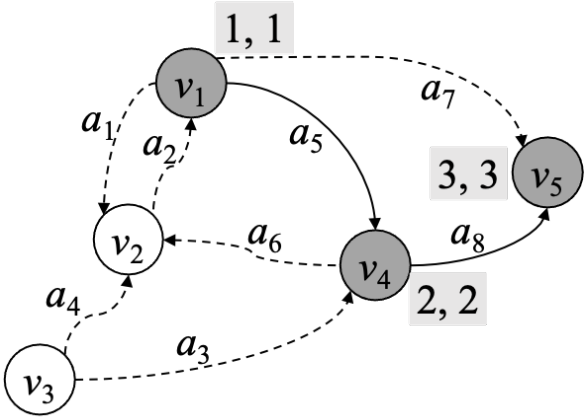
```
foreach s ∈ V do
    if s.visited = false then
        DFSSCC(G, s);
```

算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈 (S, u);
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出 (以下顶点组成一个强连通分支);
    do
        v ← 出栈 (S);
        输出 (v);
    while v ≠ u;
```

$v_1 \quad v_4 \quad v_5$



你理解这个算法了吗？

算法 7.2: 塔尔真强连通分支算法

输入: 有向图 $G = \langle V, A \rangle$

初值: 变量 $time$ 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false；栈 S 初值为空

```
foreach s ∈ V do
    if s.visited = false then
        DFSSCC(G, s);
```

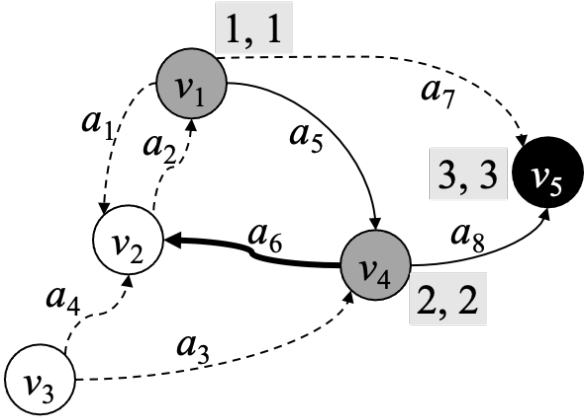
算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈 (S, u);
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出 (以下顶点组成一个强连通分支);
    do
        v ← 出栈 (S);
        输出 (v);
    while v ≠ u;
```

输出 $\{v_5\}$

$v_1 \quad v_4$



你理解这个算法了吗？

算法 7.2: 塔尔真强连通分支算法

输入: 有向图 $G = \langle V, A \rangle$

初值: 变量 $time$ 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false；栈 S 初值为空

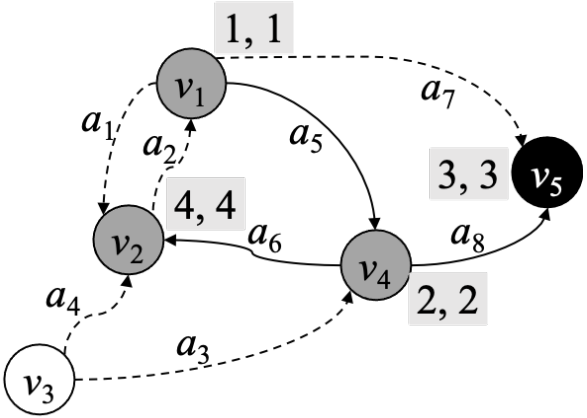
```
foreach s ∈ V do
    if s.visited = false then
        DFSSCC(G, s);
```

算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈 (S, u);
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出 (以下顶点组成一个强连通分支);
    do
        v ← 出栈 (S);
        输出 (v);
    while v ≠ u;
```

$v_1 \quad v_4 \quad v_2$



你理解这个算法了吗？

算法 7.2: 塔尔真强连通分支算法

输入: 有向图 $G = \langle V, A \rangle$

初值: 变量 $time$ 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false；栈 S 初值为空

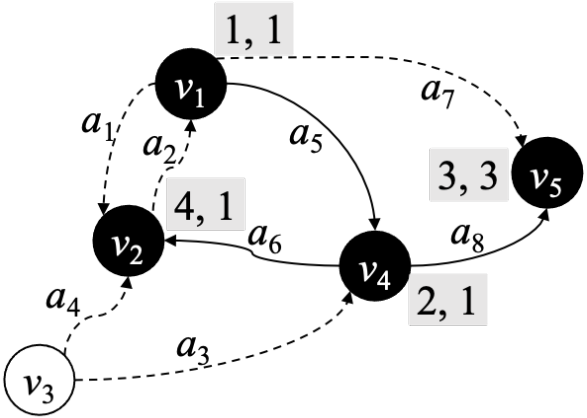
```
foreach s ∈ V do
    if s.visited = false then
        DFSSCC(G, s);
```

算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈 (S, u);
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出 (以下顶点组成一个强连通分支);
    do
        v ← 出栈 (S);
        输出 (v);
    while v ≠ u;
```

输出 $\{v_2, v_4, v_1\}$



你理解这个算法了吗？

算法 7.2: 塔尔真强连通分支算法

输入: 有向图 $G = \langle V, A \rangle$

初值: 变量 $time$ 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false；栈 S 初值为空

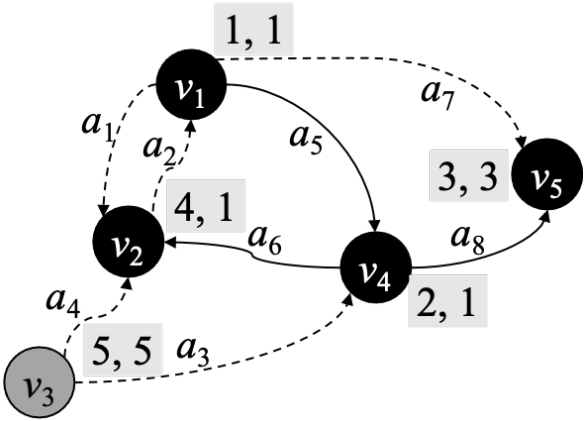
```
foreach s ∈ V do
    if s.visited = false then
        DFSSCC(G, s);
```

算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈 (S, u);
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出 (以下顶点组成一个强连通分支);
    do
        v ← 出栈 (S);
        输出 (v);
    while v ≠ u;
```

v_3



你理解这个算法了吗？

算法 7.2: 塔尔真强连通分支算法

输入: 有向图 $G = \langle V, A \rangle$

初值: 变量 $time$ 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false；栈 S 初值为空

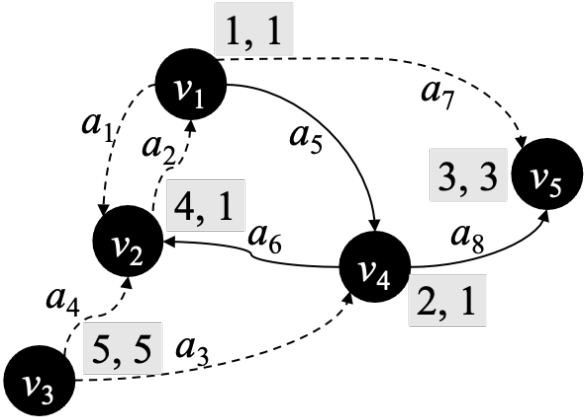
```
foreach s ∈ V do
    if s.visited = false then
        DFSSCC(G, s);
```

算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈 (S, u);
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出 (以下顶点组成一个强连通分支);
    do
        v ← 出栈 (S);
        输出 (v);
    while v ≠ u;
```

输出 $\{v_3\}$



你理解这个算法了吗？

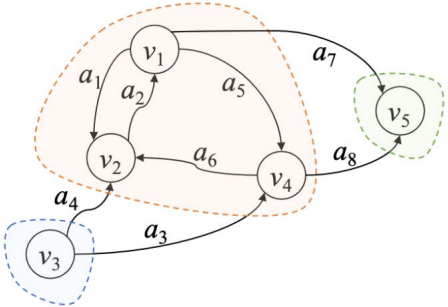
- 从 访问顶点u并将其增加到栈S中（u：所在强连通分支中被访问的首个顶点）
- 到 DFSSCC算法判定u的low属性值和d属性值是否相等
  - $H = <C, A'>$：浓缩
  - $c \in C$：含u的强连通分支
  - C'：H中从c可达的顶点的集合

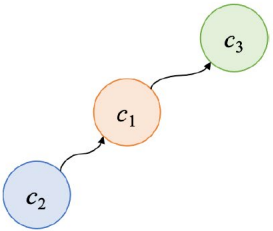
算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈（S, u）;
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出（以下顶点组成一个强连通分支）;
    do
        v ← 出栈（S）;
        输出（v）;
    while v ≠ u;
```

定理 7.4 上述时间段内访问的顶点恰组成集合 C' 中的顶点表示的所有强连通分支的顶点集的并集。





你理解这个算法了吗？

- 从 访问顶点u并将其增加到栈S中（u：所在强连通分支中被访问的首个顶点）
- 到 DFSSCC算法判定u的low属性值和d属性值是否相等
  - $H = <C, A'>$：浓缩
  - $c \in C$：含u的强连通分支
  - C'：H中从c可达的顶点的集合

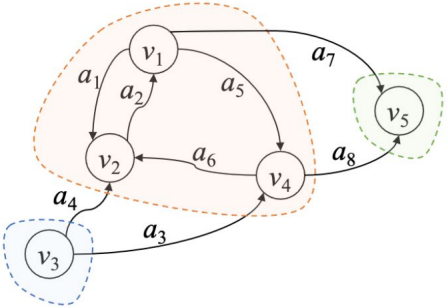
算法 7.3: DFSSCC

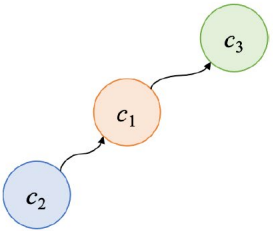
输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈（S, u）;
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出（以下顶点组成一个强连通分支）;
    do
        v ← 出栈（S）;
        输出（v）;
    while v ≠ u;
```

定理 7.4 上述时间段内访问的顶点恰组成集合 C' 中的顶点表示的所有强连通分支的顶点集的并集。

在DFS树中，u的后代顶点恰由这些强连通分支中的顶点（除u外）组成





你理解这个算法了吗？

- 从 访问顶点u并将其增加到栈S中（u：所在强连通分支中被访问的首个顶点）
- 到 DFSSCC算法判定u的low属性值和d属性值是否相等
  - $H = <C, A'>$：浓缩
  - $c \in C$：含u的强连通分支
  - C'：H中从c可达的顶点的集合

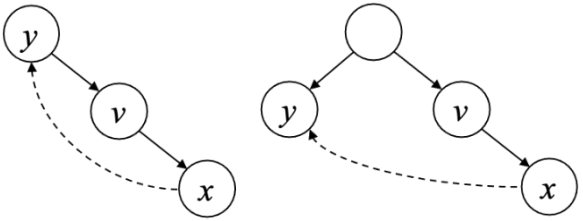
算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈（S, u）;
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出（以下顶点组成一个强连通分支）;
    do
        v ← 出栈（S）;
        输出（v）;
    while v ≠ u;
```

引理 7.3 在上述时间段的结束时刻，顶点 u 的 low 属性值和 d 属性值相等，顶点 c 表示的强连通分支中除 u 外的每个顶点的 low 属性值都小于 d 属性值。

$$v.\text{low} \le x.\text{low} \le y.\text{d} < v.\text{d}\,.$$



你理解这个算法了吗？

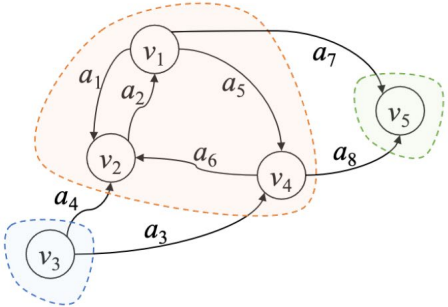
- 从 访问顶点u并将其增加到栈S中（u：所在强连通分支中被访问的首个顶点）
- 到 DFSSCC算法判定u的low属性值和d属性值是否相等
  - $H = <C, A'>$：浓缩
  - $c \in C$：含u的强连通分支
  - C'：H中从c可达的顶点的集合

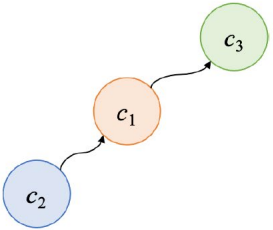
算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈（S, u）;
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出（以下顶点组成一个强连通分支）;
    do
        v ← 出栈（S）;
        输出（v）;
    while v ≠ u;
```

定理 7.5 在上述时间段的结束时刻，顶点 u 在栈 S 中，且从栈顶到 u 恰存储了顶点 c 表示的强连通分支的顶点集。





你能解释这个算法的时间复杂度吗？

- 对于阶为n、弧数为m的有向图，塔尔真强连通分支算法的时间复杂度为$O(n+m)$。

算法 7.2: 塔尔真强连通分支算法

输入: 有向图 $G = \langle V, A \rangle$

初值: 变量 $time$ 初值为 0；顶点集 V 中所有顶点的 visited 初值为 false；栈 S 初值为空

```
foreach s ∈ V do
    if s.visited = false then
        DFSSCC(G, s);
```

算法 7.3: DFSSCC

输入: 有向图 $G = \langle V, A \rangle$，顶点 u

```
time ← time + 1;
u.d ← time;
u.low ← u.d;
u.visited ← true;
入栈（S, u）;
foreach ⟨u, v⟩ ∈ A do
    if v.visited = false then
        DFSSCC(G, v);
        u.low ← min{u.low, v.low};
    else if v ∈ S then
        u.low ← min{u.low, v.d};
if u.low = u.d then
    输出（以下顶点组成一个强连通分支）;
    do
        v ← 出栈（S）;
        输出（v）;
    while v ≠ u;
```

你能解释这些概念吗？

- 最短有向路
- 距离

思考题 7.20 在有向图中，距离满足对称性吗？满足三角不等式吗？

你能解释这些概念吗？

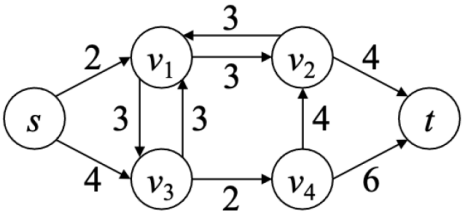
- 流网络<V, A, c, s, t>
  - c：容量函数
  - s/t：源/汇
- 流
  - （弧的）流量
  - 零流
- 可行流
  - 容量约束
  - 守恒约束

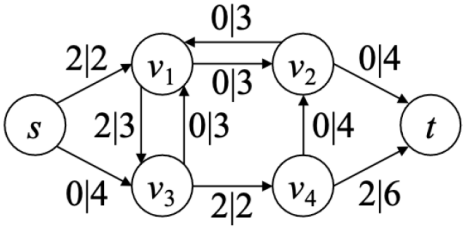
思考题 7.21 每个流网络都有可行流吗？

思考题 7.22 对于可行流 f，请比较源的净流出量 $f^+(s) - f^-(s)$ 和汇的净流入量 $f^-(t) - f^+(t)$。

- （流的）值
  - 最大流

思考题 7.23 流网络一定有最大流吗？若有，则唯一吗？



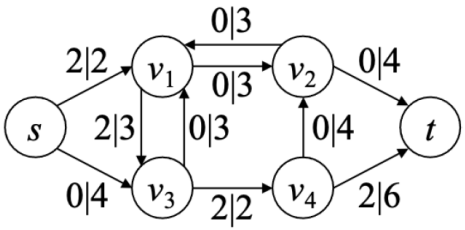


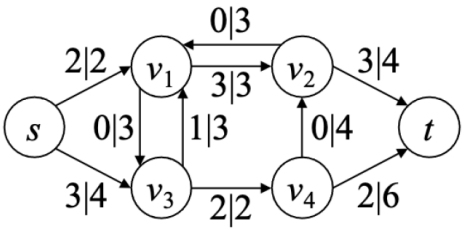
你能解释这些概念吗？

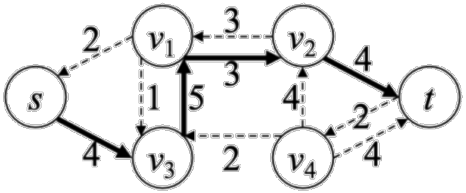
- 剩余容量
- 剩余网络

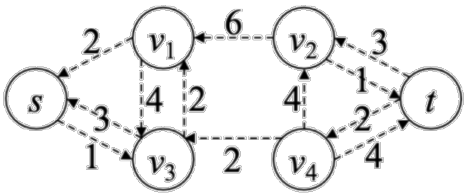
思考题 7.25 流网络 G 的弧集 A 和剩余网络 G_f 的弧集 A_f 相同吗?

思考题 7.26 请比较弧集 A 的容量和与弧集 A_f 的权和。







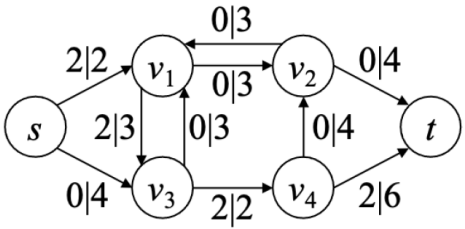


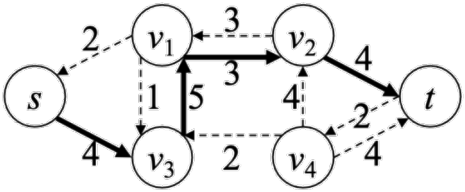
你能解释这些概念吗？

- f增广路
- 可增量

思考题 7.28 每个剩余网络都有增广路吗？若有，则唯一吗？

思考题 7.29 如何利用增广路得到一个值更大的流？

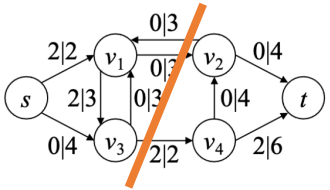




请证明下述定理

引理 7.4 对于流网络 G、可行流 f、顶点集 V 的划分 S 和 T，所有尾在顶点子集 S 中、头在顶点子集 T 中的弧的流量和记作 $f^+(S)$，所有头在 S 中、尾在 T 中的弧的流量和记作 $f^-(S)$，则

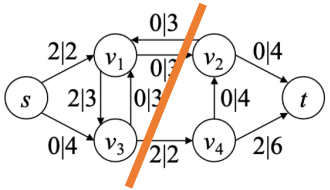
$$f^+(s) - f^-(s) = \sum_{v \in S} (f^+(v) - f^-(v)) = f^+(S) - f^-(S). \tag{7.12}$$



请证明下述定理

引理 7.4 对于流网络 G、可行流 f、顶点集 V 的划分 S 和 T，所有尾在顶点子集 S 中、头在顶点子集 T 中的弧的流量和记作 $f^+(S)$，所有头在 S 中、尾在 T 中的弧的流量和记作 $f^-(S)$，则

$$f^+(s) - f^-(s) = \sum_{v \in S} (f^+(v) - f^-(v)) = f^+(S) - f^-(S)\,. \tag{7.12}$$



$$\begin{aligned} val(f) &= f^+(s) - f^-(s) \\ &= f^+(S) - f^-(S) \\ &\le f^+(S) \\ &\le c(S,T)\,. \end{aligned}$$

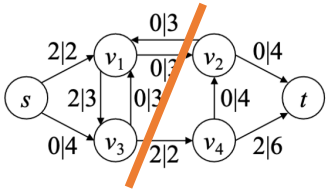
■ 源汇割、容量

推论 7.2 对于流网络 G、可行流 f、源汇割 $C_{S,T}$，$val(f) \le c(S,T)$。

请证明下述定理

引理 7.4 对于流网络 G、可行流 f、顶点集 V 的划分 S 和 T，所有尾在顶点子集 S 中、头在顶点子集 T 中的弧的流量和记作 $f^+(S)$，所有头在 S 中、尾在 T 中的弧的流量和记作 $f^-(S)$，则

$$f^+(s) - f^-(s) = \sum_{v \in S} (f^+(v) - f^-(v)) = f^+(S) - f^-(S)\,. \tag{7.12}$$



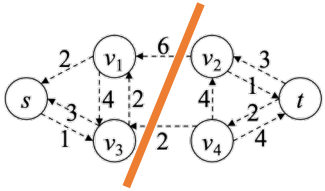
$$\begin{aligned} val(f) &= f^+(s) - f^-(s) \\ &= f^+(S) - f^-(S) \\ &\le f^+(S) \\ &\le c(S,T)\,. \end{aligned}$$

■ 源汇割、容量

推论 7.2 对于流网络 G、可行流 f、源汇割 $C_{S,T}$，$val(f) \le c(S,T)$。

定理 7.6 对于流网络 G 和可行流 f，f 是最大流当且仅当剩余网络 G_f 不含 f 增广路。

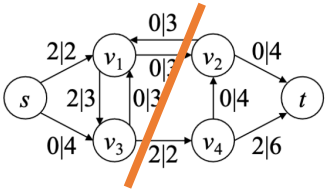
- 充分性：$f^+(S)$=?，$f^-(S)$=?



请证明下述定理

引理 7.4 对于流网络 G、可行流 f、顶点集 V 的划分 S 和 T，所有尾在顶点子集 S 中、头在顶点子集 T 中的弧的流量和记作 $f^+(S)$，所有头在 S 中、尾在 T 中的弧的流量和记作 $f^-(S)$，则

$$f^+(s) - f^-(s) = \sum_{v \in S} (f^+(v) - f^-(v)) = f^+(S) - f^-(S)\,. \tag{7.12}$$



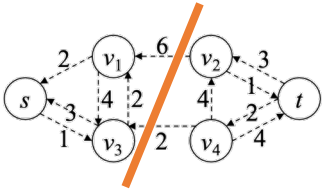
$$\begin{aligned} val(f) &= f^+(s) - f^-(s) \\ &= f^+(S) - f^-(S) \\ &\le f^+(S) \\ &\le c(S,T)\,. \end{aligned}$$

■ 源汇割、容量

推论 7.2 对于流网络 G、可行流 f、源汇割 $C_{S,T}$，$val(f) \le c(S,T)$。

定理 7.6 对于流网络 G 和可行流 f，f 是最大流当且仅当剩余网络 G_f 不含 f 增广路。

- 充分性：$f^+(S)$=?，$f^-(S)$=?



定理 7.7 (最大流最小割定理) 对于任意一个流网络，最大流的值等于所有源汇割的容量的最小值。

- 为什么 $val(f_{\max}) \le c_{\min}$？
- 为什么 $val(f_{\max}) \ge c_{\min}$？

你理解这个算法了吗？

- r的含义？
- r’的含义？

算法 7.4: 福特-法尔克森算法

输入: 流网络 $G = \langle V, A, c, s, t \rangle$

初值: 流 f 初值为零流

```
while G_f 含 f 增广路 P do
    r ← min_{⟨u,v⟩∈P经过的所有弧} r(u, v);
    foreach ⟨u, v⟩ ∈ P经过的所有弧 do
        if ⟨v, u⟩ ∈ A then
            r′ ← min{f(⟨v, u⟩), r};
            f(⟨v, u⟩) ← f(⟨v, u⟩) − r′;
            r ← r − r′;
        if ⟨u, v⟩ ∈ A then
            f(⟨u, v⟩) ← f(⟨u, v⟩) + r;
输出 (f);
```

你理解这个算法了吗？

算法 7.4: 福特-法尔克森算法

输入: 流网络 $G = \langle V, A, c, s, t \rangle$

初值: 流 f 初值为零流

while G_f 含 f 增广路 P **do**

  $r \leftarrow \min\limits_{\langle u,v \rangle \in P\text{经过的所有弧}} r(u,v)$;

  foreach $\langle u,v \rangle \in P$经过的所有弧 **do**

    if $\langle v,u \rangle \in A$ **then**

      $r' \leftarrow \min\{f(\langle v,u \rangle), r\}$;

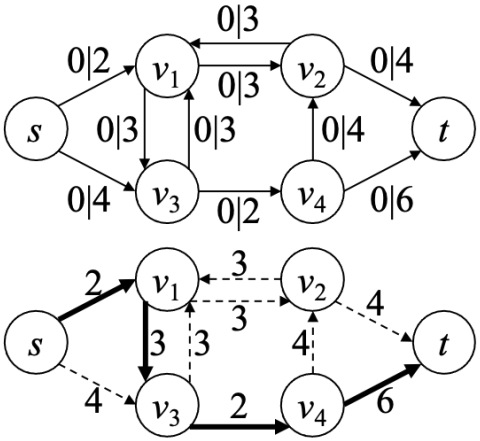
      $f(\langle v,u \rangle) \leftarrow f(\langle v,u \rangle) - r'$;

      $r \leftarrow r - r'$;

    if $\langle u,v \rangle \in A$ **then**

      $f(\langle u,v \rangle) \leftarrow f(\langle u,v \rangle) + r$;

输出 (f);



你理解这个算法了吗？

算法 7.4: 福特-法尔克森算法

输入: 流网络 $G = \langle V, A, c, s, t \rangle$

初值: 流 f 初值为零流

while G_f 含 f 增广路 P **do**

  $r \leftarrow \min\limits_{\langle u,v\rangle \in P\text{经过的所有弧}} r(u,v)$;

  foreach $\langle u,v\rangle \in P$经过的所有弧 **do**

    if $\langle v,u\rangle \in A$ **then**

      $r' \leftarrow \min\{f(\langle v,u\rangle), r\}$;

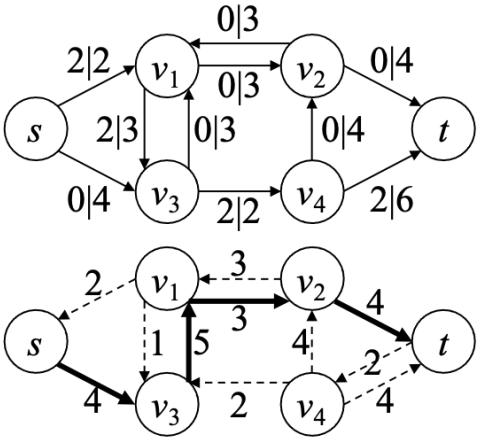
      $f(\langle v,u\rangle) \leftarrow f(\langle v,u\rangle) - r'$;

      $r \leftarrow r - r'$;

    if $\langle u,v\rangle \in A$ **then**

      $f(\langle u,v\rangle) \leftarrow f(\langle u,v\rangle) + r$;

输出 (f);



你理解这个算法了吗？

算法 7.4: 福特-法尔克森算法

输入: 流网络 $G = \langle V, A, c, s, t \rangle$

初值: 流 f 初值为零流

while G_f 含 f 增广路 P **do**

  $r \leftarrow \min\limits_{\langle u,v\rangle \in P\text{经过的所有弧}} r(u,v)$;

  foreach $\langle u,v\rangle \in P$经过的所有弧 **do**

    if $\langle v,u\rangle \in A$ **then**

      $r' \leftarrow \min\{f(\langle v,u\rangle), r\}$;

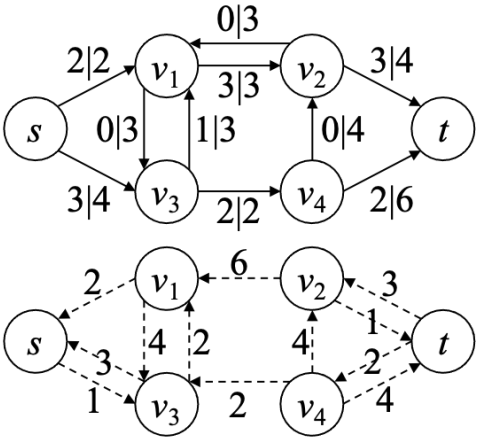
      $f(\langle v,u\rangle) \leftarrow f(\langle v,u\rangle) - r'$;

      $r \leftarrow r - r'$;

    if $\langle u,v\rangle \in A$ **then**

      $f(\langle u,v\rangle) \leftarrow f(\langle u,v\rangle) + r$;

输出 (f);



你理解这个算法了吗？

- 当G中弧的容量包含无理数时，算法有可能无法运行结束

思考题 7.34 ($^\diamond$) 当流网络 G 中所有弧的容量都是有理数时，为什么福特-法尔克森算法一定会运行结束？

算法 7.4: 福特-法尔克森算法

输入: 流网络 $G = \langle V, A, c, s, t\rangle$

初值: 流 f 初值为零流

```
while G_f 含 f 增广路 P do
    r ← min_{⟨u,v⟩∈P经过的所有弧} r(u, v);
    foreach ⟨u, v⟩ ∈ P经过的所有弧 do
        if ⟨v, u⟩ ∈ A then
            r' ← min{f(⟨v, u⟩), r};
            f(⟨v, u⟩) ← f(⟨v, u⟩) − r';
            r ← r − r';
        if ⟨u, v⟩ ∈ A then
            f(⟨u, v⟩) ← f(⟨u, v⟩) + r;
输出 (f);
```

你能解释这个算法的时间复杂度吗？

- 对于阶为n、弧数为m的流网络G，福特-法尔克森算法的时间复杂度为$O((n+m)\mathcal{V})$。

算法 7.4: 福特-法尔克森算法

输入: 流网络 $G = \langle V, A, c, s, t \rangle$

初值: 流 f 初值为零流

```
while G_f 含 f 增广路 P do
    r ← min_{⟨u,v⟩∈P经过的所有弧} r(u, v);
    foreach ⟨u, v⟩ ∈ P经过的所有弧 do
        if ⟨v, u⟩ ∈ A then
            r' ← min{f(⟨v, u⟩), r};
            f(⟨v, u⟩) ← f(⟨v, u⟩) − r';
            r ← r − r';
        if ⟨u, v⟩ ∈ A then
            f(⟨u, v⟩) ← f(⟨u, v⟩) + r;
输出 (f);
```

你能解释这个算法的时间复杂度吗？

- 对于阶为n、弧数为m的流网络G，福特-法尔克森算法的时间复杂度为$O((n+m)\mathcal{V})$。
- 这是多项式时间算法吗？

算法 7.4: 福特-法尔克森算法

输入: 流网络 $G = \langle V, A, c, s, t \rangle$

初值: 流 f 初值为零流

while G_f 含 f 增广路 P **do**
  $r \leftarrow \min\limits_{\langle u,v\rangle \in P\text{经过的所有弧}} r(u,v)$;
  foreach $\langle u,v\rangle \in P$经过的所有弧 **do**
    if $\langle v,u\rangle \in A$ **then**
      $r' \leftarrow \min\{f(\langle v,u\rangle), r\}$;
      $f(\langle v,u\rangle) \leftarrow f(\langle v,u\rangle) - r'$;
      $r \leftarrow r - r'$;
    if $\langle u,v\rangle \in A$ **then**
      $f(\langle u,v\rangle) \leftarrow f(\langle u,v\rangle) + r$;
输出 (f);

你能解释这些概念吗？

- Max-Int
- Pseudo-polynomial-time algorithm
- h-value-bounded subproblem of U

你能解释这些概念吗？

- Max-Int
- Pseudo-polynomial-time algorithm
- h-value-bounded subproblem of U

Definition 3.2.1.1. *Let U be an integer-valued problem, and let A be an algorithm that solves U. We say that A is a* **pseudo-polynomial-time algorithm for U** *if there exists a polynomial p of two variables such that*

$$Time_A(x) = O(p(|x|, Max\text{-}Int(x)))$$

for every instance x of U.

你能解释这些概念吗？

- Max-Int
- Pseudo-polynomial-time algorithm
- h-value-bounded subproblem of U

Definition 3.2.1.1. *Let U be an integer-valued problem, and let A be an algorithm that solves U. We say that A is a* **pseudo-polynomial-time algorithm for U** *if there exists a polynomial p of two variables such that*

$$Time_A(x) = O(p(|x|, Max\text{-}Int(x)))$$

for every instance x of U.

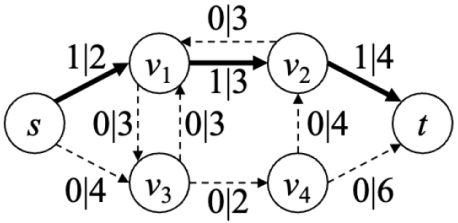
Definition 3.2.1.2. *Let U be an integer-valued problem, and let h be a non-decreasing function from $\mathbb{N}$ to $\mathbb{N}$. The* **h-value-bounded subproblem of U**, ***Value*(h)-U**, *is the problem obtained from U by restricting the set of all input instances of U to the set of input instances x with $Max\text{-}Int(x) \leq h(|x|)$.*

请证明下述定理

Theorem 3.2.1.3. *Let U be an integer-valued problem, and let A be a pseudo-polynomial-time algorithm for U. Then, for every polynomial h, there exists a polynomial-time algorithm for Value(h)-U (i.e., if U is a decision problem then Value(h)-$U \in \mathrm{P}$, and if U is an optimization problem then Value(h)-$U \in \mathrm{PO}$).*

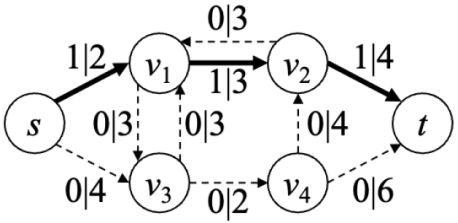
你能解释这些概念吗？

- 整数流
- 单位路径流

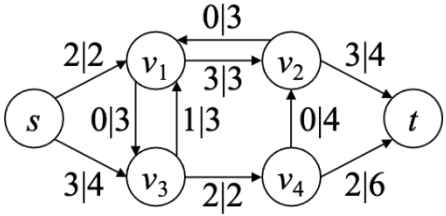


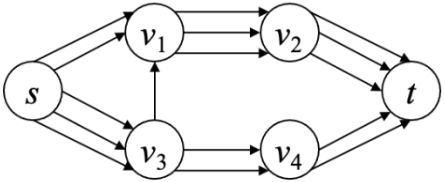
你能解释这些概念吗？

- 整数流
- 单位路径流



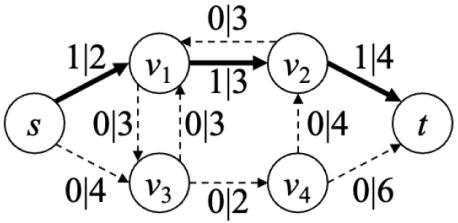
定理 7.8 对于所有弧的容量都是整数的流网络：最大流的值是整数，存在一个最大流是整数流，且该最大流可表示为若干单位路径流的和。



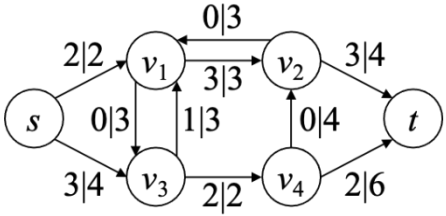


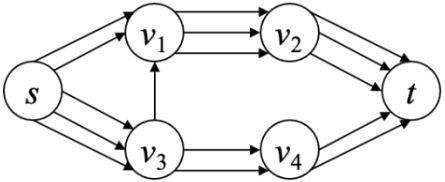
你能解释这些概念吗？

- 整数流
- 单位路径流



定理 7.8 对于所有弧的容量都是整数的流网络：最大流的值是整数，存在一个最大流是整数流，且该最大流可表示为若干单位路径流的和。

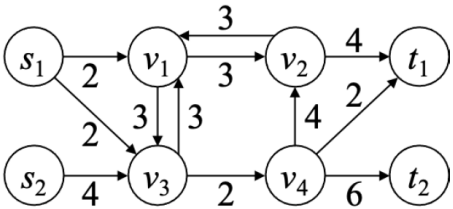


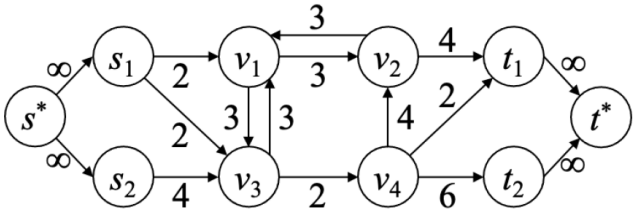


- 对于所有弧的容量都是1的流网络，最大流有什么特征？

你能解释这些概念吗？

- 单源单汇流网络
- 多源多汇流网络
- 超源、超汇





OT

- 除福特-法尔克森算法外，最大流问题还有很多其它算法，例如Edmonds-Karp算法、Dinitz算法、Push-Relabel算法等，请调研至少2种算法（其中至多1种来自上述例子），结合例子介绍算法的设计与分析，与福特-法尔克森算法比较异同并分析优劣。