- 第10周（下下周）：论文报告
  - 论文1，学号**后2位** % 4 = 1：Fast Exact Shortest-Path Distance Queries on Large Networks by Pruned Landmark Labeling
  - 论文2，学号后2位 % 4 = 2：Fast Shortest-Path Distance Queries on Road Networks by Pruned Highway Labeling
  - 论文3，学号后2位 % 4 = 3：Dinitz' Algorithm: The Original Version and Even's Version
  - 论文4，学号后2位 % 4 = 0：A New Approach to the Maximum-Flow Problem
  - 论文5，所有人可选：Keyword Search over Knowledge Graphs via Static and Dynamic Hub Labelings

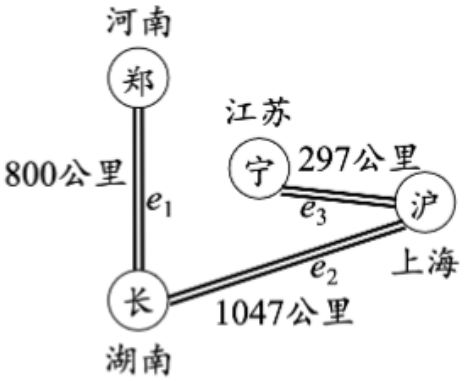
- 论文报告的要求
  - **15~25分钟**，讲解1篇论文的主要内容
  - 制作PPT讲稿，结合例子，图文并茂
  - 有余力的同学，鼓励延伸到相关内容

  - 4月27日23:59前，发到kzhang@smail.nju.edu.cn
    - 邮件标题：GTA论文报告提交
    - PPT命名：论文编号-学号-姓名.**pdf**，例如 4-221220000-张三.pdf

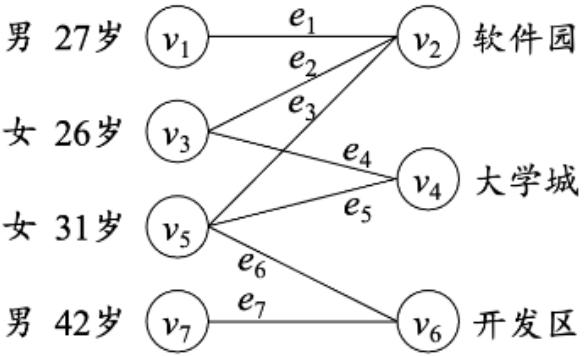
第6章 赋权图
第7章 有向图

程龚

顶点和/或边有属性（权）的图

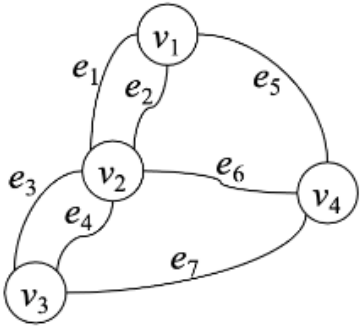


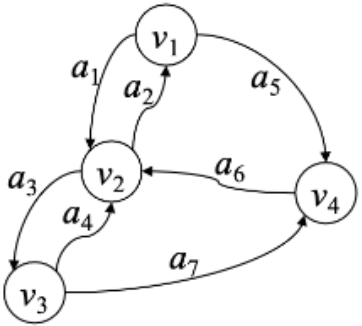
(a)

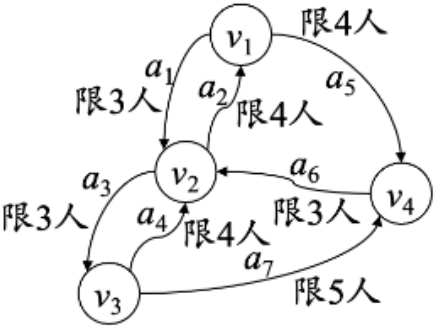


(b)

边有方向（和权）的图


v_1
v_2
v_3
v_4
e_1
e_2
e_3
e_4
e_5
e_6
e_7


v_1
v_2
v_3
v_4
a_1
a_2
a_3
a_4
a_5
a_6
a_7


v_1
v_2
v_3
v_4
a_1
a_2
a_3
a_4
a_5
a_6
a_7
限4人
限3人
限4人
限3人
限4人
限3人
限5人

本次课的主要内容

6.1 赋权图和距离
6.2 最小生成树
6.3 赋权欧拉图
6.4 赋权哈密尔顿图

7.1 有向图的定义
7.2 有向图的表示
7.3 有向图的连通
7.4 有向图的距离
7.5 流网络和最大流

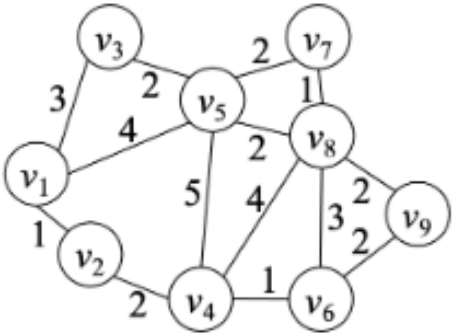
本次课的主要内容

6.1 赋权图和距离
6.2 最小生成树
6.3 赋权欧拉图
6.4 赋权哈密尔顿图

7.1 有向图的定义
7.2 有向图的表示
7.3 有向图的连通
7.4 有向图的距离
7.5 流网络和最大流

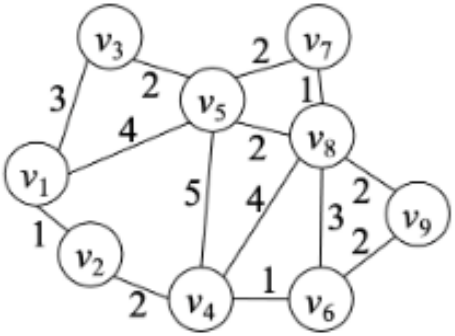
赋权图和距离

- **赋权图**：$G = <V, E, w>$
  - V：顶点的有限集合
  - E：边的有限集合
  - w：**赋权函数**，$E \rightarrow \mathcal{R}$
    - $w(e)$：边e的**权**



赋权图和距离

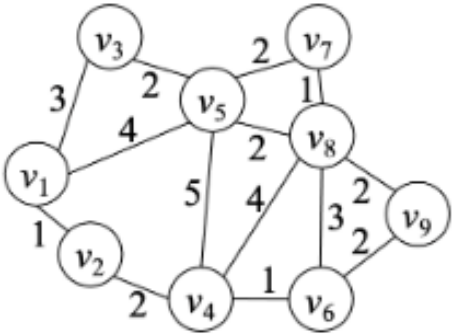
- **邻接矩阵**
  - 引入特殊数值（∞）表示不相邻



$$\begin{pmatrix}
\infty & 1 & 3 & \infty & 4 & \infty & \infty & \infty & \infty \\
1 & \infty & \infty & 2 & \infty & \infty & \infty & \infty & \infty \\
3 & \infty & \infty & \infty & 2 & \infty & \infty & \infty & \infty \\
\infty & 2 & \infty & \infty & 5 & 1 & \infty & 4 & \infty \\
4 & \infty & 2 & 5 & \infty & \infty & 2 & 2 & \infty \\
\infty & \infty & \infty & 1 & \infty & \infty & \infty & 3 & 2 \\
\infty & \infty & \infty & \infty & 2 & \infty & \infty & 1 & \infty \\
\infty & \infty & \infty & 4 & 2 & 3 & 1 & \infty & 2 \\
\infty & \infty & \infty & \infty & \infty & 2 & \infty & 2 & \infty
\end{pmatrix}$$

赋权图和距离

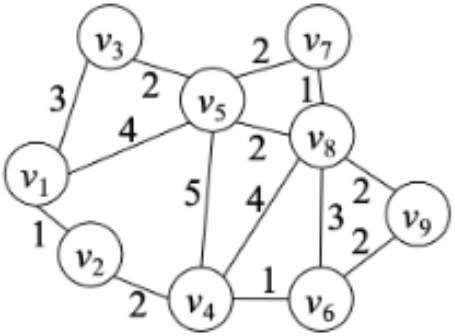
- 邻接矩阵
  - 引入特殊数值（∞）表示不相邻
- **关联矩阵**



$$\begin{pmatrix}
\infty & 1 & 3 & \infty & 4 & \infty & \infty & \infty & \infty \\
1 & \infty & \infty & 2 & \infty & \infty & \infty & \infty & \infty \\
3 & \infty & \infty & \infty & 2 & \infty & \infty & \infty & \infty \\
\infty & 2 & \infty & \infty & 5 & 1 & \infty & 4 & \infty \\
4 & \infty & 2 & 5 & \infty & \infty & 2 & 2 & \infty \\
\infty & \infty & \infty & 1 & \infty & \infty & \infty & 3 & 2 \\
\infty & \infty & \infty & \infty & 2 & \infty & \infty & 1 & \infty \\
\infty & \infty & \infty & 4 & 2 & 3 & 1 & \infty & 2 \\
\infty & \infty & \infty & \infty & \infty & 2 & \infty & 2 & \infty
\end{pmatrix}$$

赋权图和距离

- 邻接矩阵
  - 引入特殊数值（∞）表示不相邻
- 关联矩阵
- **邻接表**

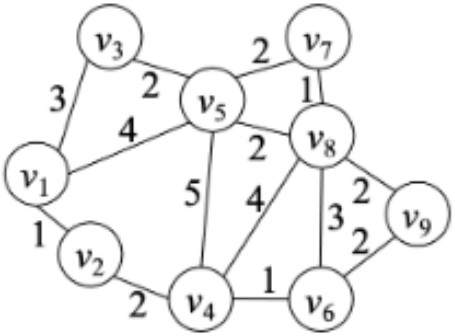


$$\begin{pmatrix}
\infty & 1 & 3 & \infty & 4 & \infty & \infty & \infty & \infty \\
1 & \infty & \infty & 2 & \infty & \infty & \infty & \infty & \infty \\
3 & \infty & \infty & \infty & 2 & \infty & \infty & \infty & \infty \\
\infty & 2 & \infty & \infty & 5 & 1 & \infty & 4 & \infty \\
4 & \infty & 2 & 5 & \infty & \infty & 2 & 2 & \infty \\
\infty & \infty & \infty & 1 & \infty & \infty & \infty & 3 & 2 \\
\infty & \infty & \infty & \infty & 2 & \infty & \infty & 1 & \infty \\
\infty & \infty & \infty & 4 & 2 & 3 & 1 & \infty & 2 \\
\infty & \infty & \infty & \infty & \infty & 2 & \infty & 2 & \infty
\end{pmatrix}$$

顶点	⟨邻点, 边权⟩ 列表
v_1	$\langle v_2, 1\rangle$, $\langle v_3, 3\rangle$, $\langle v_5, 4\rangle$
v_2	$\langle v_1, 1\rangle$, $\langle v_4, 2\rangle$
v_3	$\langle v_1, 3\rangle$, $\langle v_5, 2\rangle$
v_4	$\langle v_2, 2\rangle$, $\langle v_5, 5\rangle$, $\langle v_6, 1\rangle$, $\langle v_8, 4\rangle$
v_5	$\langle v_1, 4\rangle$, $\langle v_3, 2\rangle$, $\langle v_4, 5\rangle$, $\langle v_7, 2\rangle$, $\langle v_8, 2\rangle$
v_6	$\langle v_4, 1\rangle$, $\langle v_8, 3\rangle$, $\langle v_9, 2\rangle$
v_7	$\langle v_5, 2\rangle$, $\langle v_8, 1\rangle$
v_8	$\langle v_4, 4\rangle$, $\langle v_5, 2\rangle$, $\langle v_6, 3\rangle$, $\langle v_7, 1\rangle$, $\langle v_9, 2\rangle$
v_9	$\langle v_6, 2\rangle$, $\langle v_8, 2\rangle$

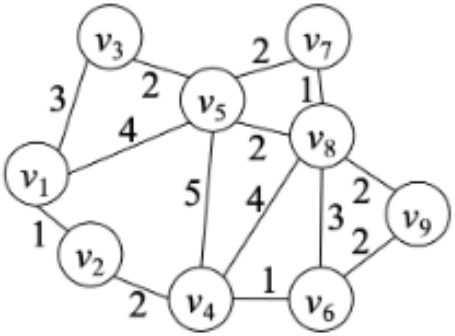
赋权图和距离

- **（赋权）长度**
  - 经过的边的权和



赋权图和距离

- （赋权）长度
  - 经过的边的权和
- **最短路**
- **距离**

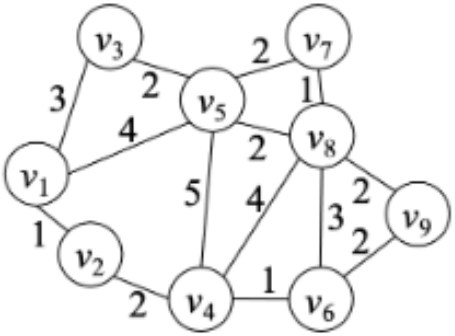


- **离心率**
- **中心点**
- **半径**
- **中心**
- **边缘点**
- **直径**

赋权图和距离

- （赋权）长度
  - 经过的边的权和
- 最短路
- 距离
  - **对于连通赋权图$G = <V, E, w>$，距离函数*dist*满足三角不等式吗：**

$$\forall u, v, w \in V,\ dist(u, v) + dist(v, w) \geq dist(u, w)$$



- 离心率
- 中心点
- 半径
- 中心
- 边缘点
- 直径

赋权图和距离

- 如何计算两个顶点间的距离？
- 如何计算一个顶点和赋权图中所有顶点间的距离？

- 戴克斯特拉算法
  （只适用于边权非负的赋权图）

算法 6.1: 戴克斯特拉算法

输入: 赋权图 $G = \langle V, E, w \rangle$，顶点 u

初值: V 中所有顶点的 d 初值为 ∞；Q 初值为 V

```
1 u.d ← 0;
2 while Q ≠ ∅ do
3 |   v ← arg min_{w∈Q} w.d;
4 |   Q ← Q \ {v};
5 |   foreach (v, w) ∈ E do
6 |   |   if w.d > v.d + w((v, w)) then
7 |   |   |   w.d ← v.d + w((v, w));
```

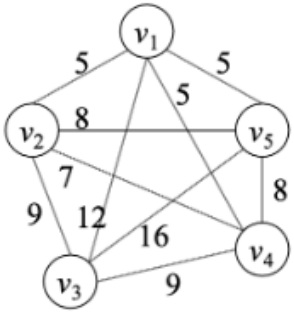
本次课的主要内容

6.1 赋权图和距离
6.2 最小生成树
6.3 赋权欧拉图
6.4 赋权哈密尔顿图

7.1 有向图的定义
7.2 有向图的表示
7.3 有向图的连通
7.4 有向图的距离
7.5 流网络和最大流

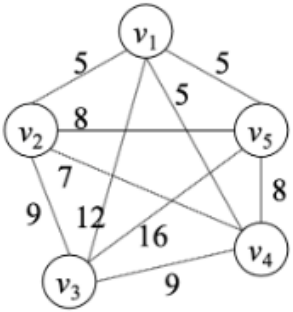
最小生成树

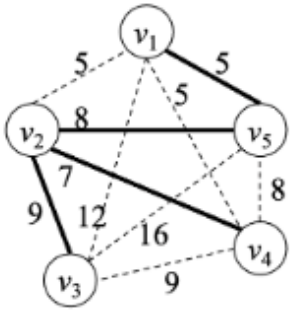
- **最小生成树**
  - 对于连通赋权图，边权和最小的生成树



最小生成树

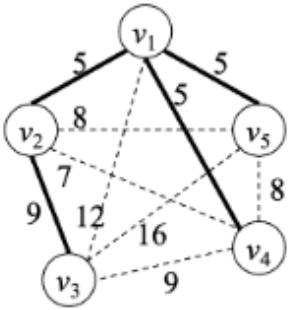
- 最小生成树
  - 对于连通赋权图，边权和最小的生成树





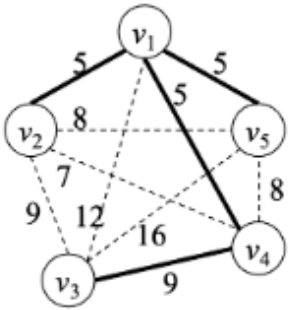
(a)

非最小生成树



(b)

最小生成树



(c)

最小生成树

最小生成树

- 如何找出最小生成树?

- 普里姆算法
- 克拉斯克尔算法

算法 6.2: 普里姆算法

输入: 连通赋权图 $G = \langle V, E, w \rangle$

初值: V 中所有顶点的 mw 初值为 ∞，parent 初值为 null；Q 初值为 V

```
1  r ← V 中任意一个顶点;
2  r.mw ← 0;
3  while Q ≠ ∅ do
4  |  v ← arg min_{u∈Q} u.mw;
5  |  Q ← Q \ {v};
6  |  if v ≠ r then
7  |  |  输出 ((v.parent, v));
8  |  foreach (v, w) ∈ E do
9  |  |  if w ∈ Q 且 w((v, w)) < w.mw then
10 |  |  |  w.mw ← w((v, w));
11 |  |  |  w.parent ← v;
```

算法 6.3: 克拉斯克尔算法

输入: 连通赋权图 $G = \langle V, E, w \rangle$

初值: E 中所有边按权从小到大排序

```
1  foreach v ∈ V do
2  |  建树 (v);
3  foreach (u, v) ∈ E do
4  |  if 查树 (u) ≠ 查树 (v) then
5  |  |  输出 ((u, v));
6  |  |  并树 (u, v);
```

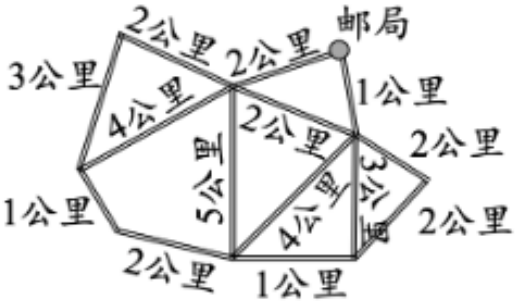
本次课的主要内容

6.1 赋权图和距离
6.2 最小生成树
6.3 赋权欧拉图
6.4 赋权哈密尔顿图

7.1 有向图的定义
7.2 有向图的表示
7.3 有向图的连通
7.4 有向图的距离
7.5 流网络和最大流

赋权欧拉图

- **中国邮递员问题**
  - 一个邮递员每次上班，要走遍他负责送信的每段路，然后回到邮局，应该怎样走才能使所走的路程最短?



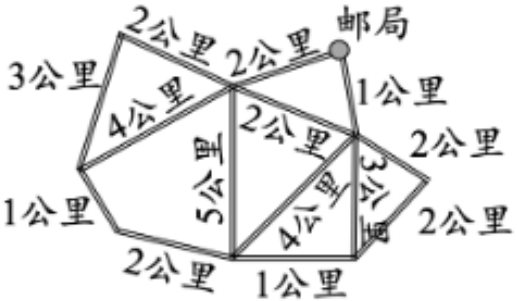
赋权欧拉图

- 管梅谷，1934年出生于中国

http://www.maths.sdnu.edu.cn/__local/C/01/4D/70D003509166569B87CF5A5AFE0_5CB5331D_4B219.jpg

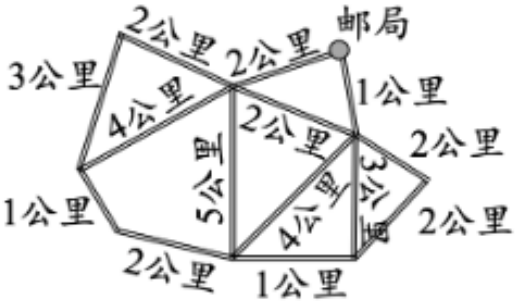
赋权欧拉图

- **邮递路线**
  - 经过每条边至少一次的闭路线



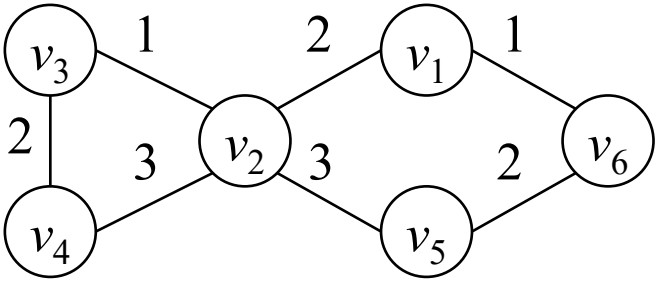
赋权欧拉图

- 邮递路线
  - 经过每条边至少一次的闭路线
- **最优邮递路线**
  - 最短的邮递路线



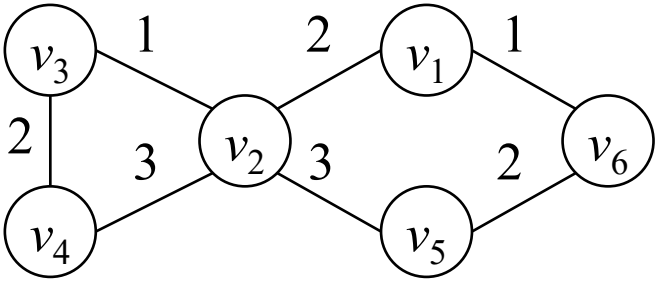
赋权欧拉图

- 赋权欧拉图的最优邮递路线是什么?



赋权欧拉图

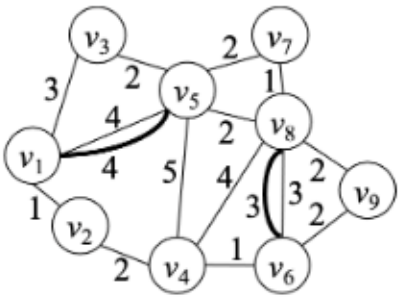
- 赋权欧拉图的最优邮递路线是什么?
- 非欧拉图的邮递路线一定重复经过边

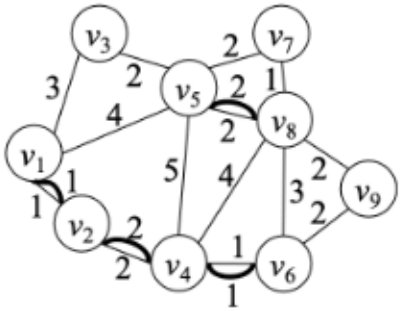


赋权欧拉图

- 赋权欧拉图的最优邮递路线是什么？

- 非欧拉图的邮递路线一定重复经过边
- 对于一条邮递路线，其每次重复经过一条边，便将一条权相同的重边增加到赋权图G中
  - 最终形成的赋权图记作G^{E}，添加的重边的集合记作E^{M}

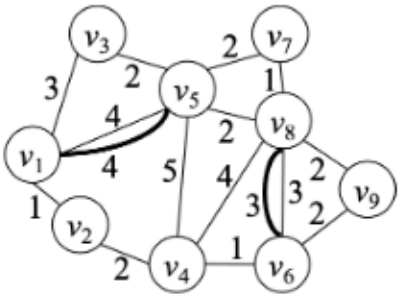


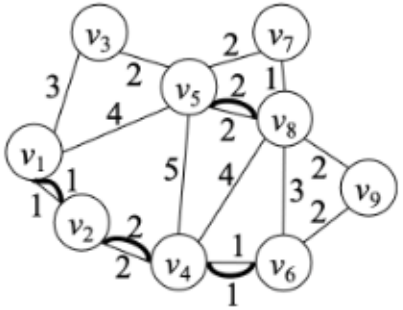


赋权欧拉图

- 赋权欧拉图的最优邮递路线是什么？

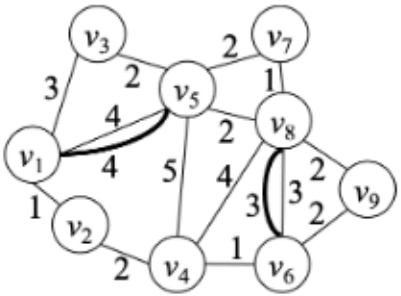
- 非欧拉图的邮递路线一定重复经过边
- 对于一条邮递路线，其每次重复经过一条边，便将一条权相同的重边增加到赋权图G中
  - 最终形成的赋权图记作G^{E}，添加的重边的集合记作E^{M}
  - G^{E}有什么特征？它的最优邮递路线是什么？

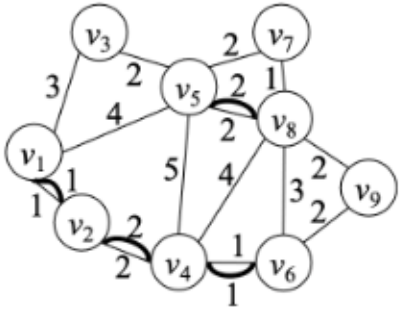




赋权欧拉图

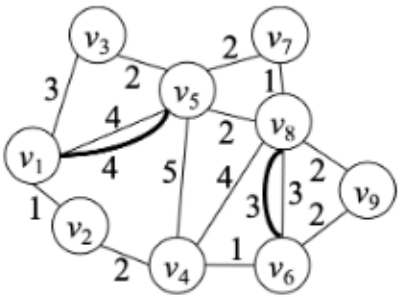
- 中国邮递员问题可以转化为下述问题
  - 如何向赋权图G中添加重边的集合E^M形成赋权欧拉图G^E，且E^M的权和最小？

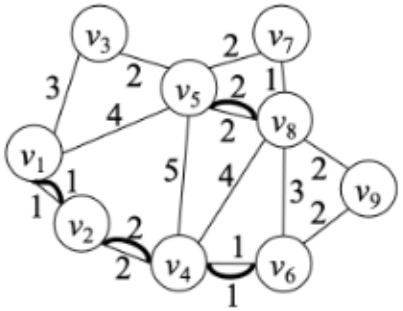




赋权欧拉图

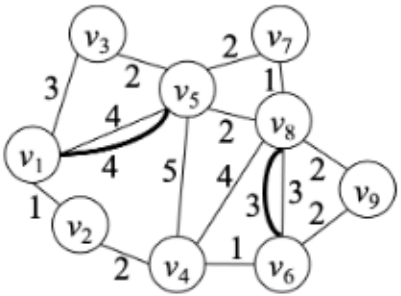
- 中国邮递员问题可以转化为下述问题
  - 如何向赋权图G中添加重边的集合E^{M}形成赋权欧拉图G^{E}，且E^{M}的权和最小？
- 存在一条最优邮递路线，其对应的重边的集合E^{M}是：以赋权图G中所有$2k$个（$k \geq 0$）度为奇数的顶点为起点和终点的k条无公共边的最短路经过的边的集合

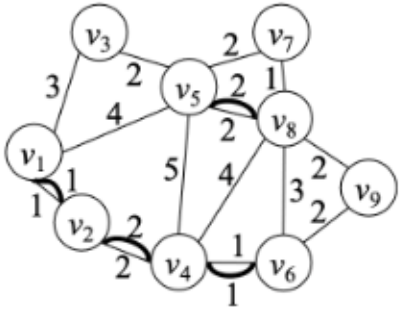




赋权欧拉图

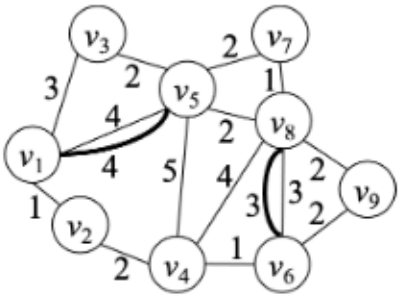
- 中国邮递员问题可以转化为下述问题
  - 如何向赋权图G中添加重边的集合E^{M}形成赋权欧拉图G^{E}，且E^{M}的权和最小？
- 存在一条最优邮递路线，其对应的重边的集合E^{M}是：以赋权图G中所有$2k$个（$k \geq 0$）度为奇数的顶点为起点和终点的k条无公共边的最短路经过的边的集合
  - 关键：$2k$个顶点如何配对？

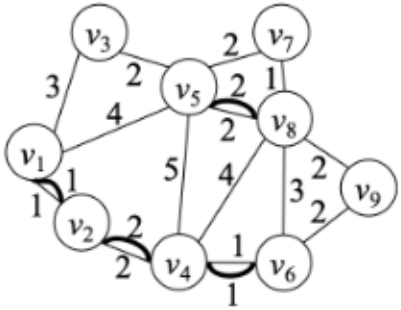




赋权欧拉图

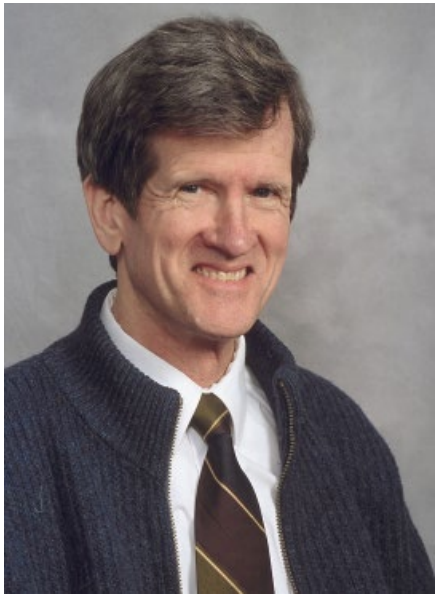
- 埃德蒙兹-约翰逊算法
  - 找出$2k$个度为奇数的顶点间长度之和最小的k条最短路
  - 将其经过的边作为重边增加到图中
  - 再找一条欧拉回路





赋权欧拉图

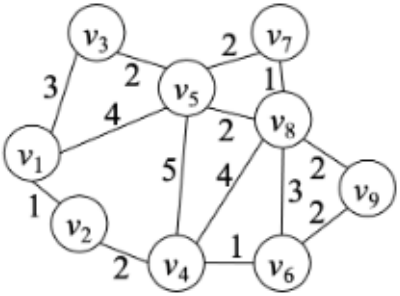
- Ellis Lane Johnson，1938年出生于美国，2000年获得约翰·冯·诺依曼理论奖

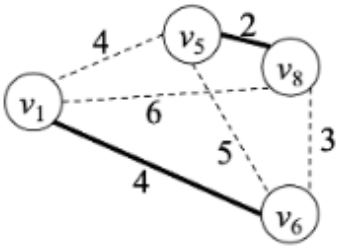


https://www.isye.gatech.edu/sites/default/files/emp-profiles/johnson_ellis_-_bust.jpg

赋权欧拉图

- V^{O}：G中所有度为奇数的顶点的集合
- G^{O}：以V^{O}作为顶点集的完全赋权图
  - 边(u, v)的权：G中顶点u和v间的距离





算法 6.4: 埃德蒙兹-约翰逊算法

输入: 非空连通且边权为非负实数的赋权图 $G = \langle V, E, w \rangle$

初值: E^{M} 初值为 $\emptyset$

1 $V^{O} \leftarrow \{v \in V \mid d(v)$是奇数$\}$;

2 $G^{O} = \langle V^{O}, E^{O}, w^{O} \rangle \leftarrow$ 完全赋权图，其中顶点集为 V^{O}，赋权函数 $w^{O}(u, v)$ 为 G 中顶点 u 和 v 间的距离;

3 $M \leftarrow G^{O}$ 的最小权完美匹配;

4 **foreach** $(u, v) \in M$ **do**

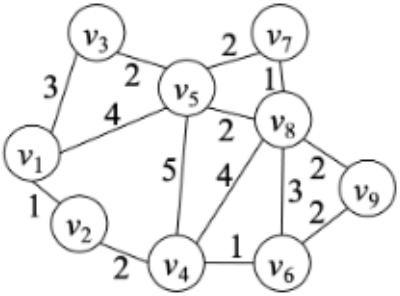
5 $\quad E_{u,v} \leftarrow G$ 中一条最短 u-v 路经过的边的集合;

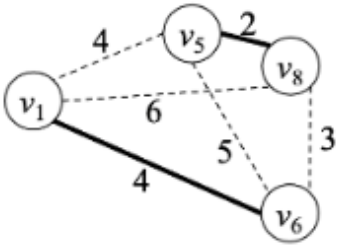
6 $\quad E^{M} \leftarrow E^{M} \cup E_{u,v}$;

7 输出 （图 $\langle V, E \cup E^{M} \rangle$ 的欧拉回路）;

赋权欧拉图

- G中$2k$个度为奇数的顶点间长度之和最小的k条最短路
  → G^{O}的**最小权完美匹配**M
  （权和最小的完美匹配）





算法 6.4: 埃德蒙兹-约翰逊算法

输入: 非空连通且边权为非负实数的赋权图 $G = \langle V, E, w \rangle$

初值: E^{M} 初值为 $\emptyset$

1 $V^{O} \leftarrow \{v \in V \mid d(v)$是奇数$\}$;

2 $G^{O} = \langle V^{O}, E^{O}, w^{O} \rangle \leftarrow$ 完全赋权图，其中顶点集为 V^{O}，赋权函数 $w^{O}(u, v)$ 为 G 中顶点 u 和 v 间的距离;

3 $M \leftarrow G^{O}$ 的最小权完美匹配;

4 **foreach** $(u, v) \in M$ **do**

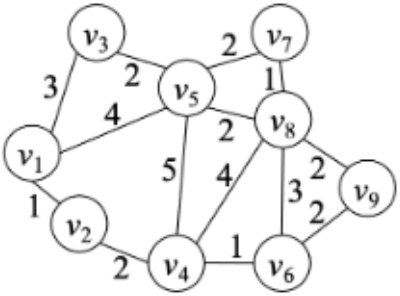
5 $\quad E_{u,v} \leftarrow G$ 中一条最短 u-v 路经过的边的集合;

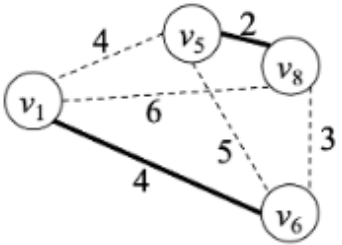
6 $\quad E^{M} \leftarrow E^{M} \cup E_{u,v}$;

7 输出 （图 $\langle V, E \cup E^{M} \rangle$ 的欧拉回路）;

赋权欧拉图

- 将匹配M中每条边的两个端点在G中的一条最短路经过的所有边作为重边增加到G中





算法 6.4: 埃德蒙兹-约翰逊算法

输入: 非空连通且边权为非负实数的赋权图 $G = \langle V, E, w \rangle$

初值: E^{M} 初值为 $\emptyset$

1 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d(v)$是奇数$\}$;

2 $G^{\mathrm{O}} = \langle V^{\mathrm{O}}, E^{\mathrm{O}}, w^{\mathrm{O}} \rangle \leftarrow$ 完全赋权图，其中顶点集为 V^{O}，赋权函数 $w^{\mathrm{O}}(u, v)$ 为 G 中顶点 u 和 v 间的距离;

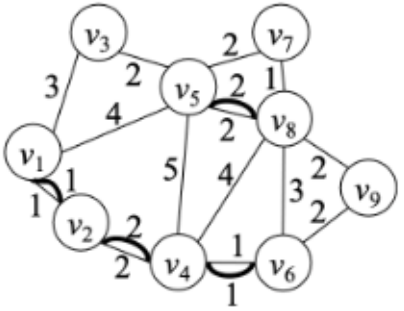
3 $M \leftarrow G^{\mathrm{O}}$ 的最小权完美匹配;

4 **foreach** $(u, v) \in M$ **do**

5 $\quad E_{u,v} \leftarrow G$ 中一条最短 u-v 路经过的边的集合;

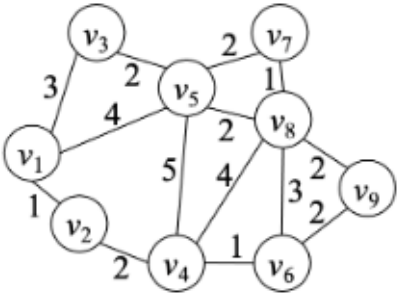
6 $\quad E^{\mathrm{M}} \leftarrow E^{\mathrm{M}} \cup E_{u,v}$;

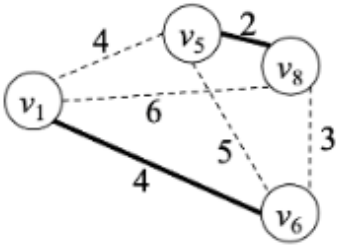
7 输出 (图 $\langle V, E \cup E^{M} \rangle$ 的欧拉回路);



赋权欧拉图

- 输出一条欧拉回路





算法 6.4: 埃德蒙兹-约翰逊算法

输入: 非空连通且边权为非负实数的赋权图 $G = \langle V, E, w \rangle$

初值: E^{M} 初值为 $\emptyset$

1 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d(v)$是奇数$\}$;

2 $G^{\mathrm{O}} = \langle V^{\mathrm{O}}, E^{\mathrm{O}}, w^{\mathrm{O}} \rangle \leftarrow$ 完全赋权图，其中顶点集为 V^{O}，赋权函数 $w^{\mathrm{O}}(u, v)$ 为 G 中顶点 u 和 v 间的距离;

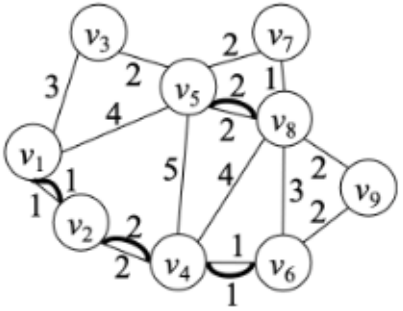
3 $M \leftarrow G^{\mathrm{O}}$ 的最小权完美匹配;

4 **foreach** $(u, v) \in M$ **do**

5 $\quad E_{u,v} \leftarrow G$ 中一条最短 u-v 路经过的边的集合;

6 $\quad E^{\mathrm{M}} \leftarrow E^{\mathrm{M}} \cup E_{u,v}$;

7 输出 (图 $\langle V, E \cup E^{M} \rangle$ 的欧拉回路);



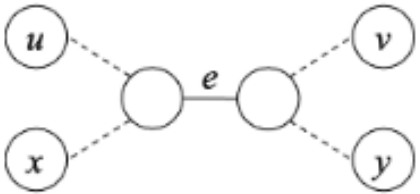
赋权欧拉图

- 存在一条最优邮递路线，其对应的重边的集合E^{M}是：
  以赋权图G中所有$2k$个（$k \geq 0$）度为奇数的顶点为起点和终点的k条无公共边的最短路经过的边的集合

赋权欧拉图

- 存在一条最优邮递路线，其对应的重边的集合E^{M}是：
  以赋权图G中所有$2k$个（$k \geq 0$）度为奇数的顶点为起点和终点的k条无公共边的最短路经过的边的集合

- 埃德蒙兹-约翰逊算法中找到的最短路不经过公共边（或经过的公共边的权为0）
  - 你能自己证明吗？



赋权欧拉图

- 时间复杂度：$O(n^3 + m)$
  - 计算G中所有度为奇数的顶点间的距离：$O(n^3)$
  - 构造完全赋权图G^{O}并计算最小权完美匹配：$O(n^3)$
  - 增加重边并计算欧拉回路：$O(n + m)$

算法 6.4: 埃德蒙兹-约翰逊算法

输入: 非空连通且边权为非负实数的赋权图 $G = \langle V, E, w\rangle$

初值: E^{M} 初值为 $\emptyset$

1 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d(v)$是奇数$\}$;

2 $G^{\mathrm{O}} = \langle V^{\mathrm{O}}, E^{\mathrm{O}}, w^{\mathrm{O}}\rangle \leftarrow$ 完全赋权图，其中顶点集为 V^{O}，赋权函数 $w^{\mathrm{O}}(u, v)$ 为 G 中顶点 u 和 v 间的距离;

3 $M \leftarrow G^{\mathrm{O}}$ 的最小权完美匹配;

4 **foreach** $(u, v) \in M$ **do**

5 $\quad E_{u,v} \leftarrow G$ 中一条最短 u-v 路经过的边的集合;

6 $\quad E^{\mathrm{M}} \leftarrow E^{\mathrm{M}} \cup E_{u,v}$;

7 输出 （图 $\langle V, E \cup E^{M}\rangle$ 的欧拉回路）;

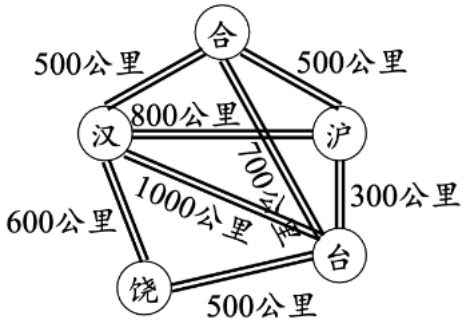
本次课的主要内容

6.1 赋权图和距离
6.2 最小生成树
6.3 赋权欧拉图
6.4 赋权哈密尔顿图

7.1 有向图的定义
7.2 有向图的表示
7.3 有向图的连通
7.4 有向图的距离
7.5 流网络和最大流

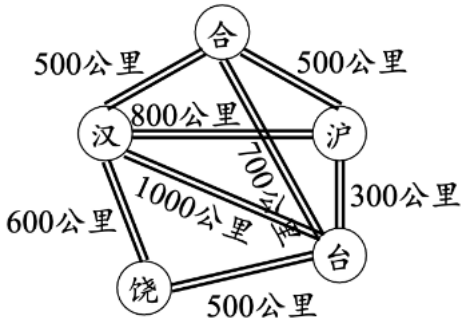
赋权哈密尔顿图

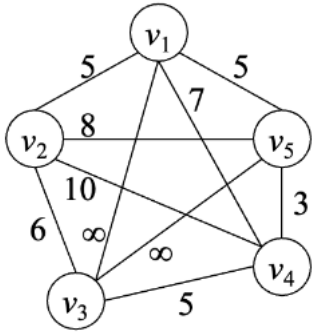
- **旅行商问题**（TSP）
  - 一个旅行商要访问他负责推销的每座城市恰一次，然后回到公司，应该按怎样的顺序访问才能使总路程最短?



赋权哈密尔顿图

- 旅行商问题（TSP）
  - 一个旅行商要访问他负责推销的每座城市恰一次，然后回到公司，应该按怎样的顺序访问才能使总路程最短?
  - 完全图且边权为非负实数的赋权图的最短哈密尔顿圈

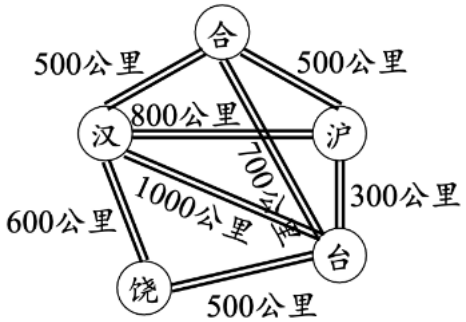


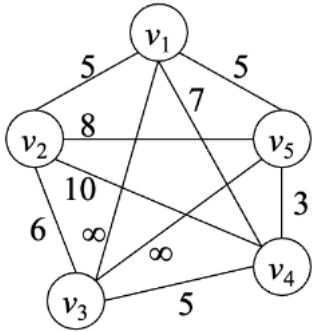


赋权哈密尔顿图

- 旅行商问题（TSP）
  - 一个旅行商要访问他负责推销的每座城市恰一次，然后回到公司，应该按怎样的顺序访问才能使总路程最短?
  - 完全图且边权为非负实数的赋权图的最短哈密尔顿圈
  - 如何将哈密尔顿圈的存在性的判定问题归约为TSP?

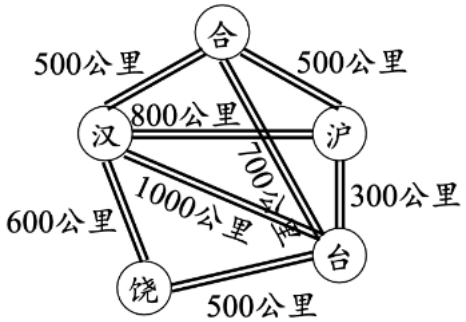
随堂小测

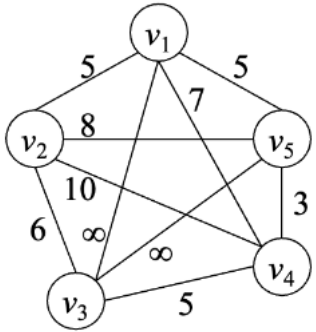




赋权哈密尔顿图

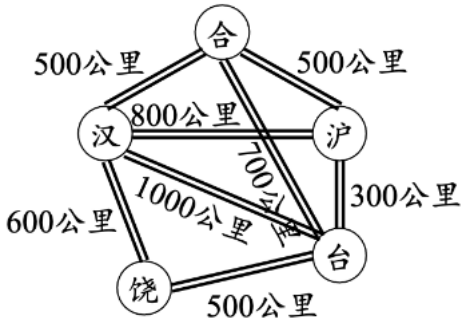
- 旅行商问题（TSP）
  - 一个旅行商要访问他负责推销的每座城市恰一次，然后回到公司，应该按怎样的顺序访问才能使总路程最短?
  - 完全图且边权为非负实数的赋权图的最短哈密尔顿圈
  - 如何将哈密尔顿圈的存在性的判定问题归约为TSP?
- **度量旅行商问题**（Δ-TSP）
  - 赋权函数w满足三角不等式：　$\forall u, v, w \in V,\ w(u,v) + w(v,w) \geq w(u,w)$

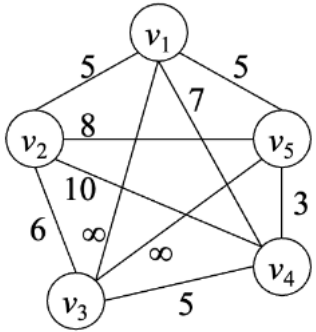




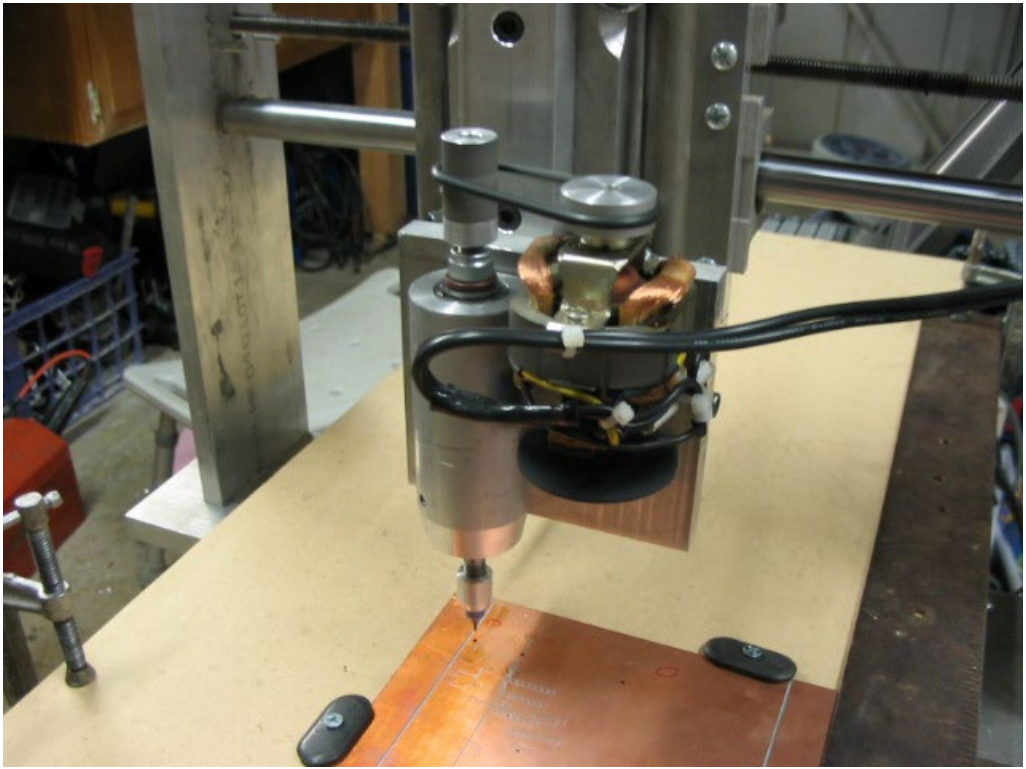
赋权哈密尔顿图

- 旅行商问题（TSP）
  - 一个旅行商要访问他负责推销的每座城市恰一次，然后回到公司，应该按怎样的顺序访问才能使总路程最短?
  - 完全图且边权为非负实数的赋权图的最短哈密尔顿圈
  - 如何将哈密尔顿圈的存在性的判定问题归约为TSP?
- 度量旅行商问题（Δ-TSP）
  - 赋权函数w满足三角不等式：$\forall u, v, w \in V,\ w(u,v) + w(v,w) \geq w(u,w)$
  - 哈密尔顿圈的存在性的判定问题可以归约为Δ-TSP吗?





赋权哈密尔顿图



http://www.buildyouridea.com/cnc/pcb_drill/pcb_drill_4.jpg

赋权哈密尔顿图

- Δ-TSP是一个NP难的优化问题，不存在多项式时间算法（除非P=NP）

赋权哈密尔顿图

- Δ-TSP是一个NP难的优化问题，不存在多项式时间算法（除非P=NP）

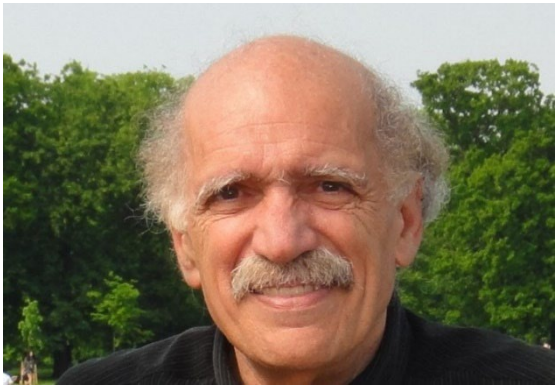
- **近似算法**
  - 如何在多项式时间内，找出一个较短（但未必最短）的哈密尔顿圈，其与最短哈密尔顿圈的长度的差异具有可以证明的界

赋权哈密尔顿图

- 最近邻点算法
- 克里斯托菲德斯-谢尔久科夫算法

赋权哈密尔顿图

- Nicos Christofides，1942年出生于塞浦路斯
- Anatoliy Ivanovich Serdyukov，1951年出生于苏联

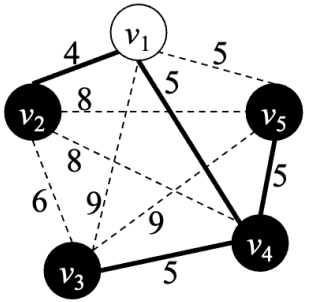


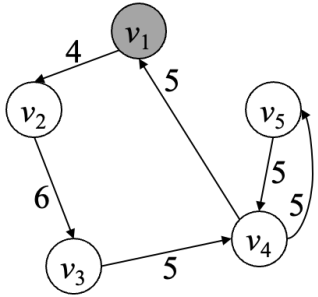
?

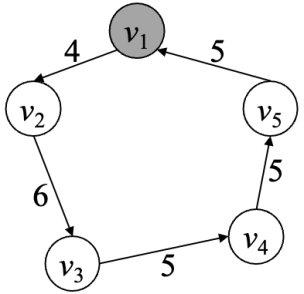
https://www.imperial.ac.uk/ImageCropToolT4/imageTool/uploaded-images/newseventsimage_1586354871721_mainnews2012_x1.jpg

赋权哈密尔顿图

- 克里斯托菲德斯-谢尔久科夫算法
  - 将最小生成树扩充为边权和较小的赋权欧拉图
  - 利用三角不等式将欧拉回路转化为较短的哈密尔顿圈







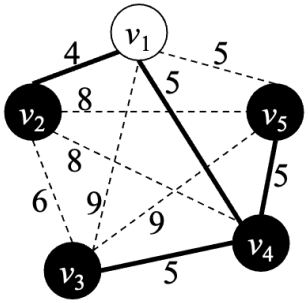
算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $T = \langle V, E_T \rangle \leftarrow G$ 的最小生成树;
2 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^{\mathrm{O}}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M \rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);

赋权哈密尔顿图

- 找出G的一棵最小生成树T
  - V^{O}：T中所有度为奇数的顶点



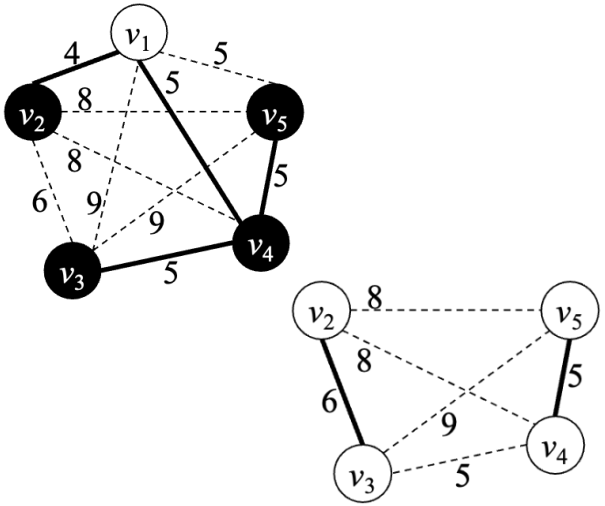
算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $T = \langle V, E_T \rangle \leftarrow G$ 的最小生成树;
2 $V^{O} \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^{O}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M \rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);

赋权哈密尔顿图

- 找出点导出子图$G[V^O]$的最小权完美匹配M



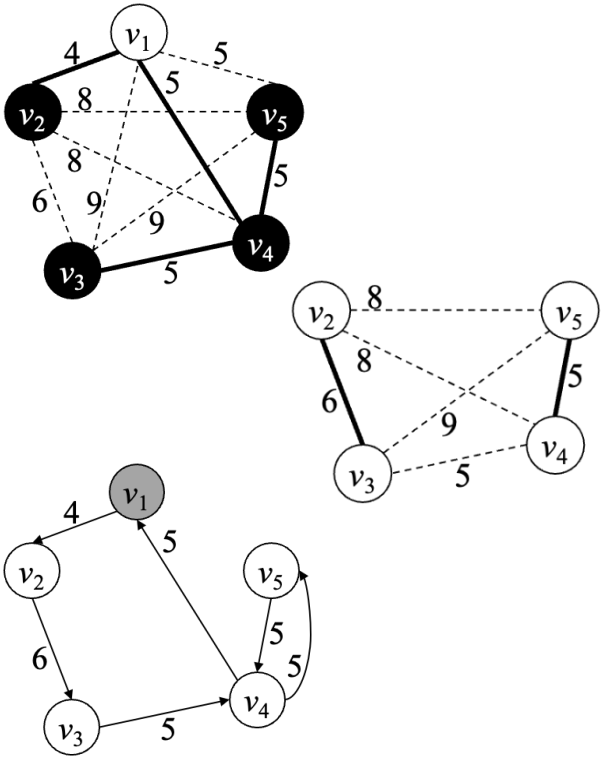
算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $T = \langle V, E_T \rangle \leftarrow G$ 的最小生成树;
2 $V^O \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^O]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M \rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);

赋权哈密尔顿图

- 将M增加到T中形成赋权欧拉图，并找出一条欧拉回路C



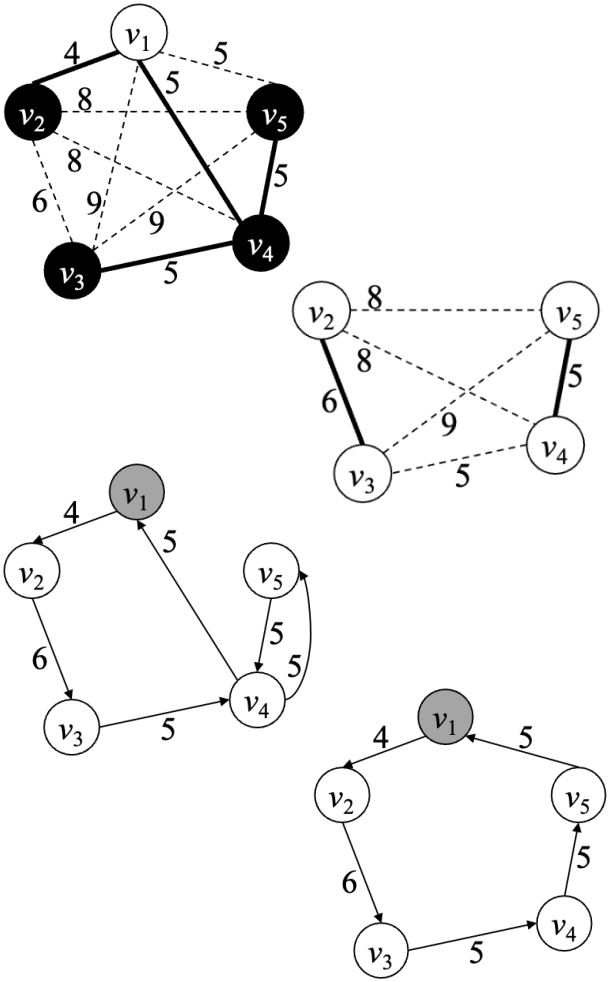
算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $T = \langle V, E_T \rangle \leftarrow G$ 的最小生成树;
2 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^{\mathrm{O}}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M \rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);

赋权哈密尔顿图

- 按序输出C经过的顶点，形成一个哈密尔顿圈



算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $T = \langle V, E_T \rangle \leftarrow G$ 的最小生成树;
2 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^{\mathrm{O}}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M \rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);

赋权哈密尔顿图

- H：算法输出的较短哈密尔顿圈
- H^*：最短哈密尔顿圈

$$
\begin{aligned}
H\text{的长度} &\le C\text{对应的 } G \text{ 中闭路线的长度} \\
&= T\text{的边权和} + M\text{的权和} \\
&\le H^*\text{的长度} + H^*\text{的长度} \cdot 0.5 \\
&= H^*\text{的长度} \cdot 1.5\,.
\end{aligned}
$$

算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $T = \langle V, E_T \rangle \leftarrow G$ 的最小生成树;
2 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^{\mathrm{O}}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M \rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);

赋权哈密尔顿图

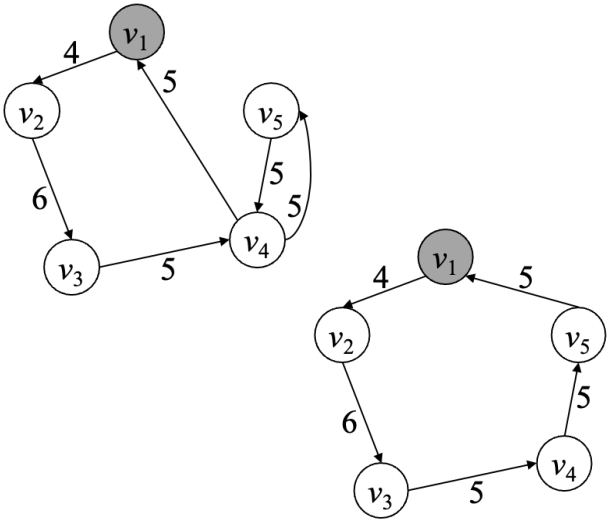
- H：算法输出的较短哈密尔顿圈
- H^*：最短哈密尔顿圈

$$
\begin{aligned}
H\text{的长度} &\le C\text{对应的 } G \text{ 中闭路线的长度} \\
&= T\text{的边权和} + M\text{的权和} \\
&\le H^*\text{的长度} + H^*\text{的长度} \cdot 0.5 \\
&= H^*\text{的长度} \cdot 1.5\,.
\end{aligned}
$$

算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $T = \langle V, E_T \rangle \leftarrow G$ 的最小生成树;
2 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d_T(v)\text{是奇数}\}$;
3 $M \leftarrow G[V^{\mathrm{O}}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M \rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);



赋权哈密尔顿图

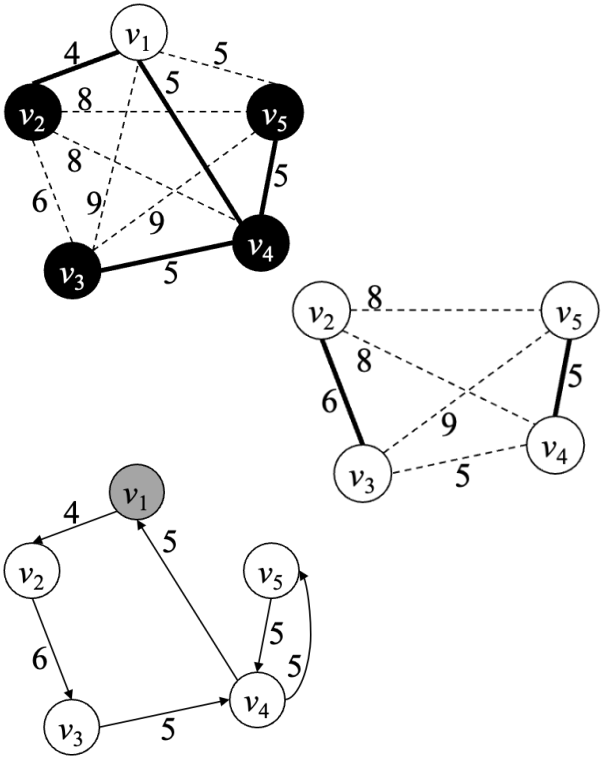
- H：算法输出的较短哈密尔顿圈
- H^*：最短哈密尔顿圈

$$
\begin{aligned}
H\text{的长度} &\le C\text{对应的 } G \text{ 中闭路线的长度} \\
&= T\text{的边权和} + M\text{的权和} \\
&\le H^*\text{的长度} + H^*\text{的长度} \cdot 0.5 \\
&= H^*\text{的长度} \cdot 1.5\,.
\end{aligned}
$$

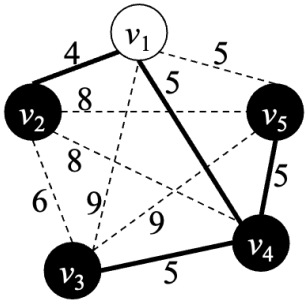
算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $T = \langle V, E_T \rangle \leftarrow G$ 的最小生成树;
2 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^{\mathrm{O}}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M \rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);



赋权哈密尔顿图



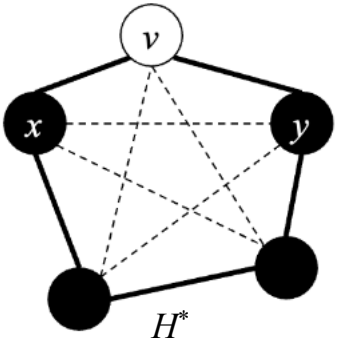
- H：算法输出的较短哈密尔顿圈
- H^*：最短哈密尔顿圈

$$
\begin{aligned}
H\text{的长度} &\le C\text{对应的 } G \text{ 中闭路线的长度} \\
&= T\text{的边权和} + M\text{的权和} \\
&\le H^*\text{的长度} + H^*\text{的长度} \cdot 0.5 \\
&= H^*\text{的长度} \cdot 1.5\,.
\end{aligned}
$$

算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w\rangle$

1 $T = \langle V, E_T\rangle \leftarrow G$ 的最小生成树;
2 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^{\mathrm{O}}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M\rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);



赋权哈密尔顿图

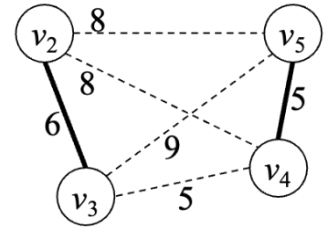
- H：算法输出的较短哈密尔顿圈
- H^*：最短哈密尔顿圈

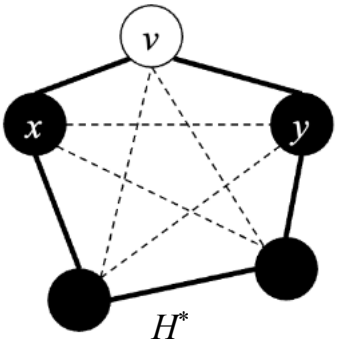
$$
\begin{aligned}
H\text{的长度} &\le C\text{对应的 } G \text{ 中闭路线的长度} \\
&= T\text{的边权和} + M\text{的权和} \\
&\le H^*\text{的长度} + H^*\text{的长度} \cdot 0.5 \\
&= H^*\text{的长度} \cdot 1.5\,.
\end{aligned}
$$

算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w\rangle$

1 $T = \langle V, E_T\rangle \leftarrow G$ 的最小生成树;
2 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^{\mathrm{O}}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M\rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);





赋权哈密尔顿图

- H：算法输出的较短哈密尔顿圈
- H^*：最短哈密尔顿圈

$$
\begin{aligned}
H\text{的长度} &\le C\text{对应的 } G \text{ 中闭路线的长度} \\
&= T\text{的边权和} + M\text{的权和} \\
&\le H^*\text{的长度} + H^*\text{的长度} \cdot 0.5 \\
&= H^*\text{的长度} \cdot 1.5\,.
\end{aligned}
$$

算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $T = \langle V, E_T \rangle \leftarrow G$ 的最小生成树;
2 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^{\mathrm{O}}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M \rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);

赋权哈密尔顿图

- Δ-TSP具有123/122不可近似性，
  即不存在近似比为小于123/122的常数的多项式时间算法（除非P=NP）

赋权哈密尔顿图

- 时间复杂度：$O(n^3)$
  - 计算最小生成树：$O(n^2)$
  - 计算最小权完美匹配：$O(n^3)$
  - 计算欧拉回路：$O(n)$

算法 6.6: 克里斯托菲德斯-谢尔久科夫算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $T = \langle V, E_T \rangle \leftarrow G$ 的最小生成树;
2 $V^{\mathrm{O}} \leftarrow \{v \in V \mid d_T(v)$是奇数$\}$;
3 $M \leftarrow G[V^{\mathrm{O}}]$ 的最小权完美匹配;
4 $C \leftarrow$ 图 $\langle V, E_T \cup M \rangle$ 的欧拉回路;
5 输出 (C 经过的不重复顶点);
6 输出 (C 的起点);

本次课的主要内容

6.1 赋权图和距离
6.2 最小生成树
6.3 赋权欧拉图
6.4 赋权哈密尔顿图

7.1 有向图的定义
7.2 有向图的表示
7.3 有向图的连通
7.4 有向图的距离
7.5 流网络和最大流

有向图的定义

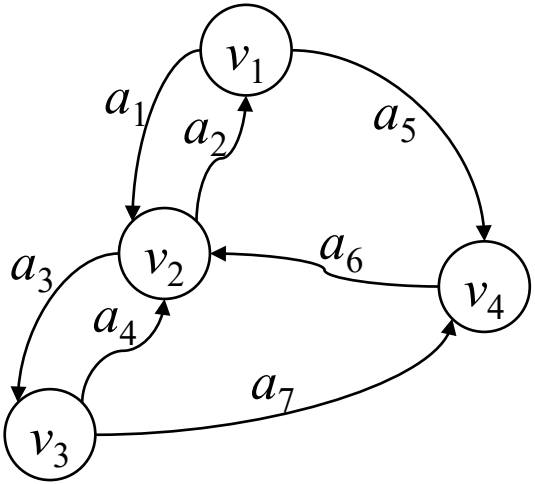
- **有向图**：G = <*V*, *A*>
  - *V*：顶点的有限集合
  - *A*：**有向边（弧）**的有限集合
    - 弧a_5 = <v_1, v_4>是*V*中顶点v_1和v_4组成的有序对
    - v_1和v_4分别称作a_5的**尾**和**头**，统称作**端点**
    - a_5和v_1（及v_4）互相**关联**
    - a_5是v_4的**入弧**，是v_1的**出弧**

- 一些术语
  - v_1和v_4**相邻**
  - v_1是v_4的**入邻点**，v_4是v_1的**出邻点**
  - a_1和a_4**相邻**

  - **重弧（平行弧）**
  - **自环**
  - **简单有向图**

  - a_1和a_2互为**反向弧**

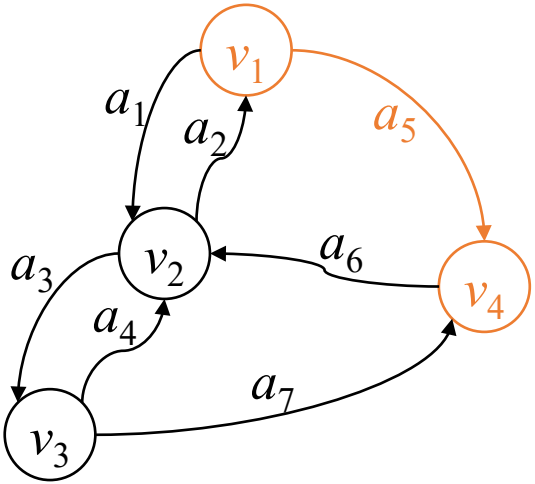
  - **阶**
  - **弧数**



有向图的定义

- **有向图**：G = <*V*, *A*>
  - *V*：顶点的有限集合
  - *A*：**有向边**（**弧**）的有限集合
    - 弧a_5 = <v_1, v_4>是*V*中顶点v_1和v_4组成的有序对
    - v_1和v_4分别称作a_5的**尾**和**头**，统称作**端点**
    - a_5和v_1（及v_4）互相**关联**
    - a_5是v_4的**入弧**，是v_1的**出弧**

- 一些术语
  - v_1和v_4**相邻**
  - v_1是v_4的**入邻点**，v_4是v_1的**出邻点**
  - a_1和a_4**相邻**
  - **重弧**（**平行弧**）
  - **自环**
  - **简单有向图**
  - a_1和a_2互为**反向弧**
  - **阶**
  - **弧数**



有向图的定义

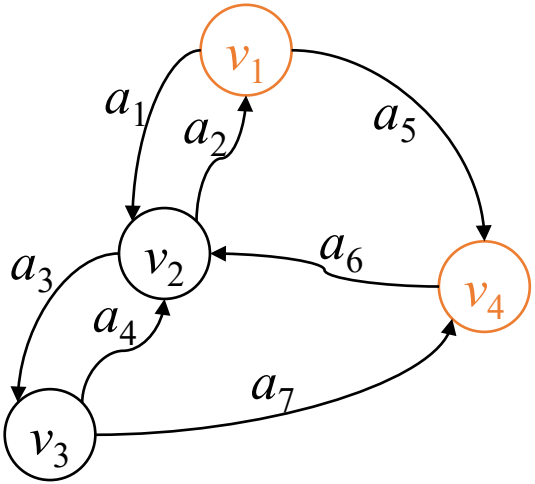
- **有向图**：G = <*V*, *A*>
  - *V*：顶点的有限集合
  - *A*：**有向边**（**弧**）的有限集合
    - 弧a_5 = <v_1, v_4>是*V*中顶点v_1和v_4组成的有序对
    - v_1和v_4分别称作a_5的**尾**和**头**，统称作**端点**
    - a_5和v_1（及v_4）互相**关联**
    - a_5是v_4的**入弧**，是v_1的**出弧**

- 一些术语
  - v_1和v_4**相邻**
  - v_1是v_4的**入邻点**，v_4是v_1的**出邻点**
  - a_1和a_4**相邻**

  - **重弧**（**平行弧**）
  - **自环**
  - **简单有向图**

  - a_1和a_2互为**反向弧**

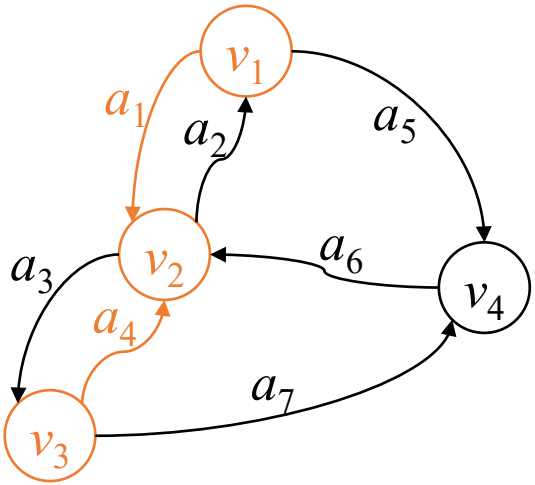
  - **阶**
  - **弧数**



有向图的定义

- **有向图**：G = <*V*, *A*>
  - *V*：顶点的有限集合
  - *A*：**有向边**（**弧**）的有限集合
    - 弧a_5 = <v_1, v_4>是*V*中顶点v_1和v_4组成的有序对
    - v_1和v_4分别称作a_5的**尾**和**头**，统称作**端点**
    - a_5和v_1（及v_4）互相**关联**
    - a_5是v_4的**入弧**，是v_1的**出弧**

- 一些术语
  - v_1和v_4**相邻**
  - v_1是v_4的**入邻点**，v_4是v_1的**出邻点**
  - a_1和a_4**相邻**
  - **重弧**（**平行弧**）
  - **自环**
  - **简单有向图**
  - a_1和a_2互为**反向弧**
  - **阶**
  - **弧数**



有向图的定义

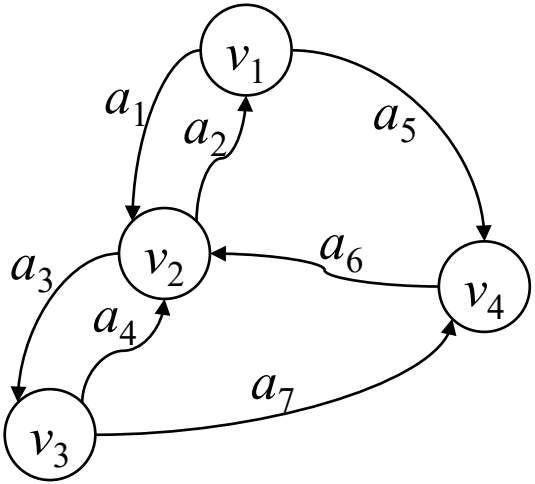
- **有向图**：G = <*V*, *A*>
  - *V*：顶点的有限集合
  - *A*：**有向边**（**弧**）的有限集合
    - 弧a_5 = <v_1, v_4>是*V*中顶点v_1和v_4组成的有序对
    - v_1和v_4分别称作a_5的**尾**和**头**，统称作**端点**
    - a_5和v_1（及v_4）互相**关联**
    - a_5是v_4的**入弧**，是v_1的**出弧**

- 一些术语
  - v_1和v_4**相邻**
  - v_1是v_4的**入邻点**，v_4是v_1的**出邻点**
  - a_1和a_4**相邻**

  - **重弧（平行弧）**
  - **自环**
  - **简单有向图**

  - a_1和a_2互为**反向弧**

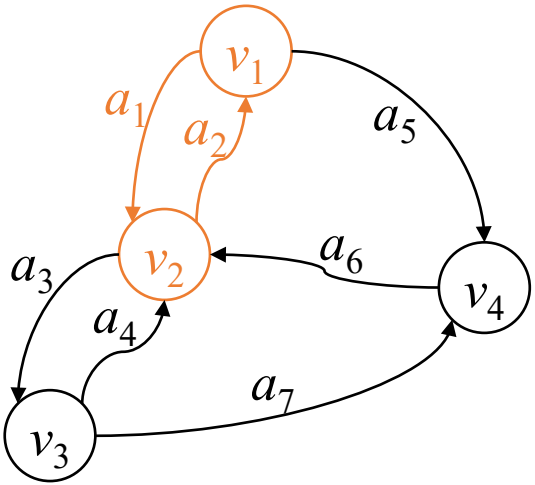
  - **阶**
  - **弧数**



有向图的定义

- **有向图**：G = <*V*, *A*>
  - *V*：顶点的有限集合
  - *A*：**有向边**（**弧**）的有限集合
    - 弧a_5 = <v_1, v_4>是*V*中顶点v_1和v_4组成的有序对
    - v_1和v_4分别称作a_5的**尾**和**头**，统称作**端点**
    - a_5和v_1（及v_4）互相**关联**
    - a_5是v_4的**入弧**，是v_1的**出弧**

- 一些术语
  - v_1和v_4**相邻**
  - v_1是v_4的**入邻点**，v_4是v_1的**出邻点**
  - a_1和a_4**相邻**
  - **重弧**（**平行弧**）
  - **自环**
  - **简单有向图**
  - a_1和a_2互为**反向弧**
  - **阶**
  - **弧数**



有向图的定义

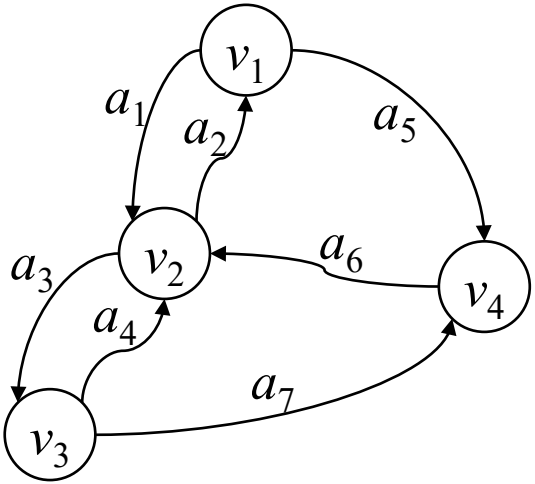
- **有向图**：G = <*V*, *A*>
  - *V*：顶点的有限集合
  - *A*：**有向边**（**弧**）的有限集合
    - 弧a_5 = <v_1, v_4>是*V*中顶点v_1和v_4组成的有序对
    - v_1和v_4分别称作a_5的**尾**和**头**，统称作**端点**
    - a_5和v_1（及v_4）互相**关联**
    - a_5是v_4的**入弧**，是v_1的**出弧**

- 一些术语
  - v_1和v_4**相邻**
  - v_1是v_4的**入邻点**，v_4是v_1的**出邻点**
  - a_1和a_4**相邻**

  - **重弧**（**平行弧**）
  - **自环**
  - **简单有向图**

  - a_1和a_2互为**反向弧**

  - **阶**
  - **弧数**



有向图的定义

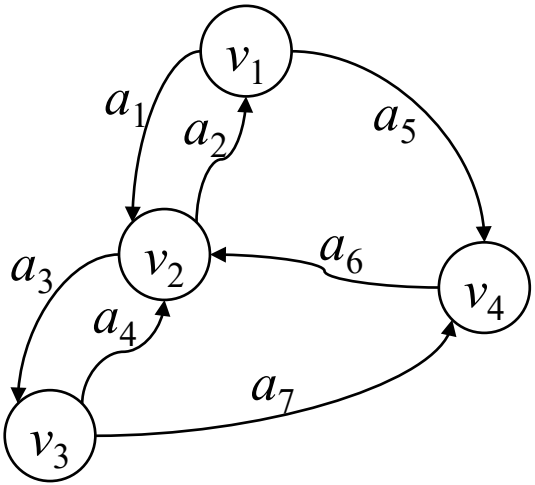
- **有向图**：G = <*V*, *A*>
  - *V*：顶点的有限集合
  - *A*：**有向边**（**弧**）的有限集合
    - 弧a_5 = <v_1, v_4>是*V*中顶点v_1和v_4组成的有序对
    - v_1和v_4分别称作a_5的**尾**和**头**，统称作**端点**
    - a_5和v_1（及v_4）互相**关联**
    - a_5是v_4的**入弧**，是v_1的**出弧**

- 一些术语
  - v_1和v_4**相邻**
  - v_1是v_4的**入邻点**，v_4是v_1的**出邻点**
  - a_1和a_4**相邻**

  - **重弧**（**平行弧**）
  - **自环**
  - **简单有向图**

  - a_1和a_2互为**反向弧**

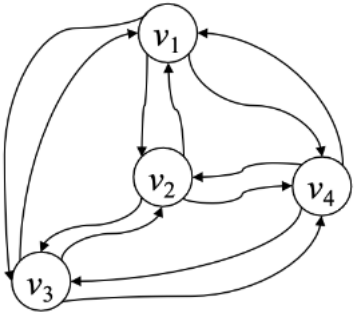
  - **阶**
  - **弧数**



阶为*n*的简单有向图的弧数的上界是多少？

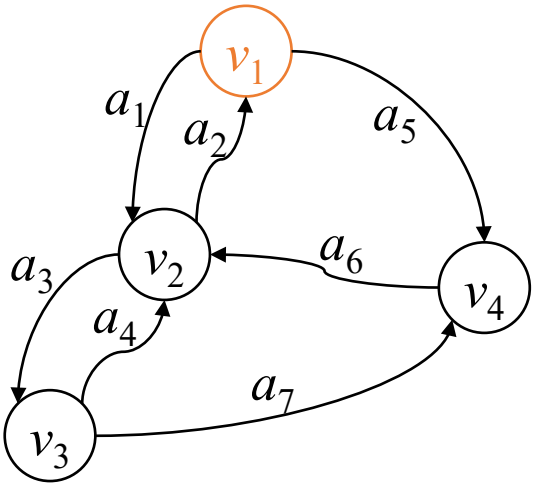
有向图的定义

- 一些术语
  - **完全有向图**



有向图的定义

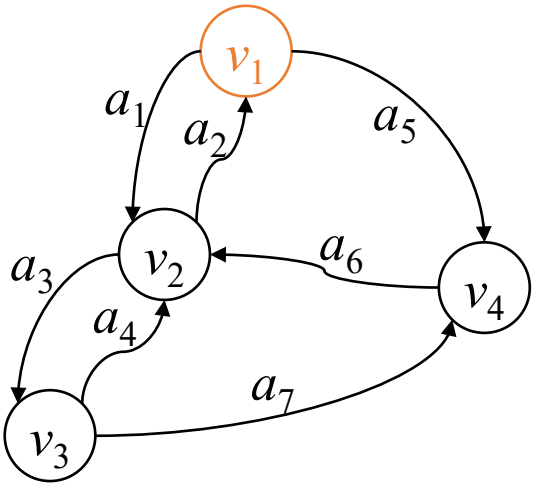
- 一些术语
  - **入度**
  - **出度**
  - **度**



有向图的定义

- 一些术语
  - **入度**
  - **出度**
  - **度**

入度和 = 出度和 = 弧数
度和 = 弧数*2

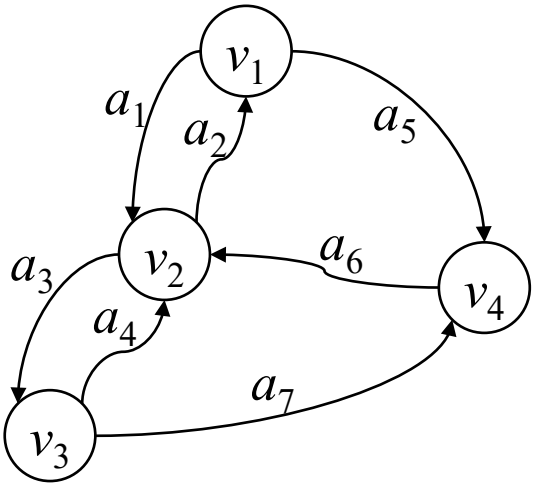


有向图的定义

- 一些术语
  - **入度**
  - **出度**
  - **度**

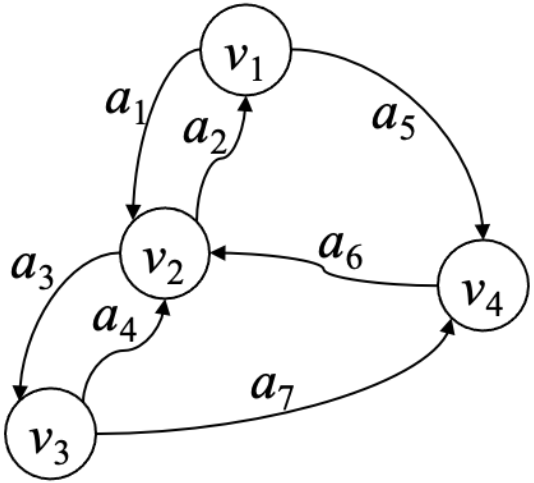
  - **入度序列**
  - **最大入度**
  - **最小入度**

  - **出度序列**
  - **最大出度**
  - **最小出度**

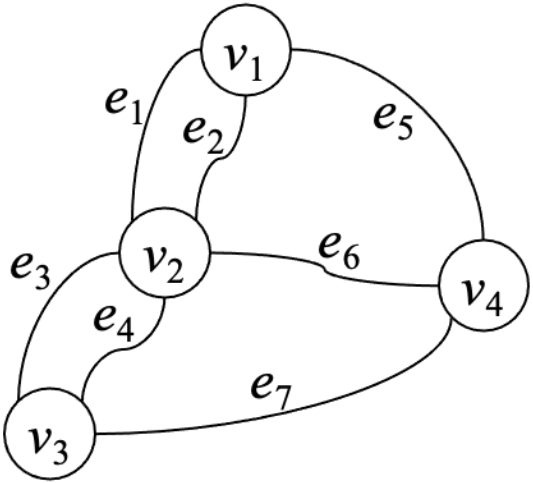


有向图的定义

- 将有向图G的每条弧（有序对）改为边（无序对）形成图H
  - H称作G的**底图**
  - G称作H的**定向**



定向

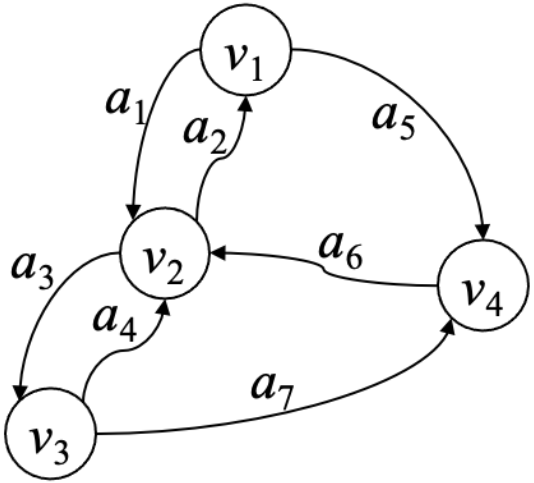


底图

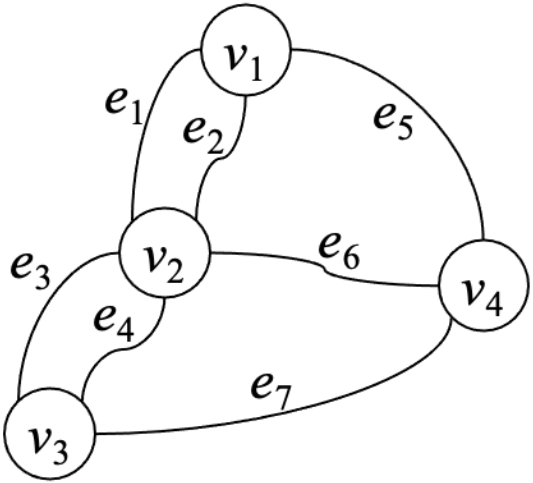
有向图的定义

- 将有向图G的每条弧（有序对）改为边（无序对）形成图H
  - H称作G的**底图**
  - G称作H的**定向**

有向图的底图唯一吗？图的定向唯一吗？



定向



底图

本次课的主要内容

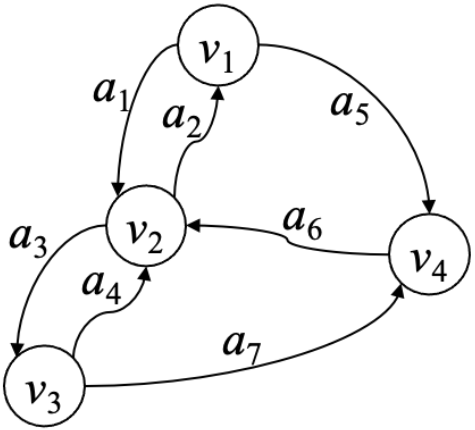
6.1 赋权图和距离
6.2 最小生成树
6.3 赋权欧拉图
6.4 赋权哈密尔顿图

7.1 有向图的定义
7.2 有向图的表示
7.3 有向图的连通
7.4 有向图的距离
7.5 流网络和最大流

有向图的表示

- **邻接矩阵**

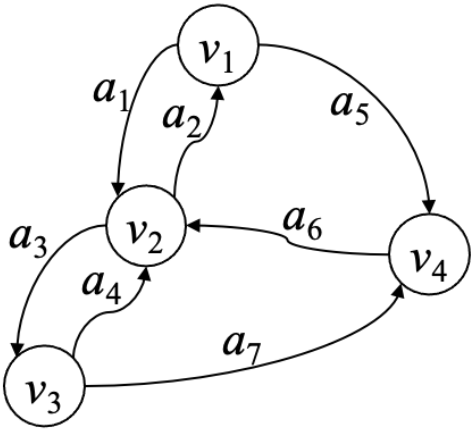
$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$



有向图的表示

- **关联矩阵**
  - 1表示出弧，-1表示入弧

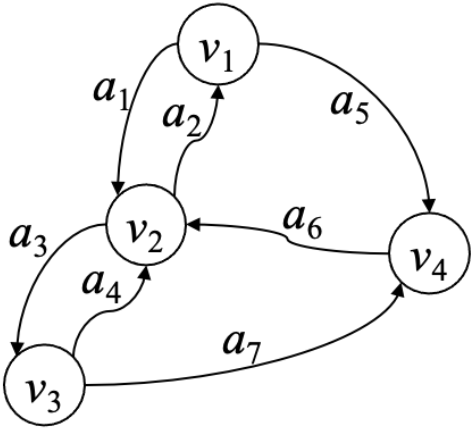
$$\begin{pmatrix} 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 \end{pmatrix}$$



有向图的表示

- **邻接表**

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



本次课的主要内容

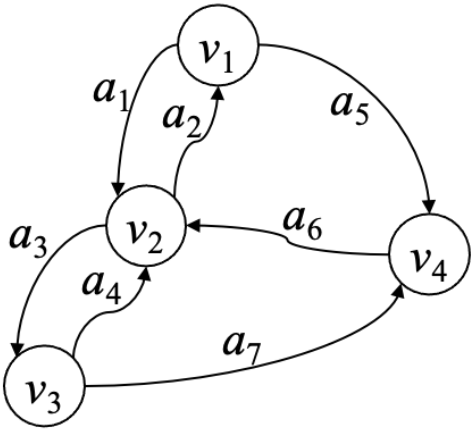
6.1 赋权图和距离
6.2 最小生成树
6.3 赋权欧拉图
6.4 赋权哈密尔顿图

7.1 有向图的定义
7.2 有向图的表示
7.3 有向图的连通
7.4 有向图的距离
7.5 流网络和最大流

有向图的连通

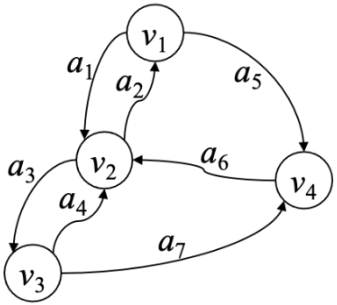
- **有向路线**
- **有向（踪）迹**
- **有向路（径）**

- **有向闭路线**
- **有向闭迹/回路**
- **有向圈**

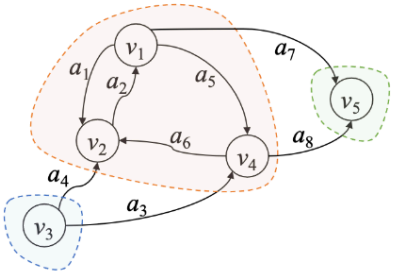


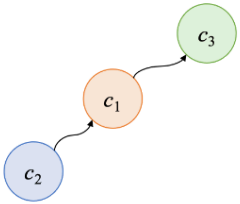
有向图的连通

- **弱连通**：底图连通
- **弱连通分支**：极大弱连通子图



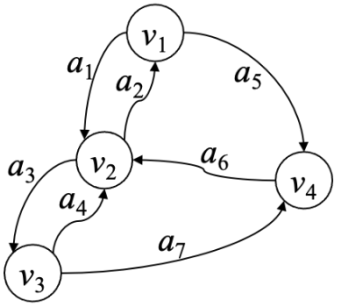
- **强连通**：每对顶点互相可达（存在有向路）
- **强连通分支**：极大强连通子图
- **浓缩**：强连通分支→顶点





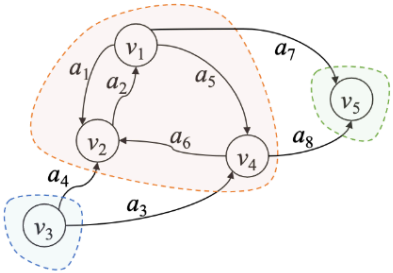
有向图的连通

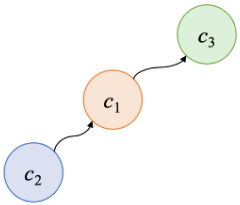
- **弱连通**：底图连通
- **弱连通分支**：极大弱连通子图



- **强连通**：每对顶点互相可达（存在有向路）
- **强连通分支**：极大强连通子图
- **浓缩**：强连通分支→顶点

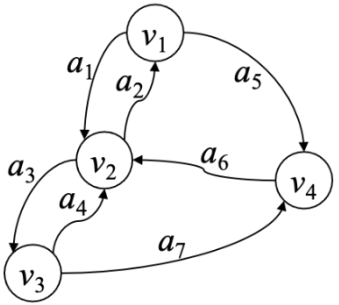
强连通图的每条弧都在某个有向圈中吗？



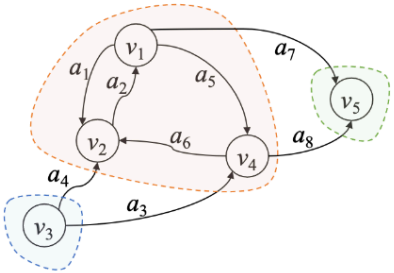


有向图的连通

- **弱连通**：底图连通
- **弱连通分支**：极大弱连通子图

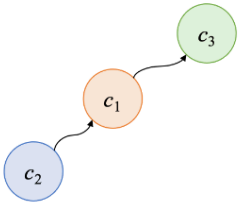


- **强连通**：每对顶点互相可达（存在有向路）
- **强连通分支**：极大强连通子图
- **浓缩**：强连通分支→顶点



强连通图的每条弧都在某个有向圈中吗？

一个顶点可以出现在图的两个强连通分支中吗？



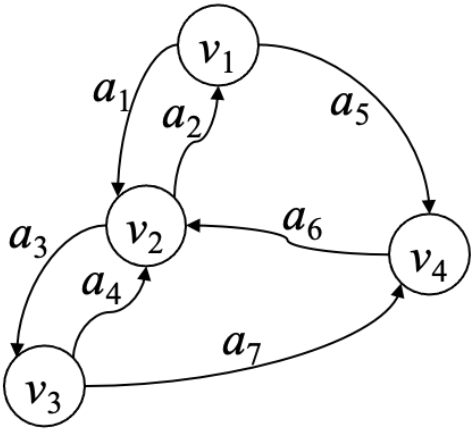
本次课的主要内容

6.1 赋权图和距离
6.2 最小生成树
6.3 赋权欧拉图
6.4 赋权哈密尔顿图

7.1 有向图的定义
7.2 有向图的表示
7.3 有向图的连通
7.4 有向图的距离
7.5 流网络和最大流

有向图的距离

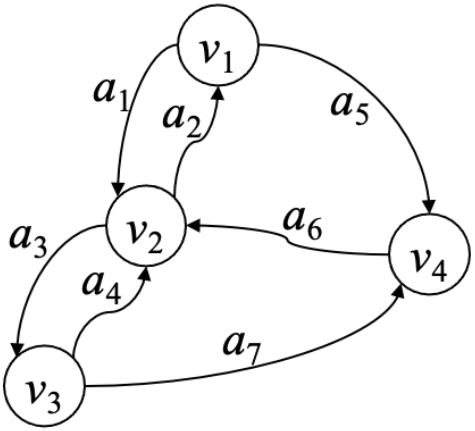
- **最短有向路**
- **距离**



有向图的距离

- 最短有向路
- 距离

- 在有向图中，
  距离满足对称性吗？
  满足三角不等式吗？



有向图的距离

- 如何计算有向图中两个顶点间的距离?
- 如何计算一个顶点和有向图中所有顶点间的距离?

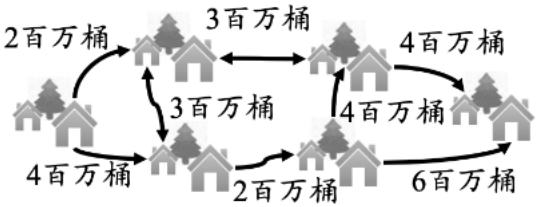
- 扩展BFS算法

本次课的主要内容

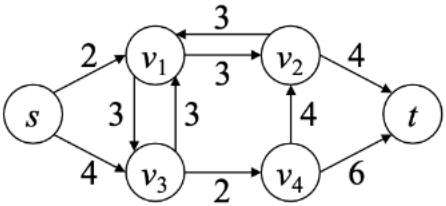
6.1 赋权图和距离
6.2 最小生成树
6.3 赋权欧拉图
6.4 赋权哈密尔顿图

7.1 有向图的定义
7.2 有向图的表示
7.3 有向图的连通
7.4 有向图的距离
7.5 流网络和最大流

流网络和最大流


2百万桶
3百万桶
4百万桶
3百万桶
4百万桶
4百万桶
2百万桶
6百万桶

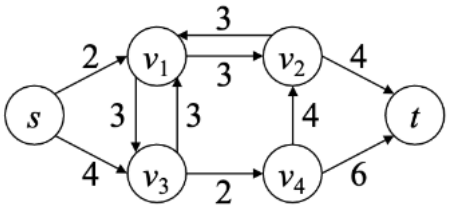
(a)


3
2
v_1
v_2
4
3
s
3
3
4
t
4
v_3
v_4
6
2

(b)

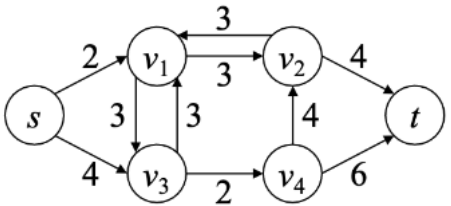
流网络和最大流

- **流网络**$G = <V, A, c, s, t>$
  - V：顶点的有限集合
  - A：弧的有限集合
  - c：$A \rightarrow R_{\geq 0}$，**容量函数**
  - $s \in V$：**源**
  - $t \in V$：**汇**

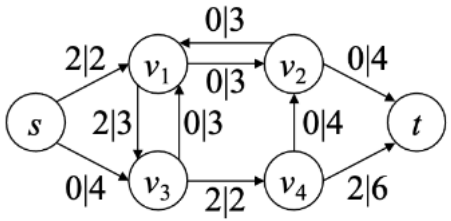


流网络和最大流

- 流网络$G = <V, A, c, s, t>$
  - V：顶点的有限集合
  - A：弧的有限集合
  - c：$A \rightarrow R_{\geq 0}$，容量函数
  - $s \in V$：源
  - $t \in V$：汇



- **流**
  - f：$A \rightarrow R$
  - $f(a)$：弧a的**流量**
- **零流**
  - 所有弧的流量均为0



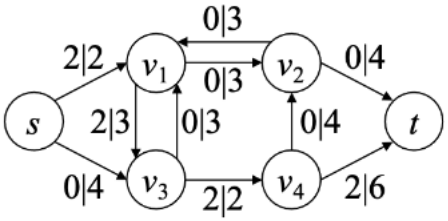
流网络和最大流

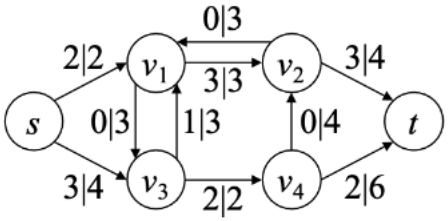
- **可行流**
  - **容量约束**

    $\forall a \in A,\ 0 \le f(a) \le c(a)$
  - **守恒约束**

    $\forall v \in V \setminus \{s,t\},\ f^-(v) = f^+(v)$

    流入量（所有入弧的流量和）= 流出量（所有出弧的流量和）





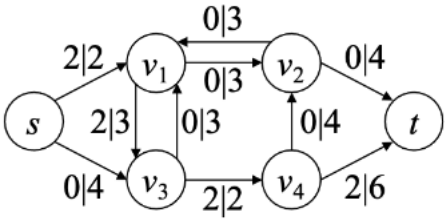
流网络和最大流

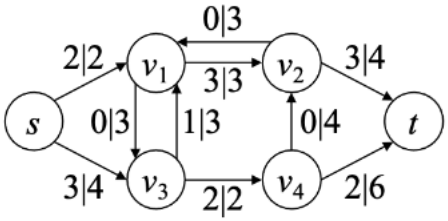
- 可行流
  - 容量约束

    $\forall a \in A,\ 0 \le f(a) \le c(a)$
  - 守恒约束

    $\forall v \in V \setminus \{s,t\},\ f^-(v) = f^+(v)$

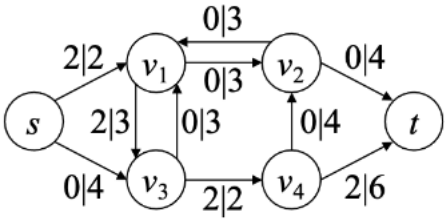
    流入量（所有入弧的流量和）= 流出量（所有出弧的流量和）
- 每个流网络都有可行流吗？

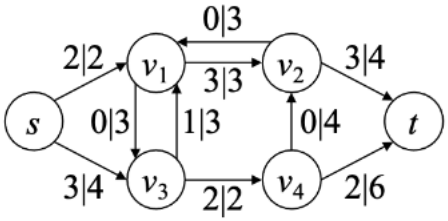




流网络和最大流

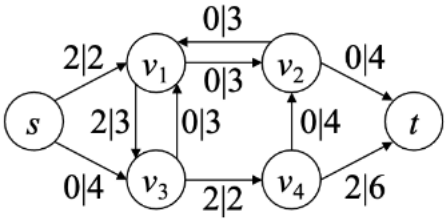
- 流的**值**
  - 源的净流出量$f^{+}(s)-f^{-}(s)$ = 汇的净流入量$f^{-}(t)-f^{+}(t)$

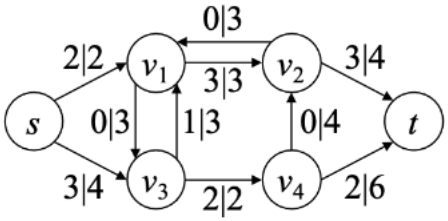




流网络和最大流

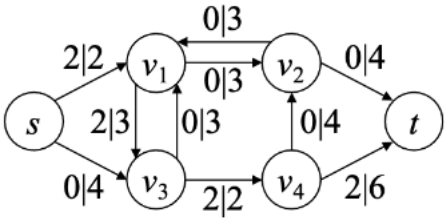
- 流的值
  - 源的净流出量$f^{+}(s)-f^{-}(s)$ = 汇的净流入量$f^{-}(t)-f^{+}(t)$
- **最大流**
  - 值最大的可行流

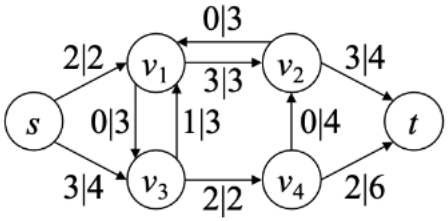




流网络和最大流

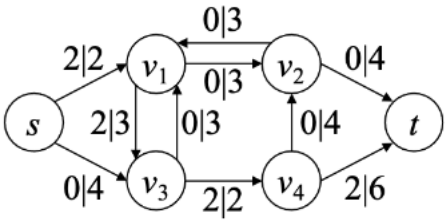
- 流的值
  - 源的净流出量$f^{+}(s)-f^{-}(s)$ = 汇的净流入量$f^{-}(t)-f^{+}(t)$
- 最大流
  - 值最大的可行流
- 如何判定最大流?

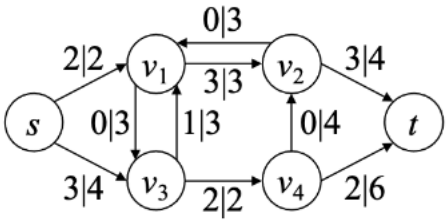




流网络和最大流

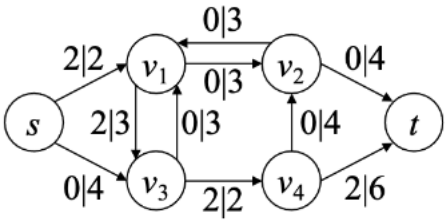
- **剩余容量**（流量的可增量）
  - r: $V \times V \rightarrow R_{\geq 0}$
    $r(u, v) = (c(<u, v>) - f(<u, v>)) + f(<v, u>)$

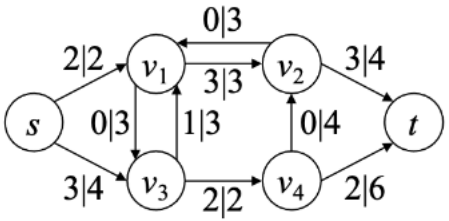


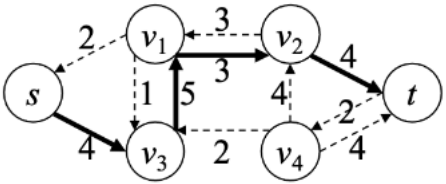


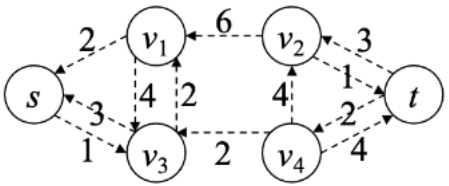
流网络和最大流

- 剩余容量（流量的可增量）
  - r: $V \times V \rightarrow \mathbb{R}_{\geq 0}$
    $r(u, v) = (c(<u, v>) - f(<u, v>)) + f(<v, u>)$
- **剩余网络**$G_f = <V, A_f, r>$
  - V: G的顶点集
  - A_f: $<u, v> \in A_f$当且仅当$r(u, v) > 0$



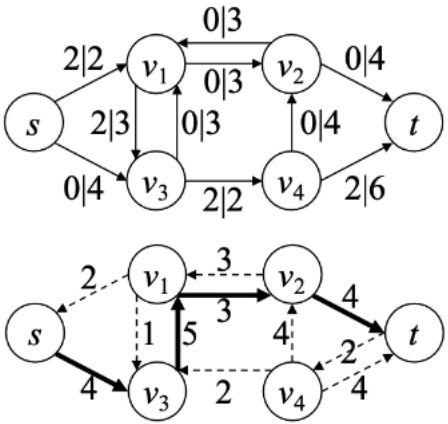


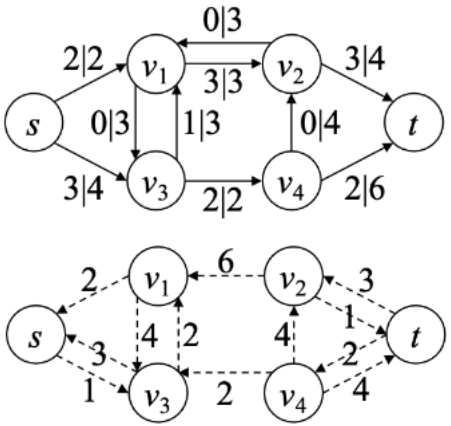




流网络和最大流

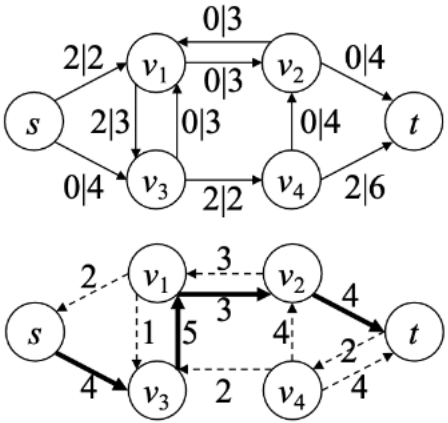
- ***f*增广路**
  - 剩余网络中的*s*-*t*有向路

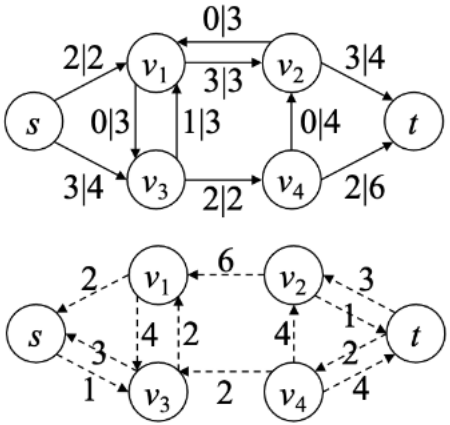




流网络和最大流

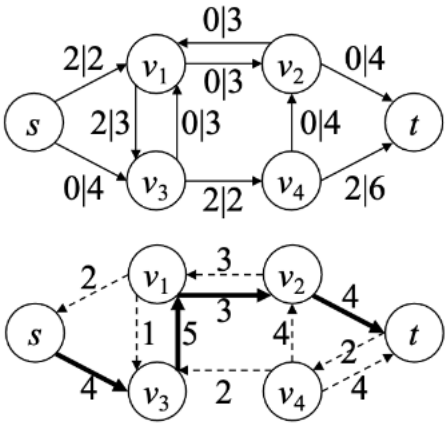
- *f*增广路
  - 剩余网络中的*s*-*t*有向路
  - **可增量**：经过的弧的最小权

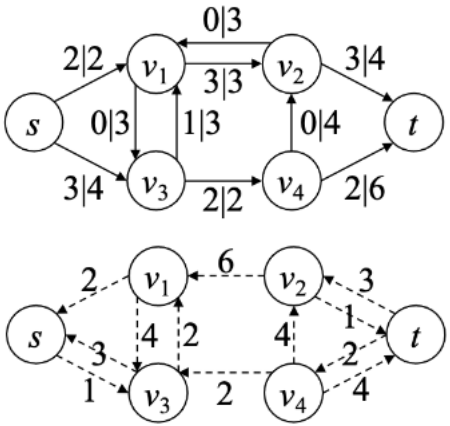




流网络和最大流

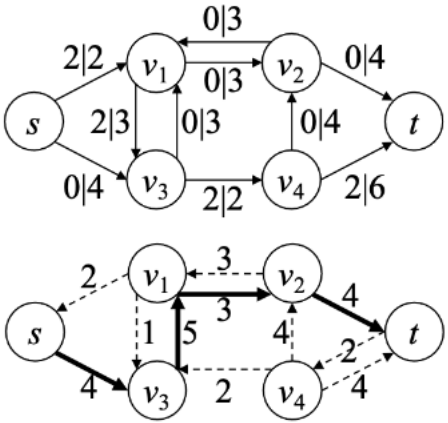
- *f*增广路
  - 剩余网络中的*s*-*t*有向路
  - 可增量：经过的弧的最小权
- 如何利用增广路得到一个值更大的流？

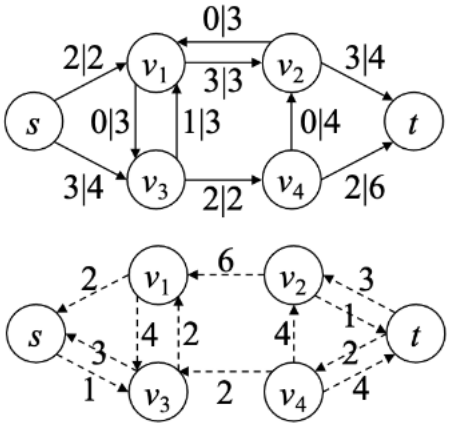




流网络和最大流

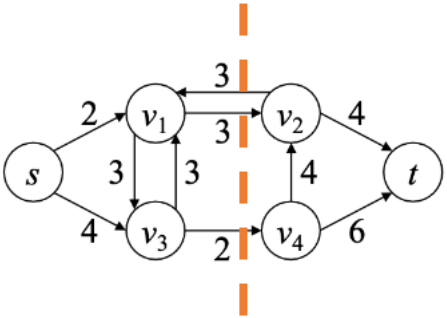
- f增广路
  - 剩余网络中的s-t有向路
  - 可增量：经过的弧的最小权
- 如何利用增广路得到一个值更大的流？
- f是最大流当且仅当G_f不含f增广路





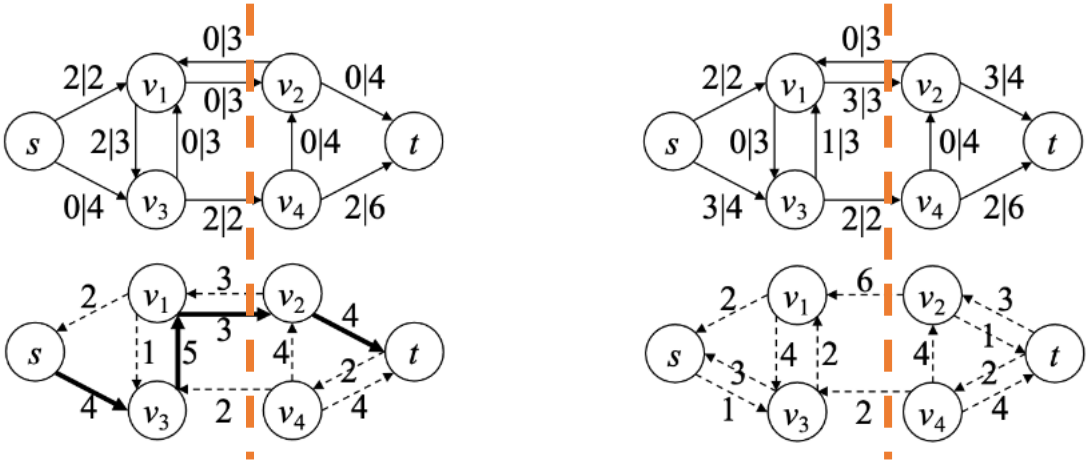
流网络和最大流

- 源汇割
  - 对于顶点集V的一种满足$s \in S$和$t \in T$的划分S和T，所有尾在S中、头在T中的弧的集合称作**源汇割**，记作$C_{S,T}$其中所有弧的容量和称做$C_{S,T}$的**容量**，记作$c(S, T)$



流网络和最大流

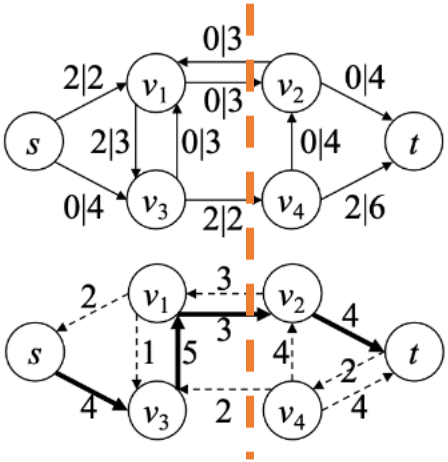
- f是最大流当且仅当G_f不含f增广路
  - $f^+(s) - f^-(s) = \sum_{v \in S} (f^+(v) - f^-(v)) = f^+(S) - f^-(S)$
    - $f^+(S)$：所有尾在S中、头在T中的弧的流量和
    - $f^-(S)$：所有头在S中、尾在T中的弧的流量和
  - $val(f) = f^+(S) - f^-(S) \le f^+(S) \le c(S, T)$
  - 当G_f不含f增广路时，$val(f) = c(S, T)$

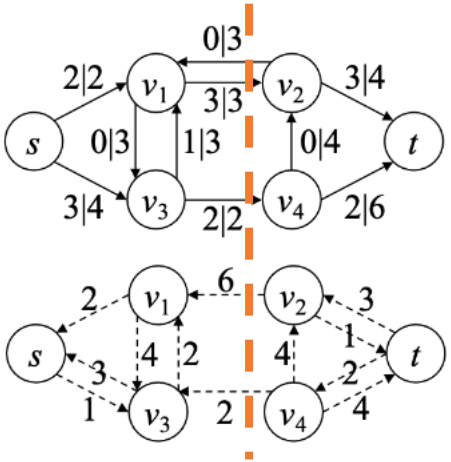


流网络和最大流

- f是最大流当且仅当G_f不含f增广路
  - $f^+(s) - f^-(s) = \sum_{v \in S} (f^+(v) - f^-(v)) = f^+(S) - f^-(S)$
    - $f^+(S)$：所有尾在S中、头在T中的弧的流量和
    - $f^-(S)$：所有头在S中、尾在T中的弧的流量和
  - $val(f) = f^+(S) - f^-(S) \le f^+(S) \le c(S, T)$
  - 当G_f不含f增广路时，$val(f) = c(S, T)$

最大流的值 = 源汇割容量的最小值



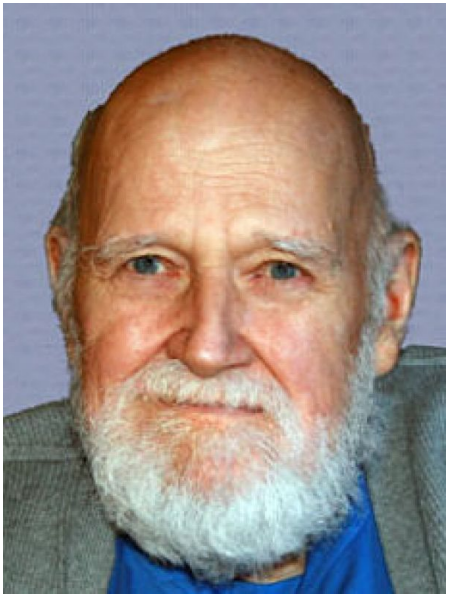


流网络和最大流

- 如何计算流网络中的最大流?

流网络和最大流

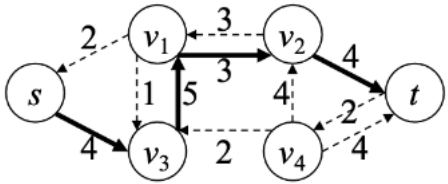
- Lester Randolph Ford Jr.，1927年出生于美国
- Delbert Ray Fulkerson，1924年出生于美国



https://www.independent.com/wp-content/uploads/2019/03/dadd_opt.jpg?resize=600,800
https://en.wikipedia.org/wiki/D._R._Fulkerson

流网络和最大流

- 福特-法尔克森算法
  - 逐步构造最大流，每步利用当前流的一条增广路得到一个值更大的流



流网络和最大流

- 福特-法尔克森算法

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: f 初值为零流

```
1  while G_f 含 f 增广路 P do
2      r ← min_{⟨u,v⟩∈P经过的所有弧} r(u, v);
3      foreach ⟨u, v⟩ ∈ P经过的所有弧 do
4          if ⟨v, u⟩ ∈ A then
5              r' ← min{f(⟨v, u⟩), r};
6              f(⟨v, u⟩) ← f(⟨v, u⟩) − r';
7              r ← r − r';
8          if ⟨u, v⟩ ∈ A then
9              f(⟨u, v⟩) ← f(⟨u, v⟩) + r;
10 输出 (f);
```

流网络和最大流

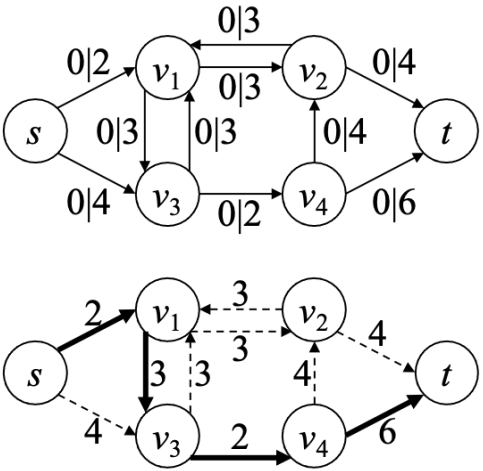
- 福特-法尔克森算法

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: 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);
```



流网络和最大流

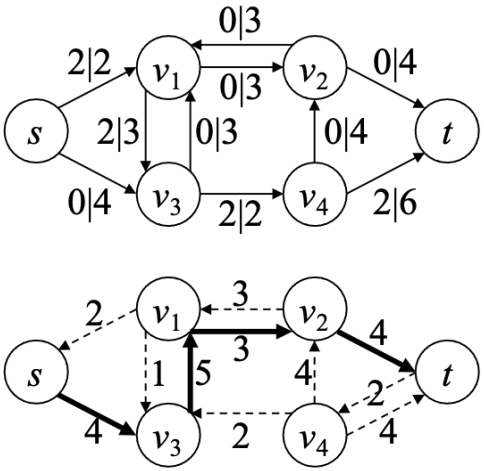
- 福特-法尔克森算法

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: 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);
```



流网络和最大流

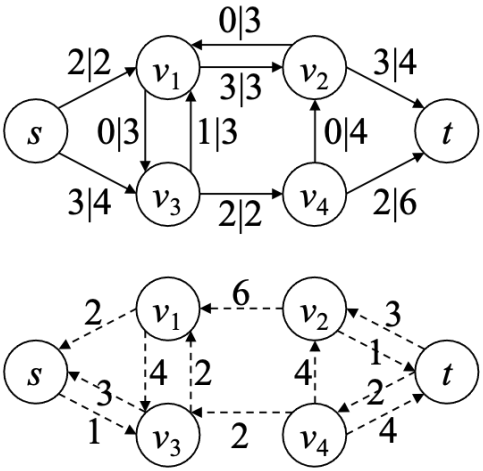
- 福特-法尔克森算法

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: 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);
```

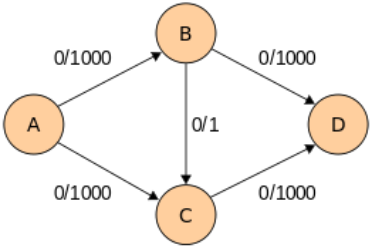


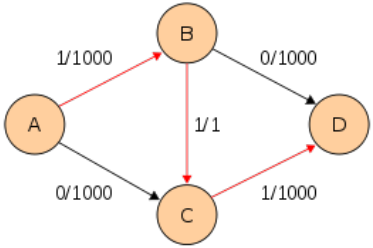
流网络和最大流

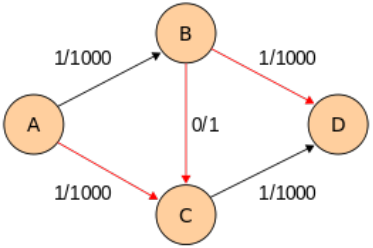
- 当弧的容量包含无理数时，算法有可能无法运行结束!
- 当所有弧的容量都是有理数时，为什么算法一定会运行结束?

流网络和最大流

- 时间复杂度：$O((n+m)\mathrm{V})$
  - 每轮while循环：$O(n+m)$
  - while循环的轮数：$O(\mathrm{V})$



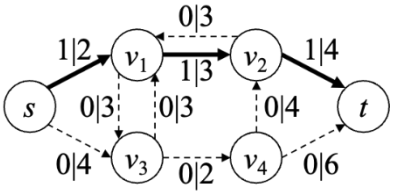


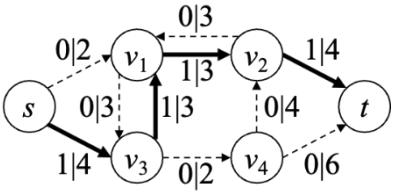


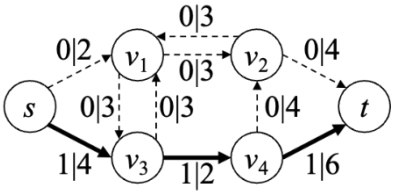
https://upload.wikimedia.org/wikipedia/commons/thumb/f/f1/Ford-Fulkerson_example_0.svg/300px-Ford-Fulkerson_example_0.svg.png

流网络和最大流

- **整数流**：所有弧的流量都是整数
- **单位路径流**

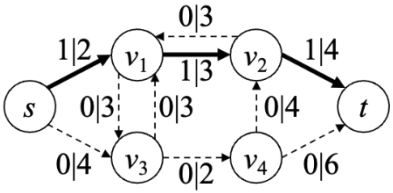


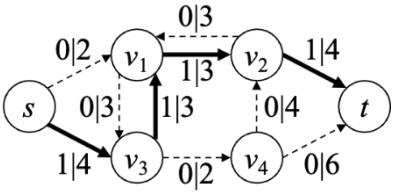


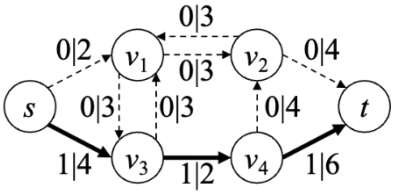


流网络和最大流

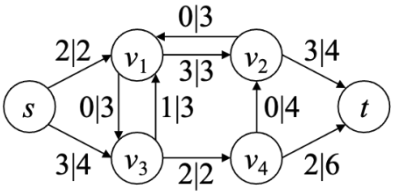
- 整数流：所有弧的流量都是整数
- 单位路径流





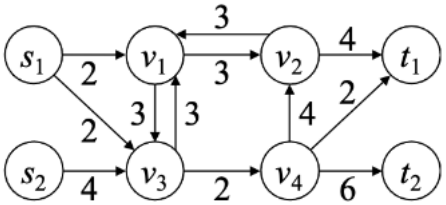


- 对于所有弧的容量都是整数的流网络：
  最大流的值是整数，存在一个最大流是整数流，
  且该最大流可表示为若干单位路径流的和。

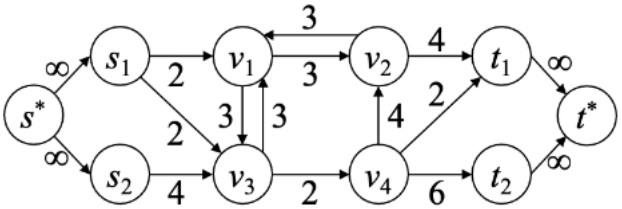


流网络和最大流

- 多源多汇流网络



- 超源、超汇



书面作业

- 练习6.5
- 练习7.1
- 练习7.17、7.18、7.22