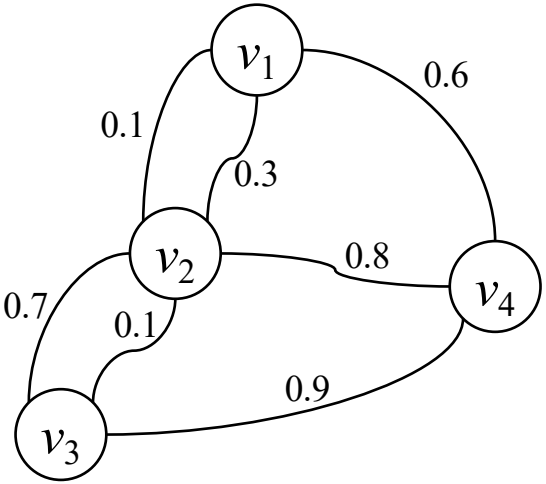
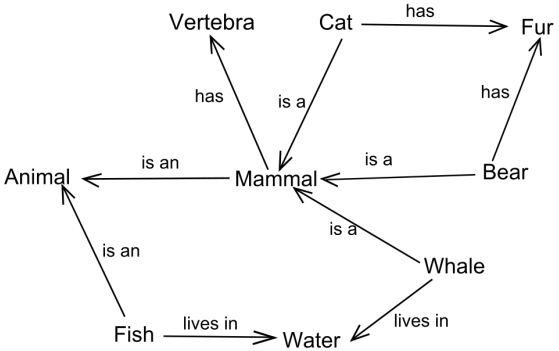
网络

程龚

从图到网络

- **网络**：顶点或边带有标记的图
  - 标记：数值、文本等任意属性

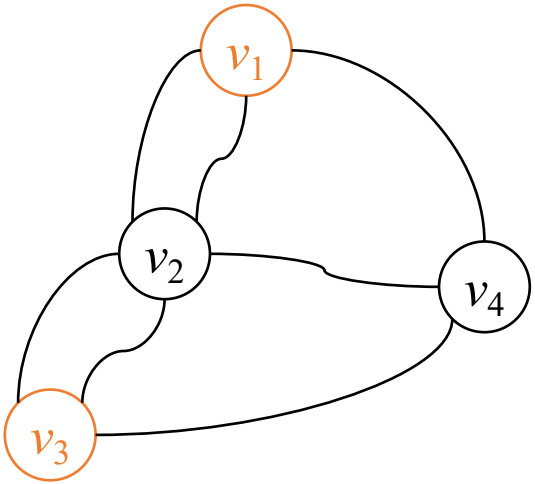


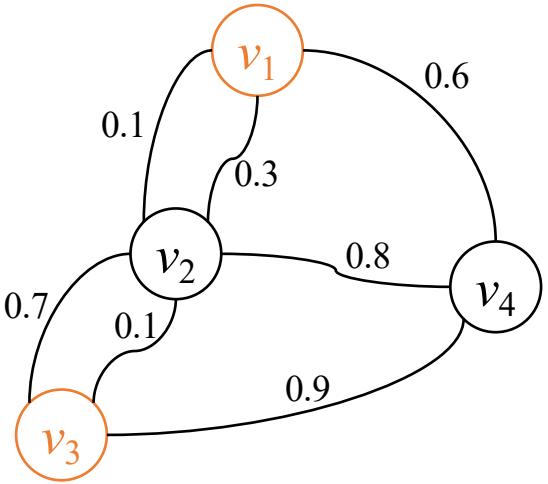


http://upload.wikimedia.org/wikipedia/commons/thumb/6/67/Semantic_Net.svg/1000px-Semantic_Net.svg.png

从图到网络（举例）

- 路的长度
  - 图：经过的边的数量
  - 网络：经过的边的权和
- 距离的计算
  - 图：BFS算法
  - 网络：Dijkstra算法





本次课的主要内容

高随祥《图论与网络流理论》

4.2 中国邮递员问题
4.4 旅行商问题
9.1 网络与网络流的基本概念
9.2 最大流问题及其标号算法

本次课的主要内容

高随祥《图论与网络流理论》

4.2 中国邮递员问题

4.4 旅行商问题

9.1 网络与网络流的基本概念

9.2 最大流问题及其标号算法

中国邮递员问题

- 管梅谷，1934-，出生于中国

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

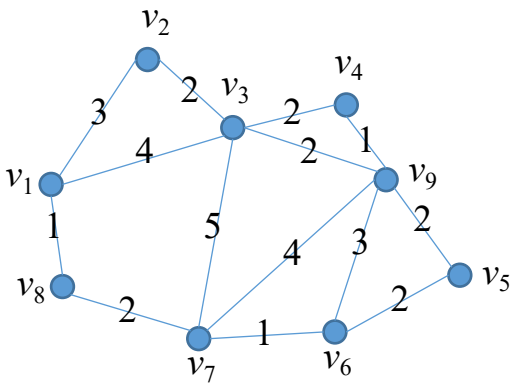
每段路上都有一个邮筒，邮递员从邮局出发，处理每个邮筒后返回邮局，走过的总路程最短。

你能用图来为这个问题建立数学模型吗？

https://map.baidu.com/

中国邮递员问题

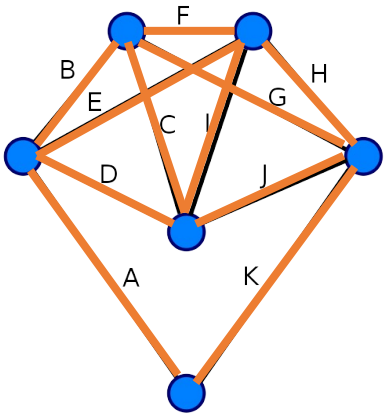
- **最优邮路**：边带权的连通网络中，经过所有边且权和最小的闭路线



欧拉图（复习）

- 欧拉迹：经过图的每条边恰一次的迹
- 欧拉回路：经过图的每条边恰一次的闭迹（回路）
- 欧拉图：含欧拉回路的图

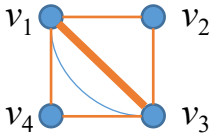
- 欧拉图的最优邮路是什么？
  - 欧拉回路



http://upload.wikimedia.org/wikipedia/commons/thumb/7/72/Labelled_Eulergraph.svg/500px-Labelled_Eulergraph.svg.png

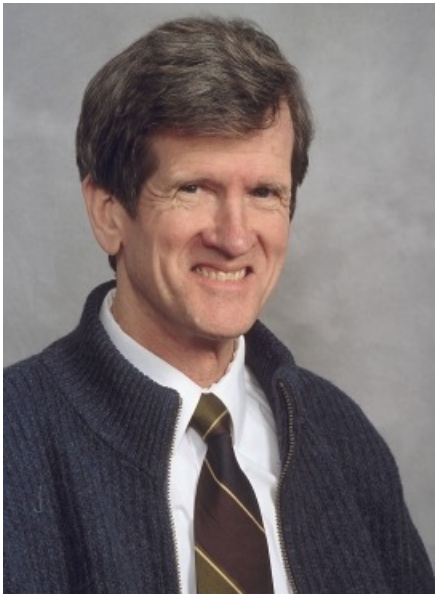
中国邮递员问题的转换

- 非欧拉图的最优邮路，有什么特征？
  - 必然要重复经过一些边。
  - 将重复走过的边作为重边添加到图中 ⇒ 新图一定是欧拉图吗？
  - 原图中的最优邮路，与新图中的欧拉回路，有什么联系？
- 据此，你能将中国邮递员问题转换为一个新的问题吗？
  1. 添加重边成为欧拉图（如果本来不是的话）。
  2. 使添加的边权和最小。
  3. 找欧拉回路。



Edmonds-Johnson算法

- Jack Edmonds，1934-，出生于美国
- Ellis L. Johnson，1938-，出生于美国



https://en.wikipedia.org/wiki/Jack_Edmonds
https://www.isye.gatech.edu/sites/default/files/emp-profiles/johnson_ellis_-_bust.jpg

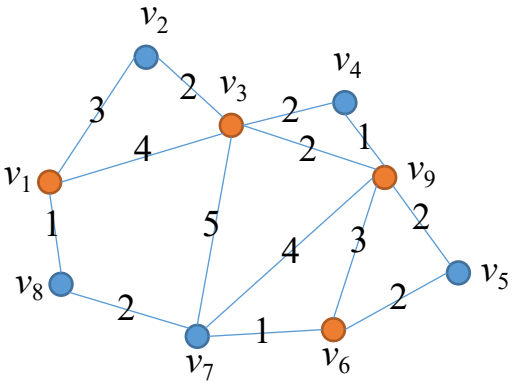
1. 添加重边成为欧拉图

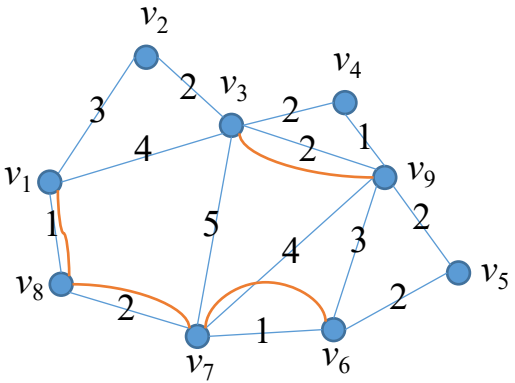
- 非空连通图G含欧拉回路当且仅当G没有顶点的度为奇数。

  ⇒ 添加重边的目标是什么？
  - 消除奇度顶点。

1. 添加重边成为欧拉图

- 如何添加重边，才能消除奇度顶点？
  - 奇度顶点 → 偶度顶点
  - 偶度顶点 → 偶度顶点





2. 使添加的边权和最小

- 设G是边带权的连通网络，G中有$2k$个奇度顶点。G^*是G的最优邮路对应的欧拉图，令$E' = E_{G^*} \setminus E_G$。则$H = G[E']$是以$G$的奇度顶点为起点和终点的$k$条无公共边的最短路之并。

  ⇒ 重边的添法？

  - 连接k对奇度顶点的k条无公共边的最短路。
  - 且边权和最小。

2. 使添加的边权和最小

- 设G是边带权的连通网络，G中有$2k$个奇度顶点。G^*是G的最优邮路对应的欧拉图，令$E' = E_{G^*} \setminus E_G$。则$H = G[E']$是以$G$的奇度顶点为起点和终点的$k$条无公共边的最短路之并。

证明：

1. H有$2k$个奇度顶点，为什么？

   G^*是欧拉图 ⇒ G^*无奇度顶点 ⇒ E'中关联到G的每个奇/偶度顶点的边有奇/偶数条 ⇒
   G中的奇度顶点在H中仍是奇度顶点，G中的偶度顶点在H中仍是偶度顶点或不出现 ⇒ 得证

2. H中没有圈，为什么？

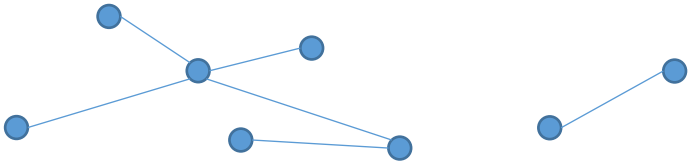
   假设H中有圈C ⇒ C在G^*中 ⇒ G^*中删去C后无奇度顶点 ⇒ 是欧拉图且边权和比G^*小 ⇒
   G^*不是最优邮路对应的欧拉图 ⇒ 矛盾 ⇒ 得证

3. 从H的任一个奇度顶点出发沿未走过的边前行直到无法继续，形成迹P_1 ⇒
   - P_1的终点是奇度顶点，为什么？
   - P_1是路，为什么？
   - P_1是最短路，为什么？

   否则 ⇒ 在G^*中用更短的路替换P_1 ⇒ 得到边权和更小的欧拉图 ⇒ G^*不是最优邮路对应的欧拉图 ⇒ 矛盾

4. 从H去掉P_1，剩余$2k$-2个奇度顶点，同理可找到P_2、P_3、……、P_k。

5. 剩余图中无圈无奇度顶点 ⇒ 无边 ⇒ H是以G的奇度顶点为起点和终点的k条无公共边的最短路之并



2. 使添加的边权和最小

- 设G是边带权的连通网络，G中有$2k$个奇度顶点。G^*是G的最优邮路对应的欧拉图，令$E' = E_{G^*} \setminus E_G$。则$H = G[E']$是以$G$的奇度顶点为起点和终点的$k$条无公共边的最短路之并。

  ⇒ 重边的添法？
  - 连接k对奇度顶点的k条无公共边的最短路。
  - 且边权和最小。

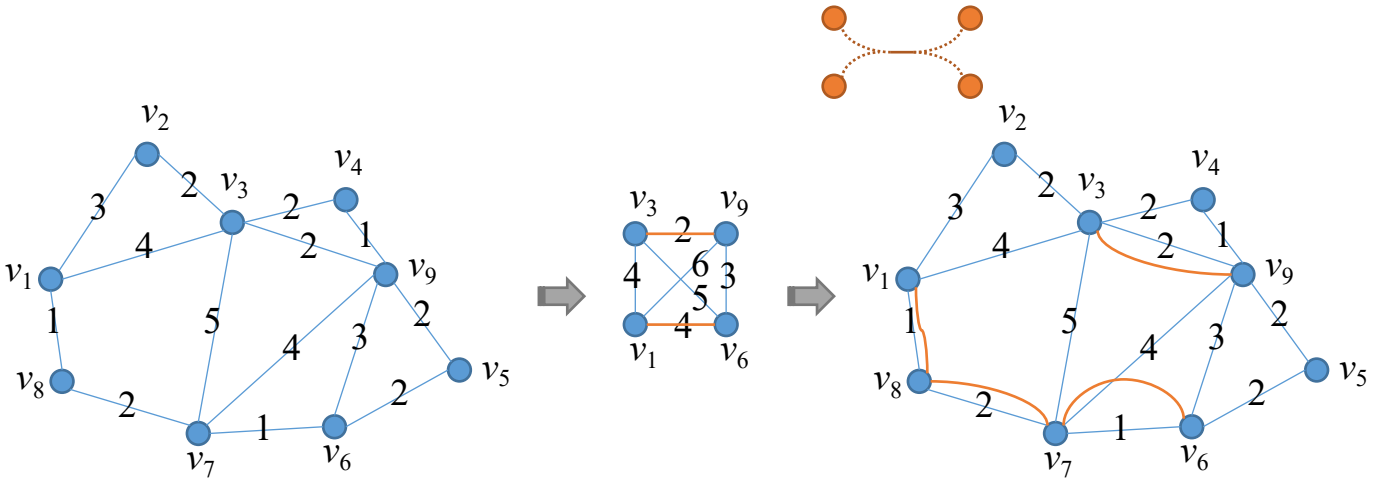
这里的关键是要确定一件什么事情？这让你联想到我们学过的哪个概念？

顶点如何配对 → 匹配

2. 使添加的边权和最小

- Edmonds-Johnson算法的关键步骤
  1. 找到所有$2k$个奇度顶点间的最短路。
  2. 构造一个完全图K_{2k}：顶点为$2k$个奇度顶点，边权为最短路的边权和。
  3. 找K_{2k}的最小权完美匹配M。
  4. 沿M对应的最短路添加重边。

为什么这k条路一定恰没有公共边？
删去公共边后，可得到权更小的完美匹配 ⇒ 矛盾



3. 找欧拉回路

- 弗勒里算法
- 希尔霍尔策算法

Edmonds-Johnson算法的时间复杂度

1. 找两两最短路
   - Floyd-Warshall算法：$O(n^3)$
2. 构造完全图：$O(n^2)$
3. 找最小权完美匹配
   - 基于花算法：$O(n^3)$
4. 添加重边：$O(m)$
5. 找欧拉回路
   - 希尔霍尔策算法：$O(n + m)$

本次课的主要内容

高随祥《图论与网络流理论》

4.2 中国邮递员问题

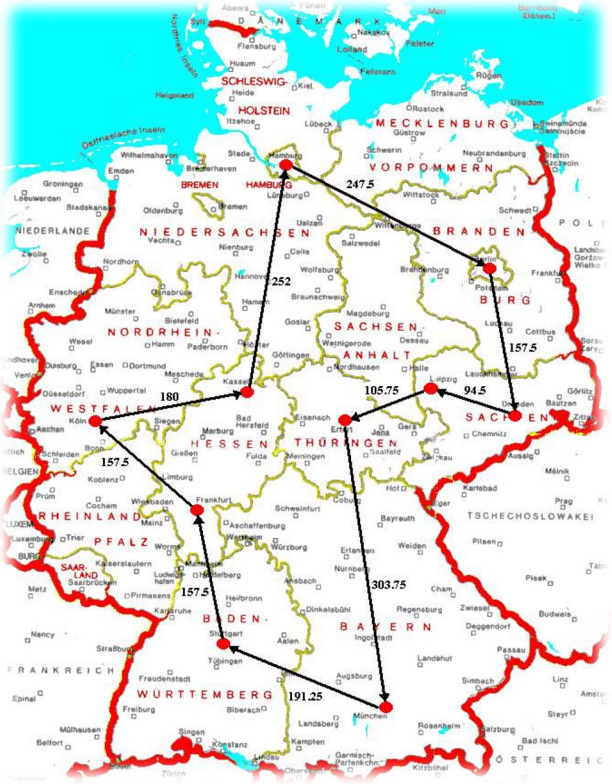
4.4 旅行商问题

9.1 网络与网络流的基本概念

9.2 最大流问题及其标号算法

旅行商经过每座城市恰一次后返回起点，经过的总路程最短。

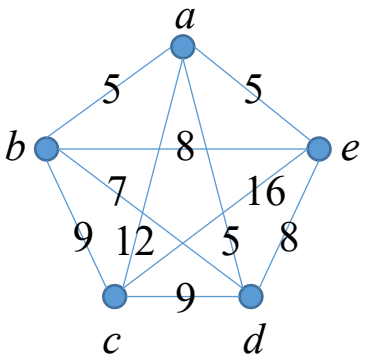
你能用图来为这个问题建立数学模型吗？

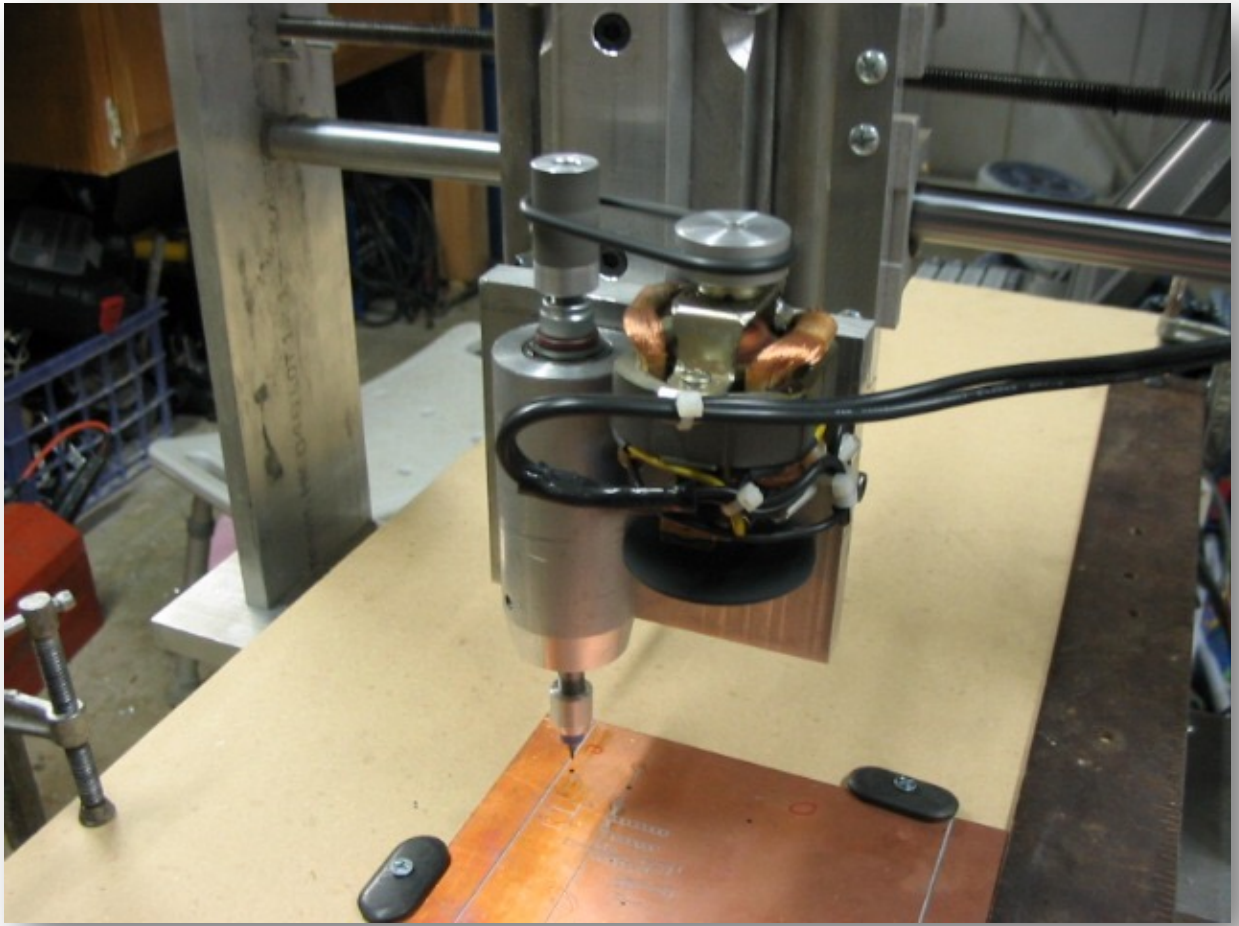


http://blog.electricbricks.com/wp-content/uploads/TSP.jpg

旅行商问题

- 边带权的连通网络中，权和最小的哈密尔顿圈
  - 通常假设：$\forall v_i, v_j, v_k \in V$, $\mathrm{w}(v_i, v_j) + \mathrm{w}(v_j, v_k) \geq \mathrm{w}(v_i, v_k)$
  - 通常只讨论边权为正数的完全图K_n（缺失的边的权设为∞）





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

哈密尔顿图（复习）

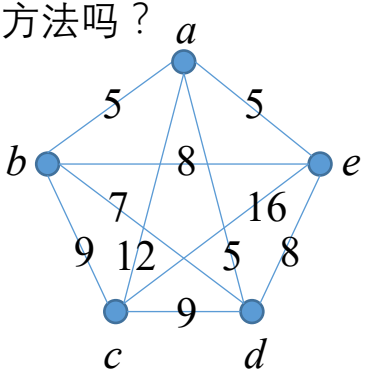
- 哈密尔顿路：经过图的每个顶点的路
- 哈密尔顿圈：经过图的每个顶点的圈
- 哈密尔顿图：含哈密尔顿圈的图

旅行商问题的难度

- 任意图中，哈密尔顿圈的存在性的判定问题：NPC
- 完全图中，找权和最小的哈密尔顿圈：NP难

前者如何规约到后者？ **随堂小测**

- 因此，通常采用近似算法
  - 高效地找出较优解
  - 你自己能想出一些方法吗？



旅行商问题的近似算法

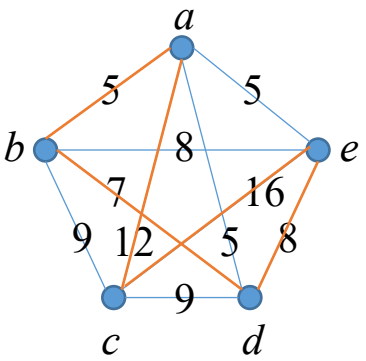
- 邻近点法
- 最小生成树法
- 最小权匹配法
- Kernighan-Lin

邻近点法

- 基本思路
  - 总是贪心地选择最近的未访问邻点前行。

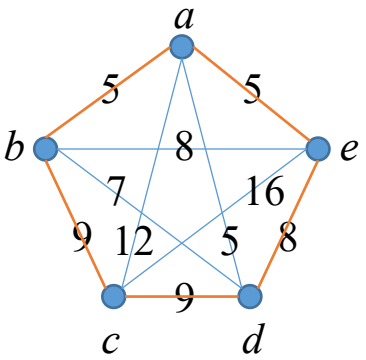
邻近点法

- 举例（从a出发）
  - 5+7+8+16+12=48



邻近点法

- 举例（从b出发）
  - 5+5+8+9+9=36



邻近点法

- 近似比$\mathrm{w}(H) / \mathrm{w}(H^*) \leq \frac{1}{2}\lceil \log_2 n \rceil$

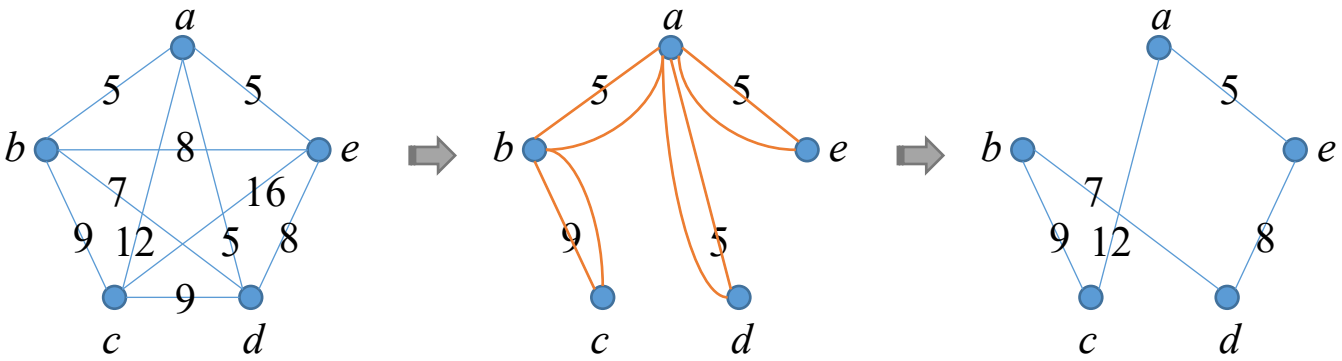
- 最终结果与以下因素有关
  - 初始点

邻近点法

- 时间复杂度：$O(n^2)$

最小生成树法

1. 找K_n的一棵最小生成树T。
2. 为T中的每条边添加重边成为T^*。
3. 找T^*的一条欧拉回路C。
4. 按C经过顶点的顺序构造哈密尔顿圈。



最小生成树法

- 近似比$w(H) / w(H^*) < 2$

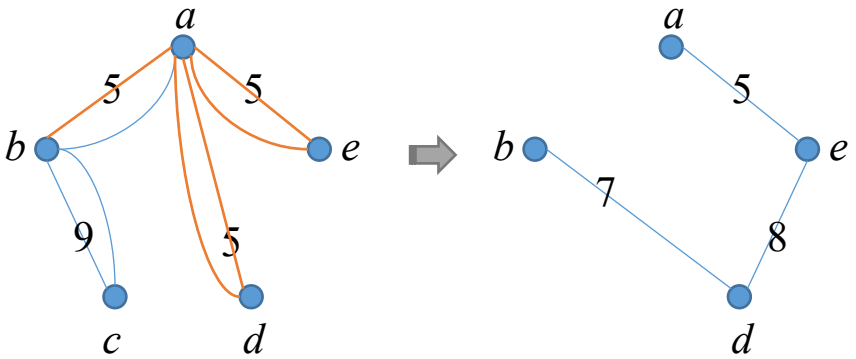
- 最终结果与以下因素有关
  - 最小生成树
  - 欧拉回路
  - 初始点

最小生成树法

- 近似比$w(H) / w(H^*) < 2$

证明：

1. 三角不等式 $\Rightarrow w(H) \le w(C) = w(T^*) = 2w(T)$
2. $w(H^*) > w(T)$，为什么？
   - H^*比生成树多一条边（这就是想到用最小生成树的原因）
3. $w(H) / w(H^*) < 2$

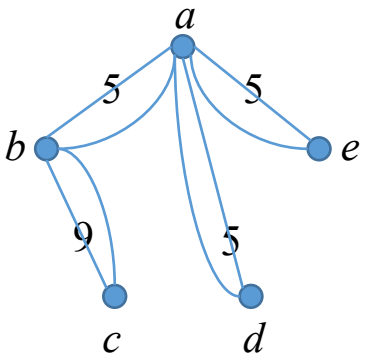


最小生成树法

- 时间复杂度
  1. 找最小生成树
     - Prim算法：$O(m + n \log n)$
  2. 添加重边：$O(n)$ //生成树的边数为$n - 1$
  3. 找欧拉回路
     - 希尔霍尔策算法： $O(n)$
  4. 沿欧拉回路前行：$O(n)$

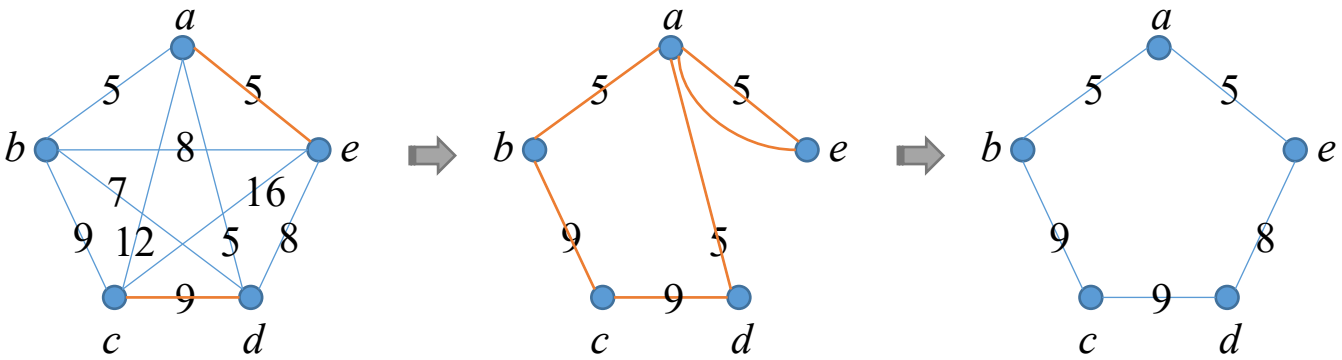
最小生成树法

- 你能根据刚才的证明，优化这个算法吗？（提示：从T到T^*需要添加那么多边吗？）
  - $w(H) \le w(C) = w(T^*) = 2w(T)$



最小权匹配法

1. 找K_n的一棵最小生成树T。
2. 找T中奇度顶点在K_n中导出子图G'的最小权完美匹配M。
3. 将M添加到T中成为T^*。
4. 找T^*的一条欧拉回路C。
5. 按C经过顶点的顺序构造哈密尔顿圈。



最小权匹配法

- 近似比$w(H) / w(H^*) < 3/2$

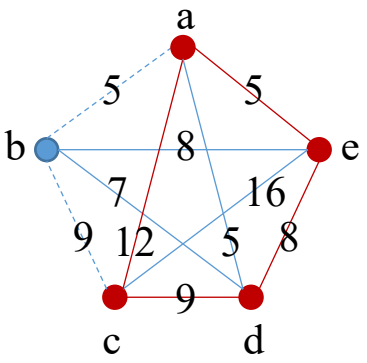
- 最终结果与以下因素有关
  - 最小生成树
  - 最小权完美匹配
  - 欧拉回路
  - 初始点

最小权匹配法

- 近似比$w(H) / w(H^*) < 3/2$

证明：

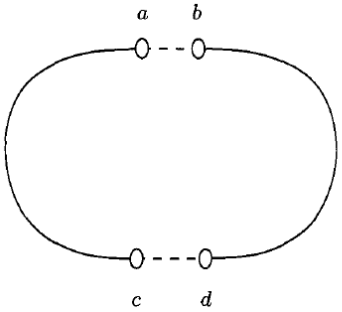
1. 三角不等式 $\Rightarrow w(H) \le w(C) = w(T^*) = w(T) + w(M)$
2. H^*比生成树多一条边 $\Rightarrow w(H^*) > w(T)$
3. 三角不等式 $\Rightarrow w(H') \le w(H^*)$，为什么？（$H'$是$G'$中的某个哈密尔顿圈）
   - H'可由H^*反复“去二添一”得到
4. $w(M) \le w(H')/2$，为什么？
   - G'的两个完美匹配可由H'中交替取边得到
5. $w(H) / w(H^*) < 3/2$

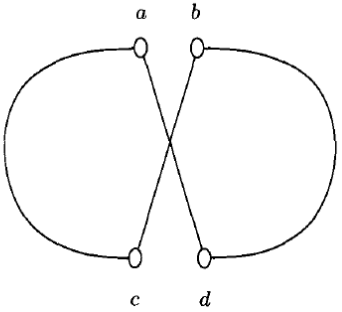


最小权匹配法

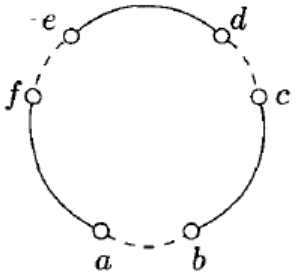
- 时间复杂度
  1. 找最小生成树
     - Prim算法：$O(m + n \log n)$
  2. 找最小权完美匹配
     - 基于花算法：$O(n^3)$
  3. 添加边：$O(n)$ //M的边数最多为$n/2$
  4. 找欧拉回路
     - 希尔霍尔策算法： $O(n)$
  5. 沿欧拉回路前行：$O(n)$

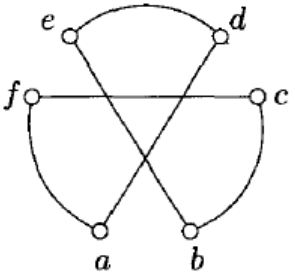
Kernighan-Lin

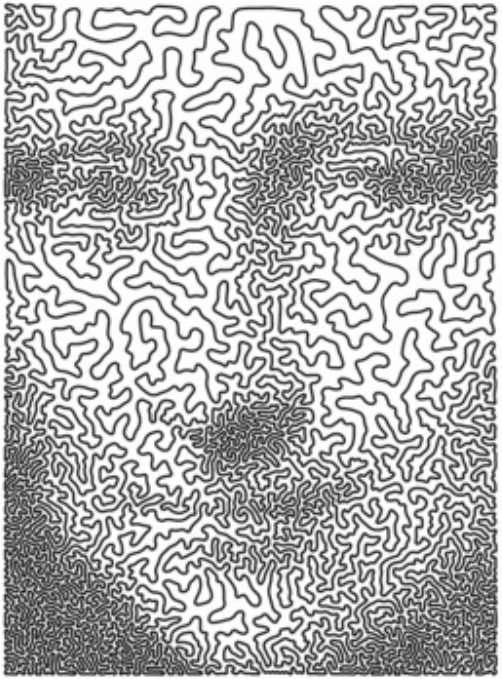




理论近似比：较差
但实际效果：较好







http://www.sciencebuzz.org/sites/default/files/imagecache/kiosk_page_full/kiosk_page_images/monalisa_0.png

本次课的主要内容

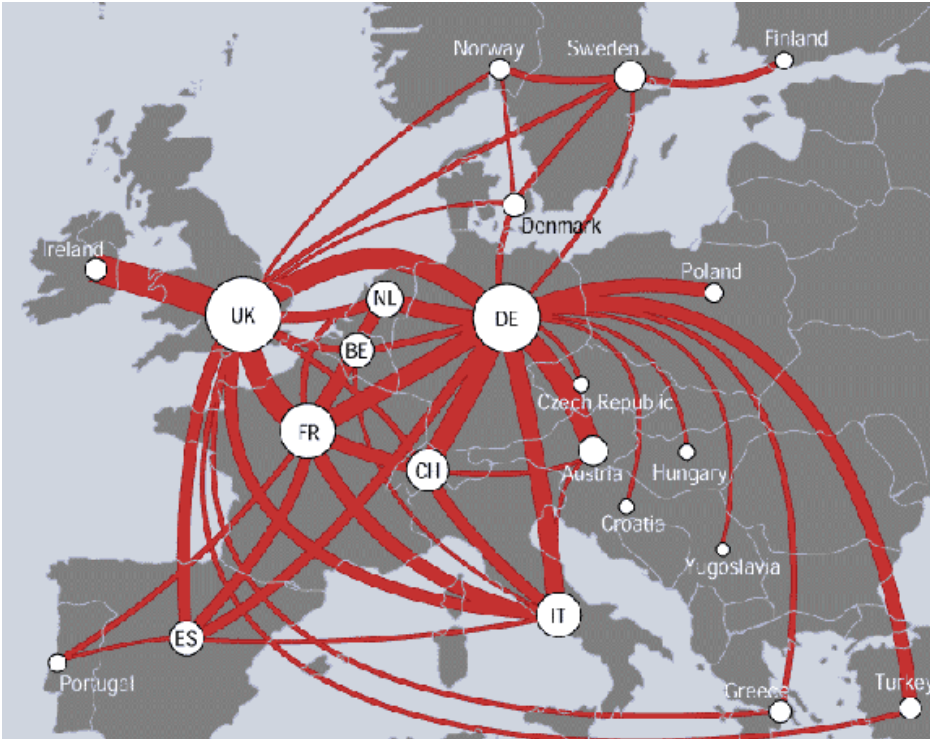
高随祥《图论与网络流理论》

4.2 中国邮递员问题

4.4 旅行商问题

9.1 网络与网络流的基本概念

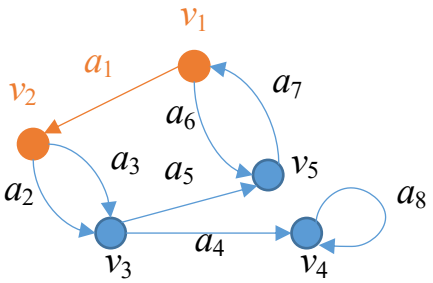
9.2 最大流问题及其标号算法



https://www.gislounge.com/wp-content/uploads/2014/04/network-flow-map.png

网络流的基本概念

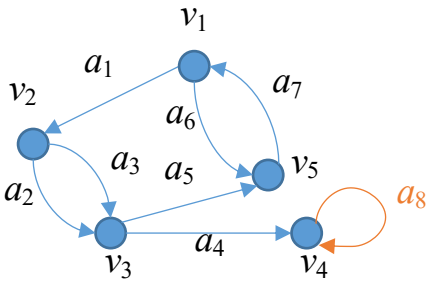
- **有向图**：$G = <V, A>$
  - V：顶点（结点）的有限集合
  - A：有向边（弧）的有限集合
    - 弧$a_1 = (v_1, v_2)$是V中顶点v_1和v_2组成的一个有序对
    - v_1称作a_1的尾，v_2称作a_1的头，统称作a_1的端点



网络流的基本概念

- 有向图：$G = <V, A>$
  - V：顶点（结点）的有限集合
  - A：有向边（弧）的有限集合
    - 弧$a_1 = (v_1, v_2)$是V中顶点v_1和v_2组成的一个有序对
    - v_1称作a_1的尾，v_2称作a_1的头，统称作a_1的端点

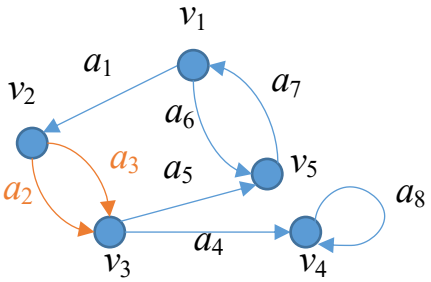
- 一些术语
  - **自环**：一条弧的头和尾是同一个顶点



网络流的基本概念

- 有向图：$G = <V, A>$
  - V：顶点（结点）的有限集合
  - A：有向边（弧）的有限集合
    - 弧$a_1 = (v_1, v_2)$是V中顶点v_1和v_2组成的一个有序对
    - v_1称作a_1的尾，v_2称作a_1的头，统称作a_1的端点

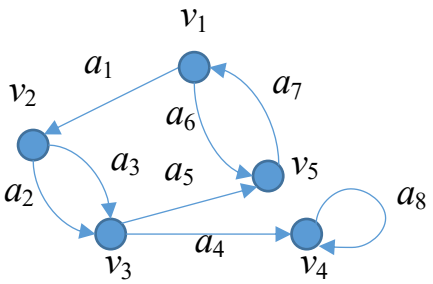
- 一些术语
  - 自环：一条弧的头和尾是同一个顶点
  - **重弧**（平行弧）：头相同、尾相同



网络流的基本概念

- 有向图：$G = <V, A>$
  - V：顶点（结点）的有限集合
  - A：有向边（弧）的有限集合
    - 弧$a_1 = (v_1, v_2)$是V中顶点v_1和v_2组成的一个有序对
    - v_1称作a_1的尾，v_2称作a_1的头，统称作a_1的端点

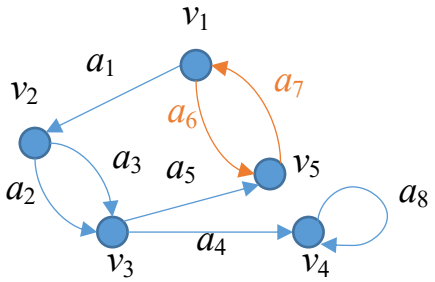
- 一些术语
  - 自环：一条弧的头和尾是同一个顶点
  - 重弧（平行弧）：头相同、尾相同
  - **简单有向图**：不含自环和重弧的有向图



网络流的基本概念

- 有向图：$G = <V, A>$
  - V：顶点（结点）的有限集合
  - A：有向边（弧）的有限集合
    - 弧$a_1 = (v_1, v_2)$是V中顶点v_1和v_2组成的一个有序对
    - v_1称作a_1的尾，v_2称作a_1的头，统称作a_1的端点

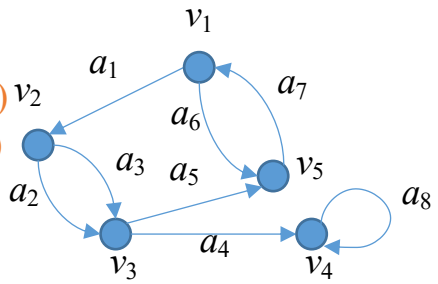
- 一些术语
  - 自环：一条弧的头和尾是同一个顶点
  - 重弧（平行弧）：头相同、尾相同
  - 简单有向图：不含自环和重弧的有向图
  - **反向弧**：头、尾相反



网络流的基本概念

- 有向图：$G = <V, A>$
  - V：顶点（结点）的有限集合
  - A：有向边（弧）的有限集合
    - 弧$a_1 = (v_1, v_2)$是V中顶点v_1和v_2组成的一个有序对
    - v_1称作a_1的尾，v_2称作a_1的头，统称作a_1的端点

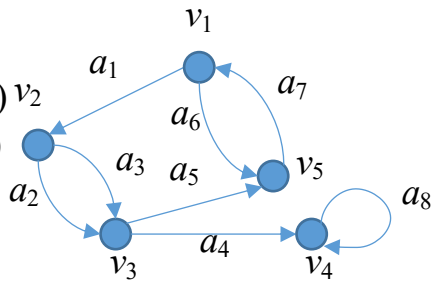
- 一些术语
  - 自环：一条弧的头和尾是同一个顶点
  - 重弧（平行弧）：头相同、尾相同
  - 简单有向图：不含自环和重弧的有向图
  - 反向弧：头、尾相反
  - **出度**：作为尾关联的弧（出弧）的数量，记作$d^+(v)$
  - **入度**：作为头关联的弧（入弧）的数量，记作$d^-(v)$



网络流的基本概念

- 有向图：$G = <V, A>$
  - V：顶点（结点）的有限集合
  - A：有向边（弧）的有限集合
    - 弧$a_1 = (v_1, v_2)$是V中顶点v_1和v_2组成的一个有序对
    - v_1称作a_1的尾，v_2称作a_1的头，统称作a_1的端点

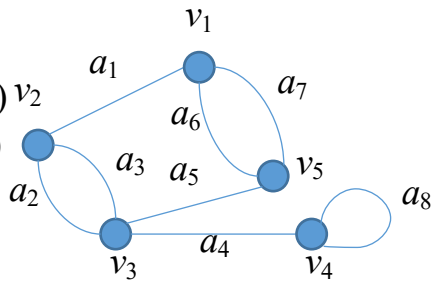
- 一些术语
  - 自环：一条弧的头和尾是同一个顶点
  - 重弧（平行弧）：头相同、尾相同
  - 简单有向图：不含自环和重弧的有向图
  - 反向弧：头、尾相反
  - 出度：作为尾关联的弧（出弧）的数量，记作$d^+(v)$
  - 入度：作为头关联的弧（入弧）的数量，记作$d^-(v)$
  - 有向路线、有向迹、有向路
  - 有向闭路线、有向闭迹、有向圈



网络流的基本概念

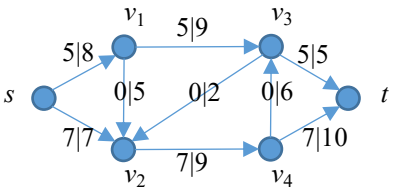
- 有向图：$G = <V, A>$
  - V：顶点（结点）的有限集合
  - A：有向边（弧）的有限集合
    - 弧$a_1 = (v_1, v_2)$是V中顶点v_1和v_2组成的一个有序对
    - v_1称作a_1的尾，v_2称作a_1的头，统称作a_1的端点

- 一些术语
  - 自环：一条弧的头和尾是同一个顶点
  - 重弧（平行弧）：头相同、尾相同
  - 简单有向图：不含自环和重弧的有向图
  - 反向弧：头、尾相反
  - 出度：作为尾关联的弧（出弧）的数量，记作$d^+(v)$
  - 入度：作为头关联的弧（入弧）的数量，记作$d^-(v)$
  - 有向路线、有向迹、有向路
  - 有向闭路线、有向闭迹、有向圈
  - 底图：有向图 → 无向图



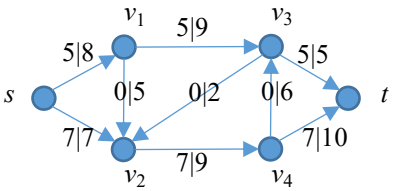
网络流的基本概念

- **流网络**
  - **容量**：定义在弧集上的非负实值函数，记作c(a)
  - **流**：定义在弧集上的实值函数，记作f(a)
    - $f^+(v)$：v关联的出弧的流和
    - $f^-(v)$：v关联的入弧的流和
  - **源点**：$s \in V$
  - **汇点**：$t \in V$



网络流的基本概念

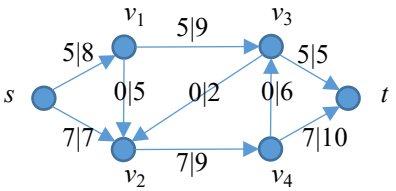
- 流网络
  - 容量：定义在弧集上的非负实值函数，记作c(a)
  - 流：定义在弧集上的实值函数，记作f(a)
    - $f^+(v)$：v关联的出弧的流和
    - $f^-(v)$：v关联的入弧的流和
  - 源点：$s \in V$
  - 汇点：$t \in V$



- **可行流**
  - 容量约束：$\forall a \in A, 0 \le f(a) \le c(a)$
  - 守恒约束：$\forall v \in V \setminus \{s, t\}, f^+(v) = f^-(v)$

网络流的基本概念

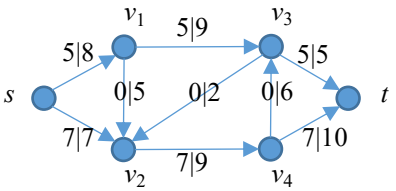
- 流网络
  - 容量：定义在弧集上的非负实值函数，记作$c(a)$
  - 流：定义在弧集上的实值函数，记作$f(a)$
    - $f^+(v)$：v关联的出弧的流和
    - $f^-(v)$：v关联的入弧的流和
  - 源点：$s \in V$
  - 汇点：$t \in V$



- 可行流
  - 容量约束：$\forall a \in A, 0 \le f(a) \le c(a)$
  - 守恒约束：$\forall v \in V \setminus \{s, t\}, f^+(v) = f^-(v)$
- 可行流一定存在吗？

网络流的基本概念

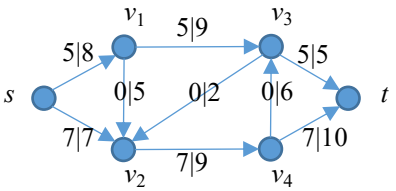
- 流网络
  - 容量：定义在弧集上的非负实值函数，记作$\mathrm{c}(a)$
  - 流：定义在弧集上的实值函数，记作$\mathrm{f}(a)$
    - $\mathrm{f}^+(v)$：v关联的出弧的流和
    - $\mathrm{f}^-(v)$：v关联的入弧的流和
  - 源点：$s \in V$
  - 汇点：$t \in V$



- 可行流
  - 容量约束：$\forall a \in A, 0 \le \mathrm{f}(a) \le \mathrm{c}(a)$
  - 守恒约束：$\forall v \in V \setminus \{s, t\}, \mathrm{f}^+(v) = \mathrm{f}^-(v)$
- 可行流一定存在吗？
  - 零值流：$\forall a \in A, \mathrm{f}(a) = 0$

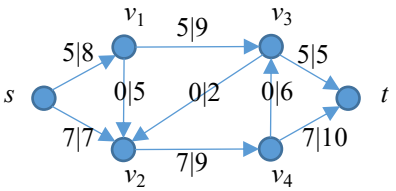
网络流的基本概念

- 流的**流量**：$f^-(t) - f^+(t)$



网络流的基本概念

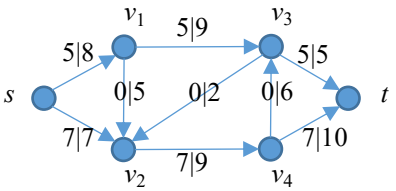
- 流的流量：$f^-(t) - f^+(t)$
- 可行流满足$f^+(s) - f^-(s) = f^-(t) - f^+(t)$，为什么？



网络流的基本概念

- 流的流量：$f^-(t) - f^+(t)$
- 可行流满足$f^+(s) - f^-(s) = f^-(t) - f^+(t)$，为什么？

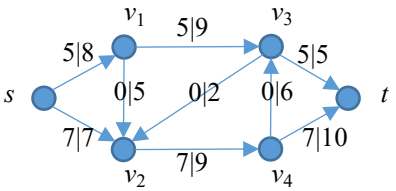
- **最大流**：流量最大的可行流



网络流的基本概念

- 流的流量：$f^-(t) - f^+(t)$
- 可行流满足$f^+(s) - f^-(s) = f^-(t) - f^+(t)$，为什么？

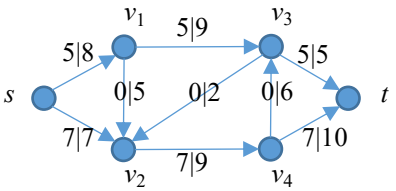
- 最大流：流量最大的可行流
- 右图是最大流吗？你是如何判断的？

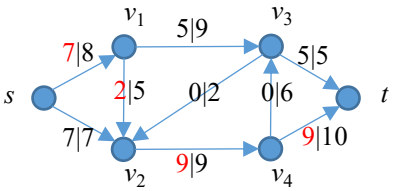


网络流的基本概念

- 流的流量：$f^-(t) - f^+(t)$
- 可行流满足$f^+(s) - f^-(s) = f^-(t) - f^+(t)$，为什么？

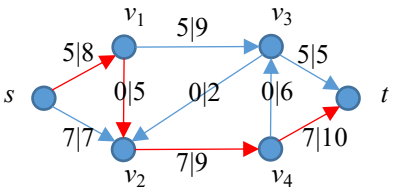
- 最大流：流量最大的可行流
- 右图是最大流吗？你是如何判断的？





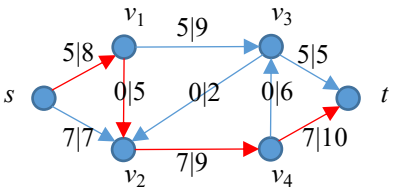
网络流的基本概念

- **f-增广路**
  - 一条s-t路
  - 经过的每条弧$a \in A$: $f(a) < c(a)$



网络流的基本概念

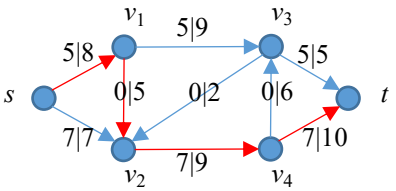
- f-增广路
  - 一条s-t路
  - 经过的每条　　弧$a \in A$: $f(a) < c(a)$



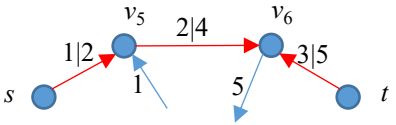
还有其它类型的增广路吗？

网络流的基本概念

- f-增广路
  - 底图中的一条s-t路
  - 经过的每条前向弧$a \in A$: $f(a) < c(a)$
  - 经过的每条后向弧$a \in A$: $f(a) > 0$

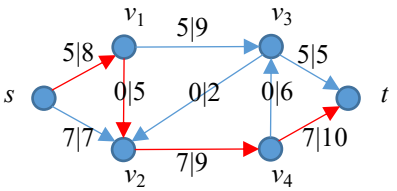


还有其它类型的增广路吗？

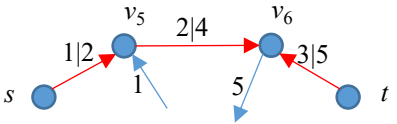


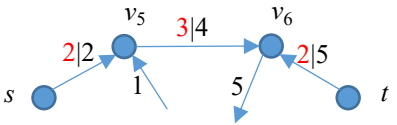
网络流的基本概念

- f-增广路
  - 底图中的一条s-t路
  - 经过的每条前向弧$a \in A$: $f(a) < c(a)$
  - 经过的每条后向弧$a \in A$: $f(a) > 0$



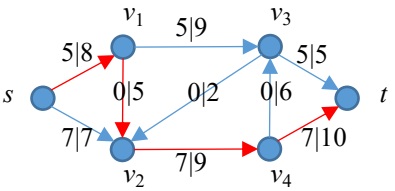
还有其它类型的增广路吗？



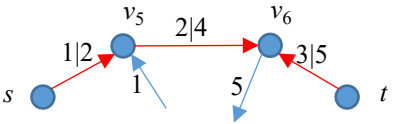


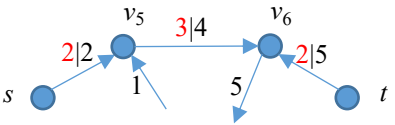
网络流的基本概念

- f-增广路
  - 底图中的一条s-t路
  - 经过的每条前向弧$a \in A$: $f(a) < c(a)$
  - 经过的每条后向弧$a \in A$: $f(a) > 0$
- 增广路的“可增量”有多少？



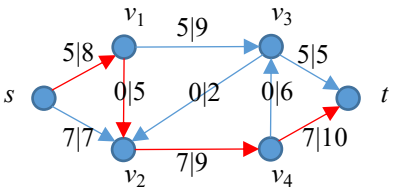
还有其它种类的增广路吗？



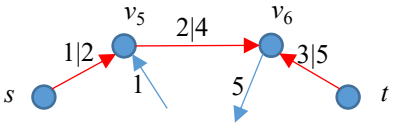


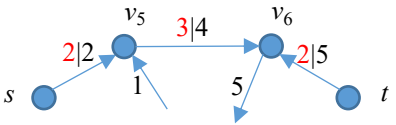
网络流的基本概念

- f-增广路
  - 底图中的一条s-t路
  - 经过的每条前向弧$a \in A$: $f(a) < c(a)$
  - 经过的每条后向弧$a \in A$: $f(a) > 0$
- 增广路的 “可增量” 有多少？
  - 弧的可增量
    - 前向弧：$c(a) - f(a)$
    - 后向弧：$f(a)$
  - 增广路的可增量
    - 经过的弧的可增量的最小值



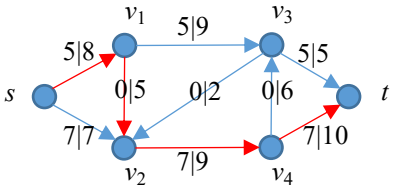
还有其它种类的增广路吗？

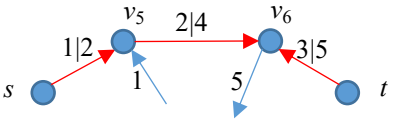




网络流的基本概念

- 对于可增量为z的f增广路，对f做如下调整，结果f'仍是一个可行流，且流量增加z：
  - 前向弧a：$f'(a) = f(a) + z$
  - 后向弧a：$f'(a) = f(a) - z$
  - 其它弧a：$f'(a) = f(a)$





证明：你能自己证明吗？

1、容量约束：$\forall a \in A, 0 \le f'(a) \le c(a)$

2、守恒约束：

3、流量增量：

$(f'^{-}(t) - f'^{+}(t)) - (f^{-}(t) - f^{+}(t))$

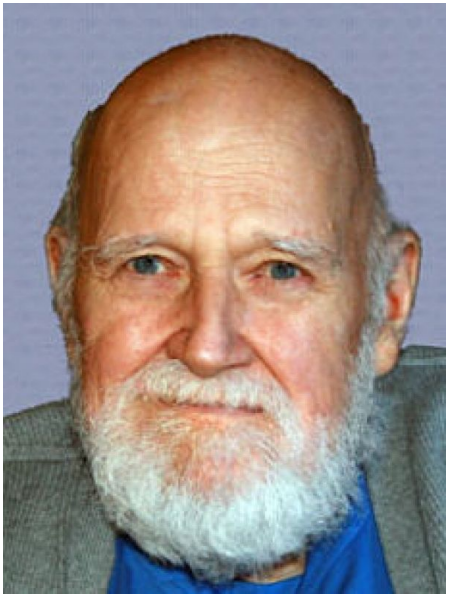
$= (f'^{-}(t) - f^{-}(t)) - (f'^{+}(t) - f^{+}(t))$

$= z - 0$ 或 $0 - (-z)$

$= z$

Ford-Fulkerson算法

- L. R. Ford Jr.，1927-2017，出生于美国
- D. R. Fulkerson，1924-1976，出生于美国



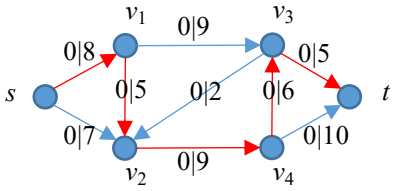
https://www.independent.com/wp-content/uploads/2019/03/dadd_opt.jpg?resize=600,800
https://en.wikipedia.org/wiki/D._R._Fulkerson

Ford-Fulkerson算法

- 基本思路
  1. 从零值流开始
  2. 搜索一条增广路
  3. 如果找到了：调整得到流量更大的流，回到第2步
  4. 否则：结束

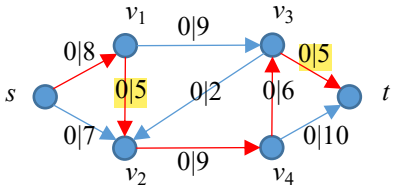
Ford-Fulkerson算法

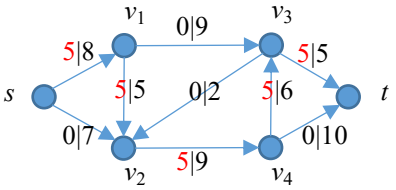
- 举例



Ford-Fulkerson算法

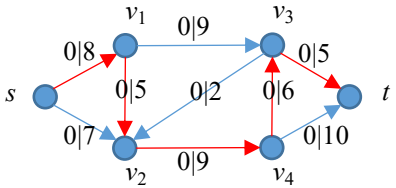
- 举例

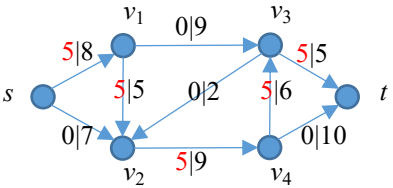


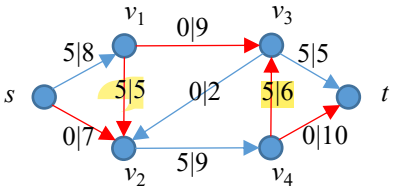


Ford-Fulkerson算法

- 举例

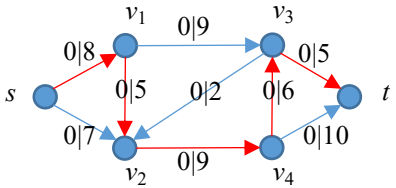


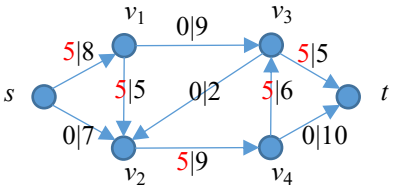


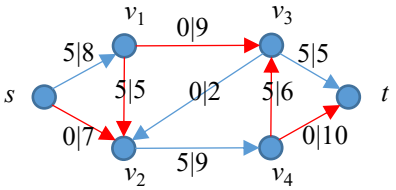


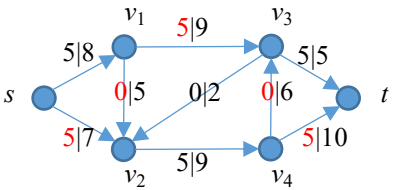
Ford-Fulkerson算法

- 举例



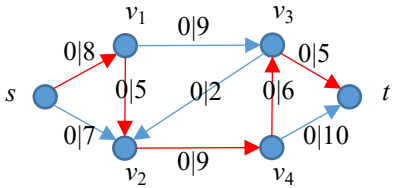


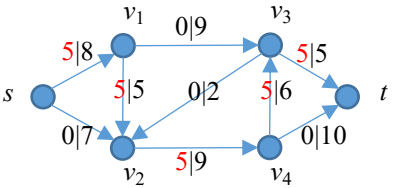


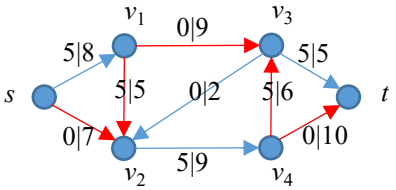


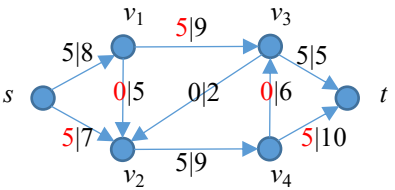
Ford-Fulkerson算法

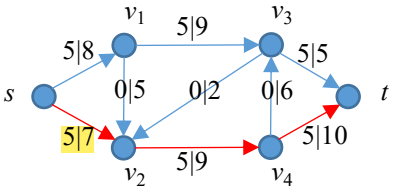
- 举例





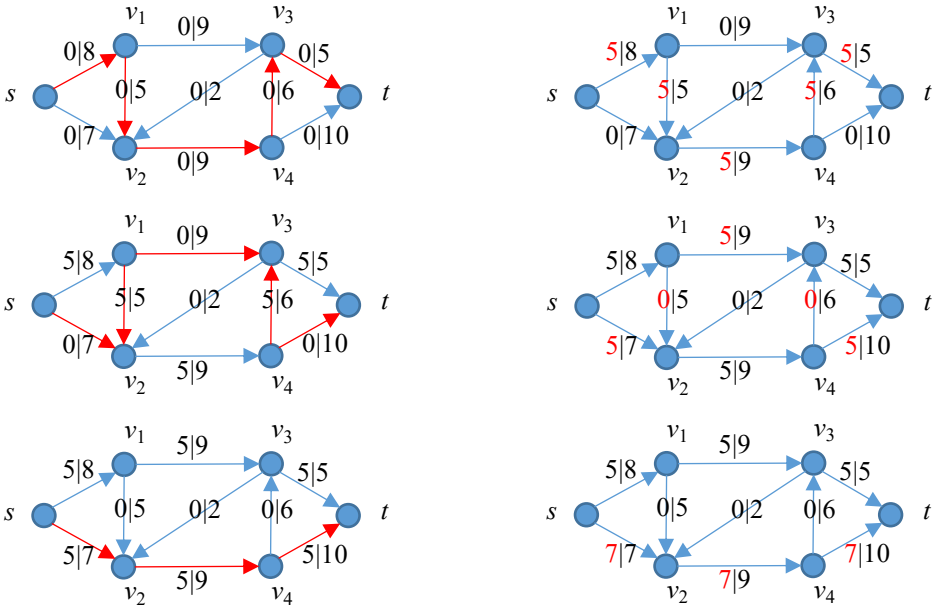






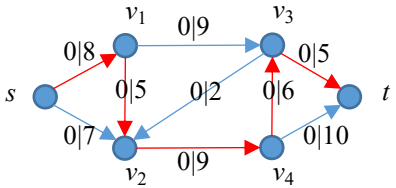
Ford-Fulkerson算法

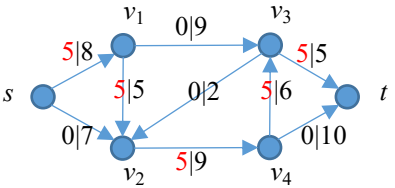
- 举例

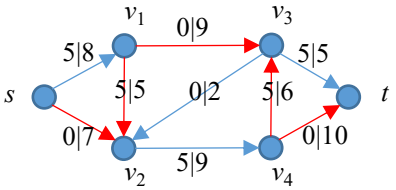


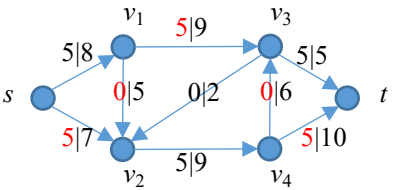
Ford-Fulkerson算法

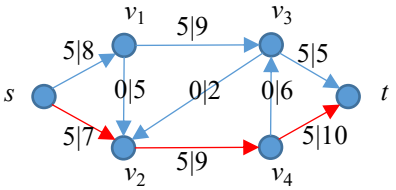
- 举例

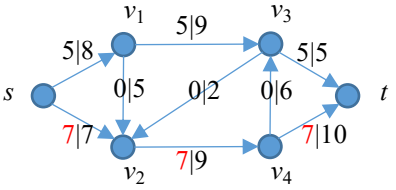


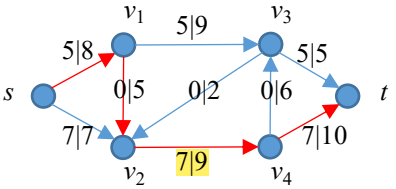






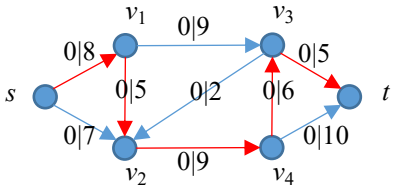


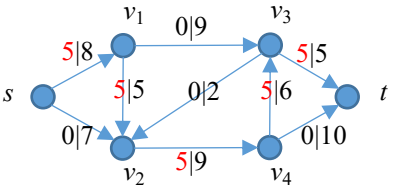


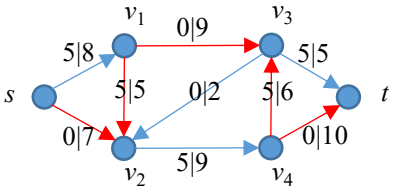


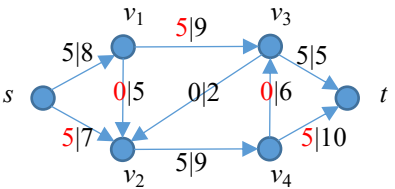
Ford-Fulkerson算法

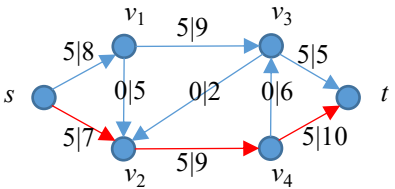
- 举例

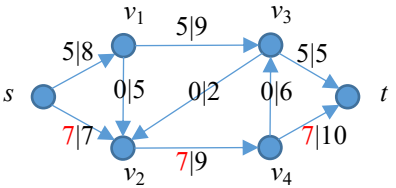


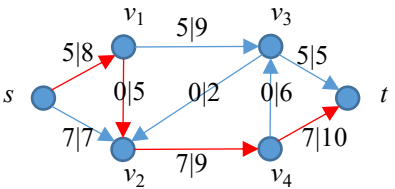


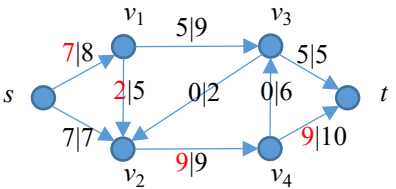






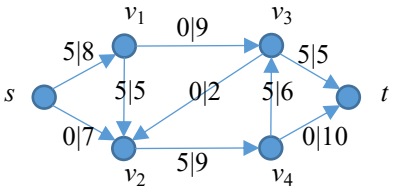


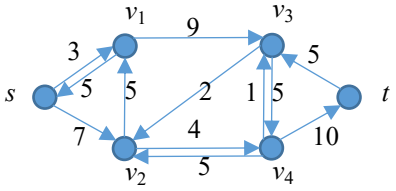


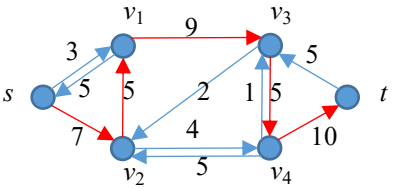


Ford-Fulkerson算法

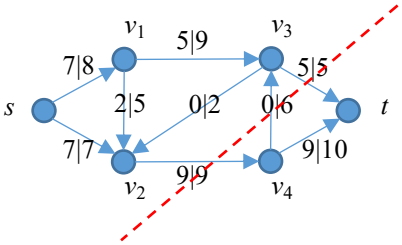
- 如何搜索一条增广路？
  - 构建余量网络

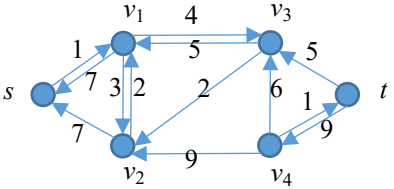


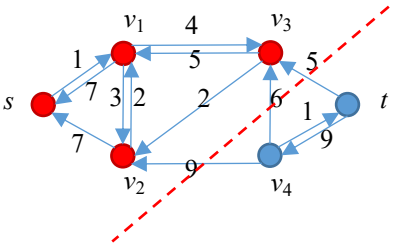




  - 找不到增广路时，余量网络呈现什么特征？

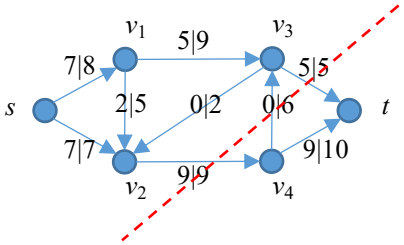






最大流与最小割

- 将顶点集任意划分为S和T，使$s \in S, t \in T = V \setminus S$
- **s-t割**
  - $[S, T] = \{<x, y> \in A: x \in S, y \in T\}$
- 割的容量
  - 弧的容量和：$\sum_{a \in [S, T]} c(a)$
- 最小割
  - 容量最小的割



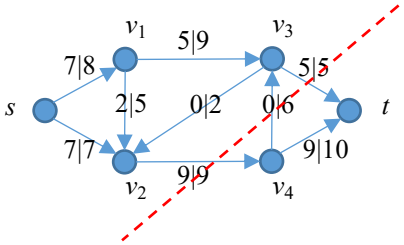
最大流与最小割

- 对网络中任一可行流f和任一割$[S, T]$，均有$f^+(s) - f^-(s) = f^+(S) - f^-(S)$，其中$f^+(S)$表示$S$关联的出弧的流和、$f^-(S)$表示$S$关联的入弧的流和。

证明：你能自己证明吗？

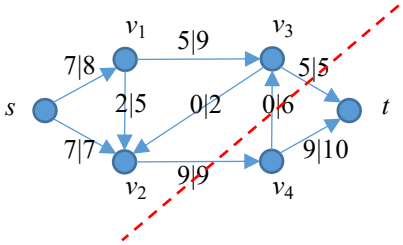
$\forall v \in S \setminus \{s\}, f^+(v) = f^-(v)$

$\Rightarrow f^+(s) - f^-(s) = f^+(s) - f^-(s) + \sum_{v \in S \setminus \{s\}} (f^+(v) - f^-(v)) = \sum_{v \in S} (f^+(v) - f^-(v)) = f^+(S) - f^-(S)$



最大流与最小割

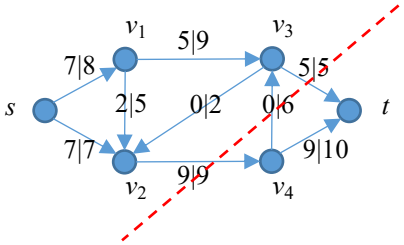
- 对网络中任一可行流f和任一割$[S, T]$，均有$f^+(s) - f^-(s) = f^+(S) - f^-(S)$，其中$f^+(S)$表示$S$关联的出弧的流和、$f^-(S)$表示$S$关联的入弧的流和。
- 对网络中任一可行流f和任一割$[S, T]$，均有f的流量不超过$[S, T]$的容量。（两者相等时f有什么特征？）



S到T的弧：f = c
T到S的弧：f = 0

最大流与最小割

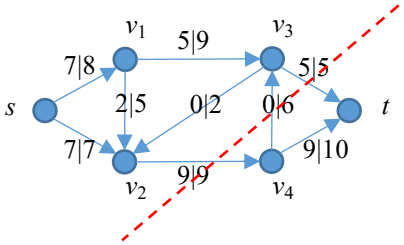
- 对网络中任一可行流f和任一割$[S, T]$，均有$f^+(s) - f^-(s) = f^+(S) - f^-(S)$，其中$f^+(S)$表示$S$关联的出弧的流和、$f^-(S)$表示$S$关联的入弧的流和。
- 对网络中任一可行流f和任一割$[S, T]$，均有f的流量不超过$[S, T]$的容量。（两者相等时f有什么特征？）
- 设f是网络中的一个可行流、$[S, T]$是一个割，若f的流量与$[S, T]$的容量相等，那么f是一个最大流而$[S, T]$是一个最小割。



S到T的弧：f = c
T到S的弧：f = 0

最大流与最小割

- 对网络中任一可行流f和任一割$[S, T]$，均有$f^+(s) - f^-(s) = f^+(S) - f^-(S)$，其中$f^+(S)$表示$S$关联的出弧的流和、$f^-(S)$表示$S$关联的入弧的流和。
- 对网络中任一可行流f和任一割$[S, T]$，均有f的流量不超过$[S, T]$的容量。（两者相等时f有什么特征？）
- 设f是网络中的一个可行流、$[S, T]$是一个割，若f的流量与$[S, T]$的容量相等，那么f是一个最大流而$[S, T]$是一个最小割。



S到T的弧：f = c
T到S的弧：f = 0

Ford-Fulkerson算法运行结束时，
恰是这种状态（为什么？），因此算法是正确的。

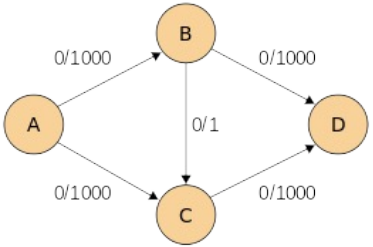
因为在余量网络中，S到T没有弧

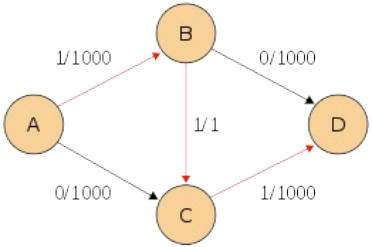
最大流与最小割

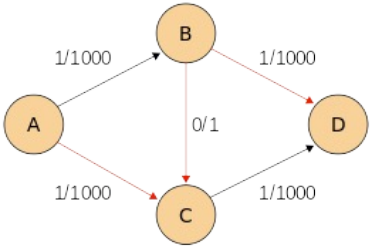
- 最大流的流量 = 最小割的容量。
- 若所有弧的容量都是整数，则最大流的流量也必为整数。

Ford-Fulkerson算法

- 时间复杂度
  - 迭代的轮数？
    - $O(f_{max})$，f_{max}为最大流的流量
  - 每轮迭代的时间：$O(n+m)$





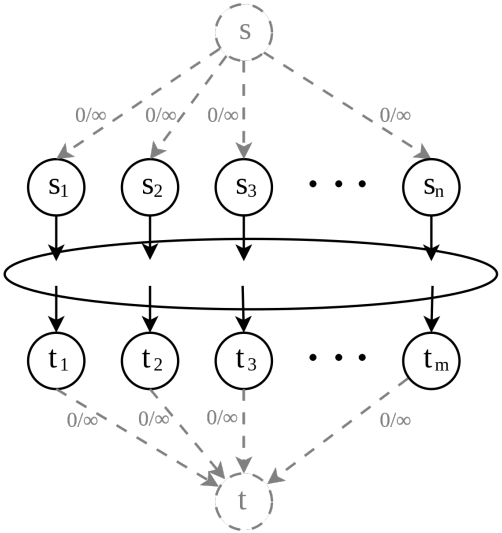


甚至，当容量包含无理数时，算法有可能永不终止。

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

最大流的应用与扩展

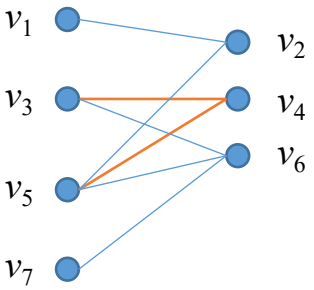
- 多源多汇网络



https://en.wikipedia.org/wiki/File:Multi-source_multi-sink_flow_problem.svg

最大流的应用与扩展

- 还记得求二分图最大匹配的匈牙利算法吗？
  与Ford-Fulkerson算法是不是有些神似？
- 你能将该问题转化为最大流问题吗？（留作练习）



最大流的应用与扩展

- 如何计算图的点（边）连通度？

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

书面作业

高随祥《图论与网络流理论》

- 练习4.12（要写出主要步骤）
- 将二分图最大匹配问题归约为最大流问题
- 将边连通度计算问题归约为最大流问题
- 将点连通度计算问题归约为最大流问题