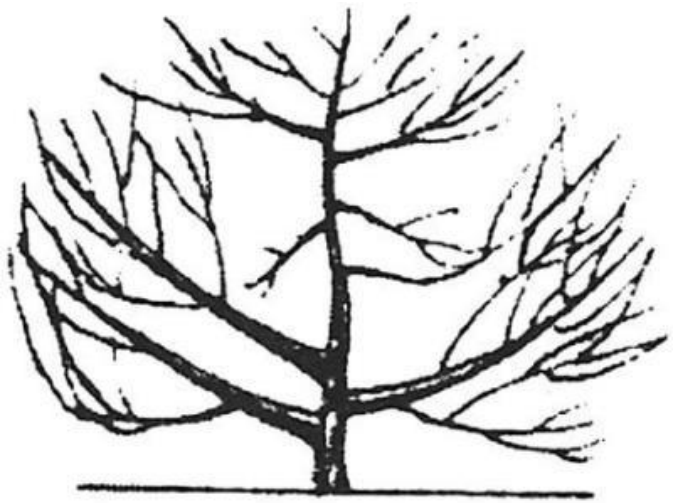
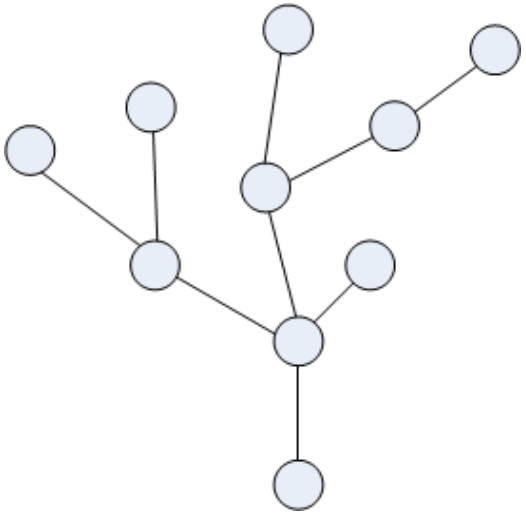
图与网络分析第2节

树、支撑树、最小树

西安交通大学电信学院系统工程研究所
翟桥柱、吴江

树：定义与性质





树：连通且无回路的(无向)图。

一定是简单图，也是最简单、基本、非常重要的图

一个无回路的图，其每一个连通分支都是一棵树，因此无回路的图也就是若干棵互不连通的树构成的图，称为**森林**

树有哪些重要特性？

树：定义与性质

定理5.3.1： 设$G=(V,E)$是无向图，且$|V|=N$，则以下陈述等价：

(1) G连通且无回路

(2) G有N-1 条边且无(初级)回路

(3) G连通且有N-1 条边

(4) G连通且每条边都是割边

(5) G的任意两顶点间有唯一(初级)路相连

(6) G无回路，但在任意一对不相邻的顶点间加一条边，则构成唯一的一个(初级)回路。

树的等价定义

定理5.3.2： 设$G=(V,E)$是一棵树，且$|V|=N\geqslant 2$，则G至少有两个次为 1 的顶点。

证明： 由$\sum_{v_i\in V} d(v_i)=2|E|=2N-2$可知

支撑树：定义与性质

树的很多重要应用，与支撑树、最小(支撑)树有关

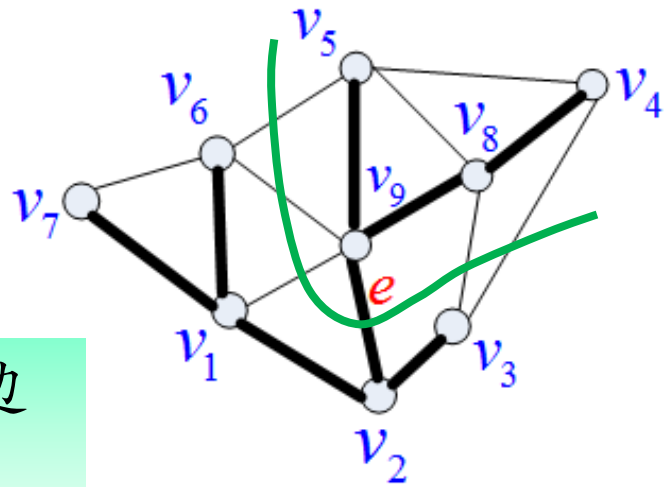
支撑树： 设$G=(V,E)$是无向图，$T=(V,E')$是G的一个支撑子图，若T是一棵树，则称其为G的一个支撑树。

反树： 设$T=(V,E')$是G的一个支撑树，则称$T^*=E\setminus E'$是T的反树。

注意：反树不是图，只是边集的一个子集

定理5.3.3： $G=(V,E)$有支撑树当且仅当G连通。

定理5.3.4： $T=(V,E')$是G的一个支撑树，$e\in E'$，则存在唯一一个割集$\Omega(e)\subseteq T^*\cup\{e\}$。



例：**支撑树**：粗边对应支撑图；**反树**：细边
唯一割集：与绿线相交的所有边

支撑树：定义与性质

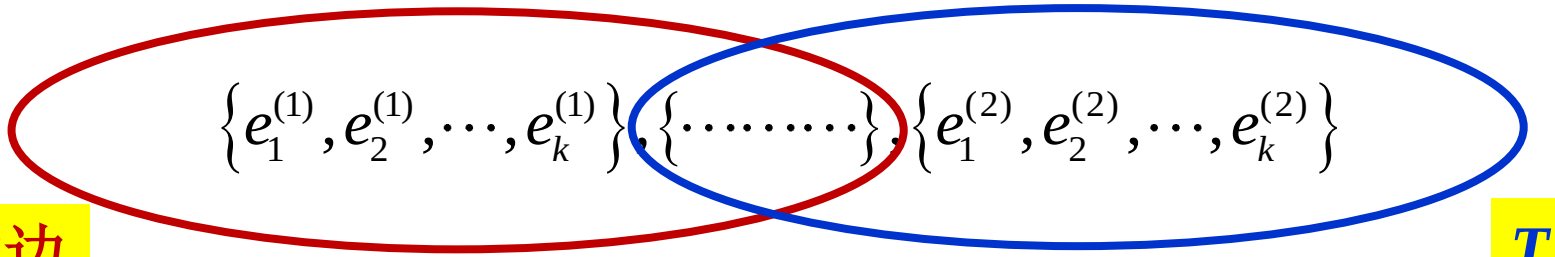
树的很多重要应用，与支撑树、最小(支撑)树有关

支撑树： 设$G=(V,E)$是无向图，$T=(V,E')$是G的一个支撑子图，若T是一棵树，则称其为G的一个支撑树。

反树： 设$T=(V,E')$是G的一个支撑树，则称$T^*=E\setminus E'$是T的反树。

注意：反树不是图，只是边集的一个子集

定理5.3.5： 设$T_1=(V,E_1)$和$T_2=(V,E_2)$是G的两个支撑树，且$|E_1\setminus E_2|=k$，则T_1和T_2可以经过k次迭代互换。



最小树：定义与性质

树的很多重要应用，与支撑树、最小(支撑)树有关

最小树： 设$G=(V,E,W)$是无向网络，$T=(V,E')$是G的一个支撑树，则称$W(T)=\sum_{e\in E'} w(e)$为树T的权。G的所有支撑树中权最小的支撑树称为G的最小树。

注意：最小树首先是支撑树；最小树可能不唯一

怎样求得最小树？

对一个有限连通图，虽然只有有限个支撑树，但不能通过枚举来寻找最小树，因支撑树数目巨大。最小树的高效获取方法建立在最优性条件基础上(最小树的充分、必要条件)

最小树：定义与性质

定理5.4.1：设$G=(V,E,W)$是无向网络，则$T=(V,E')$是G的一个最小树当且仅当$\forall e\in T^*, w(e)=\max\limits_{e'\in C(e)} w(e')$。其中，$C(e)\subseteq T+e$是$T+e$中的唯一一个初级回路。

证明：必要性，显然。充分性，广义“同一法”。任取一个最小树$\tilde{T}=(V,\tilde{E})$，证明T和$\tilde{T}$的权相同。

设$\left|\tilde{E}\setminus E'\right|=k \Rightarrow \{\tilde{e}_1,\tilde{e}_2,\cdots,\tilde{e}_k\},\{\cdots\cdots\},\{e_1,e_2,\cdots,e_k\}$

$$\Rightarrow \min\{w(\tilde{e}_1),w(\tilde{e}_2),\cdots,w(\tilde{e}_k)\}=\min\{w(e_1),w(e_2),\cdots,w(e_k)\}$$

取$e_i=\arg\min\{w(e_1),w(e_2),\cdots,w(e_k)\} \Rightarrow T+e_i-\tilde{e}_j=\tilde{T}'$

$\tilde{T}'$也是一棵最小树，且和T的不同边数仅有k-1条，…

最小树：定义与性质

定理5.4.2：设$G=(V,E,W)$是无向网络，则$T=(V,E')$是G的一个最小树当且仅当$\forall e\in E', w(e)=\min\limits_{e'\in\Omega(e)} w(e')$。其中，$\Omega(e)\subseteq T^*+e$是$T^*+e$中的唯一一个割集。

证明：必要性，显然。充分性，应用定理5.4.1。

$\forall \tilde{e}\in T^* \Rightarrow E'+\tilde{e}$的回路中任取$T$中一边$e$，形成割集，…

P208,定理5.4.3, 定理5.4.4最小树的唯一性判定条件

求最小树的两个常用算法：

- Kruskal算法(1956, 贪心算法，Greedy Algorithm)
- Dijkstra算法(1959, Prim算法的改进，标号法)

Dijkstra算法效率更高，但Kruskal算法有某些特殊应用

最小树：Kruskal算法(1956, Greedy Algorithm)

思想： 设$G=(V,E,W)$是连通的无向网络，$|V|=n$，$|E|=m$，从G的m条边中选权值尽量小的n-1条边，且不构成回路

算法：

Step 1. 对m条边按权值由小到大排序，使 $w(e_1)\le w(e_2)\le\cdots\le w(e_m)$ 置 $S=\phi, i=0, j=1.$

Step 2. 若$i=n$-1, 停，$G(S)$即为最小树，否则继续。

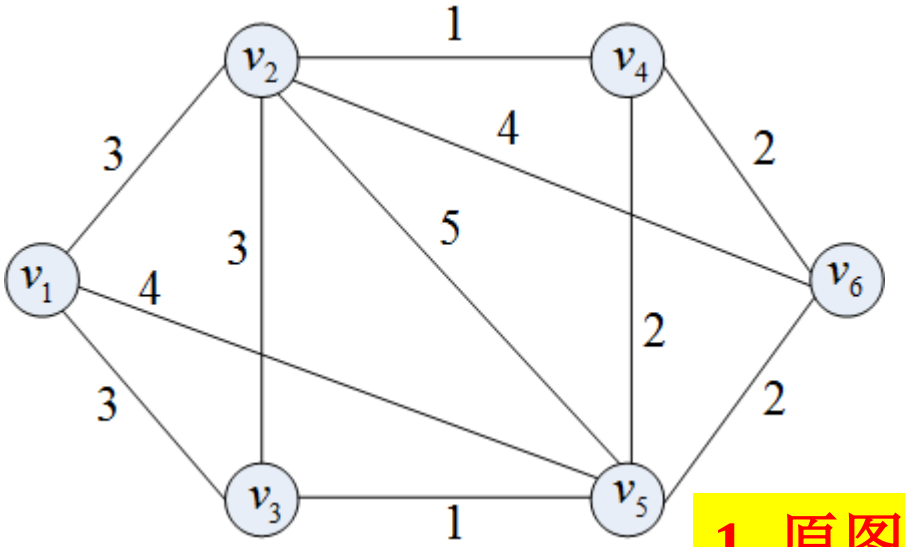
Step 3. 若$S\cup\{e_j\}$无回路，置 $S=S\cup\{e_j\}, i=i+1, j=j+1$，转Step2；否则置 $j=j+1$转Step3.

算法说明：

- 去掉环，重边中只保留权值最小边
- 正确性：基于定理5.4.1
- 无回路判断：顶点所属连通分支标记法
- 计算复杂性：P209

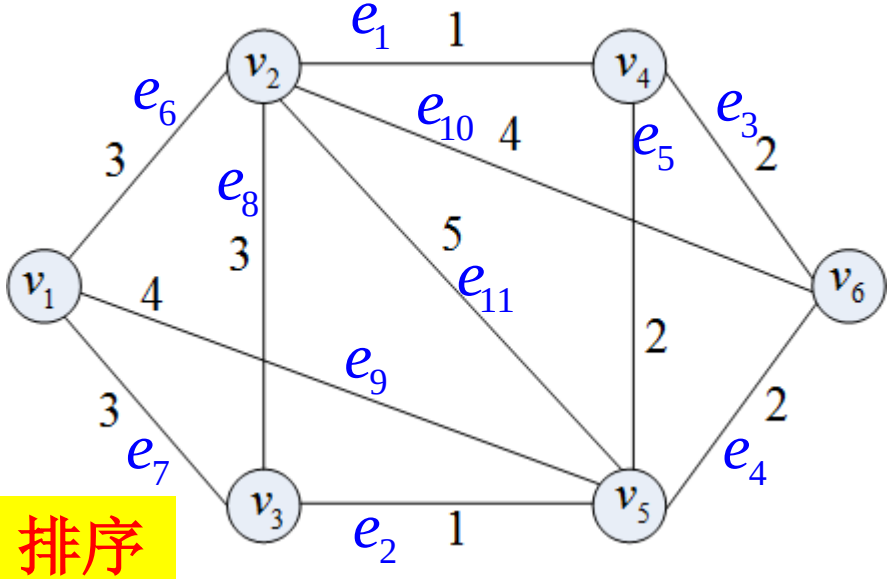
***S*: 树中的边；*i*:已找到多少条边；*j*:已检验至第几条边**

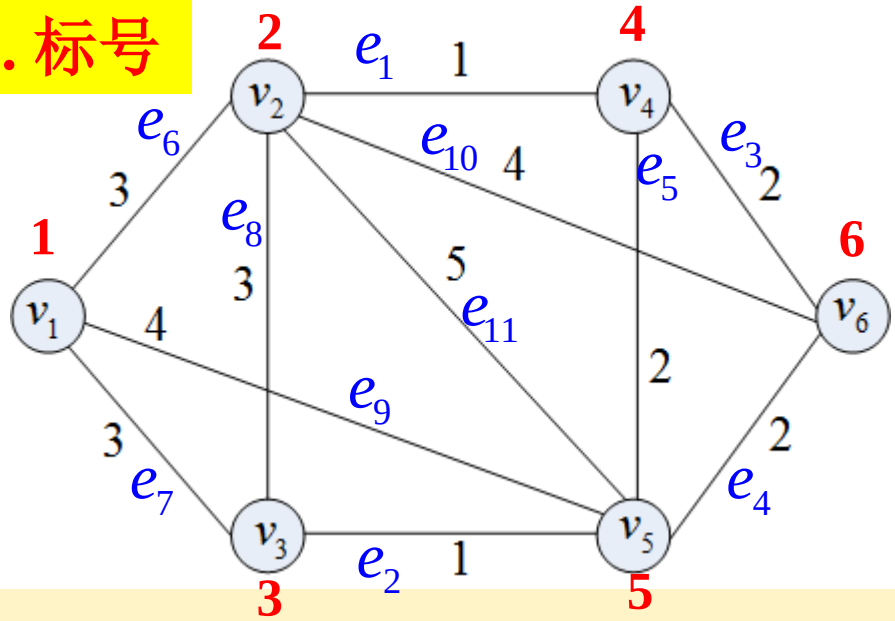
最小树：Kruskal算法(1956, Greedy Algorithm)



1. 原图　2. 排序

4. 迭代　3. 标号

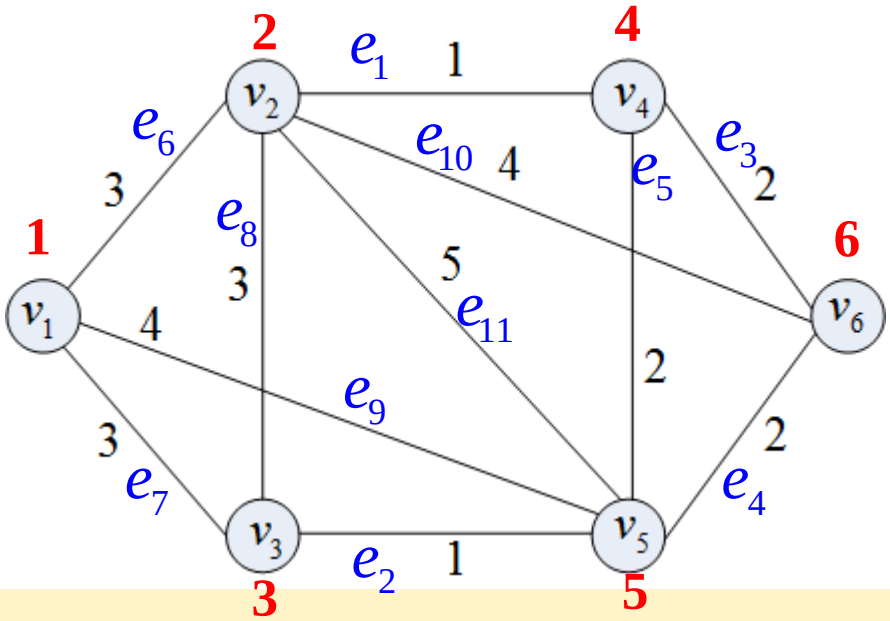




最小树：Kruskal算法(1956, Greedy Algorithm)

$S=\phi, i=0,\ j=1.$

$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

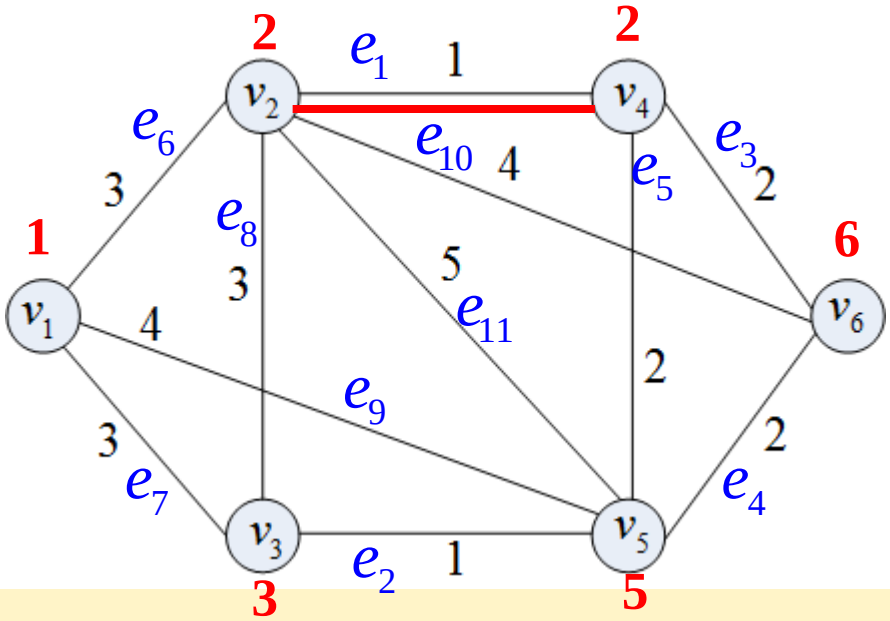


最小树：Kruskal算法(1956, Greedy Algorithm)

$S=\phi, i=0,\ j=1.$

$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

$i\neq 5, e_2$ 两端点标号不同，$S=S\cup\{e_2\}, i=2,\ j=3,$ 修改标号

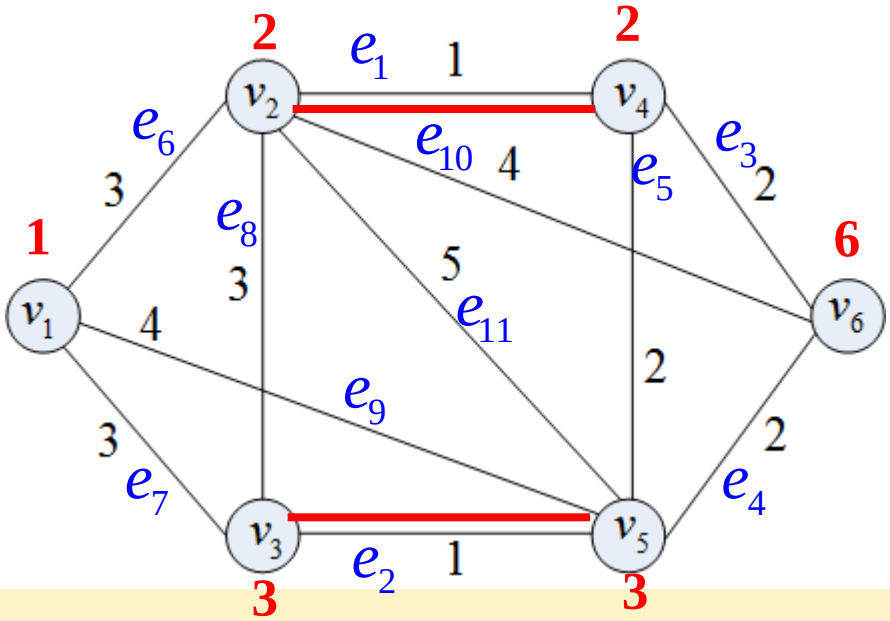


最小树：Kruskal算法(1956, Greedy Algorithm)

$S=\phi, i=0,\ j=1.$

$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

$i\neq 5, e_2$ 两端点标号不同，$S=S\cup\{e_2\}, i=2,\ j=3,$ 修改标号



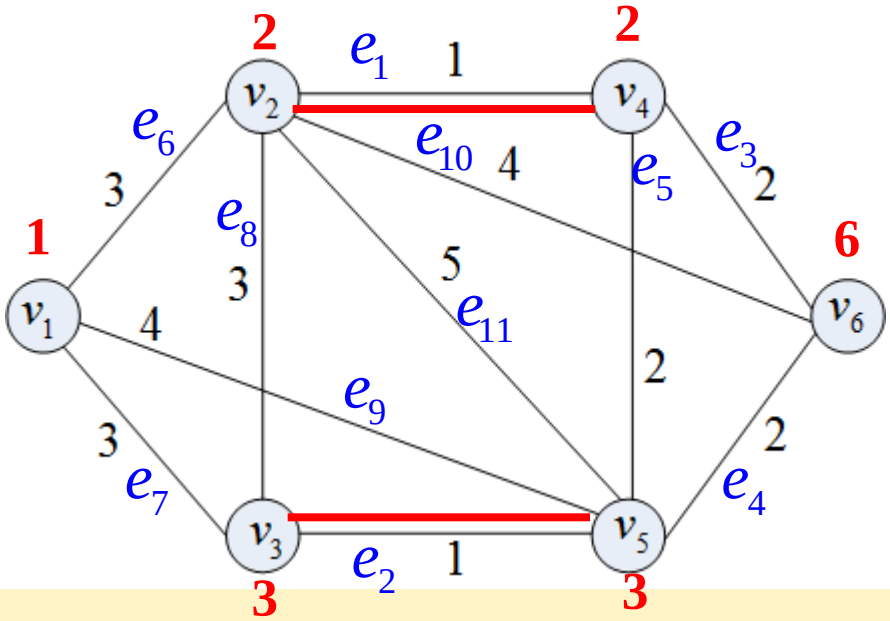
最小树：Kruskal算法(1956, Greedy Algorithm)

$S=\phi, i=0,\ j=1.$

$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

$i\neq 5, e_2$ 两端点标号不同，$S=S\cup\{e_2\}, i=2,\ j=3,$ 修改标号

$i\neq 5, e_3$ 两端点标号不同，$S=S\cup\{e_3\}, i=3,\ j=4,$ 修改标号



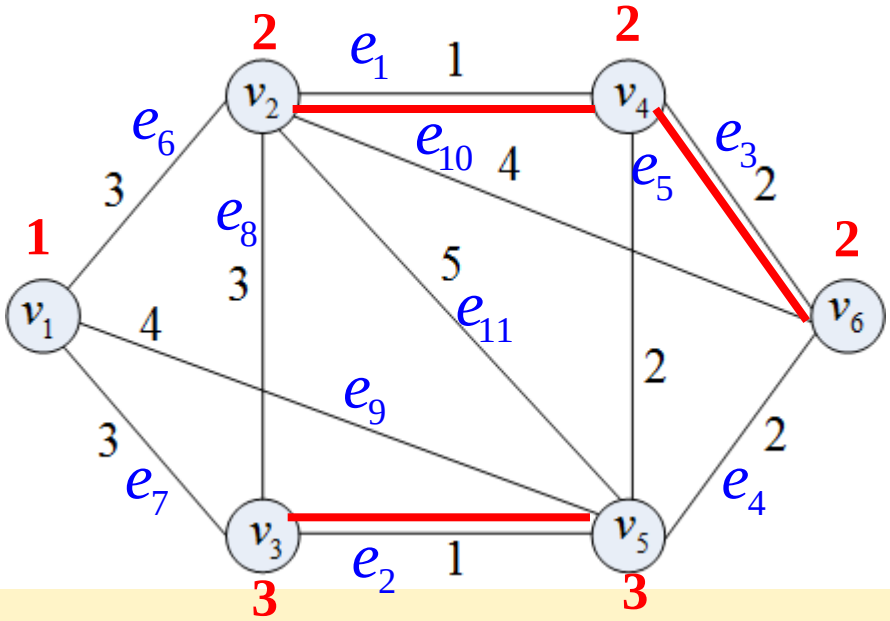
最小树：Kruskal算法(1956, Greedy Algorithm)

$S=\phi, i=0,\ j=1.$

$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

$i\neq 5, e_2$ 两端点标号不同，$S=S\cup\{e_2\}, i=2,\ j=3,$ 修改标号

$i\neq 5, e_3$ 两端点标号不同，$S=S\cup\{e_3\}, i=3,\ j=4,$ 修改标号



最小树：Kruskal算法(1956, Greedy Algorithm)

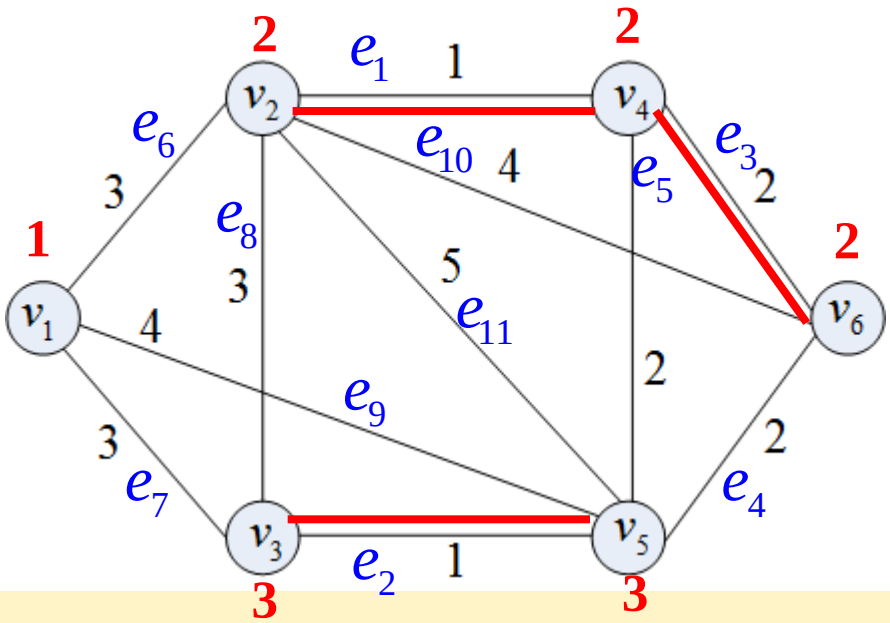
$S=\phi, i=0,\ j=1.$

$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

$i\neq 5, e_2$ 两端点标号不同，$S=S\cup\{e_2\}, i=2,\ j=3,$ 修改标号

$i\neq 5, e_3$ 两端点标号不同，$S=S\cup\{e_3\}, i=3,\ j=4,$ 修改标号

$i\neq 5, e_4$ 两端点标号不同，$S=S\cup\{e_4\}, i=4,\ j=5,$ 修改标号



最小树：Kruskal算法(1956, Greedy Algorithm)

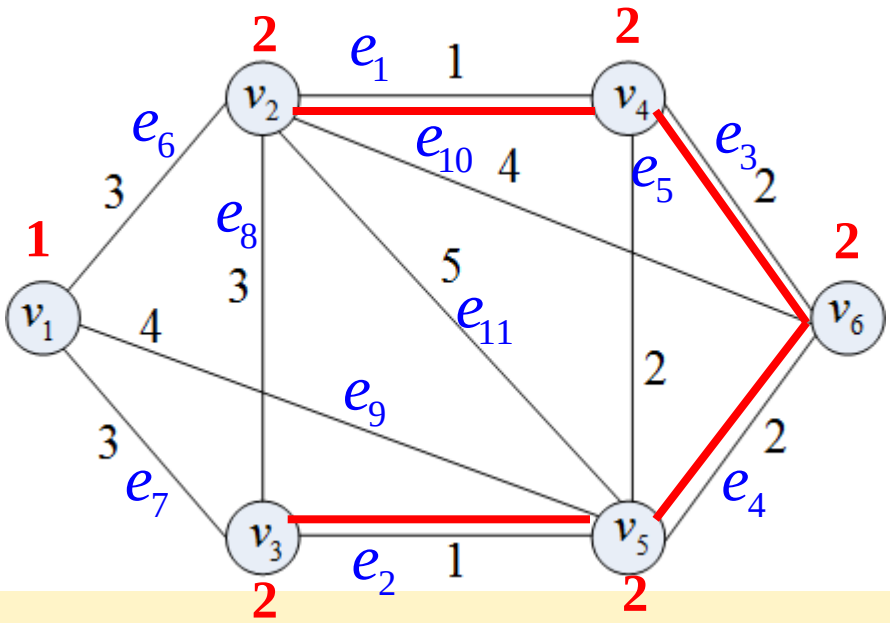
$S=\phi, i=0,\ j=1.$

$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

$i\neq 5, e_2$ 两端点标号不同，$S=S\cup\{e_2\}, i=2,\ j=3,$ 修改标号

$i\neq 5, e_3$ 两端点标号不同，$S=S\cup\{e_3\}, i=3,\ j=4,$ 修改标号

$i\neq 5, e_4$ 两端点标号不同，$S=S\cup\{e_4\}, i=4,\ j=5,$ 修改标号



最小树：Kruskal算法(1956, Greedy Algorithm)

$S=\phi, i=0,\ j=1.$

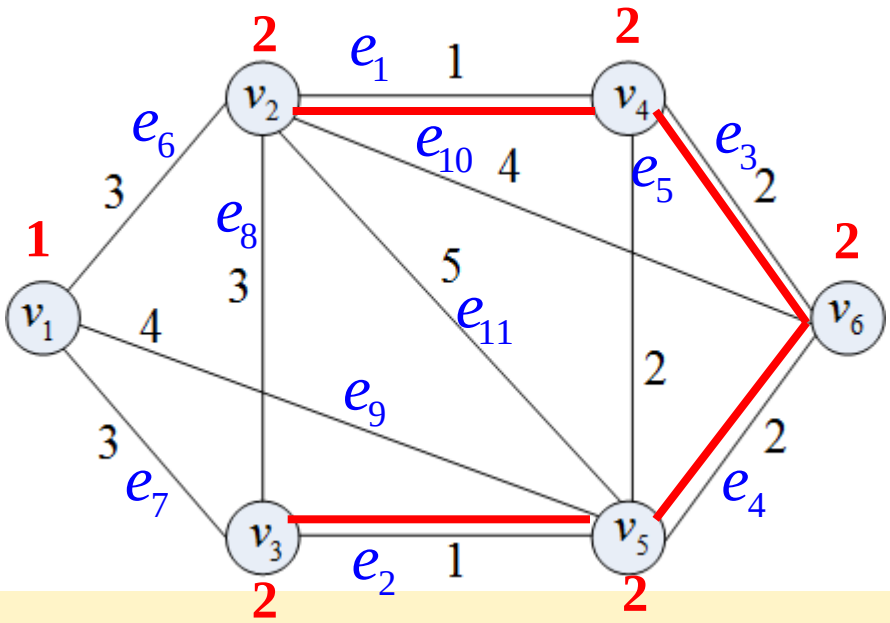
$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

$i\neq 5, e_2$ 两端点标号不同，$S=S\cup\{e_2\}, i=2,\ j=3,$ 修改标号

$i\neq 5, e_3$ 两端点标号不同，$S=S\cup\{e_3\}, i=3,\ j=4,$ 修改标号

$i\neq 5, e_4$ 两端点标号不同，$S=S\cup\{e_4\}, i=4,\ j=5,$ 修改标号

$i\neq 5, e_5$ 两端点标号相同，$j=6$



最小树：Kruskal算法(1956, Greedy Algorithm)

$S=\phi, i=0,\ j=1.$

$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

$i\neq 5, e_2$ 两端点标号不同，$S=S\cup\{e_2\}, i=2,\ j=3,$ 修改标号

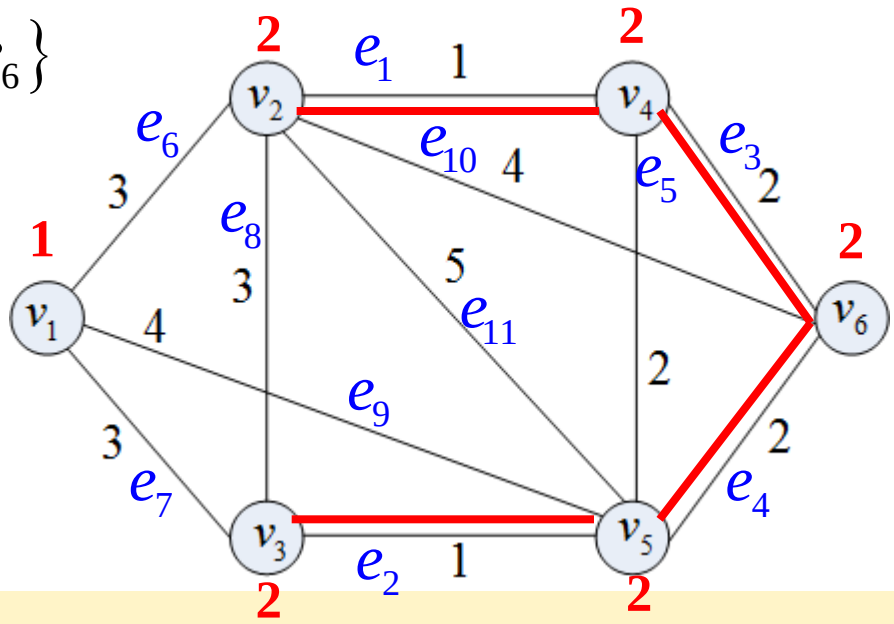
$i\neq 5, e_3$ 两端点标号不同，$S=S\cup\{e_3\}, i=3,\ j=4,$ 修改标号

$i\neq 5, e_4$ 两端点标号不同，$S=S\cup\{e_4\}, i=4,\ j=5,$ 修改标号

$i\neq 5, e_5$ 两端点标号相同，$j=6$

$i\neq 5, e_6$ 两端点标号不同，$S=S\cup\{e_6\}$

$, i=5,\ j=7,$修改标号



最小树：Kruskal算法(1956, Greedy Algorithm)

$S=\phi, i=0,\ j=1.$

$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

$i\neq 5, e_2$ 两端点标号不同，$S=S\cup\{e_2\}, i=2,\ j=3,$ 修改标号

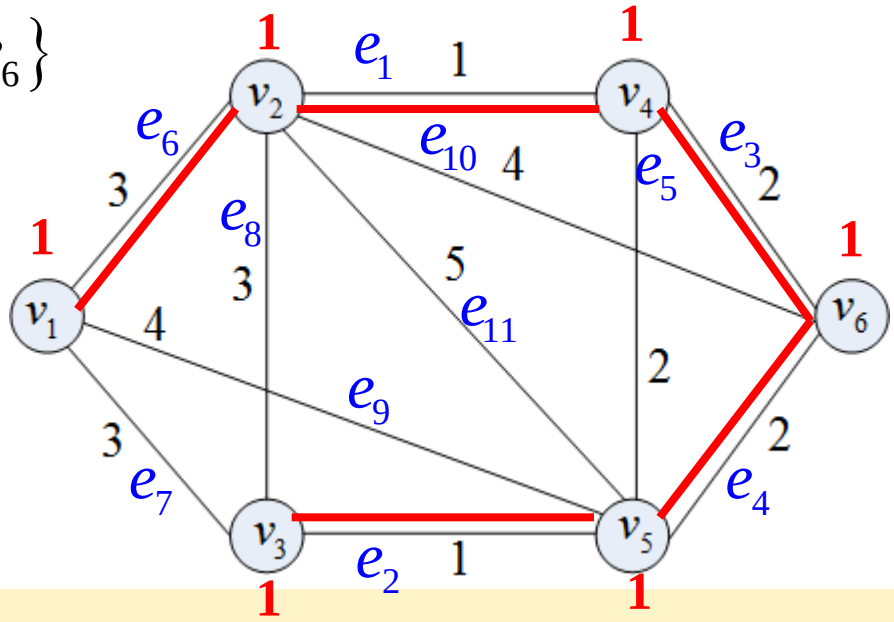
$i\neq 5, e_3$ 两端点标号不同，$S=S\cup\{e_3\}, i=3,\ j=4,$ 修改标号

$i\neq 5, e_4$ 两端点标号不同，$S=S\cup\{e_4\}, i=4,\ j=5,$ 修改标号

$i\neq 5, e_5$ 两端点标号相同，$j=6$

$i\neq 5, e_6$ 两端点标号不同，$S=S\cup\{e_6\}$

$, i=5,\ j=7,$修改标号



最小树：Kruskal算法(1956, Greedy Algorithm)

$S=\phi, i=0,\ j=1.$

$i\neq 5, e_1$ 两端点标号不同， $S=S\cup\{e_1\}=\{e_1\}, i=1,\ j=2,$ 修改标号

$i\neq 5, e_2$ 两端点标号不同，$S=S\cup\{e_2\}, i=2,\ j=3,$ 修改标号

$i\neq 5, e_3$ 两端点标号不同，$S=S\cup\{e_3\}, i=3,\ j=4,$ 修改标号

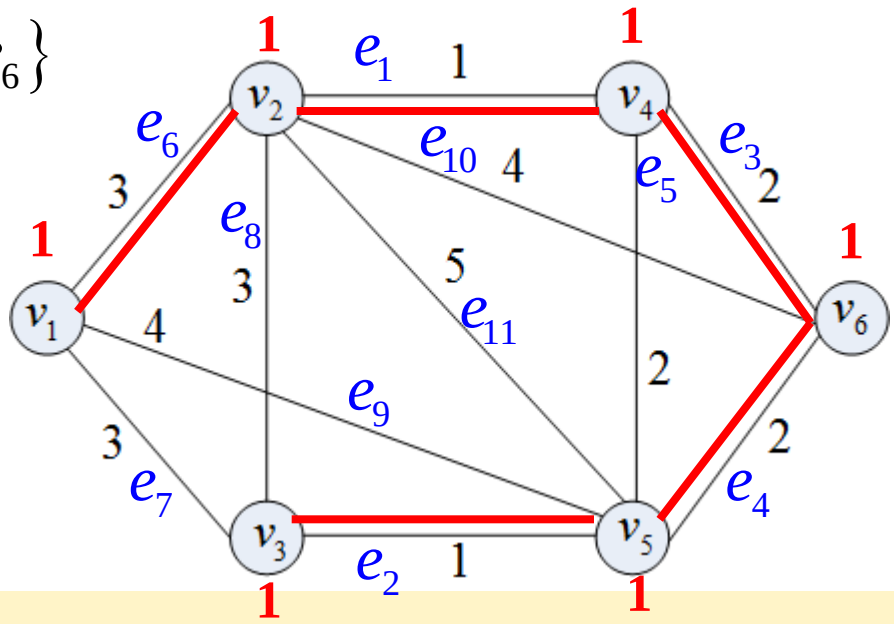
$i\neq 5, e_4$ 两端点标号不同，$S=S\cup\{e_4\}, i=4,\ j=5,$ 修改标号

$i\neq 5, e_5$ 两端点标号相同，$j=6$

$i\neq 5, e_6$ 两端点标号不同，$S=S\cup\{e_6\}$

$, i=5,\ j=7,$修改标号

$i=5,$计算结束



最小树：Dijkstra算法(1959，标号法)

出发点： 避免排序、回路判断；

设$G=(V,E,W)$是连通的无向网络，$|V|=n$，$|E|=m$

算法

Step 1. 置 $p_j=1;(2\le j\le n)$

$u_j=w_{1,j}, S=\phi, R=\{2,3,\cdots,n\}$

Step 2. 求 $u_k=\min_{j\in R}u_j=w_{i,k}$，置

$S=S\cup\{e_{p_k,k}\}, R=R\setminus\{k\}$

Step 3. 若 $R=\phi$，停；否则

对 $j\in R$,若$w_{k,j}<u_j$置$u_j=w_{k,j}$, $p_j=k$

转Step2；.

***S*: 树中的边；*R*:尚未关联到树中的顶点**

u_j:当前树外顶点到树中顶点的最小距离

p_j:指针，最小距离对应边的树中顶点号

算法说明：

- 假定G为简单图(非本质假设)
- 除初始步外，每次加入一个顶点、一条边
- 双重标号
- 显然得到一棵树
- 无需回路判断
- 计算复杂性P210
- 正确性：基于定理5.4.1或5.4.2

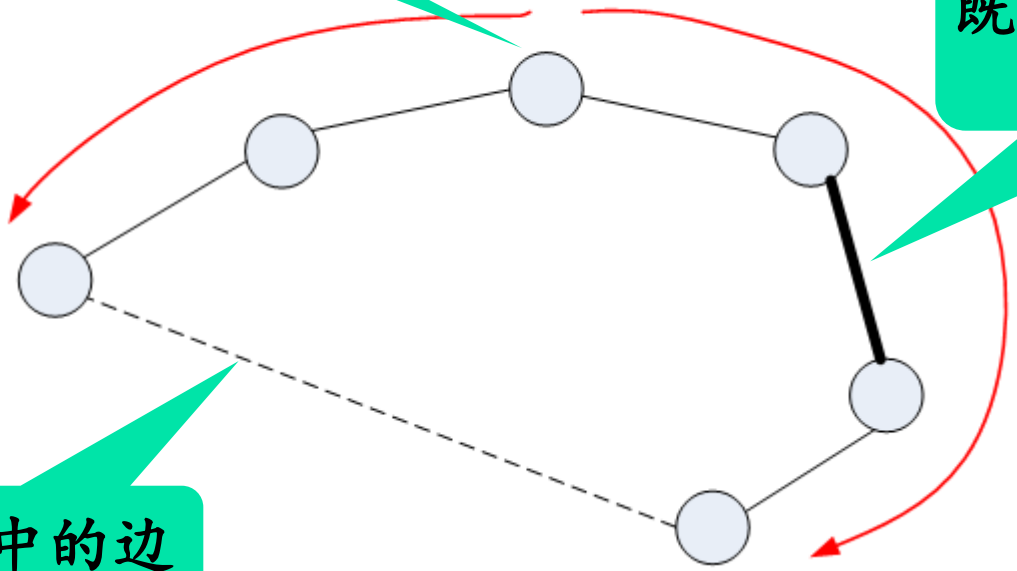
最小树：Dijkstra算法(1959，标号法)

Dijkstra算法的正确性：几何解释

$$\forall e \in T^*, w(e) = \max_{e' \in C(e)} w(e')$$

回路中第一个加入树中的顶点

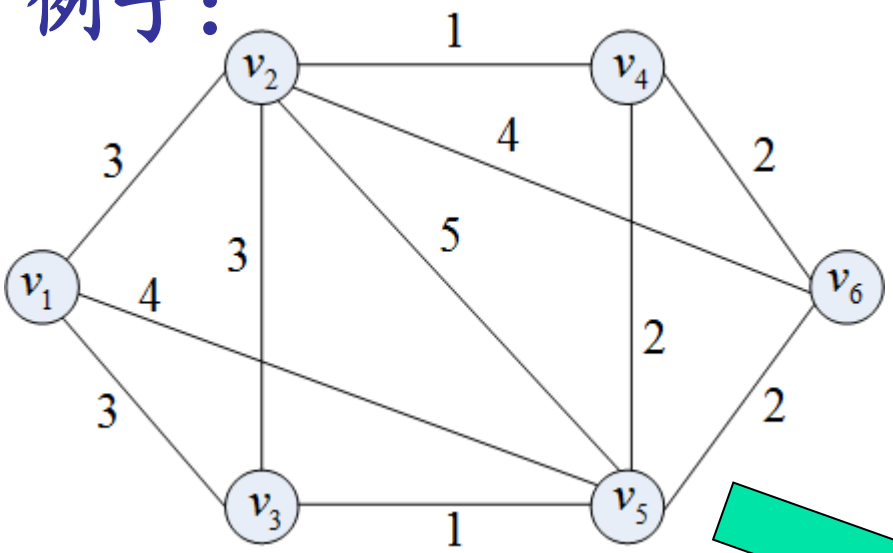
既在树中又在回路中的最大权边



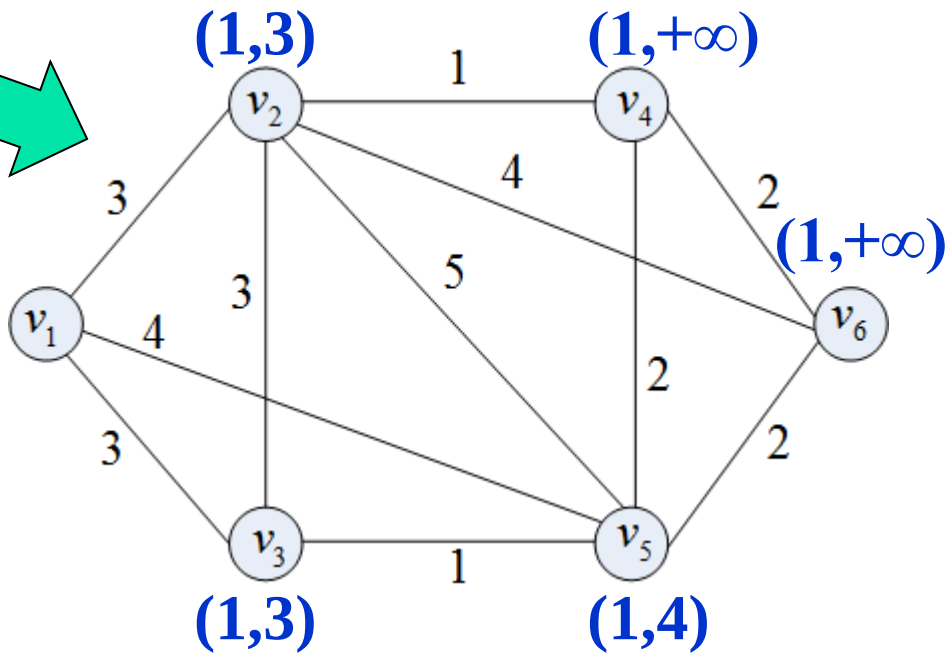
不在树中的边

最小树：Dijkstra算法(1959，标号法)

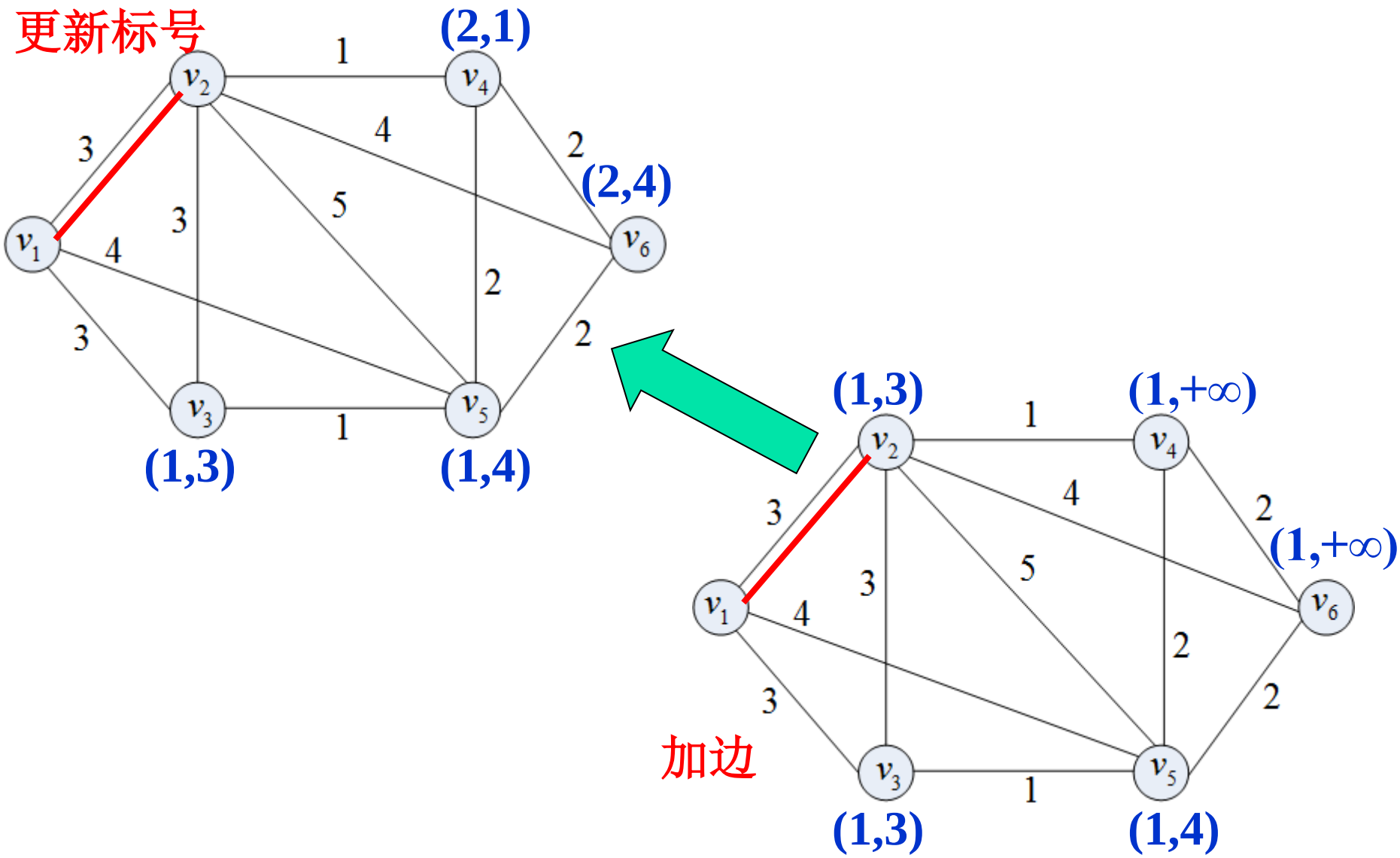
例子：



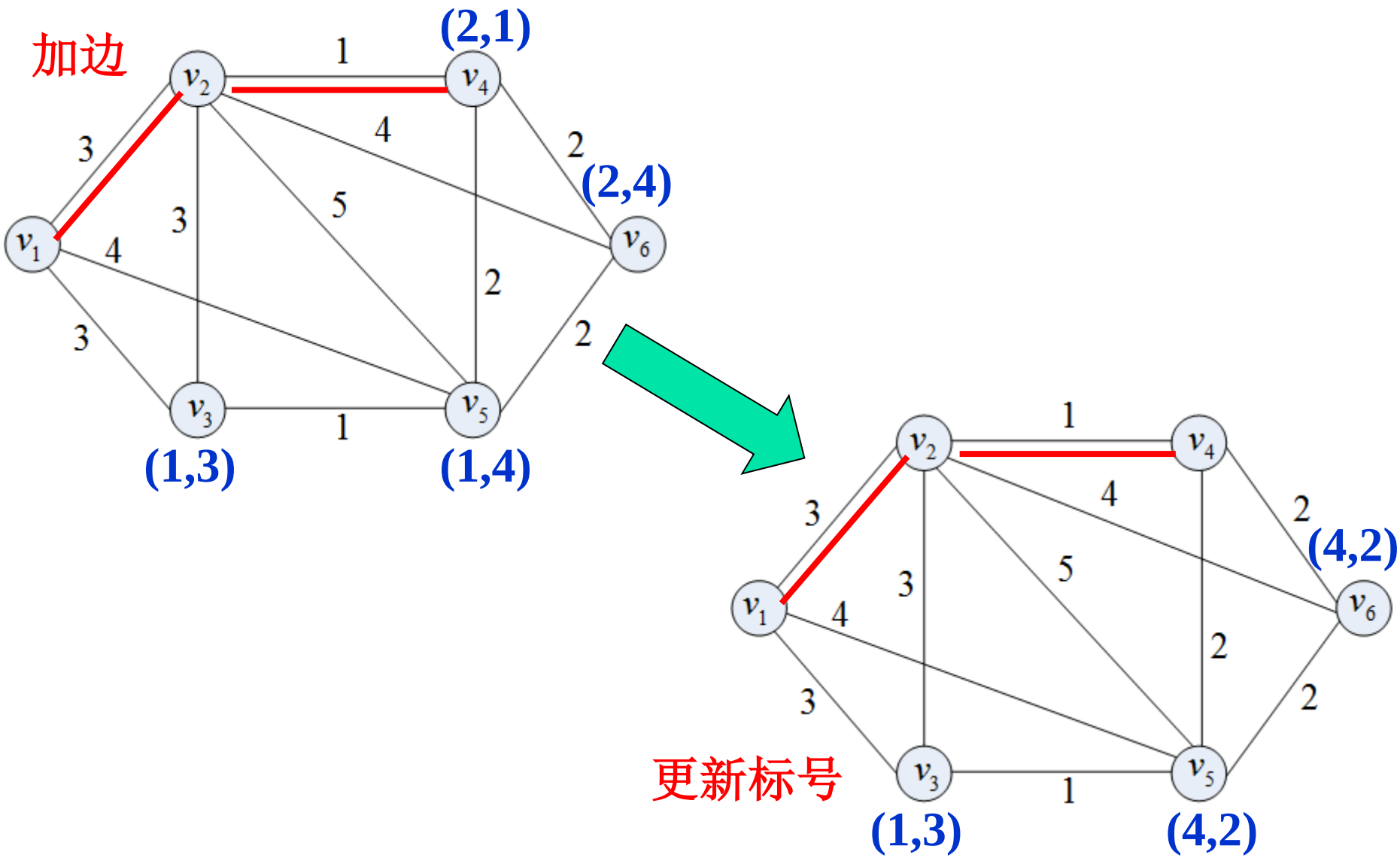
初始标号



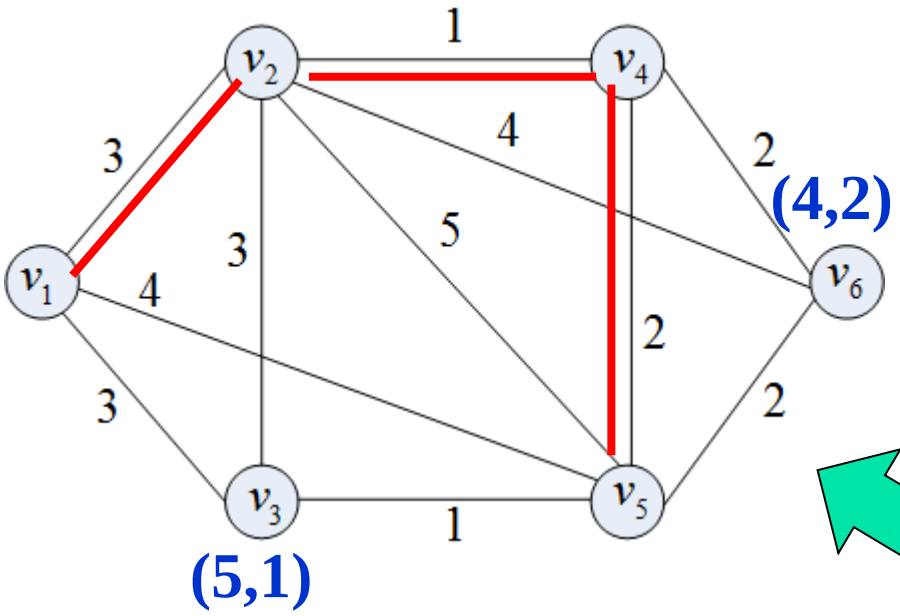
最小树：Dijkstra算法(1959，标号法)



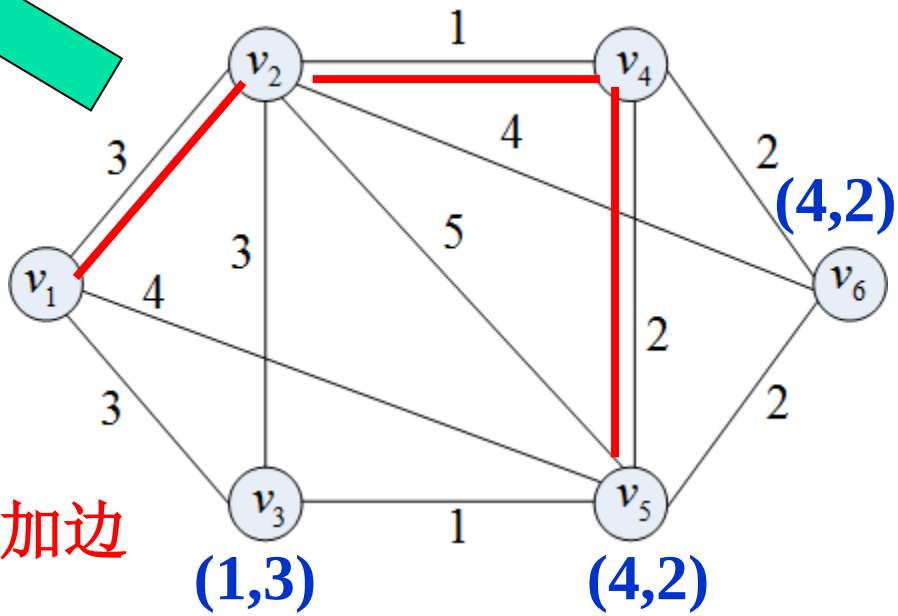
最小树：Dijkstra算法(1959，标号法)



最小树：Dijkstra算法(1959，标号法)

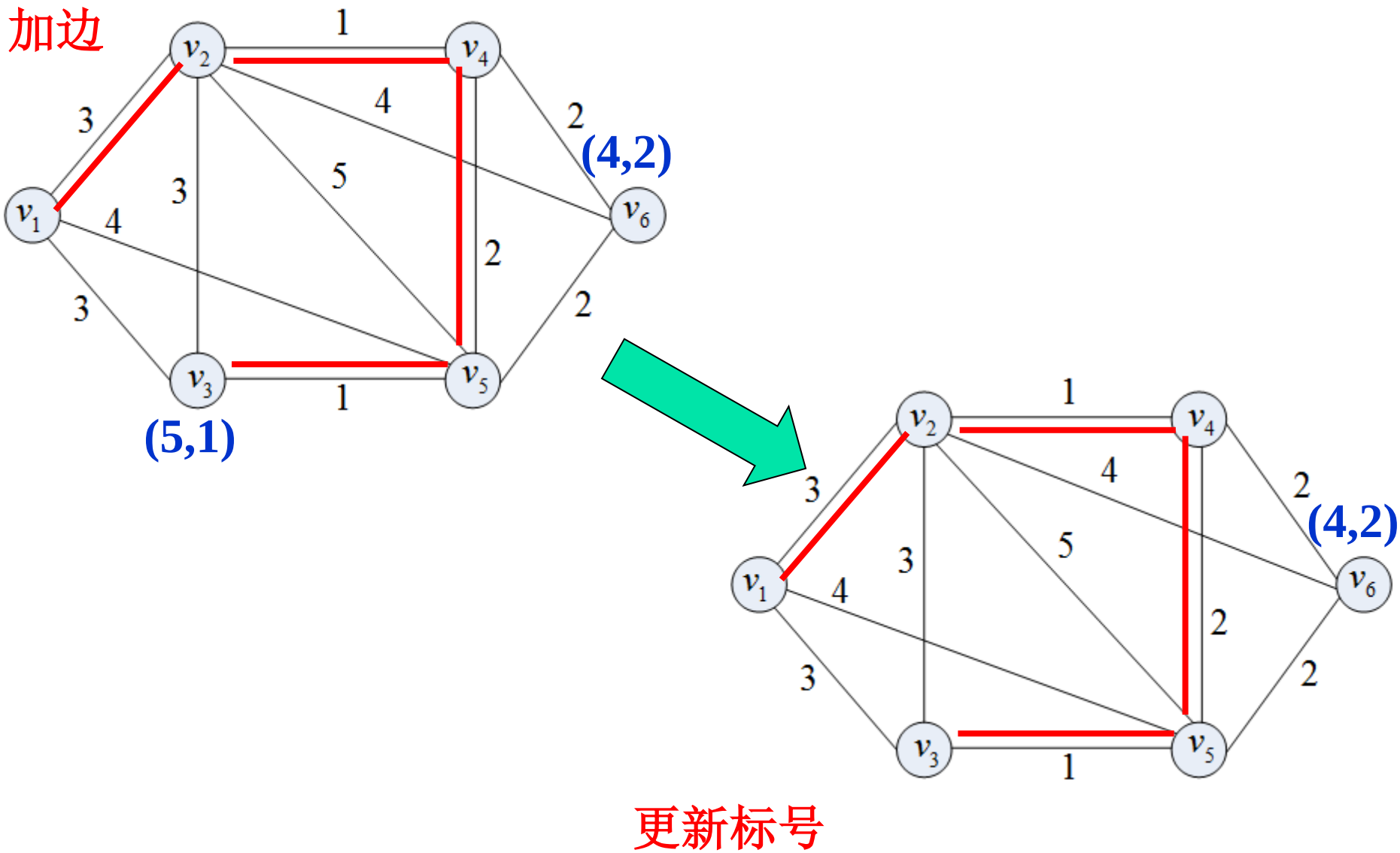


更新标号

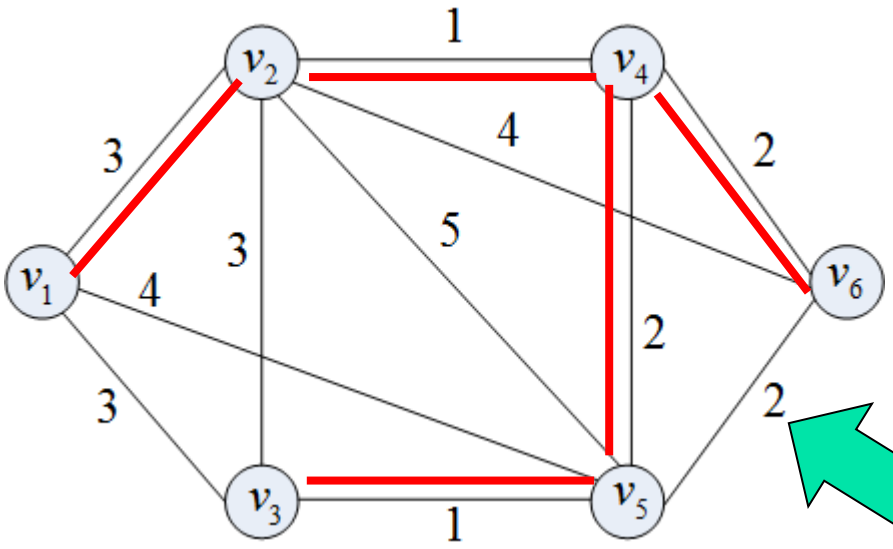


加边

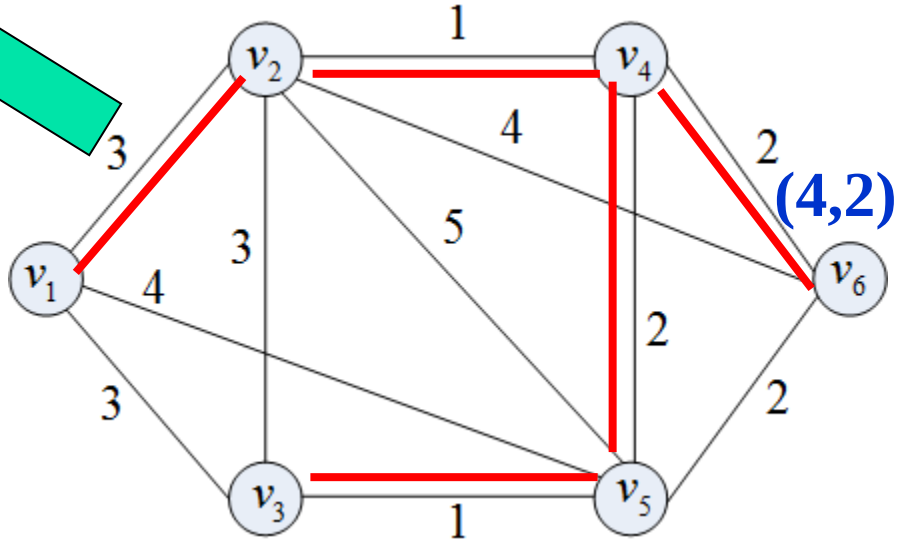
最小树：Dijkstra算法(1959，标号法)



最小树：Dijkstra算法(1959，标号法)



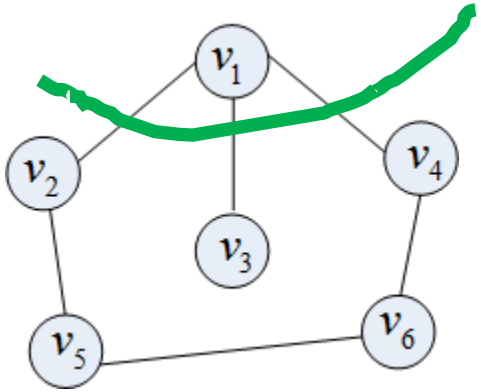
计算结束



加边

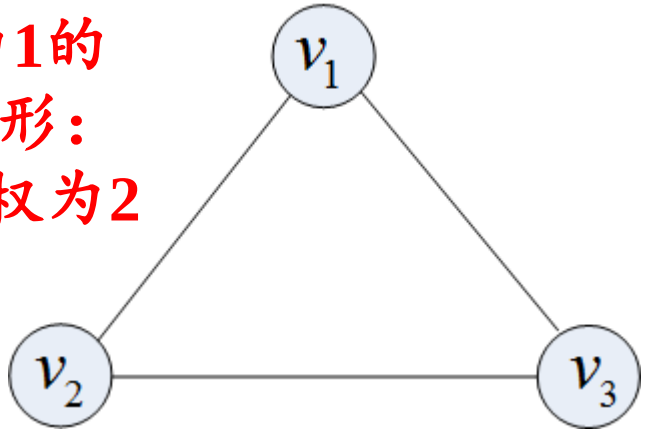
最小树：补充内容

勘误： P209算法解释，不是割集；P210图无标注

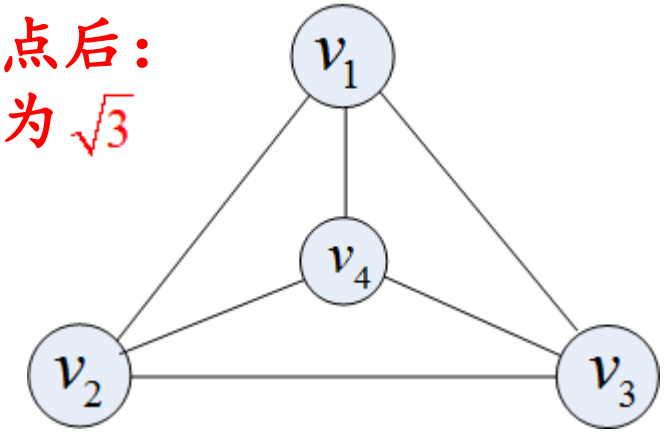


Steiner树：

边长为1的
正三角形：
最小树权为2



补充中心点后：
最小树权为 $\sqrt{3}$



谢谢！