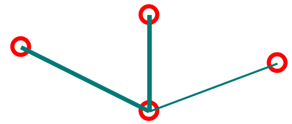
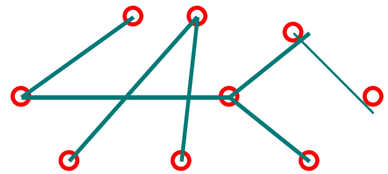
树与生成树

树 (Tree)



(a)

1.树的定义:一个连通无圈的无向图T,称之为树.



(b)

2.树叶:度数为1的顶点，称为树叶.

3.分支顶点(内顶点):度数大于1的顶点.

4.森林(forest):一个无向图的每个连通分支都是树.

5.与树定义等价的几个命题

定理1. 给定图T, 以下关于树的定义是等价的.

(1) T是无圈的连通图.

(2) T中每对顶点之间有一条且仅有一条路.

(3) T无圈但在任一对不相邻的顶点间添加一条新边e，则$T+e$包含唯一的圈.

(4) T连通的,且每条边都是割边.

(5) T连通的且$m=n-1$.

(6) T无圈且$m=n-1$.

证明: (1)⇒(2): 已知T是连通无圈的图,所以T中任意两点之间有路。若T中存在两条不同的u-v路P_1和P_2，则存在P_1的一条边$e=xy$，它不是P_2的边，因此x,y在$(P_1 \cup P_2)-e$连通，所以，$(P_1 \cup P_2)-e$存在一条x-y路P，故$P+e$是T的圈，矛盾。

(2)T中每对顶点之间有一条且仅有一条路.

(3) T无圈但在任一对不相邻的顶点间添加一条新边e，则T+e包含唯一的圈.

(4) T连通的,且每条边都是割边.

(2)⇒(3):显然T无圈，否则对圈上的任一对顶点都至少存在两条路，与(2)矛盾。设u，v是T中任意两个不相邻的点，令$e=uv$，由(2)，T中有一条唯一的u-v路，所以T+e中包含唯一的圈。

(3)⇒(4):因为T无圈，所以T的每条边都是割边；若T不连通，设T_1，T_2是T的两个分支，设$u\in V(T_1)$, $v\in V(T_2)$, 则$uv\notin E(T)$，显然T+uv不存在圈，与(3)矛盾。

(4) T连通的,且每条边都是割边.

(5) T连通的且$m=n-1$.

(4)⇒(5):关于点数用归纳法证明。

当$n=1$或2时，T是平凡图或K_2，显然有$m=n-1$。

假设$n \le k$时结论成立，往证$n=k+1$时成立。

当$n=k+1$时。取T的一条边e，由(4)，e是割边，

所以$T-e$有两个分支T_1和T_2,

因为$|V(T_1)| \le k$, $|V(T_2)| \le k$,且T_1和T_2每条边都是割边,

所以，由归纳假设，有

$|E(T_1)|=|V(T_1)|-1$， $|E(T_2)|=|V(T_2)|-1$

故$m=|E(T_1)|+|E(T_2)|+1$

$=|V(T_1)|-1+|V(T_2)|-1+1$

$=n-1$。

(5) T连通的且$m=n-1$.

(6) T无圈且$m=n-1$.

- (5)⇒(6):只需证明T无圈。对T关于顶点数归纳。
- 当$n=1$或$n=2$时，显然成立。
- 假设$n \le k$时结论成立，往证$n=k+1$时成立。
- 因为T连通，所以$\delta(T) \ge 1$，由$m=n-1$及$\sum d(v)=2m$得，T中至少存在一点u，使得$d(u)=1$。考虑$T'=T-u$，显然T'连通，且$|E(T')|=|V(T')|-1$，由归纳假设，T'无圈，所以T无圈。

(6) T无圈且$m=n-1$.

(1) T无圈的连通图.

(6)⇒(1):假设T不连通，设$T_1, T_2,\dots, T_k$为T的连通分支，则$k\geq 2$。对任意的T_i，T_i是无圈的连通图，所以T_i是树，由(6)得，

$$|E(T_i)|=|V(T_i)|-1$$

故$m=\sum_{i=1}^{k}|E(T_i)|=\sum_{i=1}^{k}|V(T_i)|-k$

$$=n-k<n-1$$

矛盾。

定理2：每一非平凡树至少有两片树叶。

证明：设T是一非平凡树，则$m=n-1$。

因为T连通且非平凡，所以$\delta(T)\geq 1$。

设T有k片树叶，则剩余的$n-k$个顶点的度至少为2。由$\sum d(v)=2m$得，$k+2(n-k)\leq m=2(n-1)$,

所以 $k\geq 2$。

树的中心问题

- G是一个图，G的一个点u的**离心率**(eccentricity)$e_G(u)$(简记$e(u)$)定义为

$$e(u)=\max\{d(u,v)|v\in V(G)\}.$$

- G的半径(radius)rad(G)定义为

$$\mathrm{rad}(G)=\min\{e(u)|u\in \mathrm{V}(G)\}.$$

- 若$e(u)=\mathrm{rad}(G)$，则称u为G的**中心点**(central vertex)
- 中心集$C(G)=\{u\in V(G)|u$为G的中心点$\}$.

- 定理 任意树T的中心或者恰含一个点，或者恰含两个相邻点。
- 证明：假设T有两个不相邻的中心点x和y
- 用$P(x,y)$表示T中x到y的唯一路
- 取$P(x,y)$中异于x, y的点z
- 令w是T中距离z最远的点，即$e(z)=d(z,w)$
- 若$P(z,w)$的第二点是$P(z,y)$的第二点，
- 则$P(z,w)\cap P(z,x)=\{z\}$.
- 若$P(z,w)$的第二点不是$P(z,y)$的第二点，
- 则$P(z,w)\cap P(z,y)=\{z\}$.
- 不妨假设$P(z,w)\cap P(z,x)=\{z\}$，则
- $\mathrm{rad}(T)=e(x)\geq d(x,w)=d(x,z)+d(z,w)>d(z,w)=e(z)$,
- 矛盾。

二. 生成树

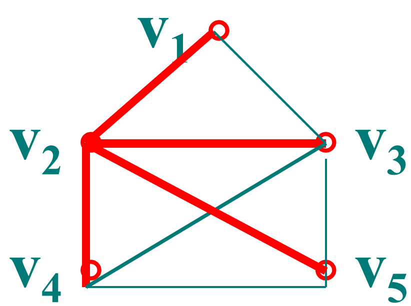
1.定义:如果图G的生成子图是树,则称此树为G的生成树.

2.弦:图G中,不在其生成树里的边,称作**弦**. 由所有弦生成的子图称为该生成树的**余树**.

定理2 连通图G中至少有一棵生成树.

证明:如果G中无圈, 则G本身就是树.

如果G中有圈,可以通过反复删去圈中的边,使之既无圈,又连通.就得到生成树.

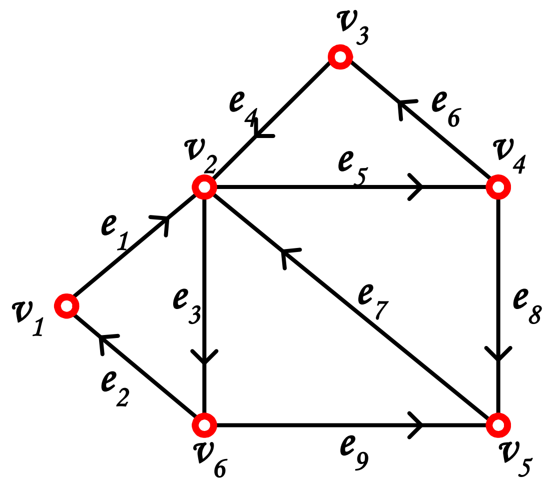


思考题:设G是有n个结点,m条边的连通图, 问要删去多少条边,才得到一棵生成树?

§ 2. 基本关联矩阵及其性质

- ----讨论对象为有向(弱)连通图G
- 基本关联矩阵:在G=<V, E>的关联矩阵B中划去任意点v_k所对应的行, 得到一个$(n\text{-}1)\times m$矩阵B_k.

$$\mathbf{B}(\mathbf{G})=\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{6\times 9}$$



$$\mathbf{B}_3=\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{5\times 9}$$

- **定理1**. ran(B)<n.
- 证明: ∵B的每列恰好只有1和-1两个非零元.
- ∴ B的第1至n-1行加到第n行后, 第n行全为0. 即B的n个行向量线性相关.
- ∴ ran(B)<n.
- **定理2**. 设B_0为B(G)的任意一k阶方阵, 则
- $$|B_0|=\pm 1\text{或}0.$$
- 证明: 对k归纳. k=1时, 成立.
- 假设k-1时成立, 则当B_0为B(G)的任一k阶方阵时, ∵ B_0为B的子阵, ∴ B_0每列最多只有2个非零元. 若其中某一列全为0或B_0中每列恰好有2个非零元, 则$|B_0|=0$.
- 假设B_0中存在只有一个非零元的列, 则按该列展开后用归纳法即可.

- 定理3. 设B为有向弱连通图G的关联矩阵, 则ranB=n-1.
- 证明: 由定理1. ranB≤n-1, 故只需证ranB≥n-1.
- 设B中线性相关最少的行数为l, 则l≤n.
- 设这l行分别与点 $v_{i_1}, v_{i_2}, \cdots v_{i_l}$ 相对应，则有

$$k_1 b(i_1) + k_2 b(i_2) + \cdots + k_l b(i_l) = 0 \qquad \forall j = 1, 2, \ldots, l. \quad k_j \neq 0 \qquad (*)$$

- ∵B的每列只有2个非零元, ∴ 这l 个行向量中, 其第t(t=1, 2, …, m)个分量最多只有2个非零元, 且不可能只有1个非零元(可以全为0元).
- 否则, (*)式不成立.
- 对B进行行、列交换, 使前l行为$b(i_1)$, …, $b(i_l)$, 并且每列都有2个非零元的换到前r列, 其余m-r列全都为0.即

$$B \to \begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix}\begin{matrix} l \\ n-l \end{matrix} = B'.$$

- 显然, ran(B)=ran(B'), 且B'依然是G的一个关联矩阵.
- 若n-l>0, 则由B'可知, G至少分为2个连通分支. 其中r条边只与l个点相关, 而其余m-r条边只与另外n-l个点相关.与G连通矛盾.
- ∴ n-l=0. ⇒ l=n. ⇒ B中最少需要n行才能线性相关, 而任何n-1行线性无关.
- ∴ ran(B)≥n-1.

- **定理4**. 有向连通图G的基本关联矩阵B_k的秩为n-1.
- **定理5**. 设B_k为有向连通图G的基本关联矩阵, C是G中的一个圈, 则C中各边所对应B_k的各列线性相关.
- 证明: 设C为G的长为l的圈, C包含l个点. (不妨设$l<n$.)
- 设这l条边对应关联矩阵B的l列, 它们构成B的子阵$B(G_C)$.
- ∵ C连通, ∴ $ran(B(G_C))=l-1$.
- **推论1**. 设H是有向连通图G的子图. 若H含有圈, 则H的诸边对应的G的基本关联矩阵各列线性相关.
- **定理6**. 令B_k是有向连通图G的基本关联矩阵, 则B_k的任意n-1阶子阵行列式≠0 ⇔ 其各列所对应的边构成G的一棵生成树.

§ 3. 生成树的计数

- 定理1. (Binet—Canch 定理)A=$(a_{ij})_{m\times n}$, B=$(b_{ij})_{n\times m}$, $m\le n$. 则 $|AB|=\sum_i A_iB_i$ 其中A_i, B_i为m阶行列式. A_i是从A中取不同的m列所成的行列式. B_i是从B中取相应的m行构成的行列式. 对全部组合求和.

- 例:

$$\mathbf{A}=\begin{bmatrix}1 & 0 & -1\\ 2 & 1 & 0\end{bmatrix}_{2\times 3},\quad \mathbf{B}=\begin{bmatrix}0 & 1\\ -1 & 0\\ 1 & 1\end{bmatrix}_{3\times 2}$$

$$\mathbf{AB}=\begin{bmatrix}-1 & 0\\ -1 & 2\end{bmatrix},\quad |\mathbf{AB}|=-2.$$

$$|\mathbf{AB}|=\begin{vmatrix}1 & 0\\ 2 & 1\end{vmatrix}\begin{vmatrix}0 & 1\\ -1 & 0\end{vmatrix}+\begin{vmatrix}1 & -1\\ 2 & 0\end{vmatrix}\begin{vmatrix}0 & 1\\ 1 & 1\end{vmatrix}+\begin{vmatrix}0 & -1\\ 1 & 0\end{vmatrix}\begin{vmatrix}-1 & 0\\ 1 & 1\end{vmatrix}=-2.$$

有向连通图的生成树计数

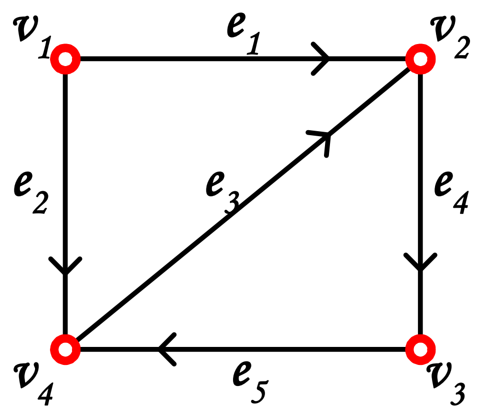
- 定理2. 设$\mathbf{B}_k$是有向连通图G=<V, E>的某一基本关联矩阵, 则G的不同生成树的数目是$|B_k B_k^T|$
- 证明: 设$\mathbf{B}_k=(\mathbf{b}_{ij})_{(n-1)\times m}$, ∵ G连通, ∴ $m\geq n$-1.

$$|B_k B_k^T| = \sum_i |B_i||B_i^T| = \sum_i |B_i|^2.$$

- 其中$|\mathbf{B}_i|$为$\mathbf{B}_k$的某一n-1阶子阵的行列式.
- 若$|\mathbf{B}_i|^2\neq 0$, ⇒ $|\mathbf{B}_i|\neq 0$, ⇒ 其所对应的边构成G的一棵生成树.
- ∵ $|\mathbf{B}_i|=\pm 1$, ∴ 如果$\mathbf{B}_i$的各列所对应的边构成G的一个棵, 则对$|B_k B_k^T|$的贡献为1.
- ⇒ 恰为G中不同生成 树的数目.

❖ 例:

$$B=\begin{bmatrix}1&1&0&0&0\\0&-1&-1&1&0\\0&0&0&-1&1\\-1&0&1&0&-1\end{bmatrix}$$



$$B_3=\begin{bmatrix}1&1&0&0&0\\0&-1&-1&1&0\\-1&0&1&0&-1\end{bmatrix}$$

$$|B_3B_3^T|=\begin{vmatrix}2&-1&-1\\-1&3&-1\\-1&-1&3\end{vmatrix}=18-1-1-3-2-3=8.$$

❖ 无向连通图的树计数.

❖ 例. 求$\mathbf{K}_n$中不同树的数目.

❖ 解: $\mathbf{K}_n$中的边任给一方向, 得到竞赛图G.

$$|B_k B_k^T| = \begin{vmatrix} n-1 & -1 & \cdots & -1 \\ -1 & n-1 & \cdots & -1 \\ & \cdots & \cdots & \\ -1 & -1 & \cdots & n-1 \end{vmatrix}_{n-1} = \begin{vmatrix} 1 & 1 & \cdots & 1 \\ -1 & n-1 & \cdots & -1 \\ & \cdots & \cdots & \\ -1 & -1 & \cdots & n-1 \end{vmatrix}$$

$$= \begin{vmatrix} 1 & 1 & \cdots & 1 \\ 0 & n & \cdots & 0 \\ & \cdots & \cdots & \\ & & \cdots & n \end{vmatrix} = n^{n-2}.$$

给定图$G=(V,E)$，$X\subset V(G),u\in X$,定义

$\Phi(X)=\{uv\mid u\in X, v\in V\setminus X\},\Phi(u,X)=\{uv\mid v\in V\setminus X\}$

- **深探法(depth-first-search):求以点u为树根的深探树**

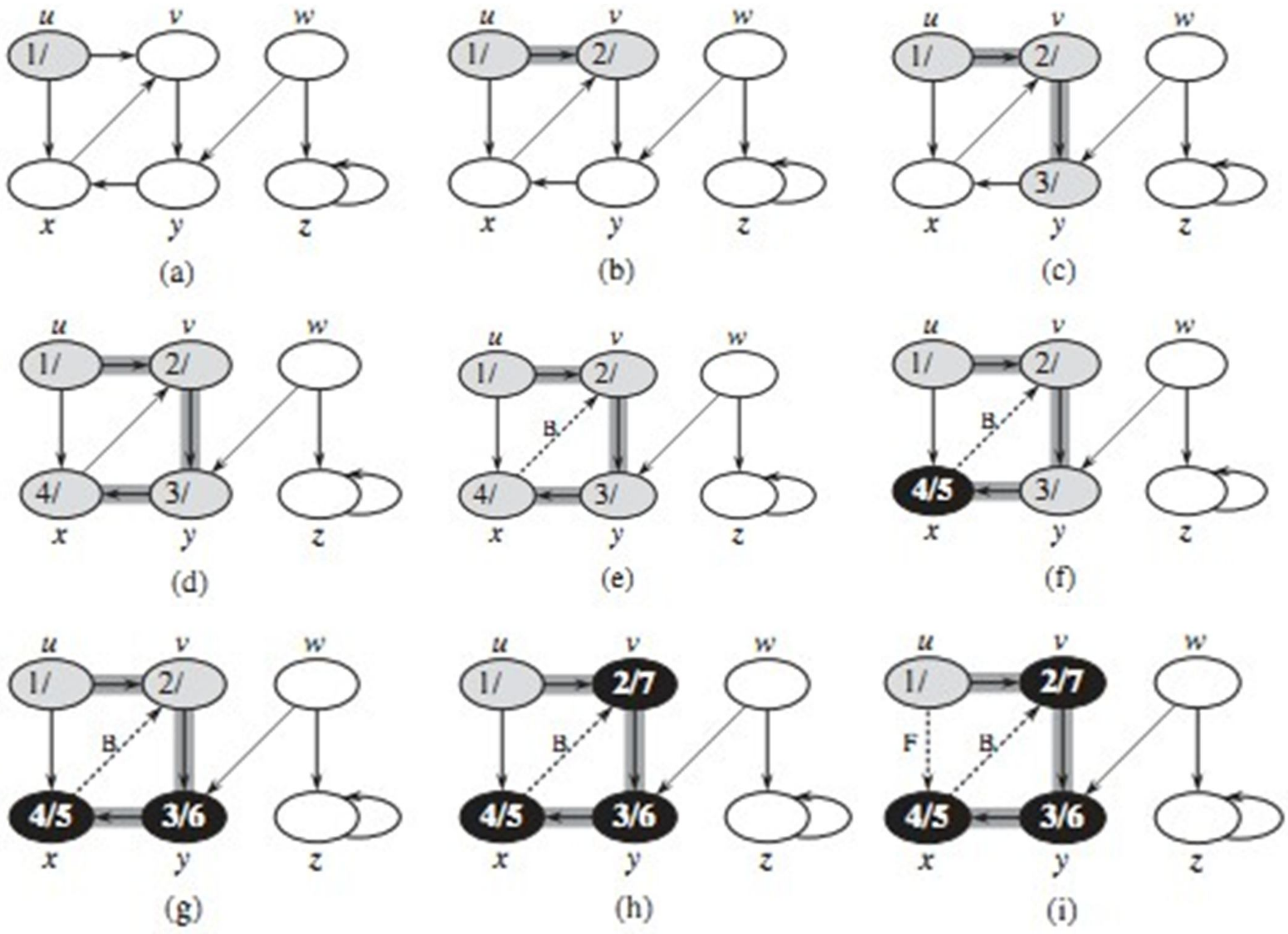
(1) $k=0$,令$X^{(0)}=\{u\}$,记$\lambda(u)=0; E^{(0)}=\varnothing$.

(2)设已知$X^{(k)}$，设点$x\in X^{(k)}$,使得

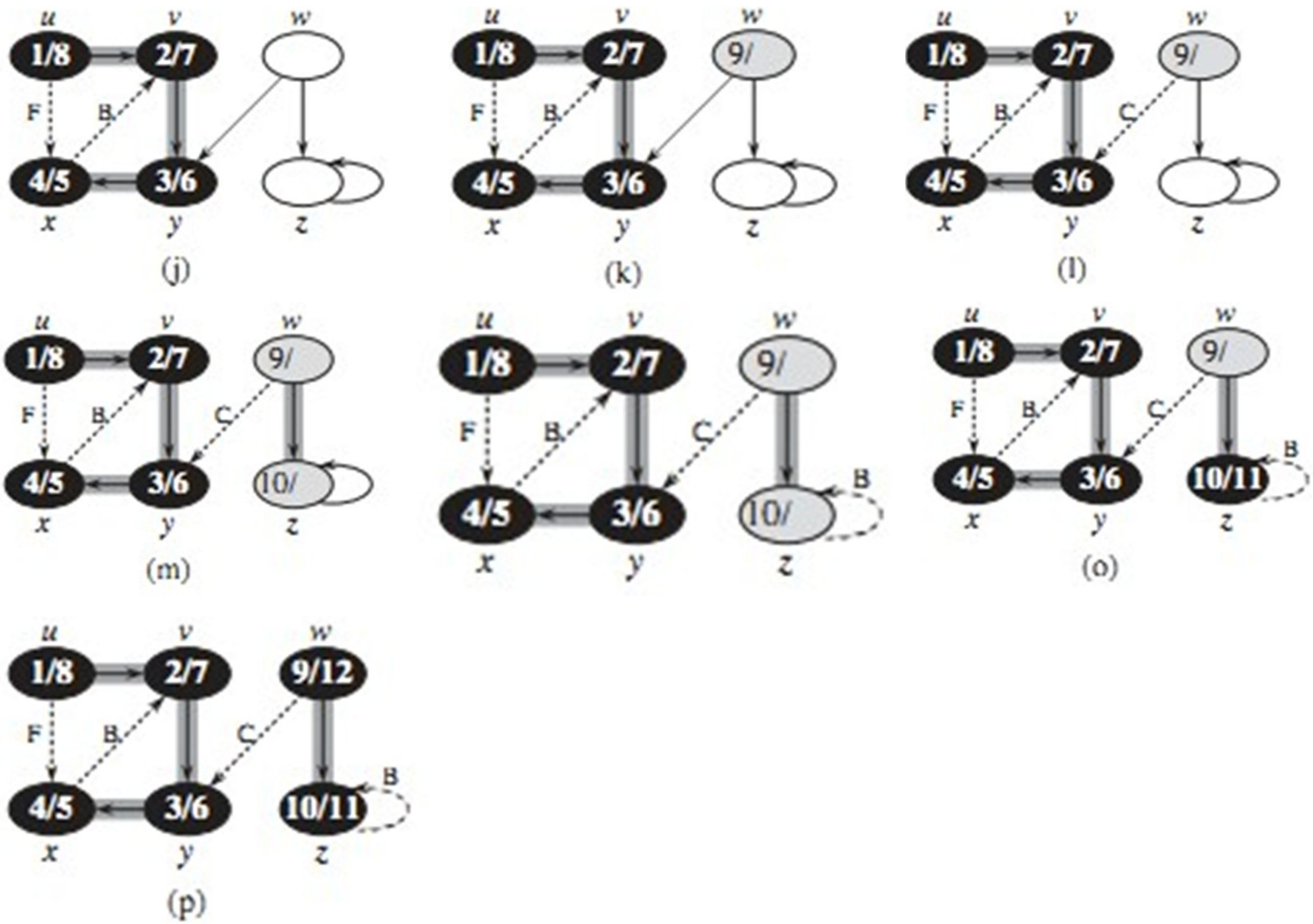
$$\lambda(x)=\max\{\lambda(v)\mid v\in X^{(k)},\Phi(v,X^{(k)})\neq\varnothing\}.$$

在$\Phi(x,X^{(k)})$中选取边, 设为xy,令$X^{(k+1)}=X^{(k)}\cup\{y\}$，$\lambda(y)=\lambda(x)+1$; $E^{(k+1)}=E^{(k)}\cup\{xy\}$，以$X^{(k+1)}$，$E^{(k+1)}$重复选边过程。

（3）如果在某一步，$X^{(k)}=V$，终止，得到图G的以点u为树根的深探树;若在某一步,$X^{(k)}\neq V$,而$\Phi(X^{(k)})=\varnothing$,终止, 此时$G$不连通。



u v w
1/
x y z
(a)
u v w
1/ 2/
x y z
(b)
u v w
1/ 2/
3/
x y z
(c)
u v w
1/ 2/
4/ 3/
x y z
(d)
u v w
1/ 2/
B
4/ 3/
x y z
(e)
u v w
1/ 2/
B
4/5 3/
x y z
(f)
u v w
1/ 2/
B
4/5 3/6
x y z
(g)
u v w
1/ 2/7
B
4/5 3/6
x y z
(h)
u v w
1/ 2/7
F B
4/5 3/6
x y z
(i)



u v w
1/8 2/7
F B
4/5 3/6
x y z
(j)
u v w
1/8 2/7 9/
F B
4/5 3/6
x y z
(k)
u v w
1/8 2/7 9/
F B C
4/5 3/6
x y z
(l)
u v w
1/8 2/7 9/
F B C
4/5 3/6 10/
x y z
(m)
u v w
1/8 2/7 9/
F B C B
4/5 3/6 10/
x y z
u v w
1/8 2/7 9/
F B C B
4/5 3/6 10/11
x y z
(o)
u v w
1/8 2/7 9/12
F B C B
4/5 3/6 10/11
x y z
(p)

- **DFS**搜索是一种在开发爬虫早期使用较多的方法。它的目的是要达到被搜索结构的叶结点
- **DFS**搜索是图论中的经典算法，利用深度优先搜索算法可以产生目标图的相应拓扑排序表，利用拓扑排序表可以方便的解决很多相关的图论问题，如最大路径问题等等。
- 因发明“深度优先搜索算法”，霍普克洛夫特与陶尔扬共同获得计算机领域的最高奖：图灵奖.

❖ 广探法(breadth-first-search):求以点u为树根的广探树

(1) $k=0$, 令$X^{(0)}=\{u\}$, 记$\lambda(u)=0; E^{(0)}=\varnothing$.

(2)设已知$X^{(k)}$， 设点$x\in X^{(k)}$,使得

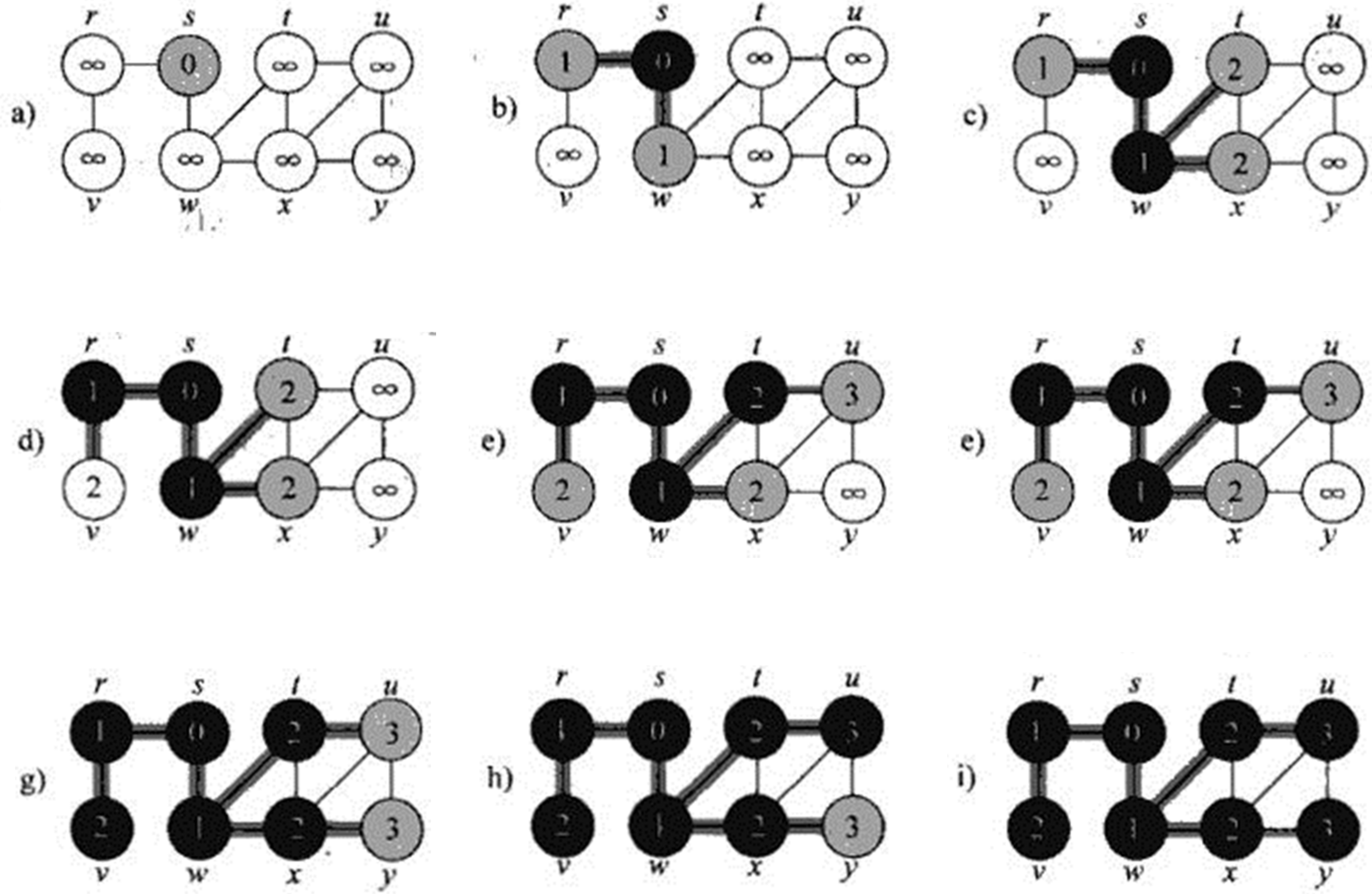
$$\lambda(x)=\min\{\lambda(v)\mid v\in X^{(k)}, \Phi(v,X^{(k)})\neq\varnothing\}.$$

在$\Phi(x,X^{(k)})$中选取所有边, 记被选边的集合为$F^{(k)}$, 把选取的边的端点放入$X^{(k)}$,得到$X^{(k+1)}$,并令这些点的λ值为$\lambda(x)+1$;

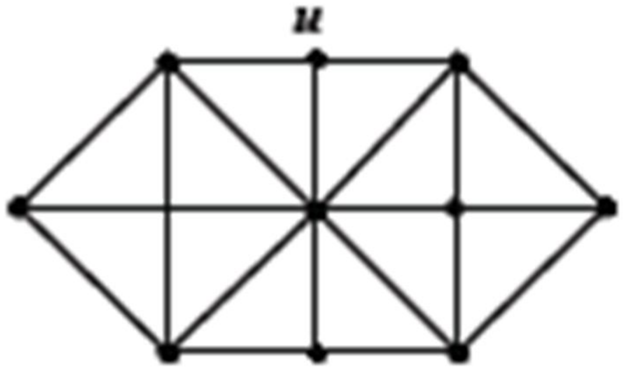
$E^{(k+1)}=E^{(k)}\bigcup F^{(k)}$， 以$X^{(k+1)}$， $E^{(k+1)}$重复选边过程。

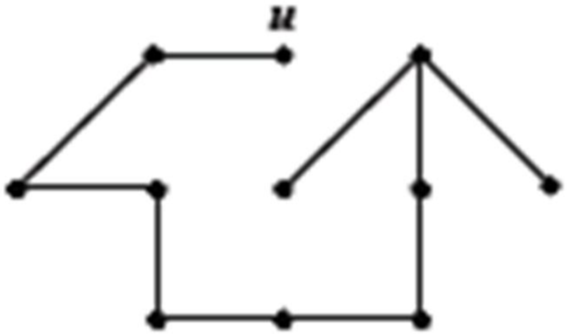
（3）如果在某一步， $X^{(k)}=V$， 终止， 得到图G的以点u为树根的广探树;若在某一步, $X^{(k)}\neq V$,而$\Phi(X^{(k)})=\varnothing$,终止, 此时$G$不连通。

性质：设T是图G中以点u为树根的广探树，则对任一点x，$\lambda(x)=d(u,x)$。

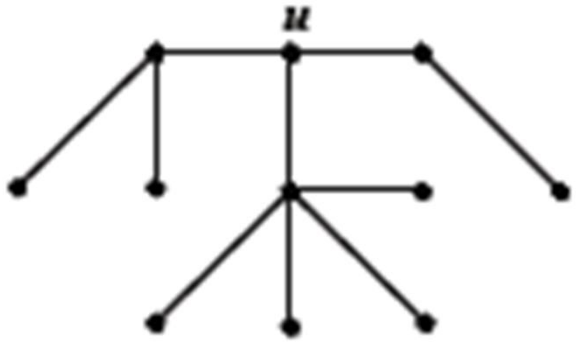

a)
b)
c)
d)
e)
e)
g)
h)
i)

- 广度优先搜索(**BFS**)算法是最简单的图的搜索算法之一，这一算法也是很多重要的图的算法的原型。**Dijksta**单源最短路径算法和**Prim**最小生成树算法都采用了与广度优先搜索类似的思想。
- 广度优先搜索算法可以求出图的直径。





深探树



广探树

- DFS和BFS的优缺点
- 1、深度优先算法占内存少但速度较慢，广度优先算法占内存多但速度较快，在距离和深度成正比的情况下能较快地求出最优解。
  2、深度优先与广度优先的控制结构和产生系统很相似，唯一的区别在于对扩展节点选取上。由于其保留了所有的前继节点，所以在产生后继节点时可以去掉一部分重复的节点，从而提高了搜索效率。
  3、这两种算法每次都扩展一个节点的所有子节点，而不同的是，深度优先下一次扩展的是本次扩展出来的子节点中的一个，而广度优先扩展的则是本次扩展的节点的兄弟点。在具体实现上为了提高效率，所以采用了不同的数据结构。

3.7 赋权图的最小生成树

1.定义:一棵生成树中的所有边的权之和称为该**生成树的权**. 具有最小权的生成树,称为**最小生成树**.

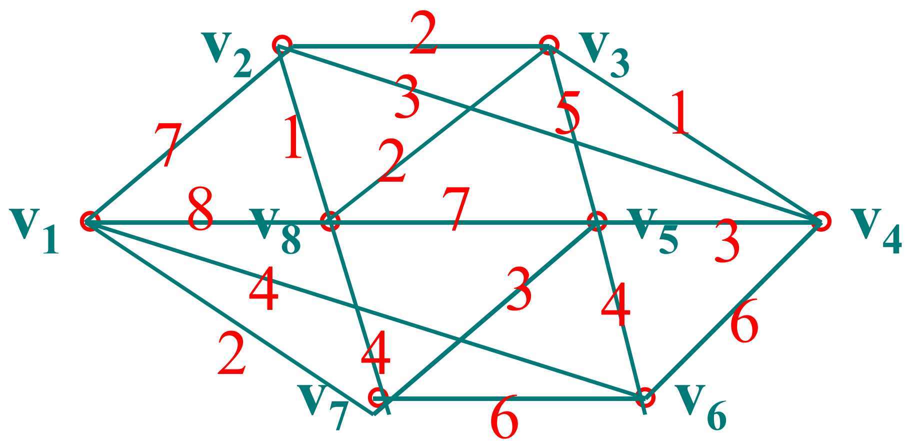
最小生成树很有实际应用价值.例如结点是城市名,边的权表示两个城市间的距离, 从一个城市出发走遍各个城市, 如何选择最优的旅行路线.又如城市间的通信网络问题,如何布线,使得总的线路长度最短.

例如:右图所示

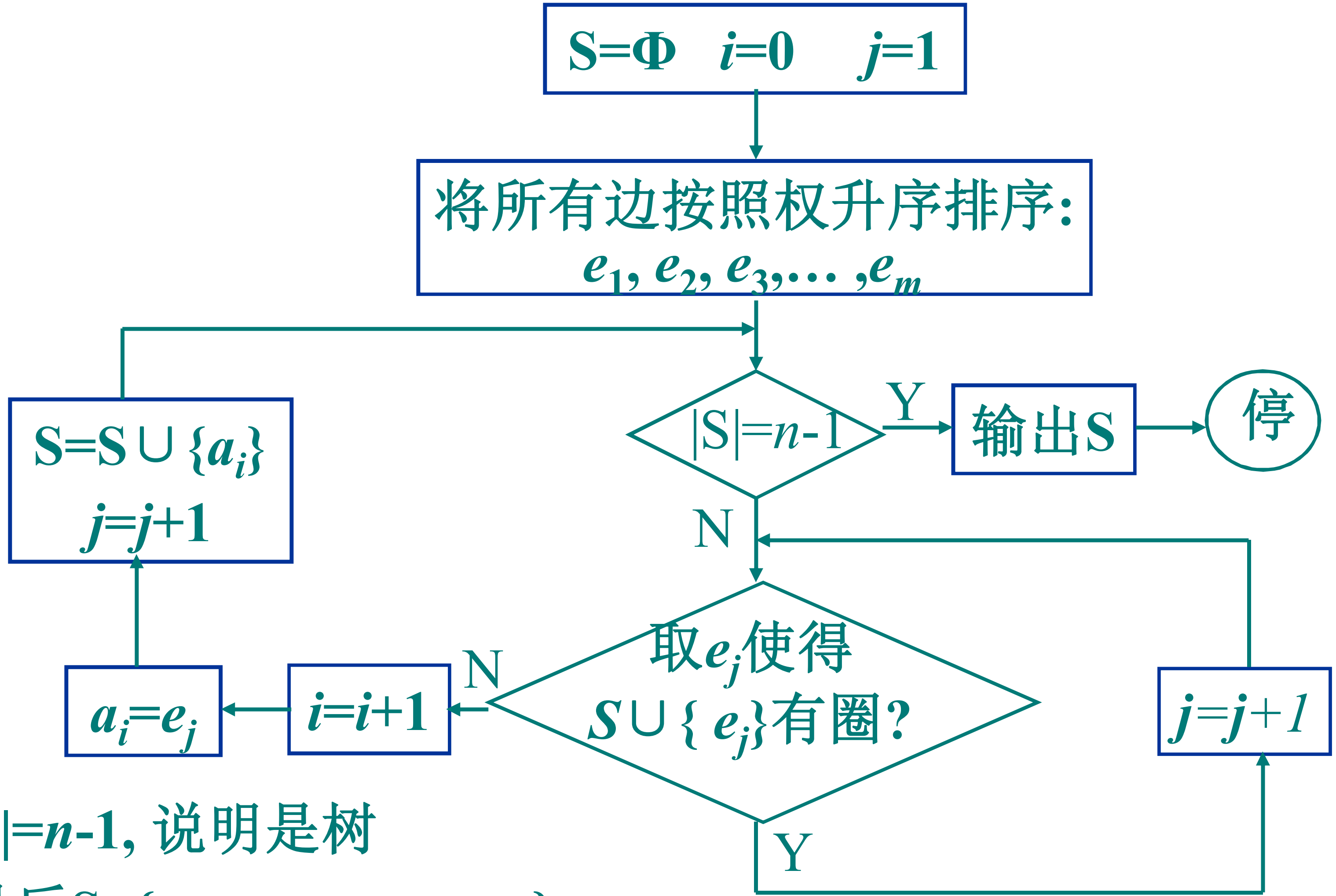
2.求最小生成树算法

---Kruskal算法:

(贪婪算法)



Kruskal算法: 设G是有n个结点,m条边($m \geq n$-1)的连通图.

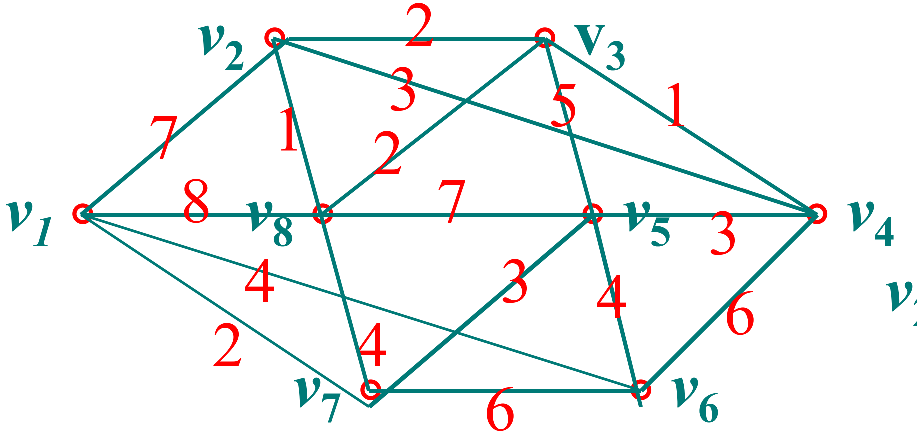


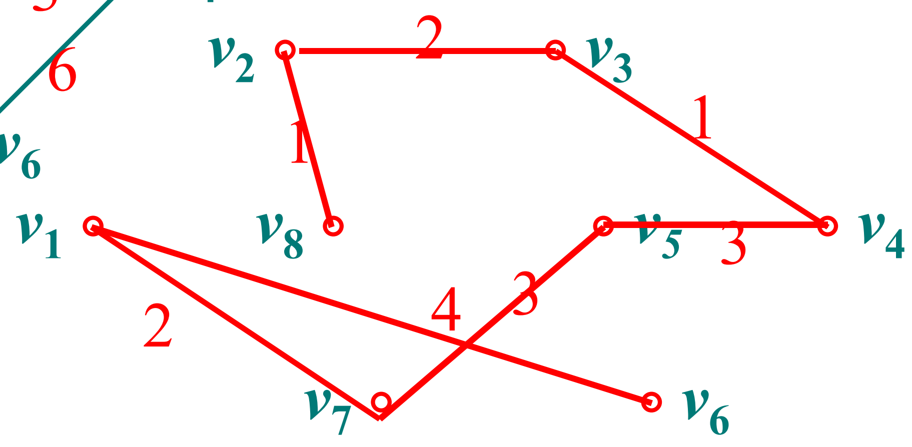
|S|=n-1, 说明是树

最后S={$a_1, a_2, a_3, \ldots, a_{n-1}$}

边按升序排序:边(v_i, v_j)记成e_{ij}

边	e_{28}	e_{34}	e_{23}	e_{38}	e_{17}	e_{24}	e_{45}	e_{57}	e_{16}
权	1	1	2	2	2	3	3	3	4
边	e_{78}	e_{56}	e_{35}	e_{46}	e_{67}	e_{58}	e_{12}	e_{18}	
权	4	4	5	6	6	7	7	8	





- 定理1.由**Kruskal**算法得到的生成树是最小生成树。
- 证明：设**Kruskal**算法得到的G的生成树为
- $T^*=\{e_1,e_2,\ldots,e_{n-1}\}$
- 假设T^*不是G的最小生成树。对G的任意一棵生成树T，定义 $f(T)=mim\{i\,|\,e_i \notin E(T)\}$

  取G的最小生成树T，使得 $f(T)$尽可能大，设$f(T)=k$

  则 $T+e_k$包含唯一的一个圈C，且C中存在边e，使得e在T中，但不在T^*中，
- 令$T'=(T-e)\bigcup e_k$，则T'也是G的生成树，且
- T'的权＝T的权＋e_k的权－e的权
- 由**Kruskal**算法，e的权≥e_k的权，故T'的权≤T的权
- 所以T'也是G的最小生成树，但$f(T')>f(T)$，矛盾

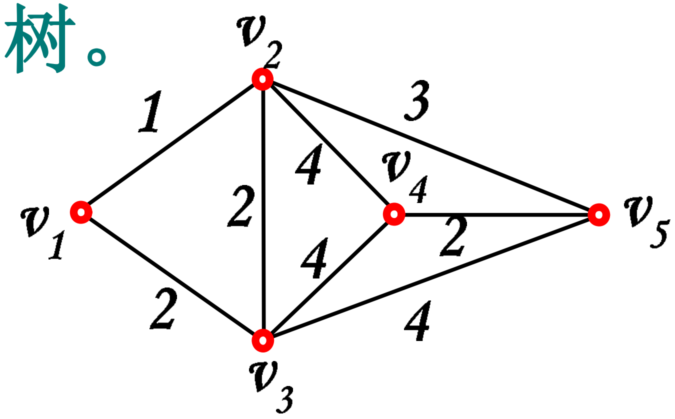
Prim算法(边割法)

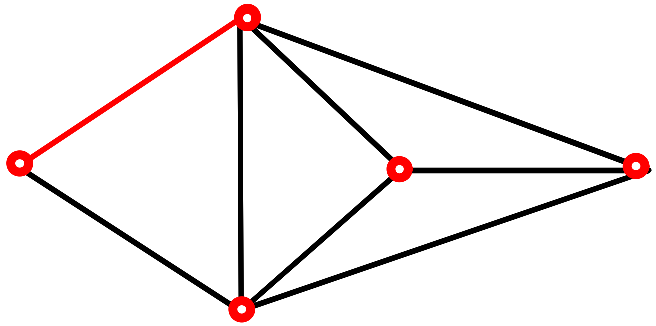
- 实质: 在n-1个边割集中, 取每个边割集的一条权最小的边, 构成G的一个生成树.
- 定义: $[S_k, \overline{S_k}] = \{uv \mid u \in S_k,\ \ v \in \overline{S_k}\}.$
- Step0. 设v为V的任一顶点. 令$S_0=\{v\}$, $E_0=\Phi$, $k=0$.
- Step1. 若$S_k=V$, 结束. 以S_k为点集, E_k为边集的图即是G的最优树. 否则转Step2.
- Step2. 构造$[S_k, \overline{S_k}]$, 若$[S_k, \overline{S_k}] = \Phi$, 则G不连通, 停止. 否则, 设

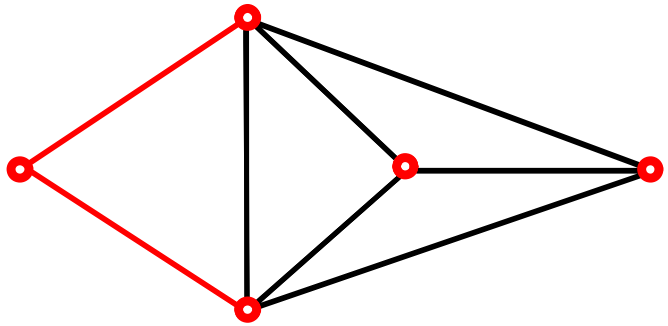
$$w(e_k) = \min_{e \in [S_k, \overline{S_k}]} w(e) \quad e_k = v_k v_k', \quad v_k \in S_k.$$

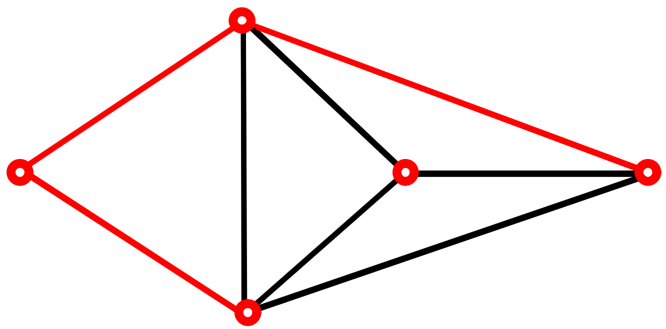
-
- 令 $S_{k+1} = S_k \cup \{v_k'\}, \quad E_{k+1} = E_k \cup \{e_k\}.$
- 置$k=k+1$. 返回Step1.

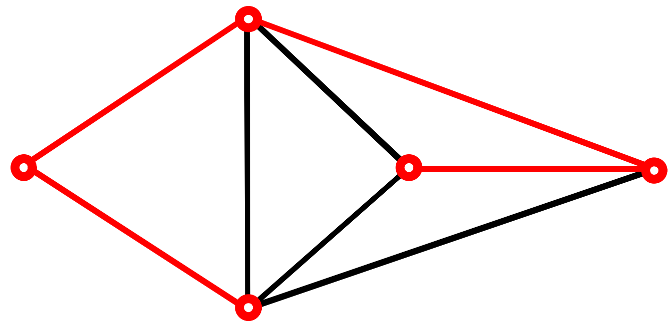
❖ 例：求图G的最小生成树。









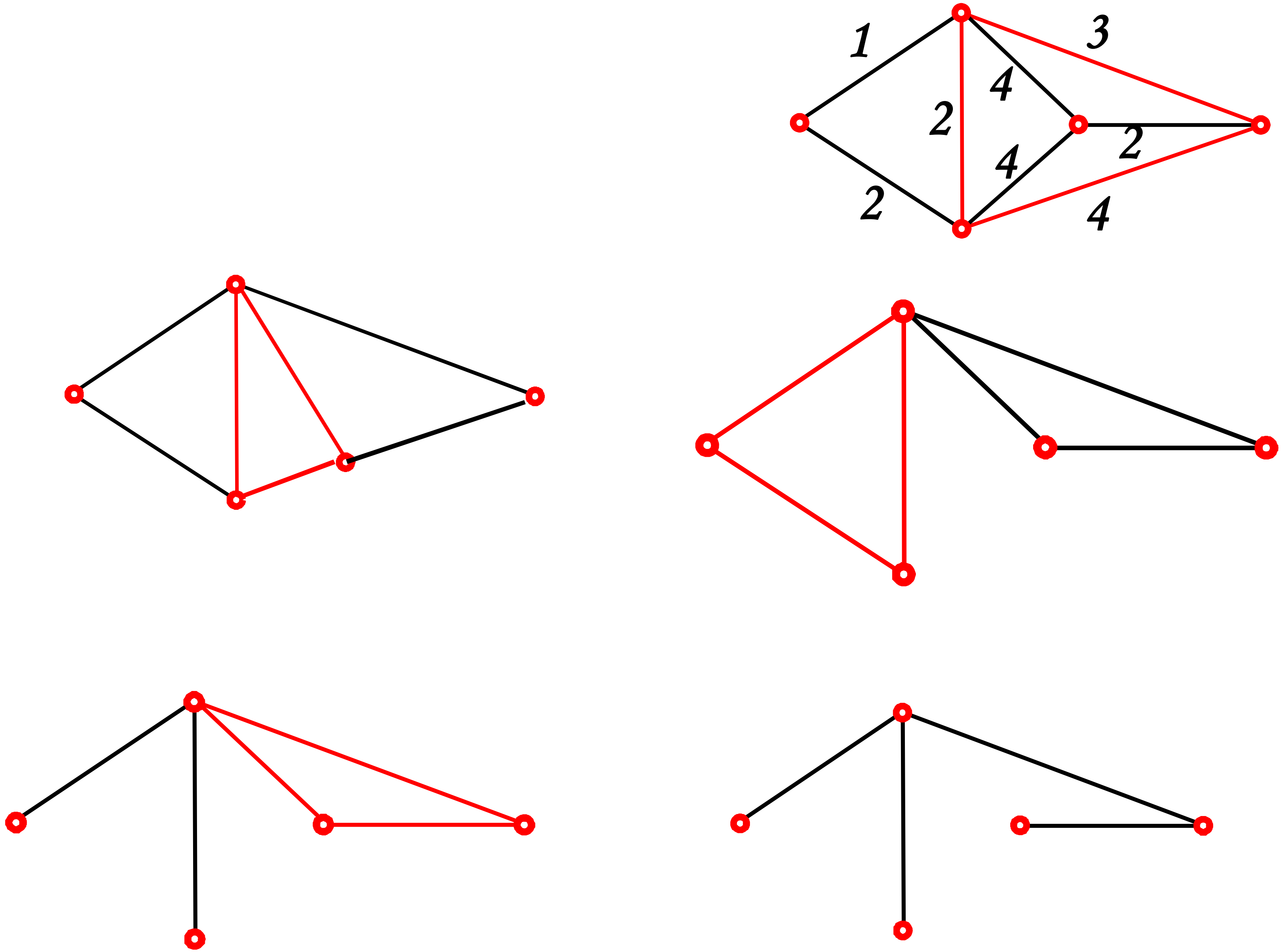


- 定理2 设V'是赋权图$G=\langle V,E\rangle$的顶点真子集，e是两个端点分别在V'和$V-V'$中的权最小的边，则G中一定存在包含e的最小生成树。
- 证明：设T_0是G的一棵最小生成树，若$e\notin E(T_0)$
- 则T_0+e包含唯一的圈C，且C中包含e及$e'=(u,v)$，使得 $u\in V',v\in V-V'$
- 由已知条件， $w(e)\leq w(e')$
- 故$(T_0+e)-e'$是G的一棵最小生成树。

- 定理3 Prim算法的结果得到赋权连通图的一棵最小生成树。
- 证明：首先证明它是一棵生成树。(归纳法)
- 初始，$S_0=\{v\}$，它是由S_0导出的树
- 设$|S_k|=k+1$，T_k是它导出的树，下一次迭代时，$S_{k+1}=S_k\cup\{u\}$，T_k中加入一条与u相连的边得到T_{k+1}
- 因此，T_{k+1}连通，且有$k+1$条边，故为树。
- 由定理2，Prim算法的结果得到赋权连通图的一棵最小生成树

破圈法

- ----Prim算法的对偶方法. 最适合于在图上作业. 当图比较大时, 还可以几个人同时在各个局部作业.
- Step0. 令G_0=G. k=0.
- Step1. 若G_k不含圈, 转Step2. 若G_k中含有圈C. 设$e_k \in E(C)$, 且 $\mathbf{w}(\mathbf{e}_k) = \max\limits_{\mathbf{e} \in \mathbf{w}(\mathbf{C})} \mathbf{w}(\mathbf{e})$
- 令$G_{k+1} = G_k - e_k$, 若$k=k+1$, 返回Step1.
- Step2. 结束. G_k为G的最小树.


1
3
4
2
2
4
2
4