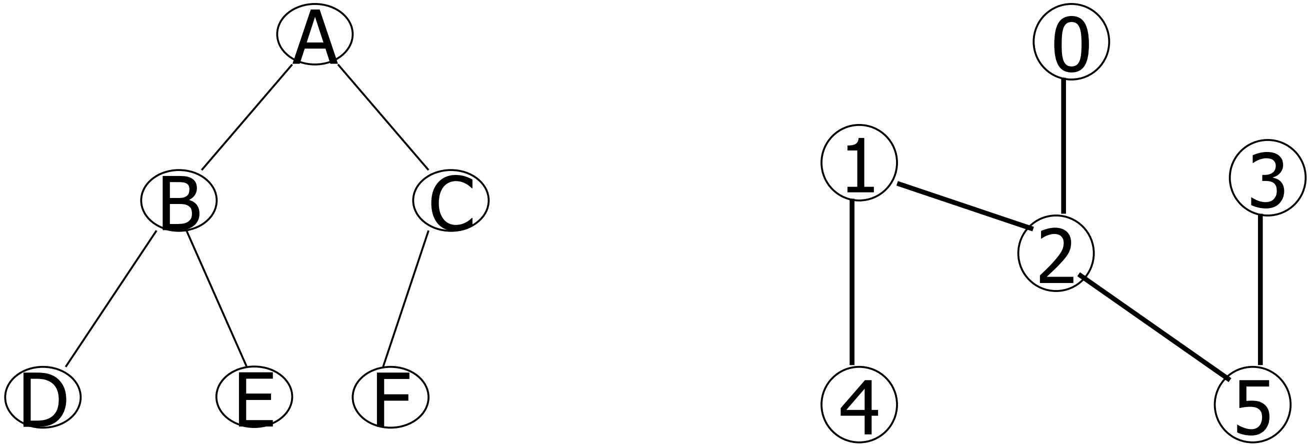
第七章：树和割集

7.1 树及其性质

7.2 生成树

7.3 割点、桥和割集



有根树　　　　自由树（无根树）

第七章：树和割集

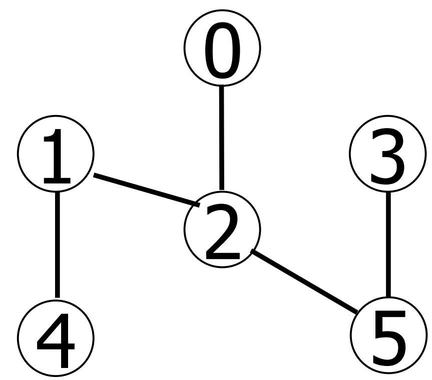
树（无圈的连通图）、最小生成树、割点和割集。

内容

树和割集

建模

一般作为极小的连通图、图的骨架来研究和应用。



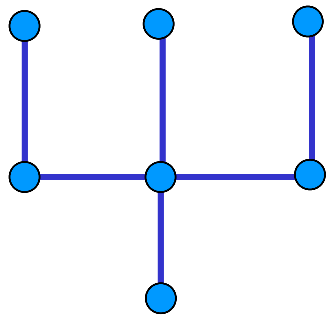
树

7.1 树及其性质

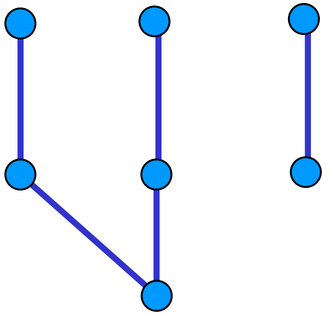
本节主要内容

1、树和森林的定义
2、树的性质
3、极小连通图的定义
4、顶点的偏心率
5、图的半径和中心点

1、树和森林的定义



树



森林

平凡树
（平凡图）

定义7.1.1　连通且无圈的无向图称为无向树，简称树。

一个没有圈的不连通的无向图称为无向森林，简称森林；

仅有一个顶点的树称为平凡树。

2、树的性质

定理7.1.1 设G=(V, E)是一个(p, q)图,则下列各命题等价:

(1) G是树

(2) G的任意两个不同顶点间有唯一的一条路联结;

(3) G是连通的且$p=q+1$;

(4) G中无圈且$p=q+1$;

(5) G中无圈且G中任意两个不邻接的顶点间加一条边,则得到一个有唯一的一个圈的图;

(6) G是连通的,并且若$p\geq 3$,则G不是K_p,G中任意两个不邻接的顶点间加一条边,则得到一个有唯一的一个圈的图。

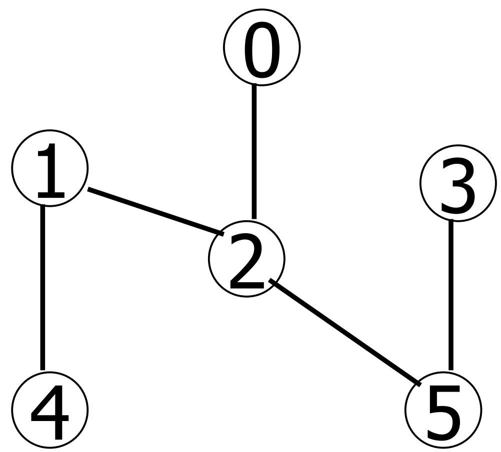
2、树的性质

定理7.1.1 设G=(V, E)是一个(p, q)图,则下列各命题等价:

(1) G是树

(2) G的任意两个不同顶点间有唯一的一条路联结。

证明略。



自由树（无根树）

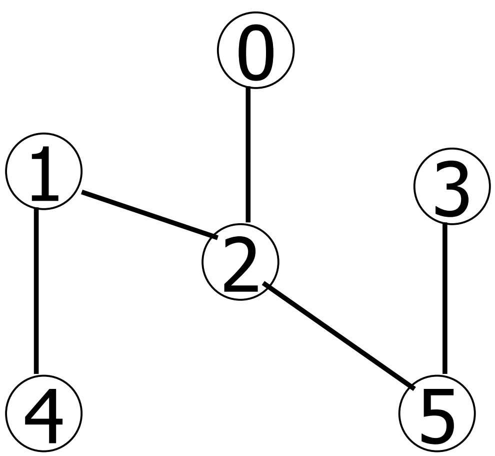
2、树的性质

定理7.1.1 设G=(V, E)是一个(p, q)图,则下列各命题等价:

(1) G是树

(5) G中无圈且G中任意两个不邻接的顶点间加一条边,则得到一个有唯一的一个圈的图;

证明略。



自由树（无根树）

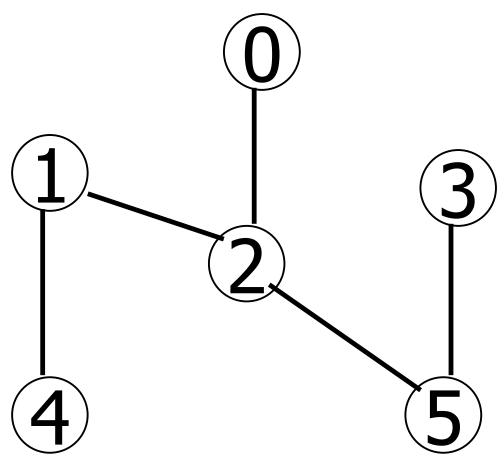
2、树的性质

定理7.1.1 设G=(V, E)是一个(p, q)图,则下列各命题等价:

(1) G是树

(6) G是连通的,并且若p≥3,则G不是K_p,G中任意两个不邻接的顶点间加一条边,则得到一个有唯一的一个圈的图。

证明略。



自由树（无根树）

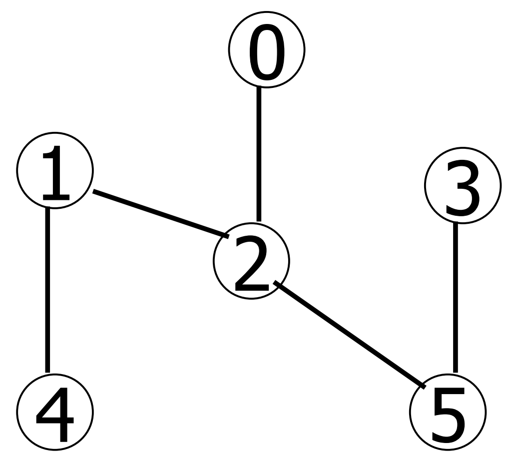
2、树的性质

定理7.1.1 设G=(V, E)是一个(p, q)图,则下列各命题等价:

(1) G是树

(3) G是连通的且p=q+1

(4) G中无圈且p=q+1



自由树（无根树）

2、树的性质

(1) G是树

⇓

(3) G是连通的且$p=q+1$;

证明：因为G是树，根据定义，G是连通的。

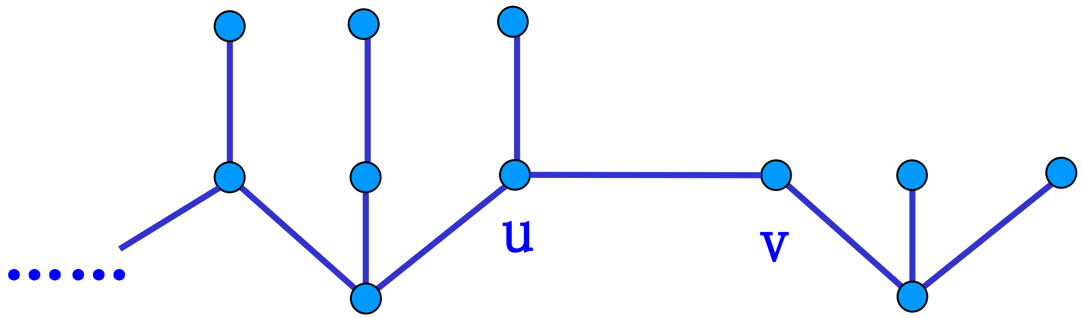
只需证明$p=q+1$

用归纳法：

当$p=1$或2时显然成立。

假设当$p\leq k$时成立，证明$p=k+1$时成立

2、树的性质

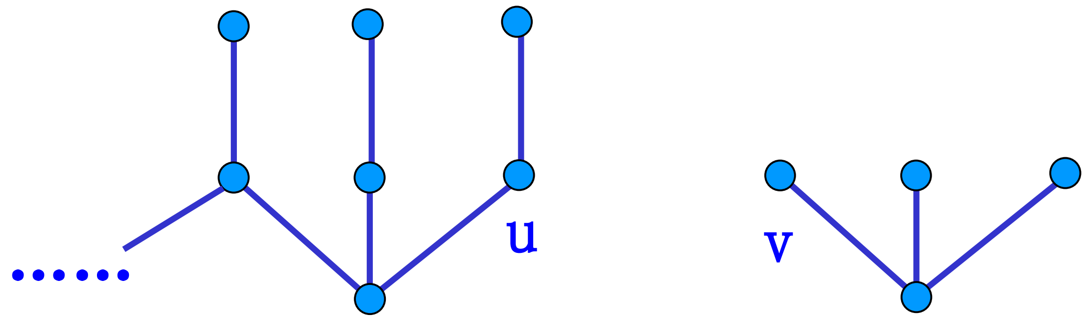


树G

证明：从树G中任选一条边uv, 去掉uv，设G′=G-uv, 需要证明G′正好是两棵树。

需要证明： G′不连通， G′有两个支，G′无圈。

2、树的性质



G′

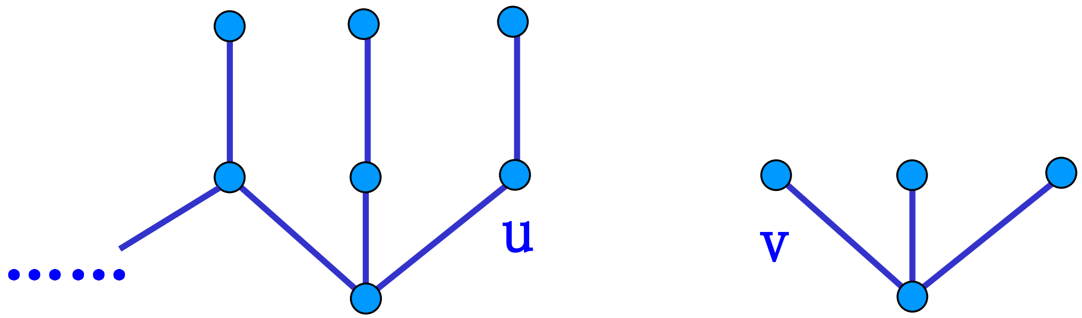
（1）如果G′是连通的，说明在G中u和v间还有别的路，加上uv形成圈，这与G是树矛盾，因此G′不连通。

（2）如果G′的支多于2个，那么加上边uv，仍然不连通，这与G是连通的矛盾，因此G′的支数是两个。

（3）因为G中无圈，所以G′中无圈。

因此G′正好由两棵树构成。

2、树的性质



G′

（4）设G′的其中一棵树为G_1，顶点数为p_1，另一棵树为G_2，顶点数为$p-p_1$。

（5）由归纳假设，G_1的边数为p_1-1, G_2边数为$p-p_1-1$

（6）原图G的边数q为篇$p_1-1+p-p_1-1+1=p-1$

因此：$p=q+1$.

2、树的性质

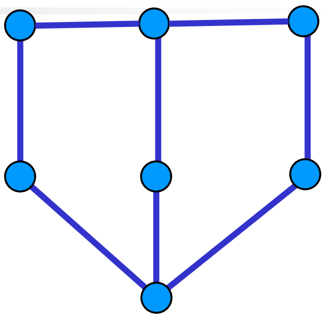
定理7.1.1 设G=(V, E)是一个(p, q)图,则下列各命题等价:

(1) G是树

(3) G是连通的且p=q+1;

(4) G中无圈且p=q+1;

2、树的性质



(3) G是连通的且p=q+1;

⇓

(4) G中无圈且p=q+1;

证明：只需证明G中无圈即可：

如果G中有圈，去掉圈上的一条边x_1，$G-x_1$还是连通的；

如果$G-x_1$中还有圈，去掉圈上的一条边x_2，$G-x_1-x_2$还是连通的；

设从G中去掉m条边后无圈的连通图为G′， G′中边数为 p-1

因此G中边数为q+m=p-1+m

这与G中边数为p-1矛盾。　　因此G中无圈

2、树的性质

定理7.1.1 设G=(V, E)是一个(p, q)图,则下列各命题等价:

(1) G是树

(3) G是连通的且p=q+1;

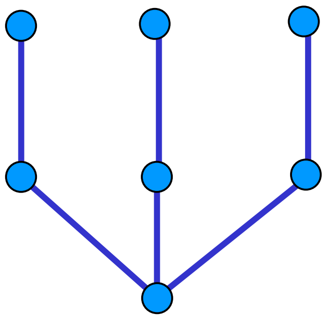
(4) G中无圈且p=q+1;

2、树的性质

(4) G中无圈且p=q+1;

⇓

(1) G是树


图G

证明：只需证明G是联通的即可

如果G不连通，任选两个分属于不同支的两个顶点，在这两个顶点间加一条边，则减少一个支；

如果所得到的图扔不连通，再任选两个分属于不同支的两个顶点，在这两个顶点间加一条边，则又减少一个支；

设在G中增加了m条边后的图G′连通了，则 G′的边数为p-1。

因此G中边数为p-1-m

矛盾， 因此G是联通的。

2、树的性质

定理7.1.1 设$G=(V,E)$是一个(p,q)图,则下列命题等价:

(1) G是树

(2) G的任意两个不同顶点间有唯一的一条路联结;

(3) G是连通的且$p=q+1$;

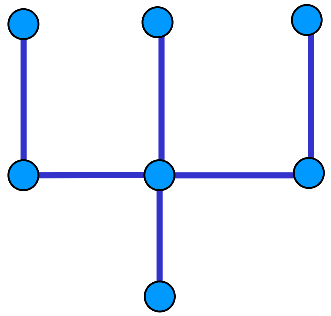
(4) G中无圈且$p=q+1$;

(5) G中无圈且G中任意两个不邻接的顶点间加一条边,则得到一个有唯一的一个圈的图;

(6) G是连通的,并且若$p\geq 3$,则G不是K_p,G中任意两个不邻接的顶点间加一条边,则得到一个有唯一的一个圈的图。

2、树的性质

推论7.1.1　任一非平凡树中至少有两个度为1的顶点。



树

证明：

设$v_1v_2\cdots v_{m-1}v_m$是树G中的一条最长路。

则v_1和v_m的度数都为1。

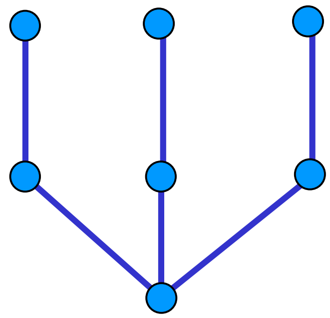
如若不然，以v_m为例：

若v_m还与除v_{m-1}以外的顶点u关联

若u不在最长路$v_1v_2\cdots v_{m-1}v_m$上，
则$v_1v_2\cdots v_{m-1}v_mu$是一条更长的路，矛盾。
若u在最长路$v_1v_2\cdots v_{m-1}v_m$上，设$u=v_i$，
则$v_iv_{i+1}\cdots v_{m-1}v_mv_i$是一个圈，矛盾。

所以v_1和v_m的度数都为1。

3、最小连通图的定义

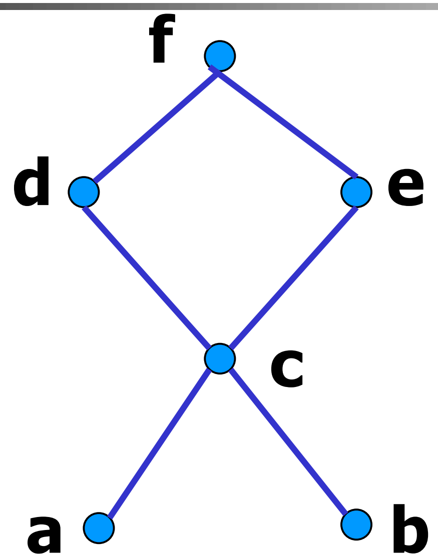


树-极小连通图

定义7.1.2　连通图G称为是极小连通图，如果去掉G的任意一条边后得到的都是不连通图。

推论7.1.2　图G是树当且仅当G是极小连通图。

4、顶点的偏心率



图G

计算图G中各点的偏心率

a,b,f的偏心率是3

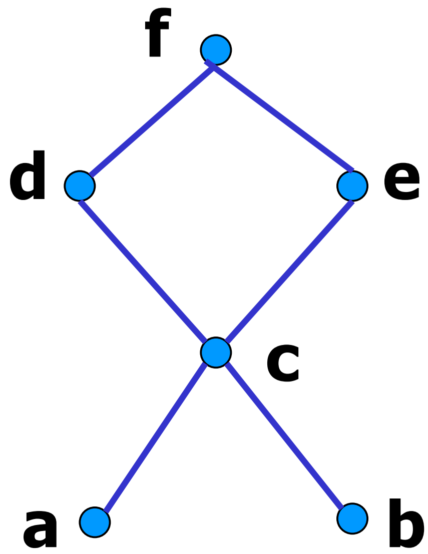
d,e,c的偏心率是2

定义7.1.3 设G=(V, E)是连通图, v∈V, 数

$$e(v) = \max_{u \in V}\{d(v,u)\}$$

称为v在G中的偏心率,

5、连通图的半径和中心点



图G

计算图G中各点的偏心率

a,b,f的偏心率是3

d,e,c的偏心率是2

计算图G的半径、中心点、中心

图G的半径是2

d,e,c是图G的中心点

{d,e,c}是图G的中心

$$r(G) = \min_{v \in V} \{e(v)\}$$

称为G的半径,满足r(G)=e(v)的顶点v称为G的中心点,G的所有中心点组成的集合称为G的中心,G的中心记为C(G)

5、连通图的半径和中心点

定理7.1.2　每棵树的中心或含有一个顶点，或含有两个邻接的顶点

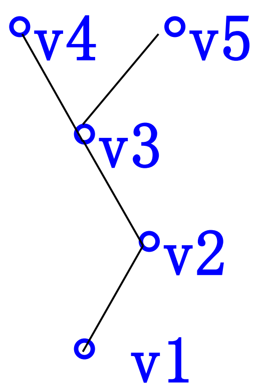


图1

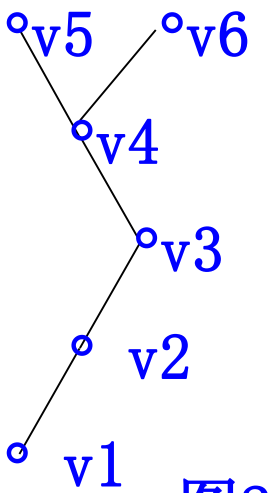


图2

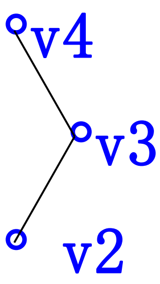


图3

离中心最远的点满足什么性质?

都是一度顶点;

如果把1度顶点都去掉，会不会引起中心点的变化?

不会引起中心点的变化。

5、连通图的半径和中心点

定理7.1.2　每棵树的中心或含有一个顶点，或含有两个邻接的顶点。

[证]

(1)一个顶点的树有一个中心，两个顶点的树有两个中心。

(2)每次把所有的一度顶点全去掉，并不引起中心的变化。

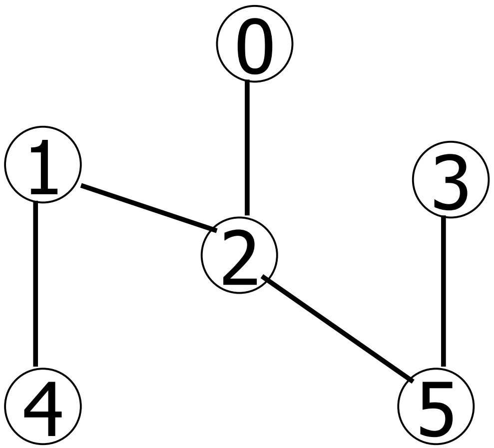
(3)每次把所有的一度顶点全去掉，经有限步必可得到一个只有一个顶点的树，或只有两个顶点的树。

5、连通图的半径和中心点

例7.1.1　任何一个非平凡的树都可用两种颜色给其顶点染色，使得每条边的两个端点不同色。

由第六章第4节中偶图的充分必要条件是图中所有圈都是偶数长可得树是偶图。

因此树可用两种颜色染色，并且每条边的两个端点不同色。



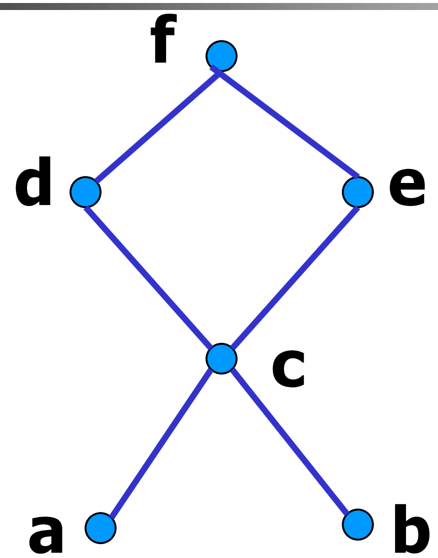
自由树（无根树）

7.2 生成树

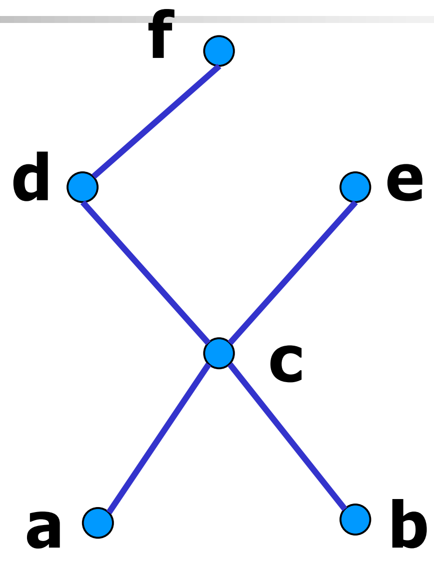
本节主要内容

1、生成树的定义
2、生成树的性质
3、生成树间的距离
4、生成树间的变换
5、最小生成树的定义
6、最小生成树的Kruskal算法
7、最小生成树的prim算法

1、生成树的定义



图G



图G的一棵生成树

定义7.2.1 设G=(V, E)是一个图, G的一个生成子图T=(V, F)如果是树,则称T是G的生成树。

(1)、若图G有生成树,则G是连通的,所以不连通图没有生成树,

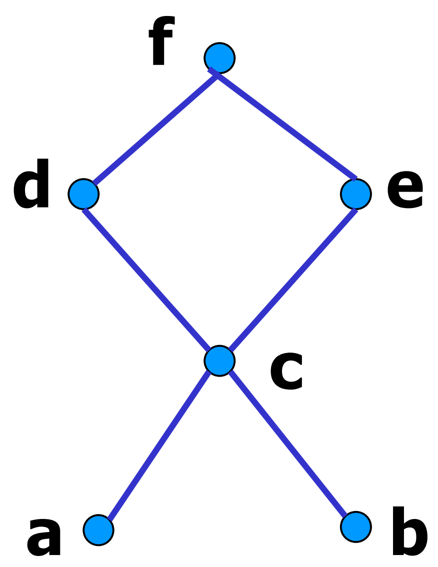
(2)、连通图都有生成树吗?

2、生成树的性质

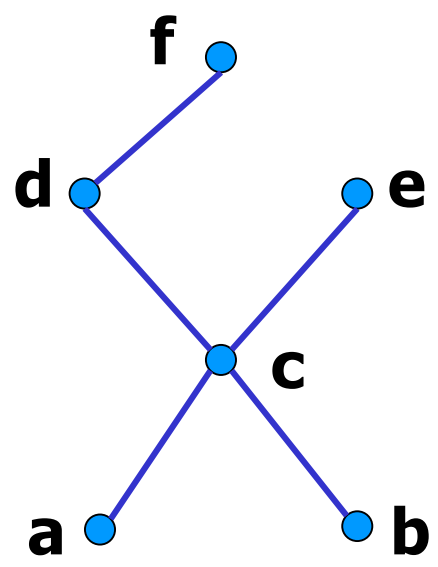
定理7.2.1 图G有生成树的充分必要条件是G为一个连通图。

证明：必要性：

若G有生成树T，由T是连通的知G是连通的。



图G

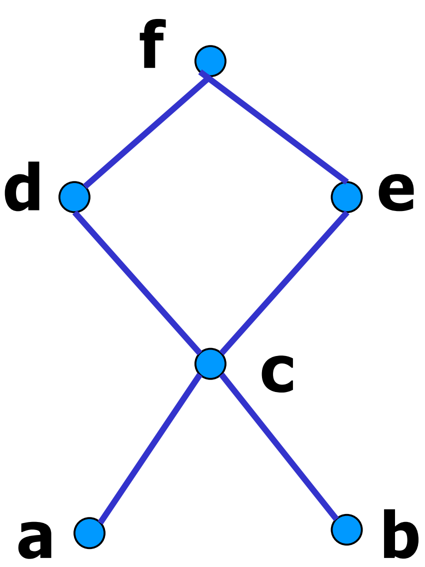


图G的一棵生成树

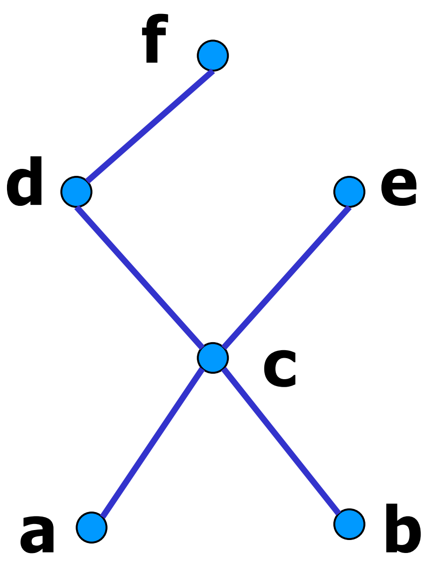
2、生成树的性质

定理7.2.1 图G有生成树的充分必要条件是G为一个连通图。

证明：充分性：



图G



图G的一棵生成树

2、生成树的性质

定理7.2.1 图G有生成树的充分必要条件是G为一个连通图。

证明：充分性：

G是连通的，若G没有圈，则G是树；

若G有圈，去掉圈上的一条边x_1，$G-x_1$还是连通的；

若$G-x_1$还有圈，去掉圈上的一条边x_2，$G-x_1-x_2$还是连通的；

………

经有限步得到一个无圈的连通图T，T就是G的生成树。

2、生成树的性质

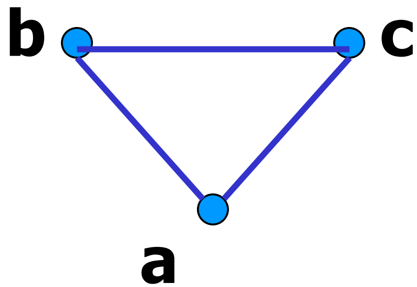
推论7.2.1 设G是一个(p, q)连通图,则 $q \geq p-1$。

定义7.2.2 设G是一个图，若G的生成子图F是一个森林，则F称为G的一个生成森林。

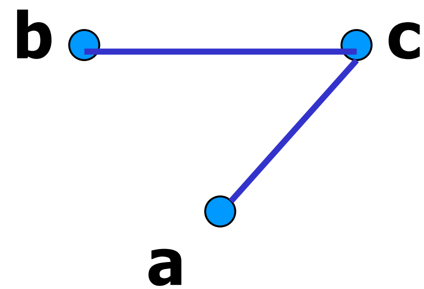
任意图都有生成森林。

2、生成树的性质

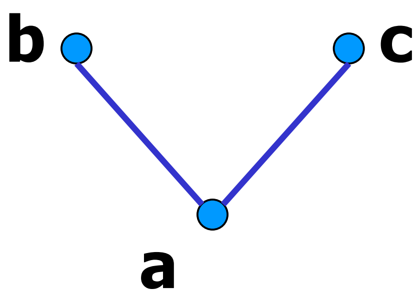
定理7.2.2 具有p个顶点的完全图K_p有p^{p-2}棵生成树, $p\geq 2$.



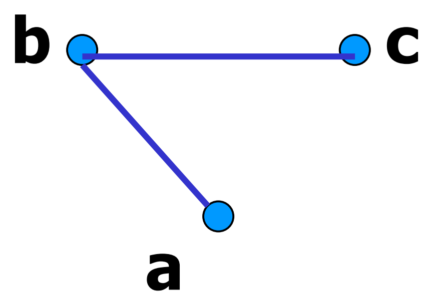
图G



图G的一棵生成树



图G的一棵生成树



图G的一棵生成树

2、生成树的性质

定理7.2.2 具有p个顶点的完全图K_p有p^{p-2}棵生成树, $p\geq 2$.

对p^{p-2}有什么感想？

每一位有p种选择，长度为$p-2$的序列的个数

对2^n有什么感想？

每一位有两种选择，长度为n的序列的个数

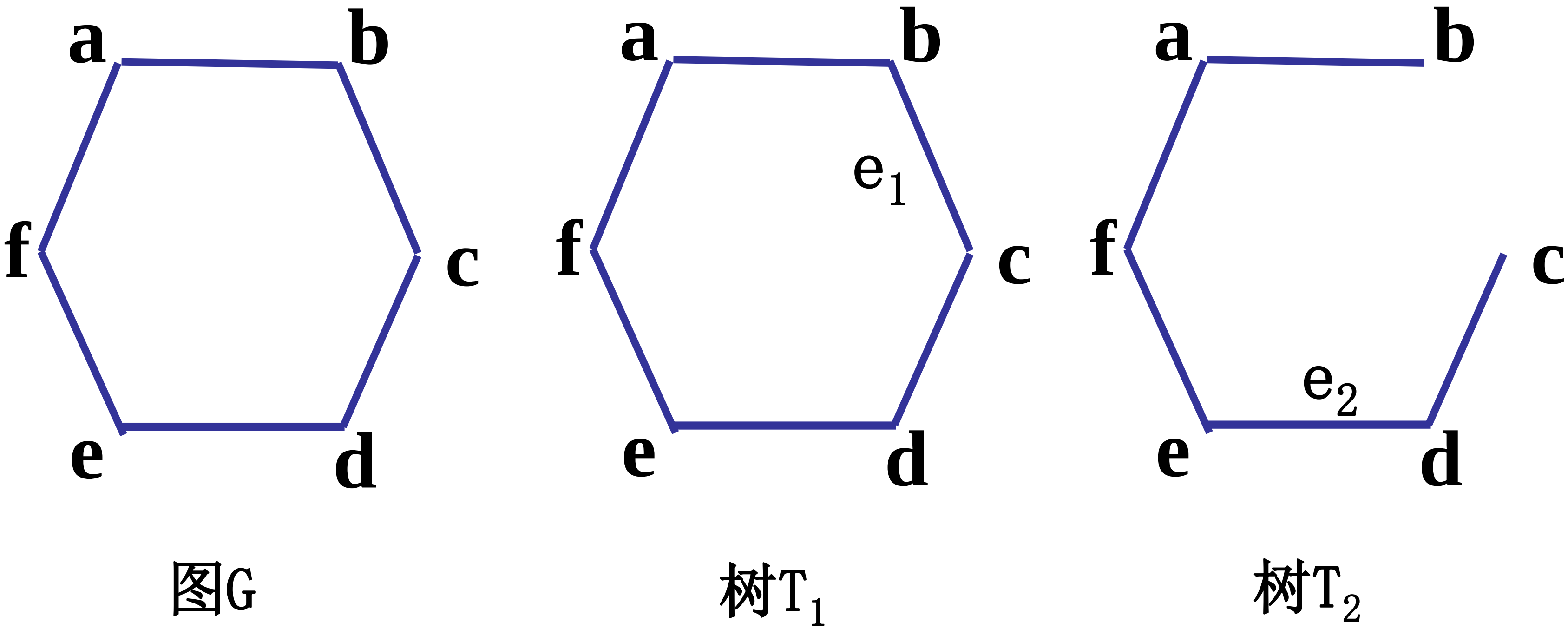
对m^n有什么感想？

每一位有m种选择，长度为n的序列的个数

建立K_p的生成树与每位有p种选择，长度为$p-2$的序列之间的一一对应。

2、生成树的性质

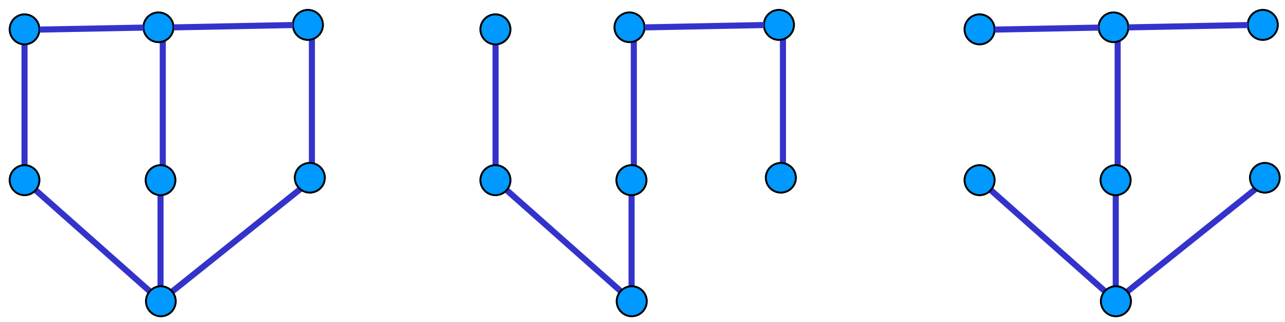
定理7.2.3 设$G=(V, E)$是连通图,$T_1=(V, E_1)$和$T_2=(V, E_2)$是G的两个不同的生成树,如果$e_1 \in E_1 \setminus E_2$,则$\exists e_2 \in E_2 \setminus E_1$,使得$(T_1-e_1)+e_2$为G的一棵生成树。



证明略

3、生成树间的距离

定义7.2.3　设T_1, T_2是G的生成树，是T_1的边但不是T_2的边的条数k称为T_1与T_2的距离，记为$d(T_1, T_2)=k$。



图G　　树T_1　　树T_2

计算T_1和T_2的距离

T_1和T_2的距离是2

3、生成树间的距离

定义7.2.3　设T_1, T_2是G的生成树，是T_1的边但不是T_2的边的条数k称为T_1与T_2的距离，记为$d(T_1, T_2)=k$。

定义距离要满足三个条件，分别是：

（1） 非负性　（2） 对称性　（3）满足三角不等式

由定义可知$d(T_1, T_2) \geqslant 0$,

并且$d(T_2, T_1)=d(T_1, T_2)$

因此满足（1） 非负性 和　（2） 对称性。

3、生成树间的距离

(3) 满足三角不等式

$$d(T_1, T_2) \leqslant d(T_1, T_3) + d(T_3, T_2)$$

是T_1的边但不是T_2的边包括在如下情况中，

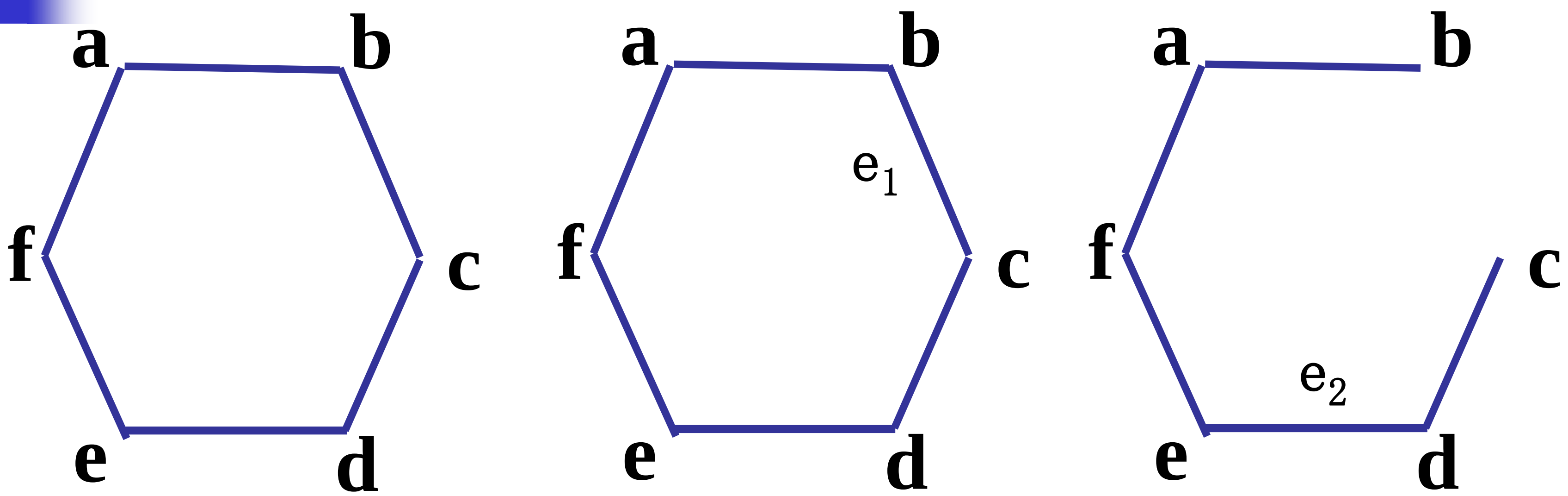
=

是T_1的边不是T_2的边也不是T_3的边，

+

是T_1的边不是T_2的边是T_3的边，

4、生成树间的变换



图G　　树T_1　　树T_2

两个生成树之间的基本树变换

若$d(T_1, T_2)=1$,

则T_1中有一条边e_1不在T_2中，T_2中也有一条边不在T_1中，

$T_2=(T_1-e_1)+e_2$

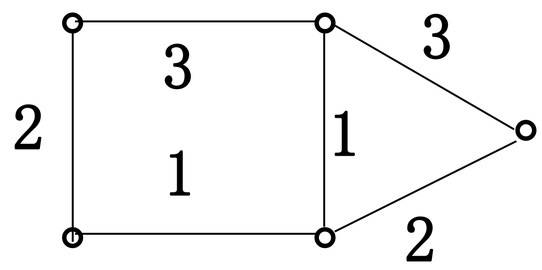
它称为从T_1到T_2的一个基本树变换。

4、生成树间的变换

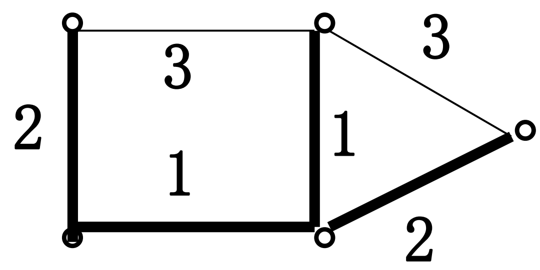
定理7.2.4　设T_0和T是G的两距离为k的生成树，则从T_0开始经k次基本树变换便可得到T。

证明略

5、最小生成树



图G



T

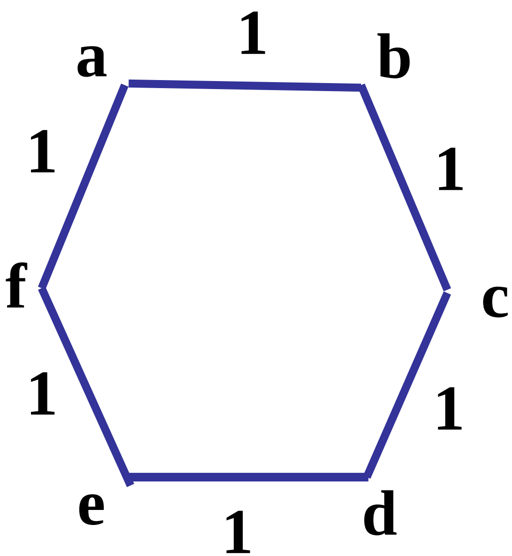
给定边带权连通图G,G中边的权是一个非负实数,生成树中各边的权之和称为该生成树的权;

图G的生成树T的权是6

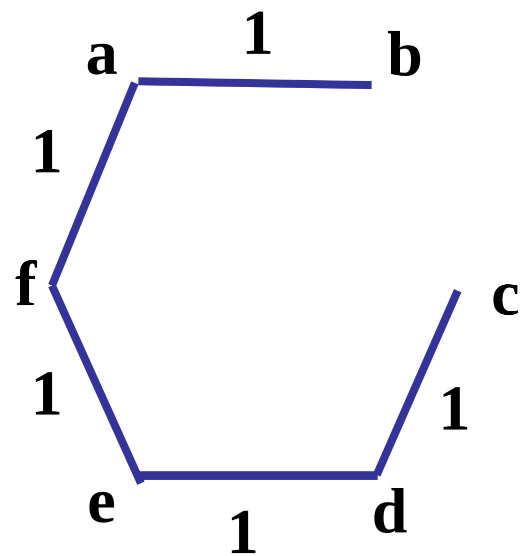
G的生成树中权最小的那个生成树就是最小生成树。

5、最小生成树

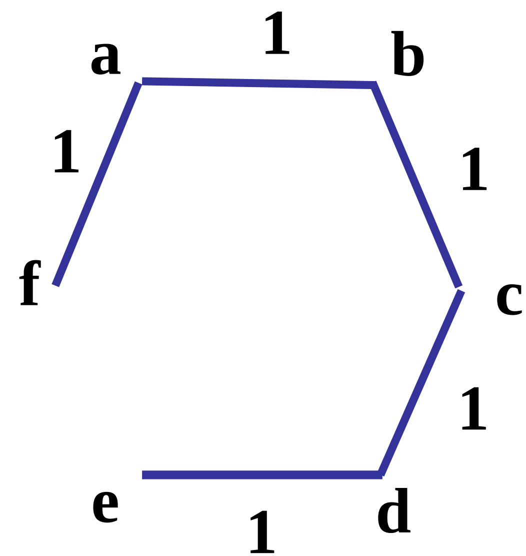
最小生成树不一定唯一



图G



树T_1

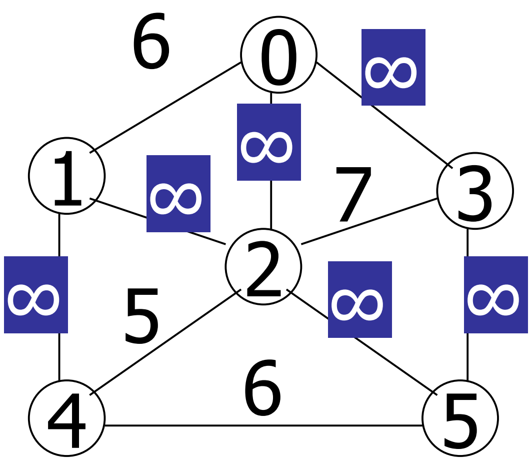


树T_2

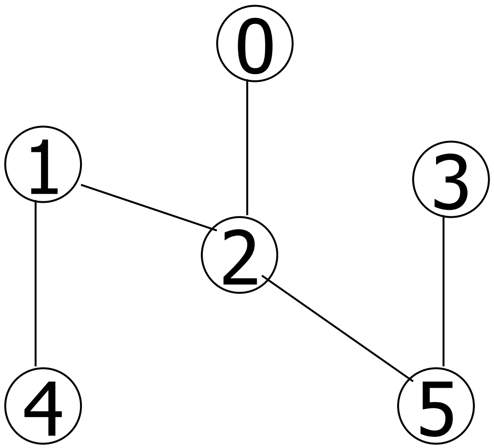
6、最小生成树Kruskal算法

定理7.2.5 设G=(V, E, w)是一个连通的边带权图，边上的权函数w非负，

$\{(V_1, E_1), (V_2, E_2), ..., (V_k, E_k)\}$是G的生成森林，k>1，$F=\bigcup_{i=1}^{k} E_i$ ，如果e=uv是E\F中权值最小的边且$u \in V_i$，$v \notin V_i$中，则存在G的一个包含$F\cup\{e\}$的生成树T，使得T的权不大于任一包含F的生成树的权。



图G



图G的生成森林（生成树）

证明略。

6、最小生成树Kruskal算法

输入:连通图G=(V,E),其中V={1,2,...,n},

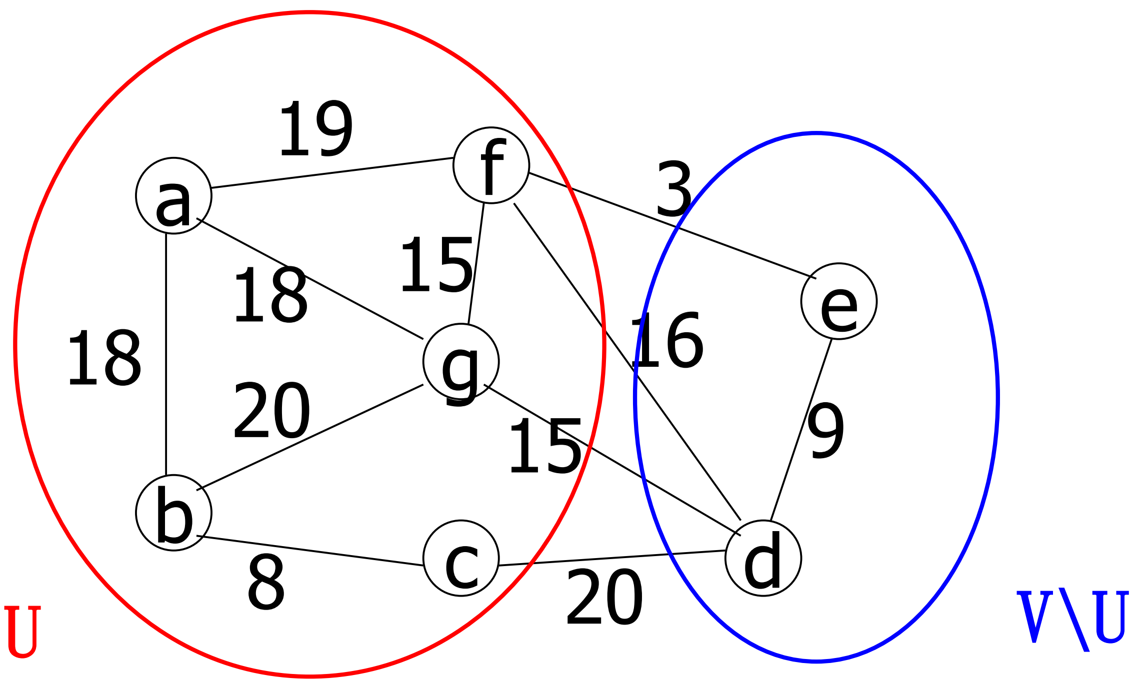
输出:一颗最小生成树:

算法:

```
void Kruskal(V,T)
{T=∅;
 ncomp=n;//连通分支
 while(ncomp>1)
{从E中取出删除权最小的边(v,u);
 if(v和u属于T中不同的连通分支)
 {T=T∪{(v,u)}; ncomp--;}
 }
}
```

7、最小生成树prim算法

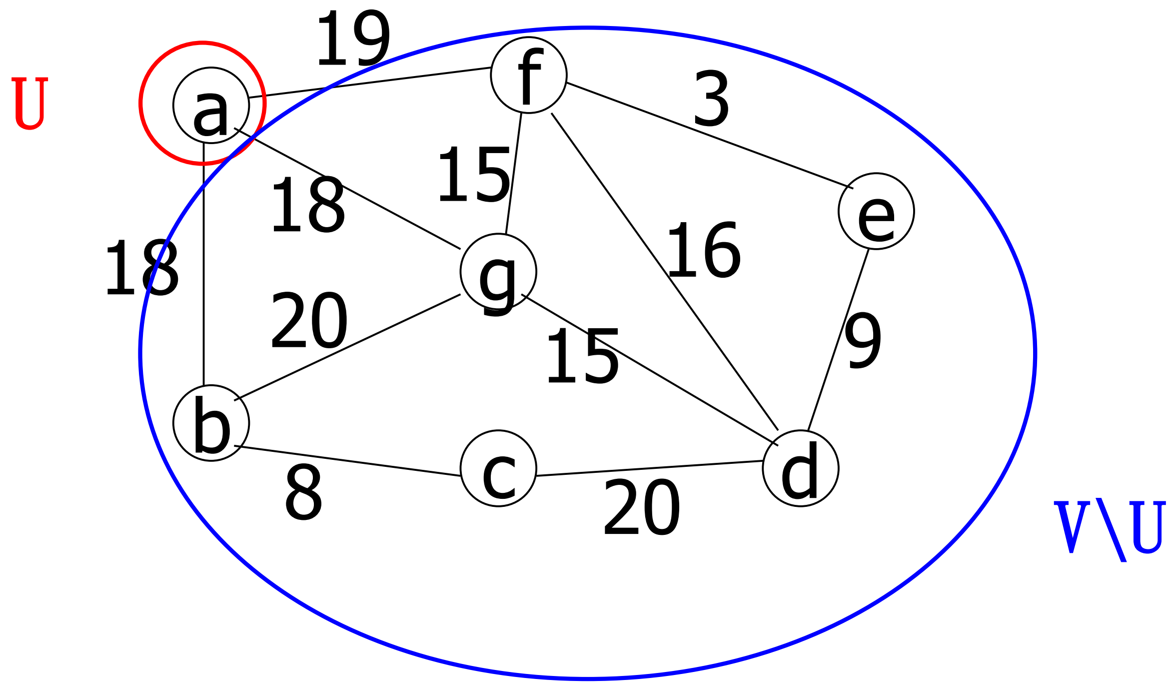
定理7.2.6 设G=(V, E, w)是一个边带权连通图, U是V的一个真子集, 如果{u, v}是u∈U, v∈V\U的G的一条边, 并且是所有的这样的边中, {u, v}的权w(u, v)最小, 则G中一定存在一个最小生成树,它以{u, v}为其中一条边。



证明略。

7、最小生成树prim算法

最小生成树的prim算法事例演示



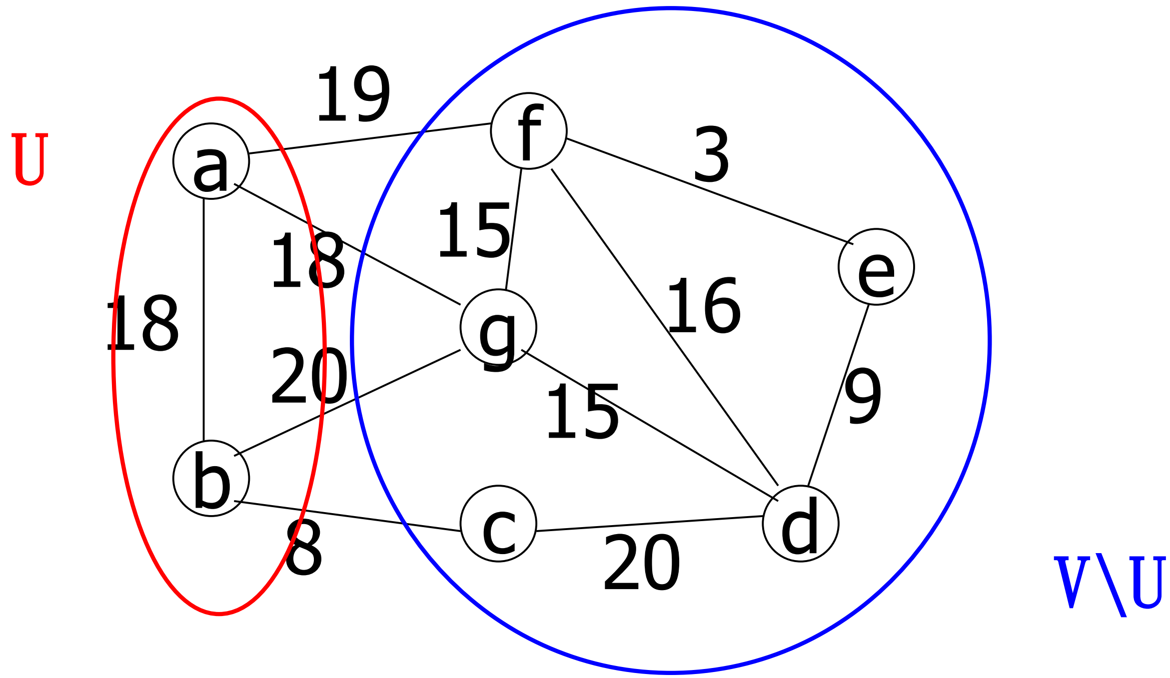
(1) U={a}, V\U={b, c, d, e, f, g}, T={}。

连接U和V\U之间最短的边是ab或ag任选一个例如ab

T={ab}, U={a, b}, V\U={c, d, e, f, g}

7、最小生成树prim算法

最小生成树的prim算法事例演示



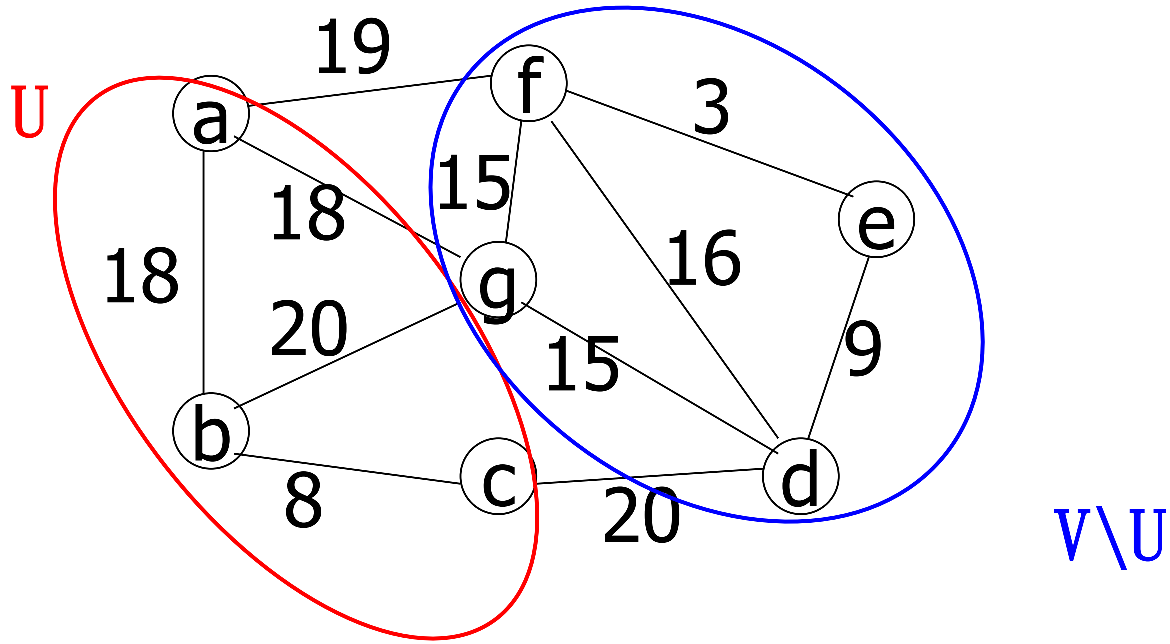
(1) T={ab}, U={a, b}, V\U={c, d, e, f, g}

连接U和V\U之间最短的边是bc

T={ab, bc}, U={a, b, c}, V\U={d, e, f, g}

7、最小生成树prim算法

最小生成树的prim算法事例演示



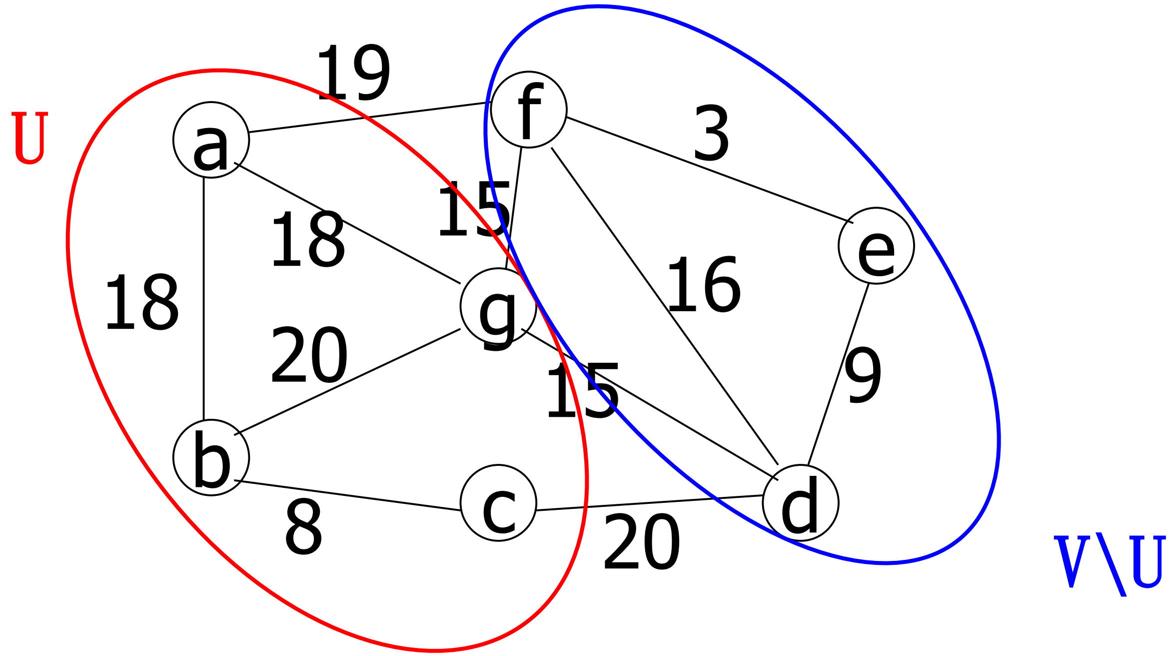
(1) T={ab, bc}, U={a, b, c}, V\U={d, e, f, g}

连接U和V\U之间最短的边是ag

T={ab, bc, ag}, U={a, b, c, g}, V\U={d, e, f}

7、最小生成树prim算法

最小生成树的prim算法事例演示



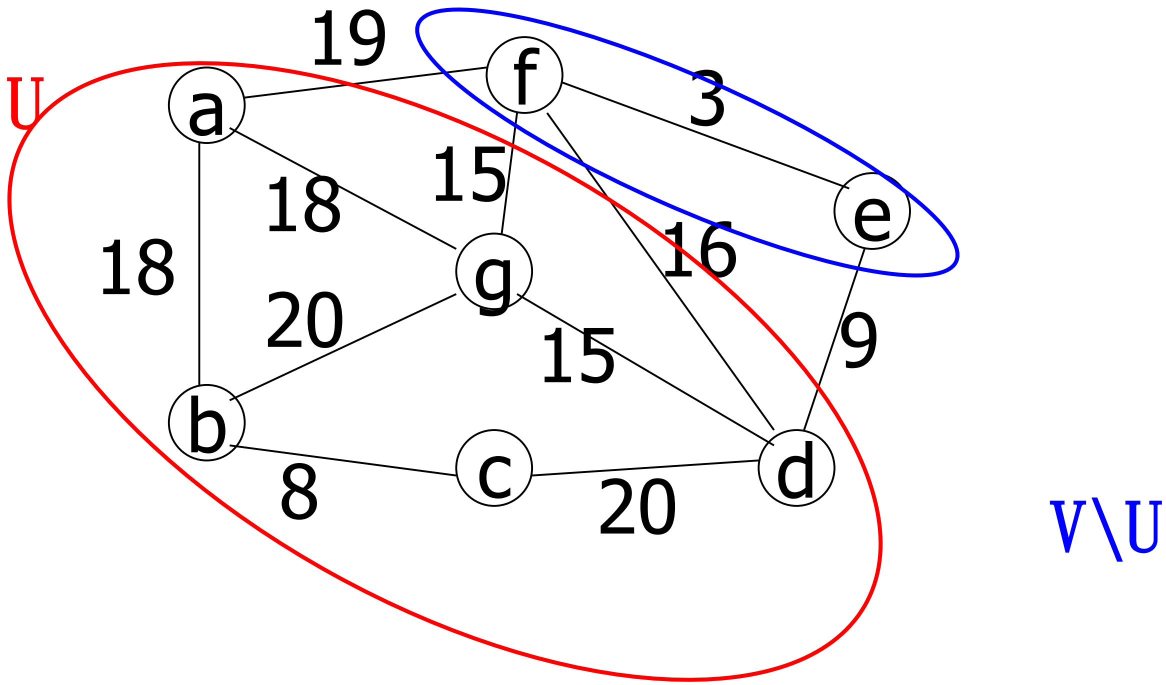
(1) T={ab, bc, ag}, U={a, b, c, g}, V\U={d, e, f}

连接U和V\U之间最短的边是gd或gf任选一个例如gd

T={ab, bc, ag, gd}, U={a, b, c, g, d}, V\U={e, f}

7、最小生成树prim算法

最小生成树的prim算法事例演示



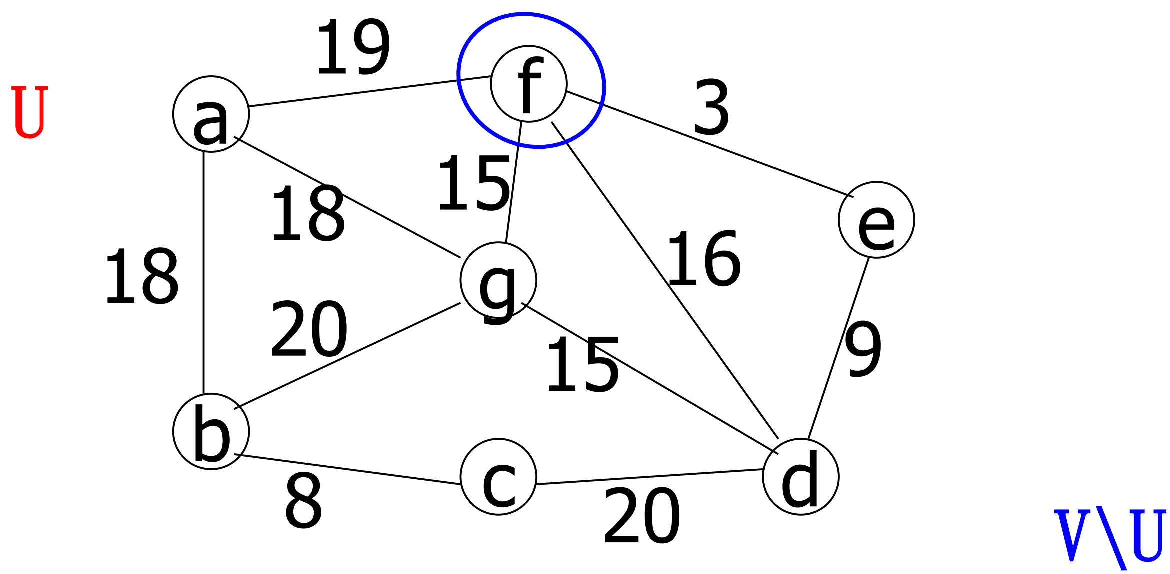
(1) T={ab, bc, ag, gd}, U={a, b, c, g, d}, V\U={e, f}

连接U和V\U之间最短的边是de

T={ab, bc, ag, gd, de}, U={a, b, c, g, d, e}, V\U={e}

7、最小生成树prim算法

最小生成树的prim算法事例演示



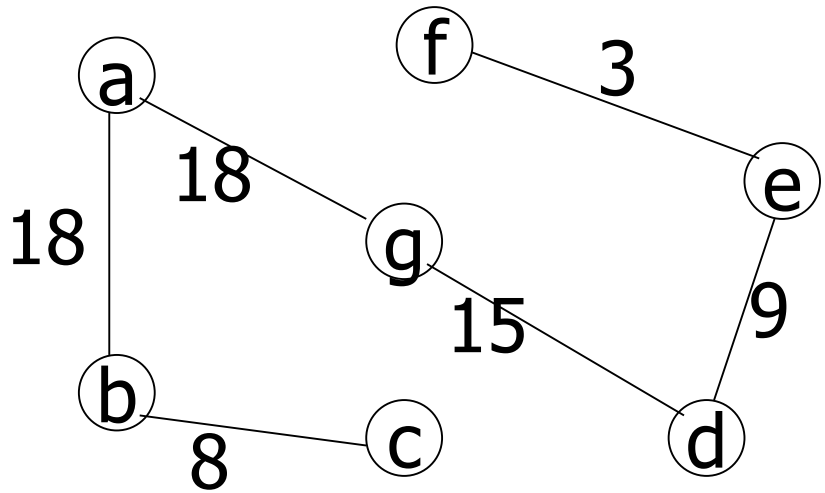
（1） T={ab, bc, ag, gd, de}, U={a, b, c, g, d, e}, V\U={e}

连接U和V\U之间最短的边是ef

T={ab, bc, ag, gd, de, ef}, U={a, b, c, g, d, e, f}, V\U={}

7、最小生成树prim算法

最小生成树的prim算法事例演示



T={ab, bc, ag, gd, de, ef}

7、最小生成树prim算法

输入:连通图G=(V,E), 其中V={V1,V2,…,Vn},

输出:一颗最小生成树T:

```
void Prim(V,T) {
    T=∅;
    U={V1};
    i=1;
    while(i<n) {
       求边{u,v}; // {u, v}是满足u∈U,
                          v∈V\U中最短的边;
       T=T∪{{u, v}};
       U= U∪{v};
       i++;
    }
}
```

**