

第三章 树

3.1 树的有关定义

- 给定一个图 $G=(V,E)$, 如果它不含任何回路, 我们就叫它是林, 如果 G 又是连通的, 即这个林只有一个连通支, 就称它是树.
 - 定义**3.1.1**
一个不含任何回路的连通图称为树, 用 T 表示. T 中的边称为树枝, 度为 **1** 的节点称为树叶.
-

有关度的若干术语

- 孤立点:度为**0**的顶点
 - 悬点:度为**1**的顶点
 - 悬边:与悬点关联的边
 - 奇点:度为奇数的顶点
 - 偶点:度为偶数的顶点
 - 正则图:各顶点度相同
 - 若度为 k ,称为 k -正则图.
 - 例如: K_n 是 $(n-1)$ -正则图
-

树的有关定义

- 树的每条边, 都不会属于任何回路. 这样的边叫割边.
 - 定义**3.1.2**
设 e 是 G 的一条边, 若 $G' = G - e$ 比 G 的连通支数增加, 则称 e 是 G 的一条割边.
 - 显然, 图 G 删去割边 $e = (u, v)$ 之后, 结点 u, v 分属于不同的分支.
-

树的有关定义

□ 定理3.1.1

$e=(u, v)$ 是割边,当且仅当 e 不属于 G 的任何回路.

证明:

必要性。设 $e=(u, v)$ 是割边,此时若 $e=(u, v)$ 属于 G 的某个回路,则 $G' = G - e$ 中仍存在 u 到 v 的道路,故结点 u 和 v 属于同一连通支, e 不是割边.矛盾.

充分性。设 e 不属于 G 的任何回路,此时若 e 不是割边,则 $G' = G - e$ 与 G 的连通支数一样.于是 u 和 v 仍属于同一连通支.故 G' 中存在道路 $P(u, v)$, $P(u, v) + e$ 就是 G 的一个回路.矛盾.

树的有关定义

□ 定理**3.1.2**

设**T**是结点数为 **$n \geq 2$** 的树, 则下列性质等价:

- 1.** **T**连通且无回路
 - 2.** **T**连通且每条都是割边
 - 3.** **T**连通且有 **$n-1$** 条边
 - 4.** **T**有 **$n-1$** 条边且无回路
 - 5.** **T**的任意两结点间有唯一道路
 - 6.** **T**无回路, 但在任两结点间加上一条边后恰有一个回路
-

T连通且无回路 \rightarrow T连通且每条都是割边 \rightarrow T连通且有 $n-1$ 条边

- **1 \rightarrow 2:** T无回路, 即T的任意边 e 都不属于回路, 由定理**3.1.1**, e 是割边。
 - **2 \rightarrow 3:** 对结点数 n 进行归纳。令 $n(T)$, $m(T)$ 分别表示树T的结点数和边数。当 $n=2$ 时命题成立。设 $n \leq k$ 时, $m(T) = n(T) - 1$ 成立。则 $n = k + 1$ 时, 由于任一边 e 都是割边, 故 $G' = G - e$ 有两个连通支 T_1, T_2 。由于 $n(T_i) \leq k$, $i = 1, 2$, 故 $m(T_i) = n(T_i) - 1$ 。所以 $m(T) = n(T) - 1$ 也成立。
-

3. T连通且有 $n-1$ 条边

4. T有 $n-1$ 条边且无回路

- **3→4:** 假定T有回路，设C是其中一条含有 $k(<n)$ 个结点的初级回路。因为T连通，所以 $V(T)-V(C)$ 中一定有结点u与C上某点v相邻，即存在边 $(u,v) \in E(T)$ ，依此类推，最终 $V(T)-V(C)$ 中的 $n-k$ 个结点需要 $n-k$ 条边才可能保持T连通，
 - 但 $|E(T)-E(C)| = (n-1)-k < n-k$. 矛盾.
-

4. T有n-1条边且无回路

5. T的任意两结点间有唯一道路

□ 4→5: 设 u, v 是T的任意两结点, 先证道路 $P(u, v)$ 的存在性, 即证明T是连通的。反证法。

如果T不是连通的, 则至少有两个连通分支 T_1, T_2 . 由已知T中无回路可知, T_1, T_2 也没有回路。根据1 → 2 → 3的证明, 再由 T_1 和 T_2 是连通的且无回路可得, $m(T_1) = n(T_1) - 1, m(T_2) = n(T_2) - 1$, 则有:

$$m(T) = m(T_1) + m(T_2) = (n(T_1) + n(T_2)) - 2 = n - 2 < n - 1$$

与已知 $m(T) = n - 1$ 矛盾。

再证唯一性。若存在两条不同的道路 $P(u, v), P'(u, v)$, 则其对称差 $P(u, v) \oplus P'(u, v)$ 至少含有一个回路。

注: $G_1 \oplus G_2 = (V, E)$, 其中 $V = V_1 \cup V_2, E = E_1 \oplus E_2$;

5. T 的任意两结点间有唯一道路

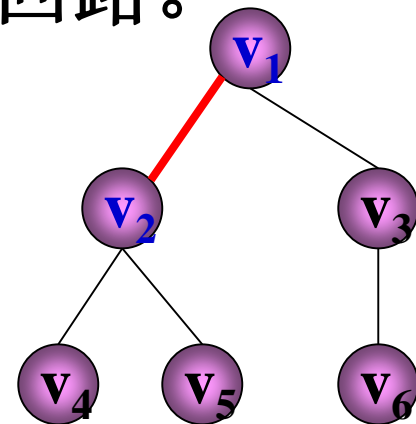
6. T 无回路,但在任两结点间加上一条边后恰有一个回路

1. T 连通且无回路

□ **5**→**6**: 显然成立

□ **6**→**1**: 只要证明 T 是连通的。反证法。

假设 T 不连通,设 T_1, T_2 为 T 中的两个连通分支。 v_1 为 T_1 中的一个顶点, v_2 为 T_2 中的一个顶点。在 T 中加边 (v_1, v_2) 不形成回路。矛盾。



总结

- 树是极小的连通图，减少一条边就不连通
 - 树是极大的不含回路的连通图，增加一条边就有回路
-

树的有关定义

□ 定理3.1.3

树**T**中一定存在树叶结点.

证明: 由于**T**是连通图, 所以任一结点 $v_i \in V(\mathbf{T})$, 都有 $d(v_i) \geq 1$. 若无树叶, 则 $d(v_i) \geq 2$. 这样

$$n - 1 = m = \frac{1}{2} \sum d(v_i) \geq n$$

矛盾.

树的有关定义

□ 定义3.1.3

如果 T 是图 G 的支撑子图, 而且又是一棵树, 则称 T 是 G 的一棵支撑树, 或称生成树, 又简称 G 的树

生成树

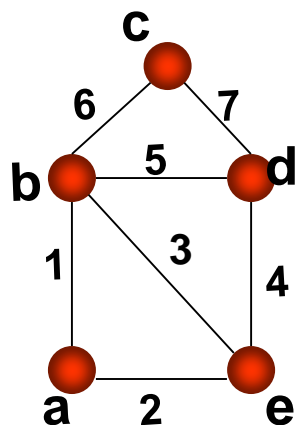
定理：任何无向连通图**G**都存在生成树。

证明：如果**G**无回路，则**G**本身就是它的生成树。

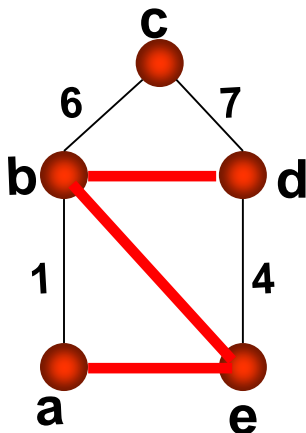
如**G**有回路，则在回路上任取一条边去掉仍是连通的，如**G**仍有回路，则继续在该回路上去掉一条边，直到图中无回路为止，此时，该图就是**G**的一棵生成树。

一个连通图的生成树可能不唯一。

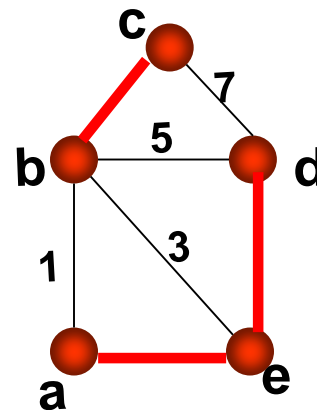
因为在取定一个回路后，就可以从中去掉任一条边，去掉的边不一样，故可能得到不同的生成树。



(a)



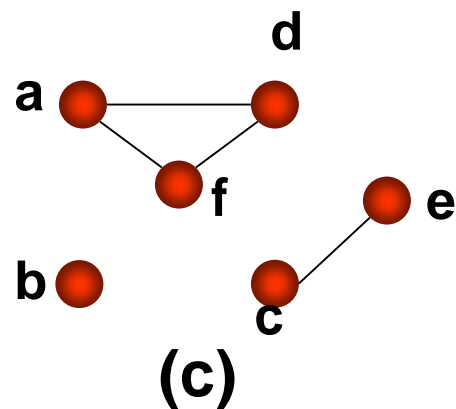
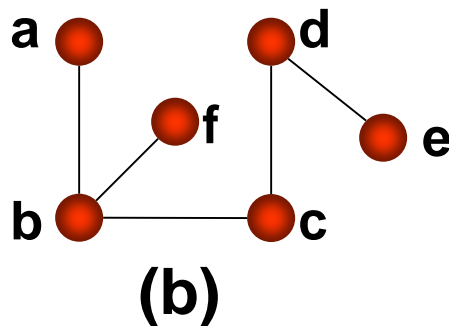
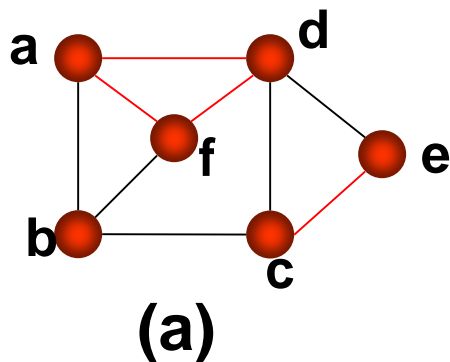
(b)



(c)

在上图(a)中，删去边2,3,5，就得到生成树 (b)，若删去边2, 4, 6,可得到生成树 (c)。

生成树



给定图**G**的一棵树**T**，我们称**G-T**，即**G**删去**T**中各边后的子图为**T**的**余树**。

注：余树不一定连通，也不一定无回路，因而余树不一定是树，更不一定是生成树。

作业

P66

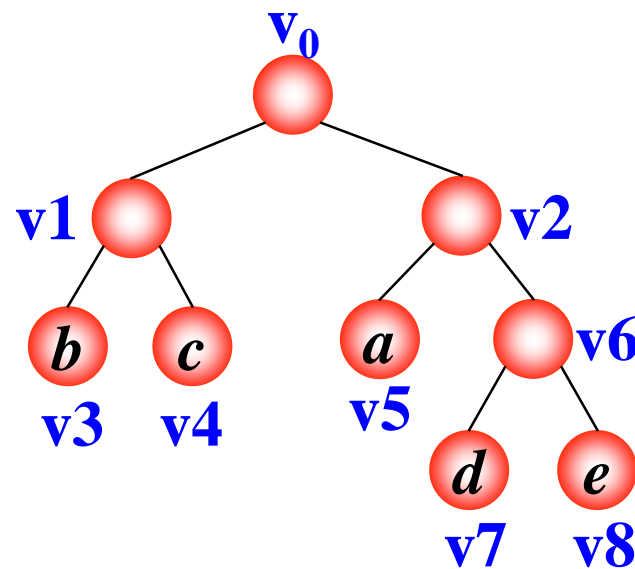
□ 1, 2

3.6 Huffman树

□ 定义3.6.1

除树叶外, 其余结点的正度最多为**2**的**外向树**称为二叉树. 如果它们的正度都是**2**, 称为完全二叉树.

外向树--若一个有向树**T**, 有且只有一个顶点入度为**0**, 其余顶点入度都为**1**, 则称**T**为外向树, 其中入度为**0**的节点称为根节点, 出度为**0**的节点称为叶节点

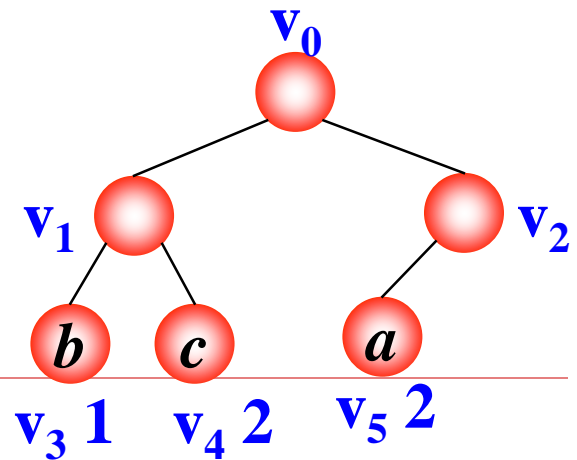


树和二叉树

- 二叉树的每个结点至多只有二棵子树
二叉树的子树有左右之分，次序不能颠倒
二叉树的第*i*层至多有 $2^{(i-1)}$ 个结点
深度为*k*的二叉树至多有 $2^k - 1$ 个结点(根结点的深度为1)
 - 树和二叉树的两个主要差别：
 - 树中结点的最大度数没有限制，而二叉树结点的最大出度数为2
 - 树的结点无左、右之分，而二叉树的结点有左、右之分
-

- 如果二叉树**T**的每个树叶结点 v_i 都分别赋以一个正实数 w_i , 则称**T**是赋权二叉树。
- 路径: 从树中一个结点到另一个结点之间的边构成这两个结点间的路径
- 从根到树叶 v_i 的路径 $P(v_0, v_i)$ 所包含的边数计为该路径的长度 l , 这样二叉树**T**带权的路径总长度为:

$$WPL = \sum_i l_i w_i, \quad v_i \text{ 是树叶}$$

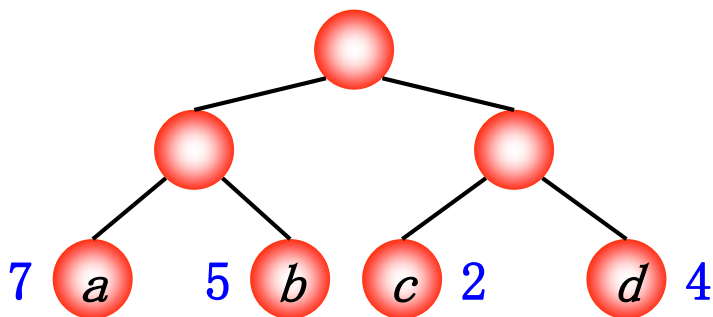


最优二叉树

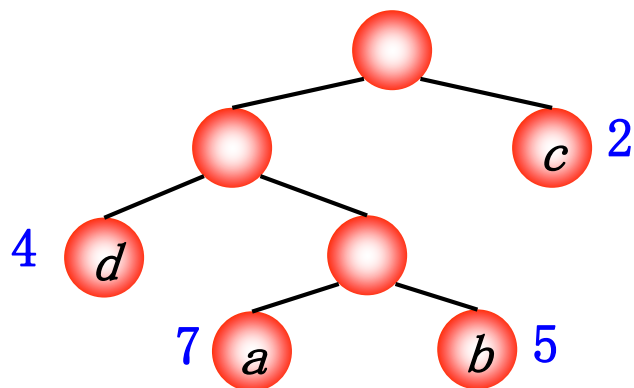
- 如果给定了树叶数目以及它们的权值, 可以构造许多不同的赋权二叉树

在这些赋权二叉树中, 必定存在带权路径总长最小的二叉树, 这样的树称为最优二叉树

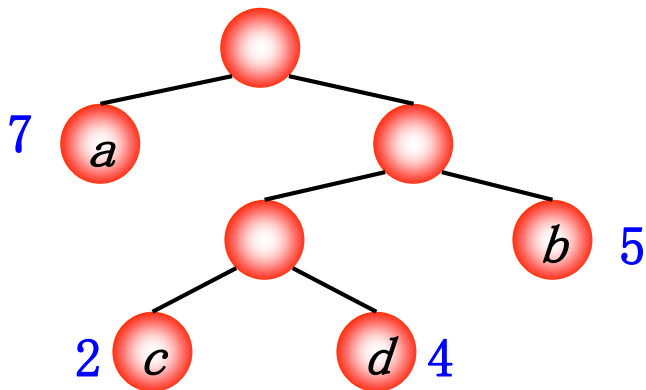
例：有4个结点 a , b , c , d , 权值分别为 7, 5, 2, 4, 试构造以此4个结点为叶子结点的二叉树。



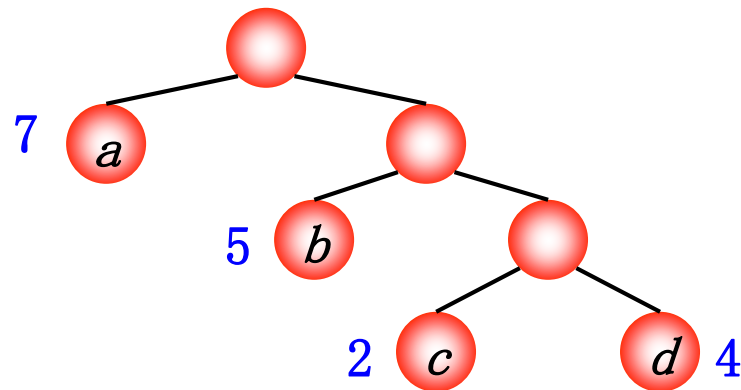
$$\text{WPL} = 7 \times 2 + 5 \times 2 + 2 \times 2 + 4 \times 2 = 36$$



$$\text{WPL} = 7 \times 3 + 5 \times 3 + 2 \times 1 + 4 \times 2 = 46$$



$$\text{WPL} = 7 \times 1 + 5 \times 2 + 2 \times 3 + 4 \times 3 = 35$$



$$\text{WPL} = 7 \times 1 + 5 \times 2 + 2 \times 3 + 4 \times 3 = 35$$

最优二叉树

带权路径长(**WPL: Weighted Path Length**)

最小的二叉树

(权值大的结点离根最近)

因为构造这种树的算法是由哈夫曼于 1952 年提出的，

所以被称为哈夫曼树，相应的算法称为哈夫曼算法。

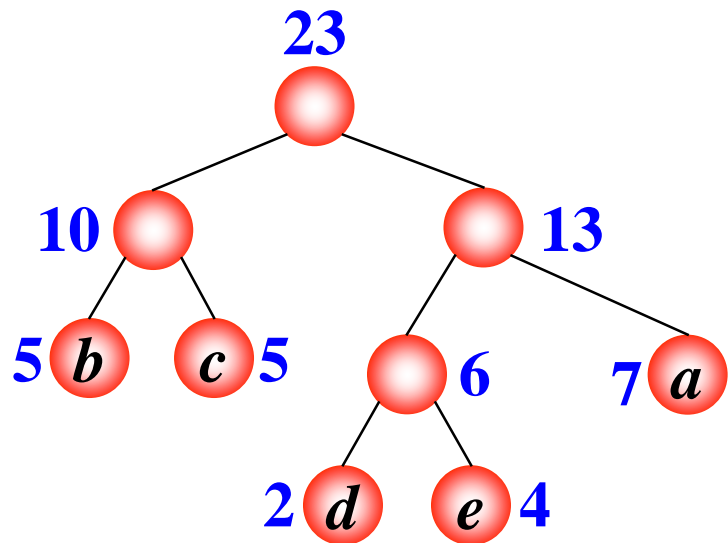
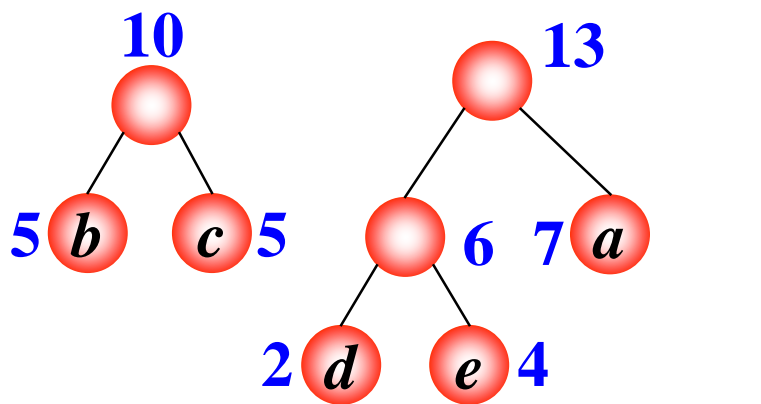
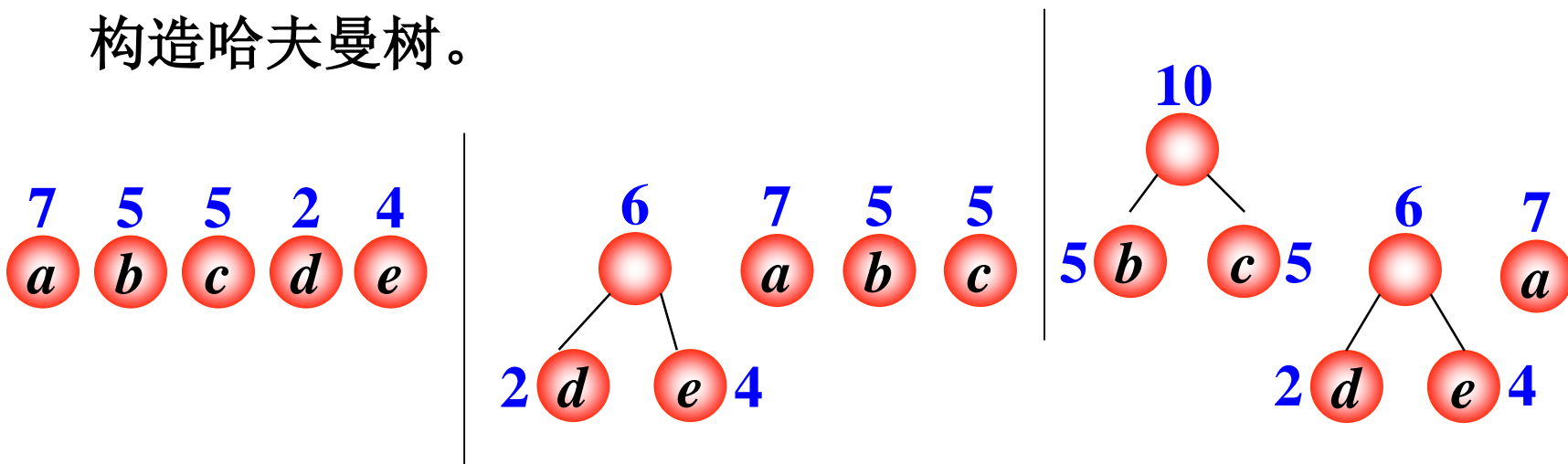
Huffman树

□ 哈夫曼算法:

1. 对 $n(\geq 2)$ 个权值进行排序,满足 $w_{i1} \leq w_{i2} \leq \dots \leq w_{in}$

2. 计算 $w_i = w_{i1} + w_{i2}$ 作为中间结点 v_i 的权, v_i 的左儿子是 v_{i1} , 右儿子是 v_{i2} . 在权序列中删去 w_{i1} 和 w_{i2} , 加入 w_i , $n \leftarrow n-1$. 若 $n=1$, 结束. 否则转**1**.

例：有5个结点 a, b, c, d, e ，权值分别为 7, 5, 5, 2, 4，构造哈夫曼树。



Huffman树

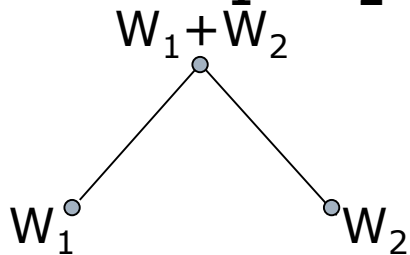
□ 定理3.6.1

由Huffman算法得到的二叉树是最优二叉树。

证明：假定 $n \geq 3$, $w_1 \leq w_2 \leq \dots \leq w_n$, 并设 T 是最优树。

则一定有 w_1 离根最远, 即 $l_1 = \max\{l_1, l_2, \dots, l_n\}$ 。否则, 假设 $w_k > w_1$ 而 $l_k > l_1$, 则有 $w_k l_1 + w_1 l_k < w_k l_k + w_1 l_1$ 。所以将 w_k 和 w_1 对调可得到 T' , 其满足 $WPL(T') < WPL(T)$, 与 T 最优矛盾。

同时立即可知, w_1 必有兄弟。否则让 w_1 赋值给该树叶的父亲结点, 就可得到路径总长更小的树。由于 w_2 是序列中次最小的权, 故可令 w_1 的兄弟是 w_2 。因此分支 $w_1 + w_2$ 可以是最优树 T 的子图。



定理3.6.1-证明(续)

设 T_n 是 n 个树叶的最优树，收缩分支 w_1+w_2 后是对应的 $n-1$ 个树叶的 T'_{n-1} 。

在 $n-1$ 个权(其中之一是 w_1+w_2)时，也有其最优二叉树 T_{n-1} ，然后将 w_1+w_2 分支展开后又得到有 n 个权的二叉树 T'_n 。

CLAIM. T'_{n-1} 和 T'_n 都是最优树。

当算法执行到 $n=2$ 时，自然是一棵最优树。再与分支收缩的过程相反进行展开，最后得到的 T'_n 一定是最优二叉树

定理3.6.1-证明(续)

CLAIM. T'_{n-1} 和 T'_n 都是最优树。

因为， T_n 和 T_{n-1} 分别是最优树，所以

$$\mathbf{WPL}(T_n) \leq \mathbf{WPL}(T'_n),$$

$$\mathbf{WPL}(T_{n-1}) \leq \mathbf{WPL}(T'_{n-1})$$

由于，

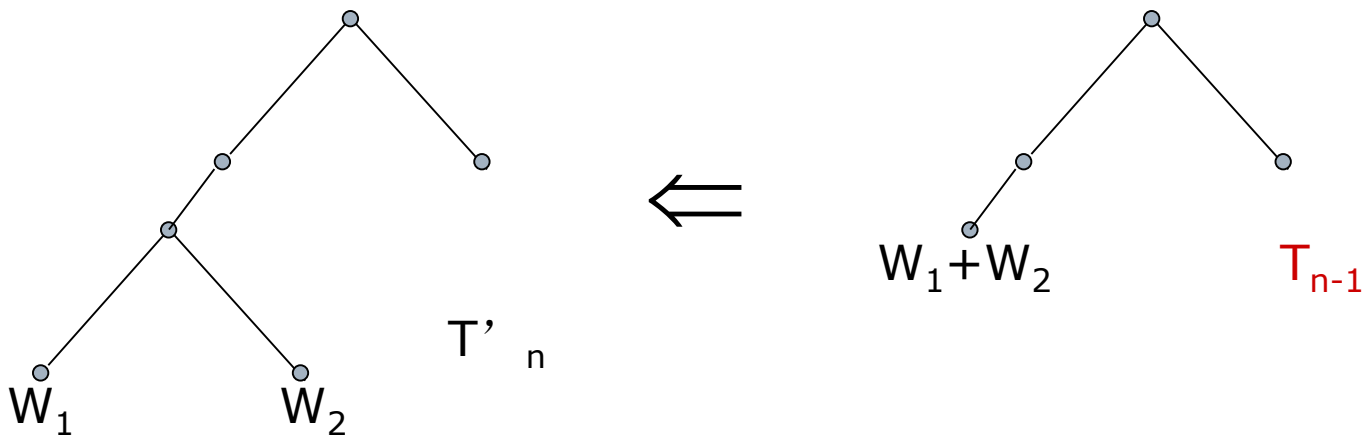
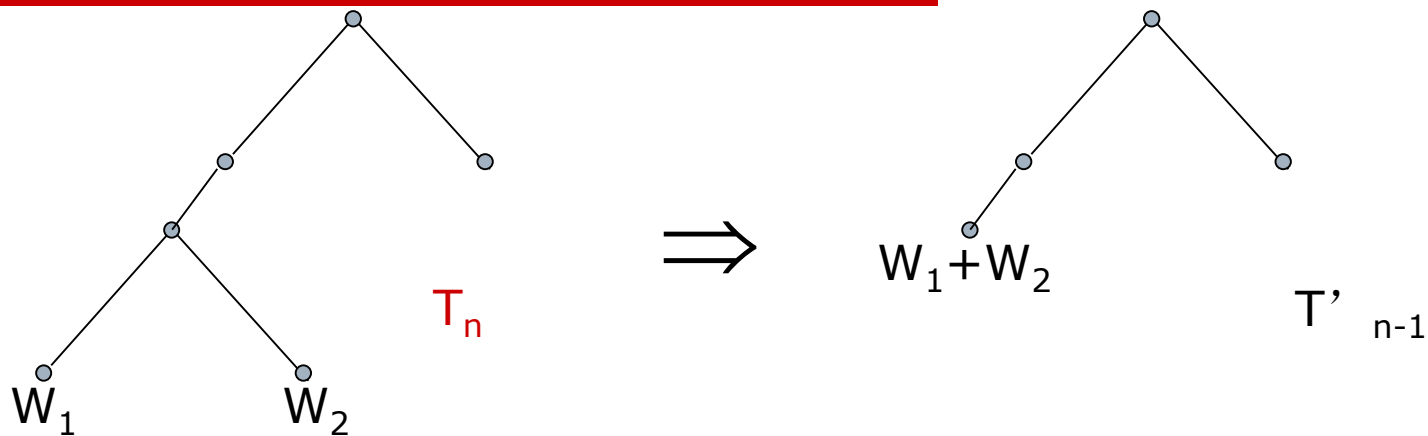
$$\mathbf{WPL}(T'_{n-1}) = \mathbf{WPL}(T_n) - (W_1 + W_2),$$

$$\mathbf{WPL}(T_{n-1}) = \mathbf{WPL}(T'_n) - (W_1 + W_2)$$

所以有 $\mathbf{WPL}(T'_{n-1}) \leq \mathbf{WPL}(T_{n-1})$

同理有 $\mathbf{WPL}(T'_n) \leq \mathbf{WPL}(T_n)$

定理3.6.1-证明(续)



最佳前缀码—哈夫曼编码

哈夫曼树的应用很广，哈夫曼编码就是其在电讯通信中的应用之一。在电讯通信业务中，通常用二进制编码来表示字母或其他字符，并用这样的编码来表示字符序列。

例：如果需传送的电文为 ‘A B A C C D A’，它只用到四种字符，用两位二进制编码便可分辨。假设 A, B, C, D 的编码分别为 00, 01, 10, 11，则上述电文便为 ‘00010010101100’（共 14 位），译码员按两位进行分组译码，便可恢复原来的电文。

在编码过程通常要考虑两个问题

- 数据的最小冗余编码问题
- 译码的惟一性问题

➤ 数据的最小冗余编码问题

在实际应用中，各个字符的出现频度是不尽相同的，有些字符出现的频率较高，有些字符出现的频率较低。我们希望用较短的编码来表示那些出现频率大的字符，用较长的编码来表示出现频率少的字符，从而**使得编码序列的总长度最小**，使所需总空间量最少。这就是最小冗余编码问题。

在上例中，若假设 A, B, C, D 的编码分别为 0, 00, 1, 01，则电文 ‘A B A C C D A’ 便为 ‘000011010’（共 9 位）。

但此编码存在多义性：可译为 ‘A A A A C C A C A’、‘B B C C D A’、‘A B A C C D A’ 等。

➤ 译码的惟一性问题

要求任一字符的编码都不能是另一字符编码的前缀！
这种编码称为**最佳前缀码**（其实是非前缀码）。

利用最优二叉树可以很好地解决上述两个问题。

最佳前缀码

定义 设 $\alpha = \alpha_1\alpha_2\cdots\alpha_{n-1}\alpha_n$ 是长度为 n 的符号串,
 $\alpha_1\alpha_2\cdots\alpha_k$ 称作 α 的长度为 k 的**前缀**, $k=1,2,\dots,n-1$.

若非空字符串 $\beta_1, \beta_2, \dots, \beta_m$ 中任何两个互不为前缀,
则称 $\{\beta_1, \beta_2, \dots, \beta_m\}$ 为**前缀码**

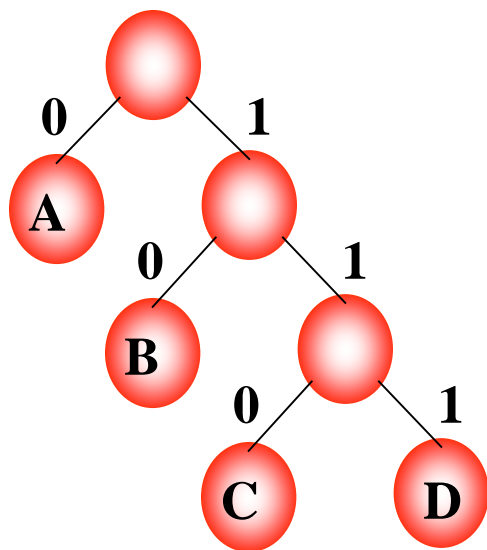
只出现两个符号(如**0**与**1**)的前缀码称作**二元前缀码**

例如 $\{0, 10, 110, 1111\}$, $\{10, 01, 001, 110\}$ 是二元前缀码
 $\{0, 10, 010, 1010\}$ 不是前缀码

用二叉树设计二进制前缀码

以电文中的字符作为叶子结点构造二叉树。然后将二叉树中结点引向其左孩子的分支标‘0’，引向其右孩子的分支标‘1’；每个字符的编码即为从根到每个叶子的路径上得到的0,1序列。如此得到的即为二进制前缀码。

例：



编码： A: 0
B: 10
C: 110
D: 111

用哈夫曼树设计总长最短的二进制前缀编码

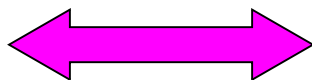
假设各个字符在电文中出现的**次数**（或频率）为 w_i ，其**编码长度**为 l_i ，电文中只有 n 种字符，则**电文编码总长**为：

$$\text{WPL} = \sum_{i=1}^n w_i l_i$$

从根到叶子的路径长度

叶子结点的权

设计电文总长最短的编码



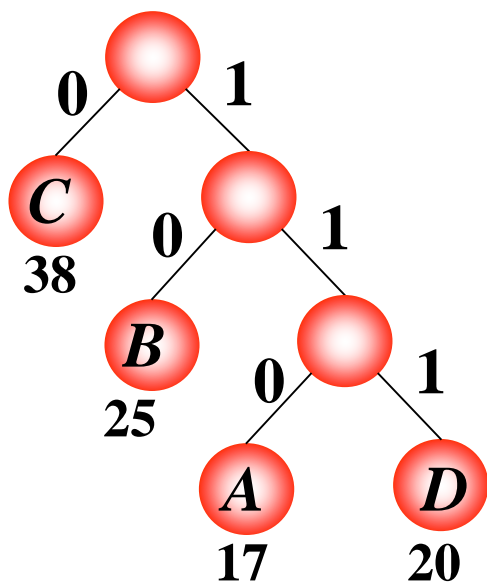
设计哈夫曼树（以 n 种字符出现的频率作权）

由哈夫曼树得到的二进制前缀码称为**哈夫曼编码**。

例：设 A, B, C, D 的频率（即权值）分别为 $17\%, 25\%, 38\%, 20\%$,

试设计哈夫曼编码（最佳前缀码）。

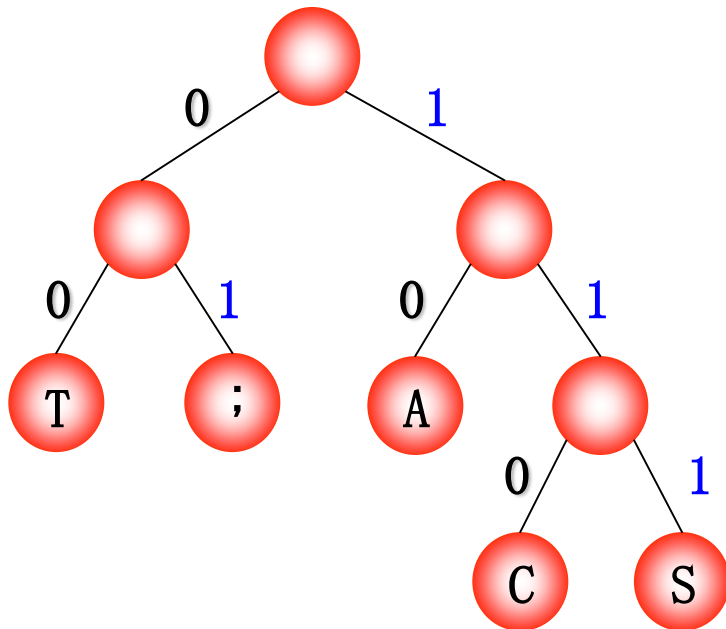
解：



编码： $C: 0$
 $B: 10$
 $A: 110$
 $D: 111$

译码

从哈夫曼树根开始，对待译码电文逐位取码。若编码是“0”，则向左走；若编码是“1”，则向右走，一旦到达叶子结点，则译出一个字符；再重新从根出发，直到电文结束。



电文为 “1101000”

译文只能是 “CAT”

3.7 最短树(最小生成树)

- 在赋权连通图中，计算该图总长最小的支撑树，即求最短树
 - 两种算法
 - **Kruskal**算法
 - **Prim**算法
-

3.7.1 Kruskal算法

□ 基本思想:

不断往**T**中加入当前的最短边**e**,如果此时会构成回路,那么它一定是这个回路中的最长边,删之。直至最后达到**n-1**条边为止。这时**T**中不包含任何回路,因此是树。

依据树的性质:

若**T**有**n-1**条边且无回路,则**T**是树

3.7.1 Kruskal算法

□ **Kruskal**算法的描述如下:

$T \leftarrow \Phi$ 。

当 $|T| < n - 1$ 且 $E \neq \Phi$ 时,

begin

1. $e \leftarrow E$ 中最短边.

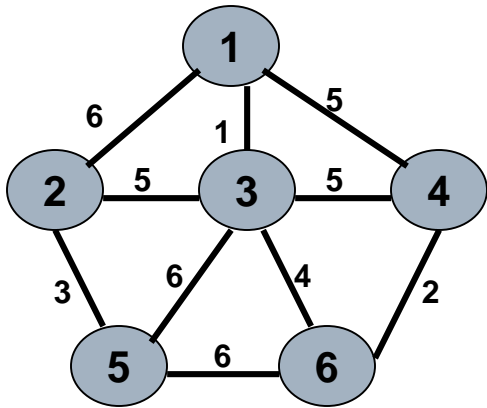
2. $E \leftarrow E - e$.

3. 若 $T + e$ 无回路, 则 $T \leftarrow T + e$.

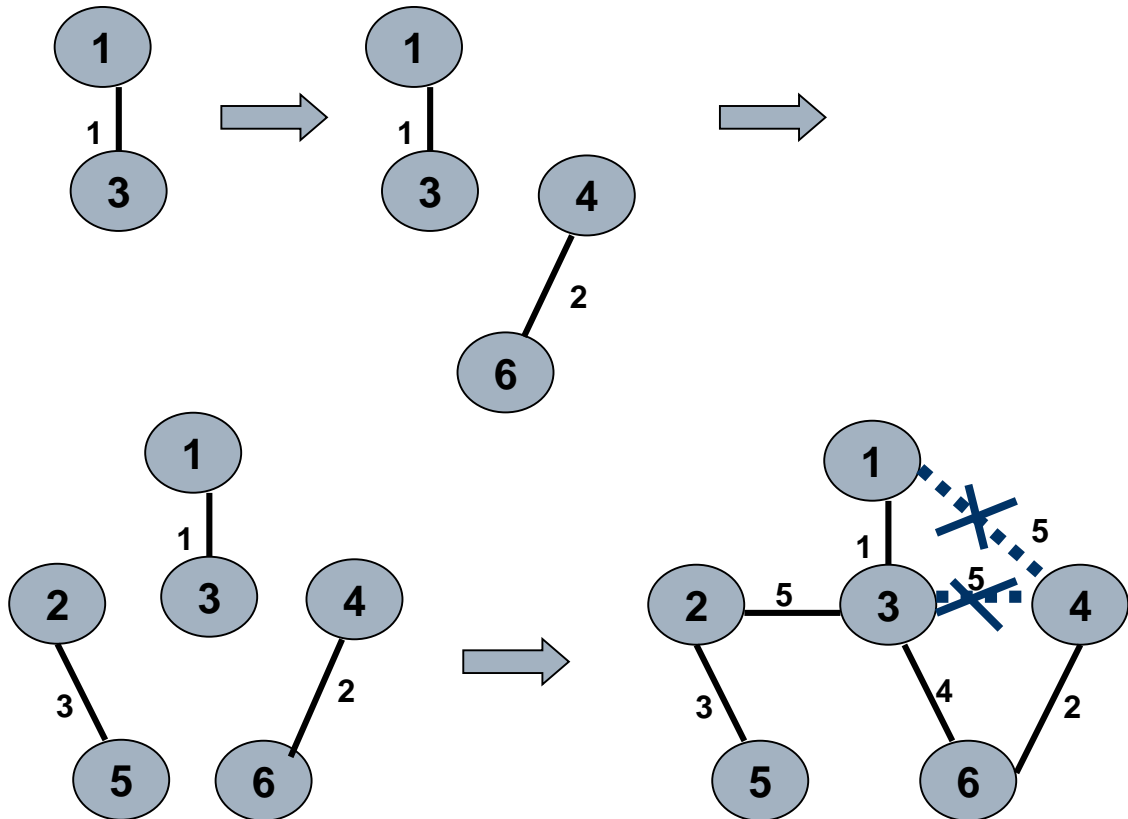
end

若 $|T| < n - 1$ 打印“非连通”, 否则输出最短树

Kruskal算法实例



图G

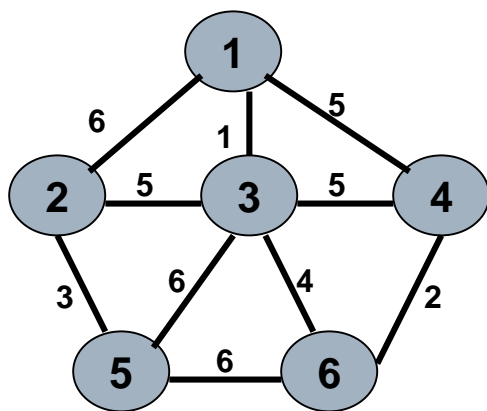


最短树

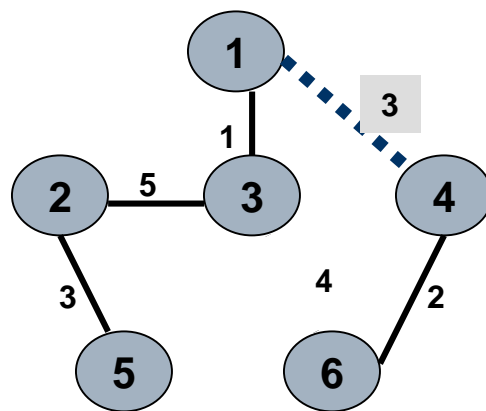
□ 定理 3.7.1

$T=(V, E')$ 是赋权连通图 $G=(V, E)$ 的最短树, 当且仅当对任意的余树边 $e \in E-E'$, 回路 $C^e(C^e \subseteq E' + e)$, 满足其边权 $w(e) \geq w(a), a \in C^e(a \neq e)$.

证明: 必要性. 如果存在一条余树边 e , 满足 $w(e) < w(a)$, $a \in C^e$, 则 $T \oplus (a, e)$ 得到新树 T' 比 T 更加短, 与 T 是最短树矛盾.



图G



T

最短树

□ 充分性.即证明: 设 $T=(V, E')$ 是 $G=(V, E)$ 的一棵树, 如果对任意的余树边 $e \in E - E'$, 回路 $C^e(C^e \subseteq E' + e)$, 满足其边权 $w(e) \geq w(a), a \in C^e(a \neq e)$, 那么 T 是最短树。

反证。若存在比 T 还短的树 T' , 则 $T' - T \neq \emptyset$, 设 $e \in T' - T$, 则 $T + e$ 构成唯一回路 C^e . 根据已知, 对任意的 T' 关于 T 的余树边 $e \in T' - T$, 它与回路 C^e 里的树枝边 $a(a \neq e)$ 相比都有 $w(e) \geq w(a)$, 则有 $w(T') \geq w(T)$, 与假设矛盾。

注: 因为 T 和 T' 都是 G 的生成树, 所以结点数和边数都相同, 分别是 n 和 $n-1$ 。

最短树

□ 定理 3.7.2

Kruskal算法的计算复杂性是

$$O(m+p * \log m)$$

其中**p**是迭代次数.

3.7.2 Prim算法

□ Prim算法的基本思想是：

首先任选一结点 v_0 构成集合 U , 然后不断在 $V-U$ 中选一条到 U 中某点(比如 v)最短的边 (u,v) 进入树 T , 并且 $U=U+v$, 直到 $U=V$

最短树

□ **Prim**算法的描述如下:

1. $t \leftarrow v_1, T \leftarrow \emptyset, U \leftarrow \{t\}$

2. **While** $U \neq V$ **do**
begin

3. $W(t, u) = \min_{v \in V-U} \{w(t, v)\}$

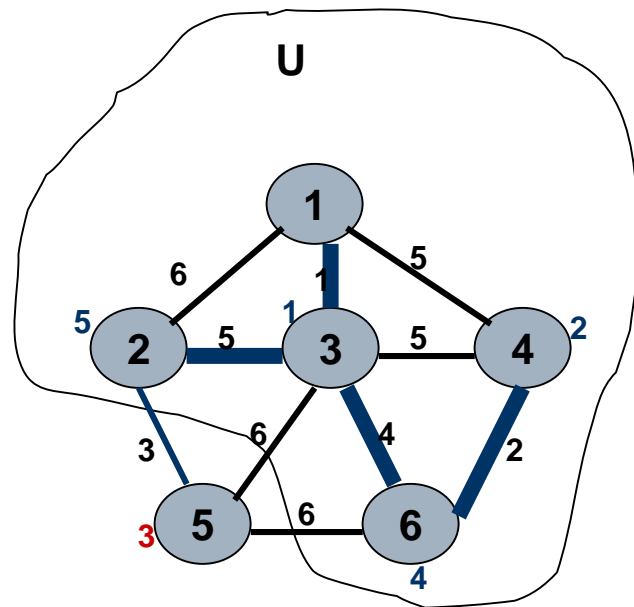
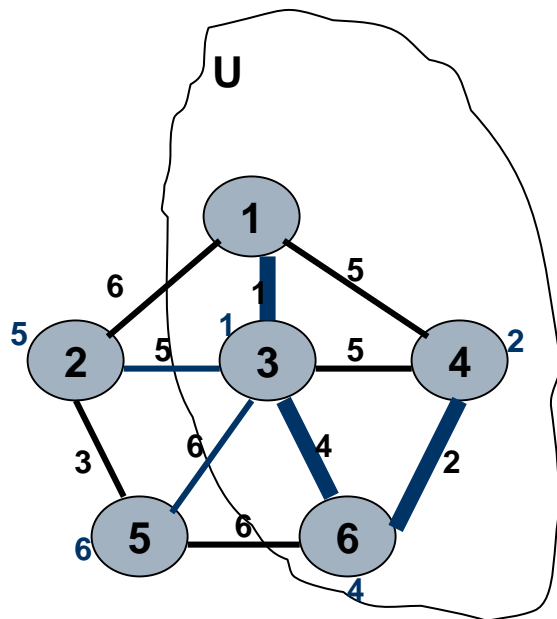
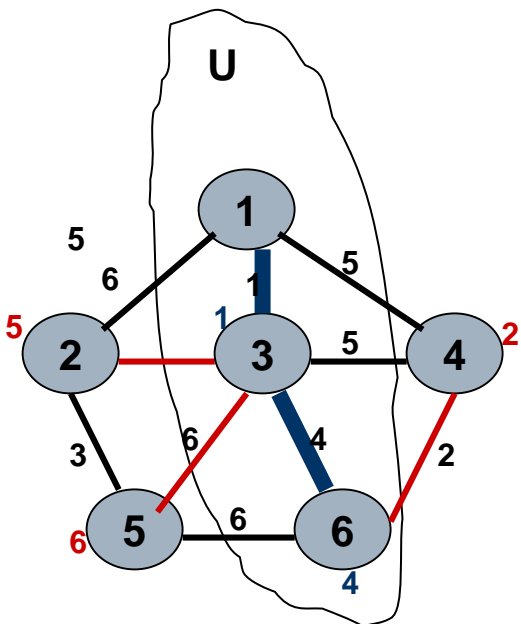
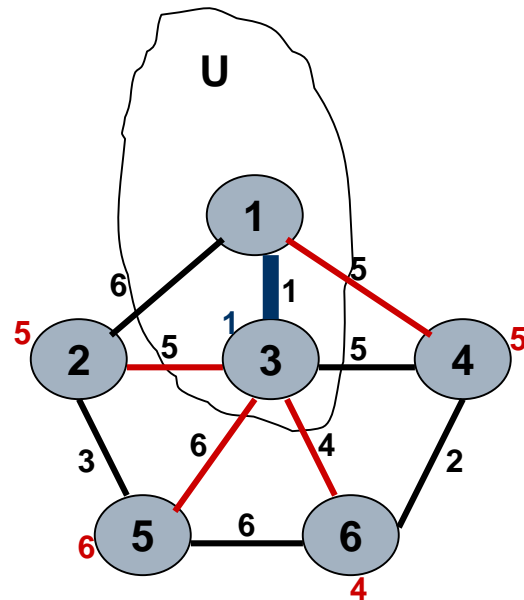
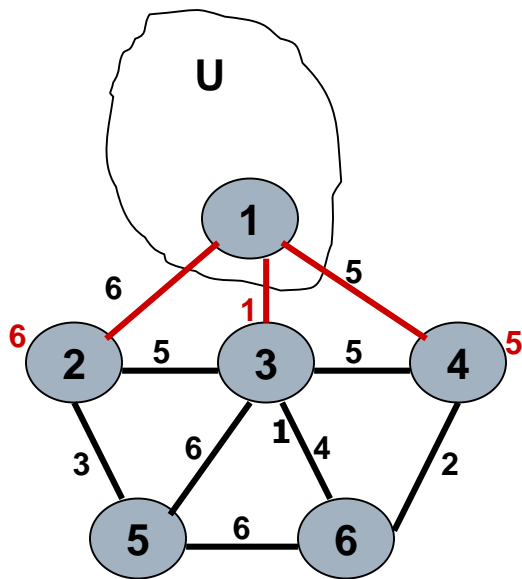
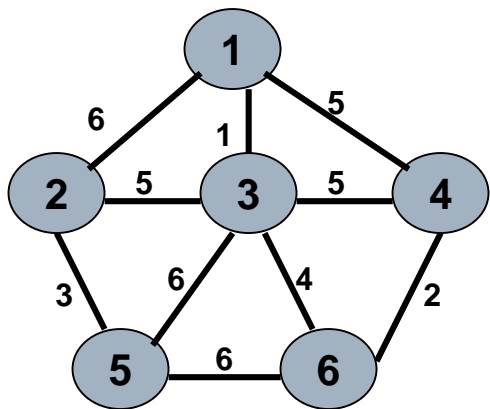
4. $T \leftarrow T + e(t, u)$

5. $U \leftarrow U + u$

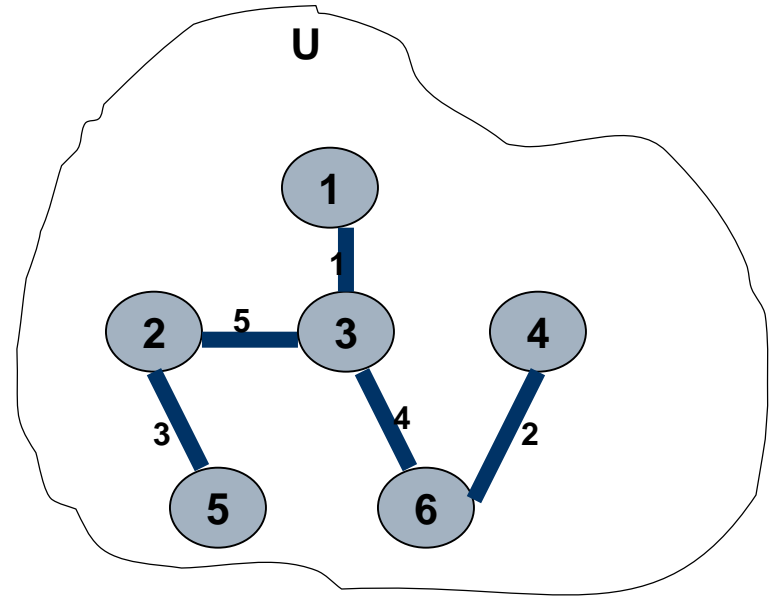
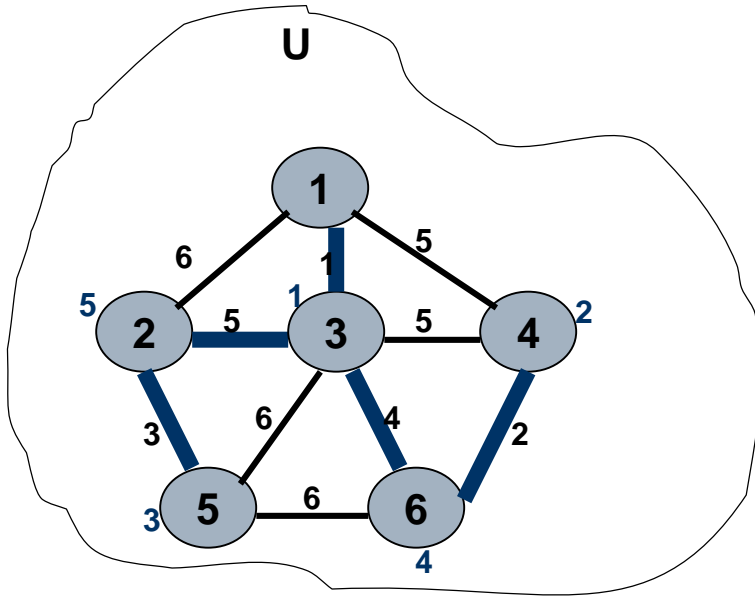
6. **For** $v \in V-U$ **do**

$W(t, v) \leftarrow \min\{w(t, v), w(u, v)\}$

end



Prim算法实例



注意：每当新的结点并入 U 之后，则调整仍在 $V - U$ 集合中的结点直接至 U 中结点的边中的最小边的距离

最短树

□ 定理 3.7.3

设 V' 是赋权连通图 $G=(V,E)$ 的结点真子集, e 是二端点分跨在 V' 和 $V-V'$ 的最短边, 则 G 中一定存在包含 e 的最短树 T .

证明: 设 T_0 是 G 的一棵最短树, 若 $e \notin T_0$, 则 $T_0 + e$ 构成唯一回路. 该回路一定包含 e 和 e' (u, v), 其中 $u \in V'$, $v \in V-V'$. 由已知条件 $w(e) \leq w(e')$, 作 $T_0 \oplus (e, e')$ 得到的仍是最短树.

最短树

□ 定理 3.7.4

Prim算法的结果是：得到赋权连通图**G**的一棵最短树。

证明：首先证明它是一棵支撑树。采用归纳法。

初始 $U = \{v_1\}$, $T = \emptyset$, 它是由 U 导出的树。

设 $|U| = i$, T 是 U 导出的树。则下一次迭代时, U 中增加一新结点 u , T 中也加入一条与 u 相连的边, 因此 T 是连通的, 则有 $|U| - 1$ 条边, 它是由 U 导出的一棵树。

因此最终 T 是**G**的支撑树。

最短树

□ 以下再证**T**是一棵最短树

设**T**₀是**G**的一棵最短树, **T** ≠ **T**₀, 由定理3.7.3, 对任意的 **e** ∈ **T** - **T**₀, 一定有最短树 **T'** = **T**₀ ⊕ (**e**, **e'**), 其中 **e'** ∈ **C**^{**e**} ∩ **T**₀. 继续对**T'** 如此处理, 直到最终 **T'** = **T**, 它仍然是最短树.

注: 根据**T**的构造过程, 对于**T**中的每条边**e**, 都是二端点分跨在某个**V'** 和**V-V'** 的最短边

作业

P66

□ 14, 16

考核范围

□ 《数理逻辑与集合论》

第 1 章；第 2 章（不含**2.10**）；第 4 章；
第 5 章（不包括存在型前束范式与**5.6**）

归结法不考核

□ 《图论与代数结构》

第 1 章：（不包括 **1.2.4, 1.2.5, 1.2.6**）

第 2 章： **2.1, 2.3, 2.4**

第 3 章： **3.1, 3.6, 3.7**
