离散数学

（第 3 版）

智能科学与技术学院 2024 级

目录

目录

7.1 无向树及其性质

定义 1.1.1

连通无回路的无向图称作 *无向树*，或简称为 *树*. 每个连通分支都是树的无向图称作 *森林*. 平凡图称作 *平凡树*. 在无向树中，悬挂顶点称作 *树叶*，度数大于等于 2 的顶点称作 *分支点*.

7.1 无向树及其性质

定义 1.1.1

连通无回路的无向图称作 *无向树*，或简称为 *树*. 每个连通分支都是树的无向图称作 *森林*. 平凡图称作 *平凡树*. 在无向树中，悬挂顶点称作 *树叶*，度数大于等于 2 的顶点称作 *分支点*.

定理 1.1.1

设 $G = \langle V, E \rangle$ 是 n 阶 m 条边的无向图，则下列各命题是等价的.

1. G 是树.
2. G 中任意两个顶点之间存在唯一的路径.
3. G 中无回路且 $m = n - 1$.
4. G 是连通的且 $m = n - 1$.
5. G 是连通的且任何边均为桥.
6. G 中没有回路，但在任何两个不同的顶点之间加一条新边后所得的图中有唯一的一个含新边的圈.

证明.

(1) $\Rightarrow$ (2). 由 G 的连通性及推论??知,$\forall u, v \in V, u$ 与 v 之间存在路径. 若路径不是唯一的,设 Γ_1 和 Γ_2 都是 u 到 v 的路径,则必存在由 Γ_1 和 Γ_2 上的边构成的回路,这与 G 中无回路矛盾.

证明.

$(1) \Rightarrow (2)$. 由 G 的连通性及推论??知,$\forall u, v \in V, u$ 与 v 之间存在路径. 若路径不是唯一的,设 Γ_1 和 Γ_2 都是 u 到 v 的路径,则必存在由 Γ_1 和 Γ_2 上的边构成的回路,这与 G 中无回路矛盾.

$(2) \Rightarrow (3)$. 首先证明 G 中无回路. 若 G 中存在关联某顶点 v 的环,则 v 到 v 存在长为 0 和 1 的两条路径(注意初级回路是路径的特殊情况),这与已知条件矛盾. 若 G 中存在长度大于等于 2 的圈,则圈上任何两个顶点之间都存在两条不同的路径,这也引出矛盾,下面用归纳法证明 $m = n - 1$.

当 $n = 1$ 时,G 为平凡图,结论显然成立. 设 $n \leqslant k$ 时结论成立,这里 $k \geqslant 1$. 当 $n = k + 1$ 时,设 $e = (u, v)$ 为 G 中的一条边,由于 G 中无回路,所以 $G - e$ 为两个连通分支 G_1, G_2. 设 n_i, m_i 分别为 G_i 中的顶点数和边数,则 $n_i \leqslant k$. 由归纳假设,$m_i = n_i - 1, i = 1, 2$. 故 $m = m_1 + m_2 + 1 = n_1 + n_2 - 2 + 1 = n - 1$. 得证当 $n = k + 1$ 时结论也成立.

证明.

(1) $\Rightarrow$ (2). 由 G 的连通性及推论??知,$\forall u, v \in V, u$ 与 v 之间存在路径. 若路径不是唯一的,设 Γ_1 和 Γ_2 都是 u 到 v 的路径,则必存在由 Γ_1 和 Γ_2 上的边构成的回路,这与 G 中无回路矛盾.

(2) $\Rightarrow$ (3). 首先证明 G 中无回路. 若 G 中存在关联某顶点 v 的环,则 v 到 v 存在长为 0 和 1 的两条路径(注意初级回路是路径的特殊情况),这与已知条件矛盾. 若 G 中存在长度大于等于 2 的圈,则圈上任何两个顶点之间都存在两条不同的路径,这也引出矛盾,下面用归纳法证明 $m = n - 1$.

当 $n = 1$ 时,G 为平凡图,结论显然成立. 设 $n \leqslant k$ 时结论成立,这里 $k \geqslant 1$. 当 $n = k + 1$ 时,设 $e = (u, v)$ 为 G 中的一条边,由于 G 中无回路,所以 $G - e$ 为两个连通分支 G_1, G_2. 设 n_i, m_i 分别为 G_i 中的顶点数和边数,则 $n_i \leqslant k$. 由归纳假设,$m_i = n_i - 1, i = 1, 2$. 故 $m = m_1 + m_2 + 1 = n_1 + n_2 - 2 + 1 = n - 1$. 得证当 $n = k + 1$ 时结论也成立.

(3) $\Rightarrow$ (4). 只要证明 G 是连通的. 假设不然,设 G 有 $s(\geqslant 2)$ 个连通分支 $G_1, G_2, \cdots, G_s$. 每个 G_i 中均无回路,因而 G_i 全为树. 由 (1) $\Rightarrow$ (2) $\Rightarrow$ (3) 可知,$m_i = n_i - 1$. 于是 $m = \sum_{i=1}^{s} m_i = \sum_{i=1}^{s} n_i - s = n - s$. 由于 $s \geqslant 2$,这与 $m = n - 1$ 矛盾.

定理1.1.1证明(续)

(4) $\Rightarrow$ (5). 只要证明 G 中每条边均为桥. $\forall e \in E$, 均有 $|E(G-e)| = n-1-1 = n-2$. 由第 5 章习题 5.50 可知, $G-e$ 不是连通图, 故 e 为桥.

定理1.1.1证明(续)

(4) $\Rightarrow$ (5). 只要证明 G 中每条边均为桥. $\forall e \in E$, 均有 $|E(G-e)| = n-1-1 = n-2$. 由第 5 章习题 5.50 可知, $G-e$ 不是连通图, 故 e 为桥.

(5) $\Rightarrow$ (6). 由于 G 中每条边均为桥, 因而 G 中无圈. 又由于 G 连通, 所以 G 为树. 由 (1) $\Rightarrow$ (2) 可知, G 中任意两个不同的顶点 u, v 之间存在唯一的路径 Γ. 设 e 是在 u, v 之间添加的新边, 则 $\Gamma \cup e$ 是一个圈, 且显然是唯一的.

定理1.1.1证明(续)

(4) $\Rightarrow$ (5). 只要证明 G 中每条边均为桥. $\forall e \in E$, 均有 $|E(G-e)| = n-1-1 = n-2$. 由第 5 章习题 5.50 可知, $G-e$ 不是连通图, 故 e 为桥.

(5) $\Rightarrow$ (6). 由于 G 中每条边均为桥, 因而 G 中无圈. 又由于 G 连通, 所以 G 为树. 由 (1) $\Rightarrow$ (2) 可知, G 中任意两个不同的顶点 u, v 之间存在唯一的路径 Γ. 设 e 是在 u, v 之间添加的新边, 则 $\Gamma \cup e$ 是一个圈, 且显然是唯一的.

(6) $\Rightarrow$ (1). 只要证明 G 是连通的. 对任意两个不同的顶点 u 和 v, 在 u, v 之间添加一条新边 e 后产生唯一的一个含 e 的圈 C. 显然, $C-e$ 为 G 中 u 到 v 的通路, 故 $u \sim v$. 由 u, v 的任意性可知, G 是连通的.

定理1.1.1证明(续)

(4) $\Rightarrow$ (5). 只要证明 G 中每条边均为桥. $\forall e \in E$, 均有 $|E(G-e)| = n-1-1 = n-2$. 由第 5 章习题 5.50 可知, $G-e$ 不是连通图, 故 e 为桥.

(5) $\Rightarrow$ (6). 由于 G 中每条边均为桥, 因而 G 中无圈. 又由于 G 连通, 所以 G 为树. 由 (1) $\Rightarrow$ (2) 可知, G 中任意两个不同的顶点 u, v 之间存在唯一的路径 Γ. 设 e 是在 u, v 之间添加的新边, 则 $\Gamma \cup e$ 是一个圈, 且显然是唯一的.

(6) $\Rightarrow$ (1). 只要证明 G 是连通的. 对任意两个不同的顶点 u 和 v, 在 u, v 之间添加一条新边 e 后产生唯一的一个含 e 的圈 C. 显然, $C-e$ 为 G 中 u 到 v 的通路, 故 $u \sim v$. 由 u, v 的任意性可知, G 是连通的.

定理 1.1.2

设 T 是 n 阶非平凡的无向树, 则 T 中至少有两片树叶.

定理1.1.1证明(续)

(4) $\Rightarrow$ (5). 只要证明 G 中每条边均为桥. $\forall e \in E$, 均有 $|E(G-e)| = n-1-1 = n-2$. 由第 5 章习题 5.50 可知, $G-e$ 不是连通图, 故 e 为桥.

(5) $\Rightarrow$ (6). 由于 G 中每条边均为桥, 因而 G 中无圈. 又由于 G 连通, 所以 G 为树. 由 (1) $\Rightarrow$ (2) 可知, G 中任意两个不同的顶点 u, v 之间存在唯一的路径 Γ. 设 e 是在 u, v 之间添加的新边, 则 $\Gamma \cup e$ 是一个圈, 且显然是唯一的.

(6) $\Rightarrow$ (1). 只要证明 G 是连通的. 对任意两个不同的顶点 u 和 v, 在 u, v 之间添加一条新边 e 后产生唯一的一个含 e 的圈 C. 显然, $C-e$ 为 G 中 u 到 v 的通路, 故 $u \sim v$. 由 u, v 的任意性可知, G 是连通的.

定理 1.1.2

设 T 是 n 阶非平凡的无向树, 则 T 中至少有两片树叶.

证明.

设 T 有 x 片树叶, 由握手定理及定理 1.1.1 可知,

$$2(n-1) = \sum d(v_i) \geqslant x + 2(n-x),$$

由上式解得 $x \geqslant 2$. □

例 1.1.1

画出所有 6 阶非同构的无向树.

例 1.1.1

画出所有 *6* 阶非同构的无向树.

解

设 T 是 6 阶无向树. 由定理 1.1.1 可知, T 的边数 $m = 5$. 由握手定理可知, T 的 6 个顶点的度数之和等于 10. 又有 $\delta(T) \geqslant 1, \Delta(T) \leqslant 5$. 于是, T 的度数列必为以下情况之一: (1) $1, 1, 1, 1, 1, 5$; (2) $1, 1, 1, 1, 2, 4$; (3) $1, 1, 1, 1, 3, 3$; (4) $1, 1, 1, 2, 2, 3$; (5) $1, 1, 2, 2, 2, 2$.

例 1.1.1

画出所有 *6* 阶非同构的无向树.

解

设 T 是 6 阶无向树. 由定理 1.1.1 可知, T 的边数 $m = 5$. 由握手定理可知, T 的 6 个顶点的度数之和等于 10. 又有 $\delta(T) \geqslant 1, \Delta(T) \leqslant 5$. 于是, T 的度数列必为以下情况之一: (1) $1,1,1,1,1,5$; (2) $1,1,1,1,2,4$; (3) $1,1,1,1,3,3$; (4) $1,1,1,2,2,3$; (5) $1,1,2,2,2,2$.

它们对应的树如图 1.1.1 所示, 其中 T_1 对应于 (1), T_2 对应于 (2), T_3 对应于 (3), T_4, T_5 对应于 (4), T_6 对应于 (5). (4) 对应两棵非同构的树, 在一棵树中两个 2 度顶点相邻, 在另一棵树中不相邻; 其他情况均对应一棵非同构的树. □

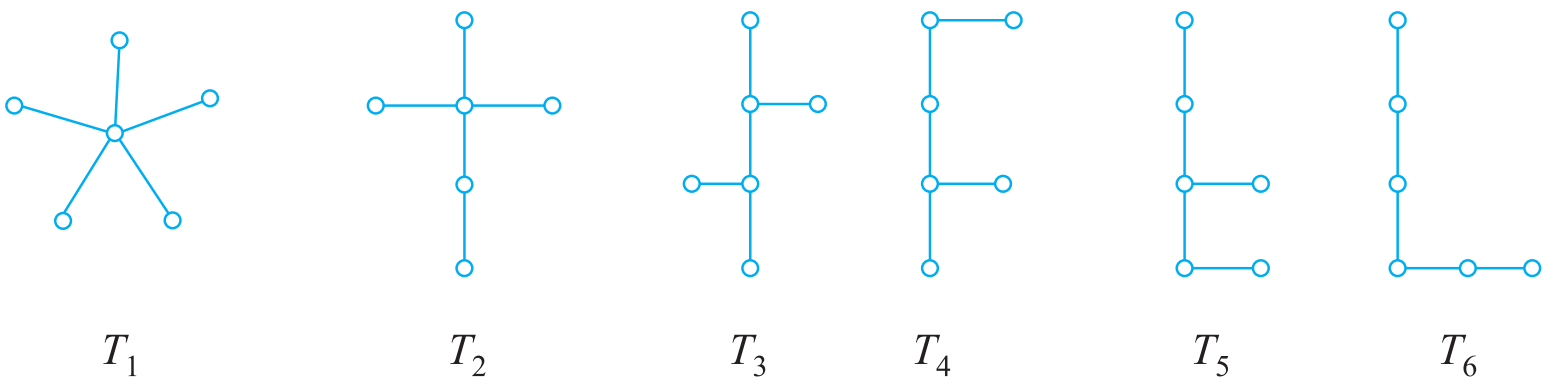


图 1.1.1

常称只有一个分支点，且分支点的度数为 $n-1$ 的 $n(\geqslant 3)$ 阶无向树为星形图，称其唯一的分支点为星心. 图 1.1.1 中的 T_1 是 6 阶星形图.

常称只有一个分支点，且分支点的度数为 $n-1$ 的 $n(\geqslant 3)$ 阶无向树为星形图，称其唯一的分支点为星心. 图 1.1.1 中的 T_1 是 6 阶星形图.

例 1.1.2

饱和碳氢化合物的同分异构体. 饱和碳氢化合物 C_nH_{2n+2} 由 n 个碳原子和 $2n+2$ 个氢原子组成，碳原子是 *4* 价键，氢原子是 *1* 价键. 当 $n=1,2,3,4$ 时，各有多少种可能的饱和碳氢化合物?

常称只有一个分支点，且分支点的度数为 $n-1$ 的 $n(\geqslant 3)$ 阶无向树为星形图，称其唯一的分支点为星心. 图 1.1.1 中的 T_1 是 6 阶星形图.

例 1.1.2

饱和碳氢化合物的同分异构体. 饱和碳氢化合物C_nH_{2n+2} 由 n 个碳原子和 $2n+2$ 个氢原子组成，碳原子是 *4* 价键，氢原子是 *1* 价键. 当 $n=1,2,3,4$ 时，各有多少种可能的饱和碳氢化合物*?*

解

用 4 度顶点代表碳原子，1 度顶点代表氢原子，表示C_nH_{2n+2} 的图的度数列由 n 个 4 和 $2n+2$ 个 1 组成. 它有 $3n+2$ 个顶点，度数之和等于 $4n+(2n+2)$ $=2(3n+1)$，因此是一棵树.

当 $n=1,2,3$ 时，都只有一棵非同构的树；当 $n=4$ 时，有 2 棵非同构的树，如图 1.1.2 所示. 这说明含有 1 个、2 个和 3 个碳原子的饱和碳氢化合物都只有一个，而含有 4 个碳原子的饱和碳氢化合物可能有 2 个. 事实上，含有 1 个、2 个和 3 个碳原子的饱和碳氢化合物分别是甲烷、乙烷和丙烷，而含有 4 个碳原子的饱和碳氢化合物有 2 个同分异构体，分别是丁烷和异丁烷.

例1.1.2解(续)

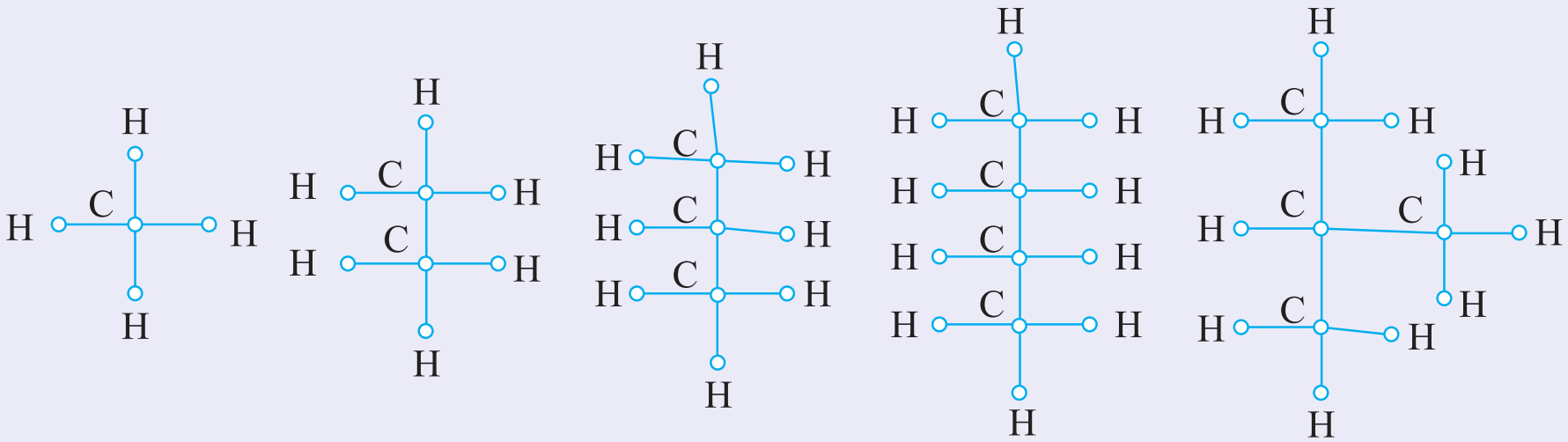


图 1.1.2

7.2 生成树

定义 1.2.1

如果无向图 G 的生成子图 T 是树，那么称 T 是 G 的生成树. 设 T 是 G 的生成树，G 的在 T 中的边称作 T 的树枝，不在 T 中的边称作 T 的弦. 称 T 的所有弦的导出子图为 T 的余树，记作 $\bar{T}$.

7.2 生成树

定义 1.2.1

如果无向图 G 的生成子图 T 是树，那么称 T 是 G 的生成树. 设 T 是 G 的生成树，G 的在 T 中的边称作 T 的树枝，不在 T 中的边称作 T 的弦. 称 T 的所有弦的导出子图为 T 的余树，记作 $\bar{T}$.

注意 $\bar{T}$ 不一定连通，也不一定不含回路. 在图 1.2.1 所示的图中，实边图为该图的一棵生成树 T，余树 $\bar{T}$ 为虚边所示，它不连通，同时含有回路.

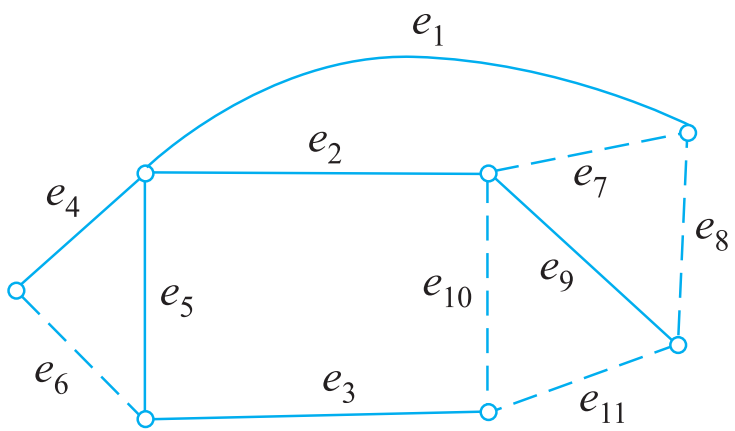


图 1.2.1

定理 1.2.1

无向图 G 有生成树当且仅当 G 是连通图.

定理 1.2.1

无向图 G 有生成树当且仅当 G 是连通图.

证明.

必要性显然. 下面证明充分性. 若 G 中无回路,则 G 为自己的生成树. 若 G 中含圈,任取一个圈,随意地删除圈上的一条边;若仍有圈,再任取一个圈并删去这个圈上的一条边,重复进行,直到最后无圈为止. 最后得到的图无圈(当然无回路)、连通且是 G 的生成子图,因而是 G 的生成树. □

定理 1.2.1

无向图 G 有生成树当且仅当 G 是连通图.

证明.

必要性显然. 下面证明充分性. 若 G 中无回路,则 G 为自己的生成树. 若 G 中含圈,任取一个圈,随意地删除圈上的一条边;若仍有圈,再任取一个圈并删去这个圈上的一条边,重复进行,直到最后无圈为止. 最后得到的图无圈(当然无回路)、连通且是 G 的生成子图,因而是 G 的生成树. □

定理 1.2.1 的证明是构造性证明,这个产生生成树的方法称作 破圈法 .

由定理 1.2.1和树的边数等于顶点数减 1 可以立即得到下述推论.

推论 1.2.1

设 G 为 n 阶 m 条边的无向连通图,则 $m \geqslant n-1$. (习题 5 第 50 题)

定理 1.2.2

设 T 为无向连通图 G 中的一棵生成树，e 为 T 的任意一条弦，则 $T \cup e$ 中存在 G 中只含一条弦 e，其余边均为树枝的圈，而且不同的弦对应的圈也不同.

定理 1.2.2

设 T 为无向连通图 G 中的一棵生成树，e 为 T 的任意一条弦，则 $T \cup e$ 中存在 G 中只含一条弦 e，其余边均为树枝的圈，而且不同的弦对应的圈也不同.

证明.

设 $e = (u, v)$，由定理 1.1.1 可知，在 T 中，u, v 之间存在唯一的路径 $\Gamma_{(u,v)}$，则 $\Gamma_{(u,v)} \cup e$ 满足要求. 不同的弦对应的圈也不同是显然的. □

定理 1.2.2

设 T 为无向连通图 G 中的一棵生成树，e 为 T 的任意一条弦，则 $T \cup e$ 中存在 G 中只含一条弦 e，其余边均为树枝的圈，而且不同的弦对应的圈也不同.

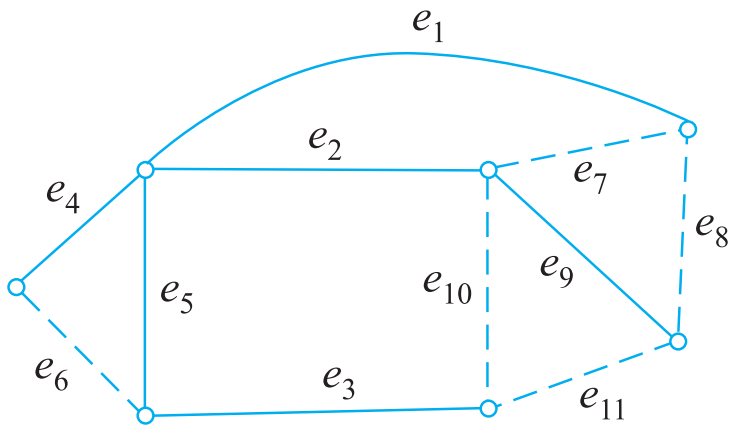
证明.

设 $e = (u, v)$，由定理 1.1.1 可知，在 T 中，u, v 之间存在唯一的路径 $\Gamma_{(u,v)}$，则 $\Gamma_{(u,v)} \cup e$ 满足要求. 不同的弦对应的圈也不同是显然的. □

由定理 1.2.2，可以给出下面的定义.

定义 1.2.2

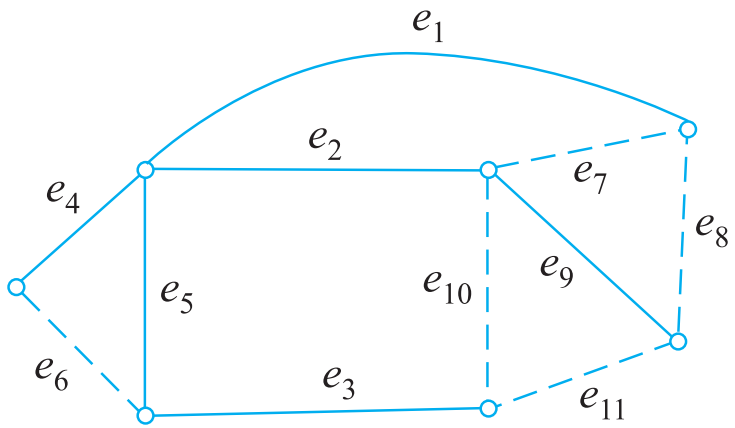
设 T 是 n 阶 m 条边的无向连通图 G 的一棵生成树，设 $e'_1, e'_2, \cdots, e'_{m-n+1}$ 为 T 的弦，$C_r, r = 1, 2, \cdots, m - n + 1,$ 为 T 添加弦 e'_r 产生的 G 中由弦 e'_r 和树枝构成的圈，称 C_r 为 G 的对应弦 e'_r 的基本回路或基本圈. 称 $\{C_1, C_2, \cdots, C_{m-n+1}\}$ 为 G 对应 T 的基本回路系统，称 $m - n + 1$ 为 G 的圈秩，记作 $\xi(G)$.



在上图中，对应生成树的弦 $e_6, e_7, e_8, e_{10}, e_{11}$ 的基本回路分别为

$$
\begin{aligned}
C_1 &= e_6\, e_4\, e_5, \\
C_2 &= e_7\, e_2\, e_1, \\
C_3 &= e_8\, e_9\, e_2\, e_1, \\
C_4 &= e_{10}\, e_3\, e_5\, e_2, \\
C_5 &= e_{11}\, e_3\, e_5\, e_2\, e_9.
\end{aligned}
$$

此图的圈秩为 5，基本回路系统为 $\{C_1, C_2, C_3, C_4, C_5\}$.



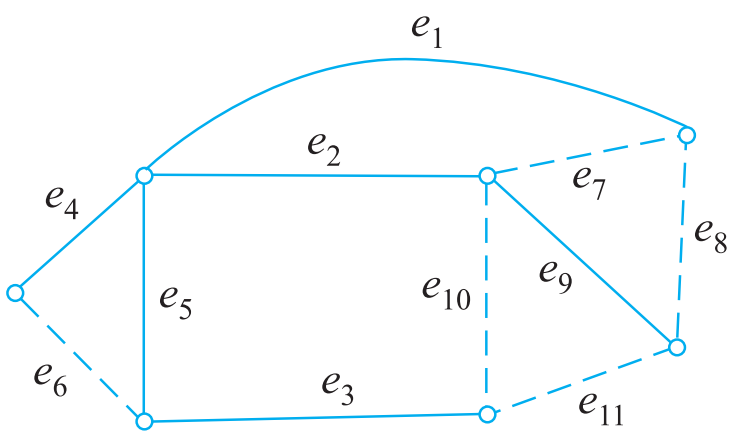
在上图中，对应生成树的弦 $e_6, e_7, e_8, e_{10}, e_{11}$ 的基本回路分别为

$$
\begin{aligned}
C_1 &= e_6\, e_4\, e_5, \\
C_2 &= e_7\, e_2\, e_1, \\
C_3 &= e_8\, e_9\, e_2\, e_1, \\
C_4 &= e_{10}\, e_3\, e_5\, e_2, \\
C_5 &= e_{11}\, e_3\, e_5\, e_2\, e_9.
\end{aligned}
$$

此图的圈秩为 5，基本回路系统为 $\{C_1, C_2, C_3, C_4, C_5\}$.

不难看出，无向连通图 G 的圈秩与生成树的选取无关，但不同生成树对应的基本回路系统可能不同.

广义回路空间



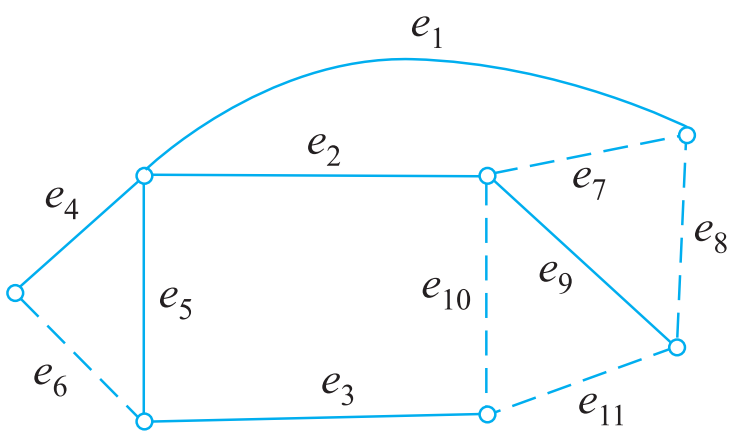
可以证明：任一简单回路都可以表示成基本回路的环和.

例如，在左图中，

$$e_1\, e_4\, e_6\, e_5\, e_2\, e_7 = C_1 \oplus C_2,$$

$$e_1\, e_4\, e_6\, e_3\, e_{11}\, e_8 = C_1 \oplus C_3 \oplus C_5.$$

广义回路空间



可以证明：**任一简单回路都可以表示成基本回路的环和.**

例如，在左图中，

$$e_1\,e_4\,e_6\,e_5\,e_2\,e_7 = C_1 \oplus C_2,$$
$$e_1\,e_4\,e_6\,e_3\,e_{11}\,e_8 = C_1 \oplus C_3 \oplus C_5.$$

无向图中的圈或若干个边不重的圈的并，称作*广义回路*. 规定 $\emptyset$ 为广义回路. 圈和简单回路都是广义回路. 记无向图 G 的广义回路的全体(含 $\emptyset$)为 C^*. 两个广义回路的环和仍是广义回路，即 C^* 中环和运算是封闭的. 在 C^* 上定义数乘运算：

$$0 \cdot C = \emptyset,\ 1 \cdot C = C,$$

易知 C^* 对环和运算与数乘运算构成数域 $F = \{0, 1\}$ 上的 $m - n + 1$ 维线性空间，称作*广义回路空间*. 任一基本回路系统均是它的一个基.

定理 1.2.3

设 T 是连通图 G 的一棵生成树，e 为 T 的树枝，则 G 中存在只含树枝 e，其余边都是弦的割集，且不同的树枝对应的割集也不同.

定理 1.2.3

设 T 是连通图 G 的一棵生成树，e 为 T 的树枝，则 G 中存在只含树枝 e，其余边都是弦的割集，且不同的树枝对应的割集也不同.

证明.

由定理 1.1.1 可知，e 是 T 的桥，因而 $T-e$ 有两个连通分支 T_1 和 T_2，令

$$S_e = \{e' | e' \in E(G) \text{ 且 } e' \text{ 的两个端点分别属于 } V(T_1) \text{ 和 } V(T_2)\}.$$

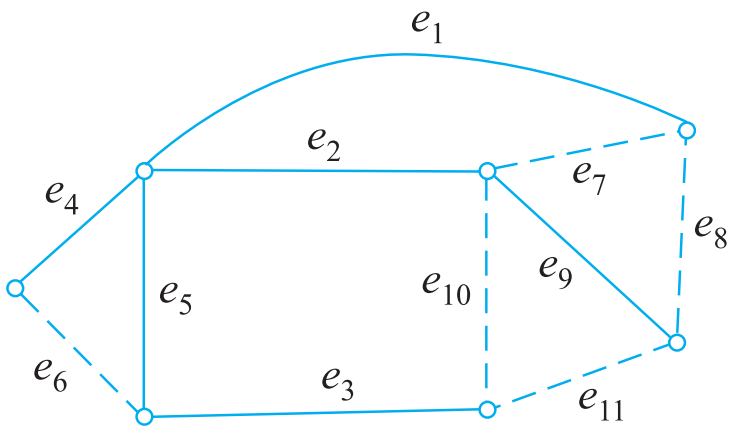
显然，S_e 为 G 的割集，$e \in S_e$ 且 S_e 中除 e 外都是弦. 因为每个割集只含一条树枝，故不同树枝对应的割集是不同的. □

定义 1.2.3

设 T 是 n 阶连通图 G 的一棵生成树，$e_1, e_2, \cdots, e_{n-1}$ 为 T 的树枝，$S_i, i = 1, 2, \cdots, n-1$，是由树枝 e_i 和弦构成的割集，则称 S_i 为 G 的对应树枝 e_i 的基本割集. 称 $\{S_1, S_2, \cdots, S_{n-1}\}$ 为 G 对应 T 的基本割集系统，称 $n-1$ 为 G 的割集秩，记作 $\eta(G)$.

定义 1.2.3

设 T 是 n 阶连通图 G 的一棵生成树，$e_1, e_2, \cdots, e_{n-1}$ 为 T 的树枝，$S_i, i = 1, 2, \cdots, n-1$，是由树枝 e_i 和弦构成的割集，则称 S_i 为 G 的对应树枝 e_i 的基本割集. 称 $\{S_1, S_2, \cdots, S_{n-1}\}$ 为 G 对应 T 的基本割集系统，称 $n-1$ 为 G 的割集秩，记作 $\eta(G)$.



在左图中，对应树枝 $e_1, e_2, e_3, e_4, e_5, e_9$ 的基本割集分别为

$$
\begin{aligned}
S_1 &= \{e_1, e_7, e_8\},\\
S_2 &= \{e_2, e_7, e_8, e_{10}, e_{11}\},\\
S_3 &= \{e_3, e_{10}, e_{11}\},\\
S_4 &= \{e_4, e_6\},\\
S_5 &= \{e_5, e_6, e_{10}, e_{11}\},\\
S_6 &= \{e_9, e_8, e_{11}\}.
\end{aligned}
$$

基本割集系统为 $\{S_1, S_2, S_3, S_4, S_5, S_6\}$，割集秩为 6.

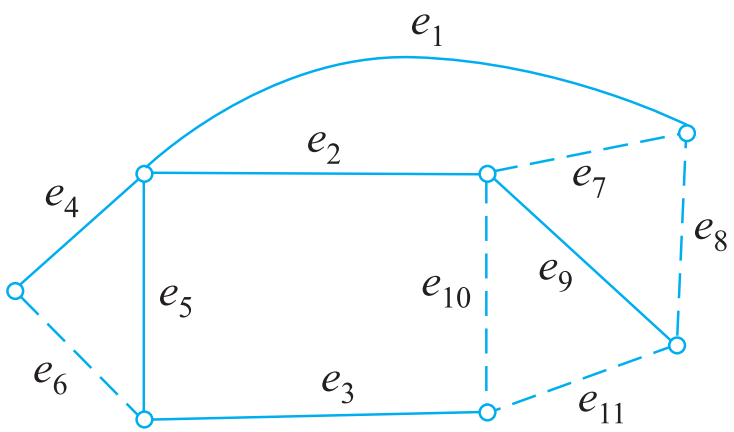
连通图 G 的割集秩 $\eta(G)$ 不因生成树的不同而改变，但不同生成树对应的基本割集系统可能不同.

广义割集空间

设无向图 $G = \langle V, E \rangle$, $\emptyset \neq V_1 \subset V$, 记 V_1 关于 V 的补集为 $\bar{V}_1$, 称 $(V_1, \bar{V}_1) = \{(u, v) | u \in V_1, v \in \bar{V}_1\}$ 为广义割集. 规定 $\emptyset$ 为广义割集. 显然, 割集是广义割集, 但广义割集不一定是割集.

广义割集空间

设无向图 $G=\langle V,E\rangle$,$\emptyset \neq V_1 \subset V$,记 V_1 关于 V 的补集为 $\bar{V}_1$,称 $(V_1,\bar{V}_1)=\{(u,v)|u\in V_1, v\in \bar{V}_1\}$ 为广义割集. 规定 $\emptyset$ 为广义割集. 显然,割集是广义割集,但广义割集不一定是割集.



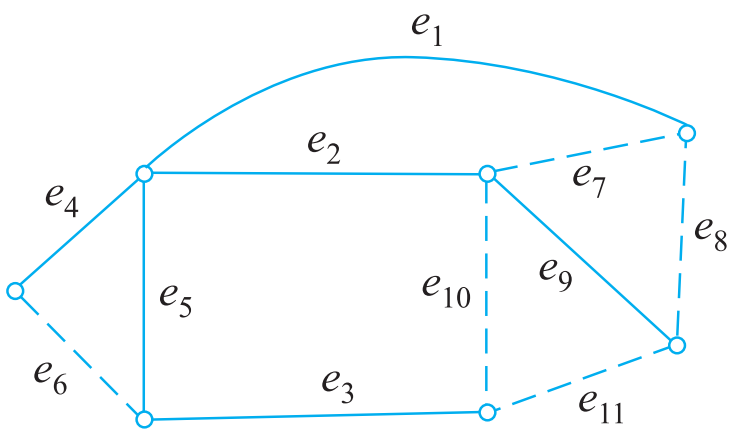
可以证明:任一广义割集都可以表示成基本割集的对称差.

例如,在左图中,

$$\{e_1,e_2,e_3\}=S_1\oplus S_2\oplus S_3,$$
$$\{e_1,e_7,e_9,e_{11}\}=S_1\oplus S_6.$$

广义割集空间

设无向图 $G=\langle V,E\rangle$,$\emptyset\neq V_1\subset V$,记 V_1 关于 V 的补集为 $\bar{V}_1$,称 $(V_1,\bar{V}_1)=\{(u,v)|u\in V_1,v\in\bar{V}_1\}$ 为广义割集. 规定 $\emptyset$ 为广义割集. 显然,割集是广义割集,但广义割集不一定是割集.



可以证明:任一广义割集都可以表示成基本割集的对称差.

例如,在左图中,

$$\{e_1,e_2,e_3\}=S_1\oplus S_2\oplus S_3,$$
$$\{e_1,e_7,e_9,e_{11}\}=S_1\oplus S_6.$$

记 G 的广义割集的全体(含 $\emptyset$)为 S^*. 在 S^* 上定义数乘:

$$0\cdot S=\emptyset,\ 1\cdot S=S,$$

则 S^* 关于对称差运算和数乘运算构成数域 $F=\{0,1\}$ 上的 $n-1$ 维线性空间,称作广义割集空间. 任一基本割集系统均是它的一个基.

连通带权图中的最小生成树

定义 1.2.4

设无向连通带权图 $G = \langle V, E, W \rangle$, T 是 G 的一棵生成树, T 的各边权之和称为 T 的权, 记作 $W(T)$. G 的所有生成树中权最小的生成树称为 G 的最小生成树.

连通带权图中的最小生成树

定义 1.2.4

设无向连通带权图 $G=\langle V,E,W\rangle$, T 是 G 的一棵生成树, T 的各边权之和称为 T 的权, 记作 $W(T)$. G 的所有生成树中权最小的生成树称为 G 的最小生成树.

求最小生成树已经有许多种算法, 这里介绍避圈法(Kruskal 算法).

设 n 阶无向连通带权图 $G=\langle V,E,W\rangle$ 有 m 条边. 不妨设 G 中没有环(否则, 可以将所有的环先删去), 将 m 条边按权从小到大顺序排列, 设为 $e_1,e_2,\cdots,e_m$.

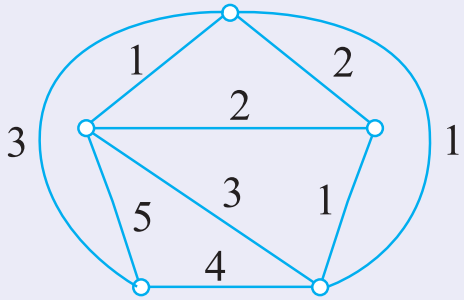
将 e_1 加入 T 中, 然后依次检查 $e_2,e_3,\cdots,e_m$.

若 $e_j, j\geqslant 2$, 与已在 T 中的边不能构成回路, 则将 e_j 加入 T 中, 否则放弃 e_j.

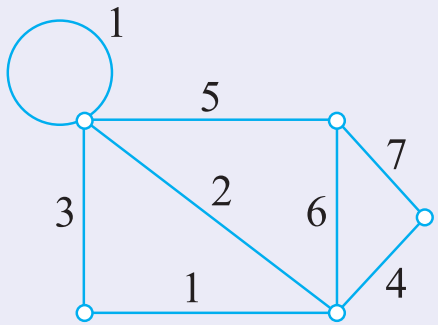
算法停止时得到的 T 为 G 的最小生成树(证明略).

例 1.2.1

求下图所示的两个图中的最小生成树.



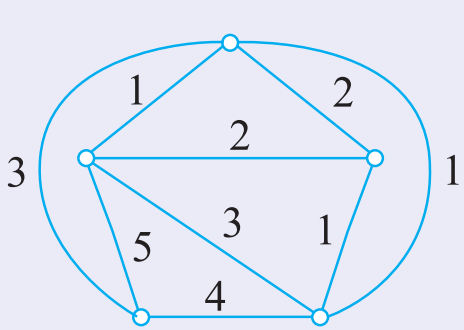
(a)



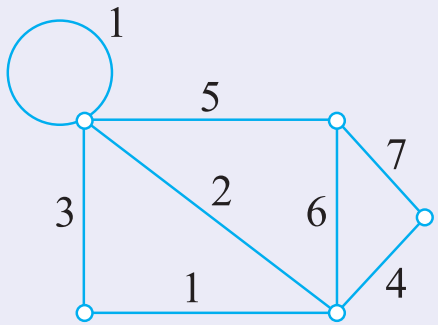
(b)

例 1.2.1

求下图所示的两个图中的最小生成树.



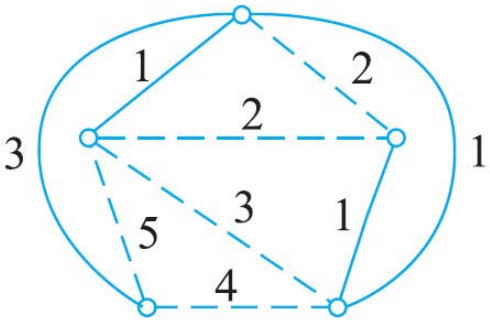
(a)



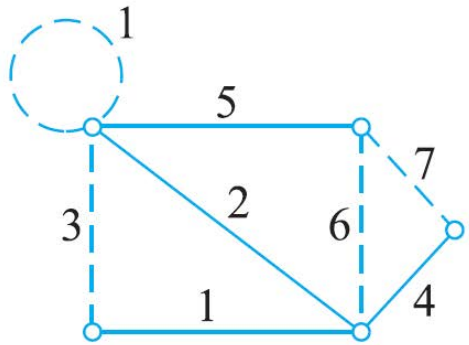
(b)

解

用避圈法,求出的最小生成树如下图中实线所示,它们的权分别为 6 和 12.



(a)



(b)

例 1.2.2

单链聚类. 设有一组离散数据 $D=\{a_1,a_2,\cdots,a_n\}$, D 上定义了一个相似度函数 d. 对于任何两个数据 $a_i,a_j\in D$, a_i 与 a_j 的相似度函数的值为 $d(i,j)$, 通常取 $d(i,j)\in[0,1]$, 并且 d 具有对称性质, 即 $d(i,j)=d(j,i)$. 给定正整数 k, $1<k<n$, D 的一个 k 聚类 是 D 的一个 k 划分 $\pi=\{C_1,C_2,\cdots,C_k\}$.

对 $\forall C_s,C_t\in\pi$, 定义它们之间的距离 $D(C_s,C_t)=\min\{d(i,j)|a_i\in C_s,a_j\in C_t\}$.

k 聚类 π 的最小间隔 $D(\pi)=\min\{D(C_s,C_t)|C_s,C_t\in\pi,1\leqslant s<t\leqslant k\}$.

一个自然的问题是: 给定数据集 D 和 D 上的相似度函数 d 以及正整数 k, 如何求使得 $D(\pi)$ 达到最大值的 k 聚类 π?

例 1.2.2

单链聚类. 设有一组离散数据 $D=\{a_1,a_2,\cdots,a_n\}$, D 上定义了一个相似度函数 d. 对于任何两个数据 $a_i,a_j\in D$, a_i 与 a_j 的相似度函数的值为 $d(i,j)$, 通常取 $d(i,j)\in[0,1]$, 并且 d 具有对称性质, 即 $d(i,j)=d(j,i)$. 给定正整数 k, $1<k<n$, D 的一个 k 聚类 是 D 的一个 k 划分 $\pi=\{C_1,C_2,\cdots,C_k\}$.

对 $\forall C_s,C_t\in\pi$, 定义它们之间的距离 $D(C_s,C_t)=\min\{d(i,j)|a_i\in C_s,a_j\in C_t\}$.

k 聚类 π 的最小间隔 $D(\pi)=\min\{D(C_s,C_t)|C_s,C_t\in\pi,1\leqslant s<t\leqslant k\}$.

一个自然的问题是: 给定数据集 D 和 D 上的相似度函数 d 以及正整数 k, 如何求使得 $D(\pi)$ 达到最大值的 k 聚类 π?

可用 *Kruskal* 算法解决该问题. 定义带权完全图 $G=\langle V,E,W\rangle$, 其中 $V=\{1,2,\cdots,n\}$, 对 $\forall i\neq j\in V$, 边 (i,j) 的权为 $d(i,j)$. 根据算法, 先将边按照权从小到大的顺序排序为 $e_1,e_2,\cdots,e_{n(n-1)/2}$. 初始 T 中没有边, 是由 n 个孤立顶点构成的森林, 即 T 有 n 个连通分支. 接着, 依次按照权从小到大的顺序考察 G 的每条边, 只要不构成圈就把它加到 T 中. 在加入边的过程中计数 T 的连通分支个数. 直到 T 恰好含有 k 个连通分支时算法停止. 这时所得到的 k 个连通分支恰好就是所求聚类的 k 个子类 $C_1,C_2,\cdots,C_k$, 它的最小间隔达到最大.

7.3 根树及其应用

定义 1.3.1

若有向图的基图是无向树，则称这个有向图为 *有向树*.

一个顶点的入度为 *0*、其余顶点的入度为 *1* 的有向树称作 *根树*.

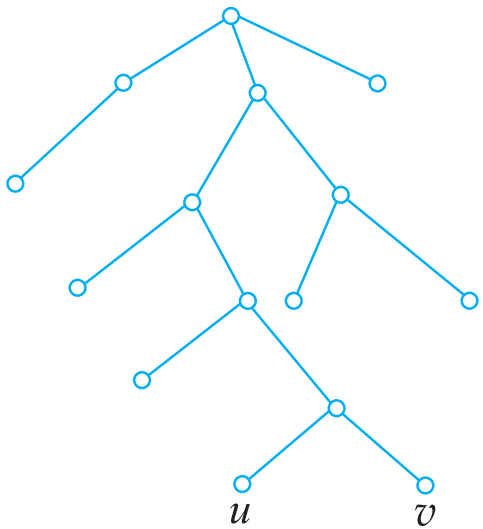
入度为 *0* 的顶点称作 *树根*，入度为 *1* 出度为 *0* 的顶点称作 *树叶*，入度为 *1* 出度不为 *0* 的顶点称作 *内点*，内点和树根统称作 *分支点*.

从树根到任意顶点 v 的路径的长度(即路径中的边数)称作 v 的 *层数*，所有顶点的最大层数称作 *树高*.

7.3 根树及其应用

定义 1.3.1

若有向图的基图是无向树，则称这个有向图为*有向树*.
一个顶点的入度为 0、其余顶点的入度为 1 的有向树称作*根树*.
入度为 0 的顶点称作*树根*，入度为 1 出度为 0 的顶点称作*树叶*，入度为 1 出度不为 0 的顶点称作*内点*，内点和树根统称作*分支点*.
从树根到任意顶点 v 的路径的长度(即路径中的边数)称作 v 的*层数*，所有顶点的最大层数称作*树高*.



在画根树时通常将树根画在最上方，有向边的方向向下或斜下方，并省去各边上的箭头，如左图所示. 在这棵根树中，有 8 片树叶、6 个内点、7 个分支点，高度为 5，在树叶 u 和 v 处达到.

家族树

常将根树看成 家族树 ,家族中成员之间的关系由下面的定义给出.

定义 1.3.2

设 T 为一棵非平凡的根树,$\forall v_i, v_j \in V(T)$,若 v_i 可达 v_j,则称 v_i 为 v_j 的 祖先 , v_j 为 v_i 的 后代 ;若 v_i 邻接到 v_j,即 $\langle v_i, v_j \rangle \in E(T)$,则称 v_i 为 v_j 的 父亲 ,而 v_j 为 v_i 的 儿子 . 若 v_j, v_k 的父亲相同,则称 v_j 与 v_k 是 兄弟 .

家族树

常将根树看成 家族树 ,家族中成员之间的关系由下面的定义给出.

定义 1.3.2

设 T 为一棵非平凡的根树,$\forall v_i, v_j \in V(T)$,若 v_i 可达 v_j,则称 v_i 为 v_j 的 祖先 , v_j 为 v_i 的 后代 ;若 v_i 邻接到 v_j,即 $\langle v_i, v_j \rangle \in E(T)$,则称 v_i 为 v_j 的 父亲 ,而 v_j 为 v_i 的 儿子 . 若 v_j, v_k 的父亲相同,则称 v_j 与 v_k 是 兄弟 .

设 T 为根树,若将 T 中层数相同的顶点都标定次序,则称 T 为 有序树 .

根据根树 T 中每个分支点儿子数以及是否有序,可以将根树分成下列各类.

1. 若 T 的每个分支点至多有 r 个儿子,则称 T 为 r 叉树 ;又若 r 叉树是有序的,则称它为 r 叉有序树 .
2. 若 T 的每个分支点都恰好有 r 个儿子,则称 T 为 r 叉正则树 ;又若 T 是有序的,则称它为 r 叉正则有序树 .
3. 若 T 是 r 叉正则树,且每个树叶的层数均为树高,则称 T 为 r 叉完全正则树 ;又若 T 是有序的,则称它为 r 叉完全正则有序树 .

根子树

定义 1.3.3

设 T 为一棵根树,$\forall v \in V(T)$,称 v 及其后代的导出子图 T_v 为 T 的以 v 为根的*根子树*.

2 叉正则有序树的每个分支点的两个儿子导出的根子树分别称作该分支点的*左子树*和*右子树*.

在所有的 r 叉树中,最常用的是 2 叉树. 下面介绍 2 叉树的应用.

根子树

定义 1.3.3

设 T 为一棵根树，$\forall v \in V(T)$，称 v 及其后代的导出子图 T_v 为 T 的以 v 为根的*根子树*.

2 叉正则有序树的每个分支点的两个儿子导出的根子树分别称作该分支点的*左子树*和*右子树*.

在所有的 r 叉树中，最常用的是 2 叉树. 下面介绍 2 叉树的应用.

定义 1.3.4

设 2 叉树 T 有 t 片树叶 $v_1, v_2, \cdots, v_t$，权分别为 $w_1, w_2, \cdots, w_t$，称

$$W(T) = \sum_{i=1}^{t} w_i l(v_i)$$

为 T 的*权*，其中 $l(v_i)$ 是 v_i 的层数. 在所有有 t 片树叶，带权 $w_1, w_2, \cdots, w_t$ 的 2 叉树中，权最小的 2 叉树称作*最优 2 叉树*.

2 叉树

图 1.3.1 所示的 3 棵树 T_1, T_2, T_3 都是带权为 $2, 2, 3, 3, 5$ 的 2 叉树. 它们的权分别为

$$W(T_1) = 2 \times 2 + 2 \times 2 + 3 \times 3 + 5 \times 3 + 3 \times 2 = 38,$$
$$W(T_2) = 3 \times 4 + 5 \times 4 + 3 \times 3 + 2 \times 2 + 2 \times 1 = 47,$$
$$W(T_3) = 3 \times 3 + 3 \times 3 + 5 \times 2 + 2 \times 2 + 2 \times 2 = 36.$$

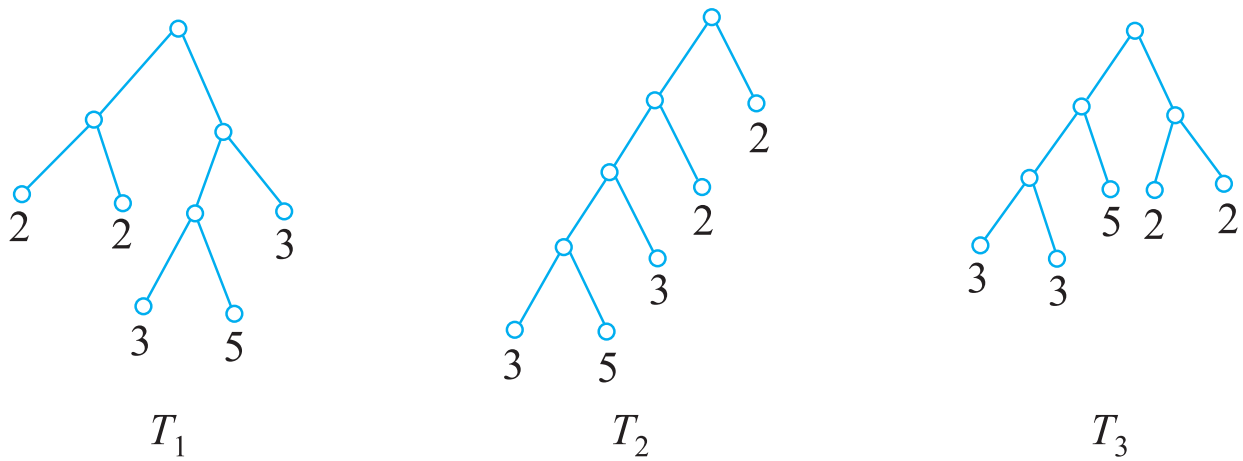


图 1.3.1

Huffman 算法

下面介绍求最优 2 叉树的算法——Huffman 算法.

Huffman 算法

给定实数 $w_1, w_2, \cdots, w_t$.

---

1 作 t 片树叶,分别以 $w_1, w_2, \cdots, w_t$ 为权.

2 在所有入度为 0 的顶点(不一定是树叶)中选出两个权最小的顶点,添加一个新分支点,它以这 2 个顶点为儿子,其权等于这 2 个儿子的权之和.

3 重复 2,直到只有 1 个入度为 0 的顶点为止.

$W(T)$ 等于所有分支点的权之和.

例 1.3.1

求带权 $2, 2, 3, 3, 5$ 的最优 2 叉树.

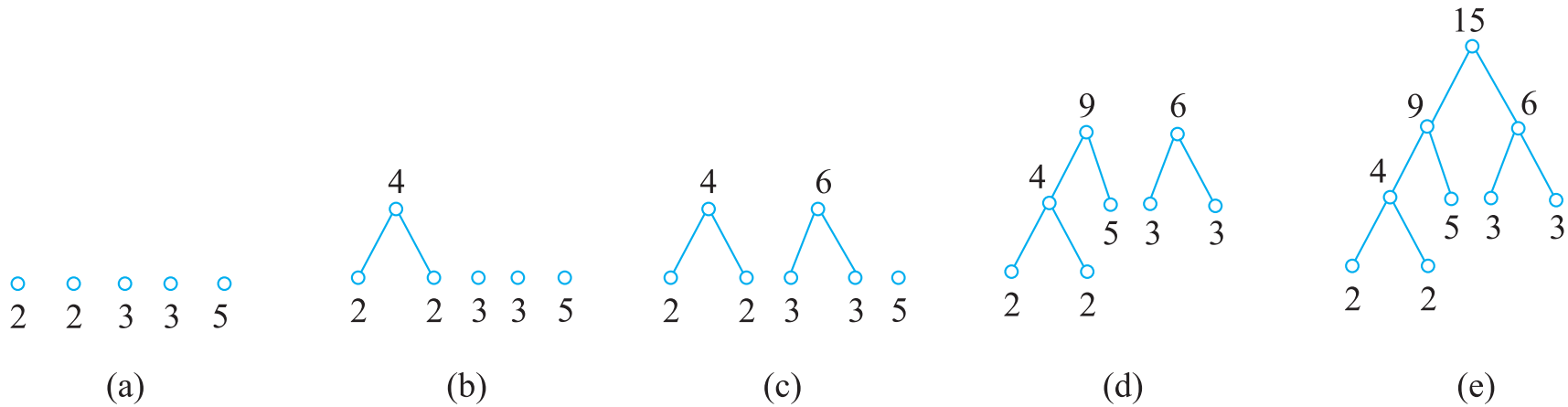


图 1.3.2

解

图 1.3.2 给出用 Huffman 算法计算最优树的过程. (e) 为最优树, $W(T) = 34$. 这表明图 1.3.1 所示的 3 棵树都不是最优树. □

在通信中，常用二进制编码表示符号. 例如，可用长为 2 的二进制编码 $00, 01, 10, 11$ 分别表示 A, B, C, D. 称这种表示法为等长码表示法. 若在传输中，这些符号出现的频率大体相同，用等长码表示是很好的方法. 但当它们出现的频率相差悬殊时，为了节省二进制数位，以达到提高效率的目的，需要采用非等长的编码.

在通信中，常用二进制编码表示符号. 例如，可用长为 2 的二进制编码 $00, 01, 10, 11$ 分别表示 A, B, C, D. 称这种表示法为等长码表示法. 若在传输中，这些符号出现的频率大体相同，用等长码表示是很好的方法. 但当它们出现的频率相差悬殊时，为了节省二进制数位，以达到提高效率的目的，需要采用非等长的编码.

定义 1.3.5

设 $\alpha_1\alpha_2\cdots\alpha_{n-1}\alpha_n$ 是长为 n 的符号串，称其子串 $\alpha_1, \alpha_1\alpha_2, \cdots, \alpha_1\alpha_2\cdots\alpha_n$ 为该符号串的*前缀*. 设 $A = \{\beta_1, \beta_2, \cdots, \beta_m\}$ 是一个符号串集合，若 A 的任意两个符号串都互不为前缀，则称 A 为*前缀码*. 由 0-1 符号串构成的前缀码称作*2 元前缀码*.

在通信中，常用二进制编码表示符号. 例如，可用长为 2 的二进制编码 $00, 01, 10, 11$ 分别表示 A, B, C, D. 称这种表示法为等长码表示法. 若在传输中，这些符号出现的频率大体相同，用等长码表示是很好的方法. 但当它们出现的频率相差悬殊时，为了节省二进制数位，以达到提高效率的目的，需要采用非等长的编码.

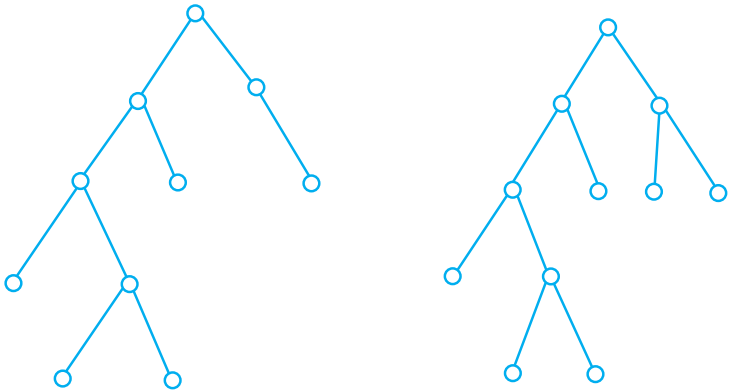
定义 1.3.5

设 $\alpha_1\alpha_2\cdots\alpha_{n-1}\alpha_n$ 是长为 n 的符号串，称其子串 $\alpha_1, \alpha_1\alpha_2, \cdots, \alpha_1\alpha_2\cdots\alpha_n$ 为该符号串的*前缀*. 设 $A = \{\beta_1, \beta_2, \cdots, \beta_m\}$ 是一个符号串集合，若 A 的任意两个符号串都互不为前缀，则称 A 为*前缀码*. 由 0-1 符号串构成的前缀码称作*2 元前缀码*.

可以用 2 叉树产生 2 元前缀码. 设 T 是具有 n 片树叶的 2 叉树，则 T 的每个分支点有 1 或 2 个儿子. 设 v 为 T 的分支点，若 v 有两个儿子，在由 v 引出的两条边上，左边的标 0，右边的标 1. 若 v 只有一个儿子，由它引出的边可以标 0，也可以标 1. 设 v_i 是 T 的任意一片树叶，从树根到 v_i 的通路上各边的标号(0 或 1)按照通路上边的顺序组成的符号串放在 v_i 处，t 片树叶的 t 个符号串组成一个 2 元码. 由做法可知，树叶 v_i 处的符号串的前缀均在从树根到 v_i 的通路上的顶点(不含 v_i)处达到，因而所得符号串集合必为前缀码. 若 T 是 2 叉正则树，则由 T 产生的前缀码是唯一的.

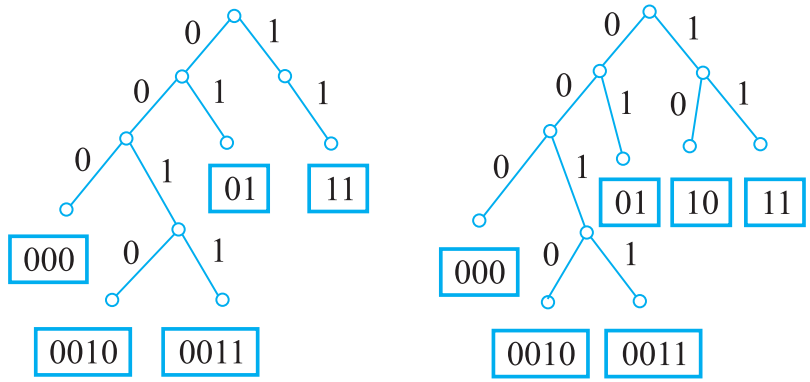
例 1.3.2

求图 *1.3.3* 所示的两棵 *2* 叉树所产生的 *2* 元前缀码.



(a) (b)

图 1.3.3

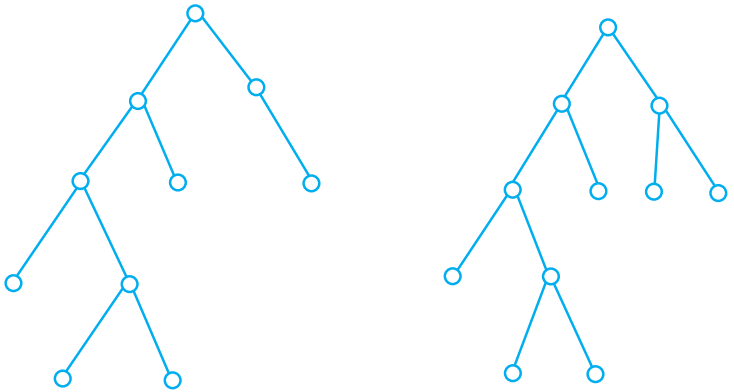


(a) (b)

图 1.3.4

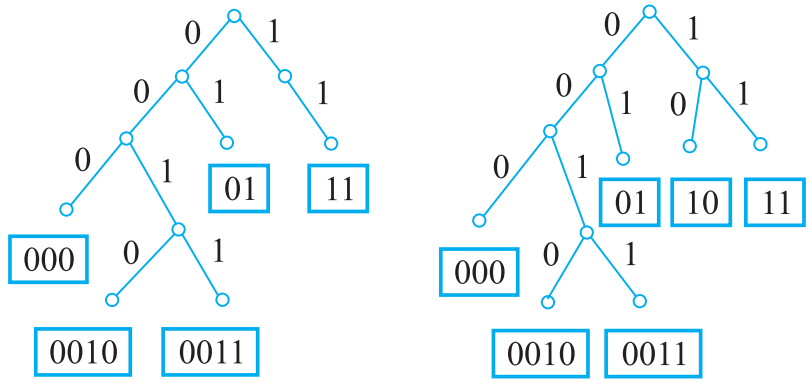
例 1.3.2

求图 *1.3.3* 所示的两棵 *2* 叉树所产生的 *2* 元前缀码.



(a) (b)

图 1.3.3



(a) (b)

图 1.3.4

解

图 1.3.3(a) 是 2 叉树,但不是正则的. 将每个分支点引出的两条边分别标 0 和 1,若将树根右儿子引出的边标 1,如图 1.3.4(a) 所示,产生的前缀码为 $\{11, 01, 000, 0010, 0011\}$. 若将树根右儿子引出的边标 0,则产生前缀码为 $\{00, 10, , 111, 1100, 1101\}$. 图 1.3.3(b) 是 2 叉正则树,它只能产生唯一的前缀码,标定后的 2 叉正则树如图 1.3.4(b) 所示,前缀码为 $\{01, 10, 11, 000, 0010, 0011\}$.

上面产生的任一个前缀码都可以用来传输 5 个符号，如 A, B, C, D, E. 但当在文本中这些字母出现频率不同时，传输这个文本所用的二进制位数是不同的. 设共有 t 个符号，用树叶 v_i 处的二进制串表示的符号出现的频率为 c_i，v_i 处的二进制串的长度等于 v_i 的层数 $l(v_i)$，因而传输 m 个符号使用的二进制位数为 $m\sum_{i=1}^{t} c_i l(v_i)$. 显然，用以各个符号出现的频率为权的最优 2 叉树产生的前缀码所用的二进制位数最少. 称这个由最优 2 叉树产生的前缀码为 最佳前缀码.

上面产生的任一个前缀码都可以用来传输 5 个符号，如 A, B, C, D, E. 但当在文本中这些字母出现频率不同时，传输这个文本所用的二进制位数是不同的. 设共有 t 个符号，用树叶 v_i 处的二进制串表示的符号出现的频率为 c_i，v_i 处的二进制串的长度等于 v_i 的层数 $l(v_i)$，因而传输 m 个符号使用的二进制位数为 $m\sum_{i=1}^{t} c_i l(v_i)$. 显然，用以各个符号出现的频率为权的最优 2 叉树产生的前缀码所用的二进制位数最少. 称这个由最优 2 叉树产生的前缀码为最佳前缀码.

例 1.3.3

在通信中，八进制数字出现的频率如下.

0:	*25%*	*1:*	*20%*
2:	*15%*	*3:*	*10%*
4:	*10%*	*5:*	*10%*
6:	*5%*	*7:*	*5%*

求传输它们的最佳前缀码，并求传输 $10^n, n \geqslant 2,$ 个按上述比例出现的八进制数字需要多少个二进制数字？若是用等长的(长为 *3*)的码字传输需要多少个二进制数字*?*

例1.3.3(续)

解

用 100 个八进制数字中各数字出现的个数，即以 100 乘各频率为权，用 Huffman 算法求最优 2 叉树，如图 1.3.5(a) 所示. 它产生的最佳前缀码为

$$01 \quad 传 \quad 0 \qquad\qquad 11 \quad 传 \quad 1$$
$$001 \quad 传 \quad 2 \qquad\qquad 100 \quad 传 \quad 3$$
$$101 \quad 传 \quad 4 \qquad\qquad 0001 \quad 传 \quad 5$$
$$00000 \quad 传 \quad 6 \qquad\qquad 00001 \quad 传 \quad 7$$

例1.3.3(续)

解

用 100 个八进制数字中各数字出现的个数,即以 100 乘各频率为权,用 Huffman 算法求最优 2 叉树,如图 1.3.5(a) 所示. 它产生的最佳前缀码为

01	传	0	11	传	1
001	传	2	100	传	3
101	传	4	0001	传	5
00000	传	6	00001	传	7

设图 1.3.5(a) 中树为 T,传输 100 个按题中给定的频率出现的八进制数字所用二进制数字个数等于 $W(T)$,它等于各分支点权之和:

$$W(T) = 10 + 20 + 35 + 60 + 100 + 40 + 20 = 285.$$

传输 10^n 个按题中给定频率出现的八进制数字需要 $10^{n-2} \times 285 = 2.85 \times 10^n$ 个二进制数字. 而用长为 3 的 0,1 组成的符号串传输 10^n 个八进制数字(如 000 传 0,001 传 1,$\cdots$,111 传 7)要用 3×10^n 个二进制数字. □

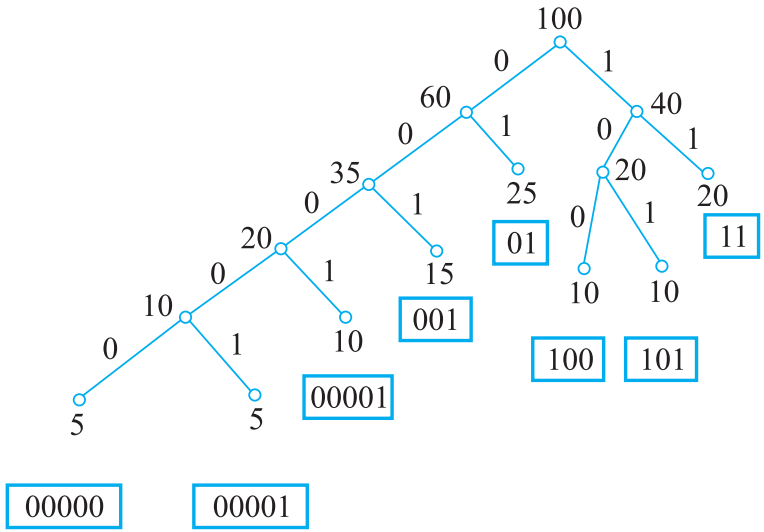
最后还要说明一点，最佳前缀码并不唯一. 由于每一步选择两个最小的权的选法可能不唯一，而且两个权对应的顶点所放的左右位置也可以不同，画出的最优树可能不同. 当然，它们的权相等，都是最优树.

图 1.3.5(b) 所示的 2 叉正则树也是例 1.3.3 对应的最优树，其权等于

$$10 + 20 + 40 + 100 + 60 + 35 + 20 = 285.$$

与图 1.3.5(a) 中的权相等. 各数字对应的编码如下：

11	传	0	01	传	1	101	传	2	001	传	3
1000	传	4	1001	传	5	0000	传	6	0001	传	7



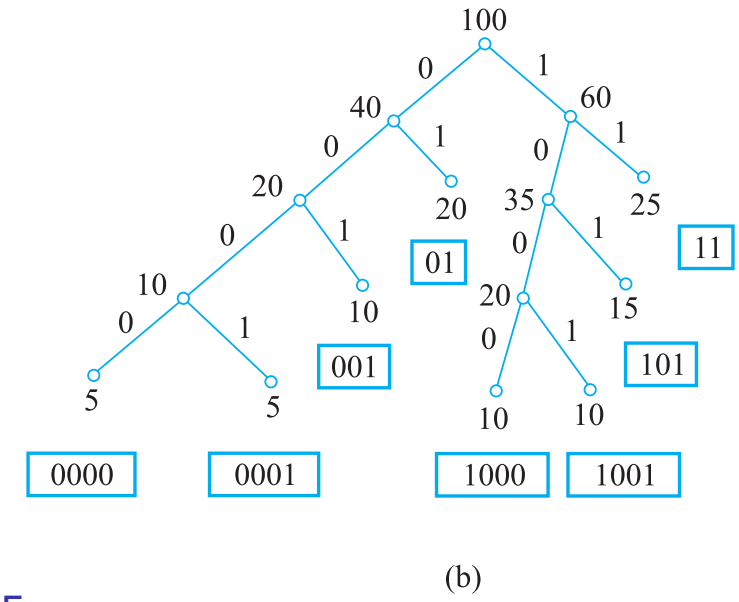


图 1.3.5

2 叉树的周游

对一棵根树的每个顶点都访问一次且仅一次称作 行遍 或 周游 一棵树.

对于 2 叉有序正则树有以下 3 种周游方式.

1. 中序行遍法. 访问的次序为: 左子树、树根、右子树.
2. 前序行遍法. 访问的次序为: 树根、左子树、右子树.
3. 后序行遍法. 访问的次序为: 左子树、右子树、树根.

2 叉树的周游

对一棵根树的每个顶点都访问一次且仅一次称作 行遍 或 周游 一棵树.
对于 2 叉有序正则树有以下 3 种周游方式.

1. 中序行遍法. 访问的次序为: 左子树、树根、右子树.
2. 前序行遍法. 访问的次序为: 树根、左子树、右子树.
3. 后序行遍法. 访问的次序为: 左子树、右子树、树根.

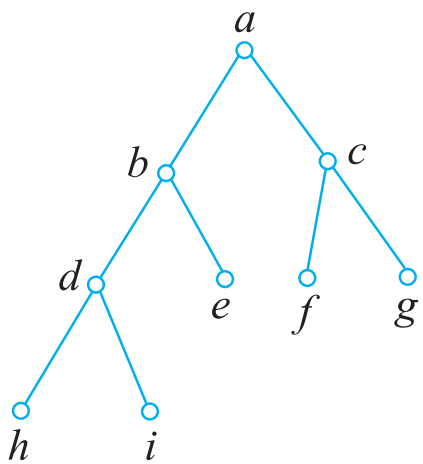


图 1.3.6

对图 1.3.6 所示的 2 叉有序正则树 T, 按照中序、前序、后序行遍的周游结果如下.

$$((h\,\underline{d}\,i)\,\underline{b}\,e)\,\underline{a}\,(f\,\underline{c}\,g),$$
$$\underline{a}\,(\underline{b}\,(\underline{d}\,h\,i)\,e)\,(\underline{c}\,f\,g),$$
$$((h\,i\,\underline{d})\,e\,\underline{b})\,(f\,g\,\underline{c})\,\underline{a},$$

上式中 $\underline{v}$ 表示 v 为根子树的树根.

利用 2 叉有序正则树可以表示四则运算的算式，然后根据不同的访问方法得到不同的算法.

用 2 叉有序正则树表示算式的方法如下：参加运算的数都放在树叶上，然后按照运算的顺序依次将运算符 $(+, -, *, \div)$ 放在分支点上，每个分支点表示一个运算，同时也表示这个运算的结果，它以它的两个运算对象为儿子，并规定被除数、被减数放在左边.

利用 2 叉有序正则树可以表示四则运算的算式，然后根据不同的访问方法得到不同的算法.

用 2 叉有序正则树表示算式的方法如下：参加运算的数都放在树叶上，然后按照运算的顺序依次将运算符 $(+,-,*,\div)$ 放在分支点上，每个分支点表示一个运算，同时也表示这个运算的结果，它以它的两个运算对象为儿子，并规定被除数、被减数放在左边.

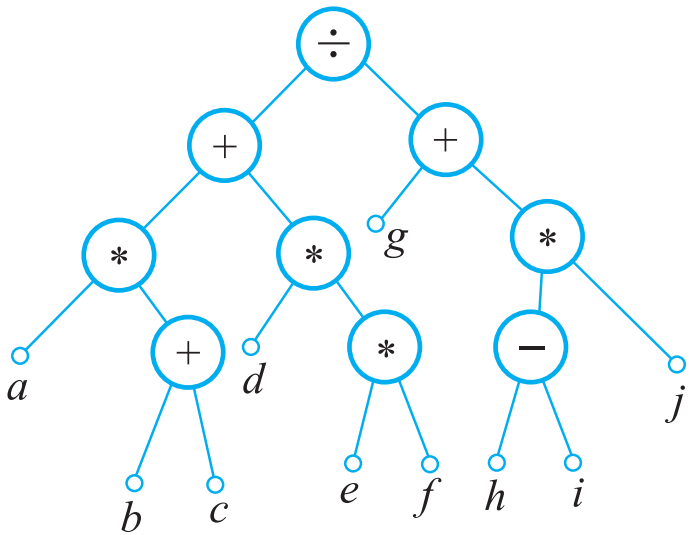


图 1.3.7

例 1.3.4

1. 用 2 叉有序正则树表示以下算式.

$$(a*(b+c)+d*e*f)\div(g+(h-i)*j).$$

2. 用 3 种行遍法访问这棵 2 叉树，写出访问结果.

解

(1) 表示算式的 2 叉树如图 1.3.7 所示.

例1.3.4(续)

(2) 中序行遍法访问结果为:

$$((a*(b+c))+(d*(e*f)))\div(g+((h-i)*j)). \tag{1.3.1}$$

前序行遍法访问结果为:

$$\div(+(*a(+bc))(*d(*ef)))(+g(*(-hi)j)). \tag{1.3.2}$$

后序行遍法访问结果为:

$$((a(bc+)*)(d(ef*)*)+)(g((hi-)j*)+)\div. \tag{1.3.3}$$

在式 (1.3.1) 中，利用四则运算规则省去一些括号，得到原算式，所以中序行遍法访问结果是还原算式.

消去式 (1.3.2) 中的全部括号，得

$$\div + * a + bc * d * ef + g * - hij. \tag{1.3.4}$$

对式(1.3.4)的运算规则为：从右到左每个运算符对它后面紧邻的两个数进行运算. 在这种算法中，由于运算符在它的两个运算对象之前，所以称作 前缀符号法 或 波兰符号法 .

在式 (1.3.1) 中，利用四则运算规则省去一些括号，得到原算式，所以中序行遍法访问结果是还原算式.

消去式 (1.3.2) 中的全部括号，得

$$\div + * a + bc * d * ef + g * - hij. \tag{1.3.4}$$

对式(1.3.4)的运算规则为：从右到左每个运算符对它后面紧邻的两个数进行运算. 在这种算法中，由于运算符在它的两个运算对象之前，所以称作 *前缀符号法* 或 *波兰符号法*.

消去式 (1.3.3) 中的全部括号，得

$$abc + *def * * + ghi - j * + \div . \tag{1.3.5}$$

对式 (1.3.5) 的运算规则为：从左到右每个运算符对它前面紧邻的两个数进行运算. 由于运算符在它的两个运算对象之后，所以称此种算法为 *后缀符号法* 或 *逆波兰符号法*.