Chapter 7
树

Discrete Mathematics

November 29, 2011

黄正华, 数学与统计学院, 武汉大学

7.1

Contents

1 树与生成树

树与生成树

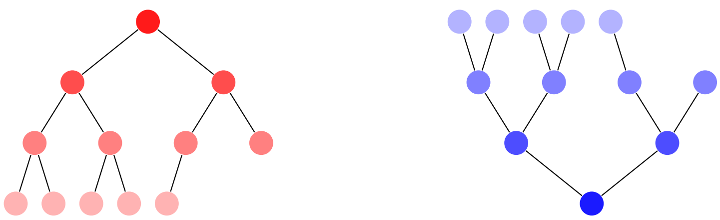
1. 树的定义
2. 生成树
3. 最小生成树
4. Kruskal 算法

树是图论中重要的概念之一, 在计算机科学中有广泛的运用. 7.3

树的定义

Definition 1. 一个连通且无回路的无向图称为树 (tree).

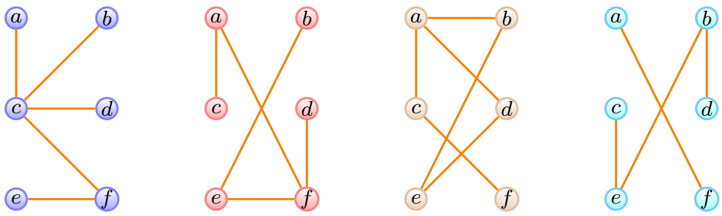
- 树中度数为 1 的结点叫树叶 (leave);
- 度数大于 1 的结点叫分枝点 (branched node) 或内点.
- 如果一个无向图的每个连通分支是树, 则称为森林 (forest).



7.4

树的定义

如图,



(a) G_1 (b) G_2 (c) G_3 (d) G_4

- G_1 和 G_2 是树, 它们都是没有回路的简单图.
- G_3 不是树, 因为结点 a, b, e, d 构成回路.
- G_4 不是树, 因为它不连通.

7.5

Theorem 2. *给定图 T, 下列关于树的定义是等价的:*

1. *无回路的连通图.*
2. *无回路且 $e = v - 1$. (其中 e 为边数, v 为结点数.)*
3. *连通且 $e = v - 1$.*
4. *无回路, 但增加一条边, 得到且仅得到一个回路.*
5. *连通, 但删除一条边则不连通.*
6. *每对结点间有且仅有一条路.*

7.6

给定图 T, 下列关于树的定义是等价的:

1. 无回路的连通图.
2. 无回路且 $e = v - 1$. (其中 e 为边数, v 为结点数.)

证: 由 (1) 证 (2). 归纳法证明.

当 $v = 2$ 时, 因 T 无回路且连通, 则 $e = 1$, 所以 $e = v - 1$ 成立.

设 $v = k - 1$ 时命题成立. 当 $v = k$ 时, 因 T 无回路且连通, 因而至少有一个度数为 1 的结点.

否则, 因各点皆连通且度数大于等于 2. 从某结点 v_i 出发, 可达另一个结点 v_j, 再继续, 可经由一些结点后返回某结点 v_i. 这样就产生了回路. 与假设矛盾. 故至少有一个度数为 1 的结点.

删除该结点及其关联的一条边得 $k - 1$ 个结点的子图 T', 它仍是连通的, 且 $e' = v' - 1$, 即 $e - 1 = (v - 1) - 1$, 整理得 $e = v - 1$. □

7.7

给定图 T, 下列关于树的定义是等价的:

1. 无回路的连通图.
2. 无回路且 $e = v - 1$. (其中 e 为边数, v 为结点数.)
3. 连通且 $e = v - 1$.
4. 无回路, 但增加一条边, 得到且仅得到一个回路.
5. 连通, 但删除一条边则不连通.
6. 每对结点间有且仅有一条路.

证: 由 (6) 证 (1). 每对结点间有路, 则 T 连通.

因每对结点间仅有一条路, 则 T 无回路. 否则, 回路上的两点之间至少有两条道路, 与所设矛盾.

故 T 是无回路的连通图. □ 7.8

树的概念是亚瑟 · 凯莱 [1] 提出的.

饱和碳氢化合物与树

英国数学家亚瑟 · 凯莱在 1857 年发明了树, 当时他试图列举饱和碳氢化合物 $\mathbf{C}_n\mathbf{H}_{2n+2}$ 的同分异构体.

左图为什么是树?

结点数:

$$v = n + (2n + 2) = 3n + 2$$

结点度数之和:

$$4 \times n + 1 \times (2n + 2) = 6n + 2$$

则边数 $e = 3n + 1$. 因是连通图, 且 $e = v - 1$, 所以是树.

7.9

Theorem 3. *任一棵树中至少有两片树叶.*

证: 设树 $T = \langle V, E\rangle$, $|V| = v$. 因 T 连通, 对任意 $v_i \in V$, 有 $\deg(v_i) \geqslant 1$, 而

$$\sum \deg(v_i) = 2e = 2(v - 1) \tag{1}$$

- 如果 T 的每个结点的度数皆大于 2, 则 $\sum \deg(v_i) \geqslant 2v$, 与 (1) 式矛盾.
- 如果 T 只有一个结点的度数等于 1, 则 $\sum \deg(v_i) \geqslant 2(v - 1) + 1$, 即 $\sum \deg(v_i) \geqslant 2v - 1$, 也与 (1) 式矛盾.

故 T 中至少有两个结点的度数等于 1, 即有两片树叶. □ 7.10

[1]Arthur Cayley (1821–1895) 17 岁进入剑桥三一学院学习, 1849 年获律师资格, 在其律师生涯中写下了超过 300 篇的数学论文. 1863 年返回剑桥任教职.

生成树

Definition 4. 若图 G 的生成子图 T 是树, 则称 T 为 G 的生成树.

- T 中的边叫做树枝 (branch);
- G 中不属于 T 的边叫做弦 (chord);
- 所有弦的集合称为 T 的补.

7.11

Example 5. 如图,

- T_1 是 G 的一棵生成树,
- e_1, e_3, e_5, e_7, e_8 是 T_1 的树枝;
- e_2, e_4, e_6 是 T_1 的弦;
- $\{e_2, e_4, e_6\}$ 是 T_1 的补.
- T_2, T_3 也是 G 的生成树.

7.12

Theorem 6. 连通图至少有一棵生成树.

证: 如果图 G 连通且无回路, 则 G 就是一棵生成树.

如果图 G 连通且有一个回路, 则删去回路上的一条边得图 G_1, 显然 G_1 也是连通的.

- 如果 G_1 无回路, 则 G_1 是生成树.
- 如果 G_1 仍有回路, 则重复上述步骤, 直到无回路为止, 可得一个与 G 有相同结点且无回路的连通图, 它是 G 的一棵生成树. □

7.13

连通图的秩

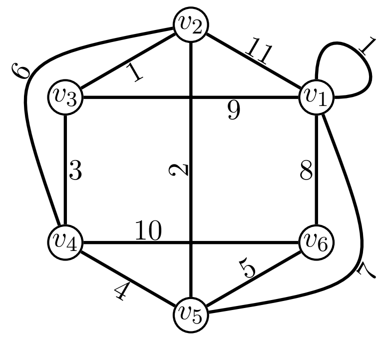
连通图的秩

设 $G = \langle V, E \rangle$ 是一个连通图, $|V| = n$, $|E| = m$, 则 G 的生成树有 $n-1$ 条边. 因此, 为得出 G 的一棵生成树应删去 $m-(n-1)$ 条边. 数 $m-(n-1)$ 称为 G 的秩.

简言之: 连通图 G 的秩是指产生 G 的一棵生成树应删去的边数. 7.14

最小生成树问题

设有 6 个村庄 v_i, $i = 1, 2, \cdots, 6$, 欲修建道路使村村可通. 现已有修建方案如下带权无向图所示, 其中边表示道路, 边上的数字表示修建该道路所需费用, 问应选择修建哪些道路可使得任意两个村庄之间是可达的, 且总的修建费用最低?



7.15

最小生成树

Definition 7. 设图 $G = \langle V, E\rangle$, 对应于 G 的每一条边 e, 指定一个正数 $C(e)$, 称为边 e 的权.

- 设 T 是 G 的生成树, T 的各边权数之和
$$\sum C(e_i)$$
称为树 T 的树权, 记作 $C(T)$.
- G 的所有生成树中树权最小者, 叫最小生成树.

最小生成树在许多实际问题中, 如交通运输, 管道铺设等, 有广泛的应用. 7.16

Kruskal 算法 [2]

Theorem 8. *设图 G 有 n 个结点, 以下算法产生一棵最小生成树:*

1. *取最小权边 e_1, 令 $i := 1$;*
2. *若 $i = n - 1$, 则结束. 否则转 ③ ;*
3. *设已选中的边为 $e_1, e_2, \cdots, e_i$. 在 G 中选不同于 $e_1, e_2, \cdots, e_i$ 的边 e_{i+1}, 使 $\{e_1, e_2, \cdots, e_i, e_{i+1}\}$ 中无回路, 且 e_{i+1} 是满足此条件的最小权边;*
4. *$i := i + 1$, 转 ② .*

7.17

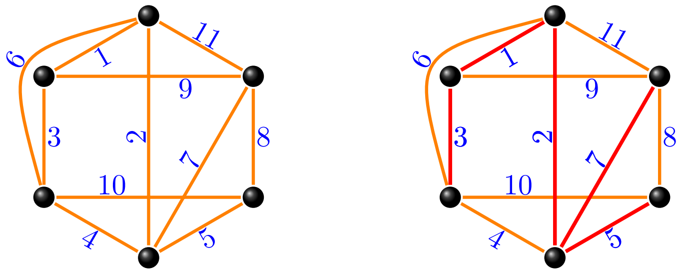
Kruskal 算法

<u>注</u>

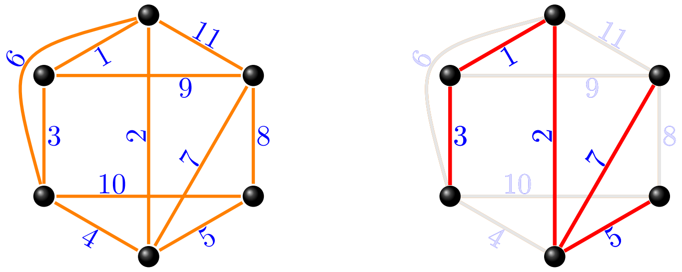
Kruskal 算法实际求解时的两种方式:

1. 逐步选取边权最小的边, 但始终保持已选取的边不构成回路, 直到边数达到 $n - 1$ 条为止.
2. 不断删除边权最大的边而保持图的连通性且无回路. 直到边数剩 $n - 1$ 条时停止.

7.18



(a) (b)



(a) (b)

Example 9 (Kruskal 算法举例). 在下图中求最小生成树 (方法 1).

图中有 6 个结点, 所以要选取 5 条边.

(注意: 边权为 1, 2, 3 的边选取了之后, 边权为 6, 4 的边就不能选了. 否则构成回路.)

7.19

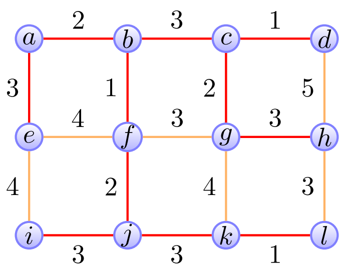
Kruskal 算法举例

Example 10. 在下图中求最小生成树 (方法 2).

7.20

Kruskal 算法举例

Example 11. 用 Kruskal 算法给出最小生成树.



[2]Joseph Bernard Kruskal (b. 1929 in New York) is an American mathematician, statistician, and psychometrician.

解: 选择的步骤如下表.

步骤	边	权
1	$\{c, d\}$	1
2	$\{k, l\}$	1
3	$\{b, f\}$	1
4	$\{c, g\}$	2
5	$\{a, b\}$	2
6	$\{f, j\}$	2
7	$\{b, c\}$	3
8	$\{j, k\}$	3
9	$\{g, h\}$	3
10	$\{i, j\}$	3
11	$\{a, e\}$	3

总计: 24

当然, 最小生成树不是惟一的, 除非图 G 中所有边的权值都不相同. 7.21

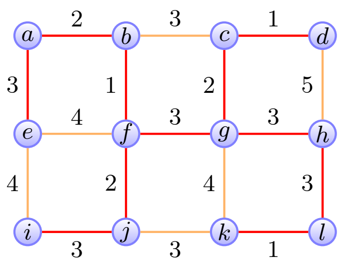
Prim 算法

另一种寻找最小生成树的算法叫做 Prim 算法, 该算法由一个权最小的边开始, 在所有相邻接的边中选择一个权值最小的边, 且不形成回路, 直到形成一个最小生成树.

注意 Prim 算法和 Kruskal 算法的区别. Prim 算法是在与已经选择的树相邻接的边中, 寻找最小权的边, 并且不构成回路. Kruskal 算法不要求下一个找到的边和已经找到的边相邻接, 只要求权最小而且不构成回路. 7.22

Prim 算法举例

Example 12. 用 Prim 算法给出其最小生成树.



解: 选择的步骤如下表.

步骤	边	权
1	$\{b,f\}$	1
2	$\{a,b\}$	2
3	$\{f,j\}$	2
4	$\{a,e\}$	3
5	$\{i,j\}$	3
6	$\{f,g\}$	3
7	$\{c,g\}$	2
8	$\{c,d\}$	1
9	$\{g,h\}$	3
10	$\{h,l\}$	3
11	$\{k,l\}$	1
		总计: 24

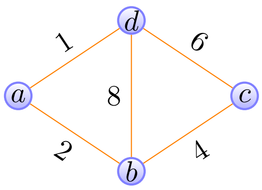
同样, 此问题答案不是惟一的. 7.23

Prim 算法和 Kruskal 算法

Prim 算法和 Kruskal 算法都是一种贪婪算法.

所谓贪婪算法 (greedy algorithm), 是指一类采用 “局部最优” 方式的说法, 它在每次循环时都只考虑如何使本次选择做到最优, 而暂不考虑总体是否达到最优.

但是, 每一步最优并不一定能保证全局最优, 例如, 如果采用贪婪算法在下图中求解由 a 到 c 的最短路径, 若在每一步都选择到已选顶点最小权值的边, 将得到路径 (a,d,c), 但这并不是从 a 到 c 的最短路径.



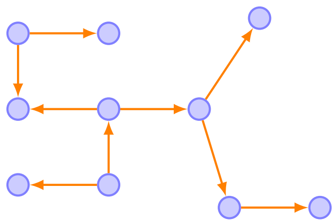
Prim 算法和 Kruskal 算法这两个贪婪算法, 对最小生成树的结果都是最优的. 7.24

2 根树及其应用

根树及其应用

1. 有向树
2. m 叉树
3. 有序树
4. 最优树
5. 前缀码

7.25



有向树

Definition 13. 如果一个有向图在不考虑边的方向时是一棵树, 则这个有向图称为*有向树*.

Example 14. 如图为一棵有向树: 7.26

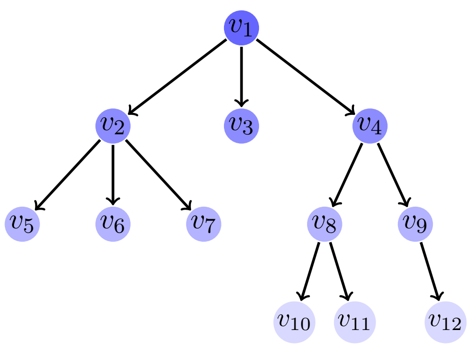
有向树

Definition 15. 一棵有向树, 如果恰有一个结点的入度为 0, 其余所有结点的入度都为 1, 则称为*根树* (rooted tree).

1. 入度为 0 的结点称为根;
2. 出度为 0 的结点称为叶;
3. 出度不为 0 的结点称为*分枝点*或*内点*.

7.27

Example 16. 如图为一棵根树:



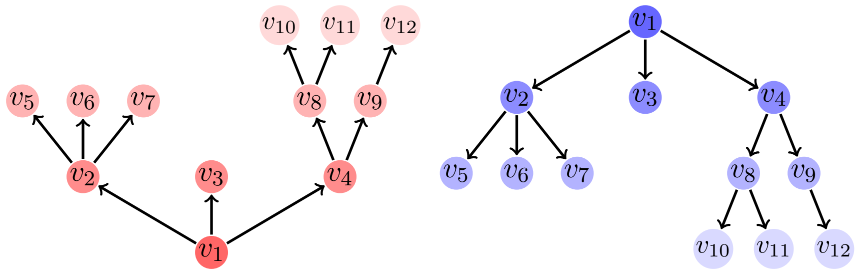
其中, v_1 为根, v_1, v_2, v_4, v_8, v_9 为*分枝点*, 其余结点为叶. 7.28

有向树

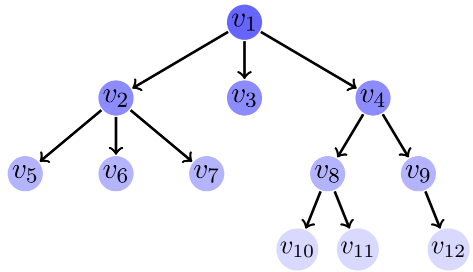
结点的层次

在一棵根树中, 从根到某个结点的单向通路的长度 (即边数) 是固定的, 它叫做该结点的*层次*.

Example 17. 如图, 结点 v_2, v_3, v_4 的层次为 1. 结点 v_{10}, v_{11}, v_{12} 的层次为 3.

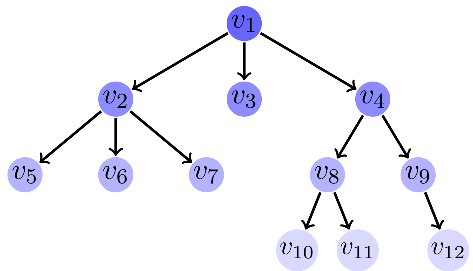


(a) (b)



7.29

根树的递归定义



根树中每一个结点都可以看作是原来树中某一棵子树的根. 从而, 根树可递归定义为:

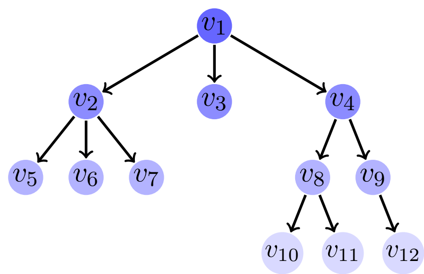
Definition 18. 根树包含一个或多个结点, 这些结点中某一个称为根, 其他所有结点被分成有限个子根树. 7.30

根树的两种画法

根树可以有树根在上或树根在下两种画法, 它们是同构的.

Example 19.

图 (a) 是根树的自然表示法. 7.31



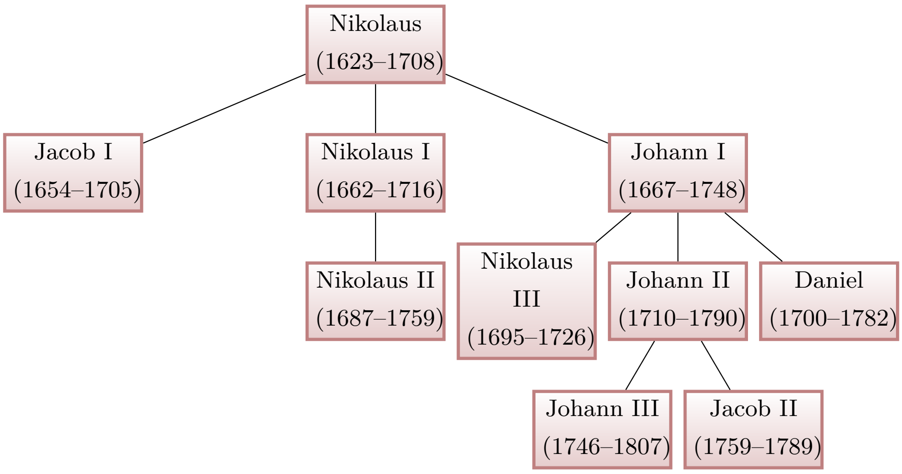
Example 20.

图中,

- v_2 称为 v_1 的儿子, v_1 称为 v_2 的父亲;
- v_1 称为 v_5 的祖先, v_5 称为 v_1 的后裔;
- v_{10}, v_{11}, v_{12} 称为兄弟.

对其它的结点有类似的说法. 树的术语起源于植物学和族谱学. 7.32

伯努利家族[3]的族谱图



7.33

m 叉树

Definition 21. 1. 在根树中, 若每个结点的出度小于等于 m, 则称该树为 m 叉树.

2. 如果每个结点的出度恰好等于 m 或 0, 则称该树为完全 m 叉树.
3. 如果完全 m 叉树所有树叶的层次都相同, 则称为正则 m 叉树. 7.34

Example 22. 7.35

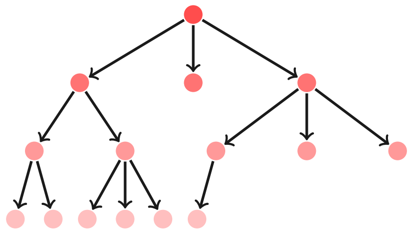
用 m 叉树表示实际问题

Example 23. M 和 E 两人进行网球比赛, 如果一人连胜两局或一共胜三局, 则比赛结束, 试用二叉树表示比赛可能发生的各种情况.

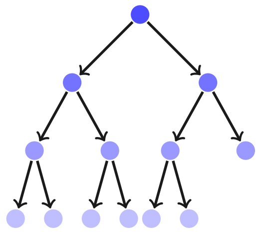
解: 用二叉树的分枝点表示每局赛事, 每局比赛的胜负标在其两个儿子结点的旁边. 从树根到树叶的每一条路对应比赛可能发生的一种情况:

- EE, MM ; EMM , MEE ; 7.36
- $EMEE$, $MEMM$;
- $EMEME$, $EMEMM$, $MEMEE$, $MEMEM$.

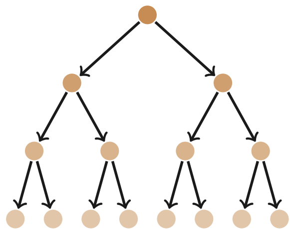
[3] 著名的瑞士数学家家族. 见: E.T. 贝尔《数学精英》P.152, 商务印书馆.



(a) 三叉树



(b) 完全二叉树



(c) 正则二叉树

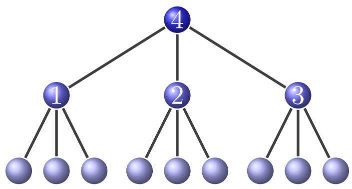
Theorem 24. 设有完全 m 叉树, 其树叶数为 l, 分枝点数为 i, 则

$$(m-1)i = l-1$$

证: 可将完全 m 叉树视为每局有 m 位选手参加比赛的单淘汰赛计划表.

树叶数 l 表示参加比赛的选手数, 分枝点表示比赛的局数. 因每局比赛将淘汰 $(m-1)$ 位选手, 故比赛结果共要淘汰 $(m-1)i$ 位选手, 最后得出一位冠军.

因此 $(m-1)i+1=l$, 即 $(m-1)i=l-1$. □



例如左图所示为 9 位选手参加比赛的情况: 7.37

$$l=9, \quad i=4, \quad m=3.$$

完全 m 叉树的性质

事实上, 这个定理还有一个简单的证明.

对完全 m 叉树, 设其树叶数为 l, 分枝点数为 i, 则结点总数 n 为

$$n=l+i, \qquad 或 \qquad n=mi+1.$$

所以 $l+i=mi+1$, 即 $(m-1)i=l-1$.

其中 $n=mi+1$ 是因为每个分枝点都有 m 个儿子, 且全部结点中只有树根不是某个结点的儿子. 这个结论可以作为一个定理.

Theorem 25. 对完全 m 叉树, 若分枝点数为 i, 则结点总数为 $n=mi+1$. 7.38

完全 m 叉树的性质

从上面的讨论中, 我们还可以看到, 对完全 m 叉树, 知道 l, i, n 这三个中的一个, 则其它两个必定可求.

Theorem 26. *对完全 m 叉树,*

1. *若结点总数 n 为已知, 则分枝点数为 $i = (n-1)/m$, 树叶数为 $l = n - i = n - (n-1)/m$.*
2. *若分支点数 i 为已知, 则结点总数为 $n = mi + 1$, 树叶数为 $l = n - i = (m-1)i + 1$.*
3. *若树叶数 l 为已知, 则结点总数为 $n = (ml-1)/(m-1)$, 分枝点数为 $i = (l-1)/(m-1)$.* 7.39

Example 27. 假定我们开始一个发送手机短信的游戏, 要求每个收到短信的人把短信再转发给另外某 4 个人, 每人只能收到一次短信. 有些人收到短信后这样做了, 但是有些人收到短信后没有转发. 如果某个时刻已经有 100 人收到了短信而没有转发, 问有多少人发出过短信?

解: 可以用一个完全 4 叉树表示这个过程. 每个人是一个结点, 收到了短信而没有转发的人是树叶, 转发了短信的人是分枝点, 则树叶总数是 $l = 100$.

分枝点数为

$$i = \frac{l-1}{m-1} = \frac{100-1}{4-1} = 33,$$

所以一共有 33 人发出了短信. □ 7.40

有序树

在根树中, 一般将树根放在最上面, 但分枝点关联的各分枝的左中右按顺序却有种种不同的摆法. 所以, 标定根树中结点和边的次序在应用中是必要的.

- 标定了结点和边的次序的根树叫*有序树*.
- 任一有序树都可用二叉树来表示. 7.41

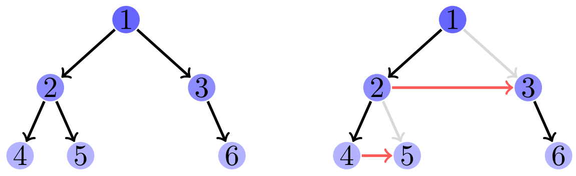
有序树写为对应的二叉树

任何一棵有序树都可以改写为一棵对应的二叉树, 方法如下:

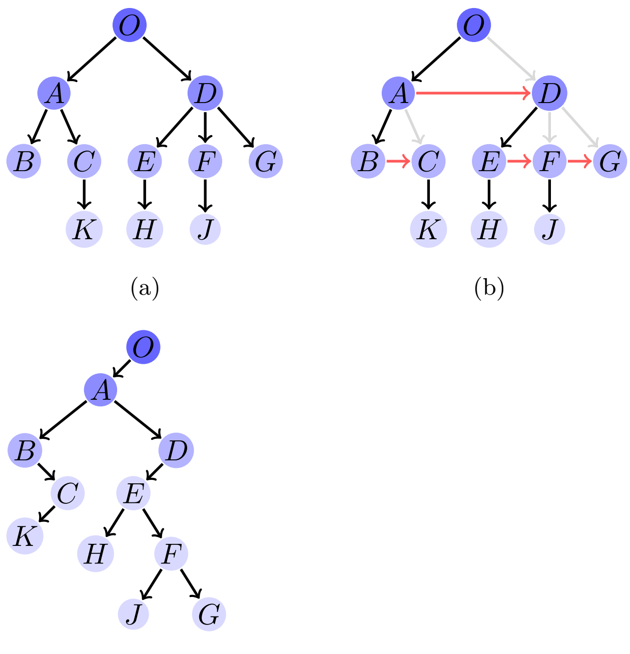
1. 删除始于每个结点除最左边的一个分枝外的其余分枝; 在同一层次中的兄弟结点之间用自左到右的有向边连接.
2. 对某个结点, 直接位于该结点下面的结点作为*左儿子*. 位于同一水平线上与该结点右邻的结点作为*右儿子*.
3. 改写之后的树根仅有一个儿子, 规定是左儿子.

下页是一个例子... 7.42

7.43



(a) (b)



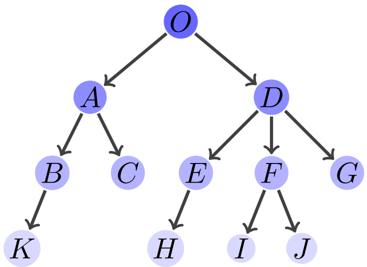
(a) (b)

(c)

通路长度

Definition 28. 1. 在根树中, 从树根到某结点的通路的边数叫该结点的通路长度.

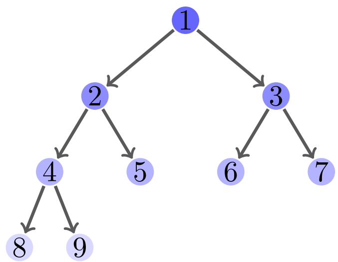
2. 将分枝点的通路长度称为内部通路长度;
3. 树叶的通路长度叫外部通路长度.



例如, 图中结点 K 的通路
长度为 3; 结点 F 的通路长 7.44
度为 2;

Theorem 29. *设完全二叉树有 n 个分枝点 (含根结点), 且内部通路长度的总和为 I, 外部通路长度的总和为 E, 则 $E = I + 2n$.*

例如



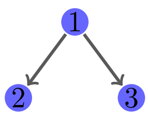
上图为一棵完全二叉树, 其中有 9 个结点, 分枝点 (即内点) 有 4 个,

$$I = 1 + 1 + 2 = 4,$$

$$E = 3 + 3 + 2 + 2 + 2 = 12.$$

证: 对分枝点数 n 进行归纳证明.

当 $n = 1$ 时, $E = 2$, $I = 0$, 故公式成立.



续证: 假设 $n = k - 1$ 时成立.

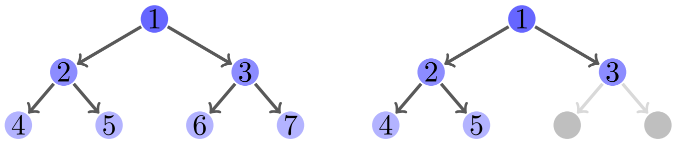
当 $n = k$ 时, 若删去一个分枝点 v (即删除该分枝点的儿子).

设该分枝点的通路长度为 l, 且 v 的两个儿子是树叶, 得出的新树为 T'.

与原树相比, T' 减少了两片通路长度为 $l + 1$ 的树叶和一个通路长度为 l 的分枝点, 所以, $E' = E - 2(l + 1) + l$, 且 $I' = I - l$.

又 T' 有 $k - 1$ 个分枝点, 所以 $E' = I' + 2(k - 1)$.

代入前两式并整理得 $E = I + 2k$. □ 7.45



(a) T (b) T'

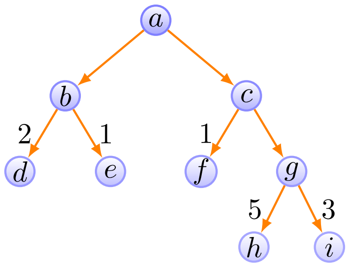


Figure 1: 带权二叉树 T

最优树

Definition 30.
- 设二叉树有 t 片树叶, 各片树叶分别带有权数 $w_1, w_2, \cdots, w_t$, 则该二叉树称为带权二叉树.
- 设带权二叉树中权为 w_i 的树叶的通路长度为 $L(w_i)$, 将
$$w(T) = \sum_{i=1}^{t} w_i L(w_i)$$
称为带权二叉树的权.
- 在所有带权 $w_1, w_2, \cdots, w_t$ 的二叉树中, $w(T)$ 最小的树称为最优树.

7.46

Example 31.
$$\begin{aligned} w(T) &= \sum_{i=1}^{t} w_i L(w_i) \\ &= 2\times 2 + 1\times 2 + 1\times 2 + 5\times 3 + 3\times 3 \\ &= 32 \end{aligned}$$

7.47

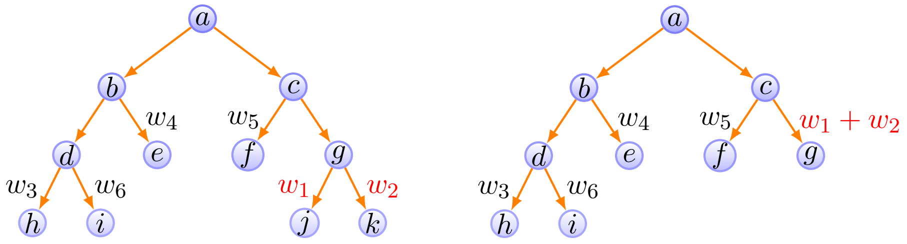
Theorem 32. 设 T 为带权 $w_1 \leqslant w_2 \leqslant \cdots \leqslant w_t$ 的最优树, 则

① 带权 w_1, w_2 的树叶 v_{w_1}, v_{w_2} 是兄弟.

② 以树叶 v_{w_1}, v_{w_2} 为儿子的分枝点的通路最长.

简而言之: 最优树中具有最小权的两片树叶是兄弟, 且其通路长度最大.

证: 设带权 $w_1, w_2, \cdots, w_t$ 的最优树中, v 是通路最长的分枝点. v 的两个儿子分别带权 w_x, w_y. 那么
$$L(w_x) \geqslant L(w_1), \qquad L(w_y) \geqslant L(w_2)$$
若 $L(w_x) > L(w_1)$, 将 w_x 与 w_1 对调得到新树 T', 则
$$\begin{aligned} w(T') - w(T) &= \big(L(w_x)w_1 + L(w_1)w_x\big) - \big(L(w_x)w_x + L(w_1)w_1\big) \\ &= L(w_x)(w_1 - w_x) + L(w_1)(w_x - w_1) \\ &= (w_x - w_1)\big(L(w_1) - L(w_x)\big) < 0 \end{aligned}$$



(a) T (b) T'

所以 $w(T') < w(T)$, 与 T 是最优树的假设矛盾, 故 $L(w_x) = L(w_1)$.

同理可证 $L(w_x) = L(w_2)$. 所以 $L(w_1) = L(w_2) = L(w_x) = L(w_y)$.

分别将 w_1, w_2 与 w_x, w_y 对调, 得出最优树, 带权 w_1, w_2 的树叶是兄弟. □ 7.48

Theorem 33. 设 T 为带权 $w_1 \leqslant w_2 \leqslant \cdots \leqslant w_t$ 的最优树, 若将以带权 w_1, w_2 的树叶为儿子的分枝点改为带权 $w_1 + w_2$ 的树叶, 得到一棵新树 T', 则 T' 也是最优树.

注意:

$$w(T) = w(T') + w_1 + w_2$$

简而言之: 求带 t 个权的最优树, 可简化为求带 $t-1$ 个权的最优树. 证: 由题设可知:

$$w(T) = w(T') + w_1 + w_2 \tag{2}$$

若 T' 不是最优树, 则必有另一棵带权 $w_1 + w_2, w_3, \cdots, w_t$ 的最优树 T''. 将 T'' 中带权 $w_1 + w_2$ 的树叶 $v_{w_1+w_2}$ 生成两个儿子 (带权 w_1, w_2 的树叶), 得到新树 $\widehat{T}$, 则

$$w(\widehat{T}) = w(T'') + w_1 + w_2 \tag{3}$$

由假设, $w(T'') \leqslant w(T')$. 如果 $w(T'') < w(T')$, 比较 (2), (3) 式可知 $w(\widehat{T}) < w(T)$, 与 T 是带权 $w_1, w_2, \cdots, w_t$ 的最优树相矛盾.

因此, $w(T'') = w(T')$. 即 T' 是带权 $w_1 + w_2, w_3, \cdots, w_t$ 的最优树. □ 7.49

前缀码

Definition 34. 给定一个序列的集合, 如果不存在一个序列是另一个序列的前缀, 则该序列的集合称为前缀码.

例如,

- $\{100, 101, 00, 01, 11\}$ 是前缀码,
- 而 $\{001, 010, 00, 01, 10\}$ 就不是前缀码.

7.50

Theorem 35. 任意一棵二叉树对应一个前缀码.

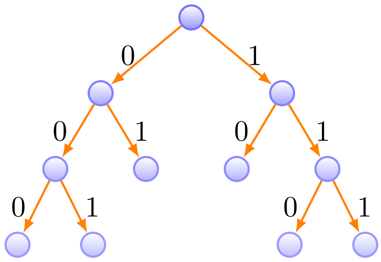


Figure 2: 带权二叉树 T

证: 给定一棵二叉树, 从每个分枝点出发, 将左枝标为 0, 右枝标为 1, 则每片树叶对应一个 0 和 1 组成的序列, 该序列是从树根到该树叶的通路上各边标号组成的.

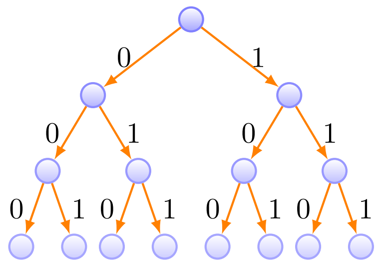
显然, 没有一片树叶对应的序列, 是另一片树叶对应序列的前缀. 所以任意一棵二叉树对应一个前缀码.

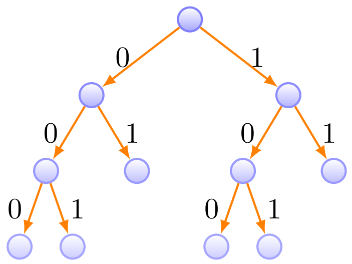
7.51

Theorem 36. 任意一个前缀码对应一棵二叉树.

证: 设 h 是某个前缀码中最长序列的长度. 作一棵高为 h 的正则二叉树, 将每个分枝点的左、右枝分别标以 0 和 1, 则每片树叶对应一个 0 和 1 组成的序列, 该序列是从树根到该树叶的通路上各边标号组成的. 因此, 长度不超过 h 的每个 0 和 1 组成的序列必对应一个结点.

将对应于前缀码中的每个序列的结点给予一个标记, 并删去标记结点的后裔及由它们射出的边. 再删去未加标记的树叶, 得到一棵新的二叉树, 其树叶就对应给定的前缀码.



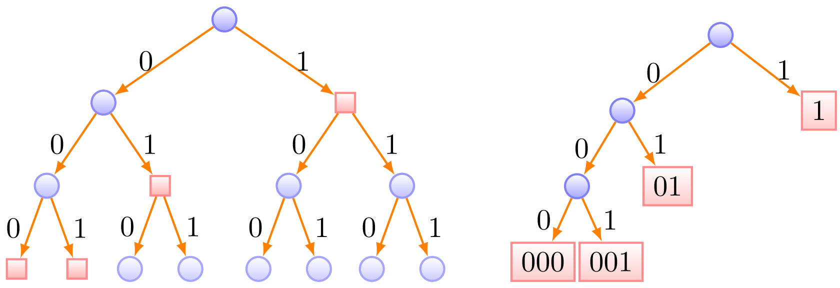


7.52

Example 37. 给定前缀码 $\{000, 001, 01, 1\}$, 求对应的二叉树.

解: 作一棵高为 3 的正则二叉树; 标记对应前缀码的结点, 并删去标记结点的后裔.

7.53



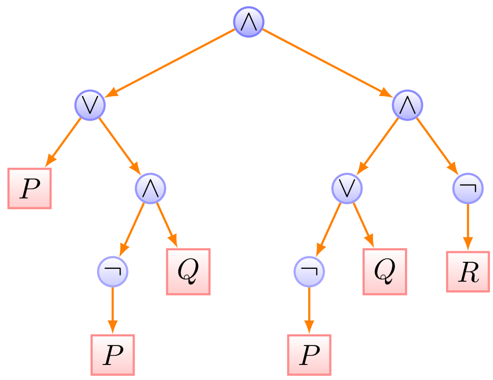
(a) T　　　(b) T'

- 由前缀码和二叉树的对应关系可知, 如果给定的前缀码对应的二叉树是完全二叉树, 则根据此前缀码可对任意二进制序列译码.
  例如, 由前例中的前缀码 $\{000, 001, 01, 1\}$ 可对任意二进制序列译码. 随意写一个由 0 和 1 组成的字符串, 例如:
  $$00010011011101001$$
  它可译为且只能译为:
  $$000, 1, 001, 1, 01, 1, 1, 01, 001$$
- 如果被译的序列的最后部分不能构成前缀码中的序列, 可约定添加 0 或 1, 直到能译出为止.
- 如果前缀码对应的二叉树不是完全二叉树, 例如 $\{000, 001, 1\}$ 也是前缀码, 但它不能对任意二进制序列译码. 上述二进制序列, 用这组编码就无法译出. 7.54

练习

用根树表示 $(P \vee (\neg P \wedge Q)) \wedge ((\neg P \vee Q) \wedge \neg R)$.

解:



7.55

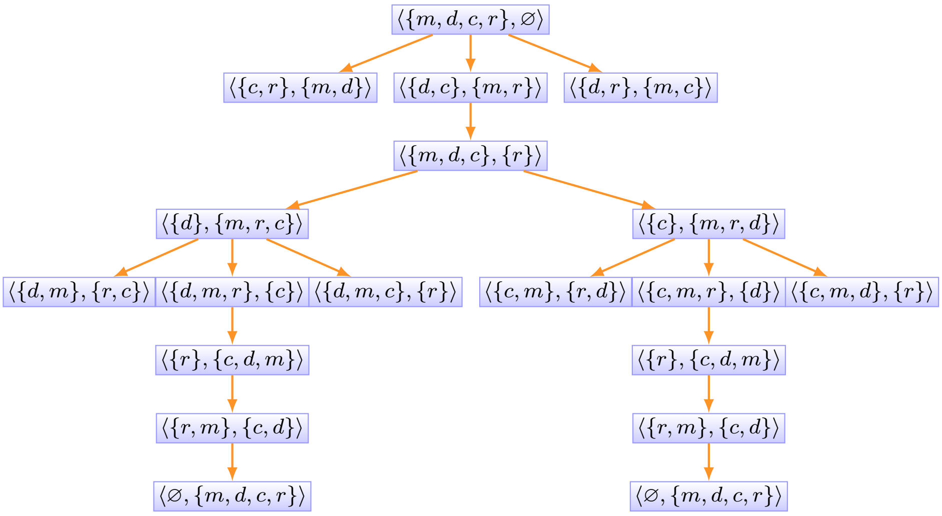
Example 38. 试用有向图描述下列问题的解:

某人 m 带一条狗 d, 一只猫 c 和一只兔子 r 过河. m 每次游过河时只能带一只动物, 而没人管理时, 狗与兔子不能共处, 猫和兔子也不能共处. 问 m 怎样把三个动物带过河去?

分析：用结点代表状态，状态用序偶 $\langle S_1, S_2\rangle$ 来表示，这里 S_1，S_2 分别是左岸、右岸的人和动物的集合. 例如初始状态为 $\langle\{m,d,c,r\}, \varnothing\rangle$，最终的目的即是 $\langle\varnothing, \{m,d,c,r\}\rangle$.

注意不能出现集合 $\{d, r\}$，$\{c, r\}$. 当出现集合 $\{d, r\}$ 或 $\{c, r\}$ 时，方案终止. 如果出现之前已有的状态，方案也终止，如下一页图中第 4 层出现的 $\langle\{d,m,c\}, \{r\}\rangle$ 和 $\langle\{c,m,d\}, \{r\}\rangle$，都是返回到了第 2 层的状态 $\langle\{m,d,c\}, \{r\}\rangle$，从而方案终止. 7.56

解: 注意到不能出现集合 $\{d, r\}$ 或 $\{c, r\}$，描述上述问题的有向图如下.



7.57