树

Chapter 7

本章概要

- 树的概述
- 树的遍历
- 生成树
- 最小生成树

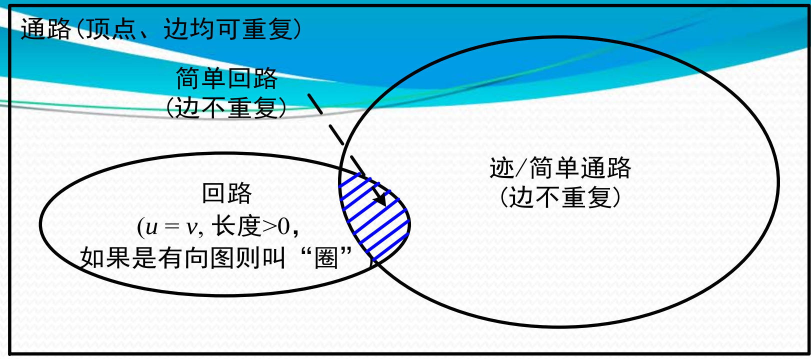
树的概述

Section 7.1

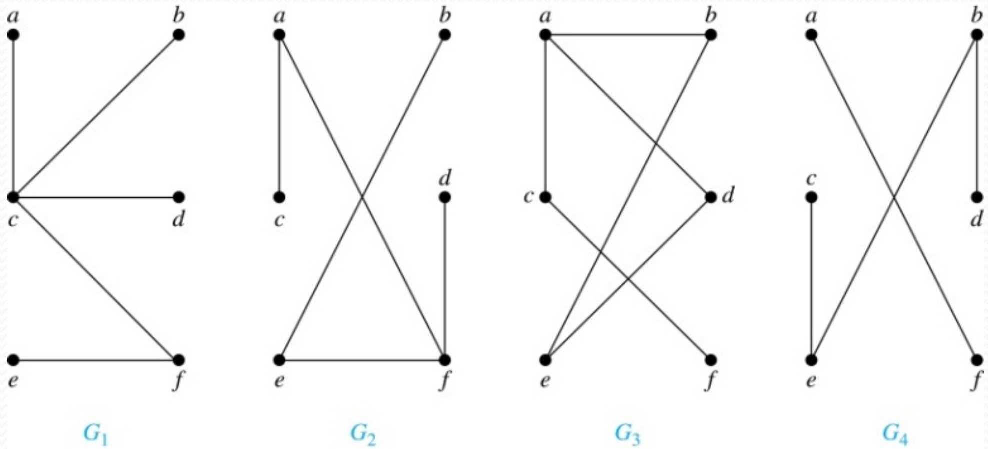
本节概要

- 树的介绍
- 有根树
- 树的性质

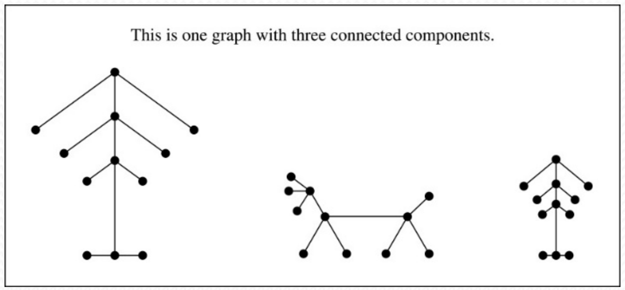
树



定义：树是没有简单回路的连通无向图。



定义：不含简单回路但不一定连通的图称为森林，森林的每一个连通分支都是一棵树。



树

定理：一个无向图是树当且仅当在它的每对顶点之间存在唯一简单通路。

证明：假设T是一个树。则T是没有简单回路的连通图。设x和y是T的两个顶点。因为T是连通的，所以存在一条简单通路连接x和y(根据P3036.4中的定理1：在连通无向图的每一对不同顶点之间都存在简单通路)。而且，这条通路必然是唯一的。因为假如存在第二条这样的通路，那么从x到y的两条路将能够组成一个回路。因此在树的任意两个顶点之间存在唯一简单通路。

现在假设图T的任意两个顶点之间有一条唯一的简单通路，那么T是连通的，因为它的任意两个顶点之间有一条通路。此外，T不能有简单的回路，因为如果有一个简单的回路，在一些顶点之间会有两条通路。

因此，在任意两个顶点之间存在唯一简单通路的图是树。 ◀

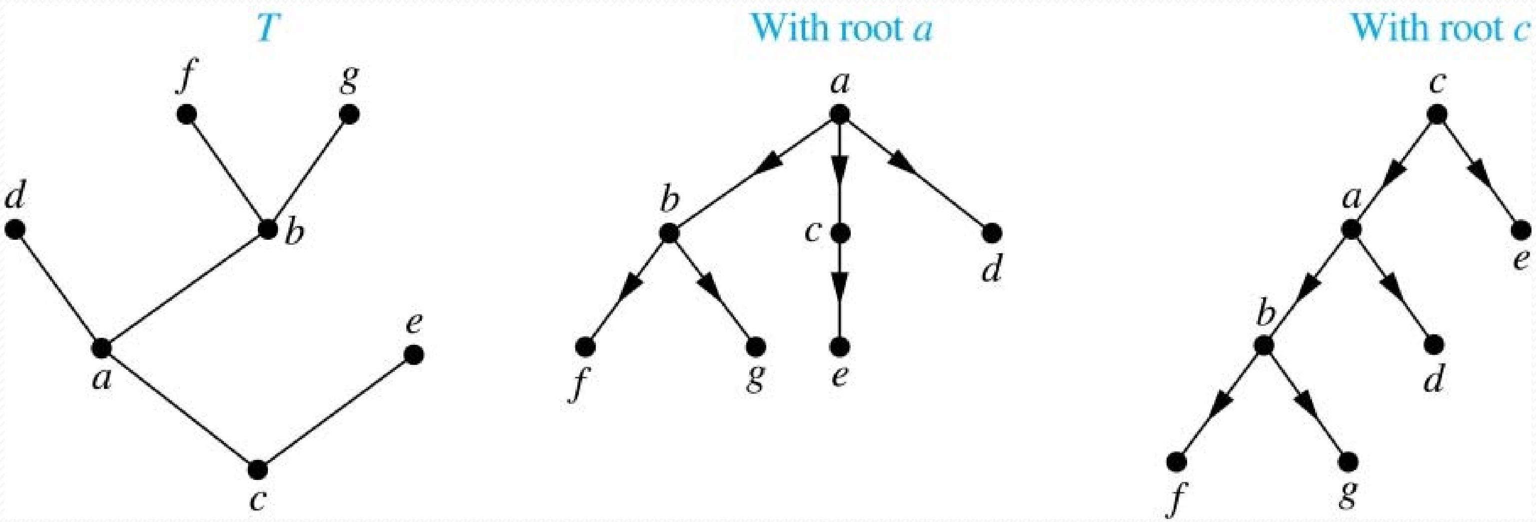
树的几个等价定义

- 给定图T，以下关于树的定义是等价的：
  - 无简单回路的连通图；
  - 无简单回路且$e = v - 1$，其中e为边数，v为结点数；
  - 连通且$e = v - 1$；
  - 无简单回路且增加一条新边，得到一个且仅一个简单回路；
  - 连通且删去任何一个边后不连通；
  - 每一对顶点之间有一条且仅一条简单通路。

有根树（根树）

定义：有根树是指定的一个顶点作为根并且每条边的方向都离开根的树。

当选择不同的顶点作为根时，会产生不同的有根树。

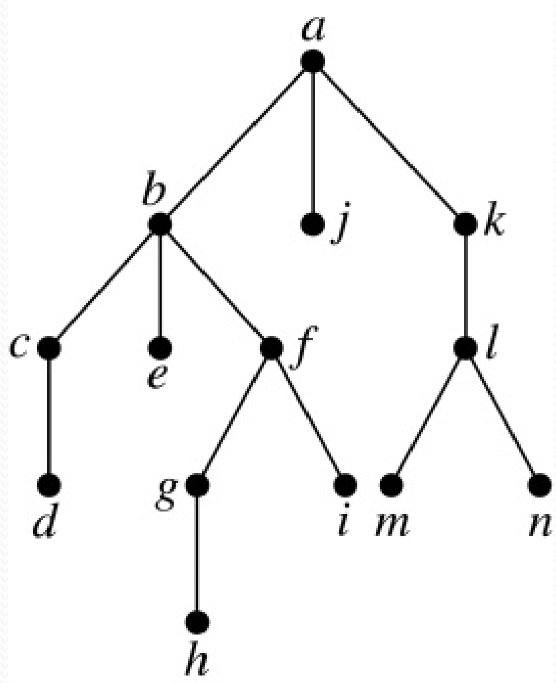


有根树的术语

- 对于无向图来说，可以指定任何一个顶点作为树根。对于有向图而言，入度为0的顶点被称为树根，其他所有顶点的入度为1。
- 如果v是根以外的有根树的顶点，则v的父亲顶点是唯一的顶点u。这样就有一条从u到v的有向边。如果u是v的父亲顶点，则v称为u的儿子顶点。具有相同父亲顶点的顶点称为兄弟顶点。
- 非根顶点的祖先是从根到该顶点通路上的顶点，不包括该顶点自身但包括根。顶点v的后代是以v作为祖先的顶点。
- 树的顶点若没有儿子则称为树叶。树叶的出度为0。在一棵树中，除了树叶以外，其他所有的顶点统称为内点。
- 若a是树中的顶点，则以a为根的子树是由a和a的后代以及这些顶点所关联的边所组成的该树的子图。

顶点的层数和树的高度

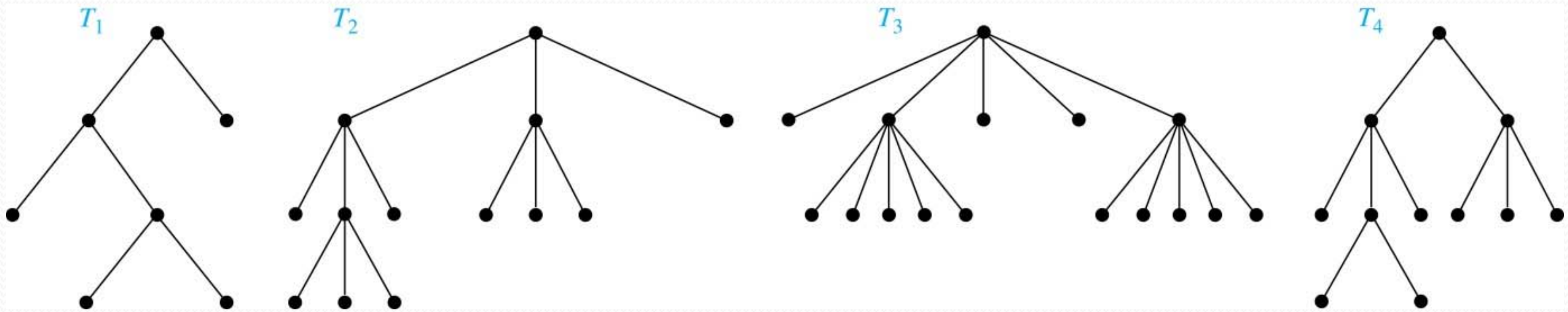
- 当使用树时，我们通常希望有根树，其中每个顶点的子树包含大致相同长度的路径。
- 为了使这个想法更精确，我们需要一些定义:
  - 有根树中顶点的层数是从根到该顶点的唯一路径的长度。
  - 有根树的高度是顶点层数的最大值。



m叉树及相关定义

定义：若有根树的每个内点都有不超过m个儿子，则它称为m叉树。若该树的每个内点都恰好有m个儿子，则称它为完全m叉树(新教材的满m叉树)。如果所有的树叶层次都相同，则称它为正则m叉树。如果所有的树叶层次都相同并且每个内点都有m个儿子，则称它为满m叉树。

把m=2的m叉树称为二叉树。



有序根树

Definition: 有序根树十八每个内点的孩子都排序的有根树。

- 定义：

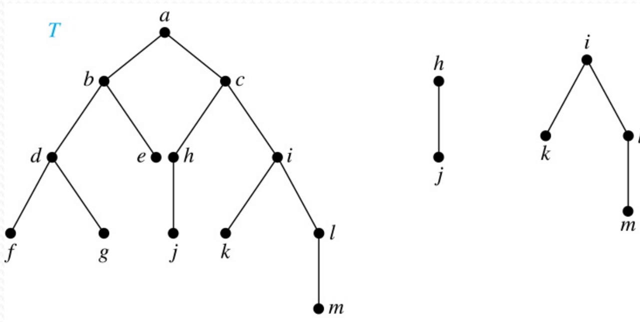
•画有序根树时，以从左向右的顺序来显示每个内点的孩子。.

Definition: 在有序二叉树中，若一个内点有两个孩子，则第一个孩子称为左子而第二个孩子称为右子。以一个顶点的左子为根的树称为左子树，而以一个顶点的右子为根的树称为该顶点的右子树。

Example: 在下图中，d的左子和右子是什么?c的左子树和右子树是什么?

Solution:

(*i*) d的左子是f，而右子是g。
(*ii*) 右边两图分别显示c的左子树和右子树。



(a) (b) (c)

树的性质

定义：带有n个顶点的树含有$n-1$条边。

证明 (数学归纳法)：

基础步骤：当 $n = 1$, 有1个顶点的树没有边。所以对于$n = 1$来说，定理为真。

归纳步骤：归纳假设有k个顶点的每棵树都有$k-1$条边，其中k是正整数。假设一棵树T有$k+1$个顶点，v是T的一片叶子，设w为v的父顶点，去掉顶点v和连接w到v的边，生成一棵k个顶点的树T'。根据归纳假设，T'有$k-1$边。因为T比T'多了1条边，我们看到T有k条边。这就完成了归纳步骤。◀

计算完全m叉树中的顶点数

定理：带有i个内点的完全m叉树含有$n = mi + 1$个顶点。

证明：除了根之外，每个顶点都是内点的儿子。因为每个内部顶点都有m个儿子节点，所以树中除了根之外还有$m*i$个顶点。因此，树包含$n = mi + 1$个顶点。◀

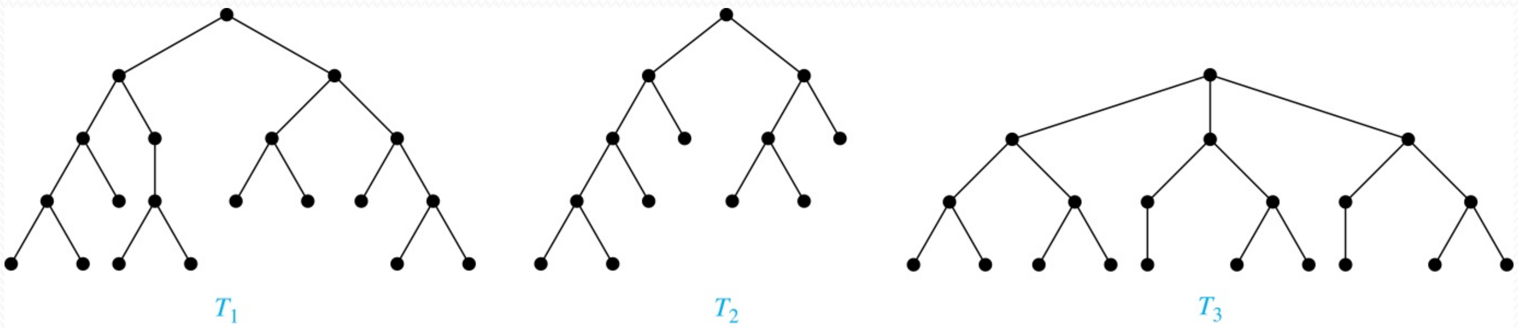
定理：设有完全m叉树，其树叶数为t，内点数为i，则$(m - 1)i = t - 1$。

计算完全m叉树中的顶点数

- **定理：**对一个完全m叉树，若有
  - n个顶点，则有 $i = (n - 1)/m$ 个内点和$t = [(m - 1)n + 1]/m$ 个树叶；
  - i个内点，则有$n = mi + 1$ 个顶点和$t = (m - 1)i + 1$ 个树叶；
  - t个树叶，则有$n = (mt - 1)/(m - 1)$ 个顶点和$i = (t - 1)/(m - 1)$个内点。

平衡的m叉树

定义：若一棵高度为h的m叉树的所有树叶都在h层或者h-1层，则这棵树是平衡的。



平衡　　　　不平衡　　　　平衡

树的遍历

Section 7.3

本节概要

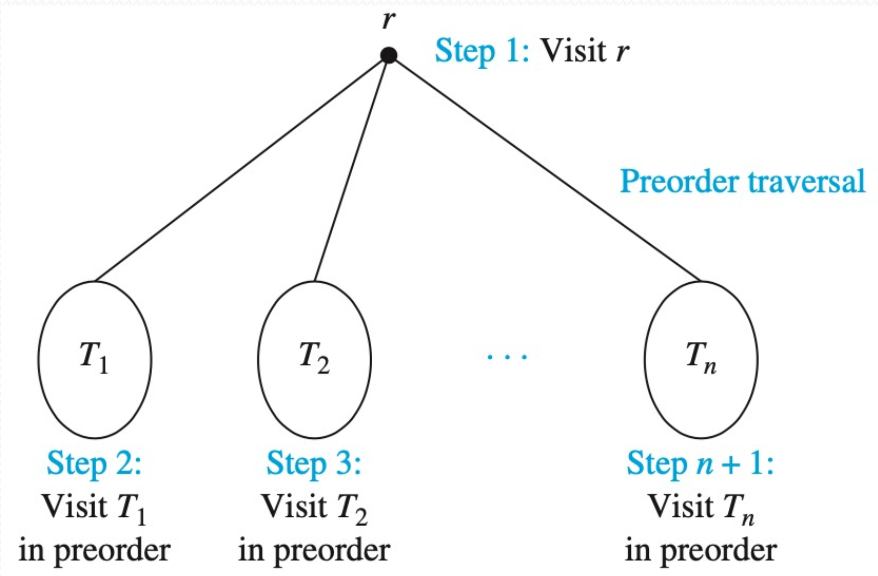
- 遍历算法

树的遍历

- 系统地访问有序树(对每个内点的孩子都线性地排序的树)的每个顶点的过程称为遍历。
- 最常用的三种遍历算法是前序遍历、中序遍历和后序遍历。

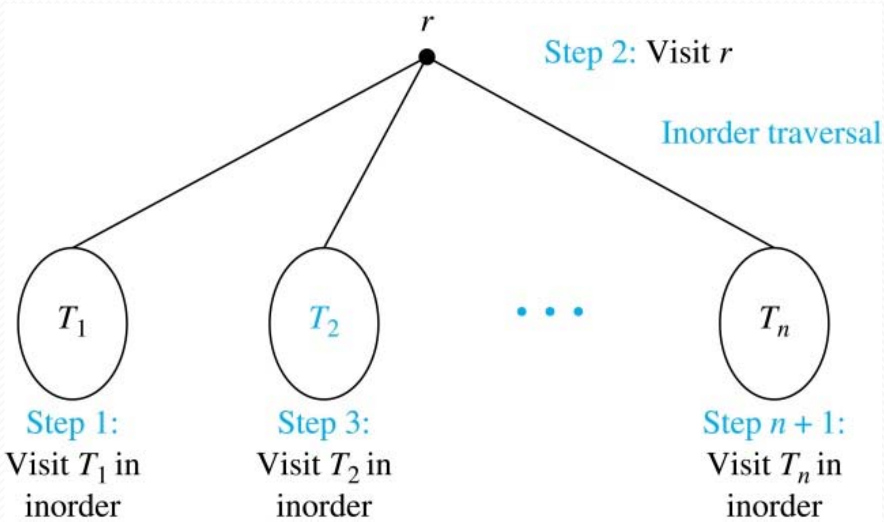
前序遍历

定义：设T是带根r的有序根树。若T只包含r，则r是T的前序遍历。否则，假定$T_1, T_2, \ldots, T_n$是T的以r为根的从左向右的子树。前序遍历先访问r。它接着以前序来遍历T_1，然后以前序来遍历T_2以此类推，直到以前序遍历了T_n为止。



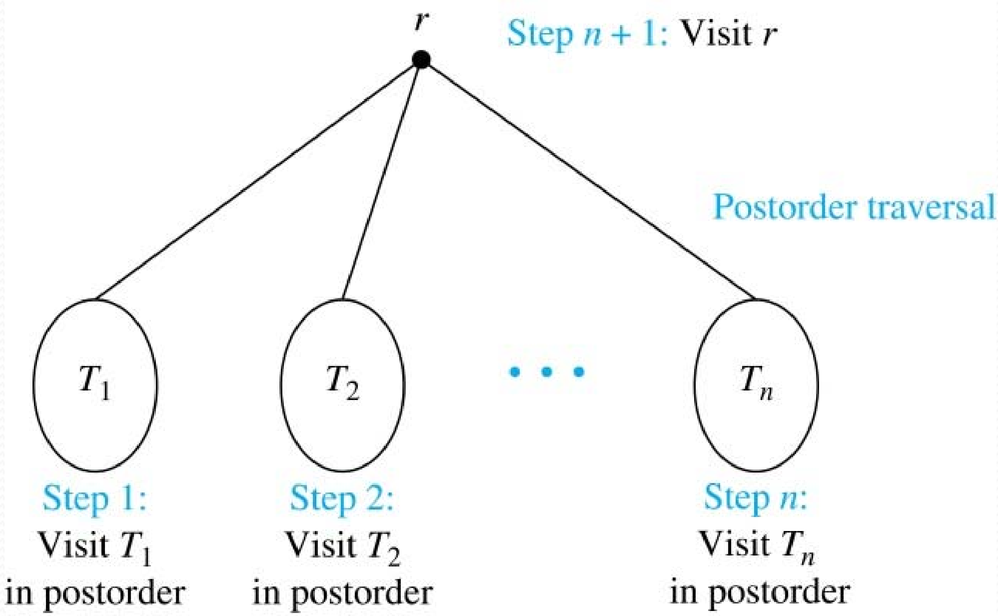
中序遍历

定义：设T是带根r的有序根树。若T只包含r，则r是T的中序遍历。否则，假定$T_1, T_2, \ldots, T_n$是T中以r为根的从左向右的子树。中序遍历首先以中序来遍历T_1，然后访问r。接着以中序来遍历T_2以此类推直到以中序遍历了T_n为止。

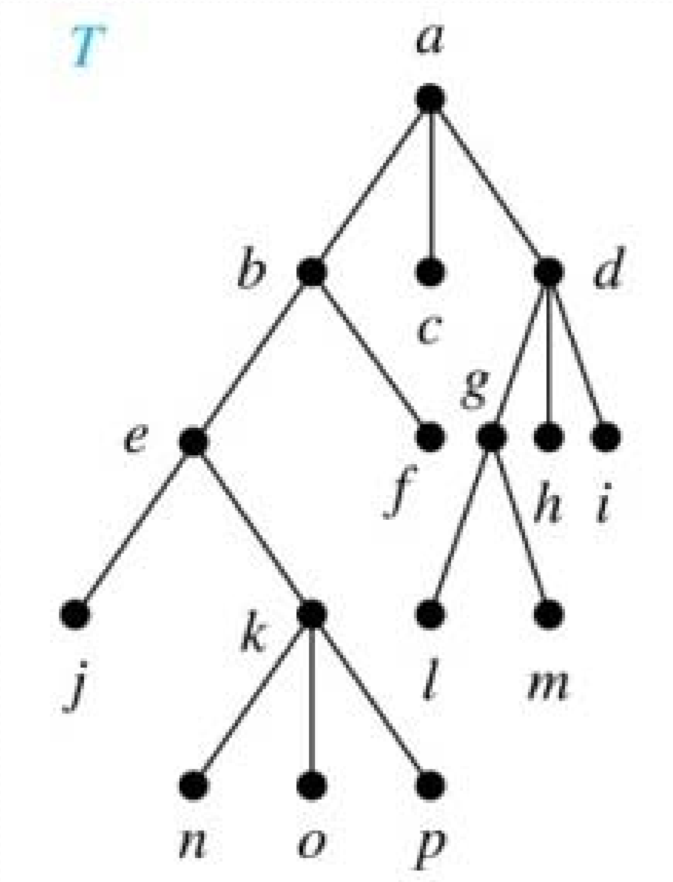


后序遍历

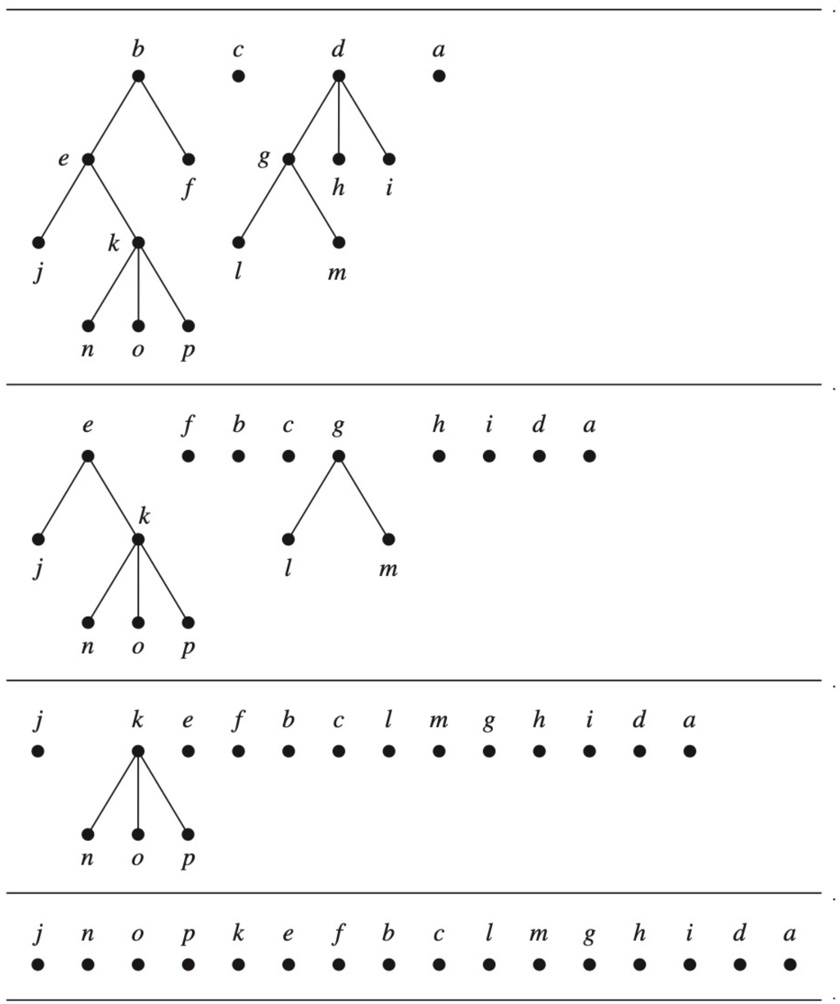
定义： 设T是带根r的有序根树。若T只包含r，则r是T的后序遍历。否则，假定$T_1, T_2, \ldots, T_n$ 是T的以r为根的从左向右的子树。后序遍历首先以后续来遍历T_1，然后以后续来遍历T_2，然后以后续来遍历T_n，最后访问r。



遍历举例



T
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p



b c d a
e f g h i
j k l m
n o p
e f b c g h i d a
k
j l m
n o p
j k e f b c l m g h i d a
n o p
j n o p k e f b c l m g h i d a

生成树

Section 7.4

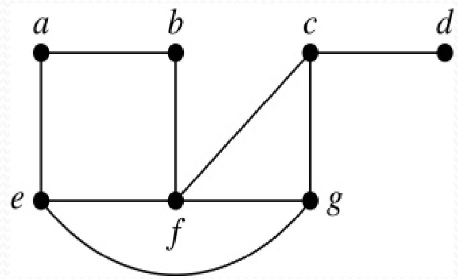
本节概要

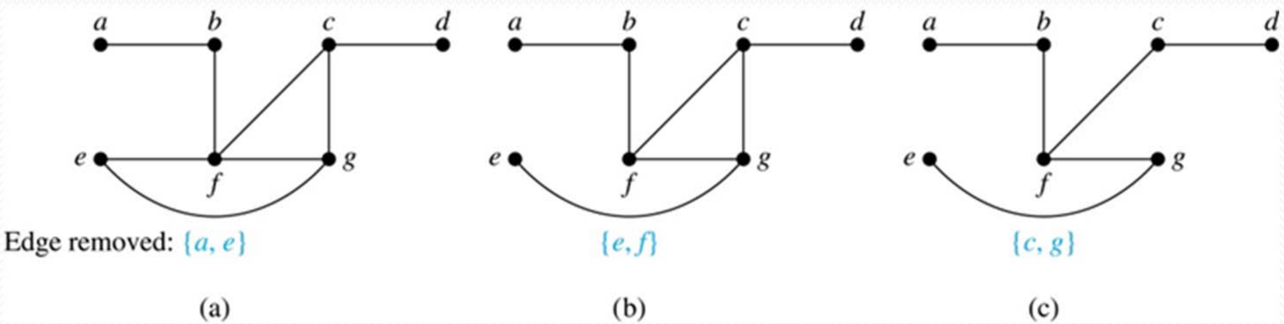
- 深度优先搜索
- 宽度优先搜索

生成树

定义：设G是简单图，如果G的生成子图G'是一棵树，则称G'是G的生成树。

例：找出图示简单图的生成树(破圈法)：



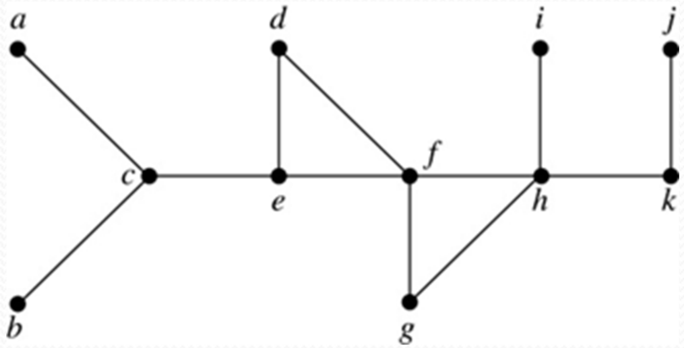


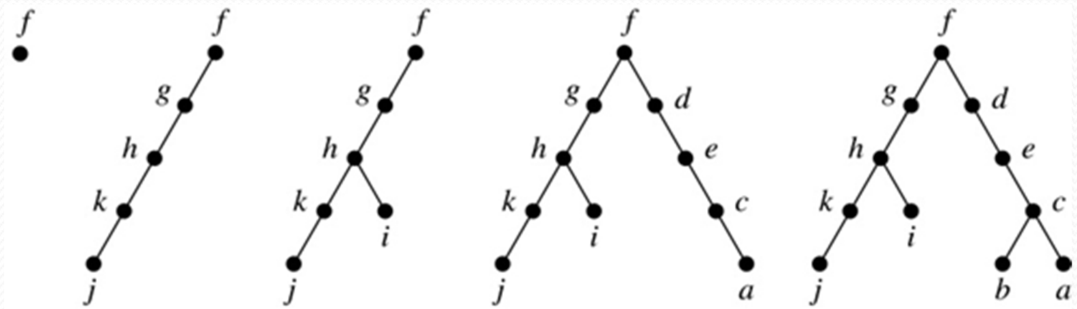
深度优先搜索

- 要使用深度优先搜索为连通的简单图构建生成树，首先需要任意选择图的一个顶点作为根。
- 通过连续添加顶点和边，形成从该顶点开始的通路，其中每条新边与通路中的最后一个顶点和通路中尚未存在的顶点关联。继续向该通路添加尽可能长的顶点和边。
- 如果通路穿过图的所有顶点，则由该通路组成的树是生成树。
- 否则，移回通路中最后一个顶点的前一个顶点，如果可能，从该顶点开始形成一条新通路，并穿过尚未访问的顶点。如果无法执行此操作，则继续向前移动一个顶点。
- 重复此过程，直到所有顶点都包含在生成树中。

深度优先搜索

例：用深度优先搜索找到图中的一颗生成树。





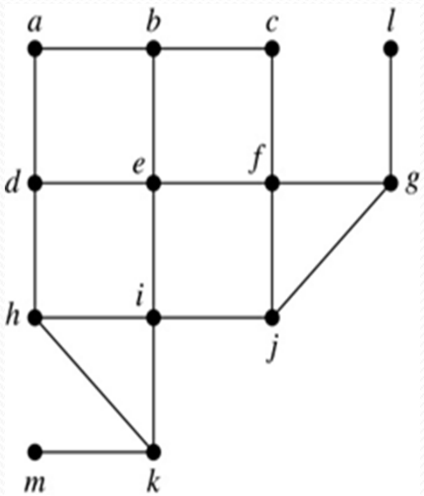
(a) (b) (c) (d) (e)

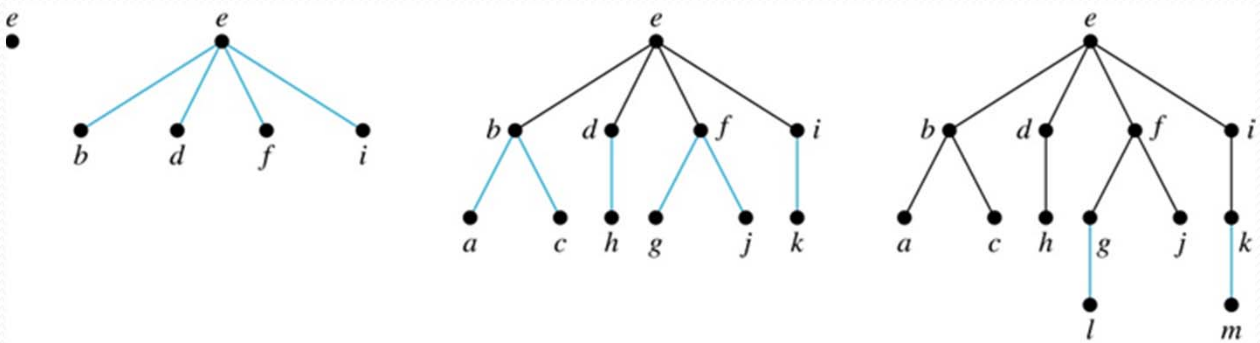
宽度优先搜索

- 使用宽度优先搜索来构造生成树，我们首先从图中任意选择一个顶点作为根。
- 然后，添加所有与这个顶点相关联的边。这个阶段新添加的顶点称为第一层的顶点。
- 下一步按顺序访问第一层上的每个顶点，只要不产生简单回路，就将与这个顶点相关联的每条边添加到树中。任意排序第一层的每个顶点的孩子。这样就产生了树在第二层上的顶点。
- 遵循相同的过程，直到已经添加了树中的所有顶点。因为边是有限的， 所以这个过程会终止。

宽度优先搜索

例：用宽度优先搜索找到右图的生成树。





最小生成树

Section 7.5

本节概要

- Prim算法
- Kruskal算法

最小生成树

- 假定图G是具有n个结点的连通图。对应于G的每一条边e，指定一个正数$C(e)$，把$C(e)$称作边e的权 (可以是长度、运输量、费用等）。G的生成树也成了一个具有树权$C(T)$的树，它是T的所有边权的和。
- 定义：在带权的图G的所有生成树中，树权最小的那棵生成树，称作最小生成树。

Prim算法

- 从图中权重最小的边开始构造生成树。依次向树中添加与当前生成树的顶点相关联、不构成回路、权重最小的边。直至生成树中的边数达到n-1。

ALGORITHM 1 Prim's Algorithm.

procedure *Prim*(G: weighted connected undirected graph with n vertices)
T := a minimum-weight edge
for $i := 1$ **to** $n - 2$
  e := an edge of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T
  $T := T$ with e added
return T {T is a minimum spanning tree of G}

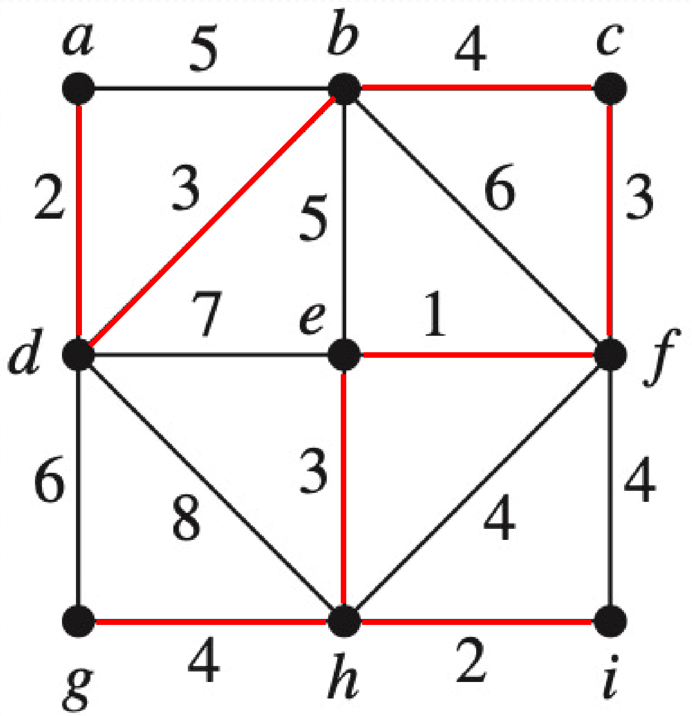
Kruskal算法

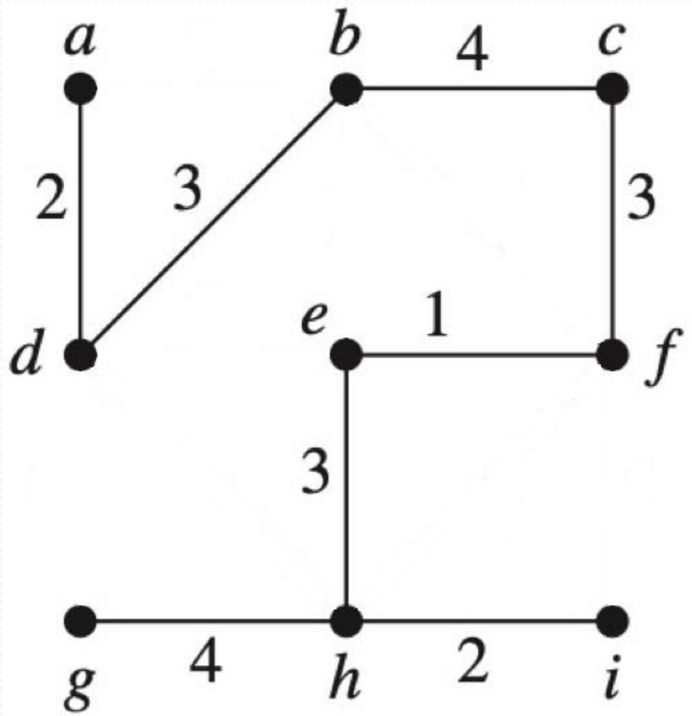
- 从图中权重最小的边开始构造生成树。依次添加与当前生成树不构成回路(避圈法)、权重最小的边。直至生成树中的边数达到n - 1。

ALGORITHM 2 Kruskal's Algorithm.

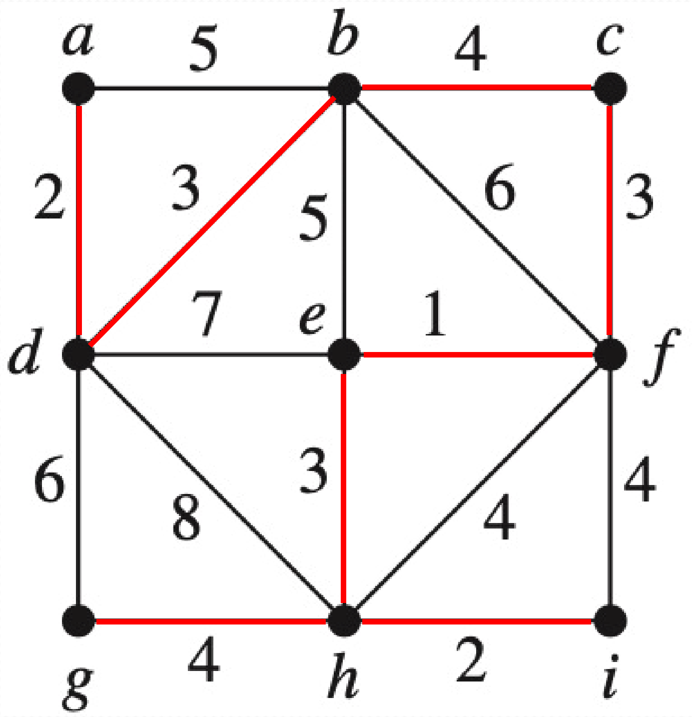
procedure *Kruskal*(G: weighted connected undirected graph with n vertices)
T := empty graph
for i := 1 **to** $n - 1$
  e := any edge in G with smallest weight that does not form a simple circuit
    when added to T
  T := T with e added
return T {T is a minimum spanning tree of G}

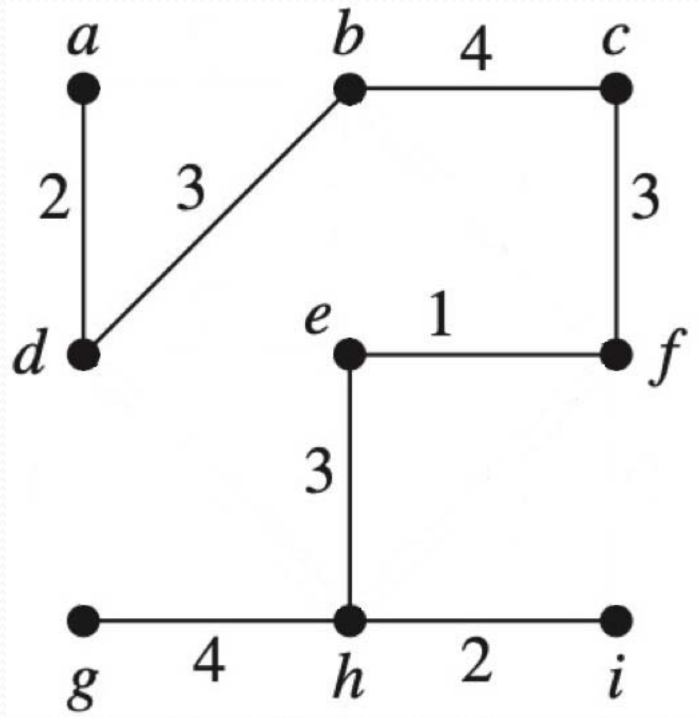
Prim算法举例





Kruskal算法举例





树的应用

Section 7.2

本节概要

- 前缀码
- 最优树与Huffman编码

前缀码

- 我们知道，在远距离通讯中，常常用0和1的字符串作为英文字母传送信息。因为英文字母共有26个，故用不等长的二进制序列表示26个英文字母时由于长度为1的序列有2个，长度为2的二进制序列有2^2个，长度为3的二进制序列有2^3个，依此类推，我们有
  - $2 + 2^2 + 2^3 + \ldots + 2^i \geq 26$
  - $2^{i+1} - 2 \geq 26, i \geq 4$

因此，用长度不超过4位的二进制序列就可表达26个不同英文字母。当使用不同长度的序列表示字母时，我们要考虑的另一个问题是如何对接收的字符串进行译码？

前缀码

- **定义：**前缀码
  - 给定一个序列的集合，若没有一个序列是另一个序列的前缀，该序列集合称为前缀码。
- 例如{000, 001, 01, 10}是前缀码，而{1, 0001, 000}就不是前缀码。
- 设前缀码：{000, 001, 01, 1}，有二进制序列00010011011101001，则该序列可译为000,1,001,1,01,1,1,01,001。

前缀码

- **定理**：任何一棵二叉树的树叶可对应一个前缀码。
- 证明：给定一棵二叉树，从每一个内点引出两条边，对左侧边标以0，对右侧边标以1，则每片树叶可以标定一个0和1的序列，它是由树根到这片树叶的通路上各边标号所组成的序列，显然，没有一片树叶的标定序列是另一片树叶的标定序列的前缀，因此，任何一棵二叉树的树叶可对应一个前缀码。
- **定理**：任何一个前缀码都对应一棵二叉树。

最优树

- 在前缀码的基础上我们进一步想到：由于字母使用的频繁程度不同，为了减少信息量，人们希望用较短的序列表示频繁使用的字母。
- 为了优化前缀码的设计，我们介绍最优树的概念以及基于最优树的Huffman编码。

最优树

- **定义：**最优树
  - 在带权二叉树（每片树叶都有权重）T中，若带权为w_i的树叶，其通路长度（层次）为$L(w_i)$，我们把
    $$w(T)=\sum_i w_i L(w_i)$$
    称为该带权二叉树的权。在所有带权$w_1, w_2, \ldots, w_t$的二叉树中，$w(T)$最小的那棵树，称为最优树。

最优树

- **定理**：设T为带权$w_1 \le w_2 \le \ldots \le w_t$的最优树，则
  - 带权为w_1，w_2的树叶是兄弟。
  - 以树叶w_1，w_2为儿子的内点，其通路长度最长。

- **定理**：设T为带权$w_1 \le w_2 \le \ldots \le w_t$的最优树，若将以权重$w_1$，$w_2$的树叶作为儿子的内点改为带权$w_1 + w_2$的树叶，得到一棵新树$T'$，则$T'$也是最优树。

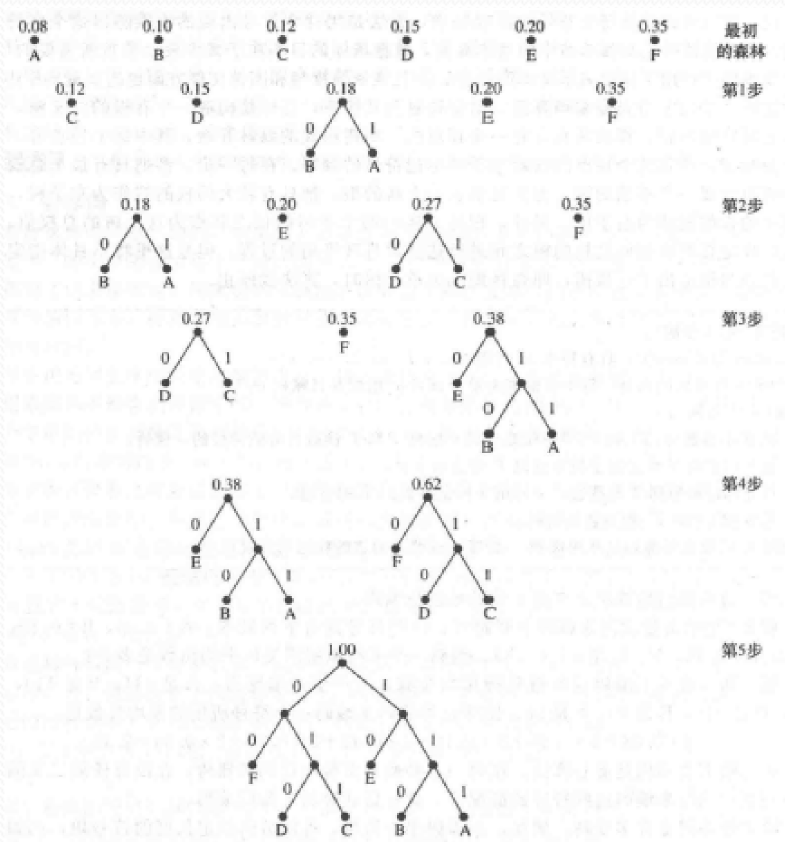
Huffman编码

- 用一个字符串中符号的出现频率作为输入，产生编码这个字符串的一个前缀码作为输出。Huffman编码是所有可能编码这些符号的前缀码中编码长度最小的前缀码。

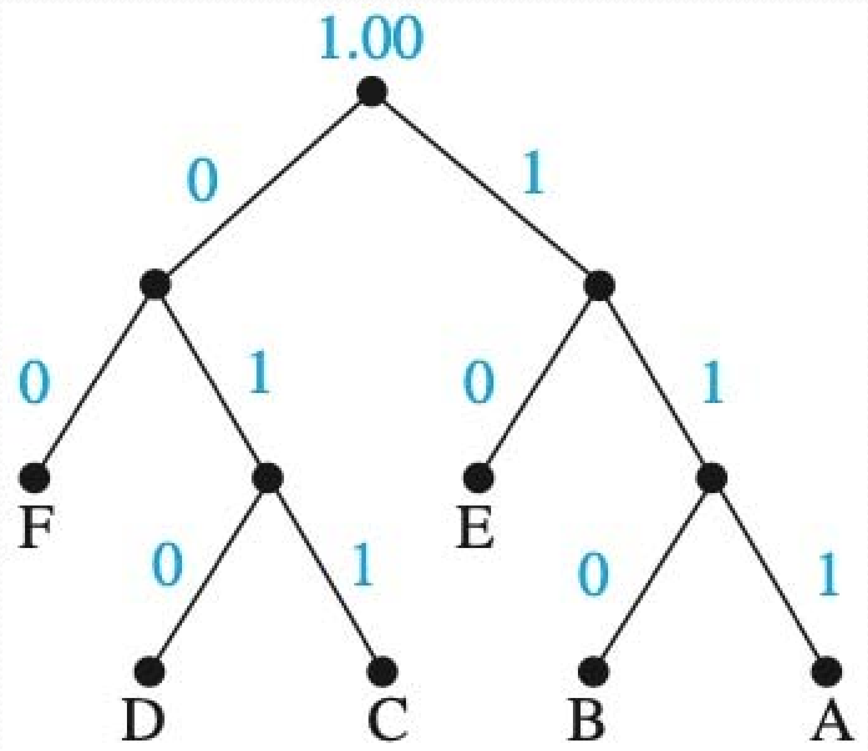
ALGORITHM 2 Huffman Coding.

procedure *Huffman*(C: symbols a_i with frequencies w_i, $i = 1, \ldots, n$)
F := forest of n rooted trees, each consisting of the single vertex a_i and assigned weight w_i
while F is not a tree
  Replace the rooted trees T and T' of least weights from F with $w(T) \geq w(T')$ with a tree having a new root that has T as its left subtree and T' as its right subtree. Label the new edge to T with 0 and the new edge to T' with 1.
  Assign $w(T) + w(T')$ as the weight of the new tree.
{the Huffman coding for the symbol a_i is the concatenation of the labels of the edges in the unique path from the root to the vertex a_i}

Huffman编码举例



Huffman编码举例



7.1~7.5作业

- 7.1
  - 1，6，8，9
- 7.2
  - 10，12
- 7.3
  - 4，6
- 7.5
  - 6，7