Applications of Trees

树的应用

树模型

- 树作为一种特殊的数据结构，作为一种数学模型，在很多领域得到了广泛的应用。从化学、生物、社会管理到信息科学等等。如：
- 1. 树状数据库 （如xml文件）
- 2. 组织机构
- 3. 计算机文件系统（目录树）
- 4. 其它应用等等

树应用

- 根树常常用来保存数据(如树状数据库),客观上就需要访问有序根树的每个结点来存取数据. (如XML文件里的数据的读取)

- 系统地访问有序根树的每个结点的过程称为树的遍历,其算法称为遍历算法.

- 想象WINDOWS操作系统中的Search, 那就是一个典型的目录树的遍历应用的例子.

树应用

- A lot of problems can be studied using trees:

1. How should items in a list be stored so that an item can be easily located? 二叉搜索树
2. What series of decisions should be made to find an object with a certain property in a collection of objects of a certain type? 为了寻找具有某种特征的对象，需要一系列的决策行为。决策树
3. How should a set of characters be efficiently coded by bit strings? 有效变长编码问题：前缀码与哈夫曼树

Binary Search Trees二叉搜索树

- **Primary goal**: to implement a searching algorithm that finds items efficiently when items are totally ordered。设计实现一种结构，来保存一系列的有序数据，并设计相应的快速查询算法。
- **Binary search trees (BST)**: sometimes called **ordered** or **sorted binary trees**, are a particular type of container: data structures (数据容器、数据结构) that store "items" (such as numbers, names etc.) in memory.
- BST结构：
  - each child of a vertex is designated as a right or left child; each internal vertex has only one left child or right child;

    每个内结点的子结点最多为两个，一个是左结点，一个为右结点；
  - Vertices are assigned keys so that the key of a vertex is larger than the keys of all vertices in its left subtree, and smaller than the keys of all vertices in its right subtree. 每个结点都指定一个关键值，该值不小于其左子树的所有结点的值；不大于其所有右结点的值；

Binary Search Trees二叉搜索树

- They allow fast lookup, addition and removal of items,…
- 搜索树的两个问题：
  - 如何维护搜索树
  - 如何从搜索树里查找数据
- 数据结构课里将会学习有关BST的递归算法的实现、复杂度等问题，这里不重复讲授、不做要求。
- 自己看教材中的例子…

Decision Tree 决策树

- Rooted trees can be used to model problems in which a series of decisions leads to a solution. This kind of model is called “decision tree” 用根树建立决策模型
- 决策树：实际上就是一棵m元根树。它的所有的叶结点就是最终可能存在的决策结果；树的高度h代表着到达获得最终决策结果所需要的最多的决策次数；树的每个内结点的每一棵子树代表着下一步可能的决策；
- 如果一棵决策树是二叉树，那么说明什么？
- 决策树在机器学习和人工智能中应用很多；有些程序在编写实现前，也经常画决策树，确定每一种情况下的走向。

Decision Tree Examples

E.g.： Suppose there are 7 coins, all with the same weight, and a counterfeit coin that weighs less than the others. How many weighings are necessary using a balance scale to determine which of the eight coins is the counterfeit one? Give an algorithm for finding this counterfeit coin. 7枚同重真币与1枚较轻的伪币的寻找判别

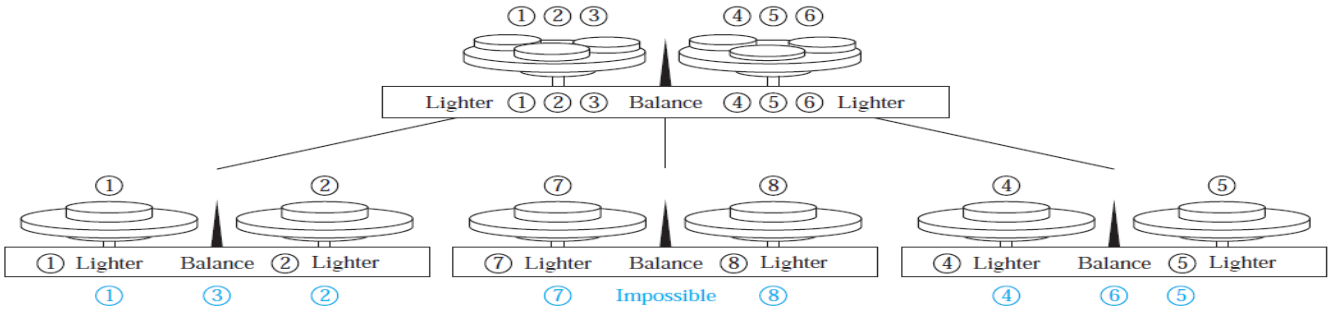


FIGURE 3 **A Decision Tree for Locating a Counterfeit Coin. The counterfeit coin is shown in color below each final weighing.**

EXAMPLE 4 We display in Figure 4 a decision tree that orders the elements of the list a, b, c. ◀

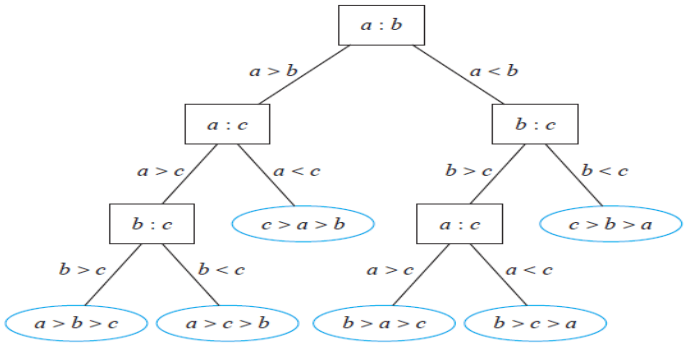
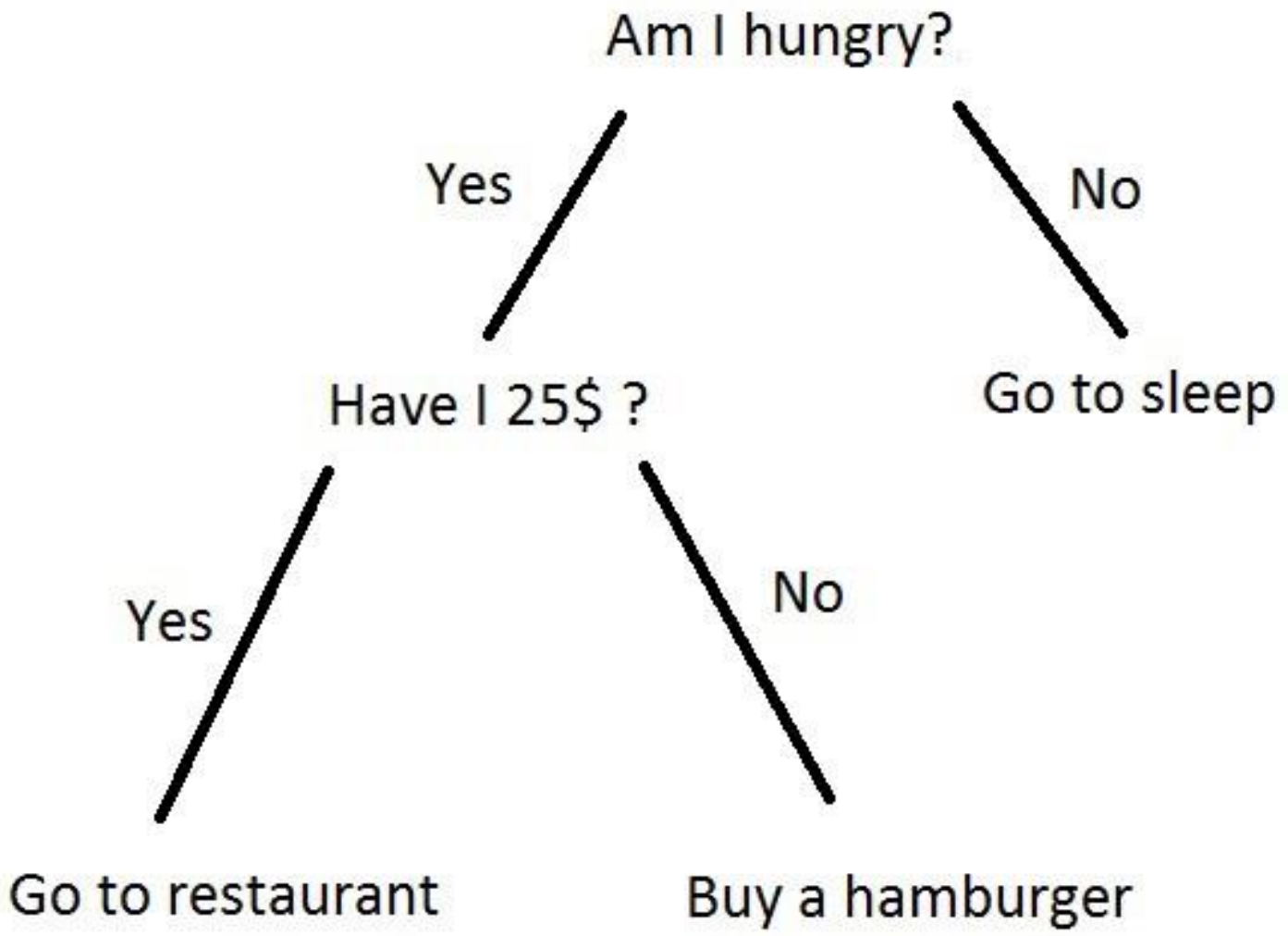


FIGURE 4 **A Decision Tree for Sorting Three Distinct Elements.**

BinaryDecision Tree Example



树的应用

- 搜素树、决策树、树的遍历等等都会在数据结构里继续学习

树的遍历

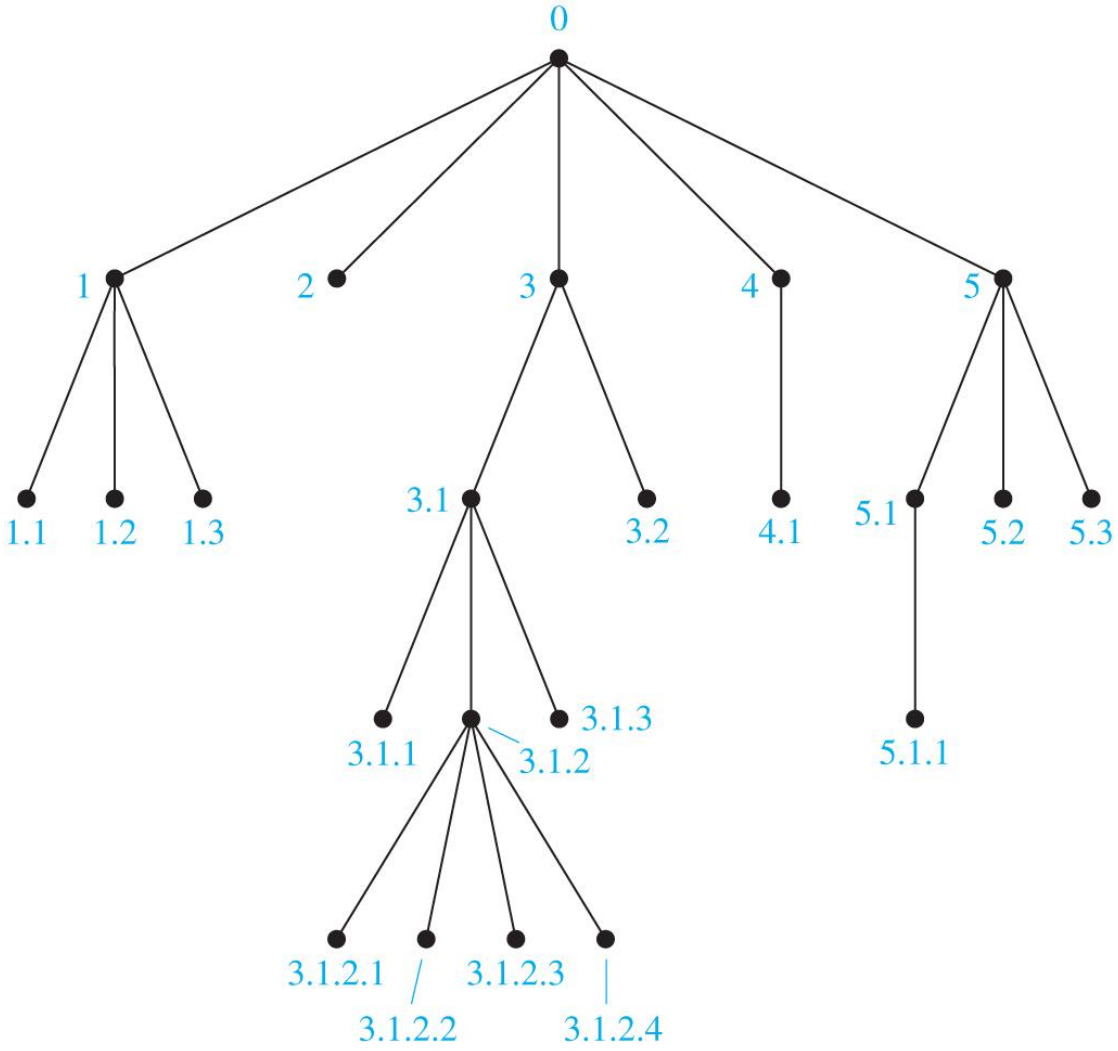
- **应用对象：**顺序地访问有序根树的所有结点。目的是能访问到保存在根树每个结点上的数据。

- 有序根树的通用地址系统

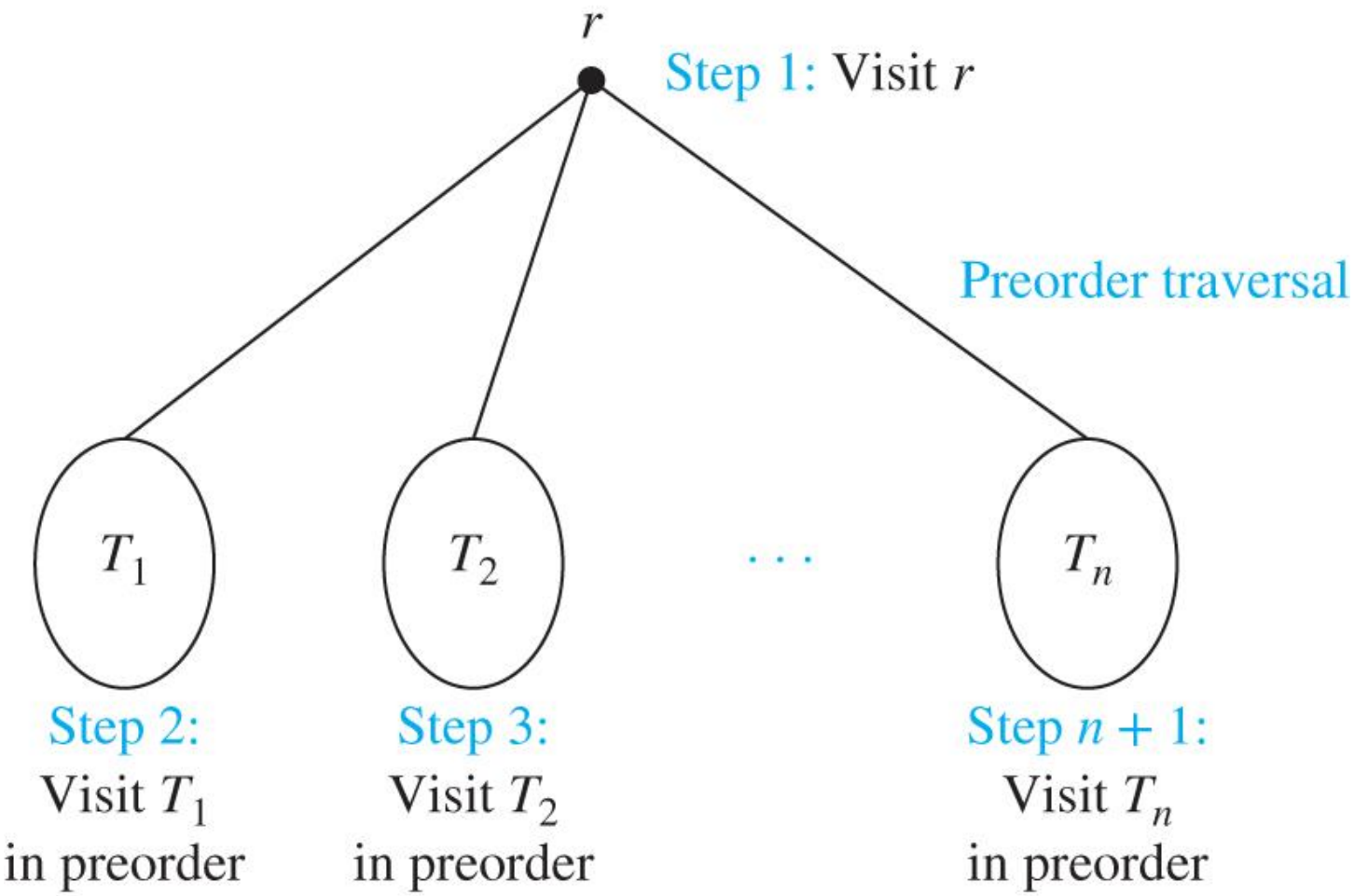
- 有序根树遍历的常用的递归算法
  - 前序遍历（先根算法）：先访问根
  - 中序遍历（中根算法）：中根访问
  - 后序遍历（后根算法）：最后访问根

有序根树的通用地址系统

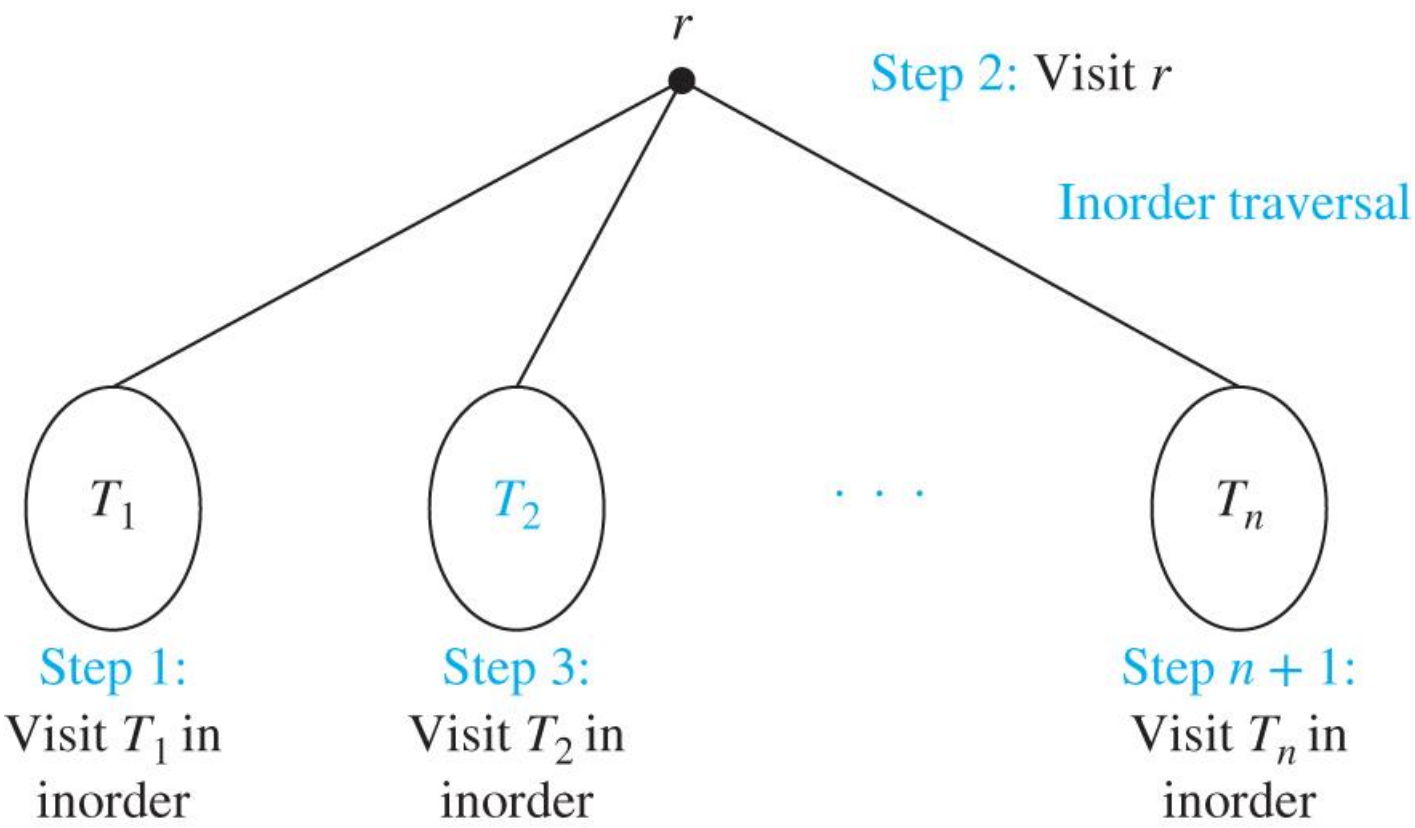
一种可以完全排序根树所有结点的方法。



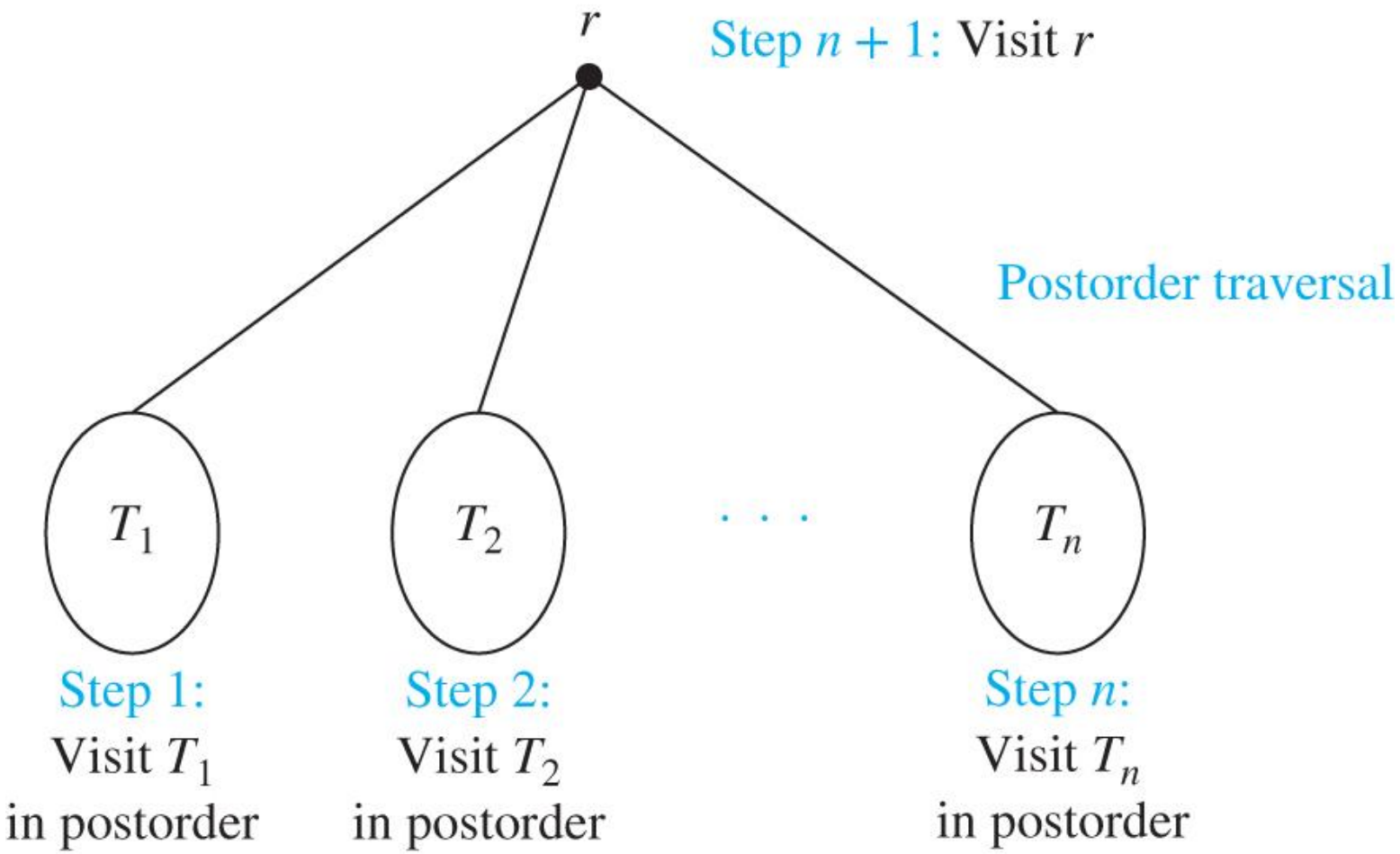
前序遍历（先根算法）：先访问根



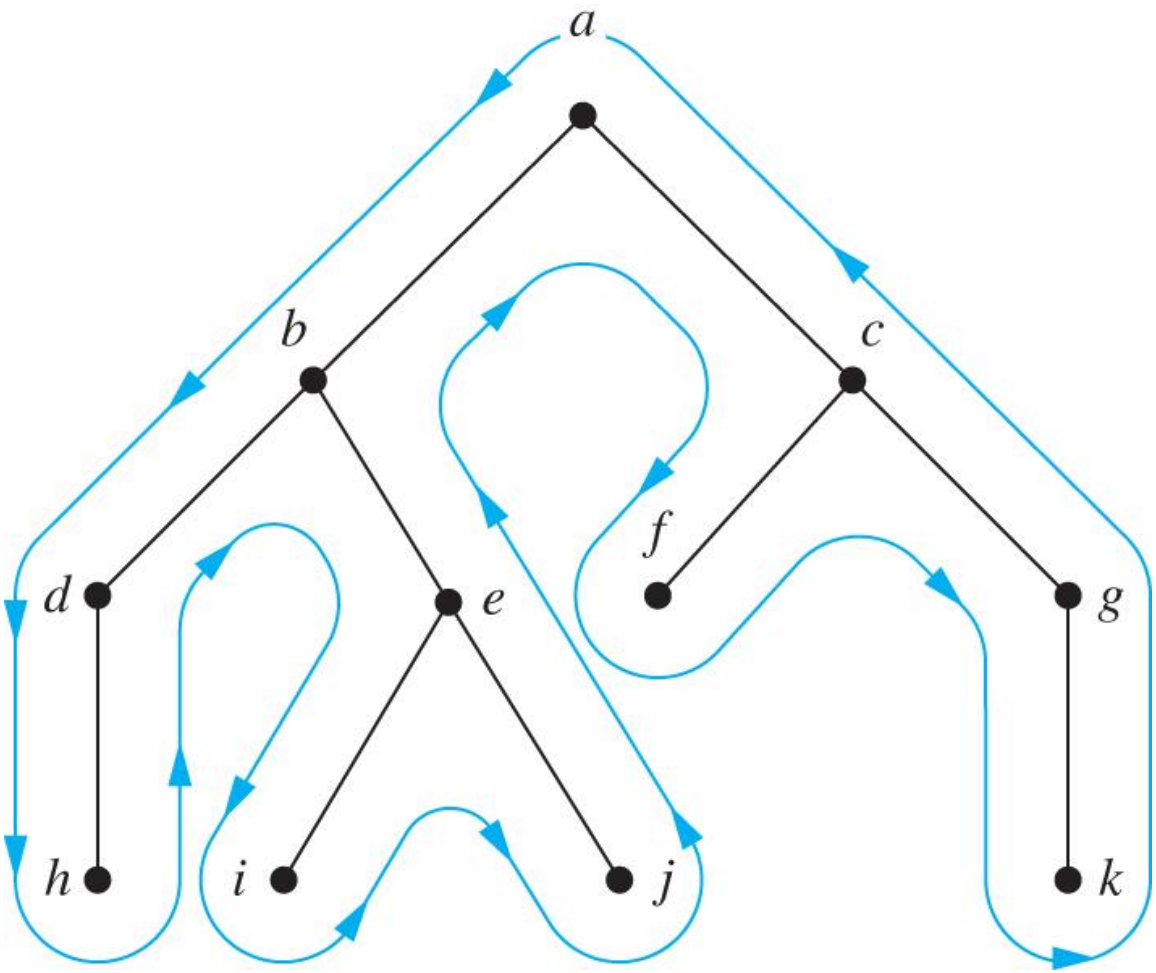
中序遍历（中根算法）：中根访问



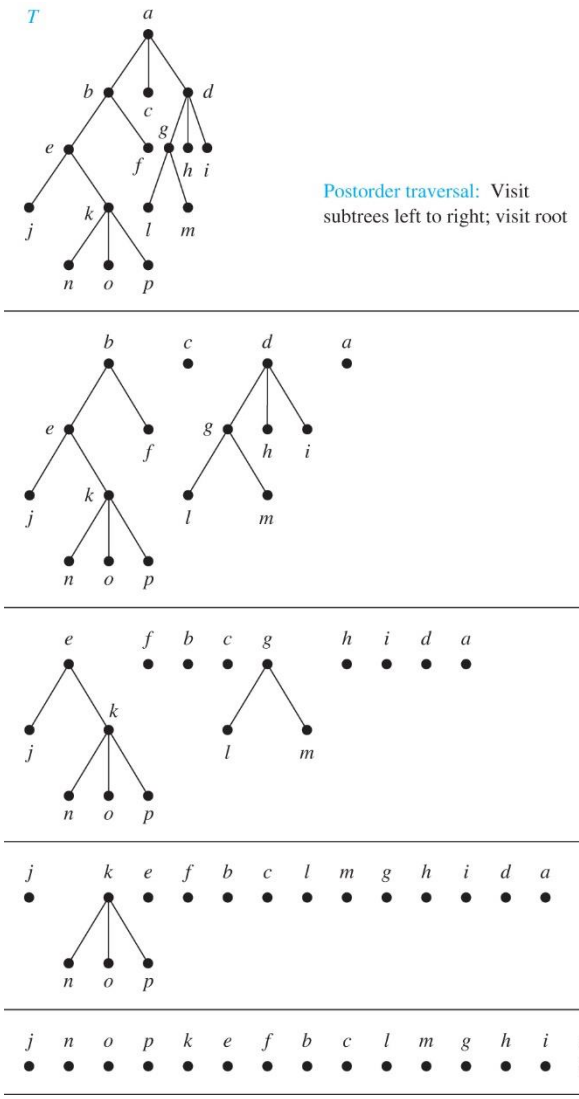
后序遍历（后根算法）：最后访问根



前序遍历（先根算法）：先访问根举例



后序遍历（后根算法）举例


T
Postorder traversal: Visit subtrees left to right; visit root

Prefix Code(前缀码)--Huffman code
树的应用举例

如何设计编码，使得存储和通讯更简单，更省时。
以26个英文字母为例进行编码（用二进制串表示相关数据）。

定长编码：每个字符（或者对象）用相同长度的二进制串表示。

那么26个字母需要用多少位才能区分开来？
每个字母用5位长度二进制数编码才可以完全区分开来。

如何设计编码，才能使得码长较短甚至最短（有利于存储和通讯）？

变长编码：用短编码实现使用频率高的字母编码，用较长的码对实现使用频率低的字母编码。

Prefix Code(前缀码)--Huffman code

变长编码可以使得平均码长较短，节省资源。但变长编码如何保证编码和解码的唯一性和正确性？

为了保证没有位串对应着多个字母序列（保证唯一性、正确性），设计编码集时，可以令一个字母的编码位串永远不会出现在另一个字母位串的开头部分（不是前缀）。
具有这种性质的编码集就称为前缀码。

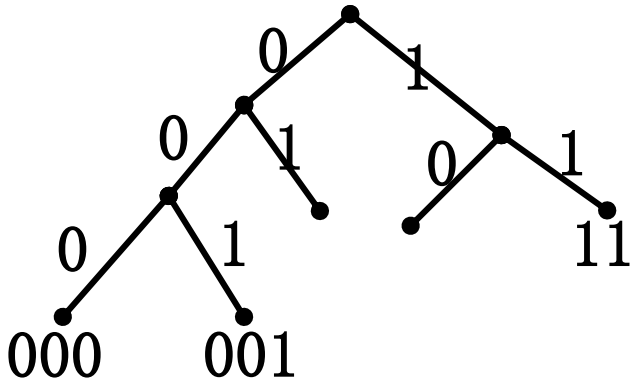
例如：010， 01011 这两个编码就不符合上面要求
但{010， 0110， 001}就是前缀码

Prefix Code(前缀码)--Huffman code

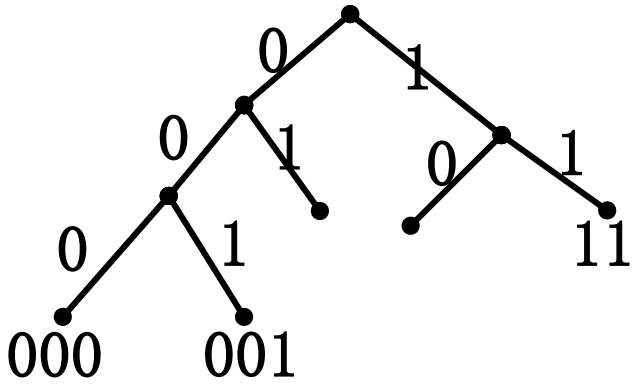
前缀码定义：一个编码集中的一个字母或符号对应的的01串永远不会出现在另一个字母符号对应的01串的开头(也即不能是前缀)

设$\{S_1, S_2, \cdots S_n\}$是一个前缀码集合（代表着n个不同的字母或者字符或者n个对象），那么任一个S_i都不会是另一个S_j的前缀；其中每个S_i都是01位串 （二进制数据）。

由01串形成的前缀码与二元树的关系如下图所示(注意观察叶结点对应的二进制串的特征)：



为什么二叉树的所有叶结点的编码集是前缀码?



1. 观察分析叶结点的二进制码的形成
2. 二叉树叶结点的非空子集对应的二进制码集是一个前缀码

Prefix Code(前缀码)--Huffman code

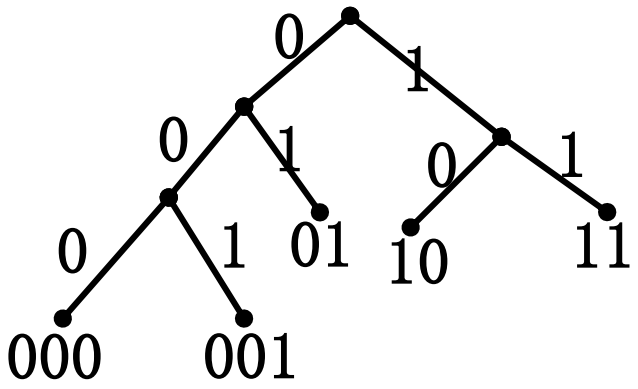
- 问题：那么如何利用前缀码的理论构造有效的变长编码集？涉及到最优树（ Huffman 树的问题）

- 假定设计好了一个前缀码集合后，对字母进行了编码。那么怎么解码？
- 1. 查表法
- 2.构造一颗二叉树，使得该树的叶结点对应于前缀码集合的元素（二进制串）
- 3. 二叉树的构造方法
- 4. 构造了二叉树后，利用二叉树进行解码。给一组01前缀码, 如何构造和使用二元树来分析解码一段01串？

利用二叉树解码

1. 给定前缀码集$\{S_1, S_2, \cdots S_n\}$，构造一颗二叉树，其所有叶结点对应的二进制码就是这个前缀码集。

2. 已知前前缀码对应的二叉树，如何解码？



Huffman Tree Coding哈夫曼编码

- 那么如何设计一套合理的前缀码，使得平均码长最短？需要建立一个目标函数，然后利用目标函数解决优化问题。

- **思考问题**：十个人排队取水, 每个人的桶大小不一样, 那么怎么安排队列, 使得总的等待时间最小?

- **想像一下**：在计算机文件目录系统中存储文件时，把最常用的和不常用的文件，怎么存放有利于查找应用？

最优树问题(哈夫曼树、哈夫曼编码)

- **问题**：给定一组权$w_1,w_2,w_3,\ldots,w_t$, 是否存在一棵二元树，其t片树页分别带上这些权，并且使得权与路长的积之和最小。**目标函数如下：**

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

- 应用举例：将一组信息按树的结构保存起来，使得每个叶结点上有一条信息。 从根到树叶的长度L表示检索这个位置上的信息所花的时间(或编码长度)，W表示该条信息的访问频度。那么如何组织这棵树，才能使信息的平均检索时间最短？

哈夫曼树（最优树）与Huffman Coding

- 使得下面目标函数值最小的二叉树，称为给定组权 $w_1,w_2,w_3,\ldots,w_t$ 下的最优二叉树（Huffman Tree)

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

- **Huffman coding :**
  - An algorithm that takes as input the frequencies (which are the probabilities of occurrences) of symbols in a string,
  - and produces as **output a prefix code** that encodes the string using the fewest possible bits, among all possible binary prefix codes for these symbols.

- 可以通过构造Huffman Tree来实现**Huffman coding。**

哈夫曼编码 （重要的算法）

- **利用上面的想法，合理设计前缀码集合。**

1：首先分析有多少需要编码的字符

2：统计它们的使用频率 $w_1,w_2,w_3,\ldots,w_t$ 并排序

3：设计构造**最优二叉树**，前缀码集合的元素个数对应着相应的二叉树的叶结点的个数；

4：最优二叉树的所有叶结点对应的编码集合就是能满足要求，使得平均码长最短的前缀码集合。

实际上，就是去构造一颗**哈夫曼树（最优树），实现 Huffman Coding.**

最优树与哈夫曼编码

- 在哈夫曼编码输入频率后，实际上就是去寻找一颗最优树，使得最优树的叶结点对应的频率就是输入的各个频率；
- 而这颗最优树的叶结点对应的那些二进制串就形成所要的能够使得平均码长最短的一组前缀码。
- 平均码长最短是用下面的目标函数来评估：

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

- 下面的两个定理保证了最优树的存在，也提供了寻找最优树的思路。

最优树有关定理

- 定理1：设T是带权$w_1<=w_2<=\ldots<=w_t$的最优树，则：
  - (1)T是一棵正则二元树(full binary tree)；
  - (2)叶结点V_{w1}为T中到根的距离最远的结点. (也即权最小的离根最远,**处于最高层的叶结点**)
  - (3)存在一棵带权w_1，w_2，…，w_t的最优树，使得带权w_1和带权w_2的结点为兄弟结点。

- 需要证明的几个点：

  (1) 存在性 (2) why full?

  (3) 为什么V_{w1}离根最远？(4)定理第3部分结论的理由？
- 思考问题:那么对应权最大的叶结点是不是就应该离根最近的叶结点?

最优树有关定理2

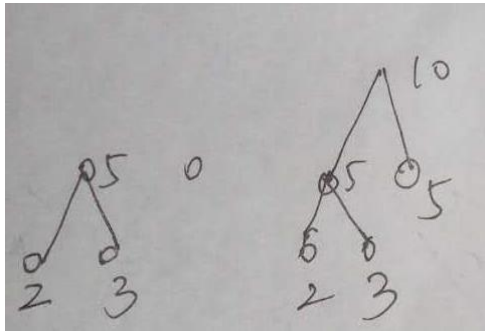
- 定理2：假设T是带权$w_1<=w_2<=\ldots<=w_t$的最优树。如果带权w_x和带权w_y的两片树叶V_{wx}和V_{wy}是兄弟. 在T中,用一片带权W_x+W_y的树叶来替代以V_{wx}和V_{wy}及其它们的父结点所组成的子图,得到一棵新树T_1,那么T_1是带权$W_{i1},W_{i2},\ldots,W_x+W_y, \ldots, W_{it-2}$的最优树.

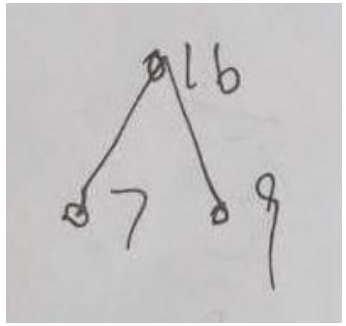
- 总结构造最优树的方法(哈夫曼算法)

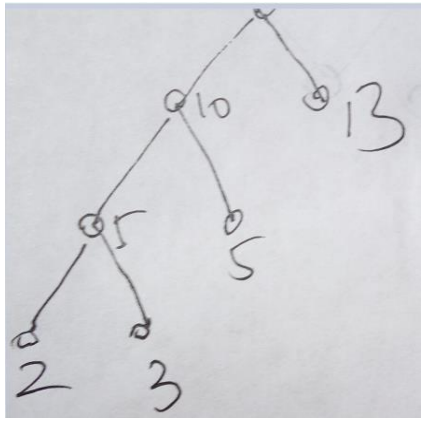
最优树构造举例

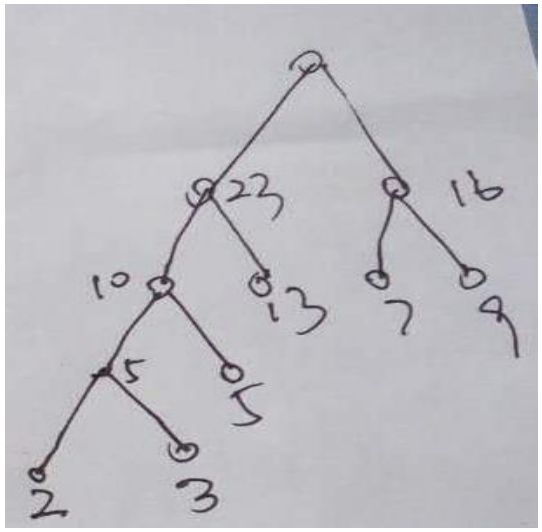
- 举例说明: 给定权{2,3,5,7,9,13}, 构造一棵最优树.
- 解：假设T是带给定权2,3,5,7,9,13的最优树。

则有一棵最优树，在这棵树中，2,3对应的结点是兄弟。 2+3=5， 在T中，将2,3对应的结点的父结点标记为权5对应的结点，再去掉2，3两个结点，则剩下的树为带权5，5，7，9，13的最优树。如此类推下去，最后构造出完整的最优树T









练习

- 7.2 节 T19 (b), (d) T21
- 课外自己做做T14, 但不用做到练习本上