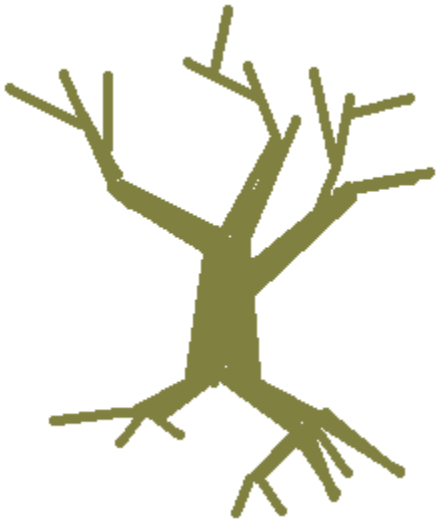
Trees 树

Discrete Mathematics and Its Applications

Kenneth H. Rosen

Trees

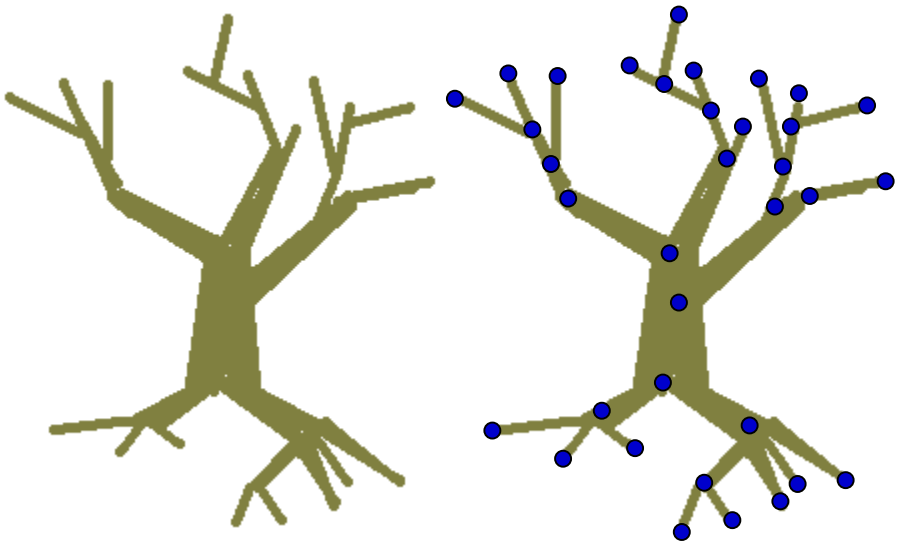
A very important type of graph in CS is called a *tree*:



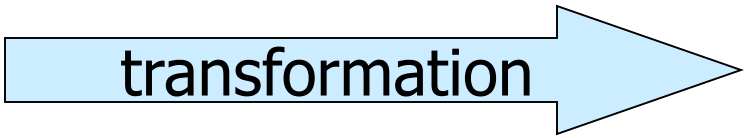
Real Tree

Trees

A very important type of graph in CS is called a *tree*:

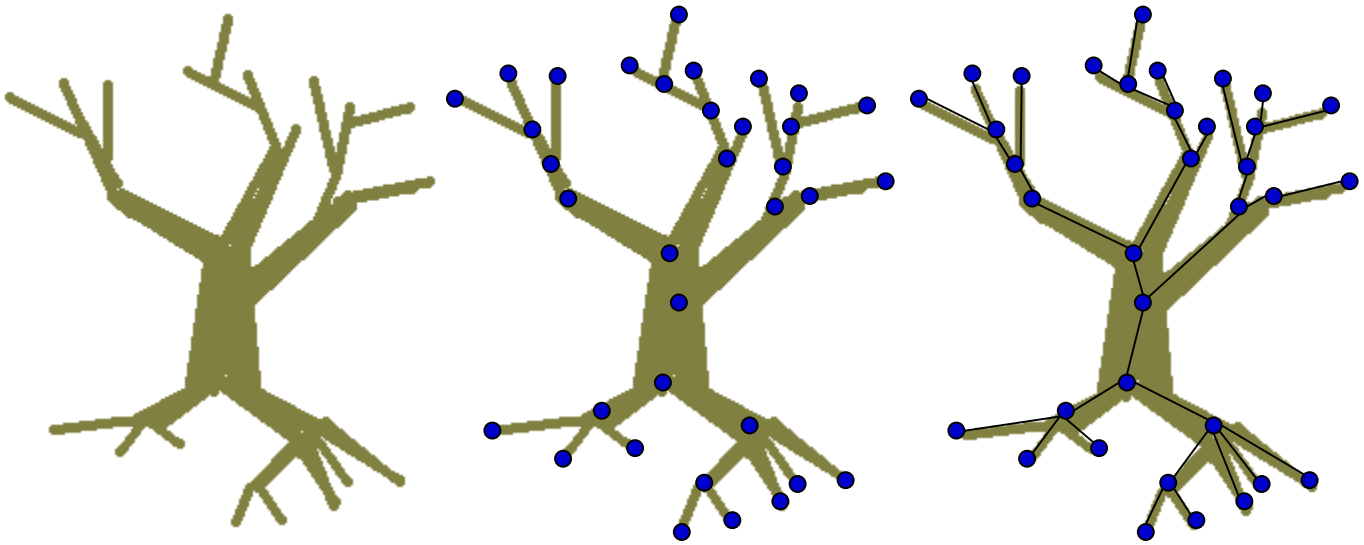


Real Tree

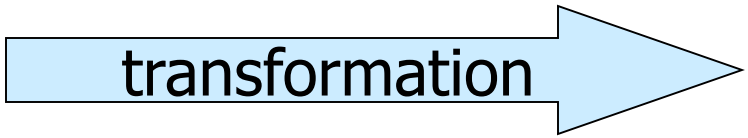


Trees

A very important type of graph in CS is called a *tree*:

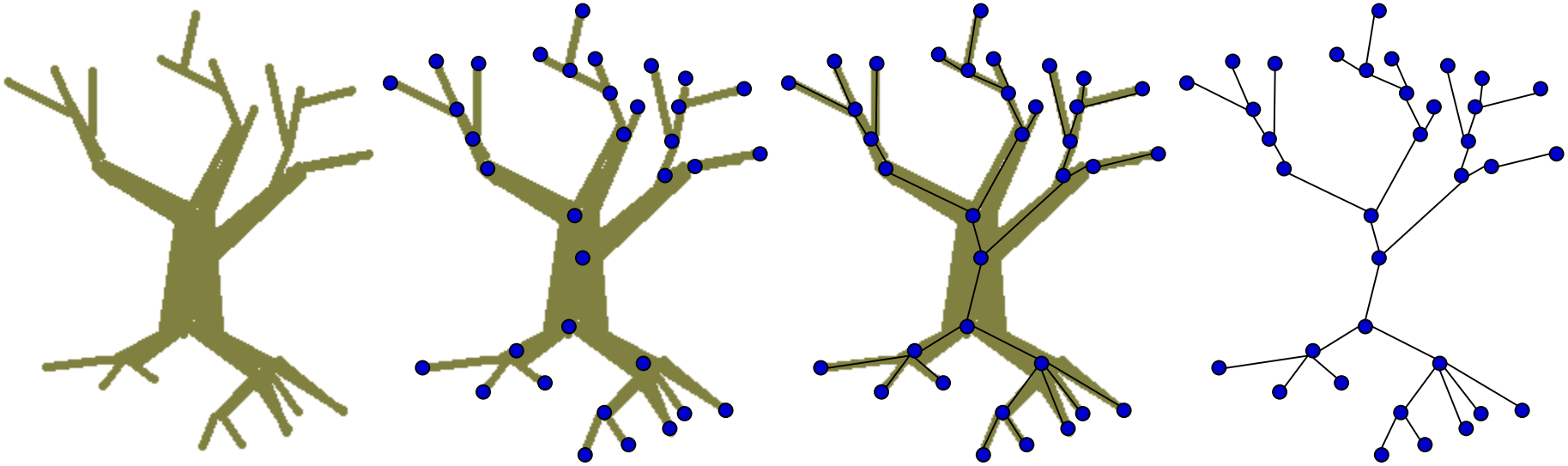


Real Tree



Trees

A very important type of graph in CS is called a *tree*:



Real Tree

transformation

Abstract Tree

Tree

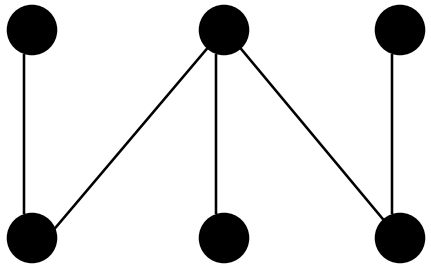
Def 1. A tree is a connected undirected graph with no simple circuits.

没有简单回路的连通无向图称为树，也称为无向树

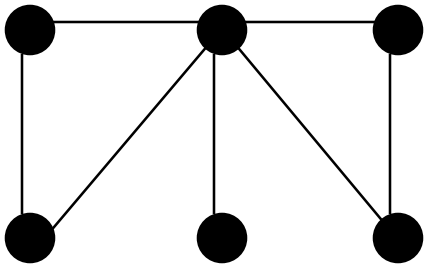
无向树中, 度为1的结点称为叶结点，简称树叶。

Which graphs are trees?

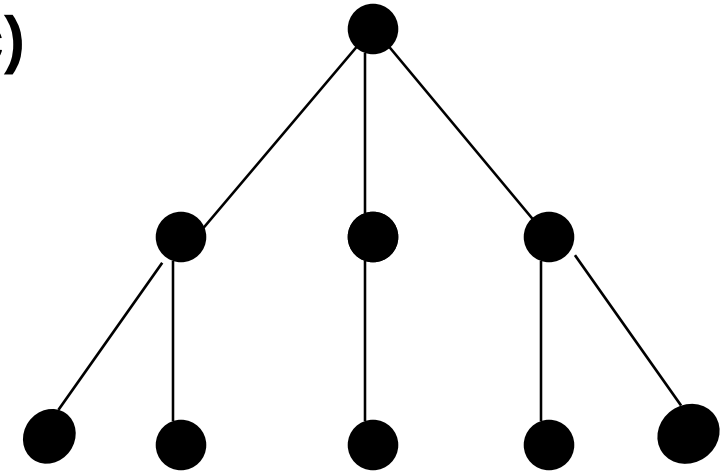
a)



b)



c)



树的基本性质

Theorem 1.一个无向图为树当且仅当任意两个不同结点之间有唯一的一条简单路 An undirected graph is a tree if and only if there is a unique simple path between any two distinct vertices.

Why?

Theorem 2. A tree with N vertices has just N-1 edges.

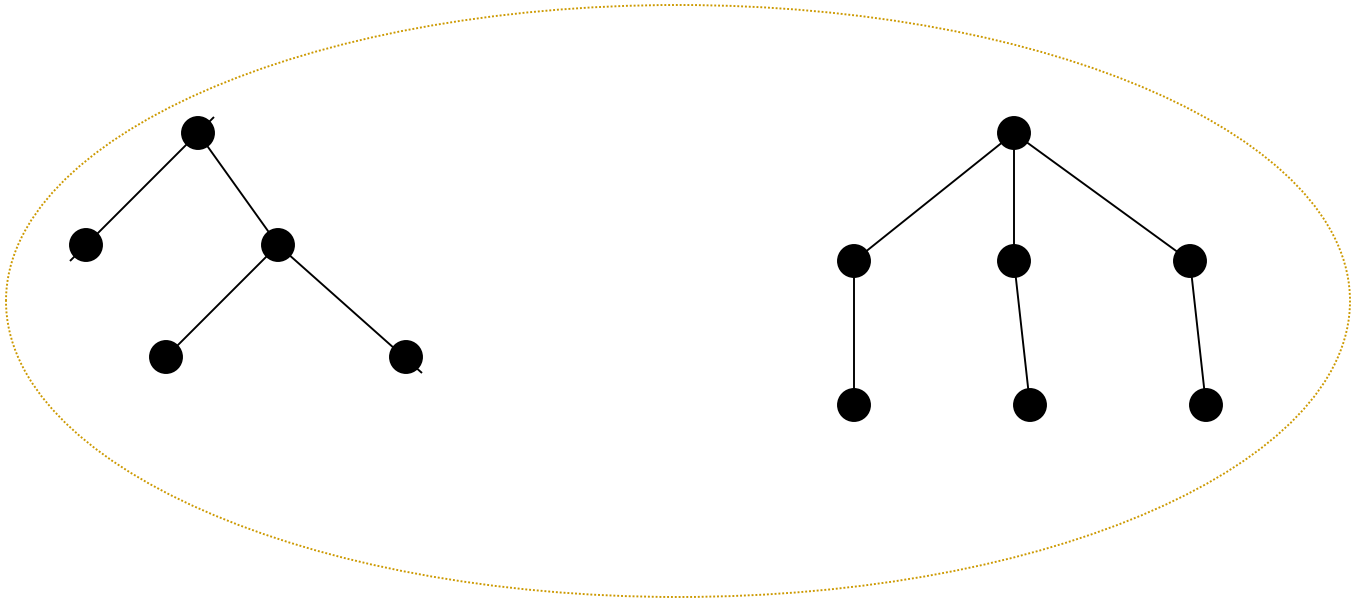
(important result)

For any (n,m)树必然有：n=m+1

Note: Using induction to prove this theorem （第2数学归纳法证明）.

Forest 树林

Graphs containing no simple circuits that are not connected, but each connected component is a tree.

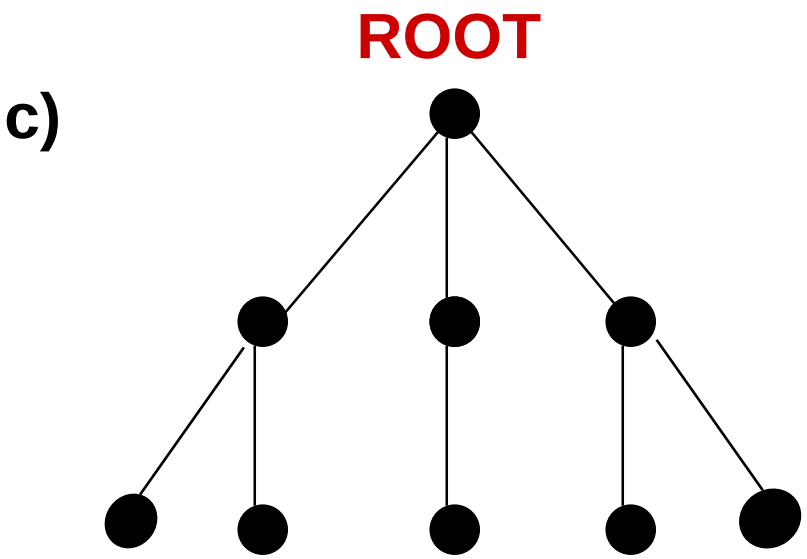


Root Tree 根树

- Think about the directory tree
- Organization tree

Specify a vertex as root
指定一个结点为根

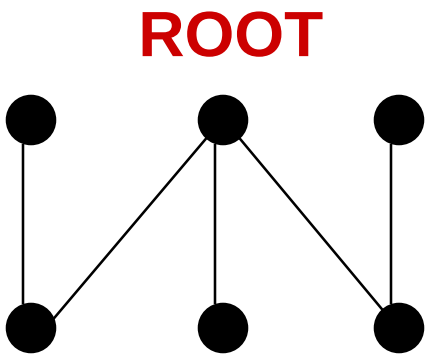
Then, direct each edge away from the root.
给每条边指定一个方向：离开根的方向



Specify a root.

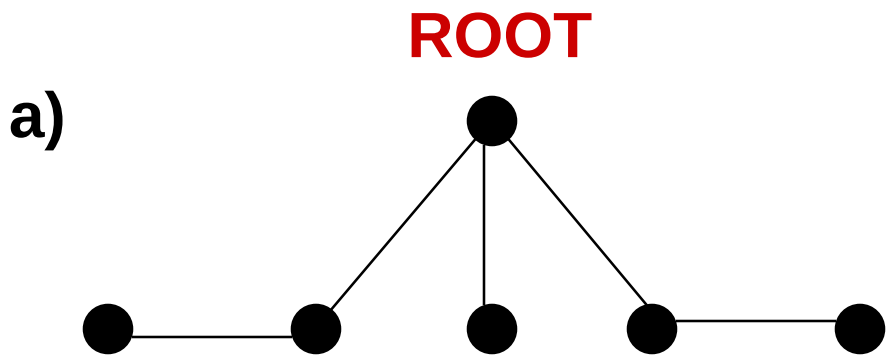
Then, direct each edge away from the root.

a)



Specify a root.

Then, direct each edge away from the root.

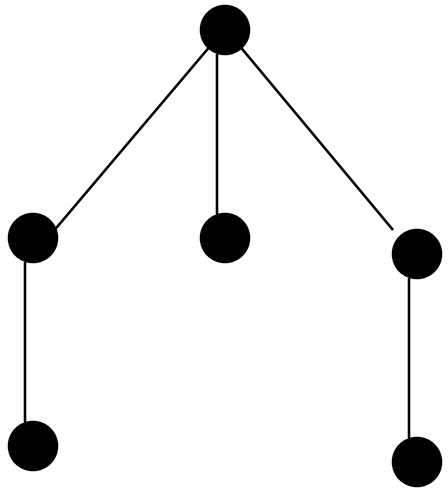


Specify a root.

Then, direct each edge away from the root.

ROOT

a)

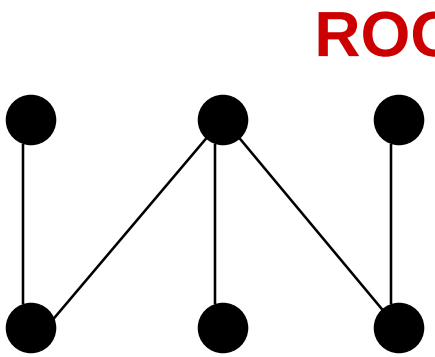


The result is a directed graph, called a rooted tree 一个有向图

What if a different root is chosen?

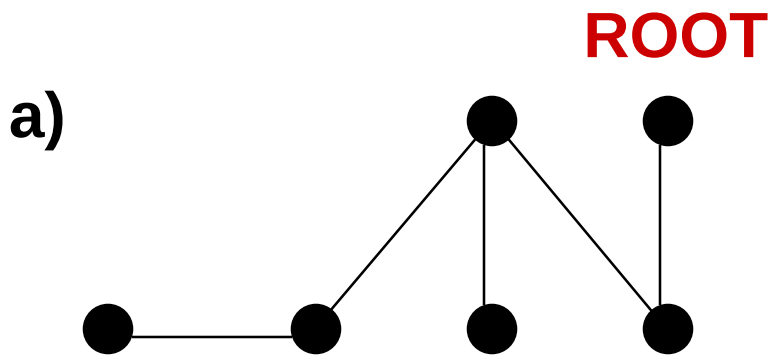
Then, direct each edge away from the root.

a)



What if a different root is chosen?

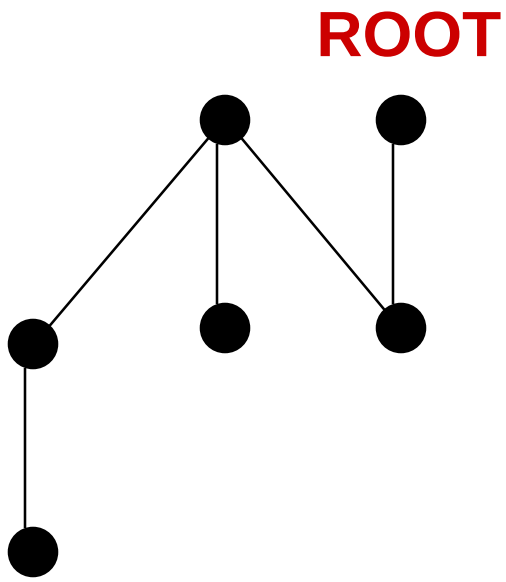
Then, direct each edge away from the root.



What if a different root is chosen?

Then, direct each edge away from the root.

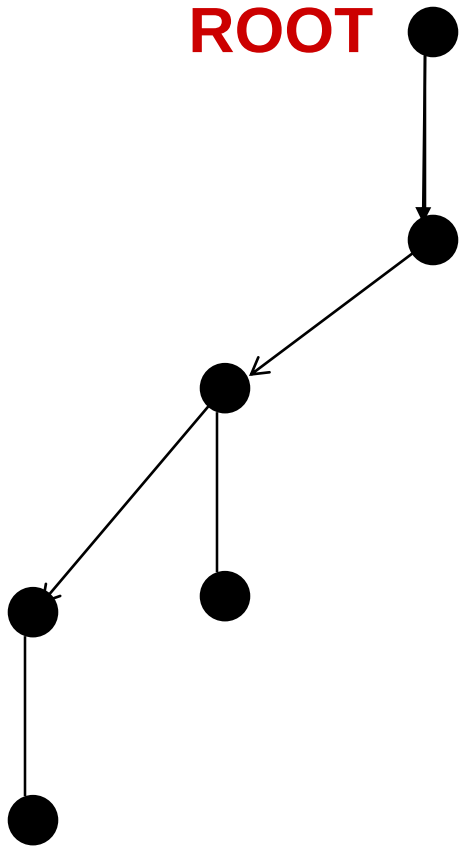
a)



What if a different root is chosen?

Then, direct each edge away from the root.

a)

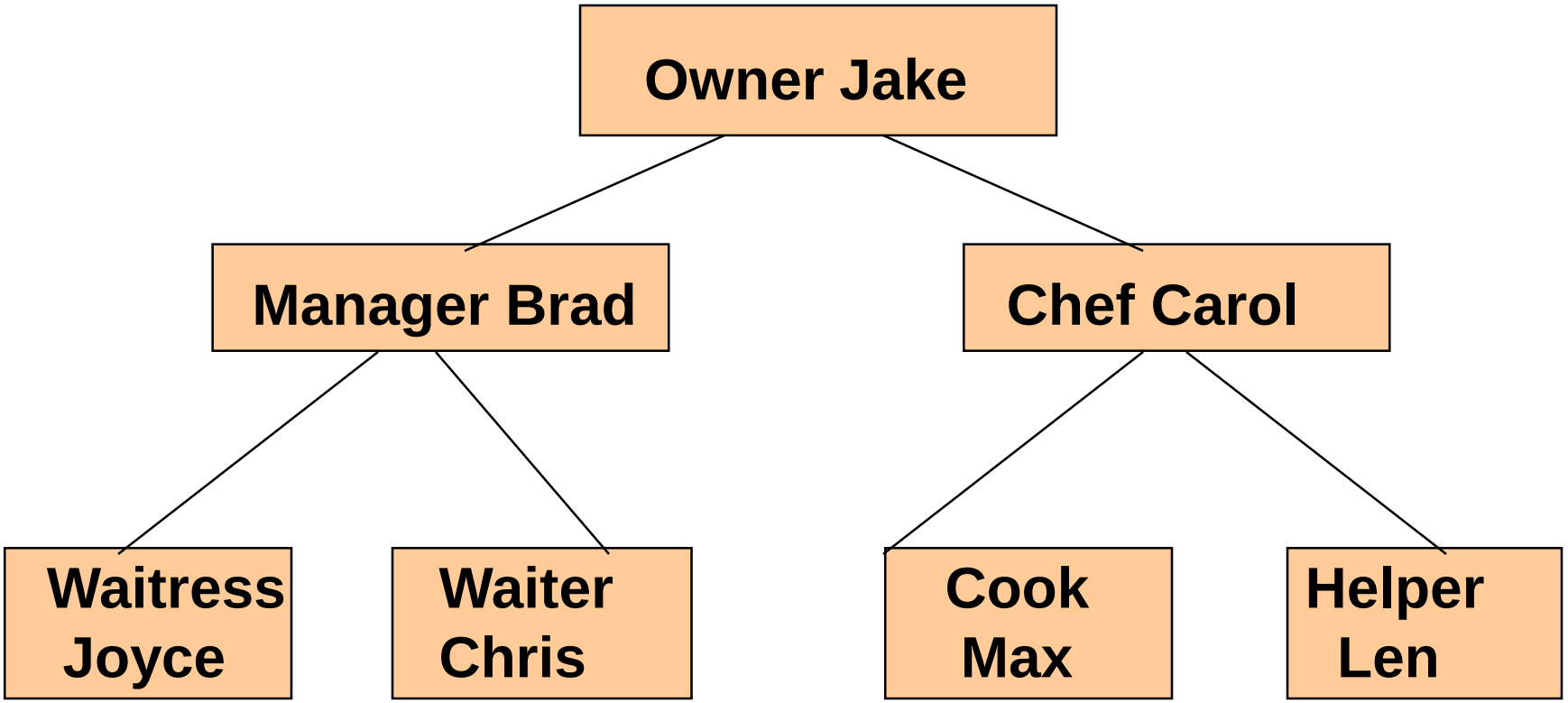


A different rooted tree

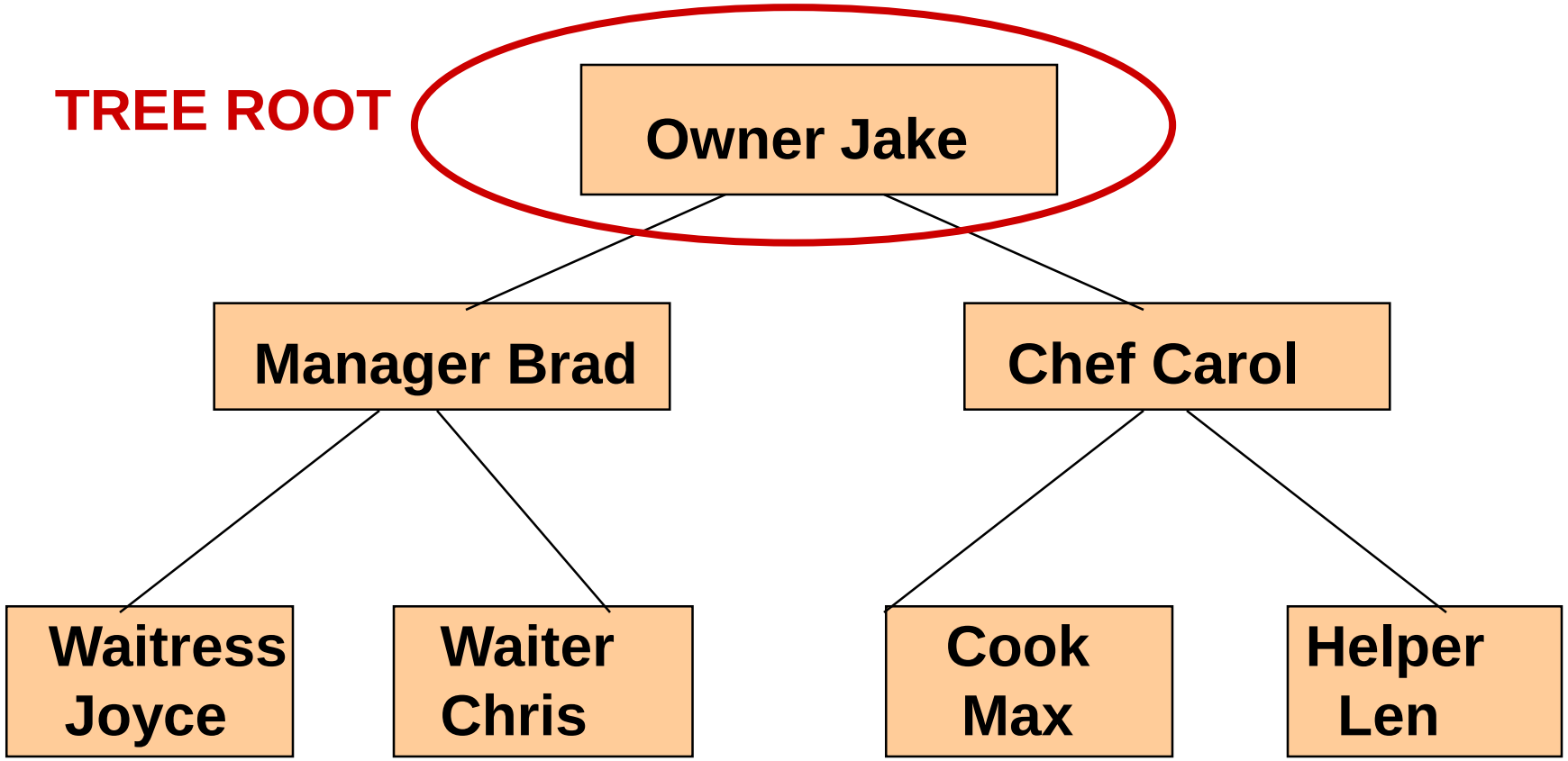
根树相关概念

- 根树：是有向树
- 根：入度为0的点（唯一的）
- 所有其它结点入度为1；
- 叶结点：入度为1出度为0的结点
- 除叶结点外，根树所有结点出度都大于0

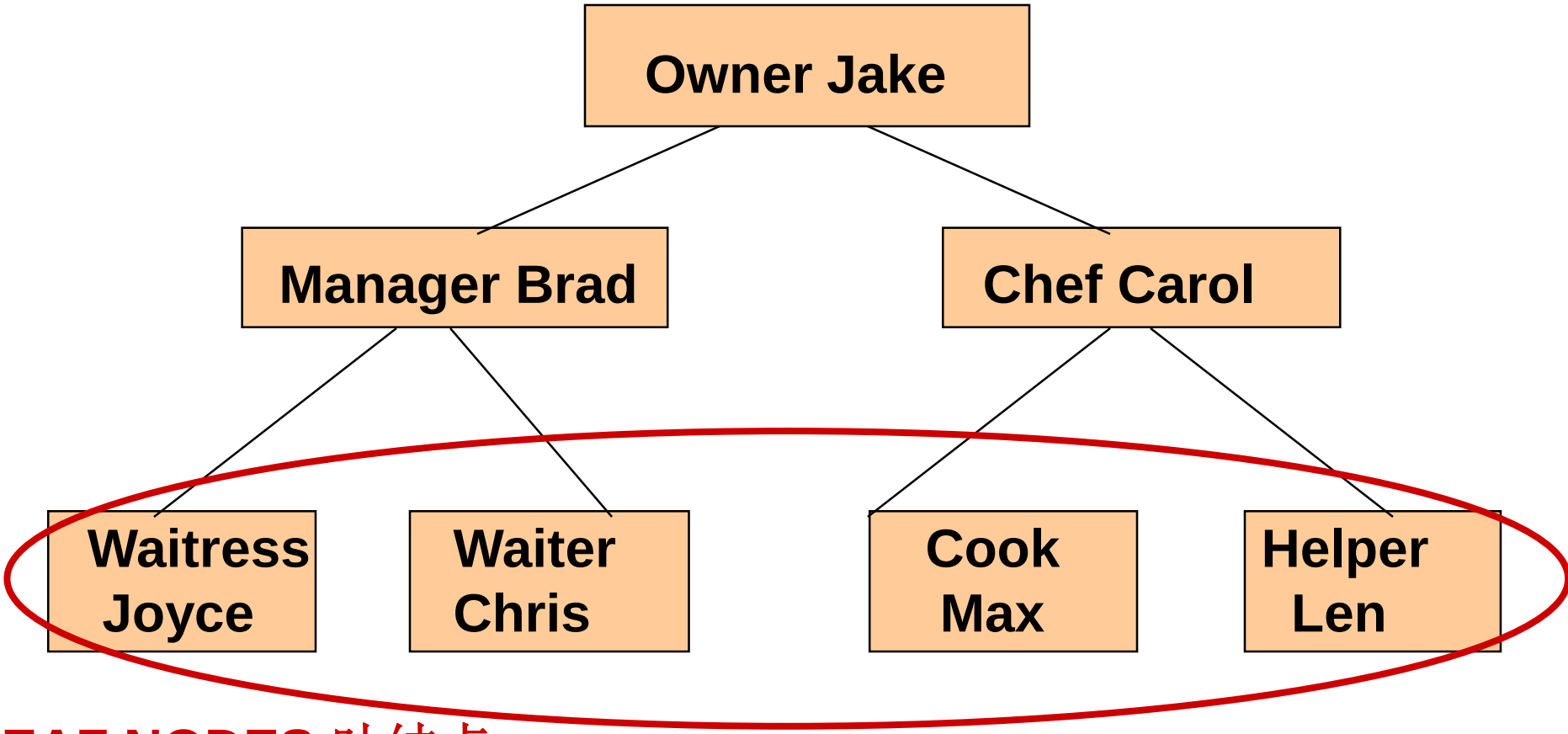
Jake's Pizza Shop Tree



A Tree Has a Root



Leaf nodes have no children

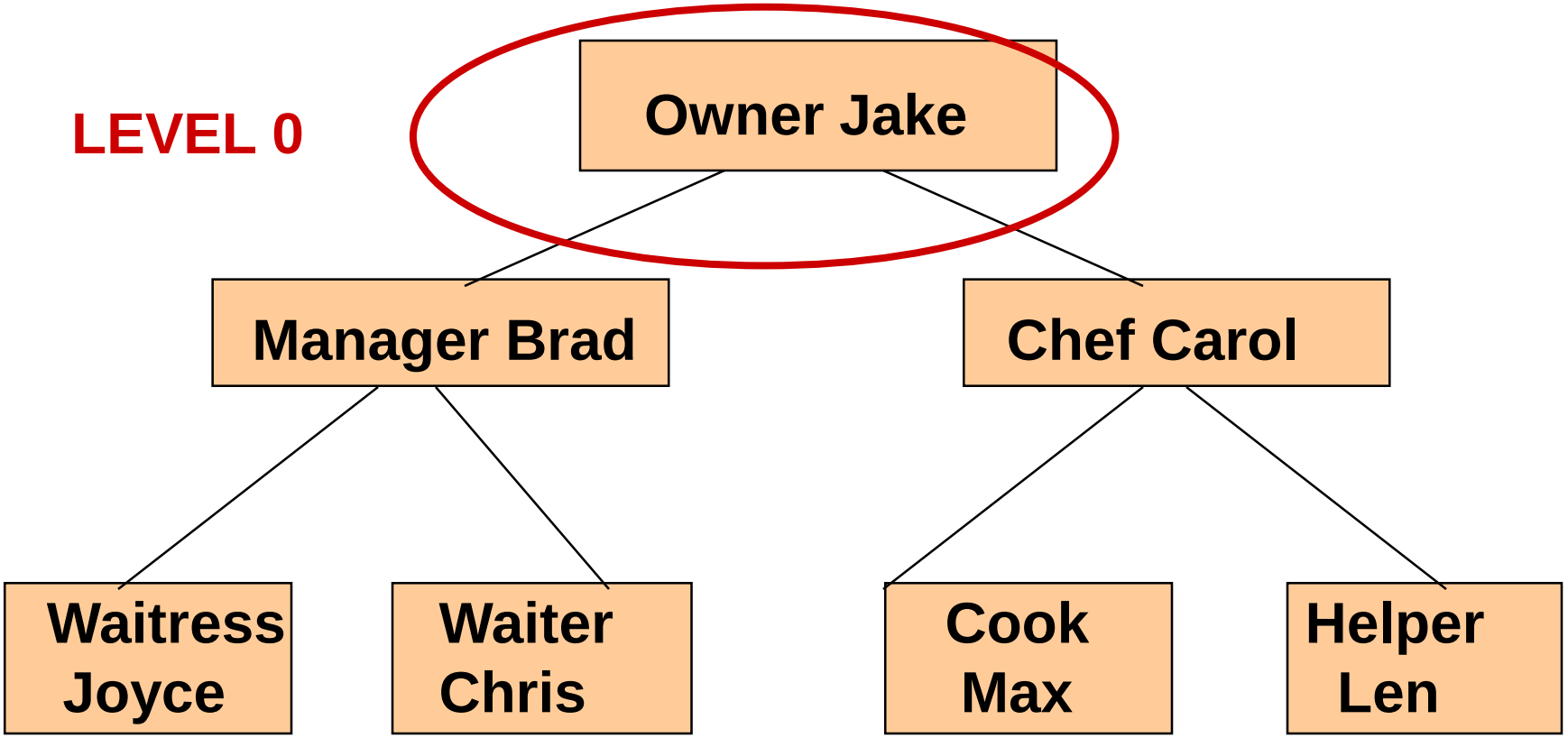


Level of Vertext 结点的层

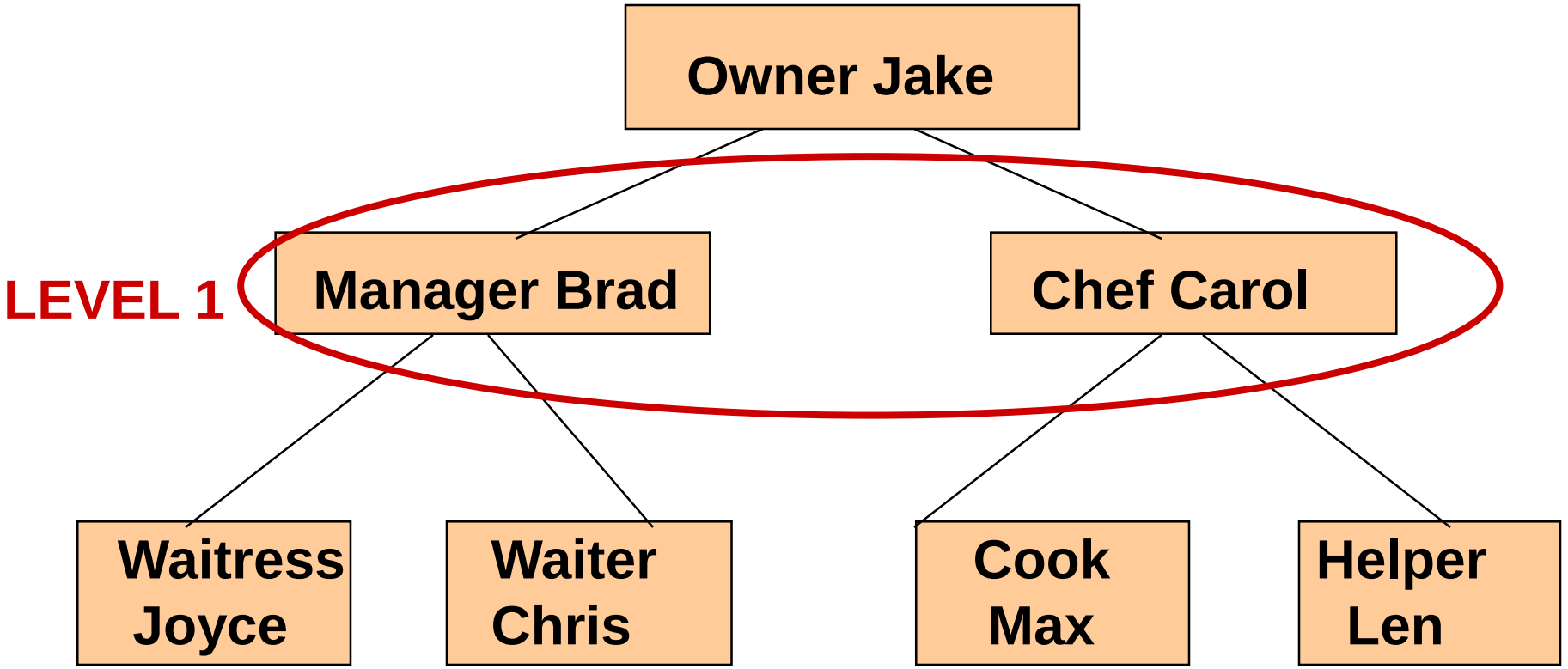
- The level of vertex v in a rooted tree is the length of the unique simple path from the root to v. 从根结点到v的唯一的简单路的长度

- 问题：这个长度是确定的吗？

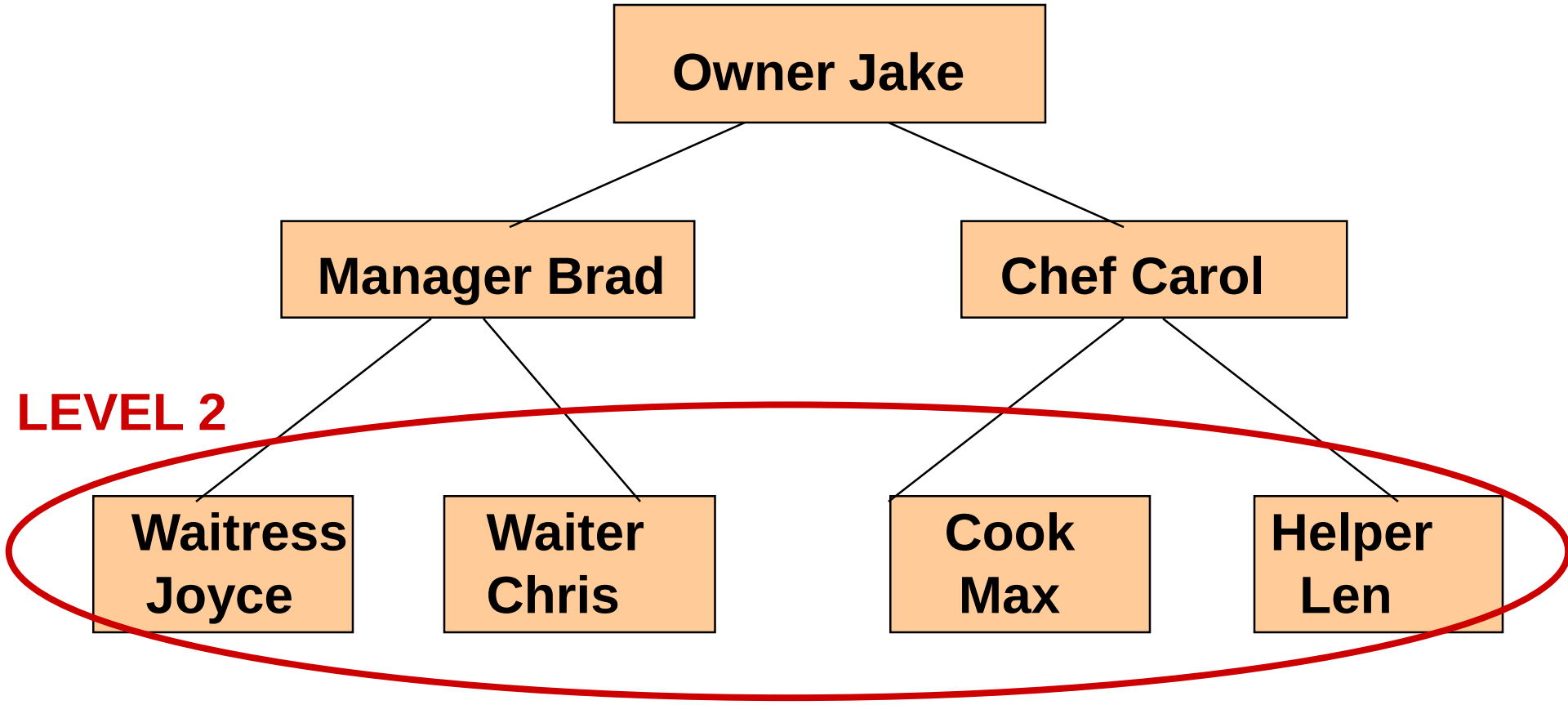
A Rooted Tree Has Levels



Level One


Owner Jake
LEVEL 1
Manager Brad
Chef Carol
Waitress Joyce
Waiter Chris
Cook Max
Helper Len

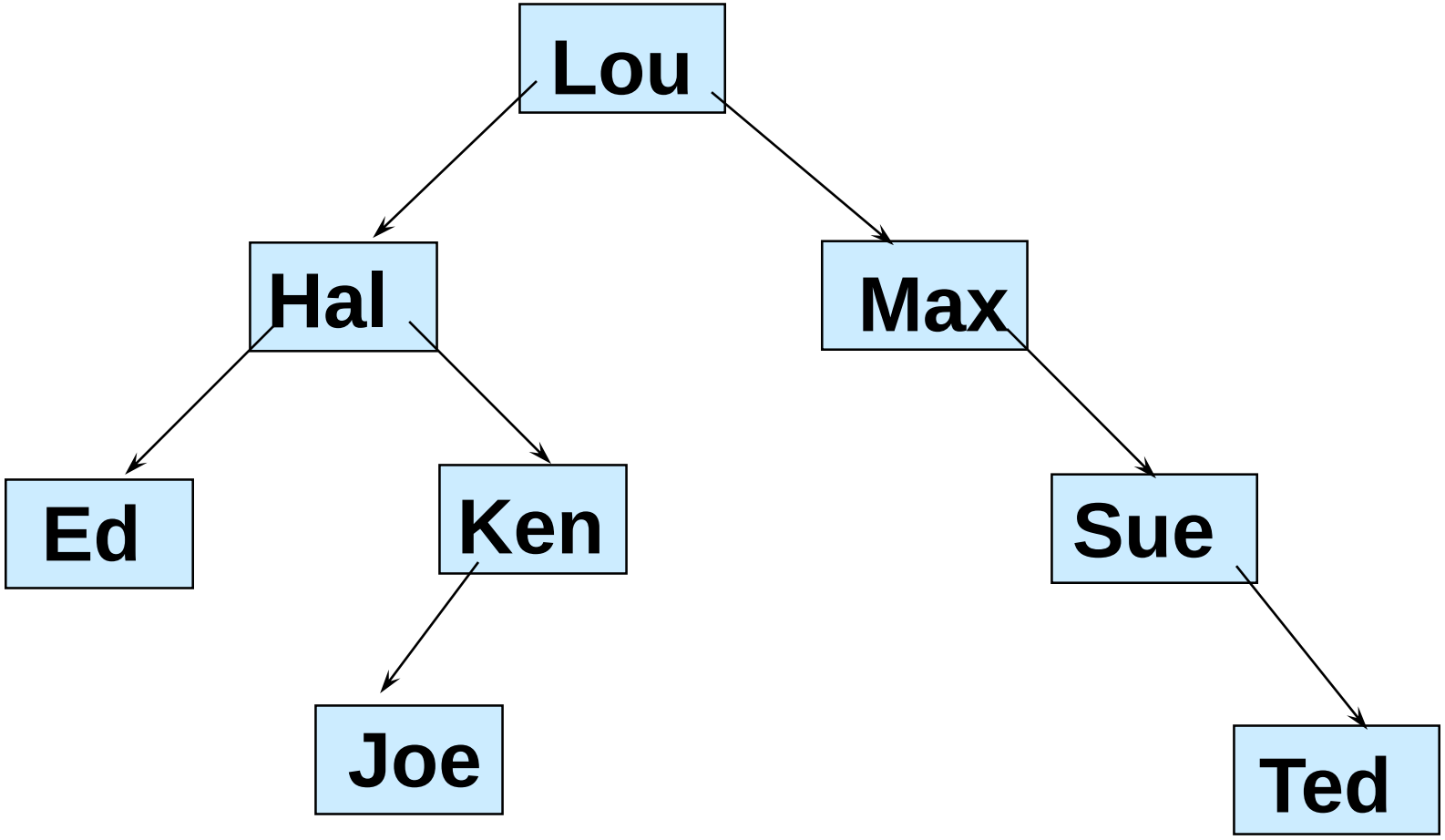
Level Two


Owner Jake
Manager Brad
Chef Carol
LEVEL 2
Waitress Joyce
Waiter Chris
Cook Max
Helper Len

Height 树高

- **The height of a rooted tree is the maximum of the levels of its vertices.**
- **最高层数**

What is the height?

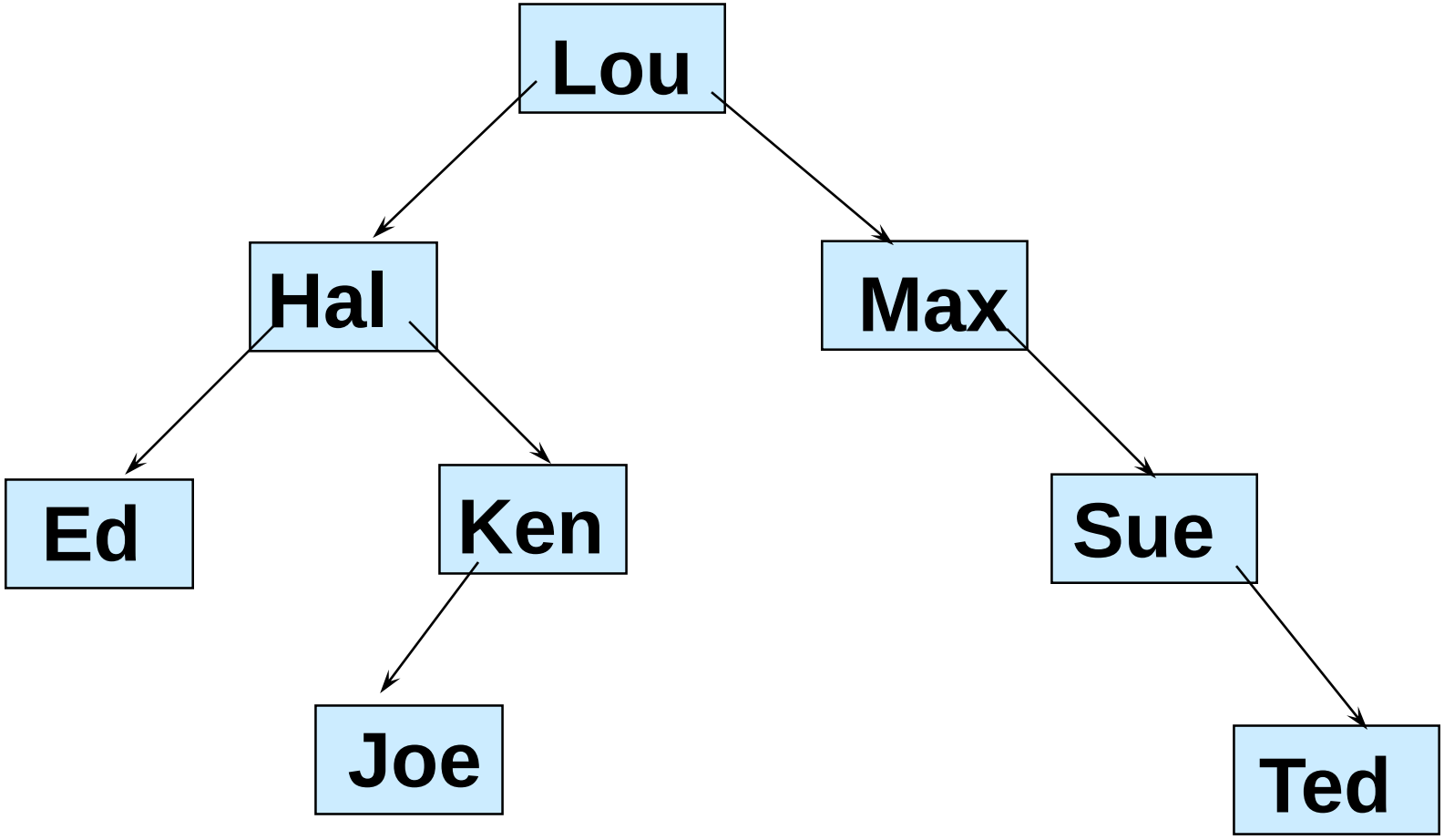


Parent 父结点、子结点

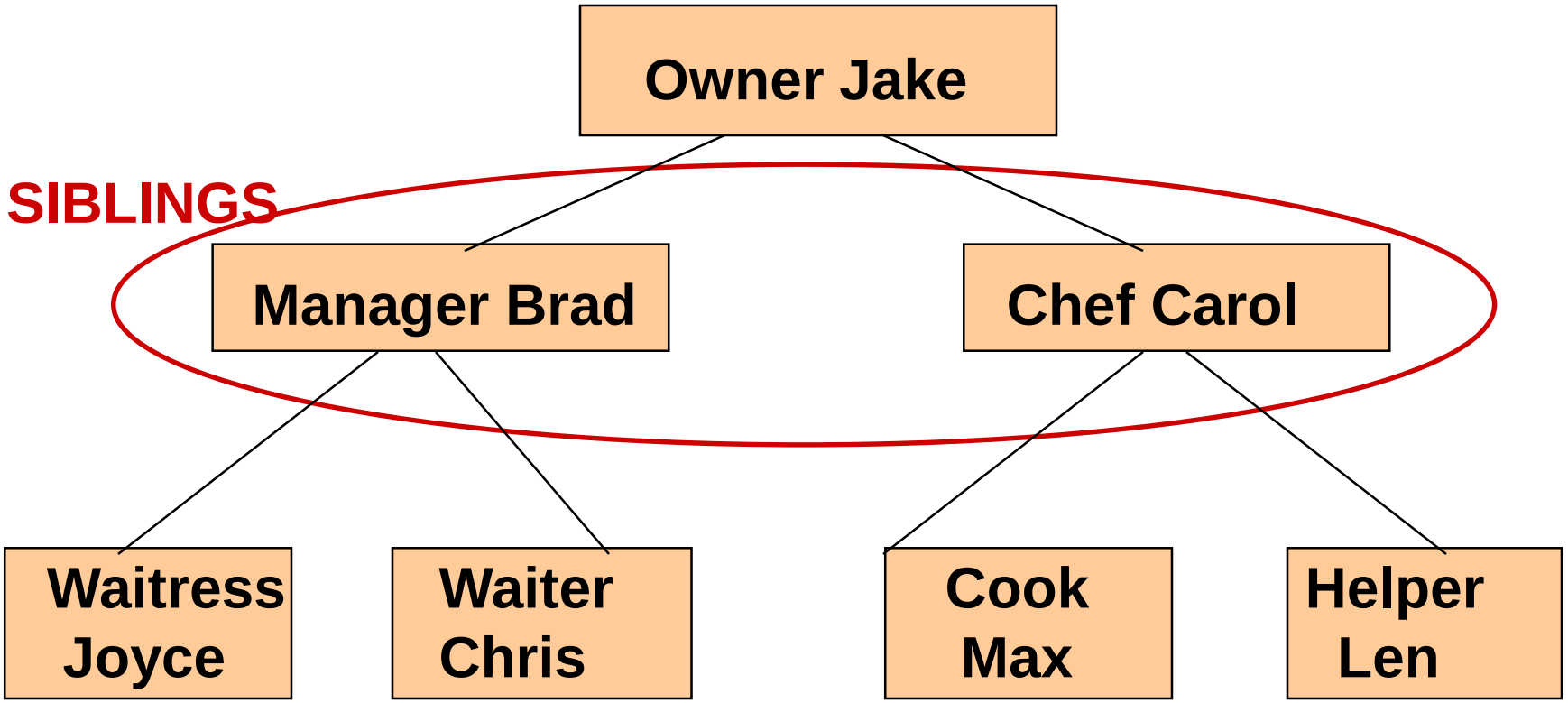
- **The parent of a non-root vertex v （非根结点）is the unique vertex u with a directed edge （有向边）from u to v； V称为u的子结点。**

- 问题：根树的任一个结点是否存在父结点？如果存在是否唯一？
- 任一结点是否存在子结点，子结点是否唯一？ 为什么？
- 根树中任一条边的起点与终点的关系？

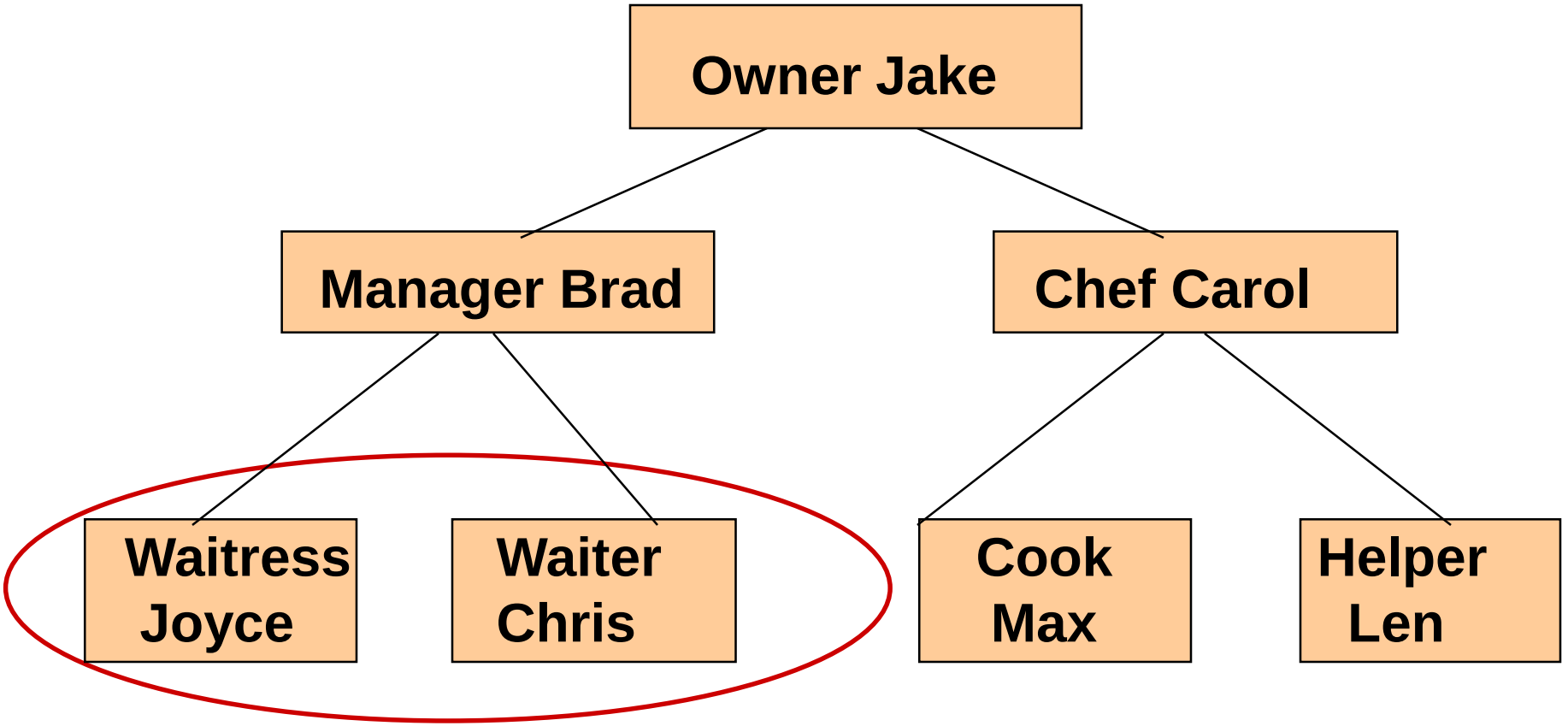
What is the parent of Ed?

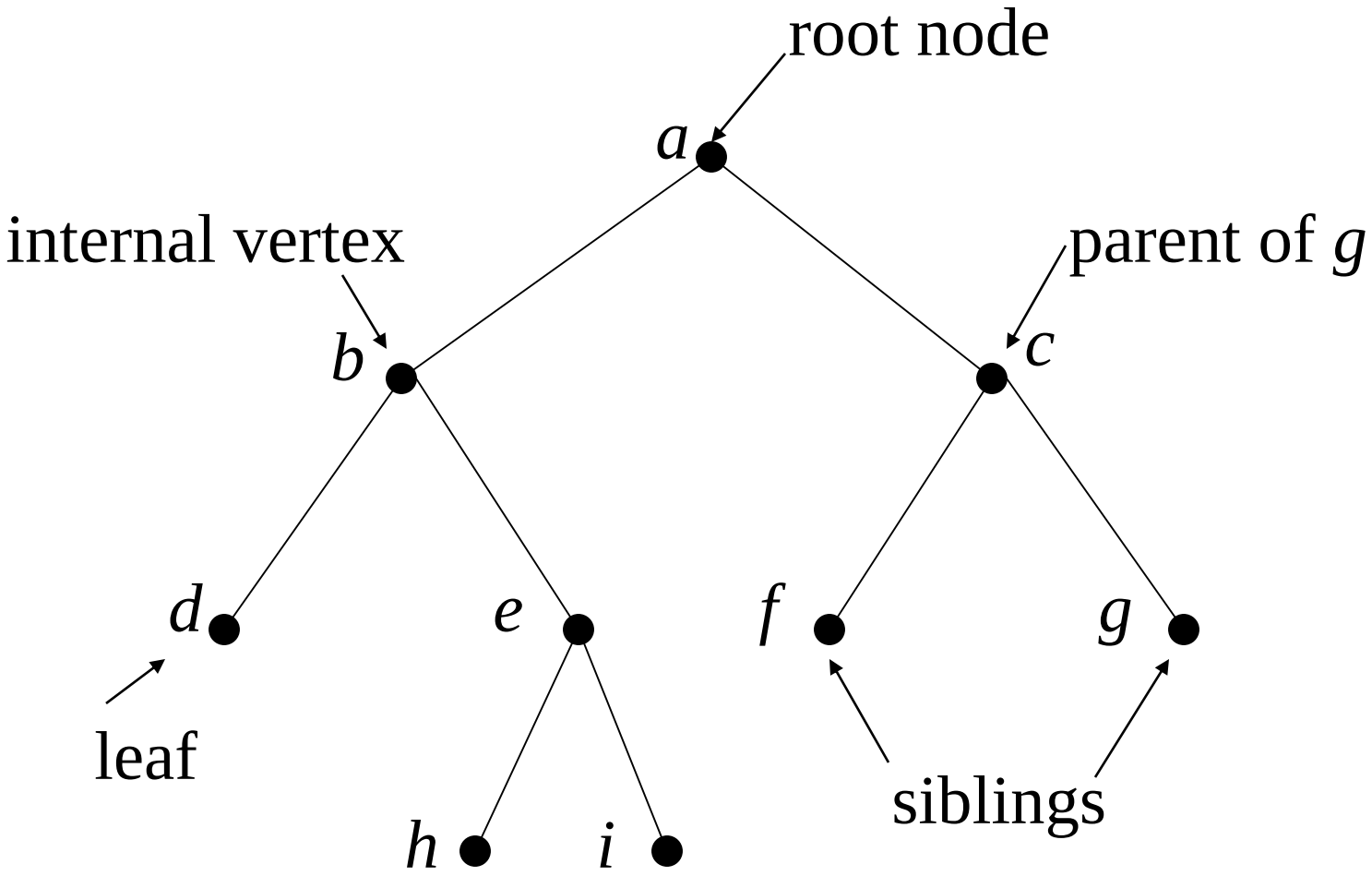


Sibling nodes （姊妹兄弟结点）have same parent



Sibling nodes have same parent



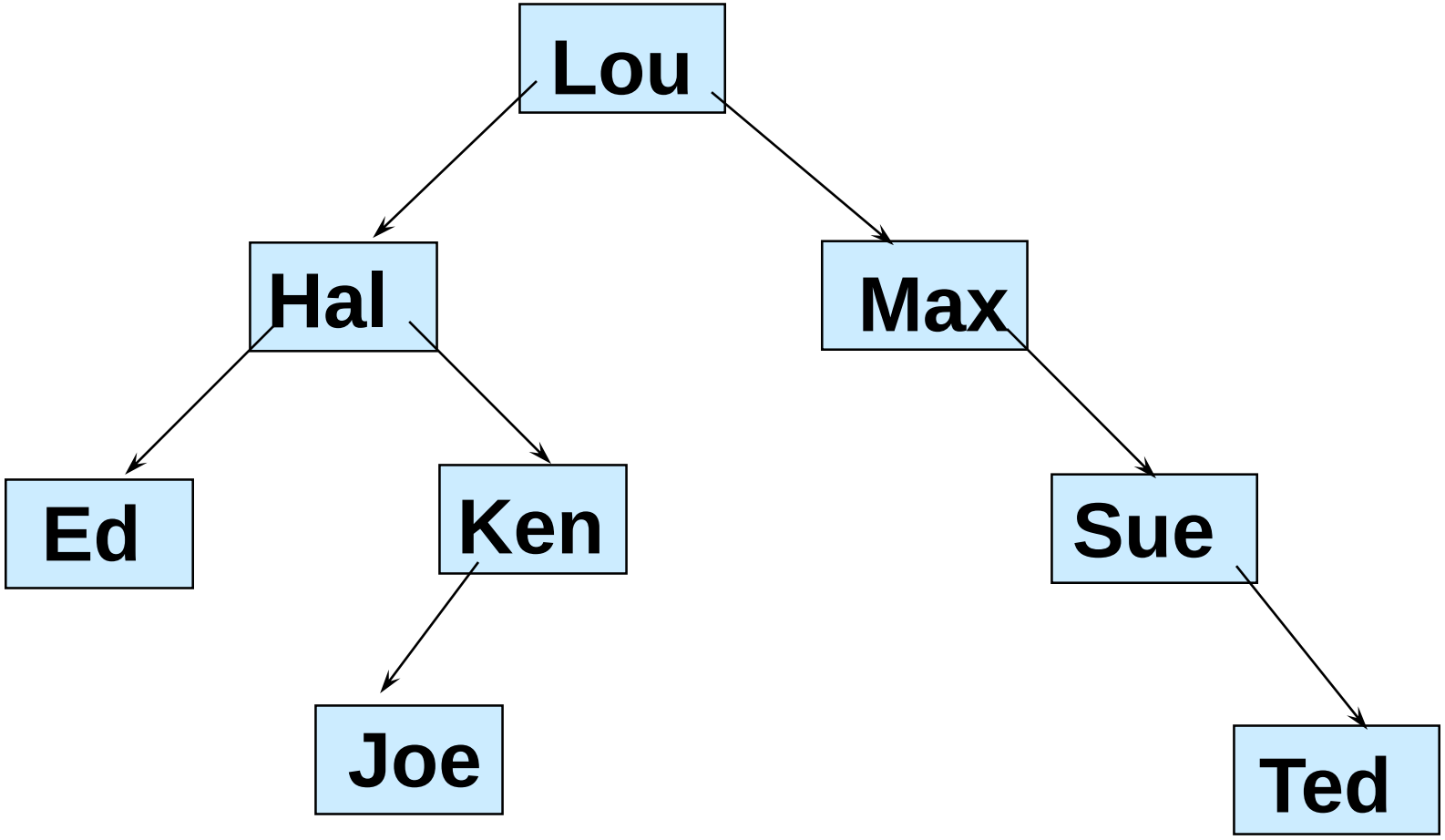

root node
a
internal vertex
parent of g
b
c
d
e
f
g
leaf
siblings
h
i

Leaf 叶结点

- **A vertex is called a leaf if it has no children.**
  没有子结点的结点称为叶结点
- **也是出度为0的点**

- **根树的叶结点的度肯定为1， 是否度为1的结点就一定是叶结点？**

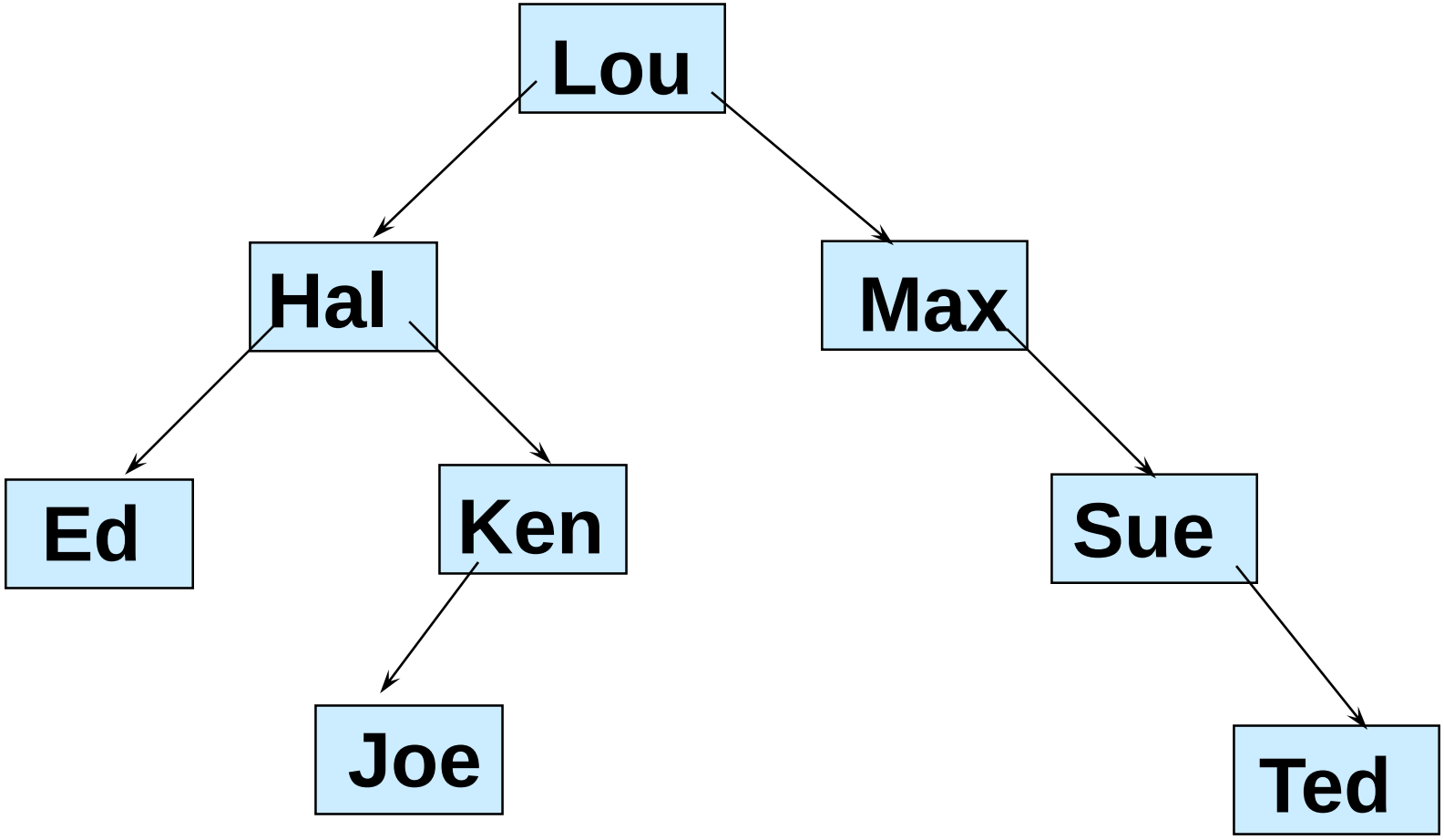
How many leaves?



Ancestors 祖先结点

- **The ancestors of a non-root vertex are all the vertices in the path from root to this vertex.**

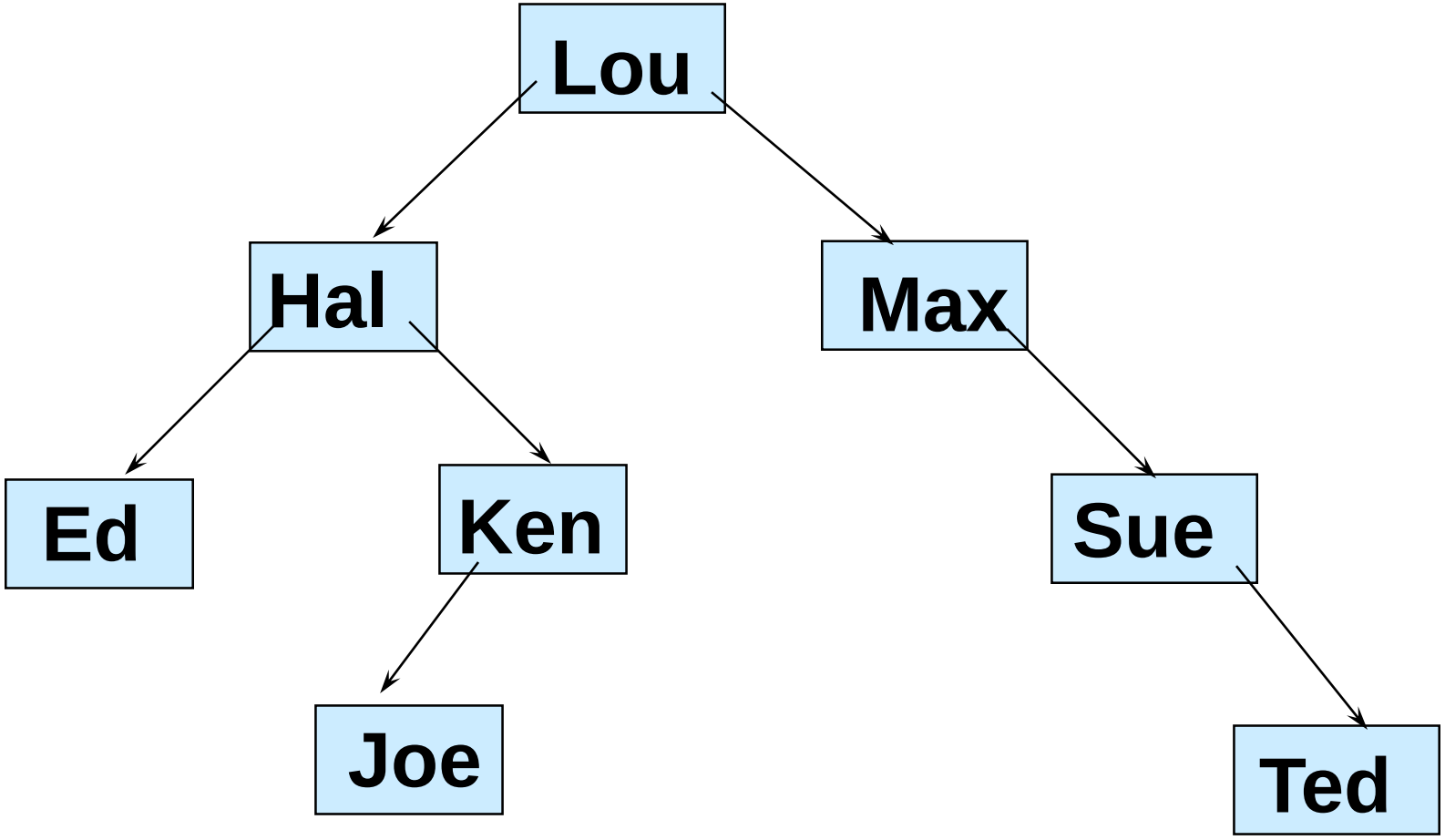
How many ancestors of Ken?



Descendants子孙后代结点

- **The descendants of vertex v are all the vertices that have v as an ancestor.**
- 子孙后代结点

How many descendants of Hal?



Internal Vertex(内结点)

A vertex that has children is called an internal vertex. 有子结点的结点称为内结点

子树：The subtree with vertex v as its root is the subgraph of the tree consisting of vertex v and its descendants and all edges incident to those descendants.

包含v及其所有子结点以及相关的边的子树

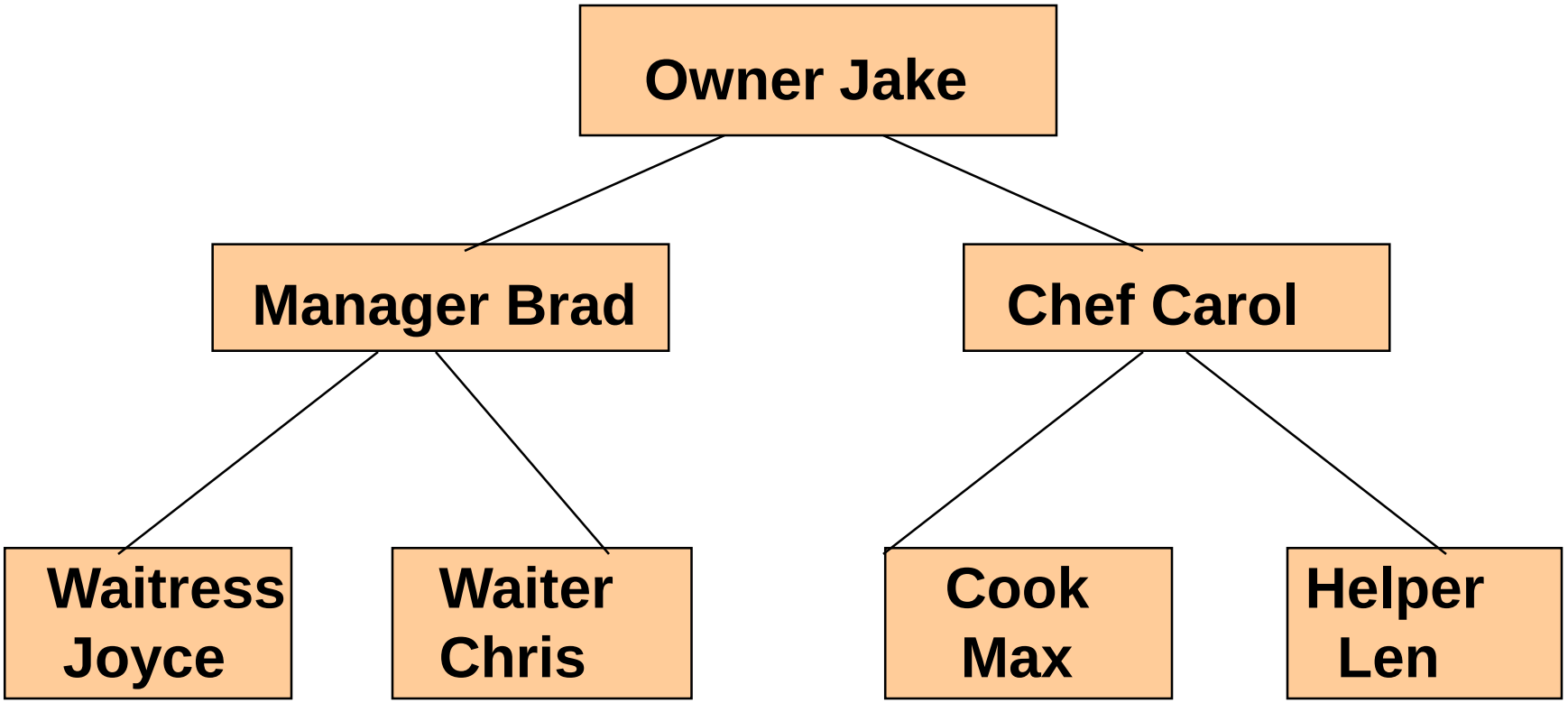
Examples…

有序根树 Ordered Rooted Tree

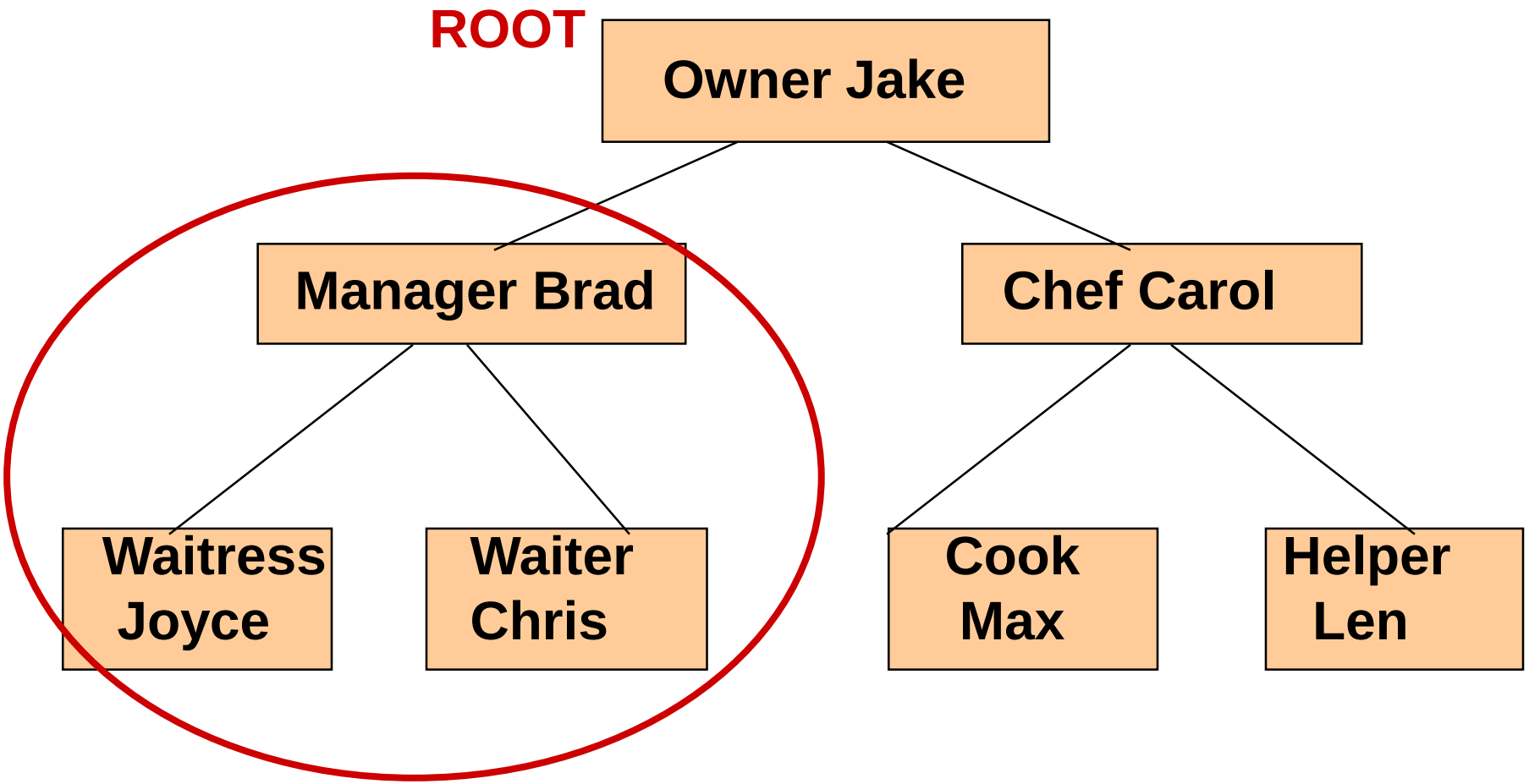
- **Definition**: An *ordered rooted tree* is a rooted tree where the children of each internal vertex are ordered.

- **有序树**：如果将根树的每个内结点的所有子结点都排个序，那这样的根树称为有序根树。

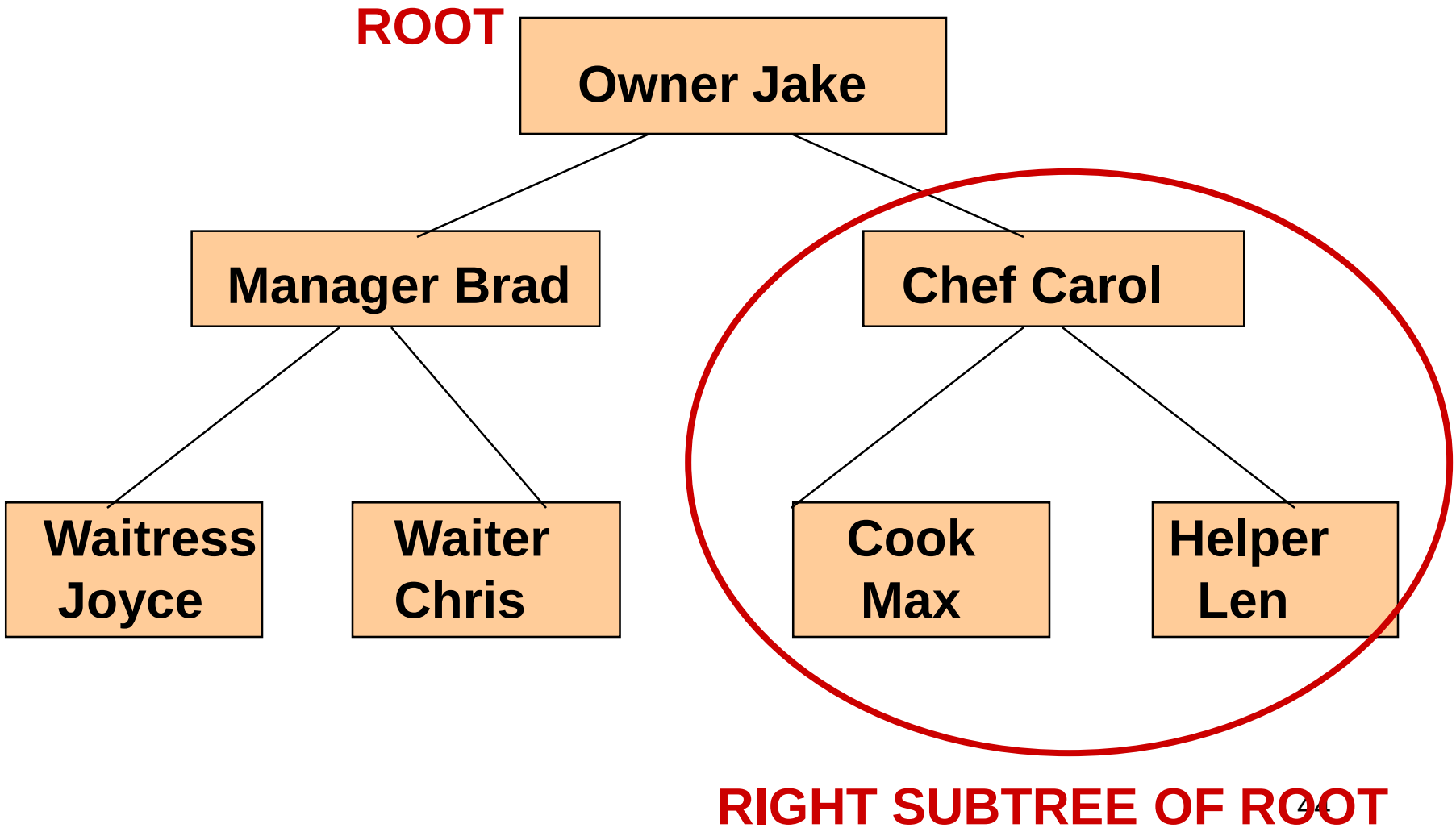
How many internal vertices?



A Subtree of a Ordered Rooted Binary Tree



Another Subtree



m-ary trees m-元树

m-元树(m叉树)：A rooted tree is called a *m-ary tree* if every internal vertex has no more than *m* children.

正则m-元树：The tree is called a *full m-ary tree* if every internal vertex has exactly *m* children. (正则m-元树，教材里翻译成满m-元树）

(complete m-ary tree 完全正则m元树)：如果一颗正则m-元根树的所有叶结点都处于同一层。（教材中完全m元树）

二元树（二叉树）： An *m*-ary tree with *m*=2 is called a *binary tree*.

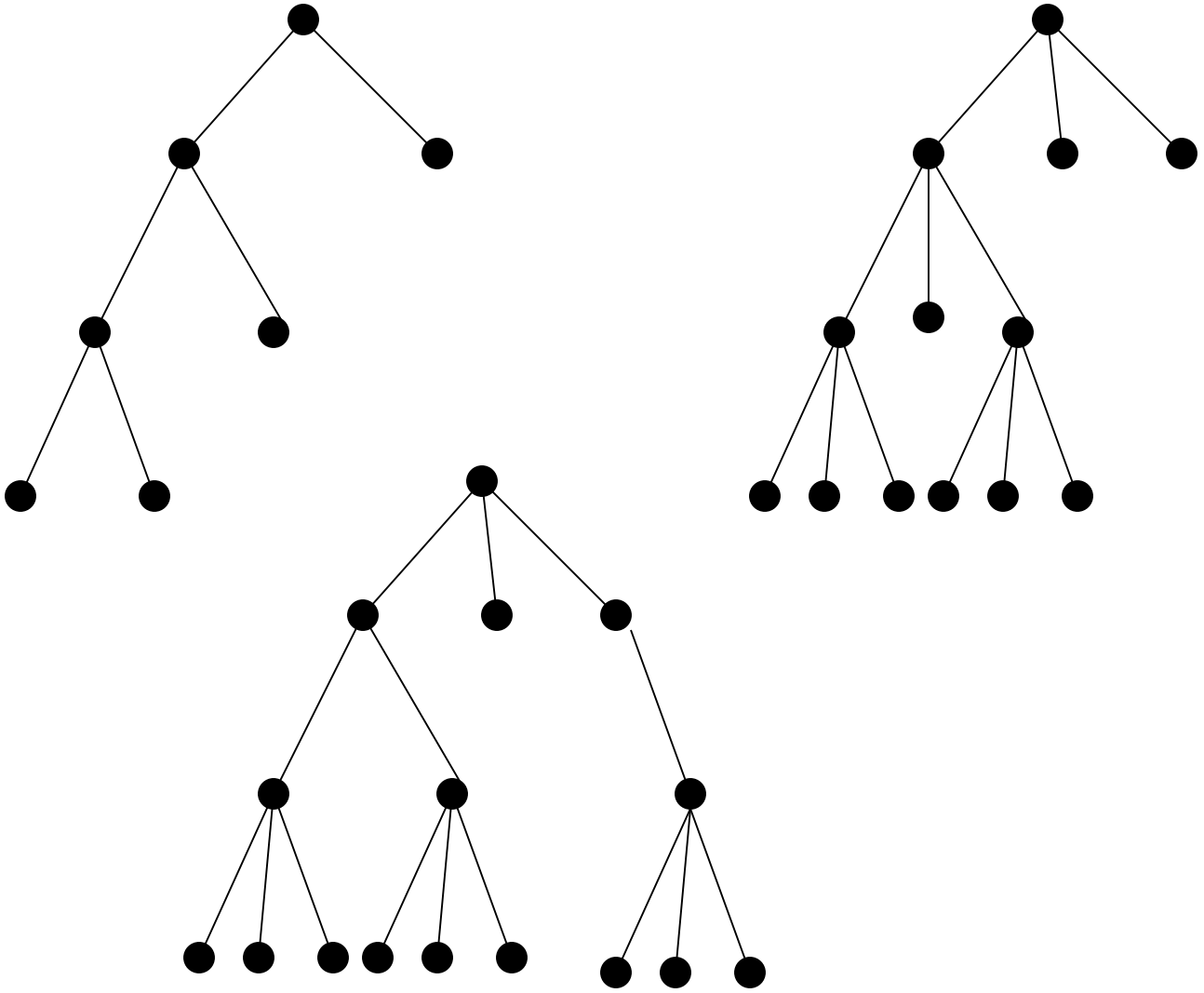
Binary Tree(二元树,二叉树)

Definition 2： A rooted tree is called a binary tree if every internal vertex has no more than 2 children.

The tree is called a full binary tree(正则二元树) if every internal vertex has exactly 2 children.

- Ordered Binary Tree:有序二叉树的每个内结点的子结点分为左子结点和右子结点

m元树的例子



Tree Properties 树的一些数量特征性质

Theorem3：A full m-ary tree with i internal vertices contains n=mi+1 vertices.（正则m元树，满m元树）

Proof: Every vertex (but root) is a sub vertex of an internal vertex. There are mi sub-vertices (total), plus the root which is not a sub vertex of any vertex.

Theorem 4. There are at most 2^H leaves in a binary tree of height H. 高度为H的m元树顶多有m^H个叶结点。

Definition: A rooted *m*-ary tree of height *h* is **balanced (**平衡的） if all leaves are at levels *h* or *h* − 1.

推论：If a m-ary tree with L leaves, then its height h> = $\lceil \log_m L \rceil$. If it is full and balanced(正则且平衡的), then its height is $H = \lceil \log_m L \rceil$.

Summary—the most important properties
树的重要性质总结

- 1. Connected 连通
- 2. no simple circuit 无简单回路
- 3. n=m+1 (or m=n-1), where n is the number of vertices, m number of edges
- 4. There is an unique simple path between any two distinct vertices. 任意两个点之间存在唯一真路（简单路）
- 5. Start from any vertex, a rooted (directed) tree can be constructed 从任何一点出发都可以形成一颗根树
- 6. 根树是有向树，从根到任一个非根结点有唯一的一条有向路。
- 根树的内结点到其任一子孙结点都有唯一的有向路

思考问题

- 如果一个n个结点的连通图，恰好有n-1条边，是否这个图一定是树？
- 定理：一个(n,m)连通图是树当且仅当n=m+1
- 如果是，能否以这两个条件作为树的定义？也就是说与树的定义的两个条件等价？
- 如果是n个结点的树林，那应该会有多少条边？

Some questions about tree

- Question 1: what is the minimum number of edges of a connected undirected simple graph with n vertices?
  n个结点的一个连通简单无向图的至少有多少条边？

- Question 2: if A is the adjacency matrix of an undirected simple graph G, can you determine whether G is a tree or not? How? 如何利用邻接矩阵判断一个简单无向图是否是一颗树？

- Question 3: which vertices in a tree are cut vertices? Which edges are cut edges? 树中哪些边是割边，哪些是割点？

- Question 4: is there any circuit in a tree? 树中有回路吗？

Exercises

- 7.1节 T6(1)
- 7.1节 T10，T11 , T16

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、社会管理道信息科学等等。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

搜素树应用

- Searching for items in a list is one of the most important tasks that arises in computer science.
- Our primary goal is to implement a searching algorithm that finds items efficiently when the items are totally ordered. This can be accomplished through the use of a binary search tree （二元搜素树）

（尤其是二叉搜素树BST应用更广）

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.

Binary Search Trees二叉搜索树

- **Binary search trees (BST)**: sometimes called **ordered** or **sorted binary trees**, are a particular type of container: data structures that store "items" (such as numbers, names etc.) in memory.
- **BST**: each child of a vertex is designated as a right or left child; each internal vertex has only one left child or right child; Vertices are assigned keys so that the key of a vertex is both larger than the keys of all vertices in its left subtree and smaller than the keys of all vertices in its right subtree.

- **Primary goal**: to implement a searching algorithm that finds items efficiently when items are totally ordered
- They allow fast lookup, addition and removal of items,…

树的应用

- 搜素树、决策树、博弈树等等都会在数据结构里继续学习
- 树的遍历算法：数据结构中的重要内容，不在离散数学里重复讲授。

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：每个字符用相同长度的2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。

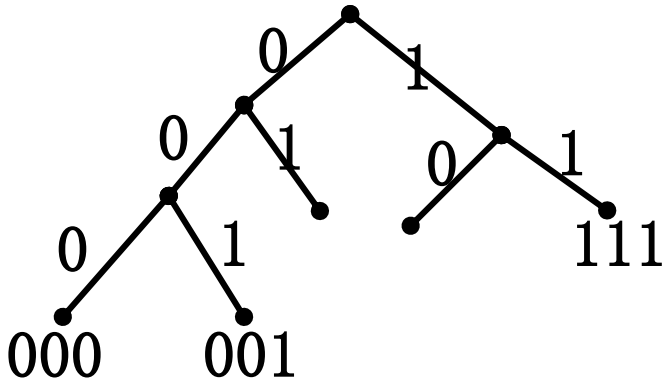
例如：010， 01011 这两个编码就不符合上面要求

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(或不能是前缀)

设$\{S_1, S_2, \ldots S_n\}$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)：



Prefix Code(前缀码)--Huffman code

- 思考问题：那么如何利用前缀码的理论构造编码集？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，使得该树的所有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。给一个01前缀码, 如何构造和用二元树来分析解释一段01串？

Huffman Tree 哈夫曼树

- 那么如何设计一套合理的前缀码，使得平均码长最短？

- **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，怎么存放有利于使用？

最优树问题(哈夫曼树、哈夫曼编码)

- **问题：**给定一组权w1,w2,w3,…,wt, 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，即：

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

- 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？

哈夫曼编码 （非常重要的算法）

- **利用上面的原理，可以合理设计前缀码集合。**

1：首先统计使用频率

2：分析有多少需要编码的字符

3：排序，设计二叉树以及对应的编码集合

，前缀码集合的基数就对应着相应的二叉树的叶结点的个数

4：利用哈弗曼编码设计前缀码集合

实际上，就是去构造一颗**哈夫曼树（最优树）**

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)以树叶结点V_{w1}为子结点的分枝结点到根的距离最远. (也即权最小的离根最远)
- (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。

问题:那么权最大的就应该离根最近?

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T1,那么T1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

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。
- 平均码长最短就是用下面的目标函数来评估：

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

练习

- 7.2 节 T10 (b), (d) T11
- 课外自己做做T14, 但不用做到练习本上