- 二叉树
  - 树
    - 基本概念
    - 实现
      - 基本操作
    - 二叉树
      - 多叉树的描述
  - 树的遍历
    - 先序遍历
      - 迭代版本
    - 中序遍历
      - 迭代版本
      - 产生的相关概念
    - 后序遍历
      - 迭代版本
    - 层次遍历——对于完全二叉树
    - 遍历序列重构二叉树
  - 应用
    - 表达式树
    - Huffman树
      - 思想
      - 构造
      - 正确性
      - 复杂度

二叉树

树

层次结构的表示

兼具 Vector 和 List 特点（查找、插入、删除）

半线性：在确定某种次序之后，具有线性特征。

基本概念

树：极小连通图、极大无环图

有根树

子树

孩子 兄弟 父亲

路径 环路

深度：`depth(v) = |path(v)|`

高度：`height(v) = max(depth(x)), x is a leaf in subtree v` 空树的高度取作 -1 。

实现

基本操作

子树连接、子树删除、子树分离

二叉树

有根有序，每个节点的儿子不超过 2 个。

`lc(), rc()`

可以引入外部节点让每个节点都有且仅有两个“儿子”。实现中通常是用 null 指针， 0 节点。

多叉树的描述

长子-兄弟描述法。长子是左孩子，兄弟是右孩子

树的遍历

按照某种次序访问树中的各个节点，每个节点恰好被访问一次。

其意义在于把树形结构转换为线性结构。

以下的序都是指把“根节点”放在什么位置而言的。

先序遍历

先遍历根节点，VLR

迭代版本

藤缠树。

自上而下访问藤上的节点。自下而上遍历各右子树。

中序遍历

在中间遍历根节点，LVR

迭代版本

访问藤上节点，再遍历其右子树。

产生的相关概念

前驱和后继

后序遍历

在后面遍历根节点，LRV

迭代版本

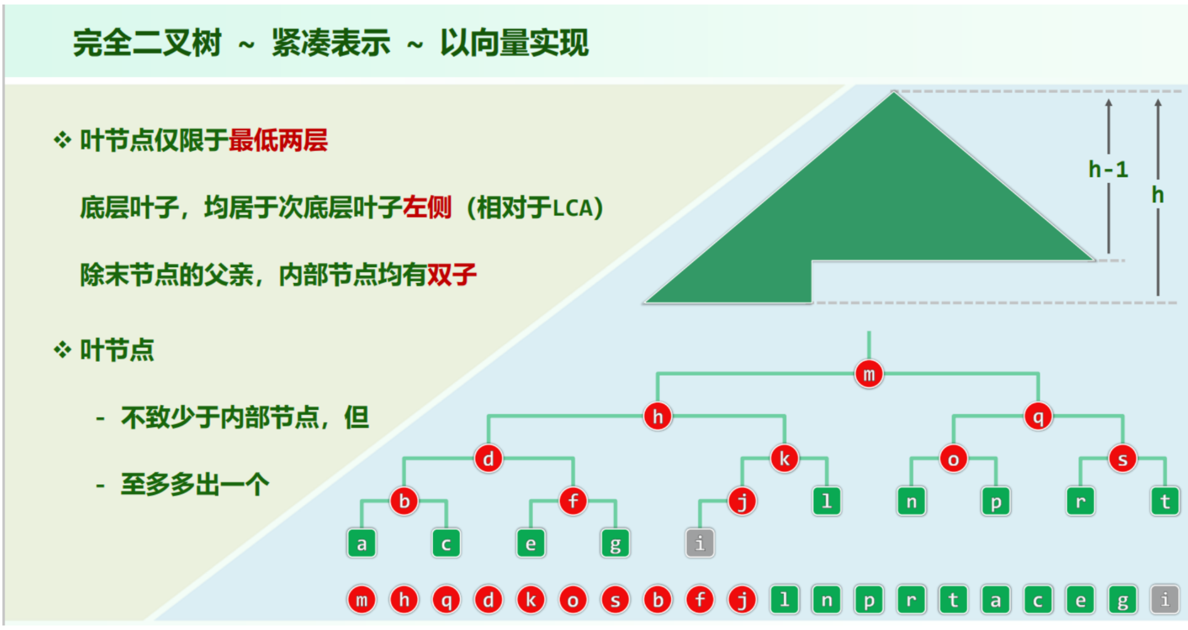
（以上的迭代版本似乎还得再看一看）

层次遍历——对于完全二叉树

用一个队列来记录将要访问的节点。

事实上就是广度优先搜索。

队列得大小先增后减，单峰且对称



叶节点只在最后两层且最后一层叶节点左对齐。度为1的节点只有左孩子。度为1的节点至多1个（可能没有）。

辅助队列的最大规模为$\lceil n/2 \rceil$（习题5-18）

遍历序列重构二叉树

- 先序 | 后序 + 中序 可以重建，在中序中找到 V，先序/后序提供了根，中序提供了LR
- 先序 + 后序，可以重建，但问题在于只有一个子树时无法确定在哪边（所以真二叉树就可以）
- 先、中、后增强序列，用一个符号如 `^` 表示补全树之后所有的 null 节点【子序列中的 NULL 比非 NULL 多一个，可以直接线性扫描？】

应用

表达式树

对表达式树进行中序遍历就可以得到原来的表达式。

Huffman树

思想

重新编码，出现的多的字符给一个短些的编码；出现的少的字符给一个长些编码；

要保证编码没有二义性，即任意两个编码不互为前后缀。

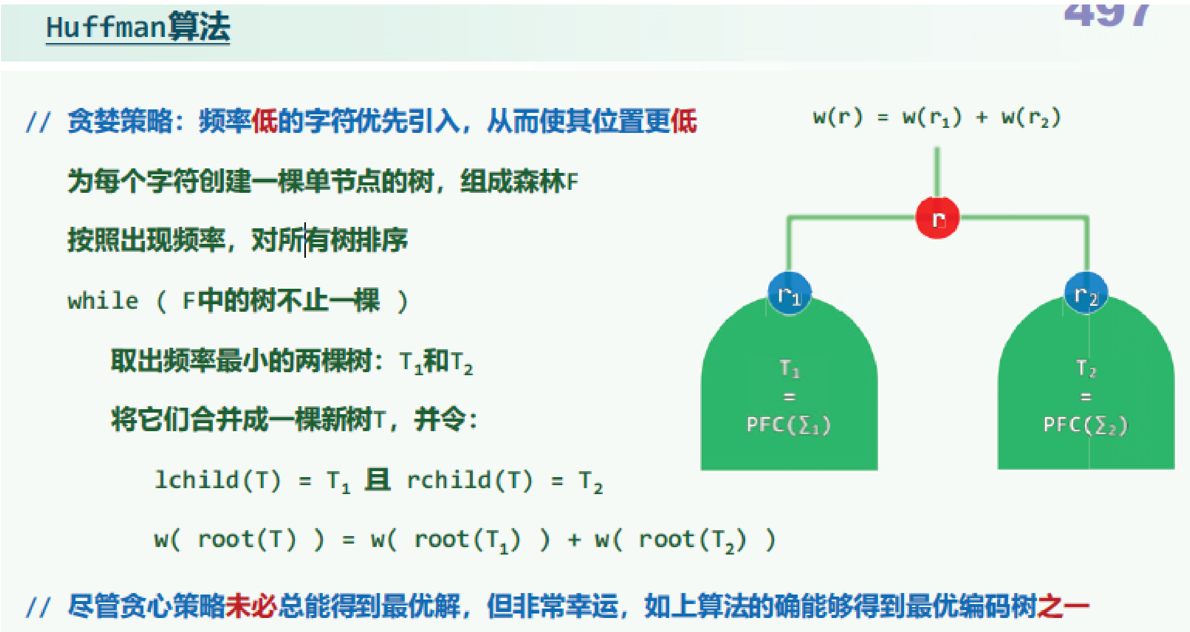
在一个 0/1 Trie 树上做编码，如果以每个节点到根的路径作为“编码”，那么所有节点都应该在叶子上。

平均编码长度 $\text{ald}(T) = \sum_{x\in\Sigma} \text{depth}(v(x))/|\Sigma|$

最优编码树：不考虑出现的概率。性质：所有叶子节点只能出现在倒数两层内。

最优带权编码树：$\mathrm{wald}(T)=\sum_x \mathrm{rps}(x)\times \mathrm{w}(x)$

构造



正确性

1. 出现频率最低的字符 x 和 y，必然在某棵最优编码树中处于最底层，且互为兄弟。否则，往下交换，`wald` 必然不会增加。
2. 数学归纳法：
    1. 设在 $|\Sigma|<n$ 时均正确。若 $|\Sigma|=n$，取 Σ 中频率最低的 x, y，不妨设两者互为兄弟。
    2. 令 $\Sigma'=(\Sigma\setminus\{x,y\})\cup\{z\}, w(z)=w(x)+w(y)$。
    3. 因为 x, y 必然是兄弟，这两者最优性必然是等价的。
    4. 所以得到 Σ' 的最优编码树之后，可以把 z 替换成 x 和 y，也必然是一个最优的树。

复杂度

$O(n^2)$

> 如果引入高级数据结构如堆等等，可以优化到 $O(n\log n)$

“小米加步枪”方法——可以做到$\mathrm{O}(nlogn)$

❖ 方案4： - 所有字符按频率排序，构成一个栈 //$\mathcal{O}$(nlogn)

- 维护另一个有序队列... //$\mathcal{O}$(n)