L11：Huffman树

吉林大学计算机学院
谷方明
fmgu2002@sina.com

学习目标

- 理解压缩和编码；
- 掌握扩充二叉树的相关定义及常用结论；
- 掌握Huffman算法、正确性证明和实现
- 掌握Huffman编码及压缩技术

背景:压缩和编码

- 数据压缩是计算机科学中的重要技术。
- 数据压缩过程称为编码，即将文件中的每个字符均转换为一个唯一的二进制位串。数据解压过程称为解码，即将二进制位串转换为对应的字符。
- 压缩的关键在于编码的方法。

等长编码

- 假设有一个文件仅由7个字符组成：

  a、***e***、***i***、***s***、***t***、***sp***（空格）和***nl***（换行），且文件中有**10**个***a***，**15**个***e***，**12**个***i***，**3**个***s***，**4**个***t***，**13**个***sp***，**1**个***nl***。

- 区分7个字符，至少要 $\lceil \log_2 7 \rceil = 3$ 位二进制；于是文件的总位数至少应该是：

  $$10\times3 + 15\times3 + 12\times3 + 3\times3 + 4\times3 + 13\times3 + 1\times3 = 174.$$

不等长编码

- 在实际的文件中，字符使用的频率是非平均的，有些字符出现的次数多，而有些字符出现的次数却非常少。如果所有字符都用等长的二进制码表示，那么将造成空间浪费。

- 文件压缩的通常策略：采用不等长的二进制码，令文件中频率高的字符的编码尽可能短。

前缀码

- 采用不等长编码可能会产生多义性。
  - 例：如果用**01**表示***a***，**10**表示***b***，**1001**表示***c***，那么对于编码**1001**，我们无法确定它表示字符***c***，还是表示字符串***ba***。
  - 原因：***b***的编码是***c***的编码的前缀。
- 前缀码：为避免多义性，要求字符集中任何字符的编码都不是其它字符的编码的前缀。显然，等长编码是前缀码。

问题的数学描述

- 设计前缀码使文件的总编码长度最短
- 设组成文件的字符集$A=\{a_1,a_2,\ldots,a_n\}$，其中，a_i的编码长度为l_i；a_i出现的次数为c_i 。要使文件的总编码最短，就必须要确定l_i，使

$$\sum_{i=1}^{n} c_i l_i$$

取最小值。

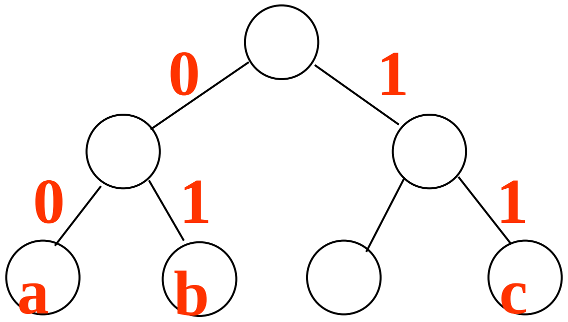
灵光一现的创造

Robert M. Fano **David A. Huffman**

- A Method for the Construction of Minimum-Redundancy Codes

Huffman的灵感

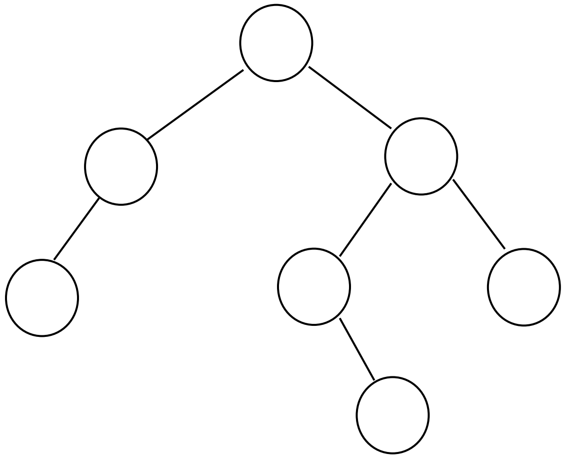
- 前缀码对应一条路径（如 0左1右）

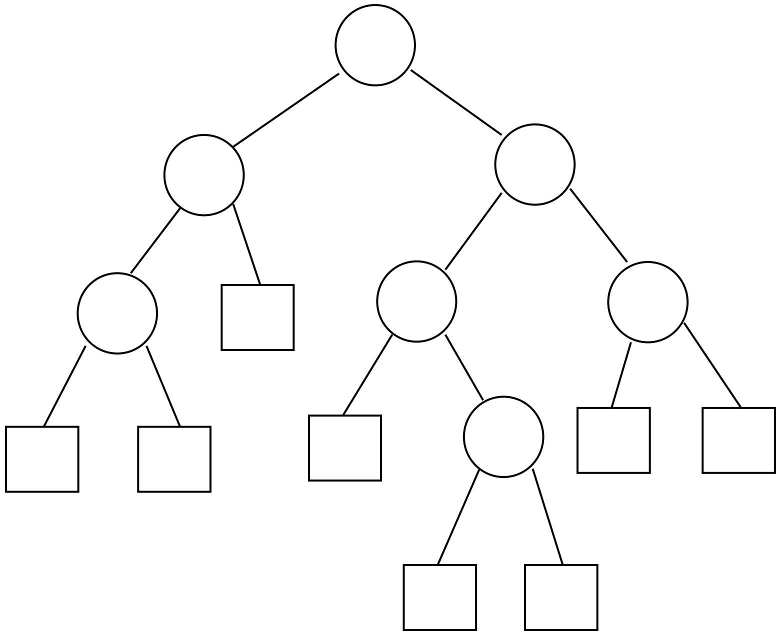


- 前缀码的集合对应一棵二叉树；编码是每个叶结点对应的路径，一个叶子不可能是其它叶子的祖先，因此一个叶子的编码不可能是其它叶子的编码的前缀；

扩充二叉树

- 定义5.5：为了使问题的处理更为方便，每当原二叉树中出现空子树时，就增加特殊的结点——空树叶，由此生成的二叉树称为扩充二叉树。



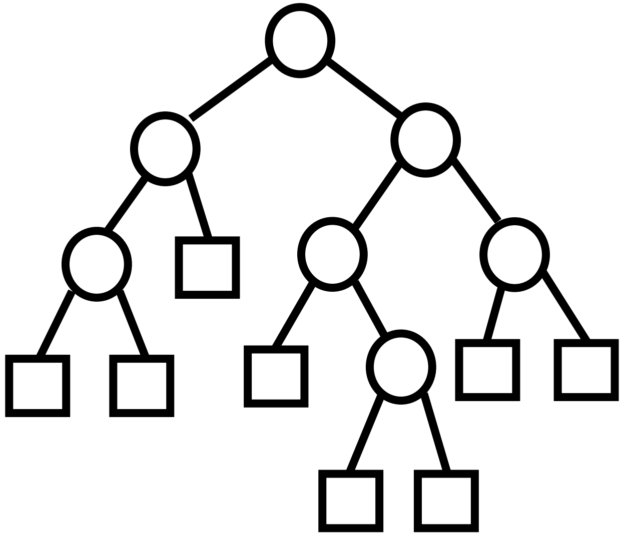


内结点和外结点

- 规定空二叉树的扩充二叉树只有一个方形结点。圆形结点称为内结点，方形结点称为外结点。
- 扩充二叉树每一个内结点都有两个儿子，每一个外结点没有儿子。

外通路长度和内通路长度

- 定义5.6： 扩充二叉树的外通路长度定义为从根到每个外结点的路径长度之和，内通路长度定义为从根到每个内结点的路径长度之和。



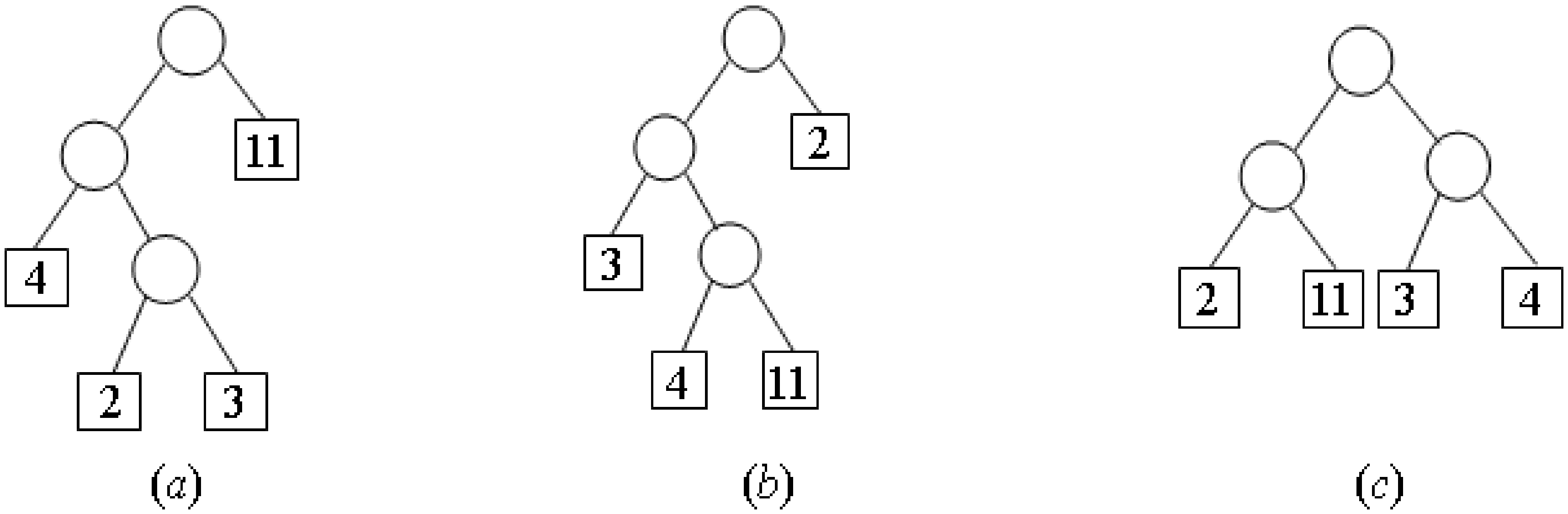
外通路长度为 3 +3 +2 +3 + 4 + 4 +3 +3=25

内通路长度为 2 +1 + 0 +2 +3 +1+2=11.

加权外通路长度

- 定义5.7： 给扩充二叉树中 ***n*** 个外结点赋上一个实数，称为该结点的权。树的加权外通路长度定义为***WPL***：

$$WPL = \sum_{i=1}^{n} w_i L_i$$



- 加权外通路长度分别是

  4*2+2*3+3*3+11*1=34

  3*2+4*3+11*3+2*1=53

  2*2+11*2+3*2+4*2=40

最优二叉树

- 定义5.8 ：在外结点权值分别为$w_0,w_1,...,w_{n-1}$的扩充二叉树中，加权外通路长度最小的扩充二叉树称为最优二叉树 。
- 文件编码问题就变成构造最优二叉树问题，每个外结点代表一个字符，其权值代表该字符的频率，从根到外结点的路径长度就是该字符的编码长度 。

哈夫曼算法

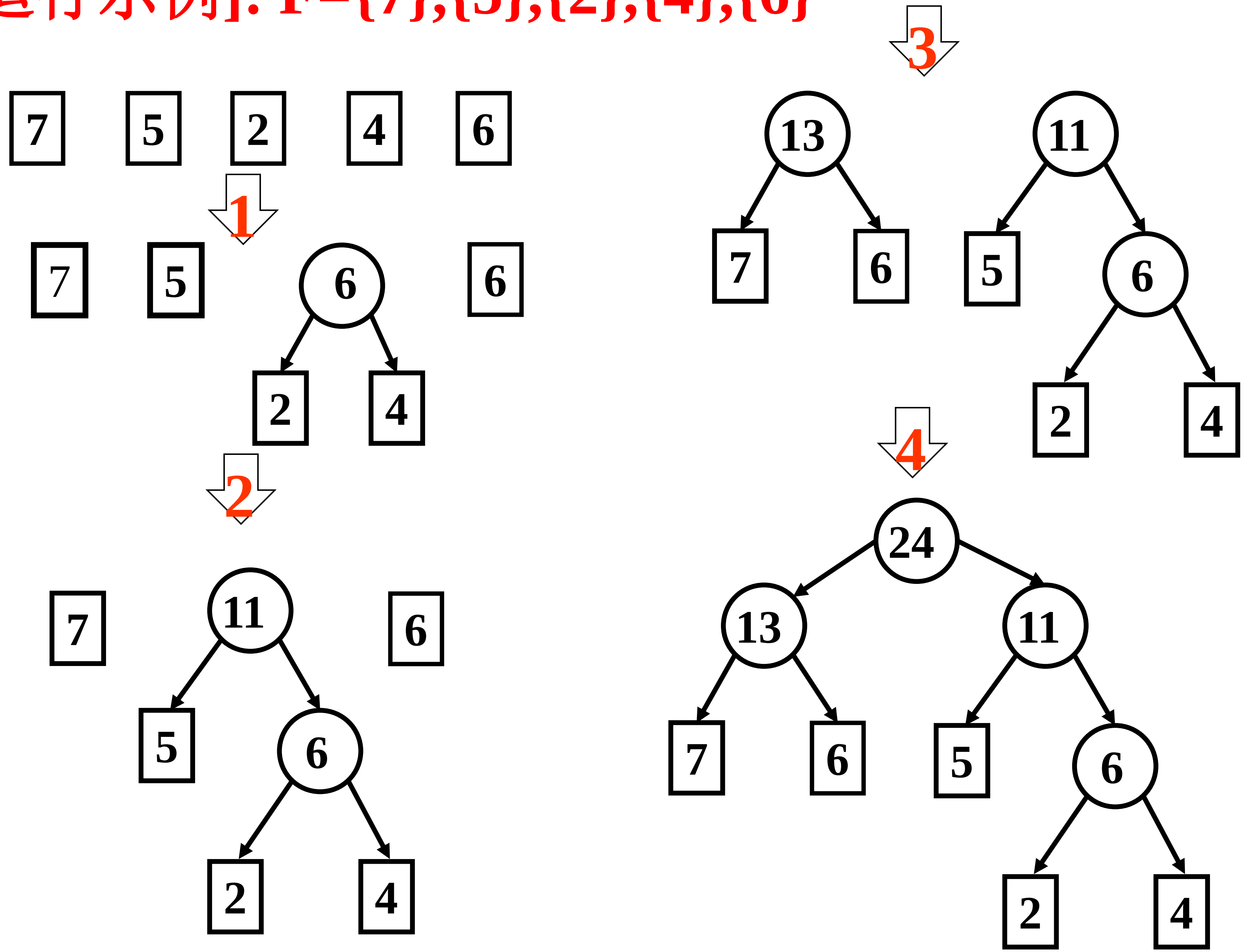
1. 初始化：每个字符都作为一棵只有一个外结点的扩充二叉树，结点的权值定义为字符在文件中出现的次数； n棵二叉树组成了一个森林；

2. 在森林中选取根结点权值最小的两棵二叉树，合并成一棵新二叉树；

- ✓ 生成一个新结点***T1***，作为两个根结点的父结点，***T1***的权值是两个根结点的权值之和。森林减少了一棵树。
- ✓ 优先选择最小权值合并的一个解释：被合并的次数会更多，路径会更长。

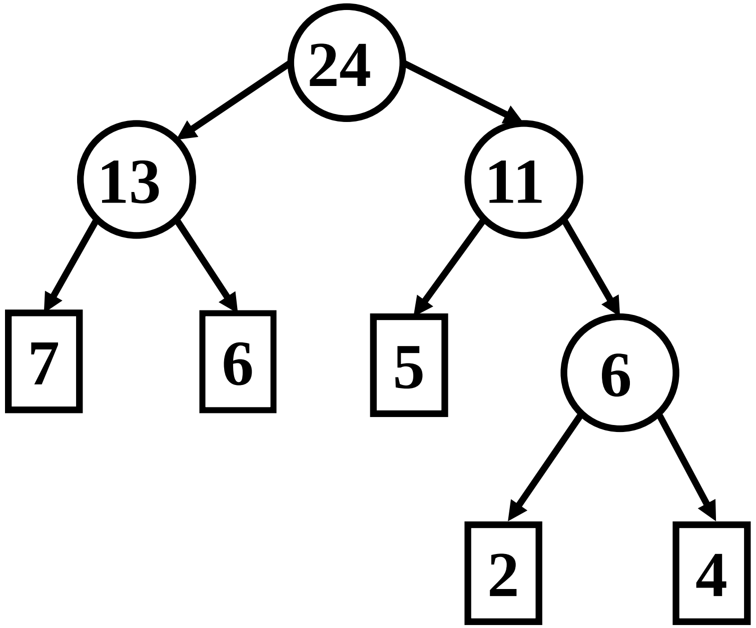
3. 对新森林重复合并操作，直到森林剩余唯一的二叉树

[运行示例]: F={7},{5},{2},{4},{6}



Huffman树

- Huffman算法构造的树
- 根结点的值？
- WPL？

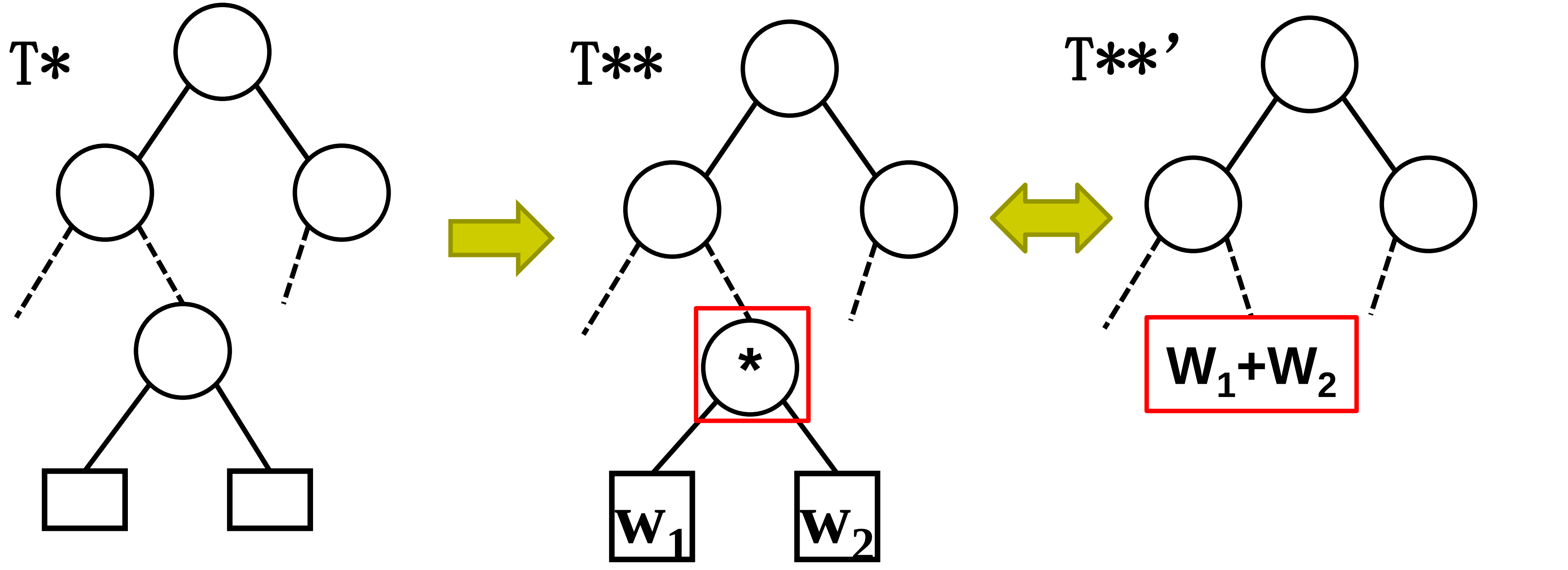


哈夫曼算法的正确性

- 引理1：在带权为w_1<=w_2 <= … <= w_n的所有最优树中，一定有一棵最优树满足w_1、w_2是兄弟结点，且层数是树高；

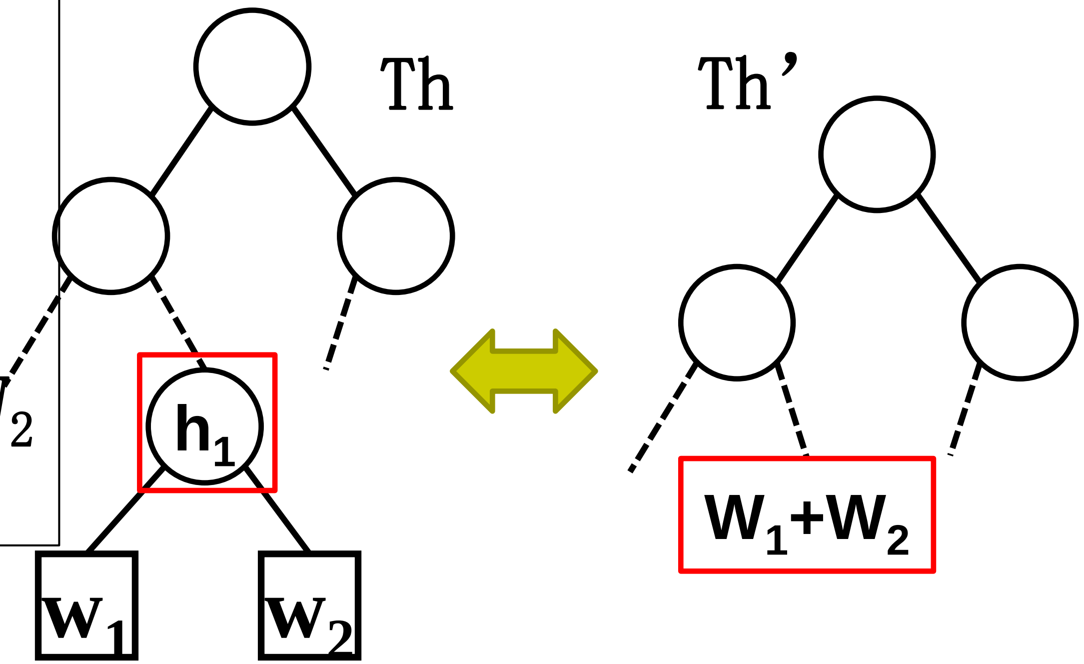
  （C+w1*d1+w2*d2 <= C+w1*d2+w2*d1）

定理5.2 在外结点权值分别为w_1<=w_2 <= … <= w_n的扩充二叉树中，由哈夫曼算法构造出的哈夫曼树的带权路径长度最小，因此哈夫曼树为最优二叉树。



$$EW(T^*) = EW(T^{**})$$
$$EW(T^{**}) \leqslant EW(Th)$$
$$EW(Th') \leqslant EW(T^{**}{}')$$

$$EW(T^{**}) = EW(T^{**}{}') + W_1 + W_2$$
$$EW(Th) = EW(Th') + W_1 + W_2$$



Huffman算法的实现（教材）

- 哈夫曼树中每个结点的结构为：

LLINK	*INFO*	*Weight*	*RLINK*

*其中，**LLINK**和**RLINK**为链接域，**INFO**为信息域，**Weight**为该结点的权值。*

- **指针数组H[n]**

H[1]->Weight <=……<= H[n]->Weight

算法Huffman

H1. [初始化]

```
for( i=1 ; i<=n ; i++ )
    H[i]->LLINK = H[i]->RLINK = 0;  //NULL
```

H2. [组合过程]

```
for( i=1 ; i<=n-1 ;i++ ){
    t          AVAIL⇐ ;
    t -> LLINK = H[i];
    t -> RLINK = H[i+1];
    t -> Weight = H[i]-> Weight + H[i+1]-> Weight ;
```

```
/*把新结点插入到数组H中保持有序*/
for( j = i+2 ; j<=n; j++ )
    if(t->Weight  >  H[j]-> Weight )
        H[j-1] = H[j];
    else break;
 H[j-1] = t;
}
```

时间复杂度$T(n)=O(n^2)$

Huffman算法的优化1

- 多次取最小：堆
  - ✓ 堆中存结点的下标（静态链表）
  - ✓ 初始有n个结点，运行时需n-1个结点
  - ✓ 时间复杂度 O(nlogn)

Huffman算法的优化2

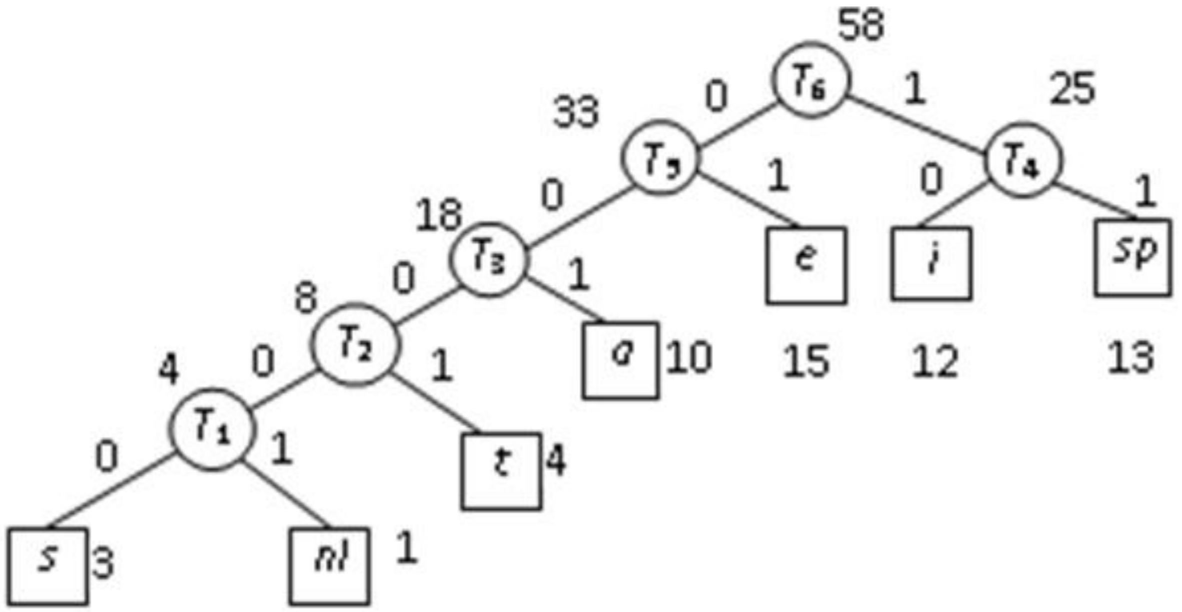
- 把结点分成两类：原始和生成，每种都递增
  - 两个递增序列取最小，O(1)
  - 数组或队列均可，建议数组
  - 时间复杂度 O(n)

Huffman算法的研究

- 并行
- 近似
- ……

哈夫曼编码

- 哈夫曼编码：将哈夫曼树每个分支结点的左分支标上0，右分支标上1，把从根结点到每个叶结点的路径上的标号连接起来，作为该叶结点所代表的字符的编码；



- 哈夫曼编码

  s： 00000

  ……

- 哈夫曼编码的长度

  1×5 + 3×5 + 4×4 + 10×3 + 15×2 + 12×2 + 13×2 = 144 .

  等长码 的长度是 **174** ，

课堂练习

设给出一段报文：

CASTCASTSATATATASA

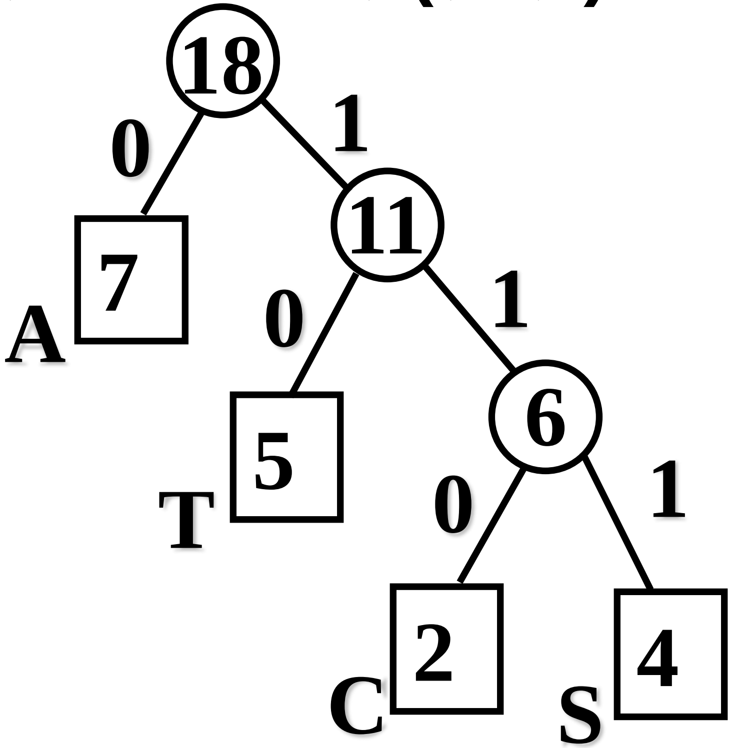
设计一种编码方法，使得报文最短。

参考答案

- 报文： CASTCASTSATATATASA

  字符集合是 { C, A, S, T }

  各个字符出现的频率(次数)是 W={ 2, 7, 4, 5 }



编码： A ： 0

T ： 10

C ： 110

S ： 111

总编码长度：

1*7+2*5+3*2+3*4=35

- 若给每个字符以等长编码

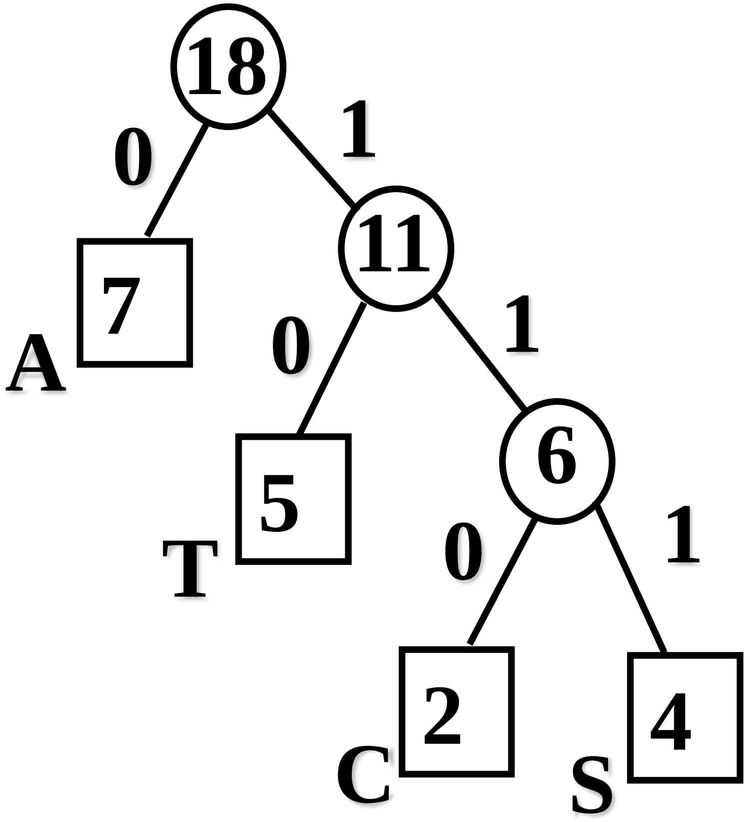
  A : 00　T : 10　C : 01　S : 11

  01001110 01001110 110010 0010 00 10001100

  则总编码长度为 **18 * 2 = 36.**

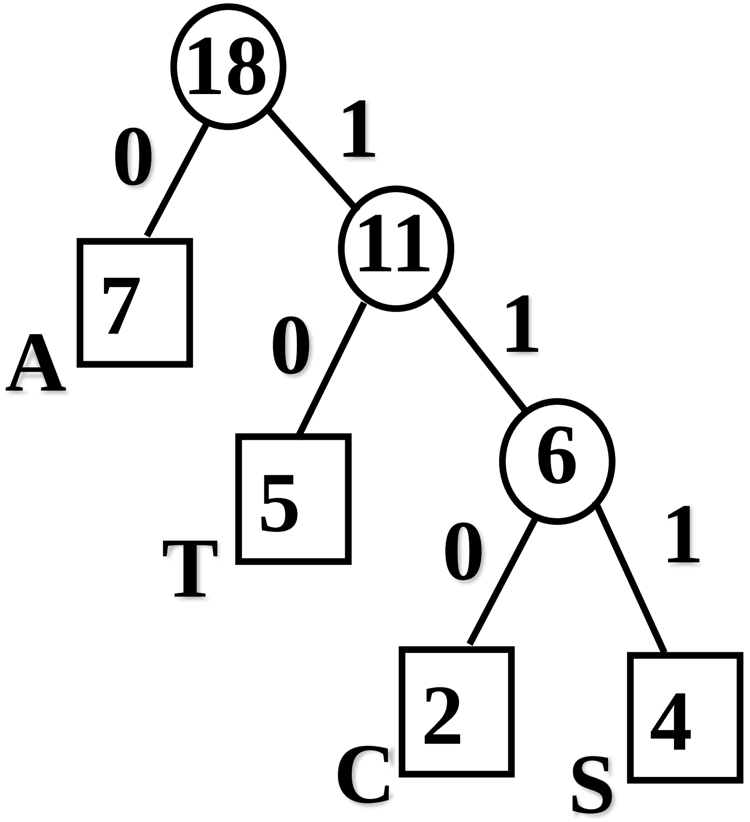
压缩的实现

- 压缩：依次将数据文件中的字符按哈夫曼树转换成哈夫曼编码。
- 将哈夫曼树存储在压缩文件的开始部分；
  - ✓ 字符及次数、字典、……
- 二进制文件读写
  - ✓ 16/32位写入和读取；
  - ✓ 末位补足或记录位数；



解压的实现

- 解压：重构哈夫曼树；
- 依次读入文件的二进制码（与写入对应），从哈夫曼树的根出发，若读入0，则向左走孩子，否则向右走，到达某一叶结点时，译出相应的字符。



思考

- 哈夫曼编码是最优的压缩方法吗？
  - ✓ 哈夫曼编码一种最常用的无损压缩编码方法。
  - ✓ 没考虑其它特性，如重复；
  - ✓ 压缩有时可以有损；
- 哈夫曼编码是否唯一？
- **如何构造k元huffman树（*k* >2）？**
  - ✓ 每次选取k个结点合并
  - ✓ 未必每次都能凑足k个，例如n=6，k=3

拓展1

- 命题：设 n 个内结点的扩充二叉树的外通路长度和内通路长度分别为E(n)和I(n)，则**E(n) = I(n) + 2n，(n≥0).** （**课后习题5-11**）

- 数学归纳法。对n进行归纳

证明

n=0时，E(0)=0, I(0) =0，命题成立。

假设n=k时成立，往推 n = k+1时也成立。

当n=k+1时，找到具有两个外结点的内结点s,

其两个外结点记为t_1,t_2.(一定能找到).

把s当成外结点，有E(k) = I(k) + 2k

设s的深度为d，则

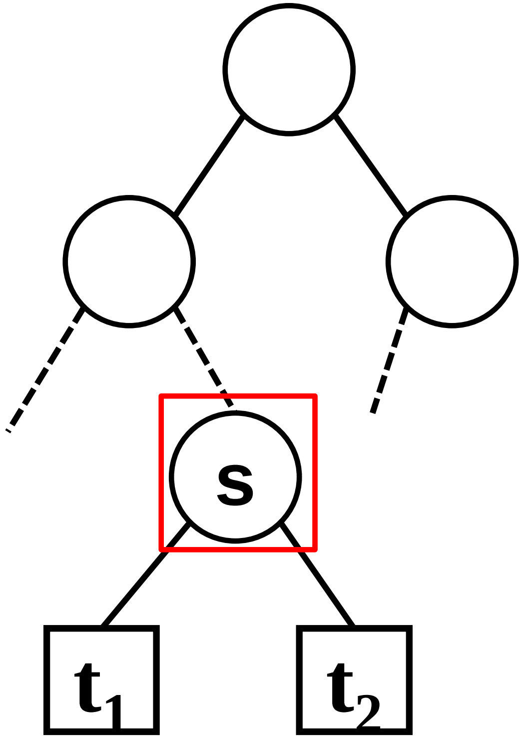
E(k+1) = E(k) – d + 2(d+1)

I(k+1) = I(k) + d

从而: E(k+1) = E(k)+d+2

= I(k) + 2k + d +2

= I(k+1) + 2k + 2 。



拓展2

- n个内结点的扩充二叉树的内通路长度I(n)的最大值为n(n-1)/2，最小值为(n+1)k -2^{k+1}+2(k=[logn]，[ ]为floor)

- 分析：最大值即为一条链，最小值为完全二叉树，证明参考堆。

总结

1. **压缩和编码**
2. **Huffman算法**
   - ✓ 扩充二叉树及一些相关定义;
   - ✓ Huffman算法的思想（自然语言描述）；
   - ✓ Huffman算法的正确性;
3. **Huffman算法的实现**
   - ✓ 教材方法和优化;
4. **Huffman编码的实现**
5. **扩充二叉树的两个常用结论**