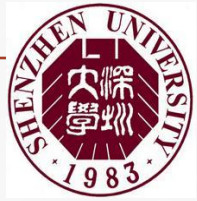
第五章 树与二叉树（三）

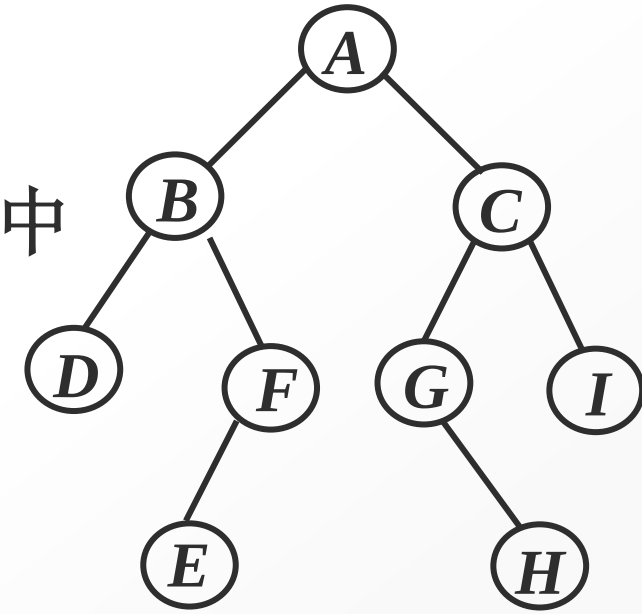


回顾

- 二叉树的遍历
- 根据先序和中序序列，或者后序和中序序列可以唯一确定一棵二叉树
- 树的存储结构

1. 双亲表示法
2. 孩子链表表示法
3. 孩子兄弟表示法

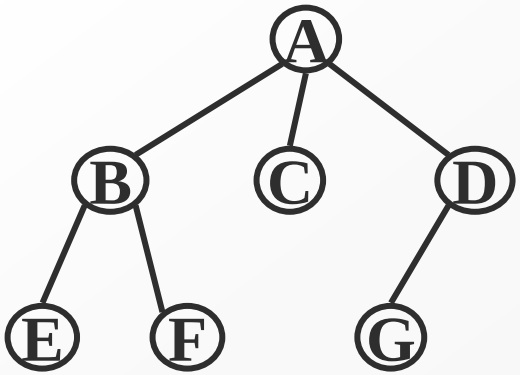
- 森林(树)与二叉树的对应关系



先序遍历序列为：A B D F E C G H I
中序遍历序列为：D B E F A G H C I
后序遍历序列为：D E F B H G I C A

树的遍历

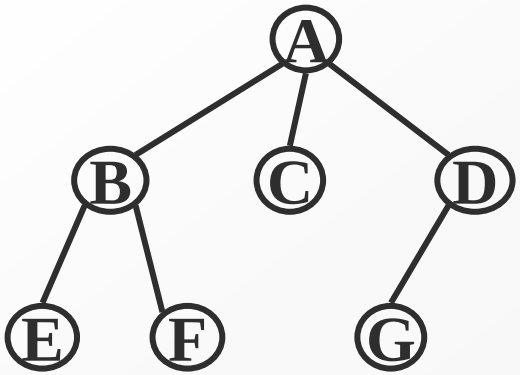
- 先根（次序）遍历
  - 当树非空时，
    I. 访问根结点；
    II. 依次先根遍历根的各颗子树



- 输出结果：ABEFCDG

树的遍历（cont.）

- 后根（次序）遍历
  - 当树非空时，
    I. 依次后根遍历根的各颗子树；
    II. 访问根结点

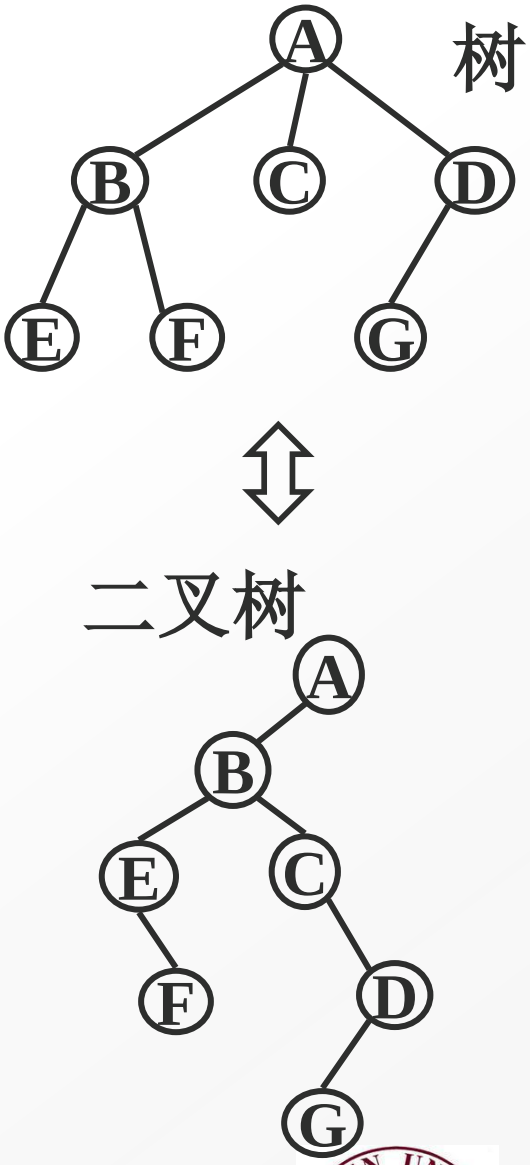


- 输出结果：EFBCGDA

树的遍历（cont.）

- 与二叉树遍历的关系

  - 树的先根遍历序列与对应的二叉树的先序遍历序列相同；
  - 树的后根遍历序列与对应的二叉树的中序遍历序列相同



遍历	树	二叉树
先序	ABEFCDG	ABEFCDG
中序	EBFACGD	EFBCGDA
后序	EFBCGDA	FEGDCBA

森林的遍历

- 先根遍历

I. 访问第一株树的根结点；

II. 按先根顺序遍历第一棵树的子树森林；

III. 按先根顺序遍历其余子树森林

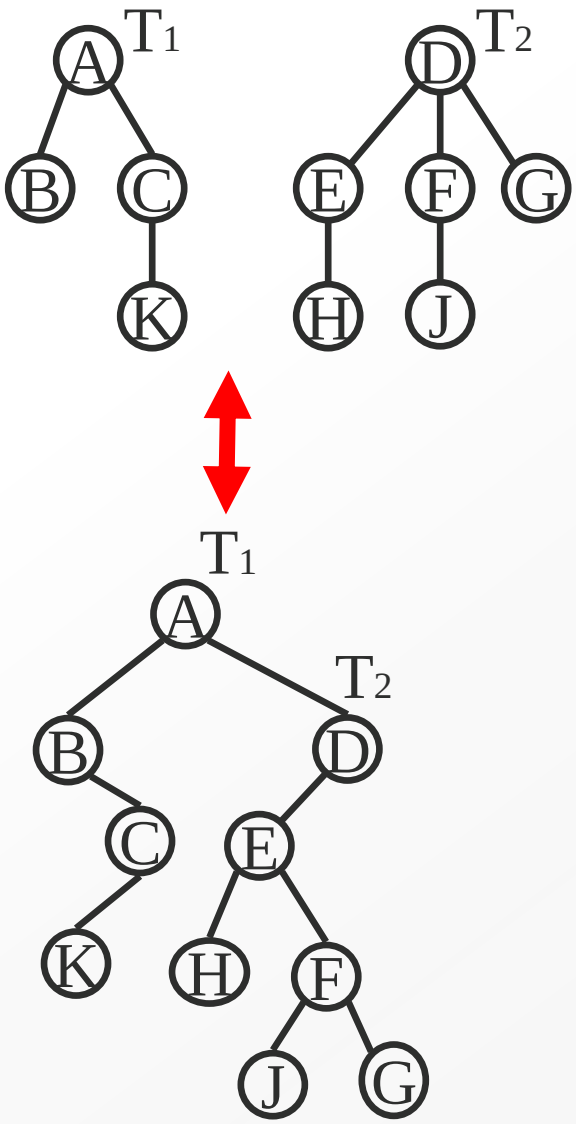
- 后根遍历

I. 按后根顺序遍历第一株树的子树森林；

II. 访问第一株树的根结点；

III. 按后根顺序遍历其余子树森林

森林的遍历（cont.）



遍历	森林	二叉树
先序	√	√
中序	×	√
后序	√	√

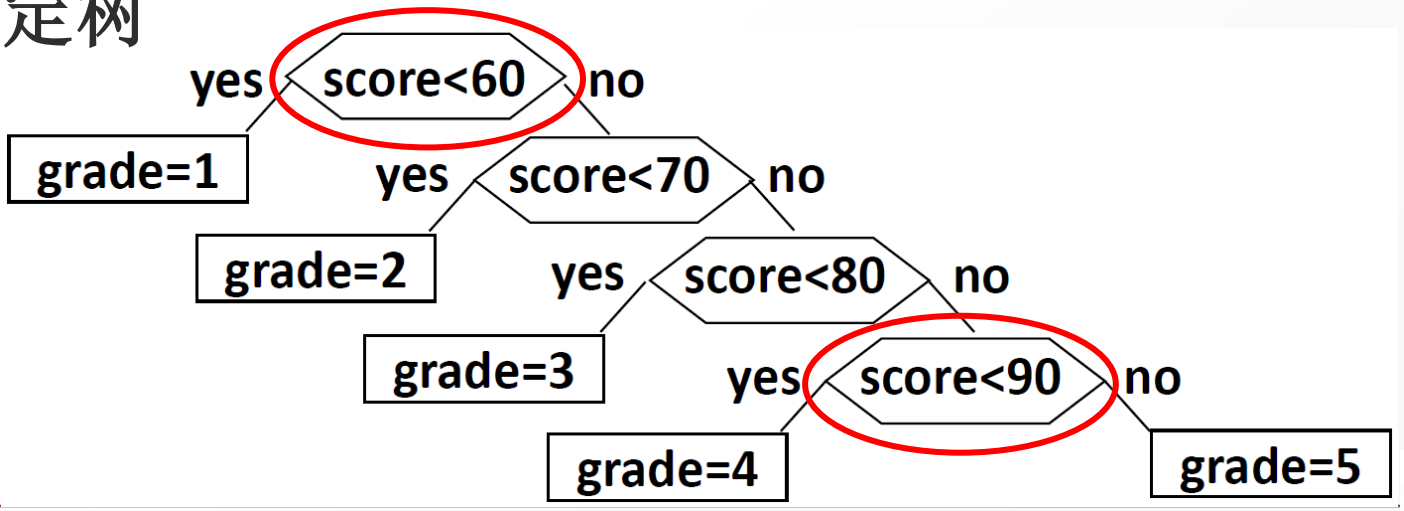
赫夫曼树及其应用

➢什么是赫夫曼树（Huffman Tree）？

［例］将百分制的考试成绩转换成五分制的成绩

```
if ( score < 60 ) grade =1;
else if ( score < 70 ) grade =2;
else if ( score < 80 ) grade =3;
else if ( score < 90 ) grade =4;
else grade =5;
```

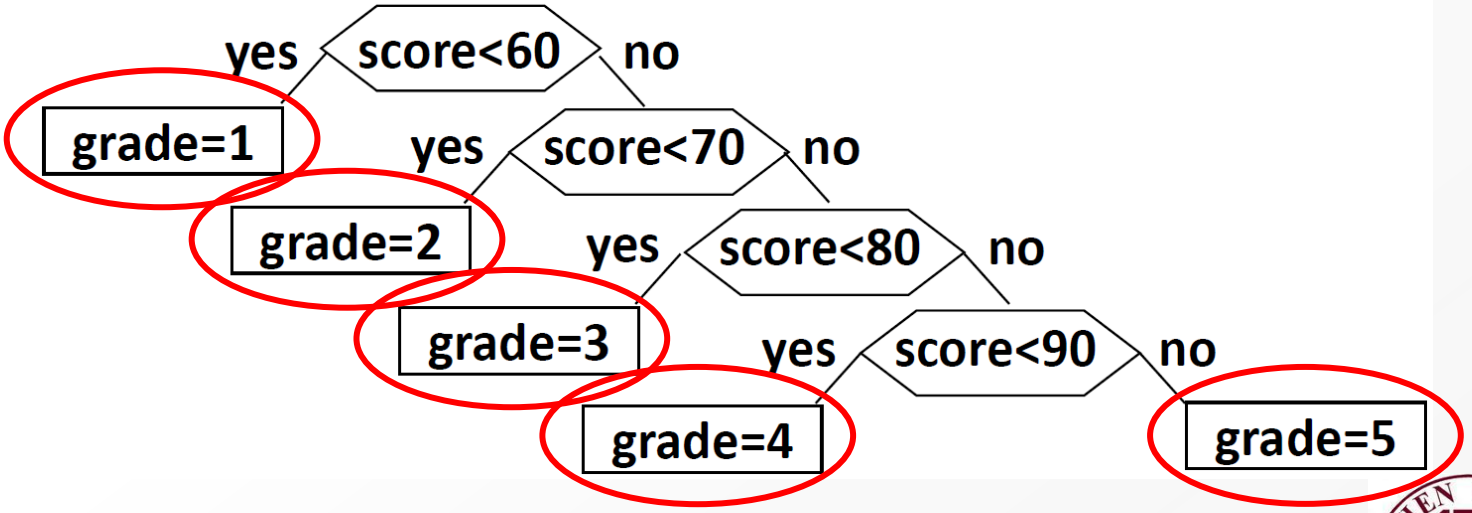
- 判定树



- 如果考虑学生成绩的分布的概率：

分数段	0-59	60-69	70-79	80-89	90-100
频率	0.05	0.15	0.40	0.30	0.10

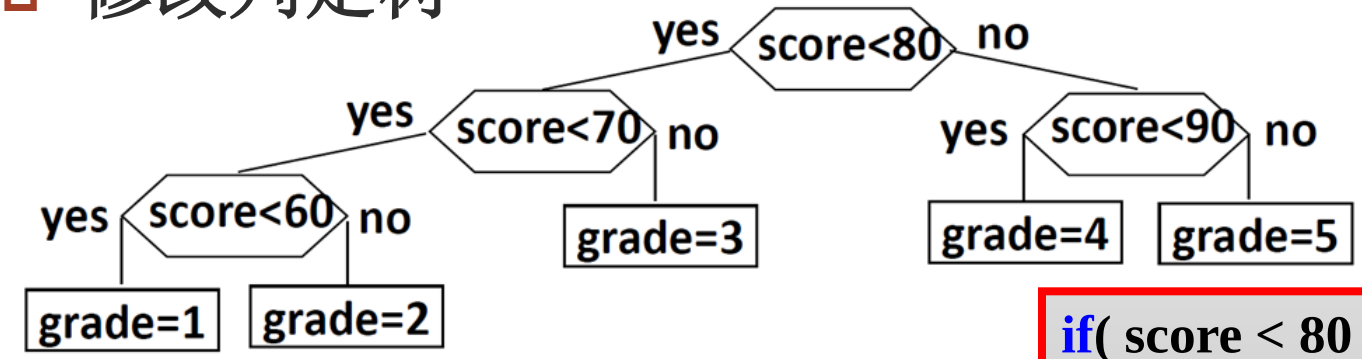
查找效率： $0.05\times1+0.15\times2+0.4\times3+0.3\times4+0.1\times4=3.15$



- 如果考虑学生成绩的分布的概率：

分数段	0-59	60-69	70-79	80-89	90-100
频率	0.05	0.15	0.40	0.30	0.10

- 修改判定树



查找效率：

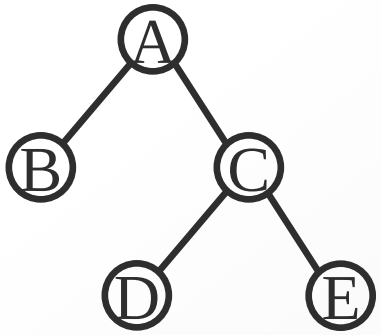
$0.05\times3+0.15\times3+0.4\times2+0.3\times2+0.1\times2 = 2.2$

```
if( score < 80 )
{   if( score < 70 )
        if( score < 60 ) grade =1;
        else grade = 2;
}else if( score < 90 ) grade =4;
else grade =5;
```

如何根据结点不同的查找频率构造更有效的树形结构？

最优二叉树

- 路径：从树中一个结点到另一个结点之间的分支构成这两个结点之间的路径
- 路径长度：路径上的分支数目
- 树的路径长度：从树根到每个结点的路径长度之和

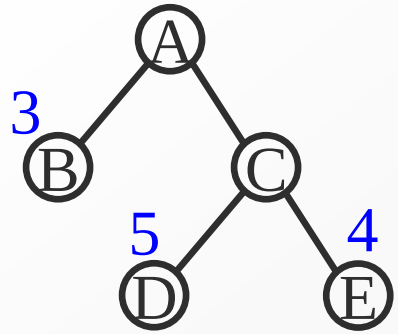


树的路径长度为：$1\times2+2\times2=6$

最优二叉树（cont.）

- 带权路径长度：从结点到树根之间的路径长度与结点上权的乘积
- 树的带权路径长度（ **WPL** , **Weighted Path Length**）：树中所有叶子结点的带权路径长度之和：

假设二叉树有n个叶子结点，每个叶子结点带权为ω_k，从根结点到每个叶子结点的长度为l_k，则带权路径长度 **WPL**=$\sum_{i=1}^{n} \omega_i \, l_k$



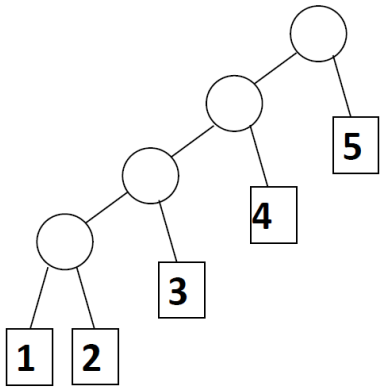
WPL= 1×3+2×5+2×4=21

最优二叉树（cont.）

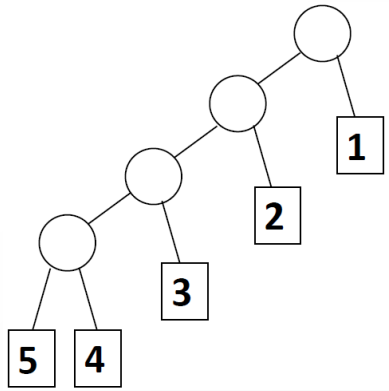
- 最优二叉树或赫夫曼树：带权路径长度WPL最小的二叉树

- 在赫夫曼树中，权值最大的结点离根最近，权值最小的结点离根最远

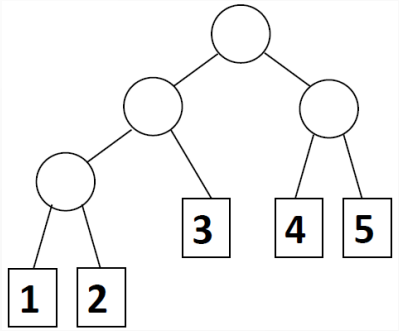
[例] 有五个叶子结点，它们的权值为{1, 2, 3, 4, 5}，用此权值序列可以构造出形状不同的多个二叉树:



WPL = 34



WPL = 50



赫夫曼树

WPL = 33

赫夫曼树的构造

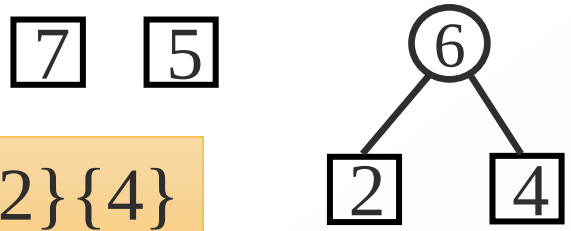
1. 将这n个权值为$(\omega_1, \omega_2, \ldots, \omega_n)$的结点构成$n$棵仅含一个根结点的二叉树，构成森林$F$
2. 在F中选取两棵根结点的权值最小的树，作为左右子树构成一棵新的二叉树，且置其根节点的权值为左右子树的权值之和
3. 从F中删除这两棵树，同时将新得到的树加入F中。
4. 重复步骤2和3，直至F中只剩下一棵树为止。

赫夫曼树的构造（cont.）

F：{7}{5}{2}{4}

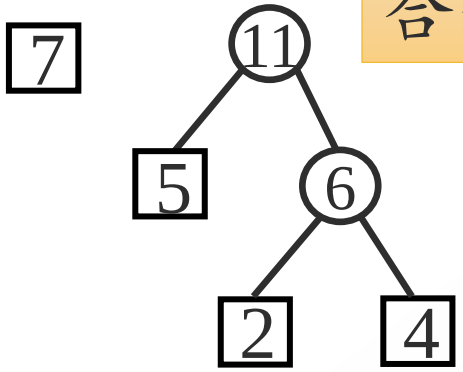
初始

F：{7}{5}{6}



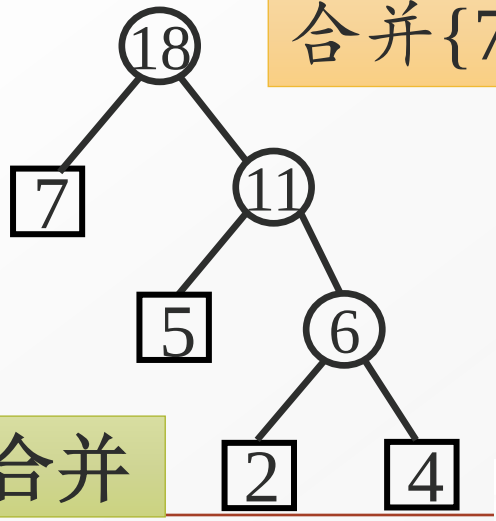
合并{2}{4}

F：{7}{11}



合并{5}{6}

F：{18}



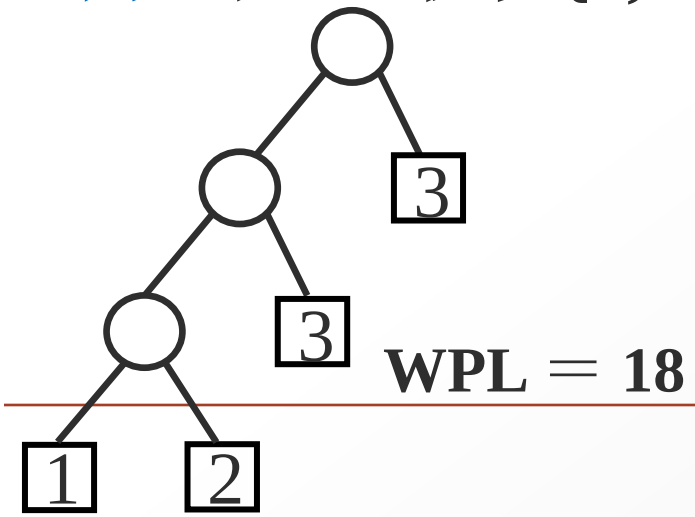
合并{7}{11}

✓ 每次把**权值最小的两棵**二叉树合并

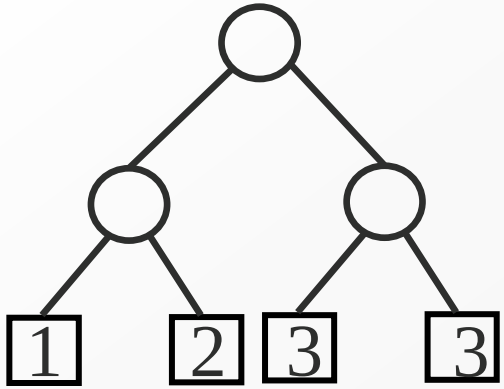
赫夫曼树的特点

- 没有度为1的结点
- n个叶子结点的赫夫曼树共有$2n-1$个结点
- 赫夫曼树的任意非叶节点的左右子树交换后仍是赫夫曼树
- 对同一组权值$\{\omega_1, \omega_2, \ldots, \omega_n\}$，是否存在不同构的两棵赫夫曼树呢？

[例]对一组权值{1, 2, 3, 3}，不同构的两棵赫夫曼树：



WPL = 18



WPL = 18

赫夫曼编码

- 给定一段字符串，如何对字符进行编码，可以使得该字符串的编码存储空间最少？

[例] 假设有一段文本，总共包含58个字符，并由以下7个字符构成：a, e, i, s, t, 空格(sp), 换行(nl), 且这7个字符出现的次数不同。如何对这7个字符进行编码，使得总编码空间最少？

[分析]

（1）用等长ASCII编码：58 ×8 = 464位；

（2）用等长3位编码：58 ×3 = 174位；

（3）不等长编码：出现频率高的字符用的编码短些，出现频率低的字符则可以编码长些？

怎么进行不等长编码？

a: 1
e: 0
s: 10
t: 11
…

1011是什么字符串的编码？

aeaa：1 0 1 1
aet：1 0 11
st：10 11

- 如何避免二义性？

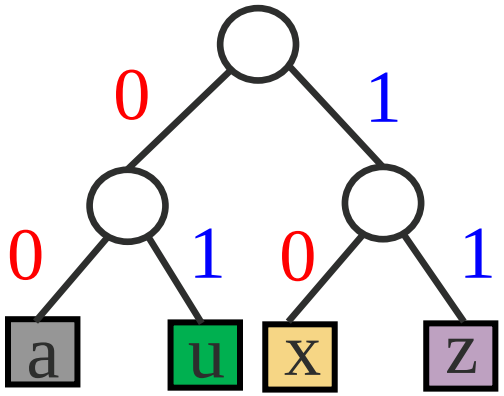
- 前缀码 *prefix code*：任何字符的编码都不是另一字符编码的前缀
  - 可以无二义地解码

二叉树用于编码

- 用二叉树进行编码：

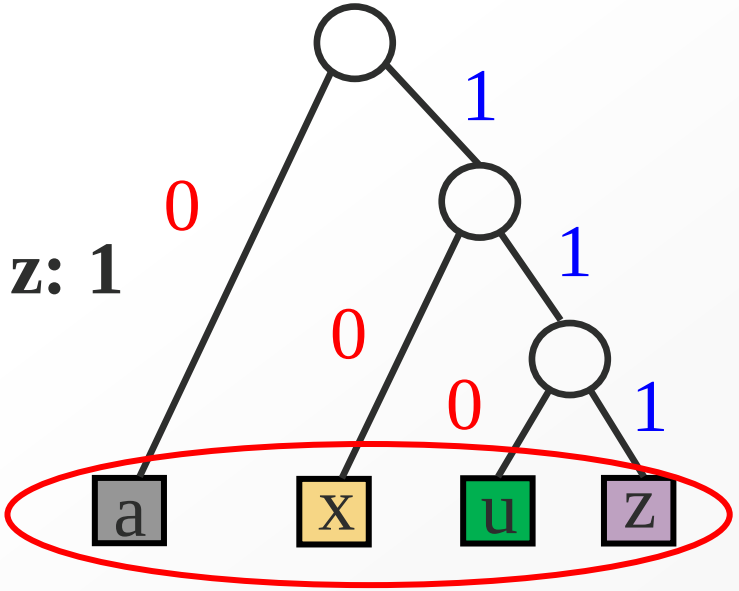
1. 左右分支：0、1
2. 字符只在叶结点上

四个字符的频率: a: 4, u: 1, x: 2, z: 1



aaaxuaxz → 0000001001001011

总长= 2×4 + 2×1 + 2×2 + 2×1 = 16



aaaxuaxz → 000101100101111

总长= 1×4 + 3×1 + 2×2 + 3×1 = 14

赫夫曼编码

- 对于给定的字符集及其每个字符出现的概率（使用频度），求该字符集的最优的前缀性编码—赫夫曼编码问题

➢ 用赫夫曼算法求字符集最优前缀编码的算法：

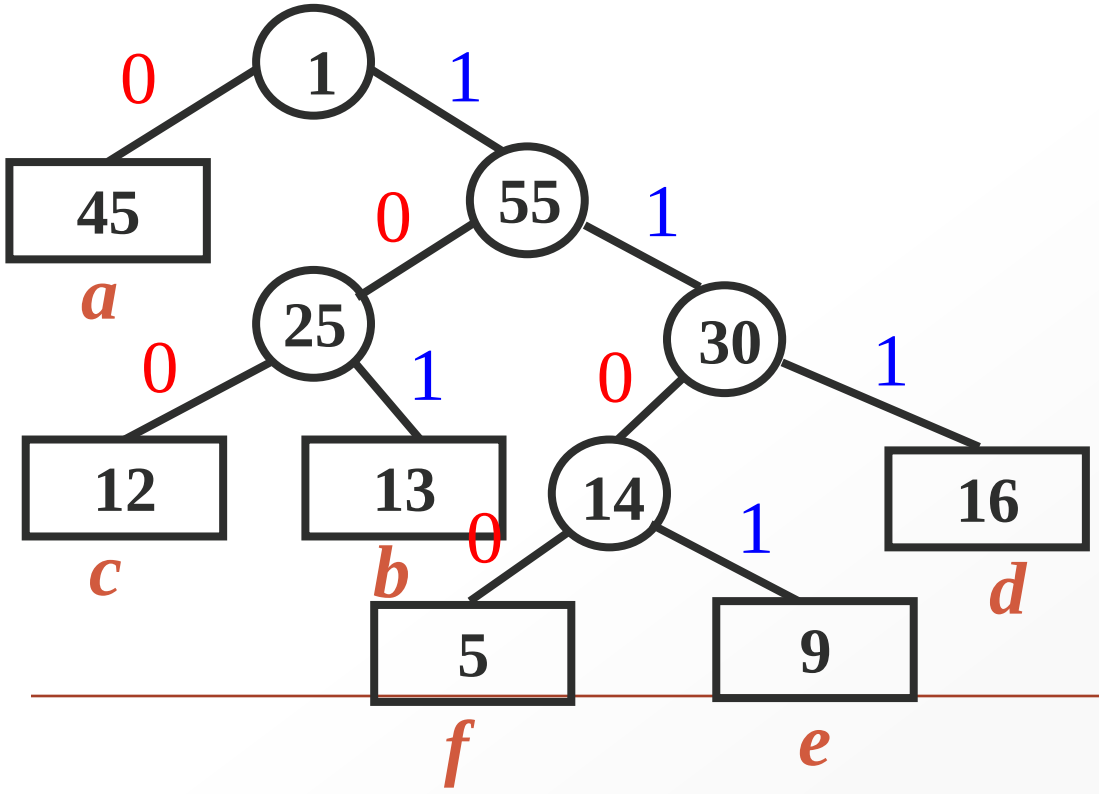
I. 使字符集中的每个字符对应一棵只有叶结点的二叉树，叶的权值为对应字符的使用频率---初始化

II. 利用huffman算法来构造一棵huffman树---构造算法

III. 对huffman树上的每个结点，左、右支分别附以0、1，则从根到叶结点路径上的分支编码（0、1序列）就是相应字符的编码---哈夫曼编码

赫夫曼编码示例

字符	*a*	*b*	*c*	*d*	*e*	*f*	总码长
频率	45	13	12	16	9	5	
等长	000	001	010	011	100	101	**300**
变长	0	101	100	111	1101	1100	**224**



	ch	bits
0	a	0
1	b	101
2	c	100
3	d	111
4	e	1101
5	f	1100
6		

编码表 H