智慧决策筑梦，人生优化前行

第17讲 并查集

信息学院（智能应用研究院）
欧 新 宇

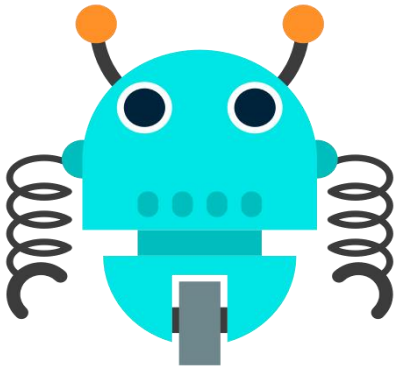
第5章 树和二叉树

考核要点

- **考核大纲**

树的基本概念及存储结构；二叉树的定义、主要特征及其存储结构，二叉树的遍历，线索二叉树的基本概念和构造；森林与二叉树的的转换；树和森林的遍历；哈夫曼树和哈夫曼编码。

- **复习要点**
  - 掌握树和二叉树的性质、遍历操作、存储结构和操作特性
  - 了解满二叉树、完全二叉树、线索二叉树、哈夫曼树的定义、性质和思想
  - 熟练掌握二叉树的前、中、后序遍历方法及算法
  - 熟练掌握哈夫曼树的实现方法及构造哈夫曼编码的方法
  - 掌握森林和二叉树的转换方法



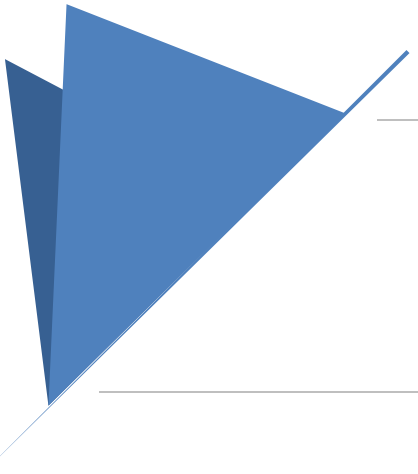
- 树的路径长度
- 哈夫曼算法和哈夫曼树
- 哈夫曼编码

第13讲 哈夫曼树及其应用

问题导入（文件传输）

如何有效地实现对文本/数据的压缩和编码？

使用尽量少的编码完成字符的转换（提高存储和传输效率）
对已经编码的数据能够实现无损解码



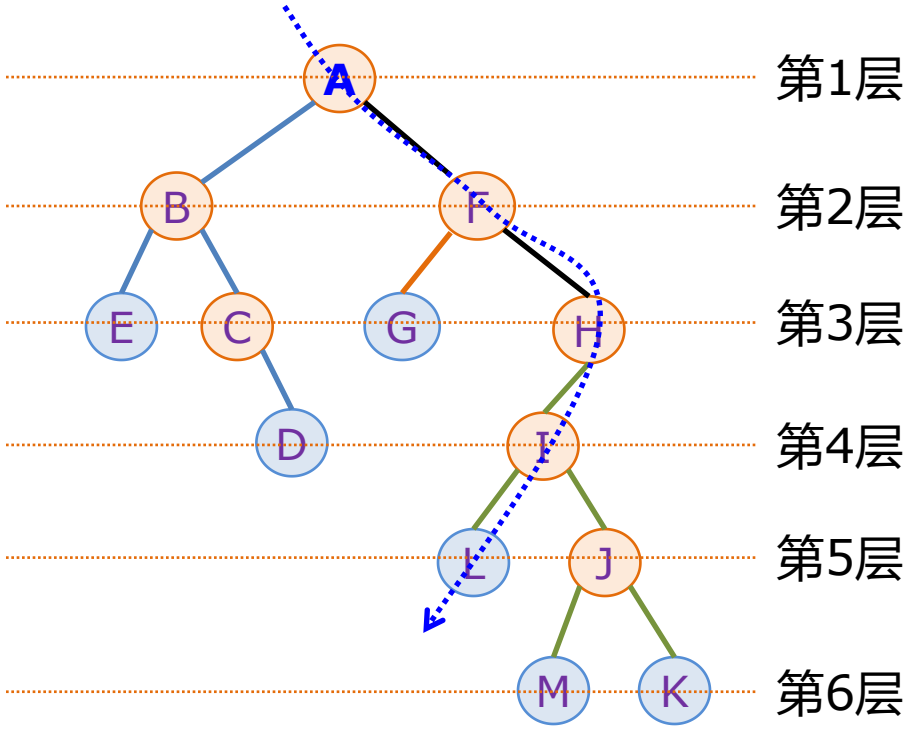
树的路径长度

/ 树的路径、内路径和外路径的长度

/ 树的加权路径长度

树的路径长度

树的路径、内路径和外路径



结点L的路径长度 = 4

结点的**路径长度**是指从根结点到该结点的路径上所包含的边的数目。

树的**内路径**长度：

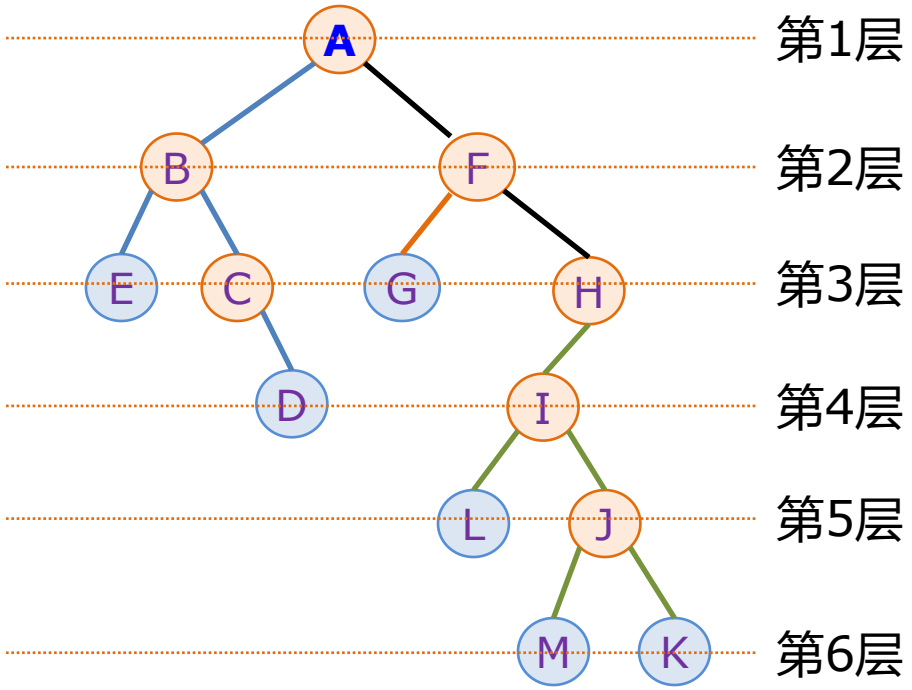
除叶子结点外，从根结点到树中其他所有结点的路径长度之和。

非叶子结点	路径长度
A	0
B	1
C	2
F	1
H	2
I	3
J	4

树的内路径长度 = 1+2+1+2+3+4 = 13

树的路径长度

树的路径、内路径和外路径



结点L的路径长度 = 4

结点的**路径长度**是指从根结点到该结点的路径上所包含的边的数目。

树的**外路径**长度：

从根结点到树中所有叶子结点的路径长度之和。

叶子结点	路径长度
E	2
D	3
G	2
L	4
M	5
K	5

树的外路径长度 = 2+3+2+4+5+5 = 21

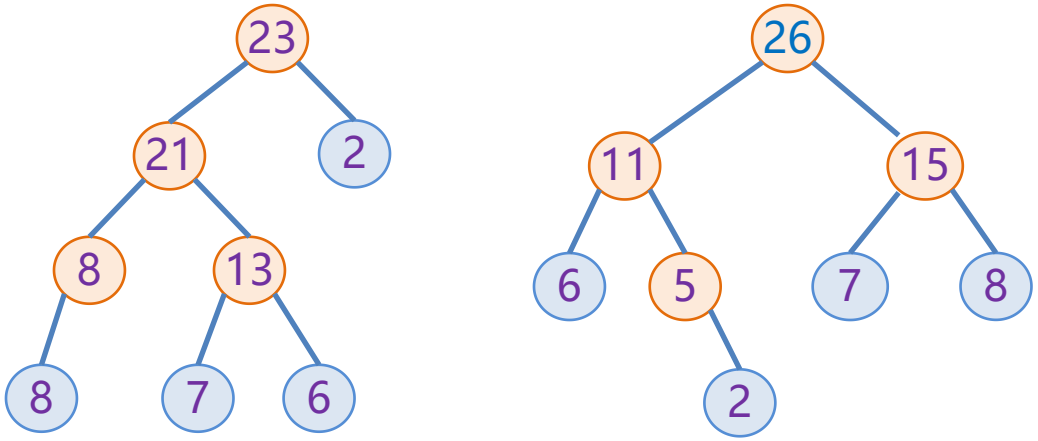
树的路径长度

树的加权路径

- **叶结点的加权路径长度：**叶结点的路径长度与权值的乘积。
- **树的加权路径长度：**所有叶结点的加权路径长度之和，记为**WPL**。设m是叶结点的数量，w_k是第k个叶结点的权值，l_k是该叶结点的路径长度，则有：

$$WPL = \sum_{k=1}^{m} w_k \times l_k$$

例1：试比较以下两棵树的加权路径长度。



$WPL_1 = 8\times3+7\times3+6\times3+2\times1 = 65$

$WPL_2 = 6\times2+2\times3+7\times2+8\times2 = 48$

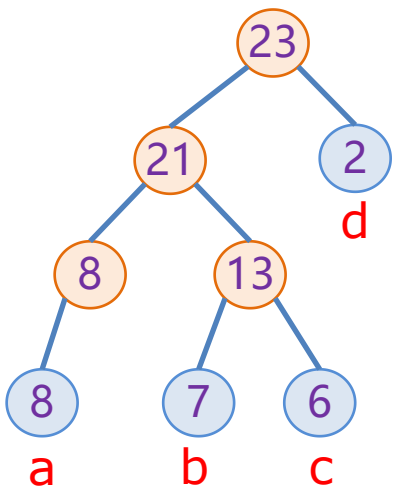
$WPL_1 > WPL_2$

树的路径长度

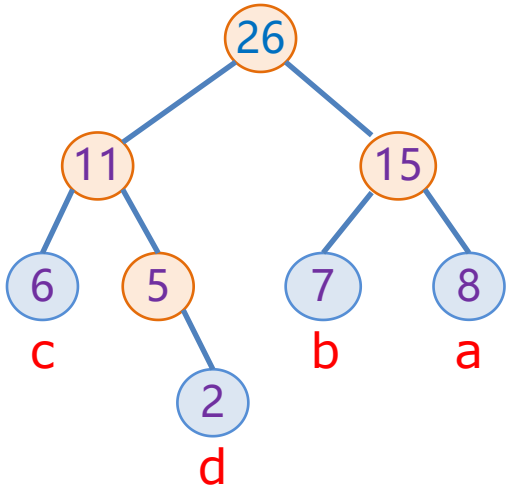
加权路径长度WPL的内涵

以字符编码应用场景为例：

- ✓ 每个叶结点表示一个字符
- ✓ 叶结点的权值表示该字符在文本中出现的频度
- ✓ 叶结点的路径长度表示该字符的编码长度



$WPL_1 = 67$

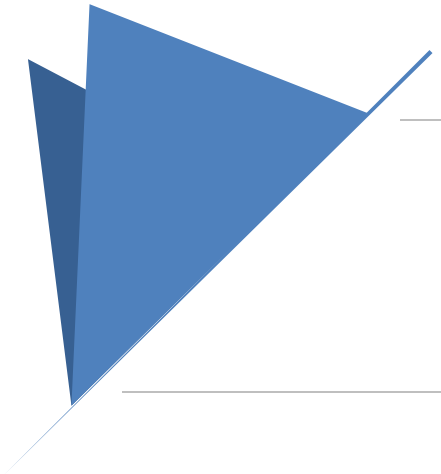


$WPL_2 = 48$

WPL表示一个文本转换为编码后最终的总编码长度

如何构建具有最小加权路径长度的扩充二叉树实现总编码长度的**压缩?**

哈夫曼树



哈夫曼树

/ 哈夫曼算法

/ 哈夫曼树

哈夫曼树

哈夫曼算法和哈夫曼树

哈夫曼算法

求具有最小加权路径长度的二叉树的算法

哈 夫 曼 树

用哈夫曼算法构造生成的二叉树

哈夫曼树的特点

- ✓ 哈夫曼树是一棵扩充二叉树
- ✓ 哈夫曼树中任一非叶节点的权值等于其左右孩子的权值之和
- ✓ 对于任意叶节点相同的二叉树而言，哈夫曼树的加权路径长度是最小的

哈夫曼树

哈夫曼算法的基本思想

1. 根据一组权值 $\{w_1,w_2,\ldots,w_n\}$，构造n棵只有根结点的森林 $F=\{T_1,T_2,\ldots,T_n\}$。

2. 从F中选取根结点权值最小的两棵树做为左右子树，构造一棵新的二叉树，左右两棵子树的权值的和做为新二叉树根结点的权值。

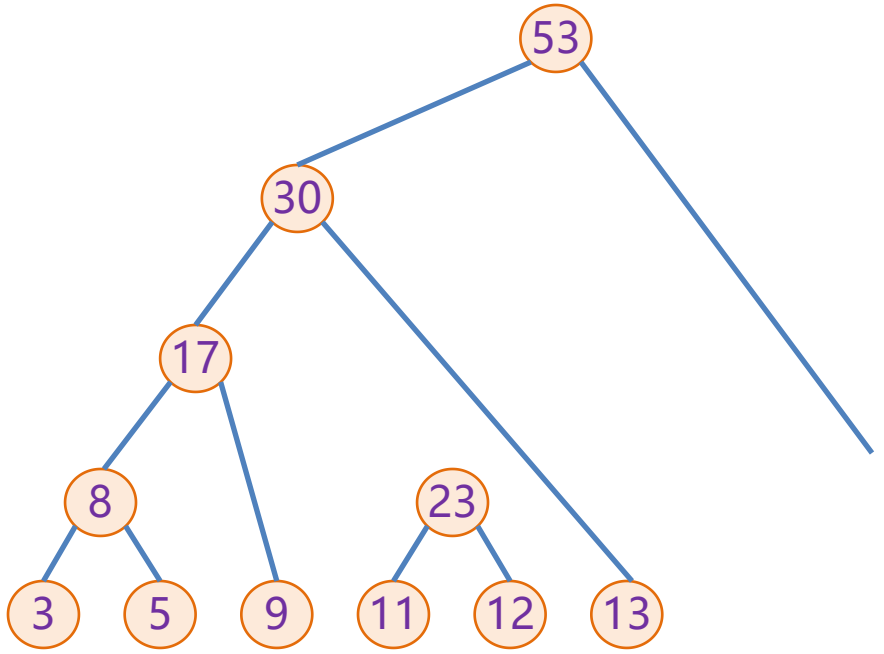
3. 删除被选出的两棵子树，并将新二叉树加入森林。

4. 重复**步骤2**和**步骤3**，直至只含有一棵二叉树为止，该树即为哈夫曼树。

哈夫曼树

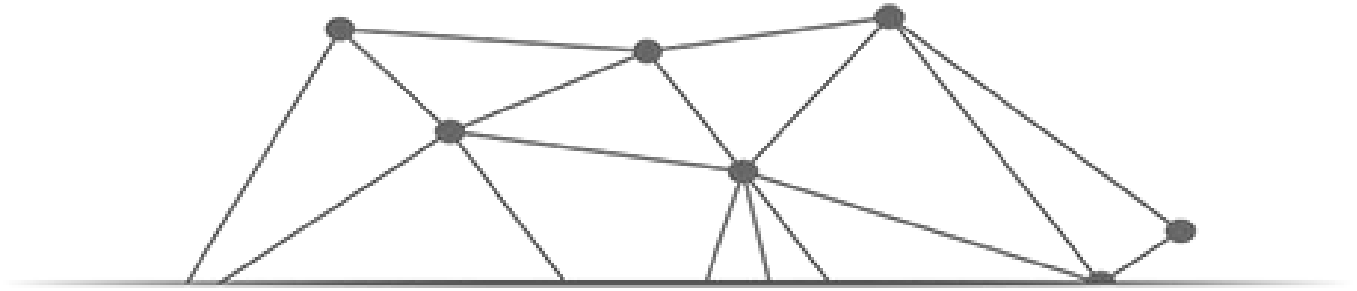
哈夫曼算法的示例

例1：已知权值集合W = {3, 5, 9, 11, 12, 13}。

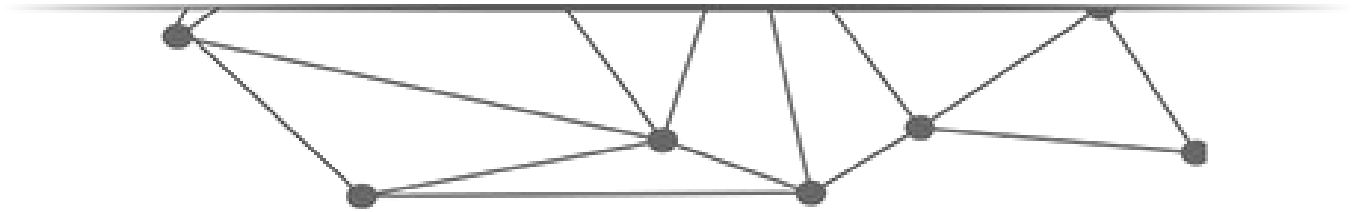


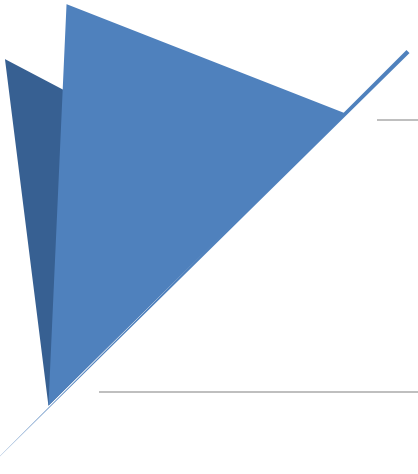
哈夫曼树是一棵扩充二叉树

哈夫曼树中任一非叶结点的权值等于其左右孩子的权值之和



课堂互动





哈夫曼编码

/ 等长编码和不等长编码

/ 基于二叉树设计的前缀编码

/ 基于哈夫曼树构建前缀编码

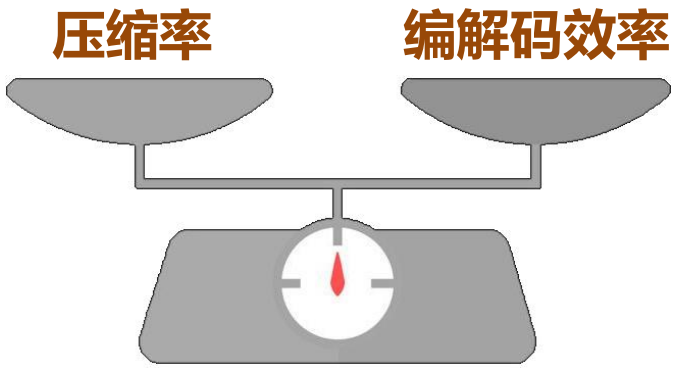
/ 哈夫曼编码的解码

哈夫曼编码

压缩编码问题

1. 无二义性，即编码/解码具有双向唯一性；

2. 权衡好压缩率和编解码效率的平衡性



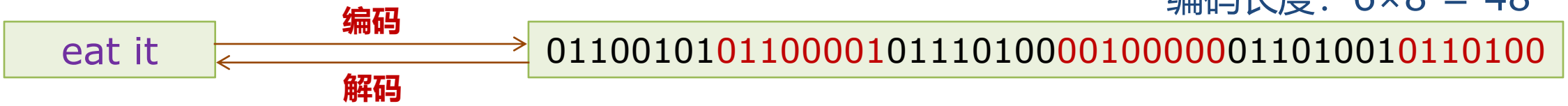
哈夫曼编码

例2：假设有一段文本，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）；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？

1. 等长ASCII编码（注：ASCII是一种国际通用的8位二进制码，a: 01100001, e: 01100101, i: 01101001, s: 01110011, t: 01110100, 空格: 00100000, 换行: 00011000）

总长度：58×8 = 464

编码长度：6×8 = 48



2. 等长的3位编码（a: 000, e: 001, i: 010, s: 011, t: 100, sp: 101, nl: 110）

总长度：58×3=174

编码长度：6×3 = 18

三位编码明显具有更好的编码效率，但是否还有更有效的办法？二位编码是否可行？ NO

不等长编码！

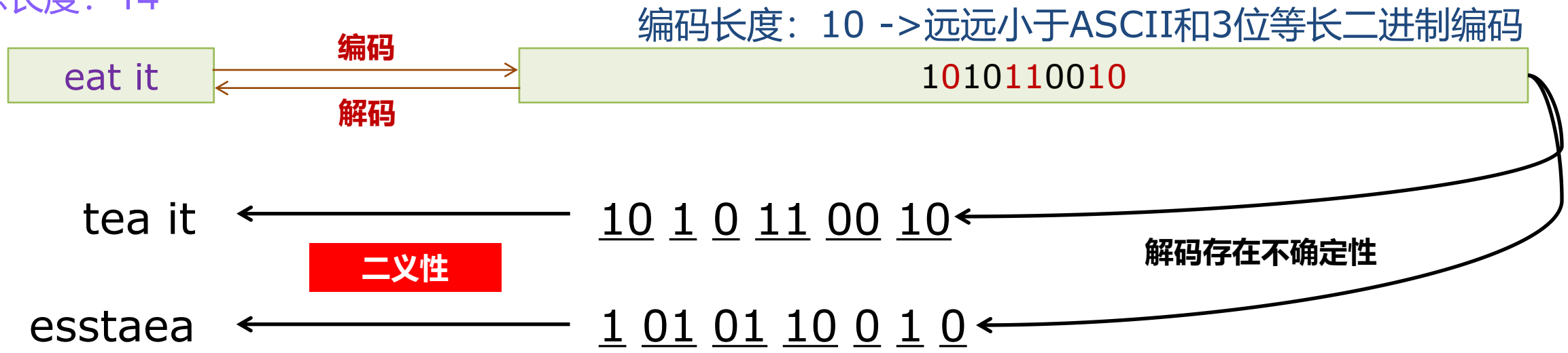
哈夫曼编码

例2：假设有一段文本，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）；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？

3. 不等长编码：每个元素对应的编码长度不一定相等

a: 0, e: 1, i: 00, s: 01, t: 10, sp: 11, nl: 001

总长度：14



原因：一个字符的编码可能是另一个字符编码的前缀

要设计不等长编码，必须使任一字符的编码都不是另一个字符的前缀——>前缀编码

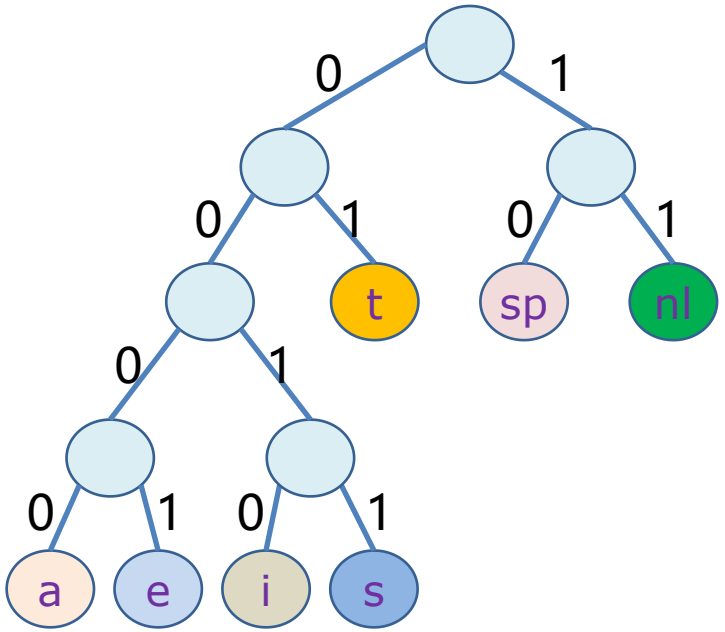
哈夫曼编码

使用二叉树设计前缀编码

任一字符的编码都不是另一字符编码的前缀

二叉树编码**原则**：

1. 左分支：0；右分支：1
2. 字符只在叶子结点
3. 字符编码的长度为根结点到叶子节点的路径长度；
4. 编码为经过路径的编码



字符	编码
a	0000
e	0001
i	0010
s	0011
t	01
sp	10
nl	11

文本：tea → 编码：01 0001 0000 (码长：10)

什么样的二叉树才是最优的？ => 最终综合编码长度最短 => 效率最高？

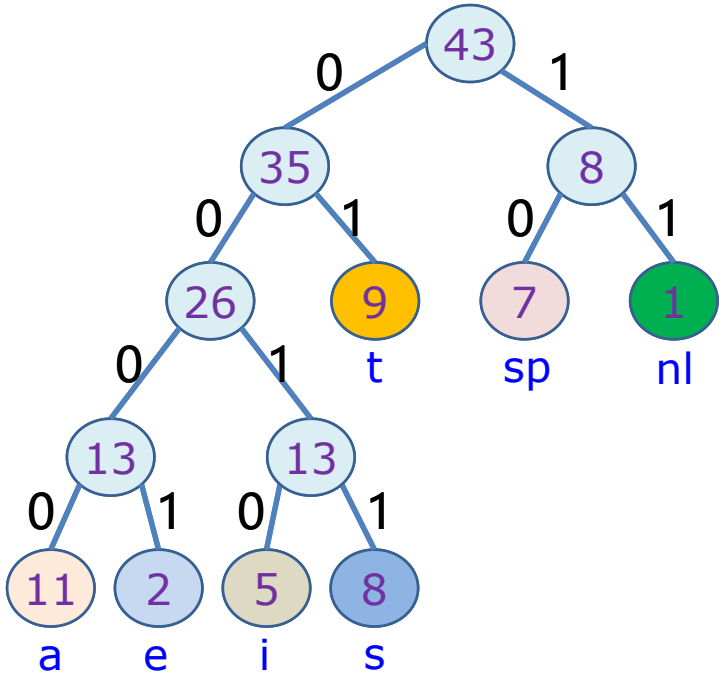
How

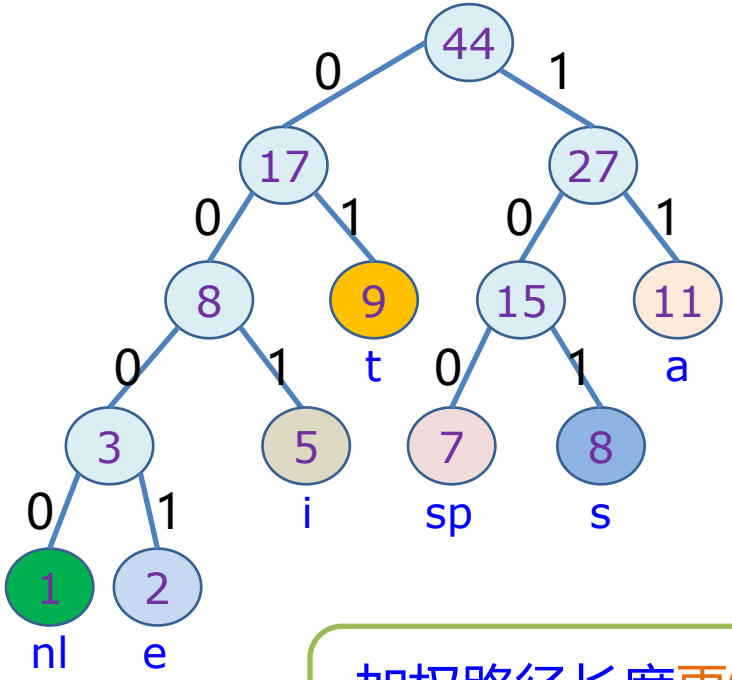
哈夫曼编码

字符使用的频度

字符	权重
a	11
e	2
i	5
s	8
t	9
sp	7
nl	1

使用带权重的二叉树设计前缀编码





加权路径长度更短的二叉树
具有更高的编码效率

$Cost_A = 11\times4+2\times4+5\times4+8\times4+9\times2+7\times2+1\times2 = 138$

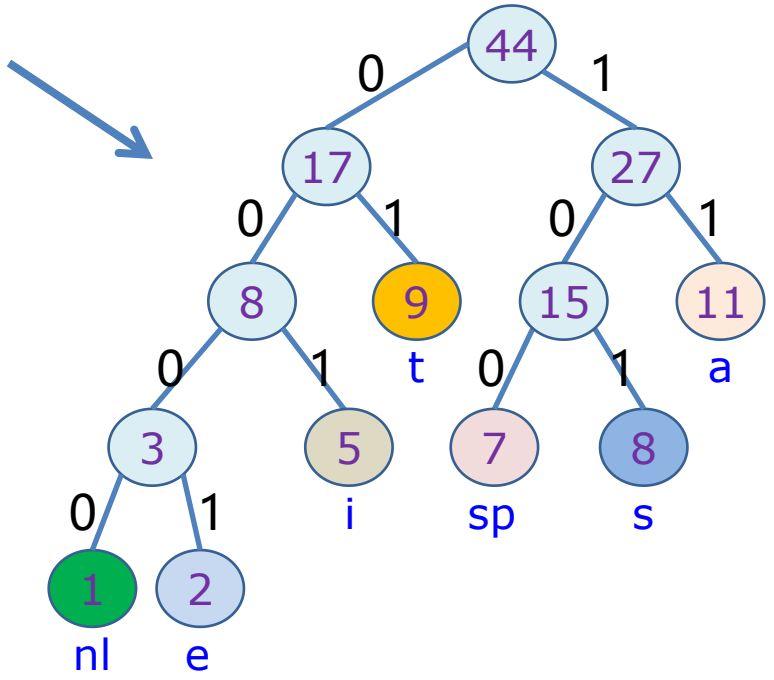
$Cost_B = 1\times4+2\times4+5\times3+9\times2+7\times3+8\times3+11\times2 = 112$

如何构建具有最小加权路径长度的二叉树?

哈夫曼编码

使用哈夫曼算法构建前缀编码二叉树

基于哈夫曼算法构建的哈夫曼树就是具有**最小加权路径长度**的二叉树，称为**最优二叉树**。使用哈夫曼树生成的编码就称为**哈夫曼编码**。



字符	编码
a	11
e	0001
i	001
s	101
t	01
sp	100
nl	0000

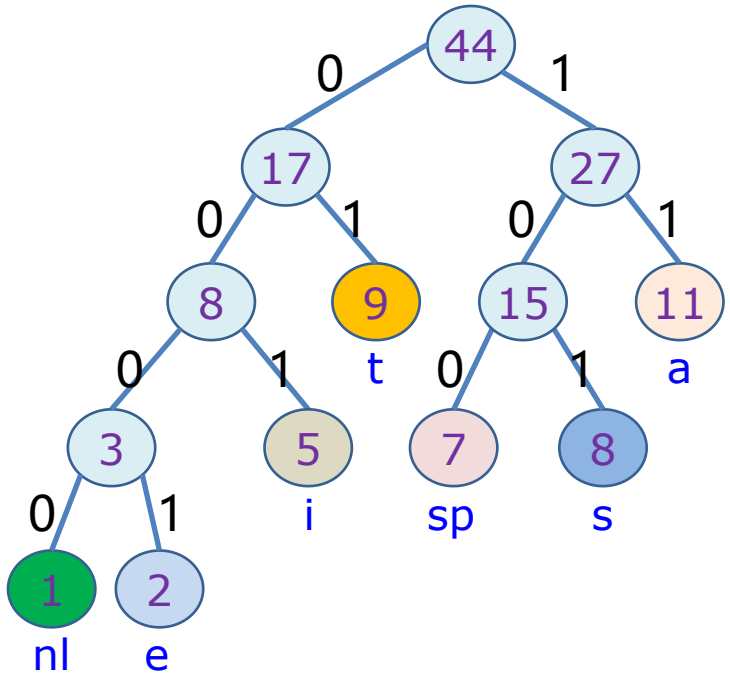
- 哈夫曼树的**特点**：
  - ✓ 权值越大的结点越靠近根；
  - ✓ 使用概率越大的字符编码越短，使用概率越小的字符编码越长。

文本：tea → 编码：01 0001 11 (码长：8)

使用哈夫曼树生成的编码在所有二叉树生成的编码中是最短的一种

哈夫曼编码

哈夫曼编码的解码过程

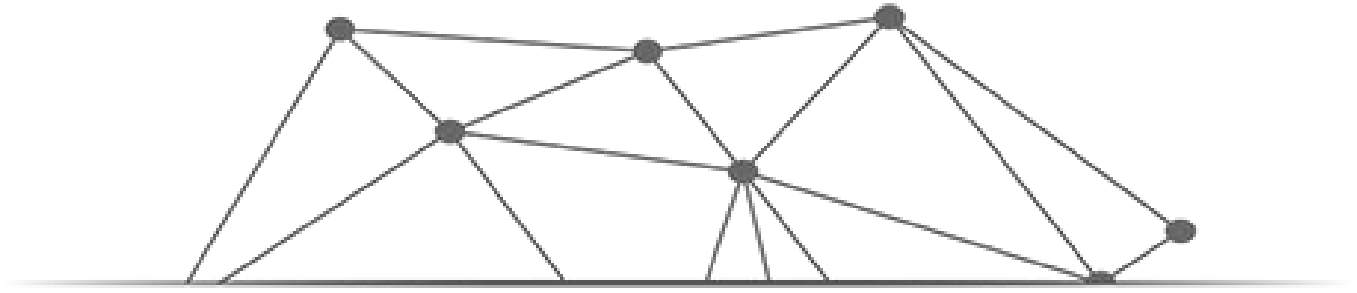


000111011000101

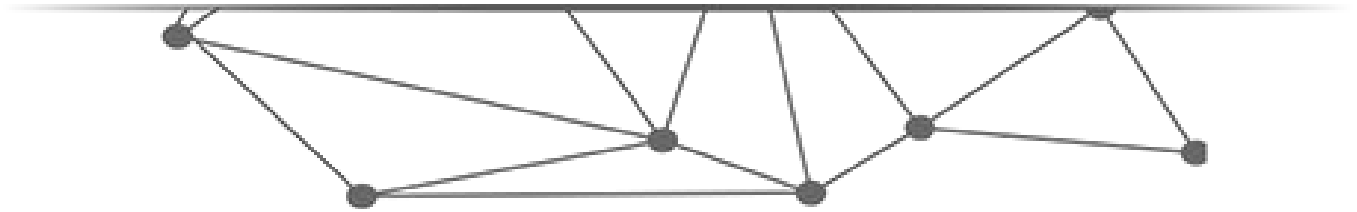
- 哈夫曼编码的解码过程：
  1. 从根节点开始，使用树遍历的方法进行解码；
  2. 遇到"0"则向左遍历，遇到"1"则向右遍历；
  3. 一旦到达叶子结点，则译出一个字符；
  4. 反复执行**步骤1~3**，直至译码完成。

0 0 0 1 1 1 0 1 1 0 0 0 0 1 0 1

e a t i t



课堂互动



第13讲 哈夫曼树及其应用

小　结

- 哈夫曼编码是不等长编码
- 哈夫曼编码是前缀编码，任一字符编码都不是另一字符编码的前缀，不会产生错译
- 哈夫曼编码树中没有度为1的结点。若叶子结点的个数为n，则哈夫曼树的结点总数为2n-1。
- **编码过程：**使用哈夫曼算法构建二叉树（哈夫曼树），然后根据哈夫曼树得到的编码表对字符进行编码。
- **译码过程：**按照左0，右1的原则，从根结点开始依次对二进制码进行遍历。每次到达叶子结点时，完成一个字符的译码。反复执行该过程，直至所有二进制码都完成解译。

欧老师的联系方式

读万卷书 行万里路 只为最好的修炼

QQ：14777591 (宇宙骑士)

Email：ouxinyu@alumni.hust.edu.cn

Website: http://ouxinyu.cn

Tel：18687840023

地址：云南财经大学南院3楼 智能应用研究院A306-2