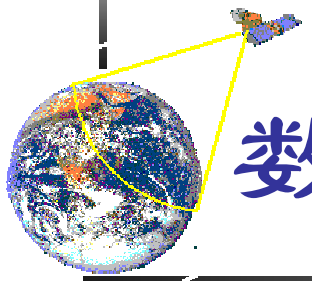


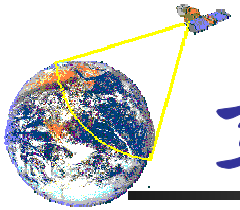
数字图像处理与分析

第六章 图像压缩1

刘定生

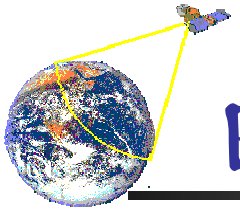
中科院中国遥感卫星地面站

2004年春季学期



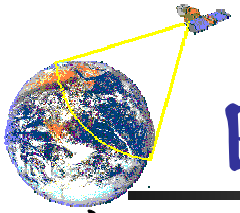
第六章　图像压缩

- 第六章　图像压缩（1）
  - 图像压缩基本概念
  - 图像压缩模型
  - 图像压缩中的信息论观点
  - 图像压缩编码的基本方法
  - 无损压缩
  - 有损压缩
  - 图像压缩标准



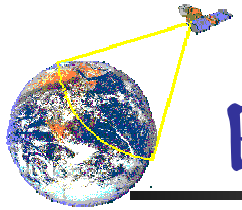
图像压缩基本概念

- 对于给定的信息，如何用最少的数据表示？
  - 信息是什么？
  - 如何表述信息？
  - 信息的特征
- 数据与信息—多种理解
  - 数据是信息的载体，信息是数据中表达的消息
  - 数据是未加工的信息，而信息是数据经过加工以后的能为某个目的使用的数据
  - ……



图像压缩基本概念

- 基础—数据的冗余与相关
  - 数据的冗余
    - ✓大多数信息的表达都存在着一定的冗余度（相关性），通过采用一定的模型和编码方法，可以降低这种冗余度
  - 设：为表达一个信息，方法A用了n1个数据，改进方法B用了n2个数据（压缩数据）：
    - ✓压缩率（压缩比）：$C_R = n_1 / n_2$
    - ✓相对数据冗余：$R_D = 1 - 1/C_R$
- 图像数据中的冗余
  - 三种数据冗余：编码冗余、像素冗余、视觉心理冗余



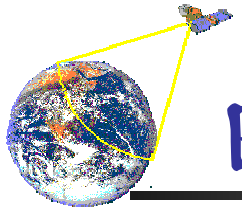
图像压缩基本概念

- 编码冗余:

如果一个图像的灰度级编码，使用了多于实际需要的编码符号，就称该图像包含了编码冗余。

例：二值图像

如果用8bits表示该图像的像素，则该图像存在着编码冗余，因为该图像的像素只有两个灰度，用一位即可表示。

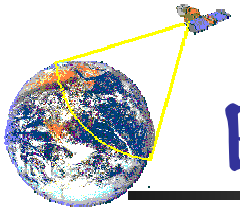


图像压缩基本概念

- 像素冗余（像素间相关）
  - 任何给定位置像素值，原理上均与相邻像素相关，都可以通过它的邻居预测到，单个像素携带的信息相对较少
  - 对于一幅图像，大量单个像素对视觉的贡献是冗余的。这是建立在对邻居值预测的基础上。

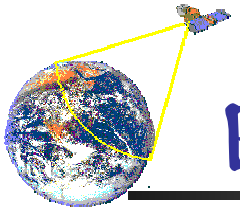
例：原图像数据：**234 223 231 238 235**

压缩后数据：**234 -11 8 7 -3**



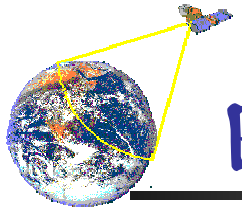
图像压缩基本概念

- 视觉心理冗余
  - 一些信息在一般视觉处理中比其它信息的相对重要程度要小，这种信息就被称为视觉心理冗余
    - ✓人眼对区域亮度的感觉不仅取决于该区域的反射光，而且取决于周围的环境
    - ✓视觉残留现象
    - ✓人眼的亮度适应能力



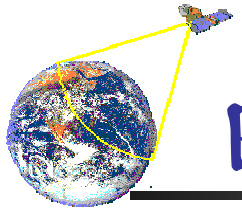
图像压缩基本概念

- 利用图像数据的冗余开展压缩
  - 因为有编码冗余和像素冗余，当我们将图像信息的描述方式改变之后，可以压缩掉这些冗余，进行“无损”压缩
  - 因为有主观视觉冗余，当我们忽略一些视觉不太明显的微小差异，可以进行所谓的“有损”压缩。



图像压缩基本概念

- 保真度准则——评价压缩算法的标准
  - 客观保真度准则—定量描述
  - 主观保真度准则—定性或定性基础上的定量描述



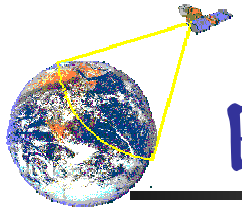
图像压缩基本概念

- 客观保真度准则

如果信息损失的程度，可以表示为原始或输入图像与压缩后又解压缩输出的图像的函数，这个函数就被称为客观保真度准则。一般表示为：

$$e(x,y) = \hat{f}(x,y) - f(x,y)$$

式中：**f(x,y)**是输入图像，$\hat{f}(x,y)$ 是压缩后解压缩的图像，**e(x,y)**是误差函数



图像压缩基本概念

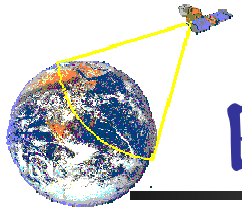
- 客观保真度准则（续）

由此可得两个图像之间的均方根误差:

$$e_{mms} = \sqrt{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y) - f(x,y)]^2}$$

均方信噪比:

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y) - f(x,y)]^2}$$



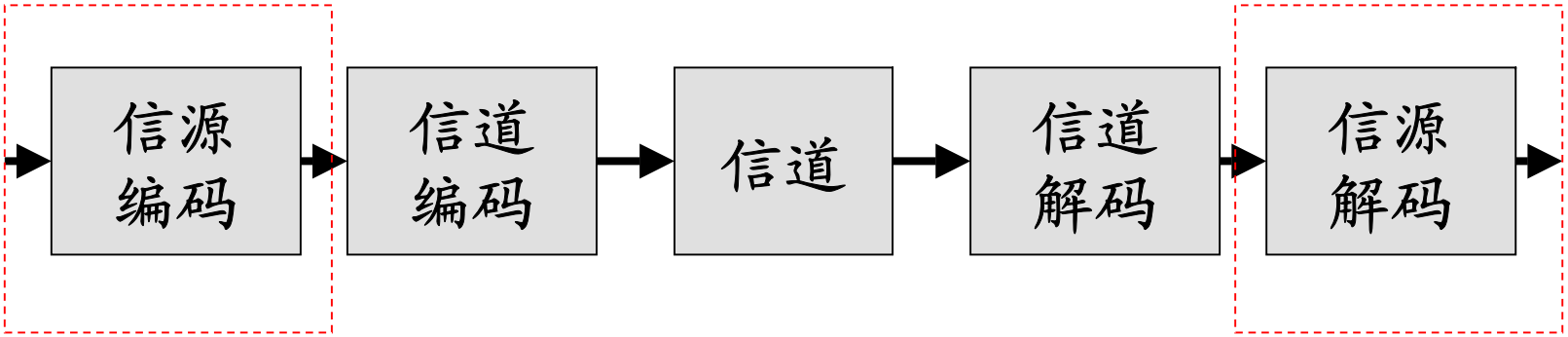
图像压缩基本概念

- 主观保真度准则

  通过视觉比较两个图像，给出一个定性的评价，如很粗、粗、稍粗、相同、稍好、较好、很好，这种评价被称为主观保真度准则。

图像压缩模型

- 图像压缩系统的一般构成

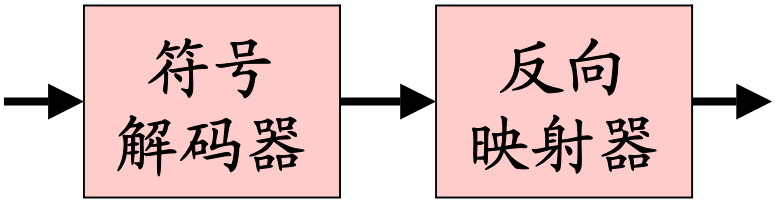


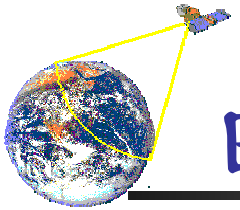
- 信源编码：完成原始数据的编码与压缩。
- 信道编码：为了抗干扰，增加一些容错、校验位，实际上是有规律地增加传输数据的冗余，以便于消除传输过程中加入的随机信号。
- 信道：传送数据（信息）的手段；如**Internet**、广播、通讯、可移 动介质

图像压缩模型

- 信源编码与解码模型
  - 信源编码模型

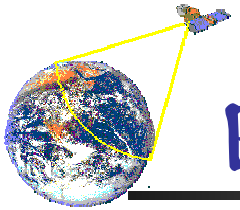
  - 信源解码模型





图像压缩模型

- 信源编码与解码模型（续）
  - 映射器：减少像素冗余，如使用字典编码。或进行图像变换。
  - 量化器：减少视觉心理冗余，仅用于有损压缩。
  - 符号编码器：减少编码冗余，如使用哈夫曼编码

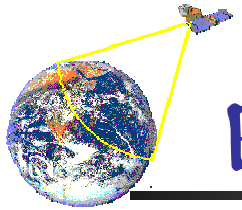


图像压缩中的信息论观点

- 图像压缩中的问题
  - 显示一幅图像究竟需要多大的数据量?
  - 是否存在可充分描述一幅图像的最小数据量?
- 信息论—回答这类问题的理论框架
  - 信息论中的信源编码理论解决的主要问题:

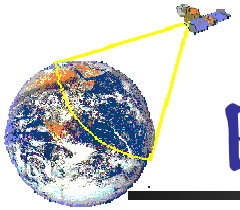
    （1）数据压缩的理论极限

    （2）数据压缩的基本途径



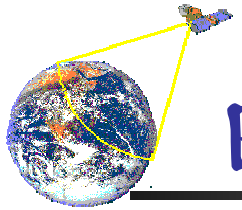
图像压缩中的信息论观点

- 信息论的发展
  - 狭义信息论
    - ✓关于通讯技术的理论，以数学方法研究通讯技术中关于信息的传输和变换规律的一门科学
  - 广义信息论
    - ✓以各种系统、各门科学中的信息为对象，广泛地研究信息的本质和特点，以及信息的获取、计量、传输、储存、处理、控制和利用的一般规律



图像压缩中的信息论观点

- 香农（C. E. Shannon）—狭义信息论的创立
  - 1948年，香农发表文章——通讯的数学原理（A mathematical Theory of Communication），奠定了信息基本理论的基础
  - 文章中，严格定义了信息的单位—“熵”的概念
  - 在此基础上，定义了信道容量的概念，并给出在不同噪音情况下无失真通信的极限传输速率
  - 信息熵—借用热力学中名词“熵”（ Entropy ）来表示一条信息中真正需要编码的信息量
  - 信息熵、热力学熵和复杂程度是互相成正比例的物理量
    - ✓ 微观状态的复杂程度就是热力学熵
    - ✓ 通讯讯号的复杂程度就是信息熵



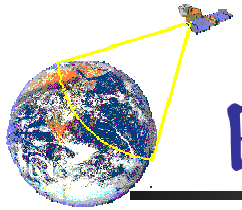
图像压缩中的信息论观点

- 信息的测量
  - 信息的产生可以被模拟为一个概率过程

    假设某随机事件E 出现的概率为$P(E)$，则该随机事件包含的信息为

$$I(E)=\log\frac{1}{P(E)}=-\log P(E)$$

  - I (E) 通常称之为E的自信息
    - 如果P (E) =1，则I (E) =0；没有信息
  - 对数底数决定来衡量信息的单位
    - 底数为m，则称之为m 元单位
    - 底数为2，则信息的单元为比特

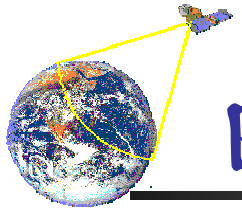


图像压缩中的信息论观点

- 信源的模型化

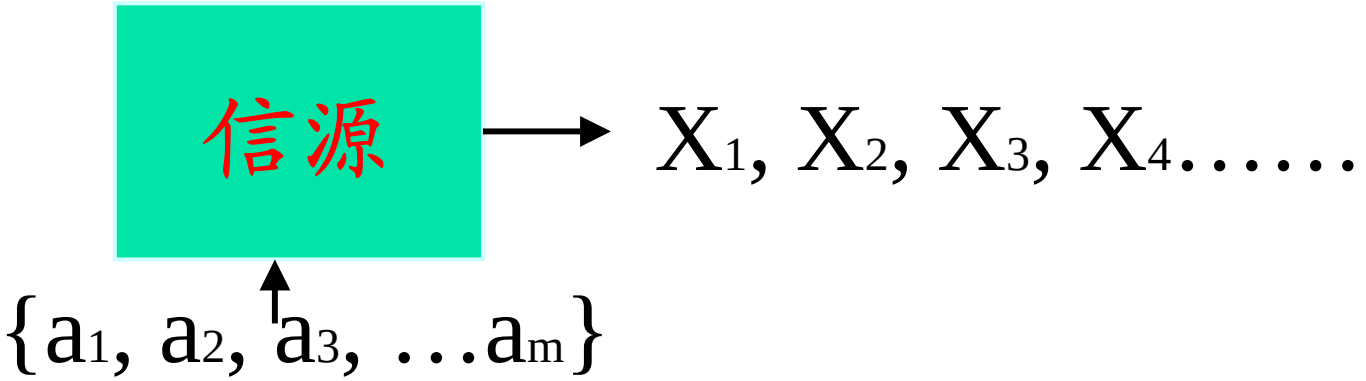
信源 → $X_1, X_2, X_3, X_4……$

- 信源被抽象为一个随机变量序列（随机过程）
- 如果信源输出的随机变量取值于某一连续区间，就叫做连续信源。比如语音信号**X(t)**
- 如果信源输出的随机变量取值于某一离散符号集合，就叫做离散信源。比如平面图像**X(x，y)**

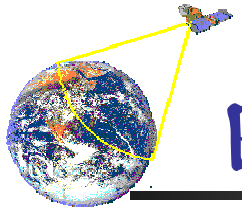


图像压缩中的信息论观点

- 信源的模型化—离散信源



  - 如果随机序列中各个变量具有相同的概率分布，则称为离散平稳信源。
  - 如果离散平稳信源的输出序列中各个变量是相互独立的，即前一个符号的出现不影响以后任何一个符号出现的概率，则称为离散无记忆平稳信源，否则称为离散有记忆平稳信源



图像压缩中的信息论观点

- 信源的模型化—离散信源
  - 假定信源输出的符号系列为 $\{a_1, a_2, \ldots\ldots a_j\}$，信源产生符号$a_i$的概率为$\mathbf{P}(a_i)$，并有：

$$\sum_{i=1}^{j} P(a_i) = 1$$

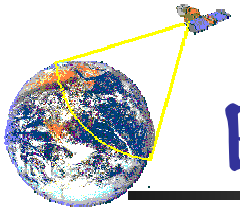
则该信源所有输出符号的集合

$$\mathbf{A} = \{a_j\}$$

及其所有输出符号的概率的集合向量

$$z = [P(a_1), P(a_2), \ldots, P(a_j)]^T$$

共同构成了信源的有限总体集合（**A,z**），该有限总体集合完全描述了信源



图像压缩中的信息论观点

- 信源的模型化（续）
  - 对于符号a_i，其概率为$P(a_i)$，相应的自信息为：

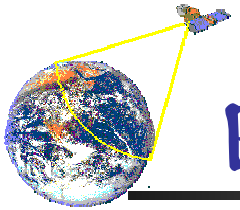
$$I(a_i) = -\log(P(a_i))$$

则该信源输出的平均信息，表示为：

$$H(z) = -\sum_{i=1}^{J} P(a_i)\log(P(a_i))$$

H(z)称之为信源的不定度或一阶熵

  - 在符号出现之前，熵表示符号集中的符号出现的平均不确定性；在符号出现之后，熵代表接收一个符号所获得的平均信息量



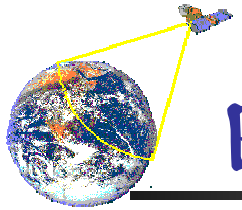
图像压缩中的信息论观点

- 信源的模型化（续）
  - 信源的熵反映了信源输出平均信息量的大小，符号概率分布越不均匀，信源的熵越小。符号概率平坦分布，即当所有符号等概率分布时，则熵最大
- 最大离散熵定理
  - 当与信源对应的字符集中的各个字符为等概率分布时，熵具有极大值**$\log_2 m$**。**m**为字符集中字符个数

$$p_j = \frac{1}{m}$$

$$H(x) = -\sum_{j=1}^{m} p_j * \log\ p_j$$

$$= -\sum_{j=1}^{m} \frac{1}{m} * \log\ \frac{1}{m} = \log\ m$$

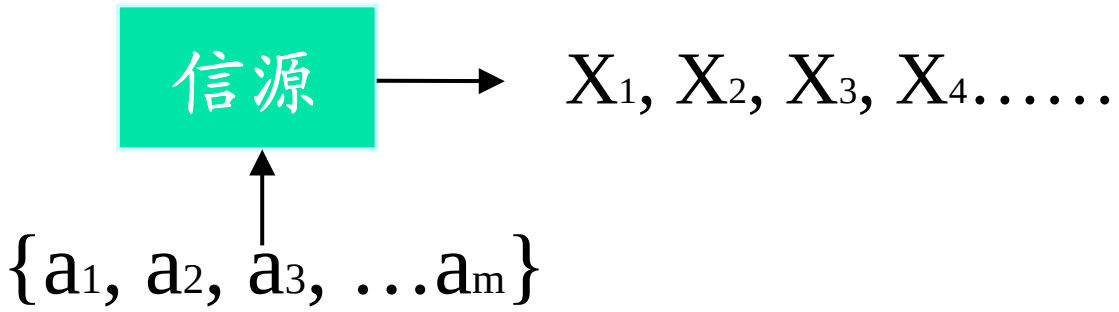


图像压缩中的信息论观点

- 最大离散熵定理的应用
  - 对于同一个信源其总的信息量是不变的，如果能够通过某种变换（编码），使信源尽量等概率分布，则每个输出符号所独立携带的信息量增大，那么传送相同信息量所需要的序列长度就越短。
  - 离散无记忆信源的冗余度隐含在信源符号的非等概率 分布之中。只要**H（X）小于$\log_2 m$**，就存在数据压缩的可能

图像压缩中的信息论观点

- 离散无记忆信源的编码



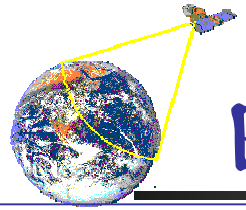
  - 如果对字符a_j的编码长度为L_j，则X的平均码长为：

$$\overline{L} = \sum_{j=1}^{m} p_j * L_j$$

  - 可有

$$L = \sum_{j=1}^{m} p_j * L_j \geq H(X) = -\sum_{j=1}^{m} p_j * \log_2 p_j$$

在$L_j = -\log_2 p_j$时，平均码长取得极小值H(X)



图像压缩中的信息论观点

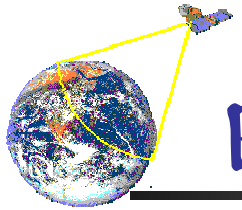
- 离散无记忆信源的编码——香农第一定理：

  对信源输出 $\mathbf{A} = \{a_j\}$，其信息熵为

$$H(z) = -\sum_{i=1}^{J} P(a_i)\log(P(a_i))$$

  当信源输出符号具有统计上独立的特性时，这类信源编码所需的平均码字长度下限为其信息熵。

- 关于离散无记忆平稳信源的结论
  - 一阶熵即为离散无记忆平稳信源的压缩极限（基本极限）
  - 只要信源不是等概率分布，就存在着数据压缩的可能性。
  - 数据压缩的基本途径之一：使各字符的编码长度尽量等于字符的信息量



图像压缩中的信息论观点

- 离散无记忆信源编码（续）
  - 实例

某信源输出符号串**aabbaccbaa**，字符串长度为 **10**，字符 **a b c** 分别出现了 **5 3 2** 次，则 **a b c** 在信息中出现的概率分别为 **0.5**、**0.3**、**0.2**，他们的熵分别为：

$$Ea = -\log_2(0.5) = 1$$

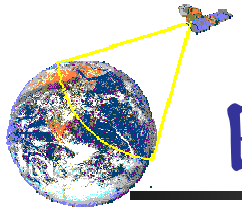
$$Eb = -\log_2(0.3) = 1.737$$

$$Ec = -\log_2(0.2) = 2.322$$

该信源的熵为：

H=0.5 Ea+0.3Eb+0.2Ec=1.4855 bit/symbol

该信源编码所需总码长为**N*H=14.855** 比特



图像压缩中的信息论观点

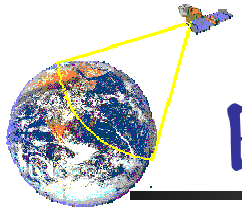
- 离散有记忆信源编码
  - 信息量的表示—条件熵

$$\lim_{n\to\infty} H(X_n \mid X_1, X_2, \ldots, X_{n-1})$$

  - 可以证明

$$H(X \mid Y) \le H(X)$$

  - 联合熵与其可能达到的最大值之间的差值反映了该有记忆信源所含的冗余度，这种冗余是由于随机变量序列之间的相关性造成的

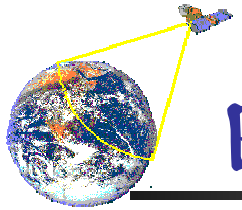


图像压缩中的信息论观点

- 关于离散有记忆平稳信源的结论
  - 离散有记忆平稳信源的压缩极限为：

$$\lim_{n\to\infty} H(X_n \mid X_1, X_2, \ldots, X_{n-1})$$

  - 压缩的基本途径之二：尽量去除各分量之间的相关性，再对各分量进行独立编码。
  - 压缩的基本途径之三：可利用条件概率进行编码，阶越高越有利。
  - 压缩的基本途径之四：可将多个分量合并成向量，利用其联合概率进行编码，联合的分量越多越有利



图像压缩中的信息论观点

- 信息系统模型
  - 将前述信源输出通过信道传输到接收端，信息接收者（信宿）接收到符号集合**B=$\{b_k\}$**，其概率**P(b_k)**与信源输出**z**的关系为

$$P(b_k)=\sum_{i=1}^{J}P(b_k\mid a_i)P(a_i)$$

  - 信宿收到的可能的符号集合概率向量:

$$v=[P(b_1),P(b_2),\ldots,P(b_k)]^T$$

  - 则有限集合（**B,v**）完整地描述了用户收到的信息

图像压缩中的信息论观点

- 信息系统模型（续）
  - $\mathbf{P(b_k \mid a_i)}$反映了从信源到信宿之间的信息信息传递关系，考虑所有符号的传递，可构成$\boldsymbol{K}x\boldsymbol{J}$传递矩阵（信道矩阵）

$$Q = \begin{bmatrix} P(b_1 \mid a_1) & P(b_1 \mid a_2) & \cdots & P(b_1 \mid a_J) \\ P(b_2 \mid a_1) & \cdots & \cdots & P(b_2 \mid a_J) \\ \cdots & \cdots & \cdots & \cdots \\ P(b_K \mid a_1) & P(b_K \mid a_2) & \cdots & P(b_K \mid a_J) \end{bmatrix}$$

  - 形成完整的信息系统模型

总体集合 (A,z)

$A = \{a_j\}$

$z = [P(a_1, a_2, \ldots, a_j]^T$

$Q = [q_{kj}]$

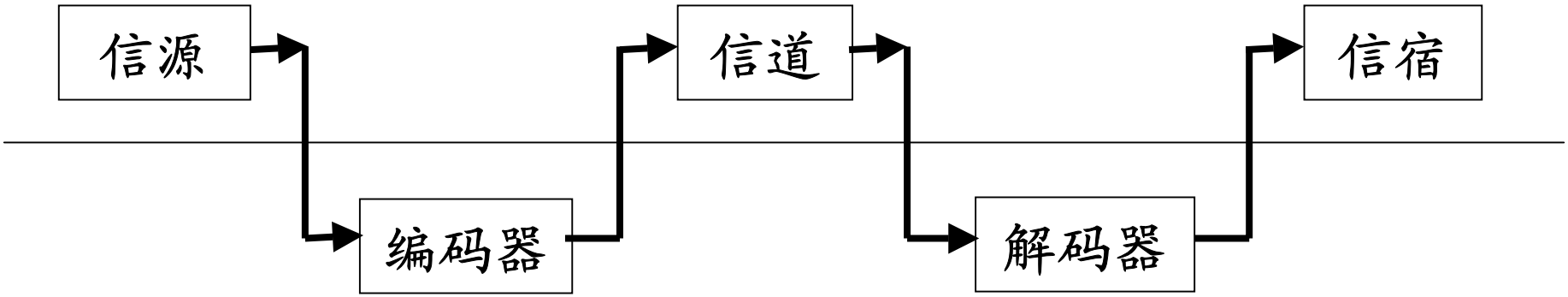
总体集合 (B,z)

$B = \{b_j\}$

$v = [P(b_1, b_2, \ldots, b_j]^T$

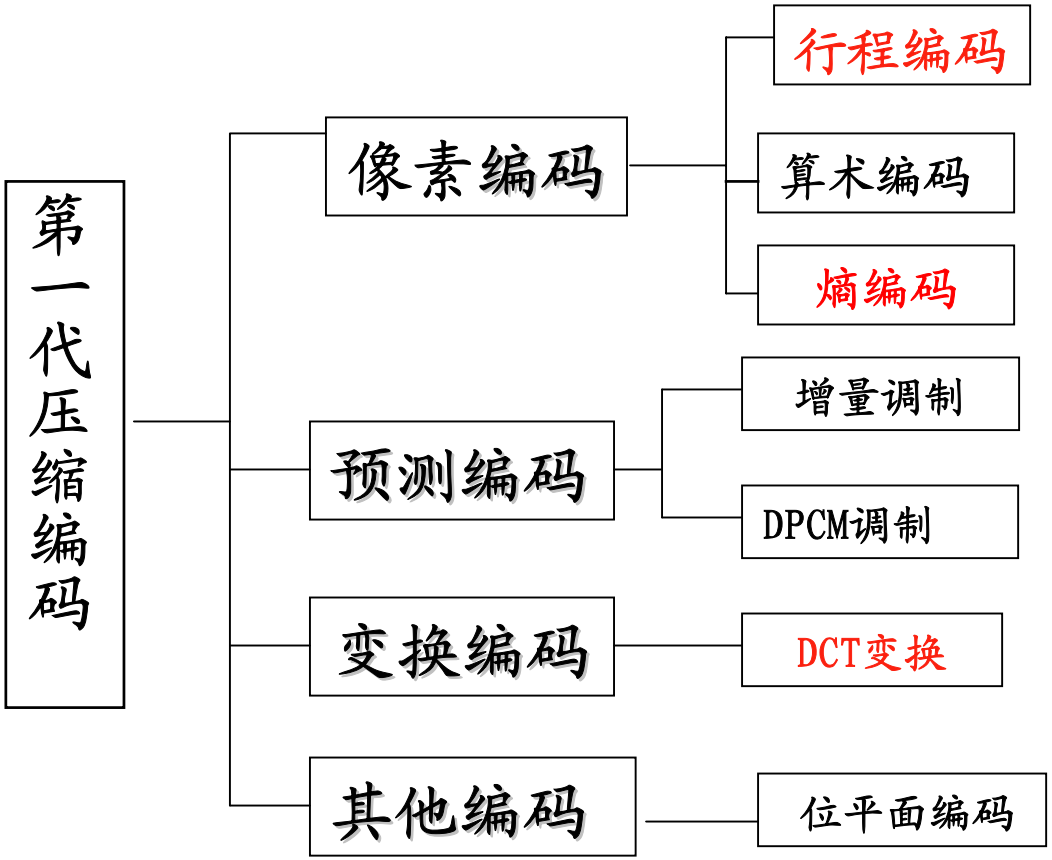
图像压缩中的信息论观点

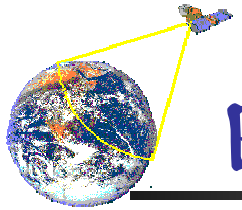
- 通信系统模型
  - 为最大化地通过信道传输信息，在具有自然性质的信源、信道与信宿之中，增加人为的编码器与解码器，形成基本通信系统模型



图像压缩编码的基本方法

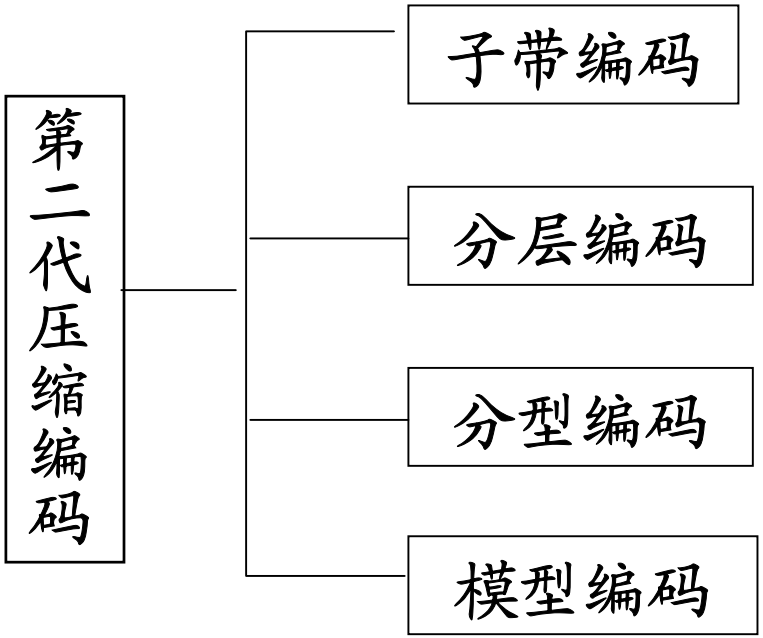
- 第一代压缩编码
  - 八十年代以前，主要是根据传统理论进行的信源编码方法





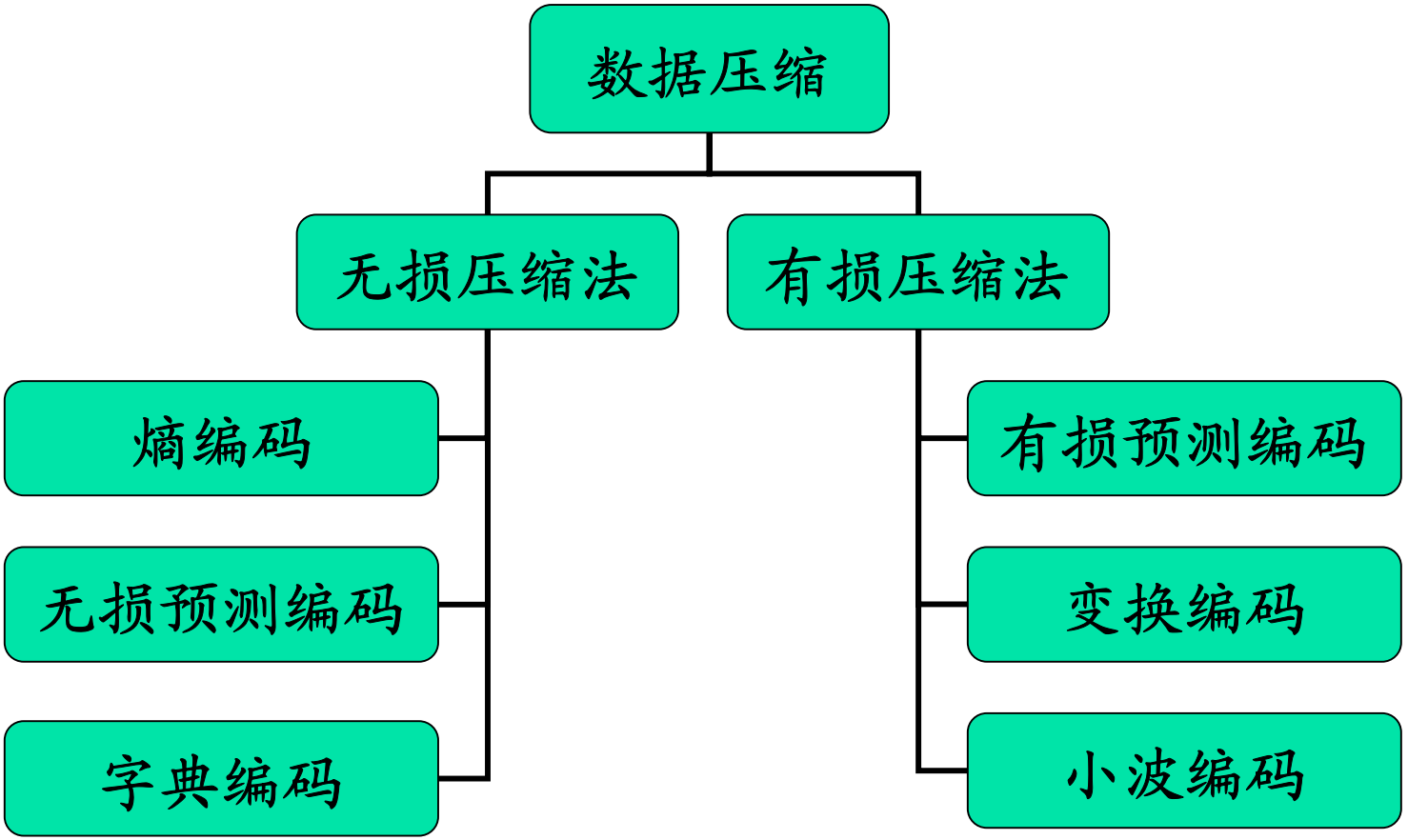
图像压缩编码的基本方法

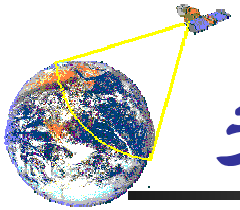
- 第二代压缩编码
  - 八十年代以后，突破信源编码理论，结合分形、模型基、神经网络、小波变换等数学工具，充分利用视觉系统生理心理特性和图像信源的各种特性。



图像压缩编码的基本方法

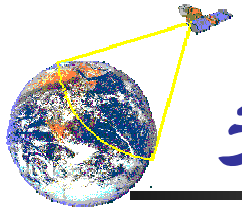
- 从信息损失角度





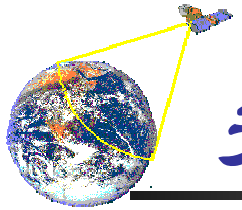
无损压缩——熵编码

- 熵编码
  - 利用香农理论，从统计的角度，设法找到一种编码使得平均码长达到熵极限
  - 基本思想：对出现概率较大的符号取较短的码长，而对出现概率较小的符号取较大的码长
  - 主要方法
    - ✓香农－范诺编码
    - ✓霍夫曼编码
    - ✓算术编码



无损压缩——熵编码

- 熵编码—霍夫曼编码
  - 具体步骤:
    - （1）初始化
    - （2）合并概率最小的两个事件
    - （3）排序
    - （4）如果事件个数大于2则重复（2）和（3）
    - （5）赋值
    - （6）编码



无损压缩——熵编码

- 熵编码—霍夫曼编码实例（续）

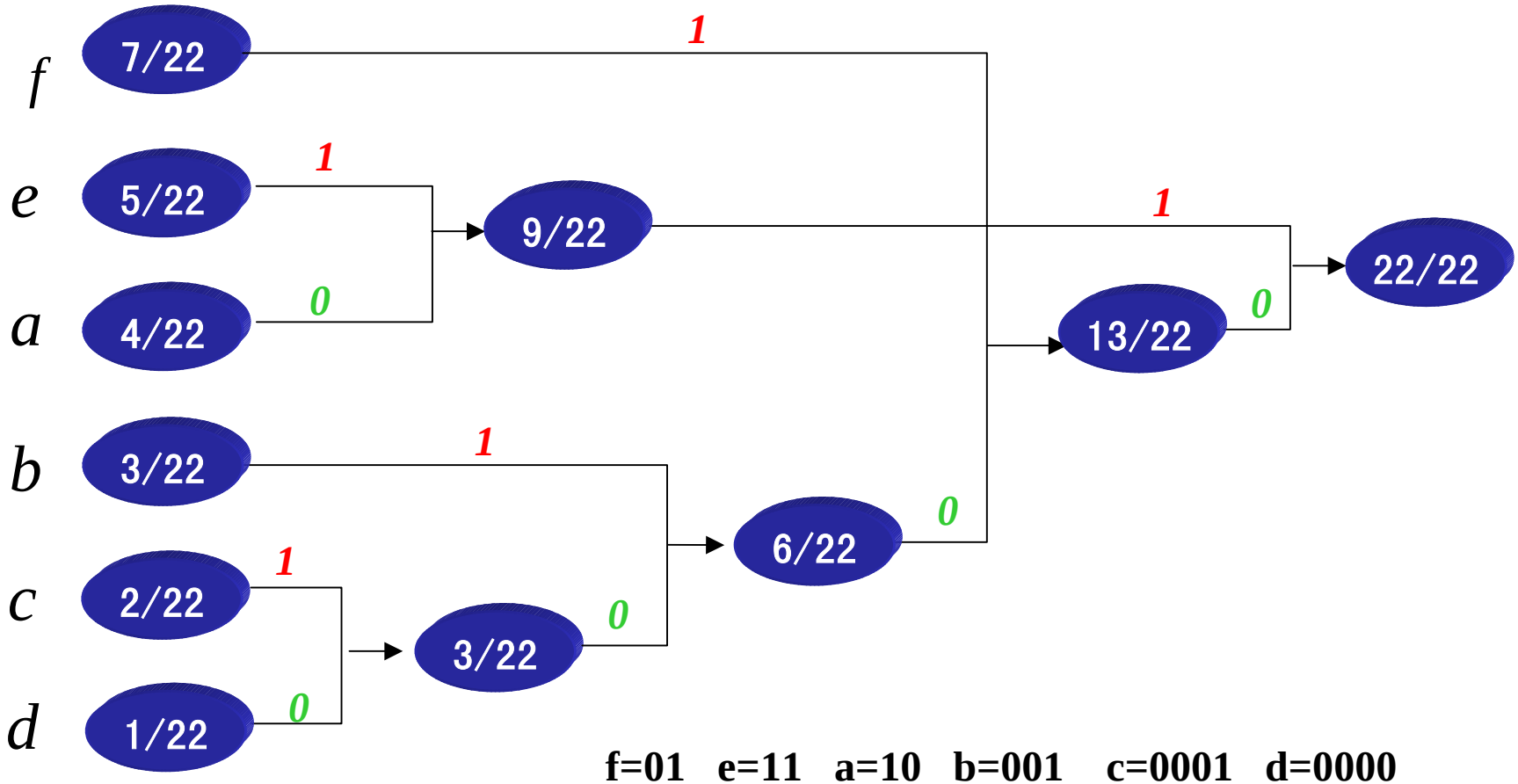
符号	S1	S2	S3	S4
出现概率	1/2	1/4	1/8	1/8
等长编码	00	01	10	11
霍夫曼	0	10	110	111

H(X) = 1.75　　L1=2　　L2=1.75

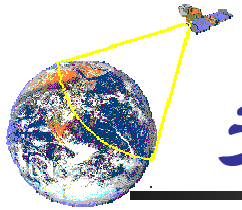
源	S1	S2	S1	S3	S2	S1	S1	S4
等	00	01	00	10	01	00	00	11
霍	0	10	0	110	10	0	0	111

无损压缩——熵编码

- 熵编码—霍夫曼编码实例（续）



f=01 e=11 a=10 b=001 c=0001 d=0000



无损压缩——熵编码

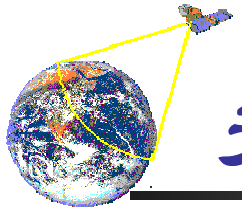
- 熵编码—霍夫曼编码（续）
  - Huffman编码在图像压缩中的实现

对一幅图像进行编码时，如果图像尺寸大于256时，这幅图像的不同码字就有可能是很大，例如极限为256个不同的码字。

对整幅图直接进行Huffman编码时，小分布的灰度值，就有可能具有很长的编码。

如：100位以上，这样不但达不到压缩的效果反而会使数据量加大，如何处理？

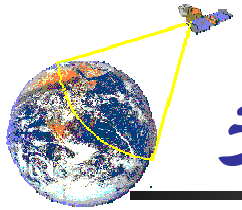
将图像分割成若干的小块，对每块进行独立的Huffman编码。例如：分成8×8的子块，就可以大大降低不同灰度值的个数（最多是64而不是256）



无损压缩——熵编码

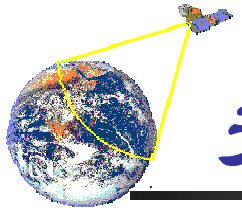
- 熵编码—霍夫曼编码（续）
  - Huffman编码在图像压缩中的实现

将图像分割成若干的小块，对每块进行独立的**Huffman**编码。例如：分成　的子块，就可以大大降低不同灰度值的个数（最多是**64**而不是**256**）



无损压缩——熵编码

- 熵编码—霍夫曼编码的特性（续）
  - 静态编码
    - ✓ 在压缩之前就建立好一个概率统计表和编码树。算法速度快，但压缩效果不是最好
  - 动态编码
    - ✓ 对每一个图像，临时建立概率统计表和编码树。算法速度慢，但压缩效果较好
  - 霍夫曼编码的局限性
    - ✓ 利用霍夫曼编码，每个符号的编码长度只能为整数，所以如果源符号集的概率分布不是2负n次方的形式，则无法达到熵极限
    - ✓ 输入符号数受限于可实现的码表尺寸
    - ✓ 译码复杂
    - ✓ 需要实现知道输入符号集的概率分布
    - ✓ 没有错误保护功能



无损压缩——熵编码

- 熵编码—香农－费诺编码
  - 香农－费诺编码与**Huffman**编码相反，采用从上到下的方法
  - 具体步骤:

（1）首先将编码字符集中的字符按照出现频度和概率进行排序。

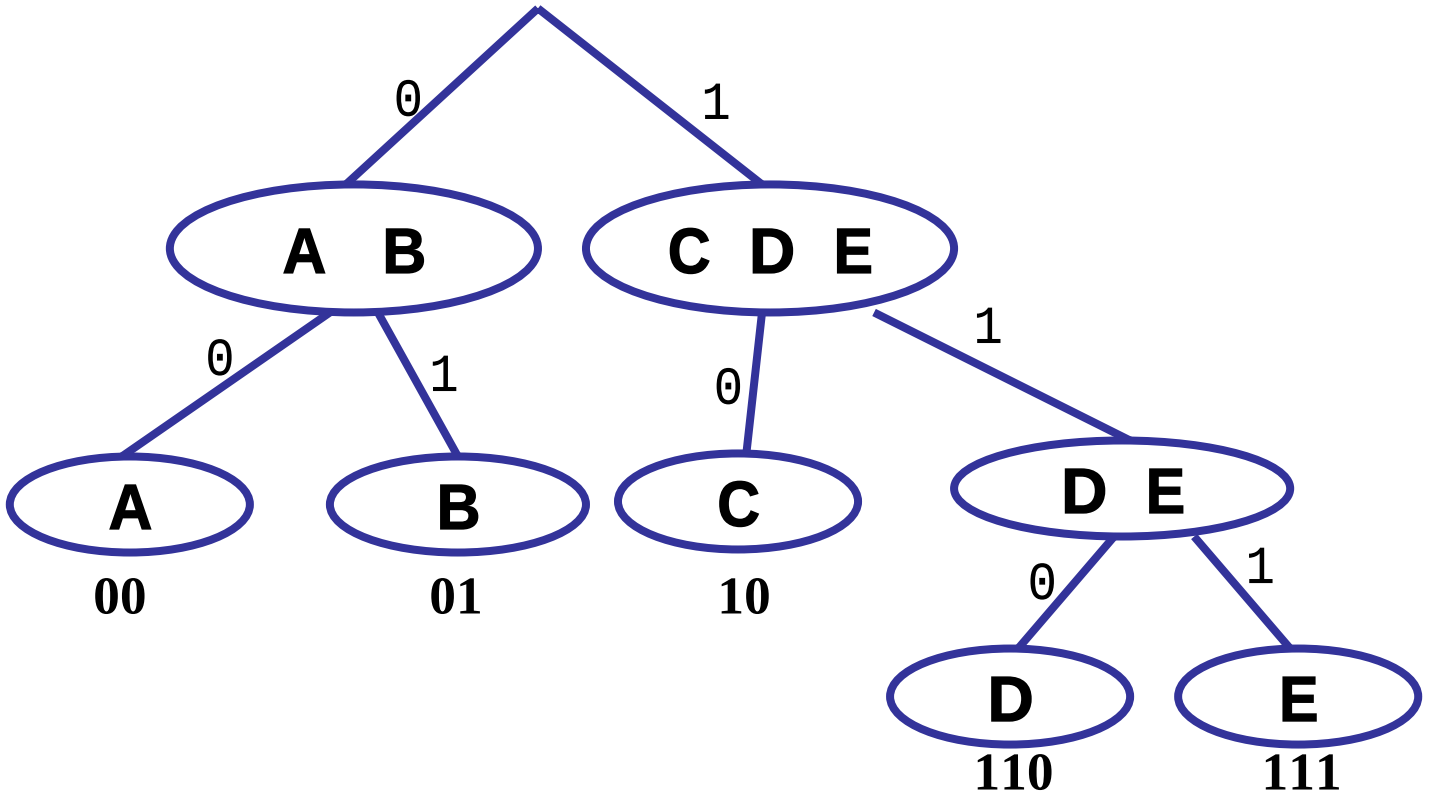
（2）用递归的方法分成两部分，使两个部分的概率和接近于相等。直至不可再分，即每一个叶子对应一个字符。

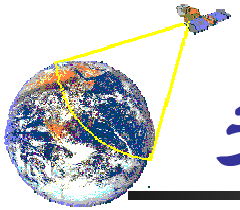
（3）编码。

无损压缩——熵编码

- 熵编码—香农－费诺编码例

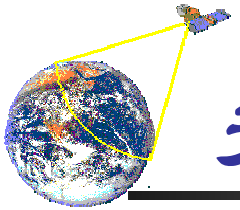
符号	A	B	C	D	E
次数	15	7	7	6	5





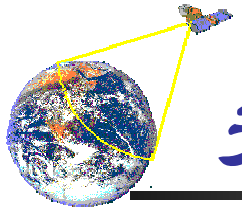
无损压缩——熵编码

- 熵编码—算术编码
  - **Huffman** 编码的局限性:
    - ✓**Huffman** 编码使用整数个二进制位对符号进行编码，这种方法在许多情况下无法得到最优的压缩效果
    - ✓假设某个字符的出现概率为 **80%**，该字符事实上只需要 $-\log_2(0.8) = 0.322$ 位编码，但 **Huffman** 编码一定会为其分配一位 **0** 或一位 **1** 的编码
    - ✓整个信息的 **80%** 在压缩后都几乎相当于理想长度的 **3** 倍左右



无损压缩——熵编码

- 熵编码—算术编码（续）
  - 基本思想：
    - ✓算术编码不是将单个信源符号映射成一个码字，而是把整个信源表示为实数线上0到1之间的一个区间，其长度等于该序列的概率
    - ✓再在该区间内选择一个代表性的小数，转化为二进制作为实际的编码输出
    - ✓消息序列中每个元素都要用来缩短这个区间
    - ✓消息序列中元素越多，所得到的区间就越小，当区间变小时，就需要更多的数位来表示这个区间
  - 采用算术编码每个符号的平均编码长度可以为小数



无损压缩——熵编码

- 熵编码—算术编码（续）
  - 例:

假定信源符号**00 01 10 11**出现的概率分别为

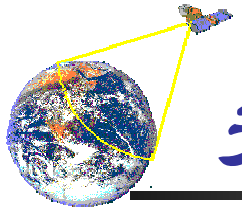
{ 0.1, 0.4, 0.2, 0.3 }

根据这些概率可把间隔**[0, 1)**分成**4**个子间隔:

[0, 0.1), [0.1, 0.5), [0.5, 0.7), [0.7, 1)

综合上述信息如下表所示

符号	00	01	10	11
概率	0.1	0.4	0.2	0.3
初始区间	[0, 0.1)	[0.1, 0.5)	[0.5, 0.7)	[0.7, 1)



无损压缩——熵编码

熵编码—算术编码（续）

- 例

符号	00	01	10	11
概率	0.1	0.4	0.2	0.3
初始区间	[0, 0.1)	[0.1, 0.5)	[0.5, 0.7)	[0.7, 1)

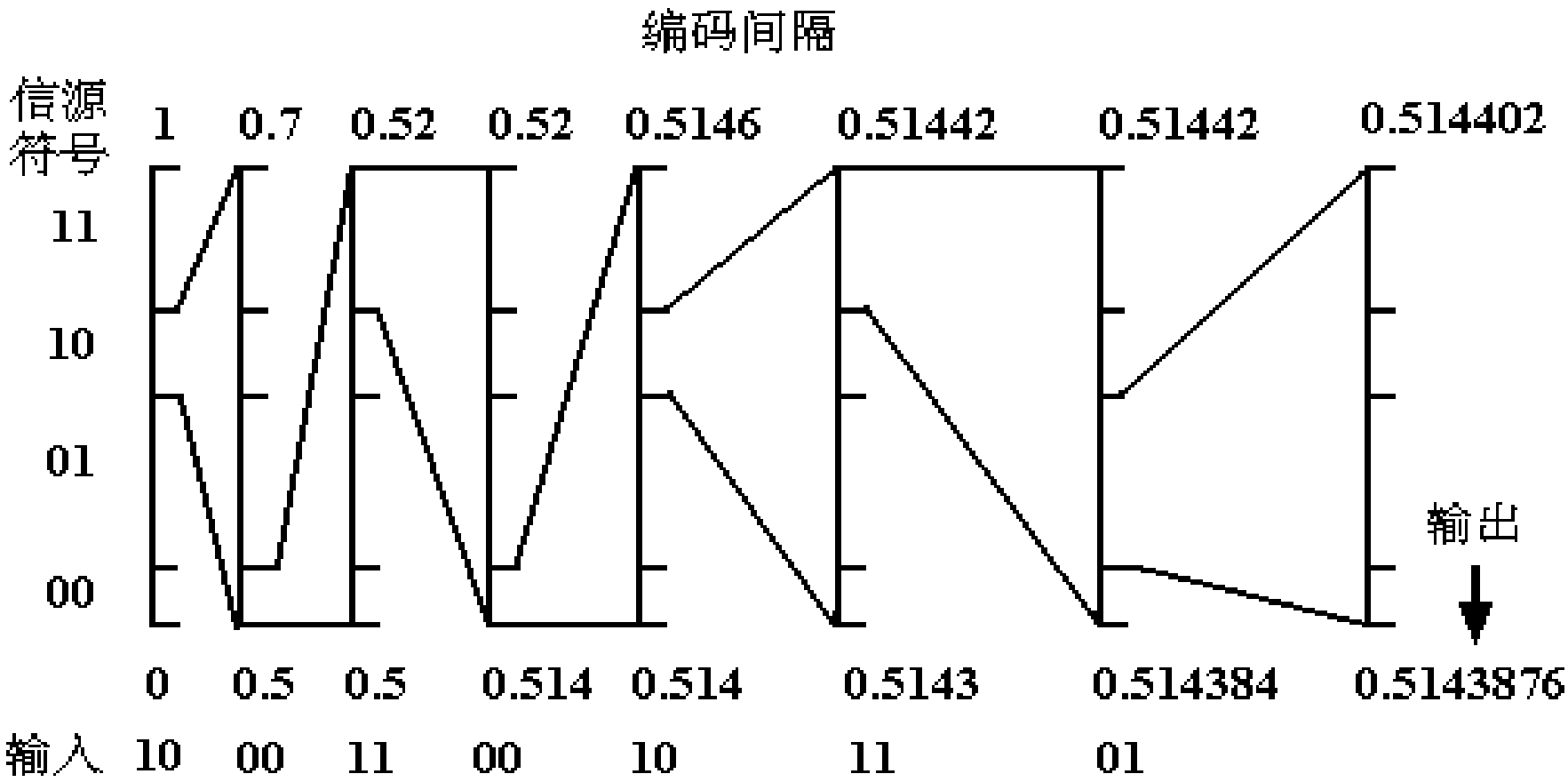
若某二进制输入消息序列为：

10 00 11 00 10 11 01

- ✓ 编码时首先输入的符号是10，找到它的编码范围是[0.5, 0.7)
- ✓ 第二个符号00的编码范围是[0, 0.1)，因此它的间隔就取[0.5, 0.7)的第一个十分之一作为新间隔[0.5, 0.52)
- ✓ 依此类推，编码第3个符号11时取新间隔为[0.514, 0.52)，编码第4个符号00时，取新间隔为[0.514, 0.5146)，… 。消息的编码输出可以是最后一个间隔中的任意数

无损压缩——熵编码

- 熵编码—算术编码（续）

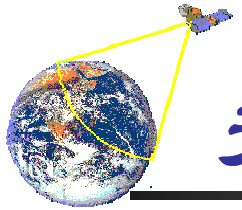


编码过程

步骤	输入符号	编码间隔	编码判决
1	10	[0.5, 0.7)	符号的间隔范围[0.5, 0.7)
2	00	[0.5, 0.52)	[0.5, 0.7)间隔的第一个1/10
3	11	[0.514, 0.52)	[0.5, 0.52)间隔的最后一个1/10
4	00	[0.514, 0.5146)	[0.514, 0.52)间隔的第一个1/10
5	10	[0.5143, 0.51442)	[0.514, 0.5146)间隔的第五个1/10开始，二个1/10
6	11	[0.514384, 0.51442)	[0.5143, 0.51442)间隔的最后3个1/10
7	01	[0.5143836, 0.514402)	[0.514384, 0.51442)间隔的4个1/10，从第1个1/10开始
8	从[0.5143876, 0.514402中选择一个数作为输出：0.5143876		

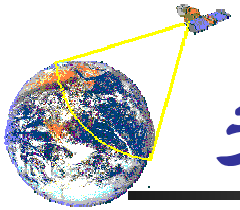
译码过程

步骤	间隔	译码符号	译码判决
1	[0.5, 0.7)	10	0.51439在间隔 [0.5, 0.7)
2	[0.5, 0.52)	00	0.51439在间隔 [0.5, 0.7)的第1个1/10
3	[0.514, 0.52)	11	0.51439在间隔[0.5, 0.52)的第7个1/10
4	[0.514, 0.5146)	00	0.51439在间隔[0.514, 0.52)的第1个1/10
5	[0.5143, 0.51442)	10	0.51439在间隔[0.514, 0.5146)的第5个1/10
6	[0.514384, 0.51442)	11	0.51439在间隔[0.5143, 0.51442)的第7个1/10
7	[0.51439, 0.5143948)	01	0.51439在间隔[0.51439, 0.5143948)的第1个1/10
7	译码的消息：10 00 11 00 10 11 01		



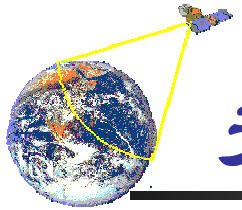
无损压缩——熵编码

- 熵编码—算术编码（续）
  - 算术编码特点
    - ✓算术编码器对整个消息只产生一个码字，这个码字是在间隔[0，1)中的一个实数，因此译码器在接受到表示这个实数的所有位之前不能进行译码
    - ✓由于实际的计算机的精度不可能无限长，运算中有可能出现溢出
    - ✓算术编码也是一种对错误很敏感的编码方法，如果有一位发生错误就会导致整个消息译错



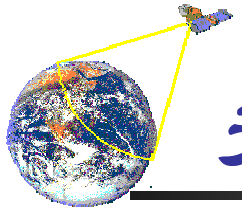
无损压缩——熵编码

- 熵编码—算术编码（续）
  - 自适应算术编码
    - ✓在静态算术编码中，信源符号概率是固定的，一般难于事先知道精确的信源概率
    - ✓在编码过程中，根据编码时符号出现的频繁程度对信源符号的概率动态地进行修改——自适应算术编码
    - ✓在编码期间估算信源符号概率的过程叫做建模
    - ✓我们不能期待一个算术编码器获得最大的效率，所能做的最有效的方法是在编码过程中估算概率
    - ✓动态建模就成为确定编码器压缩效率的关键



无损压缩——基于字典的压缩

- 基于字典的压缩(dictionary encoding)
  - 压缩方法的思路完全不同于从 **Shannon** 到 **Huffman** 到算术压缩的传统思路
  - 编码开始时不知道要编码数据的统计特性，也不一定允许你事先知道它们的统计特性，而是根据数据本身包含有重复代码这个特性
  - 字典编码主要利用数据本身包含许多重复字符串的特性。例如：吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮。 我们如果用一些简单的代号代替这些字符串，就可以实现压缩，实际上就是利用了信源符号之间的相关性。字符串与代号的对应表就是字典
  - 实用字典编码算法的核心
    - ✓ 如何动态地形成字典
    - ✓ 如何选择输出格式以减小冗余
  - 字典编码法的种类很多，归纳起来大致有两类
    - ✓ **RLE**编码——行程编码
    - ✓ **LZW**编码



无损压缩——基于字典的压缩

RLE 编码——Run Length Encoding

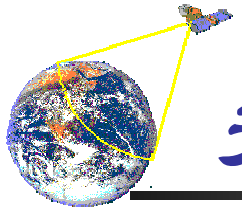
- 概念:
  - ✓ 行程: 具有相同灰度值的像素序列。
- 编码思想:
  - ✓ 去除像素冗余
  - ✓ 用行程的灰度和行程的长度代替行程本身。
- 基本原理:

通过改变图像的描述方式，来实现压缩。将一行中灰度值相同的相邻像素用一个计数值和该灰度值来代替。

- 举例:

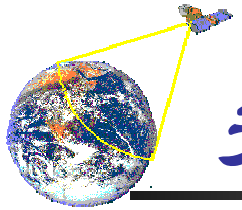
aaaa bbb cc d eeeee fffffff (共22*8=176 bits)

→4a3b2c1d5e7f (共12*8=96 bits)



无损压缩——基于字典的压缩

- **RLE 编码——Run Length Encoding**
  - 分析:
    - ✓对于有大面积色块的图像，压缩效果很好
    - ✓对于纷杂的图像，压缩效果不好，最坏情况下，会加倍图像

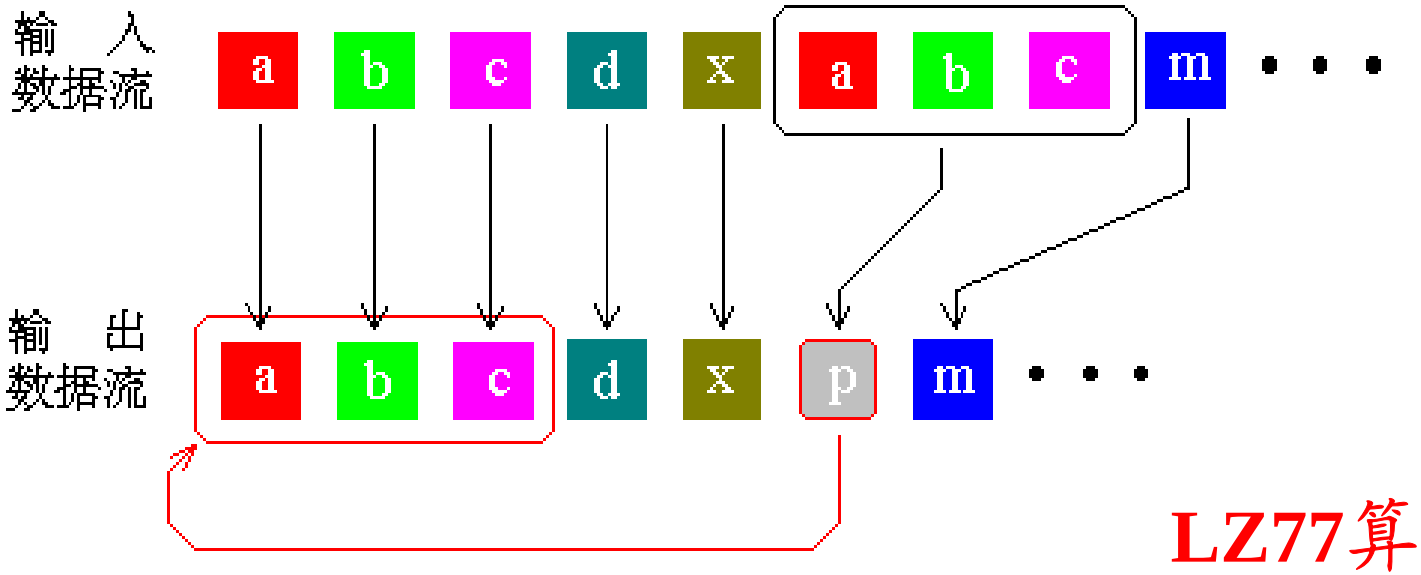


无损压缩—基于字典的压缩

- **LZW编码**
  - 背景:
    - ✓ 1977年以色列人Lempel、Ziv提出，形成LZ77 和 LZ78压缩算法
    - ✓ 1984年Welch充实，形成LZ78的一个变种－LZW算法
    - ✓ 改进的LZ77 和 LZ78、LZW 一起垄断当今通用数据压缩领域－gz、zip、arj
    - ✓ LZW继承了 LZ77 和 LZ78 压缩效果好、速度快的优点，而且在算法描述上更容易被人们接受（有的研究者认为是由于 Welch 的论文比 Ziv 和 Lempel 的更容易理解），实现也比较简单
    - ✓ 在标准图像数据格式中得到广泛应用—GIF、TIFF
  - 思路
    - ✓ 类似于中国的成语字典
    - ✓ 用数据中出现的字符序列形成字典，给出字典的索引作为压缩结果

无损压缩—基于字典的压缩

- **LZW编码（续）—两种方式**
  - 第一种：企图查找正在压缩的字符序列是否在以前输入的数据中出现过，然后用已经出现过的字符串替代重复的部分，它的输出仅仅是指向早期出现过的字符串的“指针”

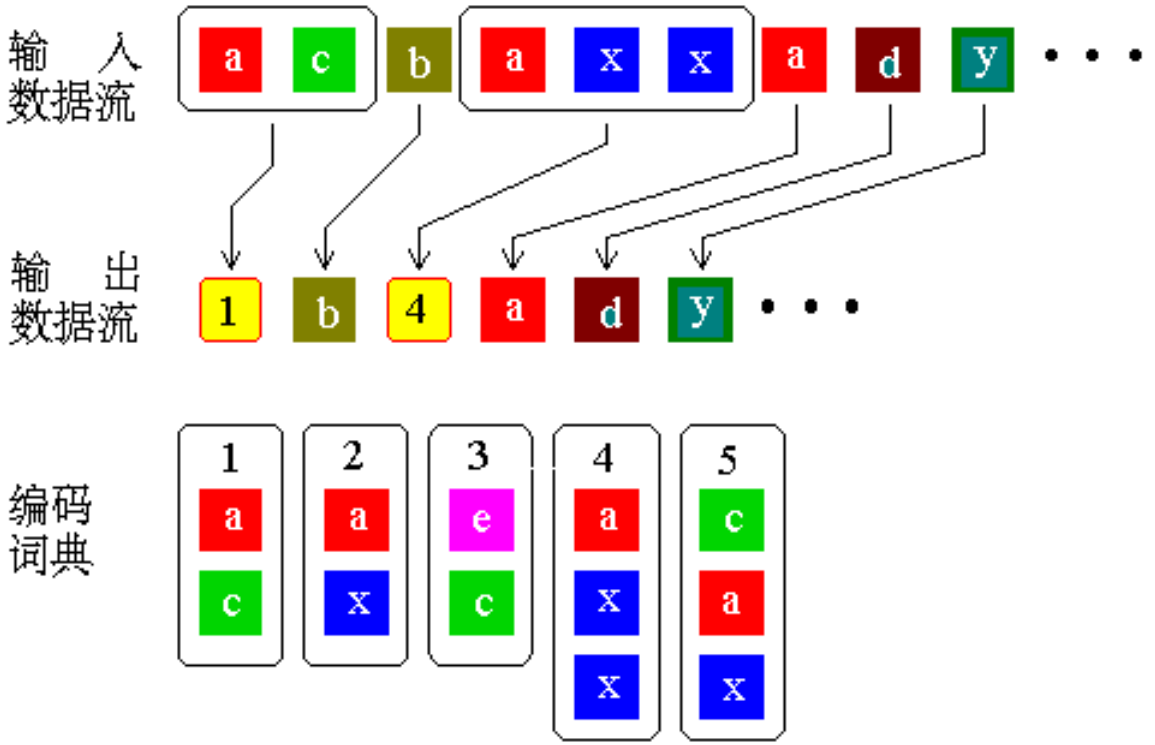


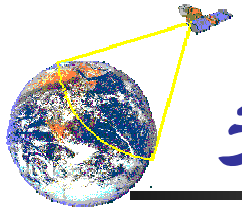
LZ77算法

无损压缩—基于字典的压缩

- **LZW编码（续）—两种方式**
  - 第二种：企图从输入的数据中创建一个“短语字典”，这种短语不一定是像“严谨勤奋求实创新”这类具有具体含义的短语，它可以是任意字符的组合。编码数据过程中当遇到已经在字典中出现的“短语”时，编码器就输出这个字典中的短语的“索引号”，而不是短语本身

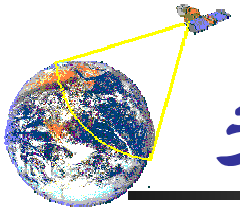
LZW(Lempel-Ziv Walch)压缩编码





无损压缩——基于字典的压缩

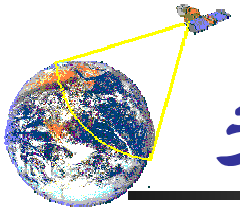
- **LZW编码（续）**
  - 基本思想：去除像素冗余
    - ✓ 在压缩过程中动态地形成一个字符序列表(字典)
    - ✓ 每当压缩扫描图像发现一个字典中没有的字符序列，就把该字符序列存到字典中
    - ✓ 用字典的地址（编码）作为这个字符序列的代码，替换原图像中的字符序列
    - ✓ 下次再碰到相同的字符序列，就用字典的地址代替字符序列
    - ✓ 压缩的结果，除了压缩图像外，不需要保留压缩过程中形成的字典，而在解压缩时， 临时恢复这个字典
  - LZW算法具有专利，所有者为美国Unisys(优利系统公司)，除了商业软件生产公司之外，可以免费使用LZW算法



无损压缩——基于字典的压缩

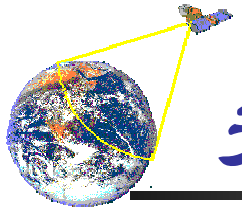
- **LZW编码（续）—编码算法**
  - LZW编码围绕称为字典的转换表来完成
  - 这张转换表存放前缀(Prefix)字符序列，并且为每个表项分配一个码字(Code word)，或者叫做序号
  - 这张转换表实际上是把8位ASCII字符集进行扩充，增加的符号用来表示在文本或图象中出现的可变长度ASCII字符串
  - 扩充后的代码可用9位、10位、11位、12位甚至更多的位来表示（Walsh用12位）
  - 12位可以有4096个不同的12位代码，即转换表有4096个表项，其中256个表项用来存放已定义的字符，剩下3840个表项用来存放前缀(Prefix)

码字	前缀(*Prefix*)
1	
…	…
193	A
194	B
…	…
255	
…	…
1305	abcdefxyF01234
…	…



无损压缩—基于字典的压缩

- **LZW编码（续）—编码算法**
  - **LZW编码器的输入是数据流，数据流可以是用8位ASCII字符组成的字符串，而输出是用*n*位(例如12位)表示的码字流，码字代表单个字符或多个字符组成的字符串**
  - **语法分析算法—贪婪分析算法(greedy parsing algorithm)**
    - ✓ 串行检查来自数据流的字符串，从中分解出已经识别的最长的字符串，也就是已经在字典中出现的最长的前缀***Prefix***。用已知的前缀***Prefix***加上下一个输入字符***C***也就是当前字符作为该前缀的扩展字符，形成新的扩展字符串－－“缀-符”串：***Prefix.C***
    - ✓ 检查字典中是否存有和它相同的“缀-符”串。如果有，那么这个“缀-符”串就变成前缀***Prefix***，继续输入新的字符，否则就把这个“缀-符”串***String***写到字典中生成一个新的前缀***Prefix***，并给一个代码。



无损压缩—基于字典的压缩

- **LZW编码（续）—编码算法**
  - **LZW编码算法的具体执行步骤**

步骤1：开始时的字典包含所有可能的单字符串，而当前前缀*P*是空的

步骤2：取当前字符*C*：*C* = 字符流中的下一个字符;

步骤3：判断“缀-符”串*P+C*是否在字典中

(1) 如果“是”：*P* = *P+C*　　(用*C*扩展*P*) ;

(2) 如果“否”

① 把代表当前前缀*P*的码字输出到码字流;

② 把“缀-符”串*P+C*添加到字典;

③ 令*P* = *C*　　(现在的*P*仅包含一个字符*C*);

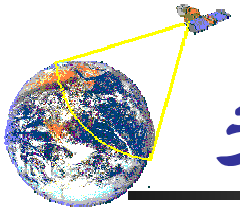
步骤4：判断码字流中是否还有码字要译

(1) 如果“是”，就返回到步骤2;

(2) 如果“否”

① 把代表当前前缀*P*的码字输出到码字流;

② 结束。



无损压缩—基于字典的压缩

- 编码实例

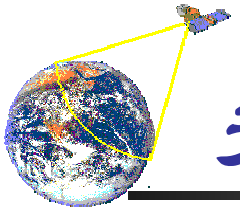
输入字符串：

位置	1	2	3	4	5	6	7	8	9
字符	A	B	B	A	B	A	B	A	C

输出码子串：

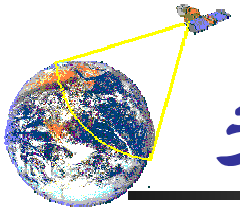
122473

步骤	位置	字典		输出
		(1)	A	
		(2)	B	
		(3)	C	
1	1	(4)	A B	(1)
2	2	(5)	B B	(2)
3	3	(6)	B A	(2)
4	4	(7)	A B A	(4)
5	6	(8)	A B A C	(7)
6	--	--	--	(3)



无损压缩—基于字典的压缩

- **LZW编码（续）—译码算法**
  - **LZW译码算法中用到两个术语:**
    - ✓ 当前码字(***Current code word***)：指当前正在处理的码字，用***cW***表示，用***string.cW***表示当前“缀-符”串;
    - ✓ 先前码字(***Previous code word***)：指先于当前码字的码字，用***pW***表示，用***string.pW***表示先前“缀-符”串。
  - **LZW译码算法开始时，译码字典与编码字典相同，它包含所有可能的前缀根(*roots*)。LZW算法在译码过程中会记住先前码字(*pW*)，从码字流中读当前码字(*cW*)之后输出当前“缀-符”串*string.cW*，然后把用*string.cW*的第一个字符扩展的先前“缀-符”串*string.pW*添加到字典中**



无损压缩—基于字典的压缩

- **LZW编码（续）—译码算法**
  - **LZW译码算法执行步骤**

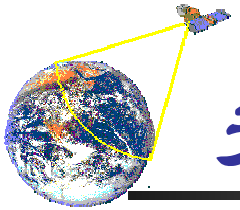
步骤1: 在开始译码时字典包含所有可能的前缀根(*Root*);

步骤2: 提取*cW*： *cW* =码字流中的第一个码字;

步骤3: 输出当前“缀-符”串*string.cW*到码字流;

步骤4: 检查先前码字*pW*: *pW* =*cW*;

步骤5: 当前码字*cW*：=码字流中的下一个码字;



无损压缩—基于字典的压缩

- **LZW编码（续）—译码算法**
  - **LZW译码算法执行步骤**

步骤6: 判断先前“缀-符”串***string.pW***是否在字典中

(1) 如果“是”:

① 把先前“缀-符”串***string.pW***输出到字符流;

② 当前前缀***P***：=先前“缀-符”串***string.pW***;

③ 当前字符***C***：=当前前“缀-符”串***string.cW***的第一个字符;

④ 把“缀-符”串***P+C***添加到字典;

(2) 如果“否”:

① 当前前缀***P***：=先前“缀-符”串***string.pW***;

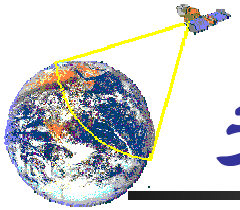
② 当前字符***C***：=当前“缀-符”串***string.cW***的第一个字符;

③ 输出“缀-符”串***P+C***到字符流，然后把它添加到字典中。

步骤7: 判断码字流中是否还有码字要译

(1) 如果“是”，就返回到步骤4;

(2) 如果“否”, 结束。



无损压缩—基于字典的压缩

- 译码实例

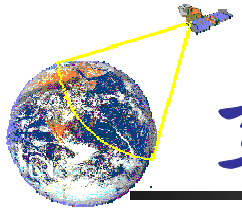
  输入码子串:

  122473

  输出字符串:

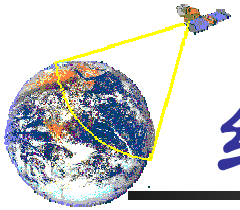
  ABBABABAC

步骤	代码	字典		输出
		(1)	A	
		(2)	B	
		(3)	C	
1	(1)	--	--	A
2	(2)	(4)	A B	B
3	(2)	(5)	B B	B
4	(4)	(6)	B A	A B
5	(7)	(7)	A B A	A B A
6	(3)	(8)	A B A C	C



第六章 图像压缩

- 习题
  - 习题1：教材P303第8、10题
  - 习题2：信息以*e*为底的单元通常称为一个奈特（nat），以10为底的单元通常称为一个哈特利（Hartley），计算将这些单元与底为2的信息单元（比特）联系起来的转换因子
  - 习题3：
    - ✓对一个三符号的信源有多少种唯一的霍夫曼编码
    - ✓构造这些唯一的编码
  - 习题4：画出LZW编码与解码算法实现流程框图



第六章（1）
结束