多媒体技术

回顾

- 视频
- 动画
- 图形
- 三维立体显示
- 数据冗余

3.1.2 数据压缩方法的分类

根据解码后数据与原始数据是否完全一致可以分为两大类:

无失真压缩，又称熵编码。由于不会失真，多用于文本、数据的压缩，非线性编辑系统为了保证视频质量，有些高档系统采用的是无失真压缩方法。

有失真压缩，又称熵压缩法。大多数图像、声音、动态视频等数据的压缩是采用有失真压缩。

3.1.2 数据压缩方法的分类

- 无失真压缩
  - 是一种泛指那些不考虑被压缩信息的性质的编码和压缩技术。
  - 它是基于平均信息量的技术把所有的数据当作比特序列，而不根据压缩信息的类型优化压缩。也就是说，平均信息量编码忽略被压缩信息的语义内容。

3.1.2 数据压缩方法的分类

- 有失真压缩
  - 冗余压缩取决于初始信号的类型、前后的相关性、信号的语义内容等。
  - 比严格的平均信息量编码的压缩率更高。

3.1.3 数据压缩技术的性能指标

- 有三个关键参数评价一个压缩系统
  - 压缩比
  - 压缩质量
  - 压缩和解压的速度
  - 另外也必须考虑每个压缩算法所需的硬件和软件。

3.1.3 数据压缩技术的性能指标

1. 压缩比

压缩性能常常用压缩比定义（输入数据和输出数据比）

例：512 × 480， 24bit/pixel(bpp)

输出15000byte 输入 = 737280byte

压缩比 = 737280/15000 = 49

3.1.3 数据压缩技术的性能指标

2. 压缩质量

压缩方法：　无损压缩　（质量不变）

　　有损压缩

有损压缩：　主观评分：看不出损失、损失明显等；

　　客观指标：信噪比等。

3.1.3 数据压缩技术的性能指标

3. 压缩解压速度

在许多应用中，压缩和解压可能不同时用，在不同的位置不同的系统中。所以，压缩、解压速度分别估计。

静态图像中，压缩速度没有解压速度严格；动态图像中，压缩、解压速度都有要求，因为需实时地从摄像机或VCR中抓取动态视频。

3.1.3 数据压缩技术的性能指标

4. 硬软件系统

有些压缩解压工作可用软件实现。设计系统时必须充分考虑:

算法复杂 – 压缩解压过程长

算法简单 – 压缩效果差

目前有些特殊硬件可用于加速压缩/解压。硬件系统速度快，但各种选择在初始设计时已确定，一般不能更改。因此在设计硬件压缩/解压系统时必须先将算法标准化。

3.2 常用的数据压缩编码

3.2 常用的数据压缩编码方法

- 统计编码
- 预测编码
- 变换编码
- 分析合成编码

3.2.1 统计编码

- 统计编码
  - 识别一个给定的流中出现频率最高的比特或字节模式，并用比原始比特更少的比特数来对其编码。
    - 频率越低的模式，其编码的位数越多，频率越高的模式编码位数越少。
    - 若码流中所有模式出现的概率相等，则平均信息量最大，信源就没有冗余。

(1) 香农-范诺编码
(2) 霍夫曼编码
(3) 算术编码
(4) 行程编码
(5) LZW编码

3.2.1 统计编码

1、香农-范诺编码

- 在香农的源编码理论中，熵的大小表示非冗余的不可压缩的信息量。
- 在计算熵时，如果对数的底数用2，熵的单位就用“香农(Sh)”，也称“位(bit)”。“位”是1948年Shannon首次使用的术语。
- 最早阐述和实现“从上到下”的熵编码方法的人是Shannon（1948年）和Fano（1949年），因此称为香农-范诺（Shannon- Fano）编码法。

3.2.1 统计编码

– 有一幅40个像素组成的灰度图像，灰度共有5级，分别用符号A，B，C，D和E表示。40个像素中出现灰度A的像素数有15个，出现灰度B的像素数有7个，出现灰度C的像素数有7个，其余情况见表

(1) 计算该图像可能获得的压缩比的理论值

(2) 对5个符号进行编码

(3) 计算该图像可能获得的压缩比的实际值

符号	A	B	C	D	E
出现的次数	15	7	7	6	5
出现的概率	15/40	7/40	7/40	6/40	5/40

3.2.1 统计编码

（1）压缩比的理论值

- 按照常规的编码方法，表示5个符号最少需要3位，如用000表示A，001表示B，…，100表示E，其余3个代码 (101，110，111)不用。这就意味每个像素用3位，编码这幅图像总共需要120位。

3.2.1 统计编码

（1）压缩比的理论值

– 按照香农理论，这幅图像的熵为

$$H(X)=-\sum_{i=1}^{n}p(x_i)\log_2 p(x_i)$$

$$=-p(A)\log_2(p(A))-p(B)\log_2(p(B))-\cdots-p(E)\log_2(p(E))$$

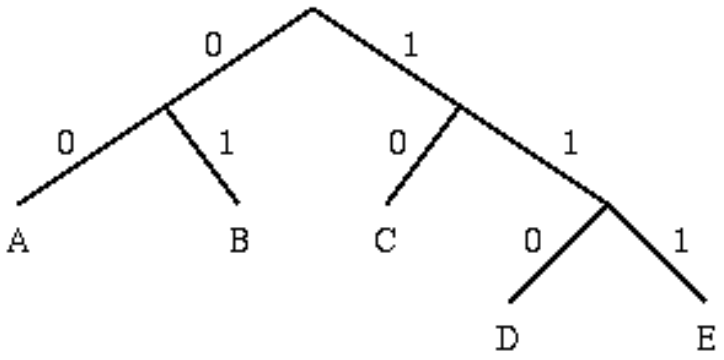
$$=(15/40)\log_2(40/15)+(7/40)\log_2(40/7)+\quad+(5/40)\log_2(40/5)\approx 2.196$$

- 这个数值表明，每个符号不需要用3位构成的代码表示，而用2.196位就可以，因此40个像素只需用87.84位就可以，因此在理论上，这幅图像的的压缩比为120:87.84≈1.37:1，实际上就是3:2.196≈1.37。

3.2.1 统计编码

（2）符号编码

- 对每个符号进行编码时采用“从上到下”的方法。
  - 首先按照符号出现的频度或概率排序，如A，B，C，D和E，见表2-2。
  - 然后使用递归方法分成两个部分，每一部分具有近似相同的次数，如图所示：



3.2.1 统计编码

表 2-2 Shannon- Fano 算法举例

符号	出现的次数($p(x_i)$)	$\log_2(1/p(x_i))$	分配的代码	需要的位数
A	15 (0.375)	1.4150	00	30
B	7 (0.175)	2.5145	01	14
C	7 (0.175)	2.5145	10	14
D	6 (0.150)	2.7369	110	18
E	5 (0.125)	3.0000	111	15

3.2.1 统计编码

（3）压缩比的实际值

- 按照这种方法进行编码需要的总位数为30+14+14+18+15 = 91，实际的压缩比为120:91≈1.32 : 1

3.2.1 统计编码

2、霍夫曼编码

- 霍夫曼(D.A. Huffman)在1952年提出和描述的“从下到上”的熵编码方法。
- 根据给定数据集中各元素所出现的频率来压缩数据的一种统计压缩编码方法。这些元素(如字母)出现的次数越多，其编码的位数就越少。
- 广泛用在JPEG, MPEG, H.26X等各种信息编码标准中。

3.2.1 统计编码

现有一个由5个不同符号组成的30个符号的字符串：

BABACACADADABBCBABEBEDDABEEEBB

- 计算
  - (1) 该字符串的霍夫曼码
  - (2) 该字符串的熵
  - (3) 该字符串的平均码长
  - (4) 编码前后的压缩比

3.2.1 统计编码

符号出现的概率

符号	出现的次数	$\log_2(1/p_i)$	分配的代码	需要的位数
A	8	1. 907	?	
B	10	1. 585	?	
C	3	3.322	?	
D	4	2.907	?	
E	5	2.585	?	
合计	30			

3.2.1 统计编码

(1) 计算该字符串的霍夫曼码

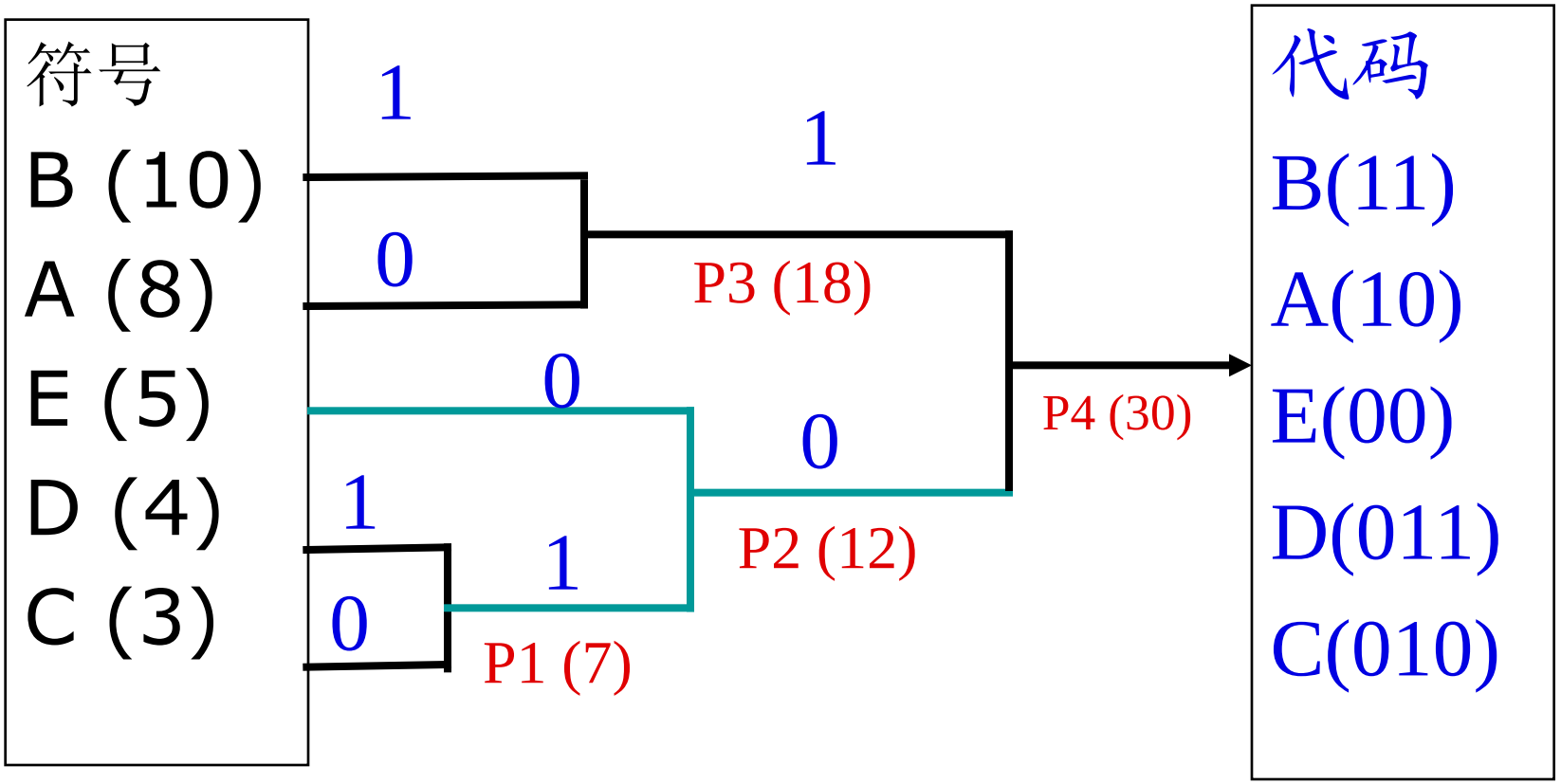
- 步骤①：按照符号出现概率大小的顺序对符号进行排序。
- 步骤②：把概率最小的两个符号组成一个节点P1。
- 步骤③：重复步骤②，得到节点P2，P3，P4，……，PN，形成一棵树，其中的PN称为根节点。

3.2.1 统计编码

(1) 计算该字符串的霍夫曼码

- 步骤④：从根节点PN开始到每个符号的树叶，从上到下标上0(上枝)和1(下枝)，至于哪个为1哪个为0则无关紧要，但通常把概率大的标成1，概率小的标成0。
- 步骤⑤：从根节点PN开始顺着树枝到每个叶子分别写出每个符号的代码。

3.2.1 统计编码


符号
B (10)
A (8)
E (5)
D (4)
C (3)
1
0
1
0
1
0
1
0
P1 (7)
P2 (12)
P3 (18)
P4 (30)
代码
B(11)
A(10)
E(00)
D(011)
C(010)

3.2.1 统计编码

5个符号的代码

符号	出现的次数	$\log_2(1/p_i)$	分配的代码	需要的位数
A	8	1.907	10	16
B	10	1.585	11	20
C	3	3.322	010	9
D	4	2.907	011	12
E	5	2.585	00	10
合计	30	1.0		67

3.2.1 统计编码

(2) 计算该字符串的熵

- $H(S)$

$$
\begin{aligned}
&=(8/30)\times\log_2(30/8)+(10/30)\times\log_2(30/10)+\\
&\quad(3/30)\times\log_2(30/3)+(4/30)\times\log_2(30/4)+\\
&\quad(5/30)\times\log_2(30/5)\\
&=[30\lg30-(8\times\lg8+10\times\lg10+3\times\lg3+4\\
&\quad\times\lg4+5\lg5)]/(30\times\log_2 2)\\
&=(44.3136-24.5592)/9.0308=2.1874\quad(\mathrm{Sh})
\end{aligned}
$$

3.2.1 统计编码

(3) 计算该字符串的平均码长

- 平均码长:

$$= (2 \times 8 + 2 \times 10 + 3 \times 3 + 3 \times 4 + 2 \times 5)/30$$
$$= 2.233 \text{ 位/符号}$$

平均码长：67/30=2.233位

(4) 计算编码前后的压缩比

- 编码前：5个符号需3位，30个字符，需要90位
- 编码后：共67位

压缩比: 90/67=1.34:1

3.2.1 统计编码

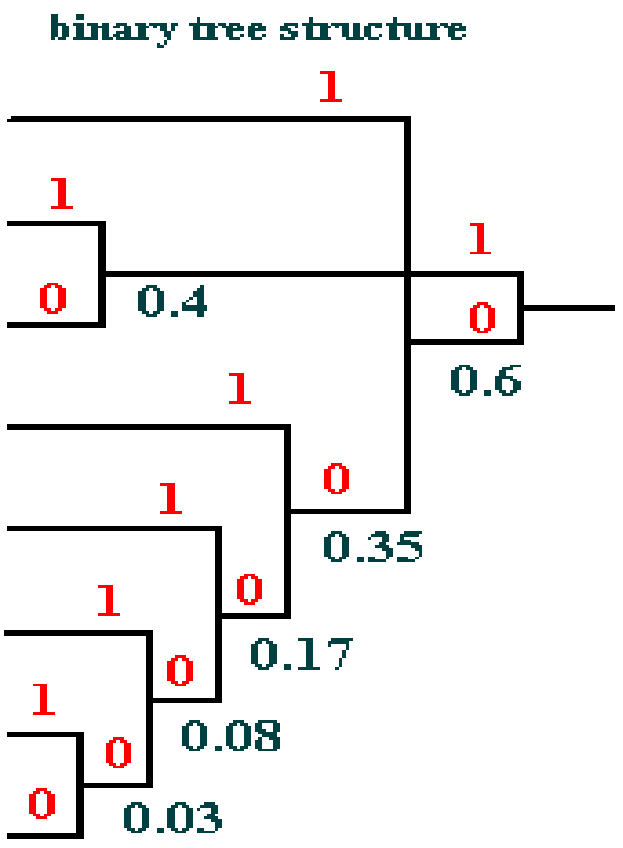
- 编码前
  - N = 8 符号: {a,b,c,d,e,f,g,h},
  - 每个符号3个位 (N =2^3=8)
  - P(a) = 0.01, P(b)=0.02, P(c)=0.05, P(d)=0.09, P(e)=0.18, P(f)=0.2, P(g)=0.2, P(h)=0.25
- 计算

  (1) 该字符串的霍夫曼码 (2) 该字符串的熵

  (3) 该字符串的平均码长 (4) 编码效率

3.2.1 统计编码

Original codewords	Huffman codewords	Symbol probabilities
111	01	P(h)=0.25
110	11	P(g)=0.2
101	10	P(f)=0.2
100	001	P(e)=0.18
011	0001	P(d)=0.09
010	00001	P(c)=0.05
001	000001	P(b)=0.02
000	000000	P(a)=0.01



3.2.1 统计编码

- 编码前每个符号的平均长度

$$\overline{L} = \sum_{i=1}^{8} 3P(i) = 3 \text{ bits/symbol}$$

- 熵

$$H = -\sum_{i=1}^{8} P(i)\log_2 P(i) = 2.5821 \text{ bits/symbol}$$

- 编码后每个符号的平均长度

$$\overline{L}_{Huf} = 2.63 \text{ bits/symbol}$$

- 编码有效性

$$H / \overline{L}_{Huf} = 98\%$$

3.2.1 统计编码

- 优点：当信源符号概率是2的负幂次方时，Huffman 编码法编码效率达到100%。一般情况下，它的编码效率要比其它编码方法的效率高，是最佳变长码。

- 缺点：Huffman 码依赖于信源的统计特性，必须先统计得到信源的概率特性才能编码，这就限制了实际的应用。通常可在经验基础上预先提供Huffman码表，此时性能有所下降。

3.2.1 统计编码

3、算术编码

- 给已知统计信息的符号分配代码的数据无损压缩技术。
- 基本思想是用0和1之间的一个数值范围表示输入流中的一个字符，而不是给输入流中的每个字符分别指定一个码字。
- 实质上是为整个输入字符流分配一个“码字”，因此它的编码效率可接近于熵。

3.2.1 统计编码

- 假设信源符号为{00, 01, 10, 11}，它们的概率分别为{ 0.1, 0.4, 0.2, 0.3 }
- 对二进制消息序列10 00 11 00 10 11 01 … 进行算术编码

3.2.1 统计编码

（1）初始化

– 根据信源符号的概率把间隔[0, 1)分成如表2-4所示的4个子间隔：[0, 0.1), [0.1, 0.5), [0.5, 0.7), [0.7, 1)。其中[x, y)的表示半开放间隔，即包含x不包含y，x称为低边界或左边界，y称为高边界或右边界。

信源符号概率和初始编码间隔

符号	00	01	10	11
概率	0.1	0.4	0.2	0.3
初始编码间隔	[0, 0.1)	[0.1, 0.5)	[0.5, 0.7)	[0.7, 1]

3.2.1 统计编码

（2）确定符号的编码范围

10 00 11 00 10 11 01 …

- 编码时输入第1个符号是10，找到它的编码范围是[0.5, 0.7)。
- 消息中第2个符号00的编码范围是[0, 0.1)，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)。
- 编码第3个符号11时，取新间隔为[0.514, 0.52)。

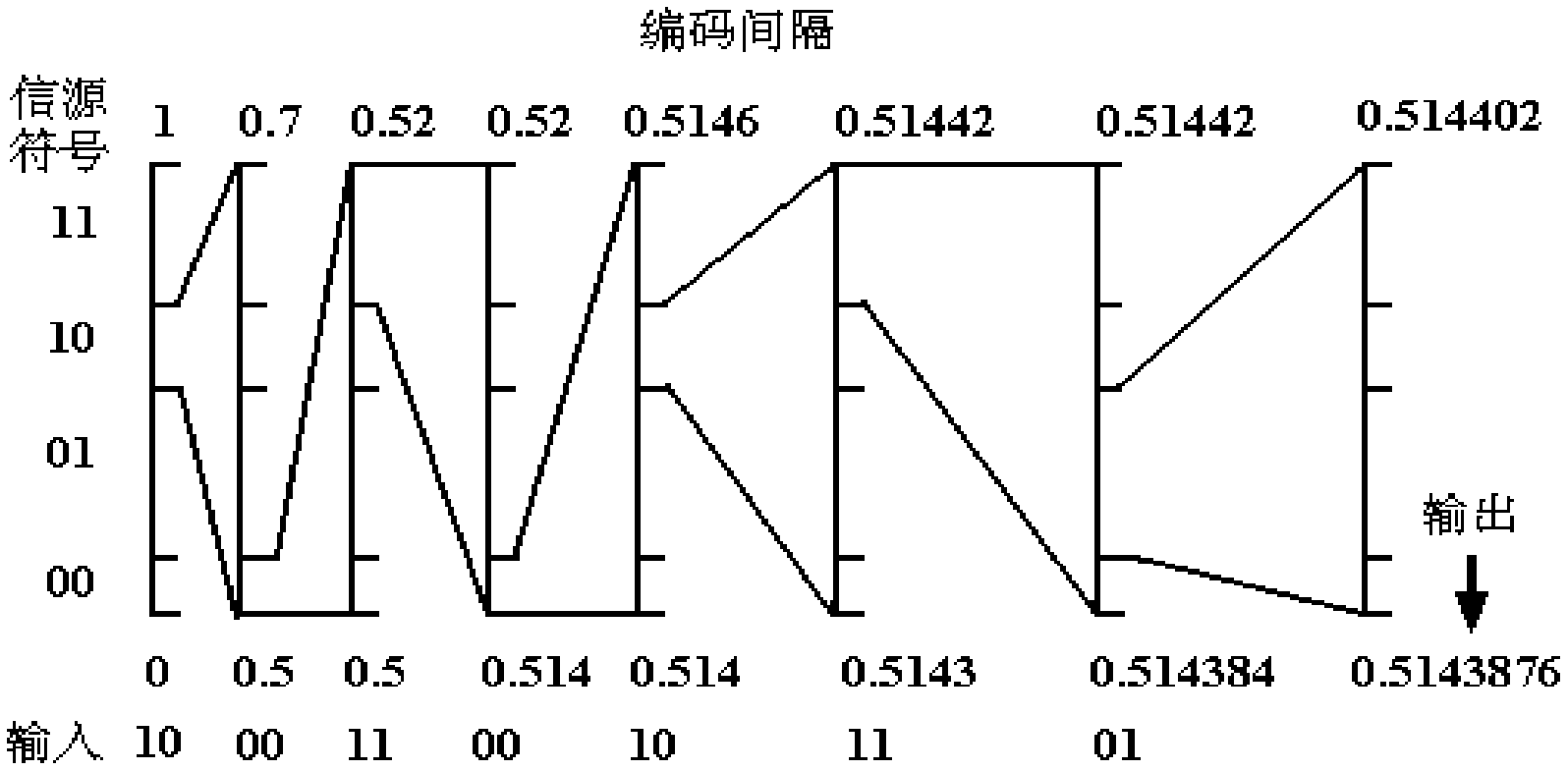
3.2.1 统计编码

（2）确定符号的编码范围

10 00 11 00 10 11 01 …

- 编码第4个符号00时，取新间隔为[0.514, 0.5146)。
- 依此类推……
- 消息的编码输出可以是最后一个间隔中的任意数。

3.2.1 统计编码


编码间隔
信源符号
11
10
01
00
1
0.7
0.52
0.52
0.5146
0.51442
0.51442
0.514402
输出
0
0.5
0.5
0.514
0.514
0.5143
0.514384
0.5143876
输入
10
00
11
00
10
11
01

3.2.1 统计编码

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]间隔的最后三个 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，两个 1/10
6	11	[0.514384, 0.51442]	[0.5143, 0.51442]间隔的最后 3 个 1/10
7	01	[0.5143876, 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(如 0.5143876)作为输出		

3.2.1 统计编码

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	[0.5143876, 0.514402]	01	0.51439 在间隔 [0.514384, 0.51442] 的第 1 个 1/10
	译码的消息：10 00 11 00 10 11 01		

3.2.1 统计编码

- 在算术编码中有几个问题需要注意：
  - 由于实际的计算机的精度不可能无限长，运算中出现溢出是一个明显的问题，但多数机器都有16-, 32-或者64位的精度，因此这个问题可使用比例缩放方法解决。
  - 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。
  - 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。

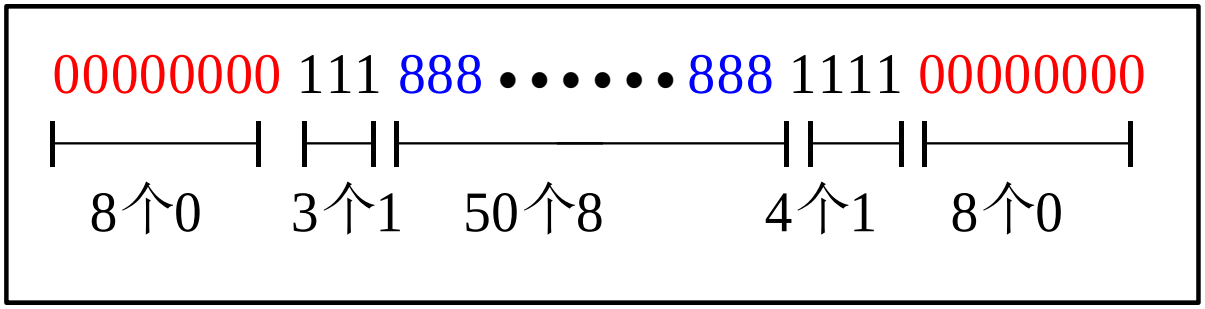
3.2.1 统计编码

4、行程编码

- 一种无损压缩数据编码技术，它利用重复的数据单元有相同的数值这一特点对数据进行压缩。
- 对相同的数值只编码一次，同时计算出相同值重复出现的次数。在JPEG，MPEG，H.261和H.263等压缩方法中，RLE用来对图像数据变换和量化后的系数进行编码。

3.2.1 统计编码

– 例: 假设有一幅灰度图像第n行的像素值如图所示。用RLE编码方法得到的代码为: 8031508418 0



3.2.1 统计编码

- **AAAAAAAAAAAAAAA**
  - **15A**
  - **压缩率：15 bytes / 2 bytes = 7.5**
- **AAAABBBBBCCCCCCCCDEEEE**
  - **4A5B8C1D4E**
  - **压缩率：22 bytes / 10 bytes = 2.2**
- **MyDogHasFleas**
  - **1M1y1D1o1g1H1a1s1F1l1e1a1s**
  - **压缩率：13 bytes / 26 bytes = 0.5**

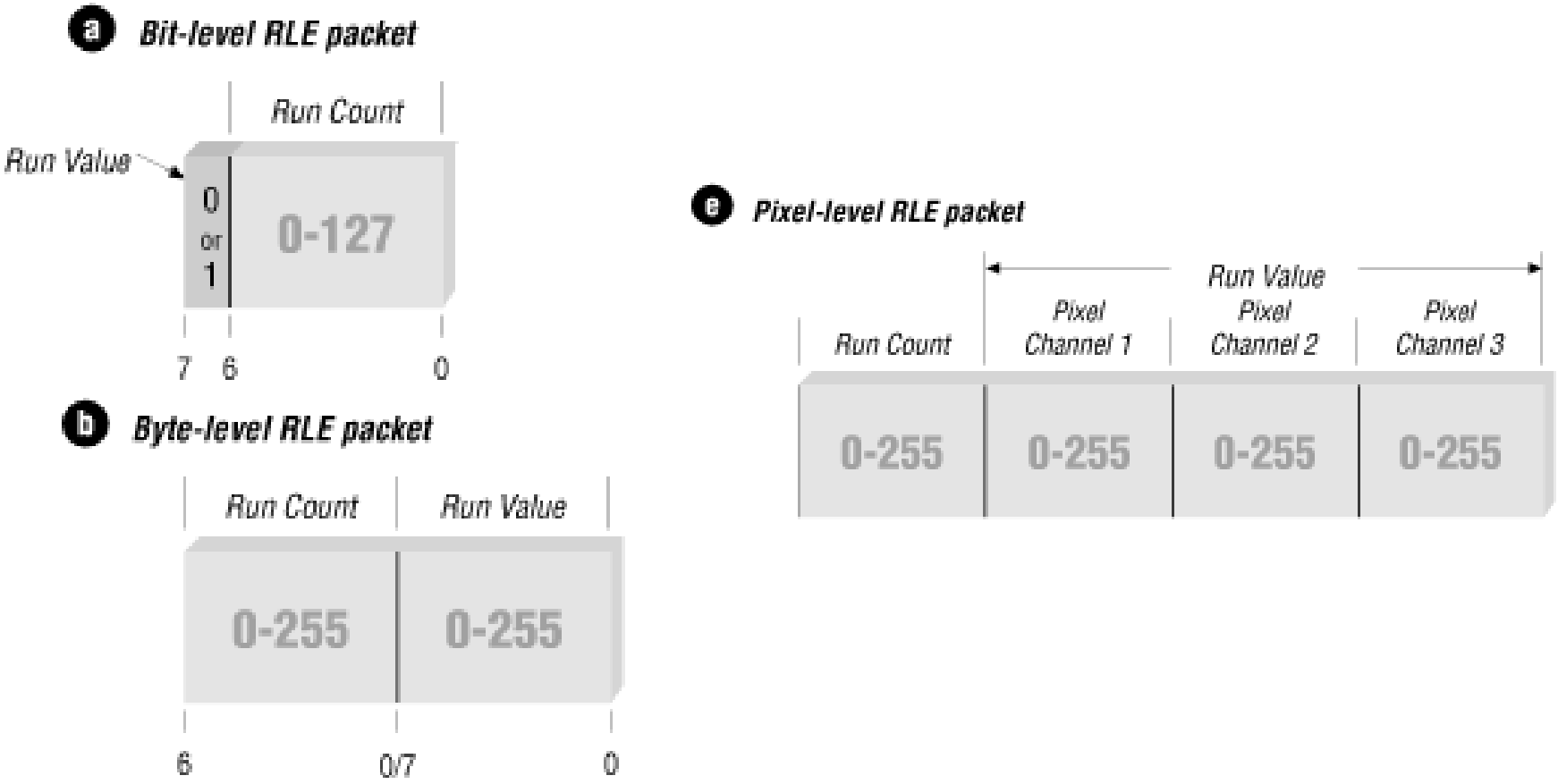
3.2.1 统计编码

- 在对图像进行编码时，沿着一定方向排列的具有相同灰度值的像素可看成是连续符号，用字串代替这些连续符号，可大幅度减少数据量。
- RLE获得的压缩比多大取决于图像本身的特点。

3.2.1 统计编码

- 对位图图像编码时，根据所编码的元素的类型，RLE编码模式可以分为：
  - 位模式
    - 对Bit进行编码，而忽略Byte和Word的界限
    - 单色图像（monochrome）
  - 字节模式
    - 对Byte进行编码，而忽略Bit和Word的界限
    - 适用于1Byte/Pixel
  - 像素模式
    - 对Pixel进行编码，一个Pixel用多个Byte表示

3.2.1 统计编码


a Bit-level RLE packet
Run Count
Run Value
0 or 1
0-127
7 6 0
b Byte-level RLE packet
Run Count
Run Value
0-255
0-255
6 0/7 0
e Pixel-level RLE packet
Run Value
Run Count
Pixel Channel 1
Pixel Channel 2
Pixel Channel 3
0-255
0-255
0-255
0-255

3.2.1 统计编码

- 一种改进方法
  - 包含3个及以上的重复时，进行编码;
    - 5555
  - 有压缩代码和非压缩代码，使用标志符号（如255）区分;
    - FF
  - 使用标志符号+计数（1或2）：表示标志符号表示值;
    - FF 01

3.2.1 统计编码

- 一种改进方法

1）三个以上的相同字符（包括标志字节和非标志字节）　[标志字节]+计数+字符字节

2）一个或两个非标志字节字符　字符字节或字符字节+字符字节

3）一个或两个标志字节　[标志字节（也是字符字节）]+计数

待压缩：**AAAA76FFFF01010101FFFFFFBC**

AAAA 76 FF02 FF0401 FF03FF BC

3.2.1 统计编码

- 一种改进方法

1）三个以上的相同字符（包括标志字节和非标志字节）　[标志字节]+计数+字符字节

2）一个或两个非标志字节字符　字符字节或字符字节+字符字节

3）一个或两个标志字节　[标志字节（也是字符字节）]+计数

待解压：FF05050201FF01FF03020505

0505050505 02 01 FF 020202 0505

3.2.1 统计编码

- 选用RLE作为压缩编码算法的图片格式有:
  - MacPaint
  - PCX
  - TIFF
  - RLE（CompuServe， Utah以及 Microsoft）

3.2.1 统计编码

5、LZW编码

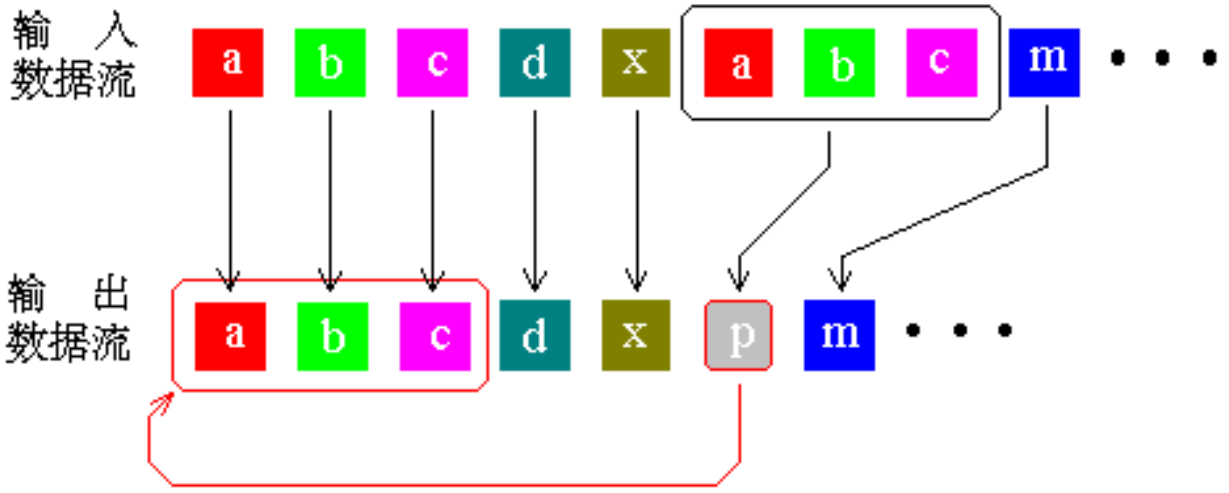
- 词典编码的思想：词典编码的根据是数据本身包含有重复代码这个特性。例如文本文件就具有这种特性。词典编码法的种类很多，归纳起来大致有两类。

3.2.1 统计编码

- 第一类词典法的想法是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”。
  - 这里所指的“词典”：是指用以前处理过的数据来表示编码过程中遇到的重复部分。

3.2.1 统计编码

- 第一类编码算法
  - 用已经出现过的字符串替代重复的部分
  - 编码器的输出仅仅是指向早期出现过的字符串的“指针”



3.2.1 统计编码

- 第二类算法的想法是企图从输入的数据中创建一个“短语词典(dictionary of the phrases)”，短语可以是任意字符的组合。编码数据过程中当遇到已经在词典中出现的“短语”时，编码器就输出这个词典中的短语的“索引号”，而不是短语本身。

3.2.1 统计编码

- 第二类编码算法
  - 从输入的数据中创建一个“短语词典”
  - 编码器输出词典中的短语“索引号”，而不是短语