第五章 信源编码

信源编码与信道编码

- **信源编码**
  - **无失真信源编码——第一极限定理**：离散信源
  - **限失真信源编码——第三极限定理**：连续信源
  - 在不失真或允许一定失真条件下，如何用尽可能少的符号来传送信源信息，以便提高信息传输率。
- **信道编码**
  - **第二极限定理**：离散和连续信道
  - 在信道受干扰的情况下如何增加信号的抗干扰能力，同时又使得信息传输率最大。
- **信源编码的作用**
  i. **符号变换**：使信源的输出符号与信道的输入符号相匹配；
  ii. **信息匹配**：使信息传输率达到信道容量；
  iii. **冗余度压缩**：使编码效率等于或接近100%。
  例题

例3-12 离散无记忆信源，输出符号概率分布如下表。 $H(X)=1.75$ bit/信源符号

通过一个无噪无损二元离散信道进行传输。

信道容量 $C=1$ bit/信道符号

	x_1	x_2	x_3	x_4
$p(x_i)$	1/2	1/4	1/8	1/8
编码1	00	01	10	11
编码2	000	001	010	011

编码1：信道传输率 $R_1=\frac{H(X)}{2}=\frac{1.75}{2}=0.875$ bit/信道符号

绝对冗余度 $=C-R_1=1-0.875=0.125$

相对冗余度 $=\frac{C-R_1}{C}=12.5\%$

编码2：信道传输率 $R_2=\frac{H(X)}{3}=0.583$ bit/信道符号

绝对冗余度 $=C-R_2=1-0.583=0.417$

相对冗余度 $=\frac{C-R_2}{C}=41.7\%$

- **信源编码的基础**
  i. **无失真编码定理**：可精确复制信源输出的消息，只适用于离散信源
  ii. **限失真编码定理**：对于连续信源，只能在失真受限制的情况下进行限失真编码

5.1 编码的概念

分组码

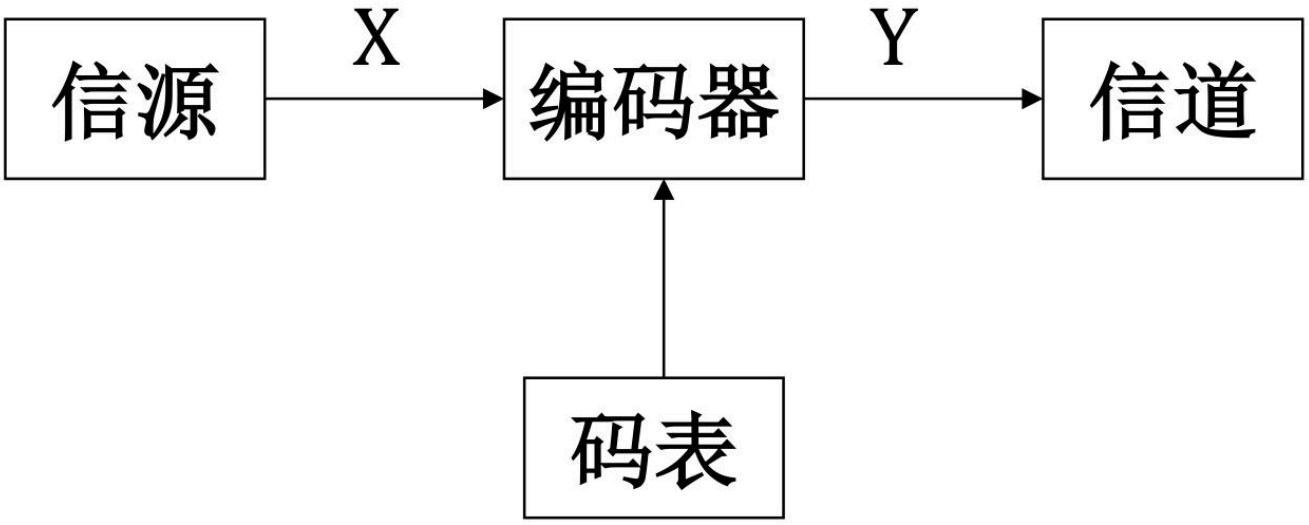
- **分组码(Block Codes)**：也叫块码
  - 将信源消息分成若干组，即符号序列 x_i ，

$$x_i = (x_{i1}x_{i2}\cdots x_{il}\cdots x_{iL})$$
$$x_{il} \in A = \{a_1, a_2, \cdots, a_i, \cdots, a_n\}$$

  - 每个符号序列 x_i 依照**固定码表**映射成一个码字 y_i ，

$$y_i = (y_{i1}y_{i2}\cdots y_{ik}\cdots y_{iK})$$
$$y_{ik} \in B = \{b_1, b_2, \cdots, b_i, \cdots, b_m\}$$

  - 只有分组码才有对应的码表，而非分组码中则不存在码表。
  - **模型**：

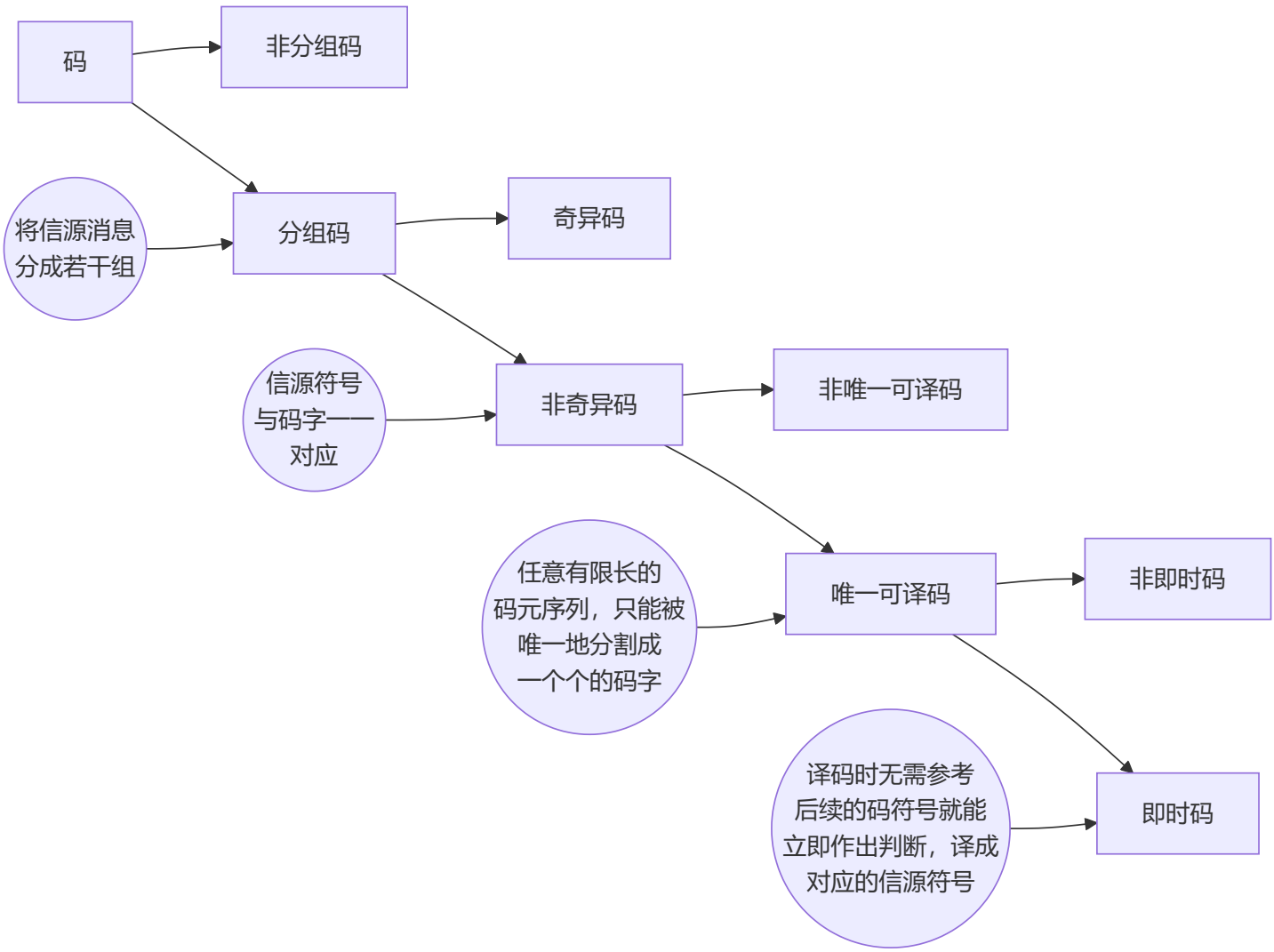


- **定长码与变长码**：码可分为两类：
  - **定长码**：码中所有码字的长度都相同，如码1就是定长码(Fixed Length Codes)。
  - **变长码**：码中的码字长短不一，如码2就是变长码(Variable Length Codes)。
  -

符号 a_i	**信源符号出现概率 $p(a_i)$**	**码1**	**码2**
a_1	$p(a_1)$	00	0
a_2	$p(a_2)$	01	01
a_3	$p(a_3)$	10	001
a_4	$p(a_4)$	11	111

- **码的属性**：
  - **奇异码与非奇异码**：
    - 若信源符号和码字是一一对应的，则该码为**非奇异码**(Non-Singular Codes)；
    - 反之为**奇异码**(Singular Codes)
  - **唯一可译码(Uniquely Decodable Codes_)**：
    - 任意有限长的码元序列，只能被唯一地分割成一个个的码字，便称为唯一可译码。
    - 奇异码不是唯一可译码
    - 唯一可译码分为非即时码和即时码
      - **非即时码**指接收端收到一个完整的码字后不能立即译码，还需等下一个码字开始接收后才能判断是否可以译码
      - **即时码**只要收到符号就表示该码字已完整，可以立即译码 ，又称为**非延长码**(Undelayed Codes)，任意一个码字都不是其它码字的前缀部分，又称为**异前缀码**(Prefix Codes)。

码的分类



即时码及其树图构造法

- 即时码(非延长码或异前缀码)是唯一可译码的一类子码，可用树图来构造
- **构造的要点**：
  - 最上端为树根A，从根出发向下伸出树枝，树枝总数等于m(**进制数**)，树枝的尽头为节点。
  - 从每个节点再伸出m个树枝，当某个节点被安排为码字后，就不再伸枝，这节点为**终端节点**。能再伸枝的节点成为**中间节点**。一直继续下去，直至都不能伸枝为止。
  - 每个节点所伸出的树枝标上码符号，从根出发到终端点所走路径对应的码符号序列则为终端节点的码字。
- **用码树图构造码**
  - 在树的生长过程中，节点生出树枝，各树枝旁标出相应的码符，为了清晰起见相同码符的树枝方向相同，终端节点表示信源符号，从树根到终端节点所经过的树枝旁的码符按经过的顺序组成的序列构成码字。
- **用码树图判断即时码**
  - 如果表示信源符号的终端节点不再延伸，或到达任一 信源符号终端节点的路径不经过其它的终端节点，这样构 造的码满足即时码条件。
- **码树与码字对应关系**

树结构概念	**编码概念**
树根	码字的起点
树枝数	码的进制数
节点	码字或码字的一部分
终端节点	码字
节数	码长
非满树	变长码（码字为叶节点）
满树	等长码（码字为叶节点）

克劳夫特不等式

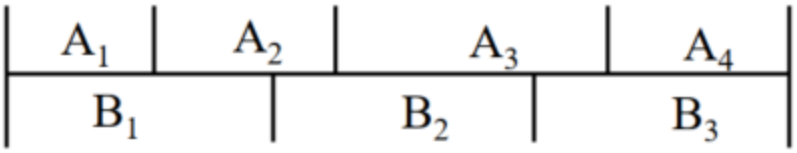
- **唯一可译码存在**的充分必要条件是各码字的长度 K_i 应符合**克劳夫特不等式**(Kraft Inequality)

$$\sum_{i=1}^{n} m^{-K_i} \leq 1$$

式中，m是编码进制数，n是信源符号数。

- 必要性表现在如果码是唯一可译码，则必定满足该不等式
- 充分性表现在如果满足该不等式，则这种码长的唯一可译码一定存在，但并不表示所有满足不等式的一定是唯一可译码。
- 所以说，该不等式是唯一可译码存在的充要条件，而不是判别一个编码是否唯一可译码的充要条件

- **唯一可译码的判断方法**
  i. **基本判断**:
    a. 观察是否是非奇异码，若是奇异码则一定不是唯一可译码。
    b. 计算是否满足Kraft不等式，若不满足一定不是唯一可译码。
    c. 将码画成一棵树图，观察是否满足即时码的树图的构造，若满足则是唯一可译码。
  ii. **尾随后缀法**:



    a. 考查 C 中所有的码字，若 W_i 是 W_j 的前缀，则将相应的后缀作为一个尾随后缀码放入集合 F_0 中
    b. 考查 C 和 F_i 两个集合, $W_i \in C$ 是 $W_j \in F_i$ 的前缀或 $W_i \in F_i$ 是 $W_j \in C$ 的前缀，则将相应的后缀作为尾随后缀码放入集合 F_{i+1} 中;
    c. $F = \cup_i F_i$ 即为码 C 的尾随后缀集合;
    d. 构造尾随后缀集合F，若F中出现了C中的元素，则算法终止，返回假(C不是唯一可译码)；否则若F中没有出现新的元素，则返回真。

5.2 无失真信源编码定理

无失真信源编码

- **模型**:

  - 信源编码器输入的消息序列:

$$X = (X_1 X_2 \cdots X_l \cdots X_L) \quad X_l \in \{a_1, \cdots a_n\}$$

输入的消息总共有 n^L 种可能的组合
输出的码字为：

$$Y=(Y_1Y_2\cdots Y_k\cdots Y_{K_L})\quad Y_k\in\{b_1,\cdots b_m\}$$

输出的码字总共有 m^{K_L} 种可能的组合。

- Y_k有m种可能取值，所以平均每个符号输出的最大信息量为$\log m$（等概分布）。
- K_L长码字的最大信息量为$K_L\log m$，用该码字表示L长的信源序列。
- 则传送一个信源符号需要的平均信息率为：

$$\overline{K}=\frac{K_L}{L}\log m=\frac{1}{L}\log M\ \text{bit/信源符号}$$

其中，$M=m^{K_L}$是Y所能编成的码字的个数。

- **无失真信源编码**
  - **实现无失真的信源编码要求**：
    - 信源符号 $X_1X_2\cdots X_l\cdots X_L$ 与码字 $Y_1Y_2\cdots Y_k\cdots Y_{K_L}$ 是一一对应的；
    - 能够无失真或无差错地从Y恢复X，也就是能正确地进行反变换或译码；
    - 传送Y时所需要的信息率最小，信息率最小就是找到一种编码方式使$\overline{K}=\frac{K_L}{L}\log m=\frac{1}{L}\log M$最小
  - **无失真信源编码定理研究内容**：
    - 最小信息率为多少时，才能得到无失真的译码？
    - 若小于这个信息率是否还能无失真地译码？

定长编码

定长编码的基本概念

- 码长K是定值，且是唯一可译码。
- 由L个符号组成的、每个符号的熵为$H_L(X)$的无记忆平稳信源符号序列$X_1X_2...X_l...X_L$（每个符号n种可能值）
- 输入的消息总共有n^L种可能的组合。
- 可用$K_L=K$个符号Y_1，Y_2，...，Y_k，...（每个符号有m种可能值）进行定长编码
- 输出的码字总共有m^K种可能的组合
- **若要对信源进行定长编码且无失真**，必须满足：

$$n^L\le m^K\text{或}\frac{K}{L}\ge\frac{\log n}{\log m}$$

  - 只有当K长的码符号序列数m^K大于或等于信源的符号数n^L时，才可能存在定长非奇异码。

渐进均分性定理（AEP）

- **定理**：$\vec{X}=(X_1X_2\cdots X_L)$，为独立同分布（i.i.d）随机变量序列，具有渐近均分性质（AEP，Asymptotic equipartition property）：

$$\forall\varepsilon>0，\ 当L\to\infty时，\ p(X_1,X_2,\cdots,X_L)\to 2^{-LH(X)}$$

- **证明**：**见无失真信源编码定理**

定长编码定理

- **定理**：对于由 L 个符号组成的，每个符号的熵为 $H_L(\vec{X})$ 的无记忆平稳符号序列 $X_1,X_2,\cdots,X_L$，可用 K_L 个符号 $Y_1,Y_2,\cdots,Y_{K_L}$（每个符号有 m 种可能值，即m进制编码）进行定长编码。对于任意 $\varepsilon>0$，$\delta>0$，只要

$$\overline{K}=\frac{K_L}{L}\log m\geq H_L(\vec{X})+\varepsilon$$

  则当 L 足够大时，必可使译码差错小于 δ；
  反之，当

$$\overline{K}=\frac{K_L}{L}\log m\leq H_L(\vec{X})-2\varepsilon$$

  时，译码差错一定是有限值，当 $L\to\infty$ 时，译码几乎必定出错。
- **证明**：**见无失真信源编码定理**
- **定长编码定理含义**
  - 当编码器容许的输出信息率，也就是当每个信源符号所必须输出的二进制码长是

$$\overline{K}=\frac{K_L}{L}\log m=\frac{1}{L}\log M$$

    时，只要 $\overline{K}>H_L(X)$，这种编码器一定可以做到**几乎无失真**，也就是收端的译码差错概率接近于零，**条件是所取的符号数 L足够大。**
  - 将定理的条件改写成：

$$K_L\log m>LH_L(X)=H(X)$$

    其中：
    - 左边：K_L长码字所能携带的最大信息。
    - 右边：L长信源序列携带的信息量。

    则：

    - 码字所能携带的信息量**大于**信源序列输出的信息量，则可以使传输**几乎无失真**，当然**条件是 L足够大**。
    - 反之，当 $\overline{K} < H_L(X)$ 时，不可能构成无失真的编码，也就是不可能做一种编码器，能使收端译码时差错概率趋于零。
    - $\overline{K} = H_L(X)$ 时，则为临界状态，可能无失真，也可能有失真。
- **信源长度 L**
    - 对定长编码，若要实现几乎无失真编码，则信源长度必须满足：

$$L \geq \frac{\sigma^2(X)}{\varepsilon^2 \delta}$$

      其中：
        - $\sigma^2(X) = E\{[I(x_i) - H(X)]^2\}$，表示信源序列的自信息方差。
        - δ：差错率要求（如10^{-6}）
        - $\varepsilon = \overline{K} - H_L(X)$
    - 证明：见无失真信源编码定理
- **编码效率η**
    - 编码效率定义为

$$\eta = \frac{H_L(X)}{\overline{K}}$$

      即信源的平均符号熵为$H_L(X)$，采用平均二进制符号码长为$\overline{K}$来编码，所得的效率。
    - 无失真编码效率总是小于1，且**最佳编码效率**为

$$\eta = \frac{H_L(X)}{H_L(X) + \varepsilon}, \varepsilon > 0$$

    - 定长编码定理从理论上阐明了编码效率接近1的理想定长编码器的存在性，它使输出符号的信息率与信源熵之比接近于1，即只要L足够大

$$\frac{H_L(X)}{\frac{K_L}{L}\log m} \to 1$$

        - 但要在实际中实现，L必须取无限长的信源符号进行统一编码。这样做实际上是不可能的，因L非常大，无法实现。
        - 当L有限时，要做到高编码效率、低错误率，对于定长编码来说是不可能做到的。

变长编码

变长编码的基本概念

- 在变长编码中，码长K_i是变化的。
- m进制平均码长：$\overline{K'}=\sum_i p(a_i)K_i$
- 根据信源各个符号的统计特性，如概率大的符号用短码，概率小的用较长的码，使得编码后平均码长降低，从而提高编码效率。（统计匹配）

单个符号变长编码定理

- **定理**：若离散无记忆信源的符号熵为 $H(X)$，每个信源符号用m进制码元进行变长编码，一定存在一种无失真编码方法，其(m进制)码字平均长度 $\overline{K'}$ 满足下列不等式

$$\frac{H(X)}{\log m}\leq\overline{K'}<\frac{H(X)}{\log m}+1$$

- **证明**：见**无失真信源编码定理**

离散平稳无记忆序列变长编码定理(香农第一定理)

- 由单个符号变长编码定理推广而来：

$$\begin{aligned}\frac{H(X)}{\log m}\leq&\overline{K'}<\frac{H(X)}{\log m}+1\\\Rightarrow\frac{LH_L(X)}{\log m}\leq&\overline{K_L}<\frac{LH_L(X)}{\log m}+1\\\Rightarrow H_L(X)\leq&\overline{K}<H_L(X)+\frac{\log m}{L}\end{aligned}$$

- **定理**：对于离散平稳无记忆信源，必存在一种无失真编码方法，使平均信息率 $\overline{K}$ 满足不等式

$$H_L(X)\leq\overline{K}<H_L(X)+\frac{\log m}{L}$$

  其中$\overline{K}=\frac{\overline{K_L}}{L}\log m$，当L足够大时，可使 $\frac{\log m}{L}<\varepsilon$，得到

$$H_L(X)\leq\overline{K}<H_L(X)+\varepsilon$$

  其中ε是任意小的正数。
- **证明**：见**无失真信源编码定理**
- **变长编码效率**
  变长编码效率的**下界**：

$$\eta = \frac{H_L(X)}{\overline{K}} > \frac{H_L(X)}{H_L(X) + \frac{\log m}{L}}$$

无失真编码效率总是小于1，可以用它来衡量各种编码方法的优劣。

为了衡量各种编码方法与最佳码的差距，定义**码的剩余度**为：

$$\gamma = 1 - \eta = 1 - \frac{H_L(X)}{\frac{\overline{K_L}}{L}\log m} = 1 - \frac{H_L(X)}{\overline{K}}$$

对某一信源和某一码符号集，若有一个唯一可译码，其平均长度小于所有其他唯一可译码的平均长度，则称该码为**最佳码**或**紧致码**。

香农编码

- 香农第一定理指出了平均码长与信源熵之间的关系，同时也指出了可以通过编码使平均码长达到极限值，这是一个很重要的极限定理。
- 香农第一定理指出，选择每个码字的长度l_i满足下式：

$$l_i = \left\lceil \log \frac{1}{p(x_i)} \right\rceil \text{（向上取整）}$$

或：

$$I(x_i) \le l_i < I(x_i) + 1$$

这种编码方法称为香农编码

- **编码步骤**

i. 将信源消息符号按其出现的概率大小依次排列：

$$p_1 \ge p_2 \ge \cdots \ge p_n$$

ii. 依照下列不等式确定整数的码长K_i：

$$-\log_2(p_i) \le l_i < -\log_2(p_i) + 1$$

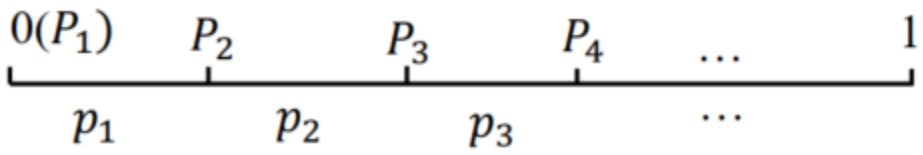
iii. 为了编成唯一可译码，计算第i个消息的累加概率：

$$P_i = \sum_{k=1}^{i-1} p(a_k)$$

iv. 将累加概率P_i变换成二进制数 。

v. 取P_i二进制数的小数点后l_i位即为该消息符号的二进制码字 。

- **香农编码图示**

- 累加概率P_i把区间$[0,1)$分割成许多小区间，每个小区间的长度等于各符号的概率p_i，小区间内的任一点可用来代表该符号： $P_i = \sum_{k=1}^{i-1} p(a_k)$
- 

- **示例**:

例5-4

设信源共7个符号，其概率和累加概率如下表所示

信源符号 a_i	符号概率 $p(a_i)$	累加概率 P_i	-log $p(a_i)$	码字长度 K_i	码 字
a_1	0.20	0	2.32	3	000
a_2	0.19	0.2	2.39	3	001
a_3	0.18	0.39	2.47	3	011
a_4	0.17	0.57	2.56	3	100
a_5	0.15	0.74	2.74	3	101
a_6	0.10	0.89	3.32	4	1110
a_7	0.01	0.99	6.64	7	1111110

$$-\log 0.17 \le K_4 < -\log 0.17 + 1$$
$$2.56 \le K_4 < 3.56, K_4 = 3$$
$$P_4 = 0.57_D = 0.1001\ldots_B$$

5.3 限失真信源编码定理

无失真与有失真信源编码

- 无失真信源编码是**保熵的**：通过信道的信息传输率R等于信源熵$H(X)$。
- 有失真信源编码属**熵压缩编码**，即编码后的信息率得到压缩。
- 采用有失真的熵压缩编码的原因：
  - 保熵编码并非总是必需的；
  - 保熵编码并非总是可能的；
  - 降低信息率有利于传输和处理。

限失真信源编码定理

- 信息率失真函数给出了失真小于D时所必须具有的最小信息率$R(D)$；只要信息率大于$R(D)$，一定可以找到一种编码，使译码后的失真小于D。
- **限失真信源编码定理**：设**离散无记忆信源**X的信息率失真函数为$R(D)$，则当信息率$R > R(D)$，只要信源序列长度L足够长，一定存在一种编码方法，其译码失真小于或等于$D+\varepsilon$，ε为任意小的正数。反之，若$R < R(D)$，则无论采用什么样的编码方法，其译码失真必大于D。
- 二元信源编码：
  - 对于任意小的ε，每一个信源符号的平均码长满足如下公式：

$$R(D) \leq \overline{K} < R(D) + \varepsilon$$

    在失真限度内使信息率任意接近$R(D)$的编码方法是存在的。然而，如果使信息率小于$R(D)$，平均失真一定会超过失真限度D。
- 对于**连续**平稳无记忆信源，无法进行无失真编码，在**限失真**情况下，有与上述定理一样的编码定理。
- 限失真信源编码定理只能说明**最佳编码是存在的**，而具体构造编码方法却一无所知。因而就不能象无失真编码那样从证明过程中引出概率匹配的编码方法。一般只能从优化的思路去求最佳编码。实际上迄今尚无合适的可实现的编码方法可接近$R(D)$这个界。

5.4 常用信源编码方法简介

变长码与存储器容量

- T秒内有N个信源符号输出，信源输出符号速率$S = N/T$，若符号的平均码长为$\overline{K}$，则信道**传输速率**需要

$$R_t = S\overline{K}$$

- N个码字的长度分别为$K_i, i = 1, \ldots, N$，即在此期间输入存储器$\sum_{i=1}^{N} K_i$ bit，输出信道$R_t T$ bit，则在**存储器里还剩**：

$$X = \sum_{i=1}^{N} K_i - R_t T$$

- 已知K_i是随机变量，其均值和方差为：

$$\overline{K} = E[K_i] = \sum_{j=1}^{m} p_j K_j$$

$$\sigma^2 = E[K_i^2] - \overline{K}^2 = \sum_{j=1}^{m} p_j K_j^2 - \overline{K}^2$$

式中m为信源符号集的元数。当N足够大时，X是许多独立同分布的随机变量之和，它近似于正态分布

$$p_X(x) = \frac{1}{\sqrt{2\pi}\sigma_X} e^{-\frac{(x-E[X])^2}{2\sigma_X^2}}$$

其均值和方差分别为：

$$E[X] = N\overline{K} - R_t T$$

$$\sigma_X^2 = N\sigma^2$$

- 若信道速率满足$R_t = S\overline{K}$，$E[X] = 0$。假设存储器容量为$2A\sigma_X$，起始时存储器为半满，则**溢出概率**为：

  $$P(X > A\sigma_X) = \varphi(-A)$$

  取空概率为：

  $$P(X < -A\sigma_X) = \varphi(-A)$$

- 若要求溢出和取空概率$P_e = 0.001$，查表得$A = 3.08$，则存储器容量为：

  $$C > 2A\sigma_X = 2A\sqrt{N}\sigma = 6.16\sqrt{N}\sigma$$

  - 码方差σ越大，要求存储器的容量也越大。
  - 时间越长，N越大，要求存储器的容量也越大。
  - 存储器容量设定后，随着时间的增长，存储器溢出和取空的概率都将增大。
- 一般来说，变长码只适用于有限长的信息传输（如传真）。实际使用时，可把长信息分段发送，也可以实时检测存储器的状态，调整输出。

分组码

哈夫曼(Huffman)编码

- **特点**
  - 哈夫曼（Huffman）编码是分组码。
  - 依据各符号出现的**概率**来构造码字。
  - 基于二叉树的编码思想，所有可能的符号在哈夫曼树上对应为一个节点，节点的位置就是该符号的码字。这些节点都是终极节点，不再延伸，不会出现前缀码，因而唯一可译。
  - 哈夫曼编码是一种效率比较高的**变长无失真信源编码**方法。

- **编码步骤**：
  i. 将信源消息符号按其出现的概率大小依次排列

  $$p_1 \geq p_2 \geq \cdots \geq p_n$$

  ii. 取两个概率最小的符号分别配以0和1两个码元，并将这两个概率相加作为一个新符号的概率，与未分配的二进符号的符号重新排队。
  iii. 对重排后的两个概率最小符号重复步骤 2 的过程。
  iv. 不断继续上述过程，直到最后两个符号配以0和1为止。
  v. 从最后一级开始，**向前**返回得到各个信源符号所对应的码元序列，即相应的码字。
- 哈夫曼编码方法得到的码**并非唯一**的。
  - 每次对信源缩减时，赋予信源最后两个概率最小的符号时，0和1是可以任意的，所以可以得到不同的哈夫曼码，但不会影响码字的长度。
  - 对信源进行缩减时，若两个概率最小的符号合并后的概率与其它信源符号的概率相同时，这两者在缩减信源中进行概率排序，其位置放置次序是可以任意的，故会得到不同的哈夫曼码。此时将影响码字的长度，一般将**合并的概率**放在靠前位置，这样可获得**较小的码方差**。
  - 不同的编法得到的码字长度K_i也不尽相同。
- 单符号信源编二进制哈夫曼码，编码效率主要决定于信源熵和平均码长之比。
  - 对相同的信源编码，其熵是一样的，采用不同的编法得到的平均码长可能不同。
  - 平均码长越短，编码效率就越高。
- **码方差**：

$$\sigma_I^2 = E\left[\left(K_i - \overline{K}\right)^2\right] = \sum_{i=1}^{q} p(a_i)(K_i - \overline{K})^2$$

- **N进制哈夫曼编码**：
  - 在N进制哈夫曼编码中，为了得到最短平均码长，有时需要对信源符号作添加，使信源符号数量满足

  $$N + k(N-1) \quad k \in N$$

  - 平均二进制码长(信息率)为

  $$\overline{K} = \frac{K_L}{L}\log_2 N$$

- **用哈夫曼方法对信源序列编码**
  - 随着序列长度L的增加，平均（二进制）码长迅速降低，接近信源熵值。
  - 例子：信源输出两个符号，概率分布为$P = (0.9, 0.1)$，信源熵$H(X) = 0.469$比特/符号。采用二进制哈夫曼编码。
    - $L = 1, \overline{K} = 1$ bit/符号

- $L=2, P'=(0.81,0.09,0.09,0.01), \overline{K}=0.645$ bit/符号
- $L=3, \overline{K}=0.533$ bit/符号
- $L=4, \overline{K}=0.494$ bit/符号

费诺(Fano)编码

- **特点**:
  - 费诺（Fano）编码方法属于概率匹配编码
  - 费诺码的编码方法实际上是一种构造码树的方法，所以费诺码是即时码 。
  - 费诺码考虑了信源的统计特性，使概率大的信源符号能对应码长短的码字，从而有效地提高了编码效率。
  - 但是费诺编码方法不一定使短码得到充分利用。尤其当信源符号较多时，若有一些符号概率分布很接近时，分两大组的组合方法会很多。可能某种分大组的结果，会使后面小组的“概率和”相差较远，使平均码长增加。
- **编码步骤**:
  i. 将信源消息符号按其出现的概率大小依次排列

$$p_1 \geq p_2 \geq \cdots \geq p_n$$

  ii. 将依次排列的信源符号按概率值分为两大组，使两个组的概率之和近于相同，并对各组赋予一个二进制码元“0”和“1”；
  iii. 将每一大组的信源符号进一步再分成两组，使划分后的两个组的概率之和近于相同，并又赋予两个组一个二进制符号“0”和“1”。
  iv. 如此重复，直至每个组只剩下一个信源符号为止。
  v. 信源符号所对应的码字即为费诺码。

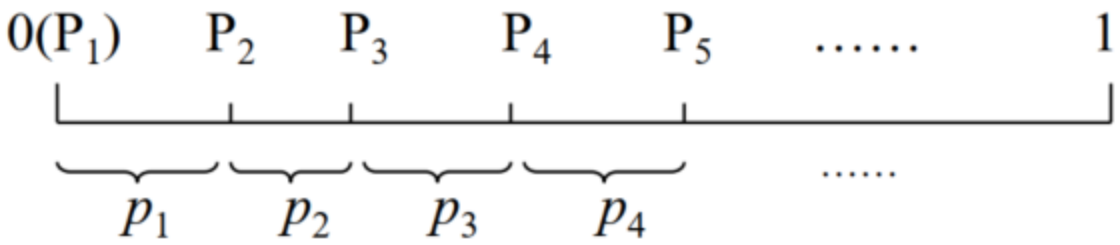
分组码的局限性

- 概率特性必须得到精确的测定，若其略有变化，还需更换码表。
- 对于二元信源，常常需要将多个符号合起来编码才能取得较好的效果。
- 当合并的符号数不大时，编码效率提高不明显，尤其是对于相关信源，结果难以令人满意。
- 而合并的符号数增大时，码表中的码字数增多，会导致设备越来越复杂 。

非分组码

算术编码

- **特点**:
  - 算术码是一种**非分组码**，其基本思路是:

    - 积累概率$P(S)$把区间$[0,1)$分割成许多小区间，每个小区间的长度等于各序列的概率$p(S)$，小区间内的任一点可用来代表这序列。
    - 

  - **与香农码的区别**：香农码考虑单个符号，算术编码考虑的是整个数据文件。
- **求序列的概率$p(S)$和累计概率$P(S)$**：
  - 信源符号集$A=\{a_1,a_2,\cdots,a_n\}$，L长信源序列共有n^L种可能序列，因为序列长度L很大，很难得到对应序列的概率和累积概率，只能从已知的信源符号概率中递推得到。
  - 信源符号概率分布为：

$$P=[p(a_1),p(a_2),\cdots,p(a_n)]=[p_1,p_2,\cdots,p_n]$$

  - 定义各符号的累积概率为：

$$P_r=\sum_{i=1}^{r-1}p_i \quad p_r=P_{r+1}-P_r$$

- **累积概率递推公式**
  - **二元序列**：
    有二元符号序列$S=011$，把3个二元符号的序列按自然二进制数排列，
    $000,001,010,011,\cdots$，则S的累积概率为：

$$P(S)=p(000)+p(001)+p(010)$$

    如果S后面接一个“0”，累积概率就成为：

$$\begin{aligned}P(S,0)=&P(0110)\\=&p(0000)+p(0001)+p(0010)+\\&p(0011)+p(0100)+p(0101)\end{aligned}$$

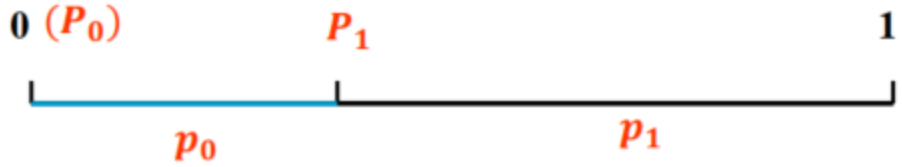
    由归一律，$p(0000)+p(0001)=p(000)$等，得：

$$P(S,0)=P(0110)=p(000)+p(001)+p(010)=P(S)$$

    同理，

$$
\begin{aligned}
P(S,1) =& P(0111) \\
=& p(0000)+p(0001)+p(0010)+ \\
& p(0011)+p(0100)+p(0101)+p(0110) \\
=& P(S)+p(0110) \\
=& P(S)+p(S)p_0
\end{aligned}
$$

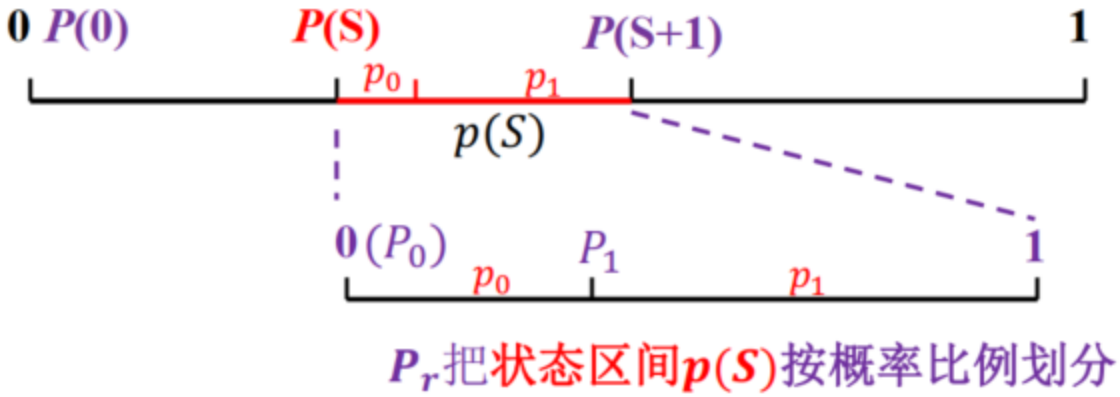
当$A=\{0,1\}$，即**二元信源**时：$P_0=0$；$P_1=p_0$



$$
\begin{aligned}
P(S,0) &= P(S) = P(S)+p(S)P_0 \\
P(S,1) &= P(S)+p(S)p_0 = P(S)+p(S)P_1
\end{aligned}
$$

统一写作

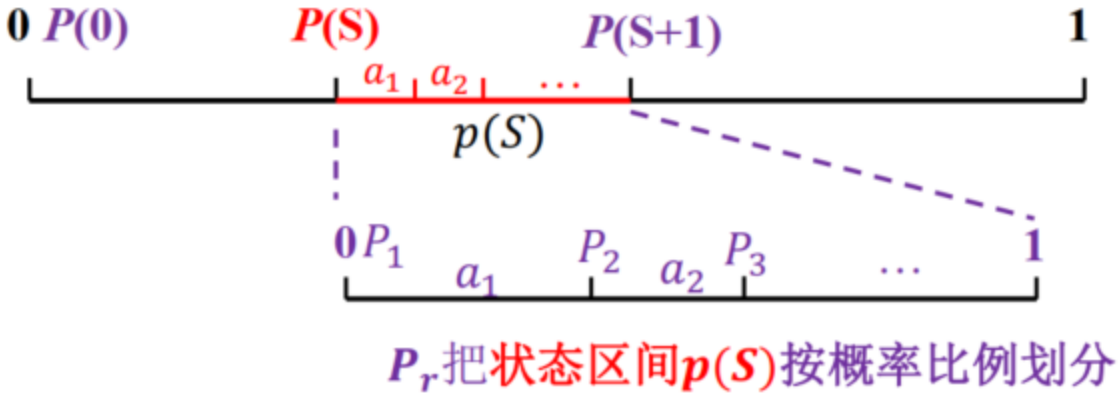
$$
P(S,r)=P(S)+p(S)P_r \quad r=0,1
$$



- **一般递推公式**

  推广到多元序列，即可得到一般的递推公式：

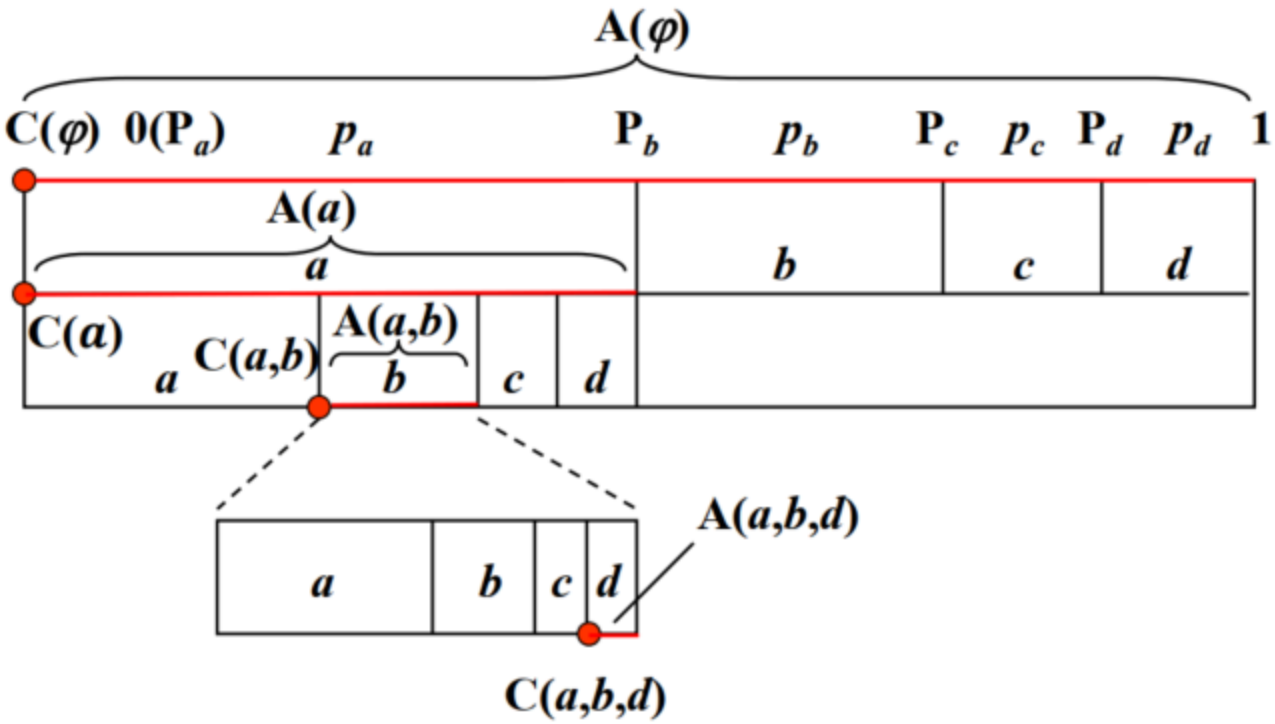
$$
\begin{aligned}
P(S,a_r) =& P(S)+p(S)P_r \\
p(S,a_r) =& p(S)p_r
\end{aligned}
$$



  - 随着序列的长度不断增加，$p(S)$越来越小，$P(S)$所在区间的长度就越短，也就可以更加精确地确定$P(S)$的位置。

- **编码过程**：
  - 实际应用中，**用码字$C(S)$表示累积概率$P(S)$，用状态区间$A(S)$表示序列的概率$p(S)$**，则递推公式为：

$$\begin{cases} C(S,r) = C(S) + A(S)P_r \\ A(S,r) = A(S)p_r \end{cases}$$



i. 设置两个存储器，起始时令$A(\varphi) = 1, C(\varphi) = 0$，$\varphi$表示空集，即起始时码字为0，状态区间为1。

ii. 每输入一个信源符号，存储器C和A就按照递推公式更新一次，直至信源符号输入完毕。

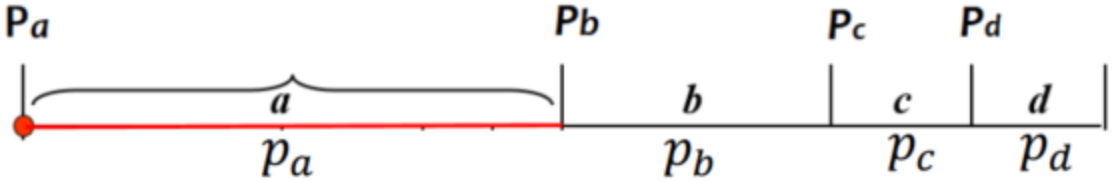
iii. 确定码长

$$L = \left\lceil \log \frac{1}{A(S)} \right\rceil$$

其中$\lceil x \rceil$代表大于或等于x的最小整数。

iv. 取存储器C小数点后L位的内容作为该序列的码字输出；如果有尾数，就进位到第L位，这样得到一个数C。

- **译码过程**：
  i. 判断接收到的码字 $C(S)$ 落在哪个初始概率区间，从而确定第一个符号。
  ii. 从 $C(S)$ 中减去对应首个符号的累积概率 P_r 。
  iii. 将相减后的数值乘以当前符号概率 p_r 的倒数，放大至 $[0,1]$ 区间，以确定下一个符号所在的概率区间，进而确定下一个符号。
  iv. 不断重复上述去掉累积概率、区间放大和确定符号的步骤，直到处理完整个码字，译出完整的符号序列。
- **示例**：有四个符号a，b，c，d构成简单序列$S = abda$，各符号及其对应概率如下表



符号	**符号概率p_i**	**符号累积概率P_j**
a	$0.100(1/2)$	0.000
b	$0.010(1/4)$	0.100
c	$0.001(1/8)$	0.110
d	$0.001(1/8)$	0.111

  - **算术编码过程如下**：
    a. 设起始状态为空序列φ，则$A(\varphi) = 1$，$C(\varphi) = 0$。
    b. 计算$C(\varphi a)$和$A(\varphi a)$：

$$\begin{cases} C(\varphi a) = C(\varphi) + A(\varphi)P_a = 0 + 1 \times 0 = 0 \\ A(\varphi a) = A(\varphi)p_a = 1 \times 0.1 = 0.1 \end{cases}$$

    c. 计算$C(ab)$和$A(ab)$：

$$\begin{cases} C(ab) = C(a) + A(a)P_b = 0 + 0.1 \times 0.1 = 0.01 \\ A(ab) = A(a)p_b = 0.1 \times 0.01 = 0.001 \end{cases}$$

d. 计算$C(abd)$和$A(abd)$:

$$\begin{cases} C(abd) = C(ab) + A(ab)P_d = 0.01 + 0.001 \times 0.111 = 0.010111 \\ A(abd) = A(ab)p_d = 0.001 \times 0.001 = 0.000001 \end{cases}$$

e. 计算$C(abda)$和$A(abda)$:

$$\begin{cases} C(abda) = C(abd) + A(abd)P_a = 0.010111 + 0.000001 \times 0 = 0.010111 \\ A(abda) = A(abd)p_a = 0.000001 \times 0.1 = 0.0000001 \end{cases}$$

f. 计算编码长度L:

$$L = \left\lceil \log \frac{1}{A(abcd)} \right\rceil = 7$$

取$C(abda)$的小数点后7位即为编码后的码字0101110。

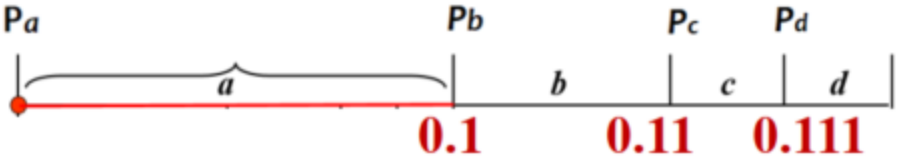
- 该信源的熵为:

$$H(X) = \frac{1}{2}\log 2 + \frac{1}{4}\log 4 + 2 \times \frac{1}{8}\log 8 = 1.75\ bit/符号$$

编码效率

$$\eta = \frac{1.75}{\frac{7}{4}} = 100\%$$

- **译码过程如下**:



a. 已知编码后的数值$C(S) = 0.010111$，因为$C(S) = 0.010111 \in [0, 0.1)$，根据符号累积概率表，第一个符号为$a$。

b. 去掉累积概率P_a: $0.010111 - 0 = 0.010111$,
放大至$[0,1]$（乘以p_a^{-1}）: $0.010111 \times 2^1 = 0.10111 \in [0, 0.110)$
第二个符号为b。

c. 去掉累积概率P_b: $0.10111 - 0.1 = 0.00111$。
放大至$[0,1]$（乘以p_b^{-1}）: $0.00111 \times 2^2 = 0.111 \in [0.111, 1)$
第三个符号为d。

d. 去掉累积概率P_d: $0.111 - 0.111 = 0$。
放大至$[0,1]$（乘以p_d^{-1}）: $0 \times 2^4 = 0 \in [0, 0.1)$

第四个符号为a。

LZ编码

- **特点**：
  - LZ编码是一种**字典编码**，无需确定信源的统计特性。
- **LZ78 编码步骤**
  - 设信源符号集$A=(a_1,a_2,\cdots,a_K)$共K个符号，输入信源符号序列$U=(u_1,u_2,\cdots,u_L)$，编码是将此序列分成不同的段。
  - **分段规则**：尽可能取最少个相连的信源符号，并保证各段都不相同。
    a. 在第i步，从s_{i-1}短语后的第一个符号开始向后搜索此前未出现过的最短短语s_i，将该短语添入字典第i段。
    b. 假设s_i去掉最后一个符号x后所得的前缀是在第j步出现的短语。
    c. 对s_i的编码为(j,x)。**对段号j，用$\lceil\log i\rceil$比特来表示，符号x用$\lceil\log K\rceil$比特来表示。**
- **LZ译码**：无需接收方提前知晓字典内容，通过码字和逐步构建的字典就能完成译码
  i. 接收到码字(j,x)后，在已建立或正在建立的字典中找到第j个短语。
  ii. 将符号x添加到找到的第j个短语后，形成新的短语。
  iii. 把新生成的短语添入字典，以便后续译码使用。
  iv. 对每个接收到的码字，不断重复上述步骤，直至完成所有码字的译码。
- **示例**：
  - **编码**：设$U=\{a_1,a_2,a_3,a_4\}$，信源序列为$a_1,a_2,a_1,a_3,a_2,a_4,a_2,a_4,a_3,a_1,a_1,a_4\cdots$，按照分段规则进行分段。符号编码表如下:

a_1	a_2	a_3	a_4
00	01	01	10

段号i	**短语**	j	x	**码字**	**编码**
1	a_1	0	a_1	$(0,a_1)$	000
2	a_2	0	a_2	$(0,a_2)$	001
3	a_1a_3	1	a_3	$(1,a_3)$	01,10
4	a_2a_4	2	a_4	$(2,a_4)$	10,11
5	$a_2a_4a_3$	4	a_3	$(4,a_3)$	100,10
6	a_1a_1	1	a_1	$(1,a_1)$	001,00
7	a_4	0	a_4	$(0,a_4)$	000,11

- **译码**：码序列为00000101101011100100010000011 。符号编码表如下：

段号i	编码	j	x	码字	短语
1	0,00	0	a_1	$(0,a_1)$	a_1
2	0,01	0	a_2	$(0,a_2)$	a_2
3	01,10	1	a_3	$(1,a_3)$	a_1a_3
4	10,11	2	a_4	$(2,a_4)$	a_2a_4
5	100,10	4	a_3	$(4,a_3)$	$a_2a_4a_3$
6	001,00	1	a_1	$(1,a_1)$	a_1a_1
7	000,11	0	a_4	$(0,a_4)$	a_4

游程编码(RLE，run-length encoding)

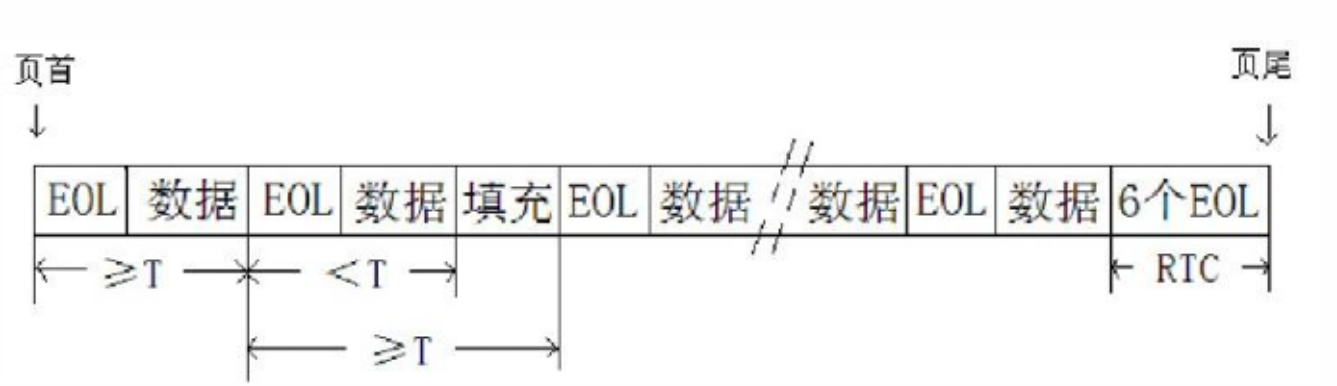
- 在二元序列中，连0段称为0游程，连1段称为1游程。
- 例如二元码序列：$000101110010001\cdots$，可变换成下列游程序列：$31132131\cdots$ 。
- 若已知二元序列以0起始，从游程序列很容易恢复成原来的二元序列。
- 游程序列是多元序列，各长度可按哈夫曼编码或其它方法处理以达到压缩码率的目的。

MH 编码(传真编码)

- **MH编码方法**
  i. 黑白游程分别对应不同的编码表。
  ii. 游程长度在$0\sim63$时，码字直接用相应的终端码（结尾码）表示。
    - 例如：
      - 白游程长度为0，查表得码字000 。
      - 黑游程长度为24 ，查表得码字11111 。
  iii. 游程长度在$64\sim1728$ ，用一个组合码加上一个结尾码为相应码字。
    - 例如：
      - 白游程长度为$65(=64+1)$，查表得码字$11011\mid000111$ 。
      - 黑游程长度为$856(=832+24)$ ，查表得码字$0000001001101\mid00000010111$ 。
  iv. 规定每行都从白游程开始，若实际出现黑游程开始的话，则在行首加上零长度白游程码字，每行结束用一个结束码（EOL）。
  v. 每行恢复成1728个像素，否则有错。
  vi. 每页文件开始第一个数据前加一个结束码。每页尾连续使用6个结束码表示结尾。

vii. 为了传输时实现同步操作，规定T为每编码行的最小传输时间。一般规定T最小为$20ms$，最大为$5s$。若编码行传输时间小于T，则在结束码之前填以足够的“0”码元（称填充码）。

- 页面数据结构



- 结尾码码表

MH码表（一）结尾码

RL长度	白游程码字	黑游程码字	RL长度	白游程码字	黑游程码字	RL长度	白游程码字	黑游程码字
0	00110101	0000110111	21	001011	000001101100	42	00101011	000011011010
1	000111	010	22	0000011	000000110111	43	00101100	000011011011
2	0111	11	23	0000100	000000101000	44	00101101	000001010100
3	1000	10	24	0101000	000000010111	45	00000100	000001010101
4	1011	011	25	0101011	000000011000	46	00000101	000001010110
5	1100	0011	26	0010011	000011001010	47	00001010	000001010111
6	1110	0010	27	0100100	000011001011	48	00001011	000001100100
7	1111	00011	28	0011000	000011001100	49	01010010	000001100101
8	10011	000101	29	00000010	000011001101	50	01010011	000001010010
9	10100	000100	30	00000011	000001101000	51	01010100	000001010011
10	00111	0000100	31	00011010	000001101001	52	01010101	000000100100
11	01000	0000101	32	00011011	000001101010	53	00100100	000000110111
12	001000	0000111	33	00010010	000001101011	54	00100101	000000111000
13	000011	00000100	34	00010011	000011010010	55	01011000	000000100111
14	110100	00000111	35	00010100	000011010011	56	01011001	000000101000
15	110101	000011000	36	00010101	000011010100	57	01011010	000001011000
16	101010	0000010111	37	00010110	000011010101	58	01011011	000001011001
17	101011	0000011000	38	00010111	000011010110	59	01001010	000000101011
18	0100111	0000001000	39	00101000	000011010111	60	01001011	000000101100
19	0001100	00001100111	40	00101001	000001101100	61	00110010	000001011010
20	0001000	00001101000	41	00101010	000001101101	62	00110011	000001100110
						63	00110100	000001100111

- 组合基干码码表

MH码表（二）组合基干码

RL长度	白游程码字	黑游程码字	RL长度	白游程码字	黑游程码字
64	11011	0000001111	960	011010100	000000111001
128	10010	000011001000	1024	011010101	0000001110100
192	010111	000011001001	1088	011010110	0000001110101
256	0110111	000001011011	1152	011010111	0000001110110
320	00110110	000000110011	1216	011011000	0000001110111
384	00110111	000000110100	1280	011011001	0000001010010
448	01100100	000000110101	1344	011011010	0000001010011
512	01100101	0000001101100	1408	011011011	0000001010100
576	01101000	0000001101101	1472	010011000	0000001010101
640	01100111	0000001001010	1536	010011001	0000001011010
704	011001100	0000001001011	1600	010011010	0000001011011
768	011001101	0000001001100	1664	011000	0000001100100
832	011010010	0000001001101	1728	010011011	0000001100101
896	011010011	0000001110010	EOL	000000000001	000000000001