

# 多媒体技术基础

## 第三章 多媒体数据压缩

### § 3.1 无损数据压缩



2022年10月19日





# 关于课程课件和实验安排

◆ 课件, <http://www.bb.ustc.edu.cn/>



## ◆ 实验

- 选做20学时实验项目
- 课后自行完成, 通过BB系统**作业区**提交:  
1份**实验报告**文件+1个**代码压缩文件**。





# 授课内容

- ◆ 第一部分 多媒体的计算
  - 第一章 多媒体计算机系统
  - 第二章 媒体处理技术
  - 第三章 多媒体数据压缩
- ◆ 第二部分 多媒体的存储
  - 第四章 多媒体数据的数字存储
- ◆ 第三部分 多媒体信息的分析与处理
  - 第五章 多媒体信息分析与处理
- ◆ 第四部分 多媒体网络应用
  - 第六章 实时多媒体通信





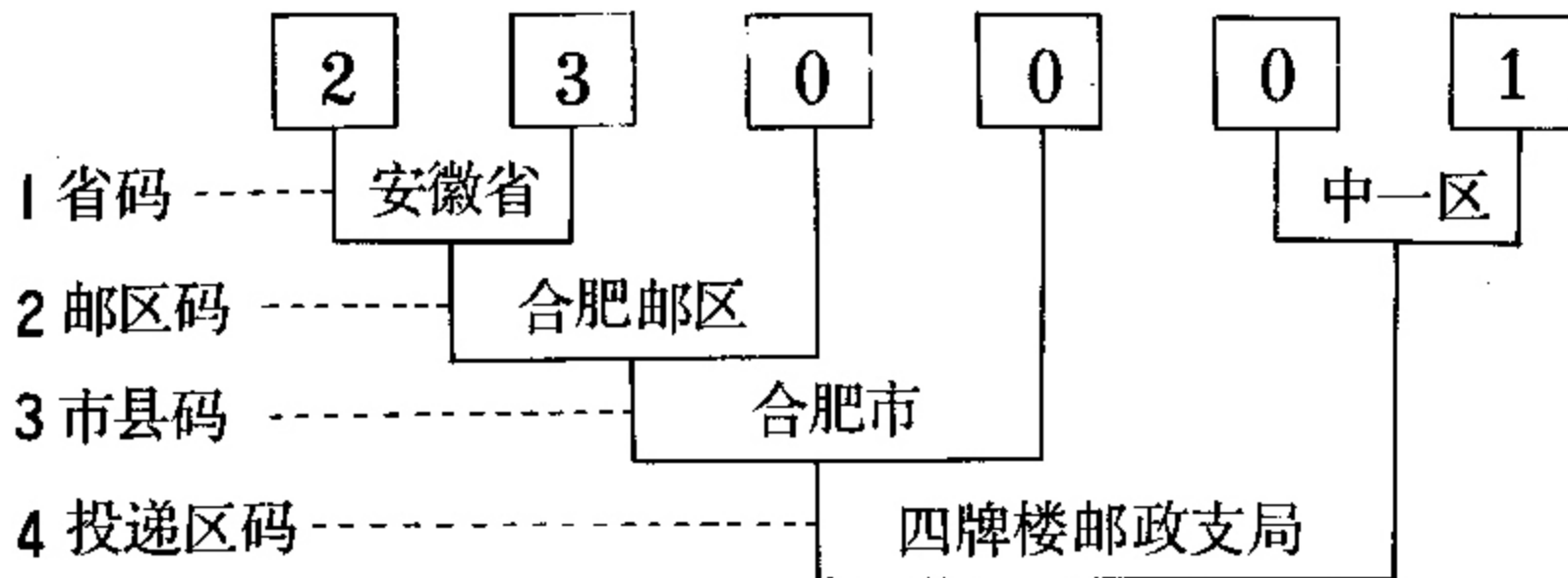
# 第三章 多媒体数据压缩

- ◆ § 3.1 无损数据压缩
  - § 3.1.1 编码与压缩基本概念
    - 编码
    - 字符编码
    - 压缩
  - § 3.1.2 压缩算法分类及评价原则
  - § 3.1.3 霍夫曼编码
  - § 3.1.4 算术编码
  - § 3.1.5 词典编码
  - § 3.1.6 RLE编码
- ◆ § 3.2 音频数据的压缩标准
- ◆ § 3.3 图像数据的压缩标准
- ◆ § 3.4 视频数据的压缩标准





# 什么是编码





# 什么是编码





# 什么是编码？

- ◆ **编码就是对应**。例如：把字符转换为比特串。
- ◆ 对数字信息的编码表现为从一个比特流转换为另一个比特流。
- ◆ 这种对应（或形式上的转换）具有不同的用途：**压缩**、加密、提高传输过程中的抗干扰能力、作为唯一的标识等等。

$\_$	$\leftrightarrow$	00000	,	$\leftrightarrow$	11011
$A$	$\leftrightarrow$	00001	.	$\leftrightarrow$	11100
$B$	$\leftrightarrow$	00002	?	$\leftrightarrow$	11101
$\vdots$	$\dots$		$\vdots$	$\vdots$	$\leftrightarrow$ 11110
$Z$	$\leftrightarrow$	11010	;	$\leftrightarrow$	11111







# Morse Code

Samuel Morse photo by student Mathew Brady, 1845

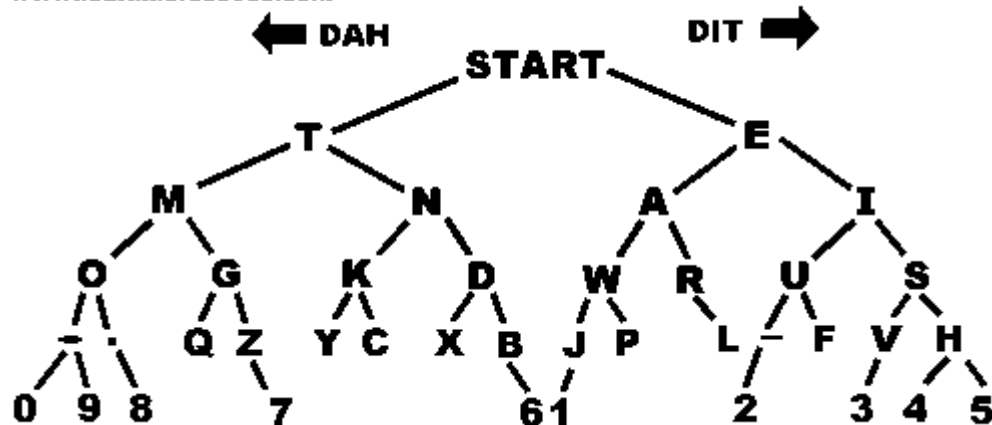
A .- I" Q ---- Y ----	1 .----
B -.. J ---- R .- Z ----	2 ..----
C -.- K -.- S... Period .----	3 ....----
D -.. L -.. T- Comma ----	4 ....----
E . M -- U... ? ..----	5 .....
F ... N - V... / ----	6 .....
G --- O --- W --- @' ----	7 .....
H .... P -.- X ----	8 .....
	9 .....
	0 ----

Samuel Morse, telegraph key circa 1860



An 1845 version of the finger key, which replaced type and portulac.

[www.learnmorsecode.com](http://www.learnmorsecode.com)







# 生活中的编码

- ◆ 身份证、电话区号和邮政编码
- ◆ 汉字的编码：GB2312、BIG5
- ◆ 网卡号、手机设备号、IP地址
- ◆ 文件的后缀名称
- ◆ 商品的一维条码
- ◆ 用于存储信息的二维条码
- ◆ RFID
- ◆ Windows操作系统里的“全球识别编码”



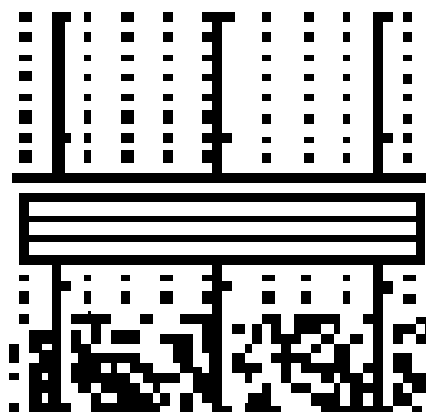


# 二维条码示例

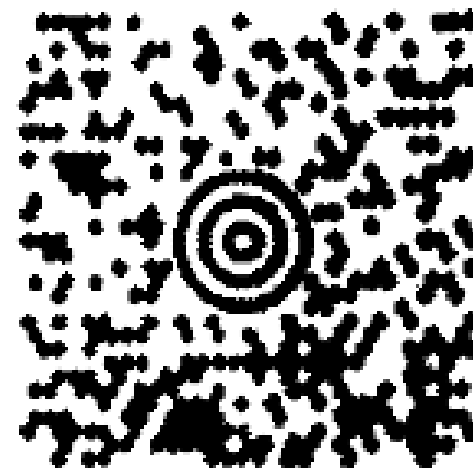
◆ 二维条码已研制出多种码制，常见的有PDF417，QR Code，Code 49，Code 16K，Code One等。



QR Code 条码



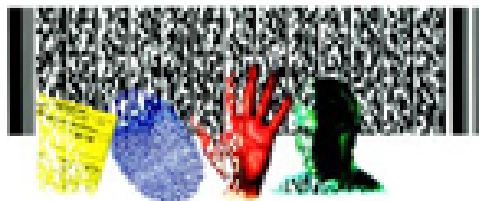
code-one



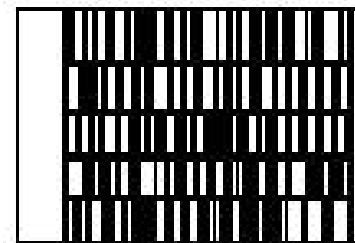
maxicode



code49



PDF417



Code 16K





# 二维码已成为常用的入口

本课程实验说明的URL





# 二维码常用于 物理世界与虚拟世界的链接

HD 1 4G+ 4G+ 65% 14:51

×

原酒标识卡

酒体存储信息 天佑德® 青稞酒

序号	名称	指标
1	酒体名称	优级有机原酒
2	酒体编码	Z8112110035H06/20110701
3	入库日期	20110701
4	仓库	原酒八库
5	库位	原酒八库35号库位
6	储存数量 (kg)	11008.12
7	原酒等级	优级
8	感官评语	测试





# RFID



含RFID标签的上海世博会门票



含RFID标签的图书

含RFID标签的运输工人身份识别卡



含RFID标签的北京奥运会门票



含RFID标签的巴西世界杯门票



RFID在身份证中的应用







# 多媒体数据的编码

- ◆ 首先需要表示媒体信息
- ◆ 在最大程度上压缩比特流
- ◆ 适合于在网络上传输





# 小结：编码

## ◆ 编码就是对应

- 一种信息形式对应为另一种信息形式
- 数字编码就是从一个比特流转换为另一个比特流

## ◆ 编码具有不同的用途

- 压缩
- 加密
- 抗干扰能力
- 作为唯一的标识
- .....

## ◆ 多媒体数据的编码

- 首先需要表示媒体信息
- 在最大程度上压缩比特流
- 适合于在网络上传输



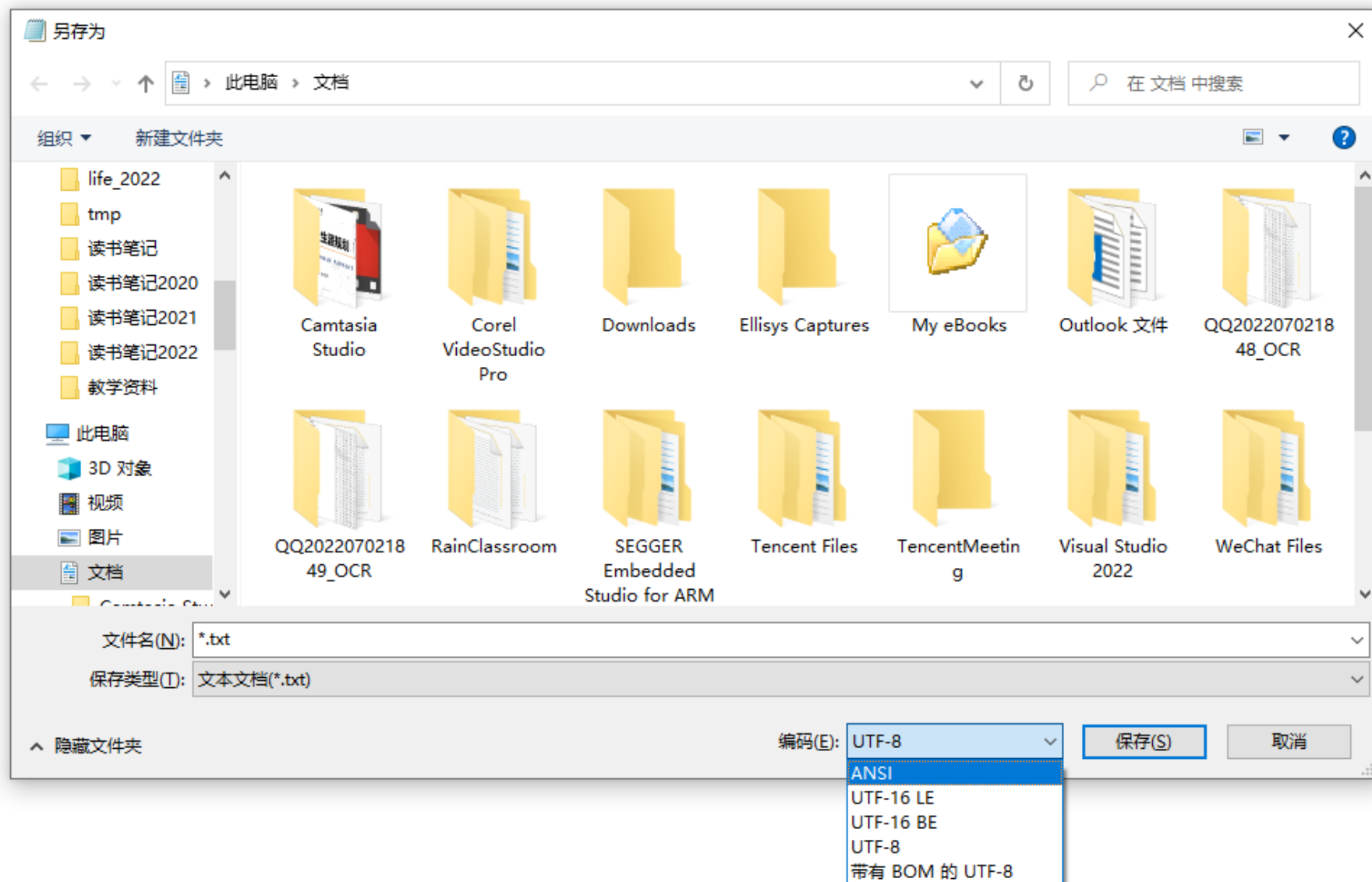


- ◆ § 3.1 无损数据压缩
  - § 3.1.1 编码与压缩基本概念
    - 编码
    - 字符编码
    - 压缩
  - § 3.1.2 压缩算法分类及评价原则
  - § 3.1.3 霍夫曼编码
  - § 3.1.4 算术编码
  - § 3.1.5 词典编码
  - § 3.1.6 RLE编码
- ◆ § 3.2 音频数据的压缩标准
- ◆ § 3.3 图像数据的压缩标准
- ◆ § 3.4 视频数据的压缩标准



# 字符编码

## 记事本程序的保存选项





# 字符编码

## IE浏览器的网页字符编码选项



- 阿拉伯语(ASMO 708)
- 阿拉伯语(DOS)
- 阿拉伯语(ISO)
- 阿拉伯语(Windows)
- 波罗的语(ISO)
- 波罗的语(Windows)
- 中欧(DOS)
- 中欧(ISO)
- 中欧(Windows)
- 简体中文(GB18030)
- 简体中文(HZ)
- 繁体中文(Big5)
- 西里尔文(DOS)
- 西里尔文(ISO)
- 西里尔文(KOI8-R)
- 西里尔文(KOI8-U)
- 西里尔文(Windows)
- 希腊语(ISO)
- 希腊语(Windows)
- 希伯来语(DOS)
- 希伯来语(ISO-逻辑)
- 希伯来语(ISO-视觉)
- 希伯来语(Windows)
- 日语(自动选择)
- 日语(EUC)
- 日语(Shift-JIS)
- 朝鲜语
- 泰语(Windows)
- 土耳其语(ISO)
- 土耳其语(Windows)
- 用户定义的
- 越南语(Windows)
- 西欧(ISO)
- 西欧(Windows)



# 字符编码

◆ 字符：计算机使用的字母、数字和符号

◆ 字符编码：用代码表示字符的方法

◆ 例如，英文字符的ASCII编码

□ ASCII=American Standard Code for Information Interchange

□ 当字节中的最高位b7=0时，其余7位(b6~b0) 产生0~127之间的128个字符的代码，按作用可分为

- 34个控制字符
- 10个阿拉伯数字
- 52个英文大小写字母
- 32个专用符号





# 7位ASCII码标准编码表

$\begin{matrix} D_6D_5D_4 \\ D_3D_2D_1D_0 \end{matrix}$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0 (30H)	@	P	,	p
0001	SOH	DC1	!	1	A(41H)	Q	a(61H)	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	—	=	M	]	m	}
1110	SO	RS	·	>	N	↑	n	~
1111	SI	US	/	?	O	_	o	DEL



# 汉字字符编码：GB 2312标准

- ◆ GB 2312标准是我国第一个汉字编码标准(1980)
- ◆ 收录字符数7445个 = 6763个汉字 + 682个全角字符
- ◆ 编码空间(codespace)或码位空间
  - 编码方法可表达的字符数目
  - 用两字节(16位)编码，最大编码空间为65 536个字符
  - 双字节字符集(double-byte character set, DBCS):  
用两字节(16位)代码表示字符构成的字符集
- ◆ GB 2312标准使用双字节的编码方法





# 汉字字符编码：UCS ISO/IEC 10646标准

- ◆ ISO/IEC 10646标准是ISO/IEC在1989年开始开发的单一字符集——“通用字符集(Universal Character Set, UCS)”字符编码标准
- ◆ 字符集：包含已知语言的所有字符以及大量的图形、印刷、数学和科学符号，内含
  - 拉丁语、希腊语、斯拉夫语、希伯来语、阿拉伯语、亚美尼亚语、格鲁吉亚语
  - 中、日、韩的象形文字，或称表意文字。
  - 古埃及的象形文字和不常见的汉字







# 汉字字符编码：Unicode

## ◆ Unicode标准

- Unicode是Unicode联盟开发的单一字符集。1988年开始开发，在1991年前后认识到，世界不需要两个不兼容的字符集。于是，开始合并双方的工作成果，从Unicode 2.0开始保持两个标准码表的兼容性，共同调整未来的扩展工作。现在两个项目仍然存在，并独立公布各自的标准。

## ◆ Unicode中的CJK汉字编码

- 2009年10月发布了Unicode 5.2.0版本，定义了107 361个字符的码位，用文字和符号构成的字符集或称字符块作为名称。Unicode 5.2.0定义的中日韩CJK统一汉字(CJK Unified Ideographs)总数为75 942个





# Unicode的历史核心版本

Title	Year	Month (Day)	ISBN
The Unicode Standard, Version 14.0	2021	September 14	978-1-936213-29-0
The Unicode Standard, Version 13.0	2020	March 10	978-1-936213-26-9
The Unicode Standard, Version 12.0	2019	March 5	978-1-936213-22-1
The Unicode Standard, Version 11.0	2018	June 5	978-1-936213-19-1
The Unicode Standard, Version 10.0	2017	June 20	978-1-936213-16-0
The Unicode Standard, Version 9.0	2016	July 19	978-1-936213-13-9
The Unicode Standard, Version 8.0	2015	August 20	978-1-936213-10-8
The Unicode Standard, Version 7.0	2014	October 8	978-1-936213-09-2
The Unicode Standard, Version 6.2	2012	November 16	978-1-936213-07-8
The Unicode Standard, Version 6.1	2012	April 23	978-1-936213-02-3
The Unicode Standard, Version 6.0	2011	February 18	978-1-936213-01-6
The Unicode Standard, Version 5.2	2009	December 23	978-1-936213-00-9



# 中日韩汉字Unicode编码表

一	丁	丂	七	乚	乚	丂	万	丈	三	上	下	丌	不	与	丂
4E00	4E01	4E02	4E03	4E04	4E05	4E06	4E07	4E08	4E09	4E0A	4E0B	4E0C	4E0D	4E0E	4E0F
丂	丑	刃	专	且	丕	世	卅	丘	丙	业	丛	东	丝	丞	丢
4E10	4E11	4E12	4E13	4E14	4E15	4E16	4E17	4E18	4E19	4E1A	4E1B	4E1C	4E1D	4E1E	4E1F
北	両	丢	𠂇	两	严	並	喪	丨	乚	个	丫	丩	中	𠂇	𠂇
4E20	4E21	4E22	4E23	4E24	4E25	4E26	4E27	4E28	4E29	4E2A	4E2B	4E2C	4E2D	4E2E	4E2F
丰	𠂇	串	弗	临	𠂇	、	、	丸	丹	为	主	井	丽	举	丩
4E30	4E31	4E32	4E33	4E34	4E35	4E36	4E37	4E38	4E39	4E3A	4E3B	4E3C	4E3D	4E3E	4E3F
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9F80	9F81	9F82	9F83	9F84	9F85	9F86	9F87	9F88	9F89	9F8A	9F8B	9F8C	9F8D	9F8E	9F8F
龐	龔	龔	龔	龔	龔	龔	龔	龔	龙	龚	龔	龔	龔	龔	龟
9F90	9F91	9F92	9F93	9F94	9F95	9F96	9F97	9F98	9F99	9F9A	9F9B	9F9C	9F9D	9F9E	9F9F
龔	龔	龔	龔	龔	龔	*									
9FA0	9FA1	9FA2	9FA3	9FA4	9FA5	9FA6	9FA7	9FA8	9FA9	9FAA	9FAB	9FAC	9FAD	9FAE	9FAF
9FB0	9FB1	9FB2	9FB3	9FB4	9FB5	9FB6	9FB7	9FB8	9FB9	9FBA	9FBB	9FBC	9FBD	9FBE	9FBF
9FC0	9FC1	9FC2	9FC3	9FC4	9FC5	9FC6	9FC7	9FC8	9FC9	9FCA	9FCB	9FCC	9FCD	9FCE	9FCF















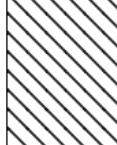














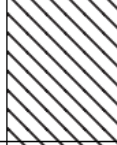














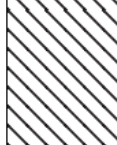







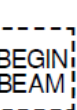







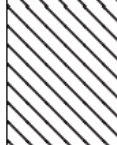







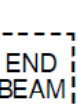







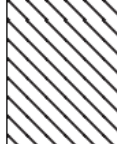







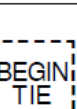







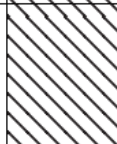




# 乐符(Musical Symbols)的Unicode码

## 1D100 ~ 1D1FF

<https://www.unicode.org/charts/PDF/U1D100.pdf>

	1D100	1D101	1D102	1D103	1D104	1D105	1D106	1D107	1D108	1D109	1D10A	1D10B	1D10C	1D10D	1D10E	1D10F
0										<i>m</i>						
1										<i>f</i>						
2											<i>c</i>					
3																
4																
5																



# 表情符号(Emoji)的Unicode码

<https://home.unicode.org/>



Adopt a Character



Emoji



Basic Info



News

Events

Connect



Membership



Press



U+1F499



U+03A6



U+3048



U+1F235



U+0E1D



U+0642



U+056F



U+1570



U+03C6



U+30FB



U+0908



U+1D17



U+201F



U+0E05



U+2193



U+1F606



U+03B3



U+3007

Everyone in the world should be able to use their own language on phones and computers.

[▶ LEARN MORE ABOUT UNICODE](#)



[ADOPT A CHARACTER](#)



U+20AC



U+0C66



U+1F617



U+2332



U+1F35F



U+2328



U+1F600



U+1F3EF



U+FF43



U+06A1



U+76CA



U+2021



U+3063



U+0D94



U+0434



U+2751



# Emoji Frequency



Adopt a Character +

Emoji +

About Emoji

Emoji Frequency

Basic Info +

News

Events

Connect +

Membership +

Press

THE UNICODE EMOJI MIRROR PROJECT

## The Most Frequently Used Emoji of 2021

By **Jennifer Daniel**, Unicode Emoji Subcommittee Chair



<https://home.unicode.org/emoji/emoji-frequency/>



# 汉字字符编码：UTF编码

## UTF-8、UTF-16、UTF-32

### ◆ 字符的编号

- Unicode 为世界上所有字符都分配了一个唯一的数字编号，这个编号范围从 0x000000 到 0x10FFFF (十六进制)，但没有规定这个编号如何存储。
- 实际系统中，编号到二进制的对应有多种方案：UTF-8、UTF-16、UTF-32。

### ◆ UTF，即Unicode转换格式(Unicode Translation Format)

### ◆ UTF-32用4个字节存储字符的编号

- 还需要考虑计算机的端模式（大端、小端）

### ◆ UTF-16使用变长字节表示字符的编号

- ① 对于编号在 U+0000 到 U+FFFF 的字符，直接用两个字节表示。
- ② 编号在 U+10000 到 U+10FFFF 之间的字符，用四个字节表示。

### ◆ UTF-8使用变长字节表示字符的编号

- ① 对于单字节的符号，字节的第一位设为 0，后面的7位为这个符号的 Unicode 码，因此对于英文字母，UTF-8 编码和 ASCII 码是相同的。
- ② 对于 n 字节的符号（ $n > 1$ ），第一个字节的前 n 位都设为 1，第 n+1 位设为 0，后面字节的前两位一律设为 10，剩下的位用来存储字符的 Unicode 码。







# UTF-8示例

## ◆ Unicode与UTF的转换示例

Unicode 码点	UTF-8 编码			
	字节 1	字节 2	字节 3	字节 4
0000~007F (128 个代码)	0xxxxxxx (ASCII 字符集, 最高位为 0)			
0080~07FF (1920 个代码)	110yyyyx (110 开始, 转换码为 2 字节)	10xxxxxx (10 开始)		
0800~FFFF (61440 个代码)	1110yyyy (1110 开始, 转换码为 3 字节)	10yyyyxx (10 开始)	10xxxxxx (10 开始)	
10000~10FFFF (1048576 个代码)	11110zzz (11110 开始, 转换码为 4 字节)	10zzyyyy (10 开始)	10yyyyxx (10 开始)	10xxxxxx (10 开始)

## ◆ 几个汉字的UTF-8编码结果

字符	Unicode	Bin	UTF-8 (Bin)	UTF-8 (H)
A	0041	01000001	01000001	41
©	00A9	10101001	11000010 10101001	C2A9
汉	6C49	01101100 01001001	11100110 10110001 10001001	E6B189
博	535A	01010011 01011010	11100101 10001101 10011010	E58D9A
岱	20C30	0 0010 0000 1100 0011 0000	11110000 10100000 10110000 10110000	F0A0B0B0





# 小结：字符编码

## ◆ 英文字符编码

- ASCII=American Standard Code for Information Interchange

## ◆ 汉字字符编码：GB 2312、BIG5

## ◆ 两套国际字符集标准合二为一

- 通用字符集(Universal Character Set, UCS)

- ISO/IEC 10646标准

- Unicode字符集

- Unicode联盟开发

## ◆ UTF (Unicode Translation Format)

- UTF-8、UTF-16、UTF-32



- ◆ § 3.1 无损数据压缩
  - § 3.1.1 编码与压缩基本概念
    - 编码
    - 字符编码
    - 压缩
  - § 3.1.2 压缩算法分类及评价原则
  - § 3.1.3 霍夫曼编码
  - § 3.1.4 算术编码
  - § 3.1.5 词典编码
  - § 3.1.6 RLE编码
- ◆ § 3.2 音频数据的压缩标准
- ◆ § 3.3 图像数据的压缩标准
- ◆ § 3.4 视频数据的压缩标准



# 什么是压缩？

## ◆两种表现形式

- 在等同的空间中容纳更多的信息
- 在等同的时间内表现更多的信息

## ◆比特流的压缩

- 用相同长度的比特流表示更多的信息
- 用尽量短的比特流表达尽量多的信息

## ◆压缩的目的

- 相同的代价处理更多的信息
- 处理等量信息的代价减小
- 提高媒体信息处理的效率
- 提高媒体信息处理的能力



# 在哪里能看到压缩？



寥寥数笔，勾勒出人物的动态



# 在哪里能看到压缩?





# 在哪里能看到压缩？

◆听的繁体字是听拆字就是“十目一心耳为王”也就是说要多听多看用心记，虚心听取别人的意见和合理的建议。







# 在哪里能看到压缩?

- ◆ 双人旁代表行为
- ◆ 十目是指十个方位的眼睛
- ◆ 一横代表天地
- ◆ 下面心字是思想



十只眼



直



心直



德



# 在哪里能看到压缩?





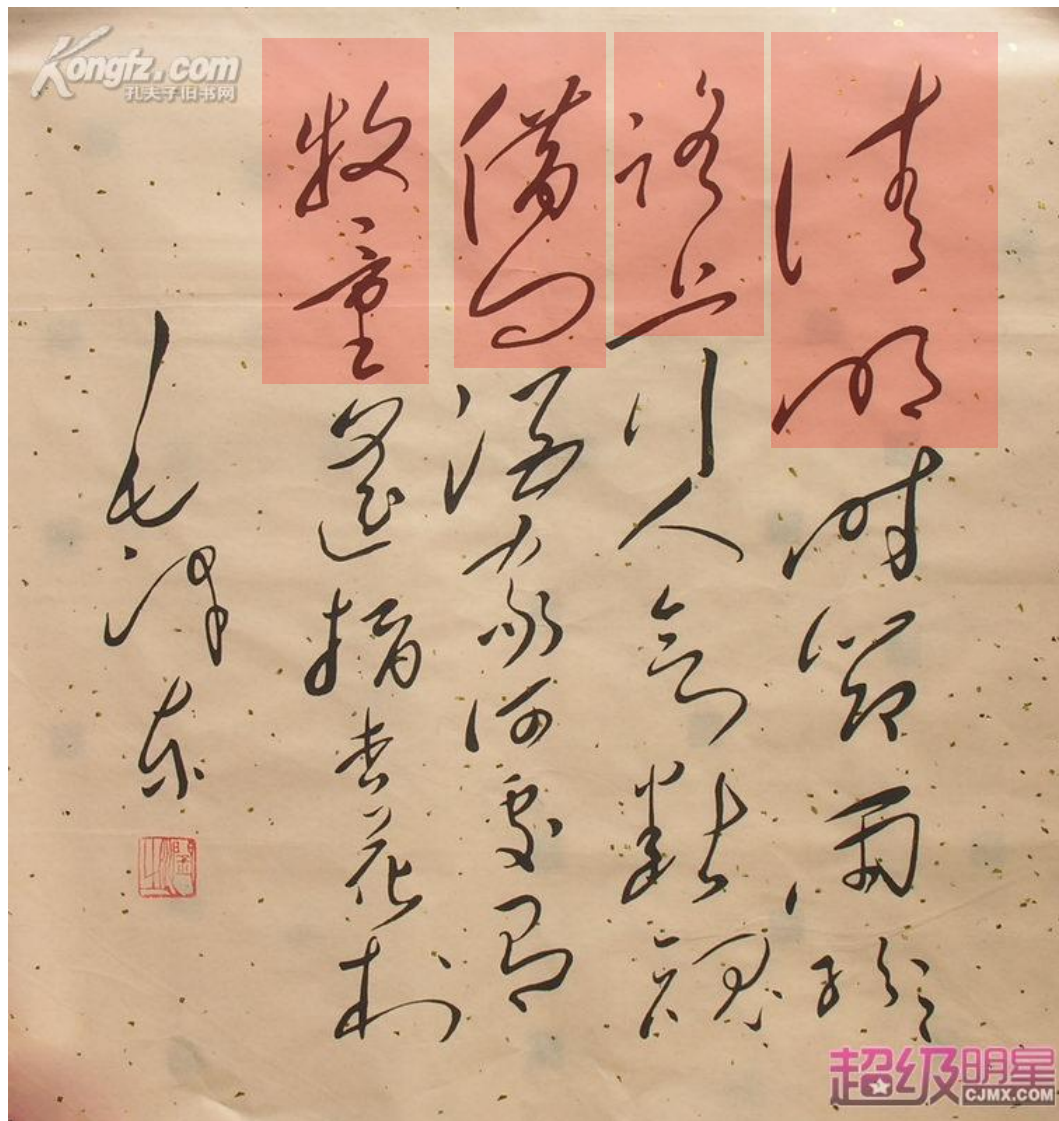
# 生活中的压缩例子

- ◆ 成语：字字珠玑
- ◆ 歇后语：高俅当太尉——一步登天
- ◆ 漫画
- ◆ 网络聊天术语：YYDS，不明觉厉
- ◆ 英语缩略语：IT，USTC
  
- ◆ 各种压缩软件（winrar/7-zip/kuaizip）
- ◆ 各种压缩后声音，如\*.MP3、\*.wma文件
- ◆ 各种压缩后图像，如\*.JPG、\*.BMP文件





# 为什么能压缩?



每句前两个字  
均可删去





# 为什么能压缩？文字冗余

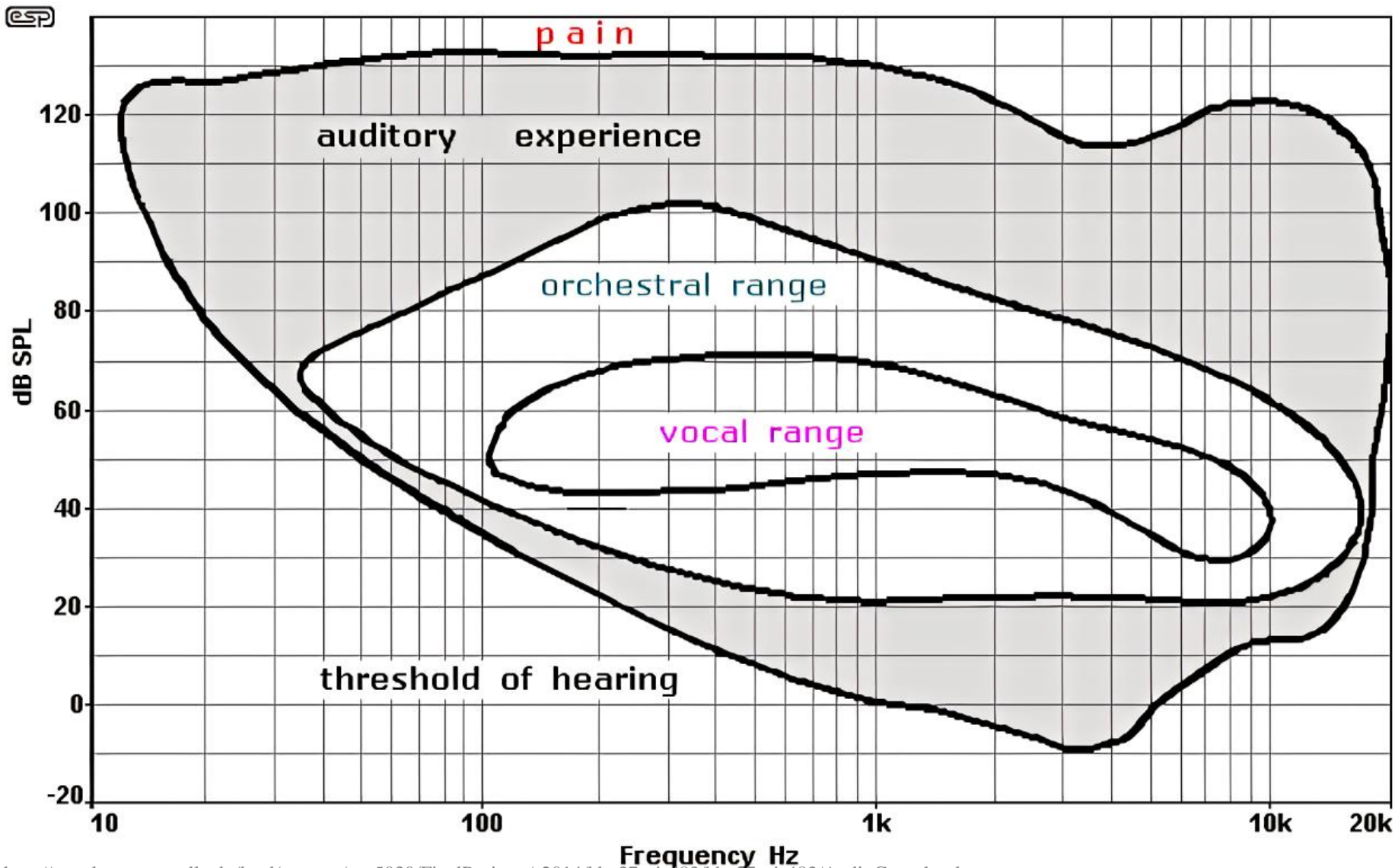
- ◆ 我们大\_\_都不喜\_\_上\_\_媒体课。
- ◆ 不用很多努力，就可以猜出完整的句子：
- ◆ 我们大家都不喜欢上多媒体课。
- ◆ 美国科学家香农指出，能猜出来的字符不运载信息，而不能猜出来的字符运载信息。

可以压缩的部分





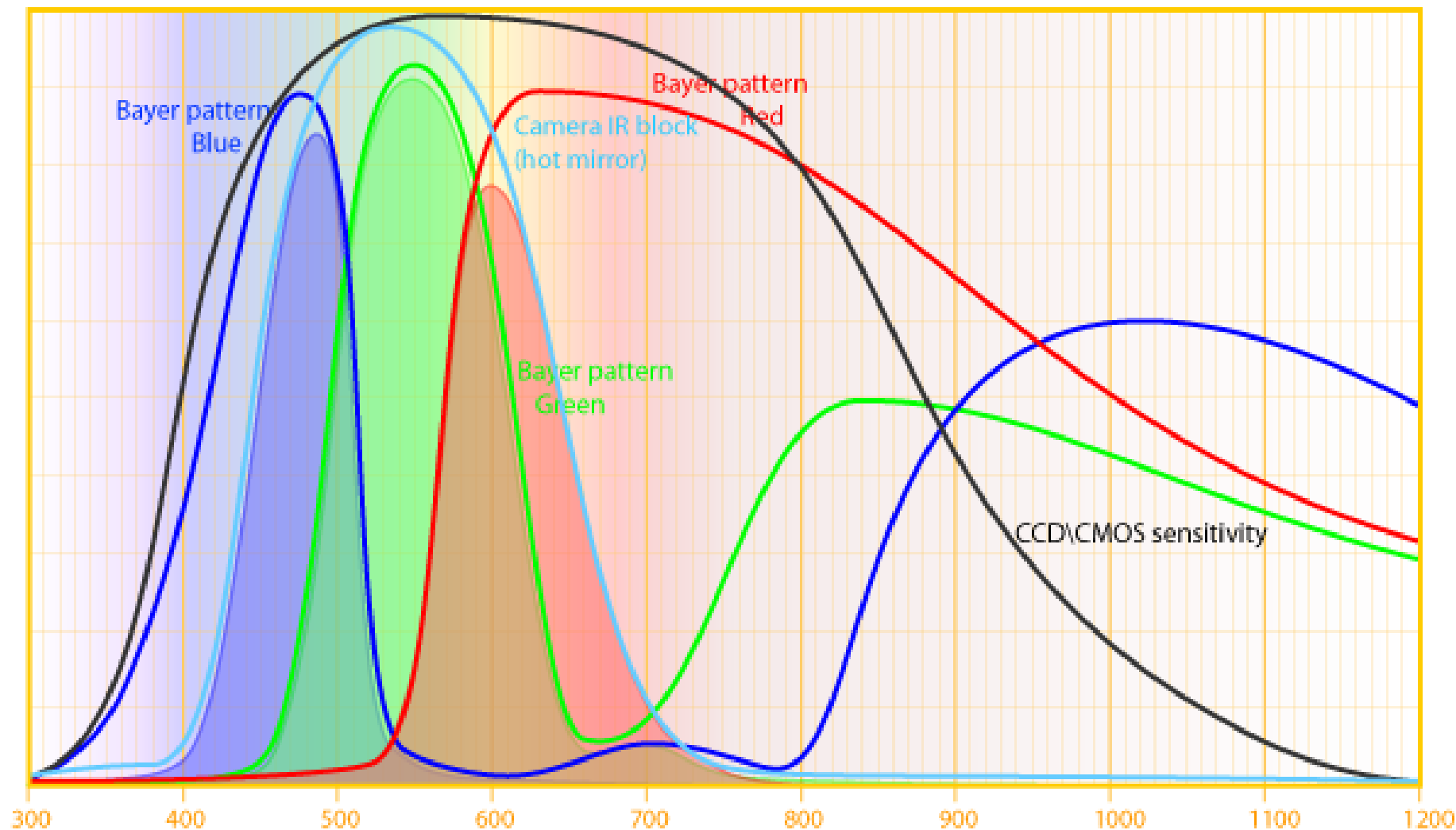
# 为什么能压缩？声音信号冗余 听觉系统的敏感度有限





# 为什么能压缩？图像信号冗余

## 视觉系统的敏感度有限

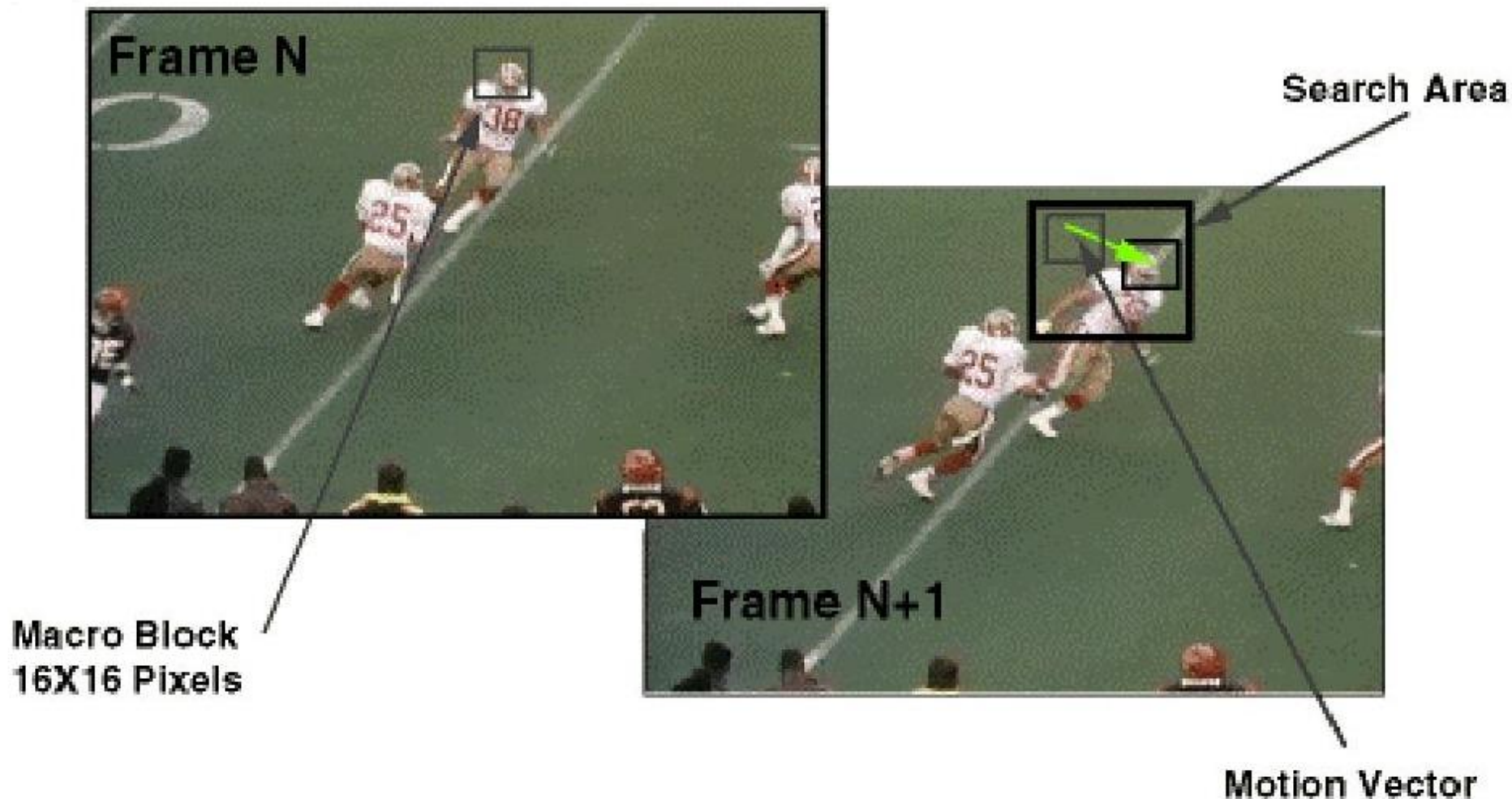






# 为什么能压缩？

## 视频信号的时间冗余





# 小结：压缩

## ◆ 编码和压缩的关系

- 压缩是一种特殊的编码
- 压缩 = 模型 + 编码

## ◆ 数据（比特流）的压缩

- 用相同长度的比特流表示更多的信息
- 用尽量短的比特流表达尽量多的信息

## ◆ 数据可被压缩的依据

- 数据本身存在冗余
- 听觉系统的敏感度有限
- 视觉系统的敏感度有限



- ◆ § 3.1 无损数据压缩
  - § 3.1.1 编码与压缩基本概念
  - § 3.1.2 压缩算法分类及评价原则
  - § 3.1.3 霍夫曼编码
  - § 3.1.4 算术编码
  - § 3.1.5 词典编码
  - § 3.1.6 RLE编码
- ◆ § 3.2 音频数据的压缩标准
- ◆ § 3.3 图像数据的压缩标准
- ◆ § 3.4 视频数据的压缩标准



# 多媒体压缩算法的分类

- ◆ 无损压缩、有损压缩
- ◆ 音频压缩、图像压缩、视频压缩
- ◆ 普通压缩算法、高速压缩算法
- ◆ 软件压缩、硬件压缩
- ◆ 通用压缩、专用压缩





# 无损压缩

- ◆无损压缩是指使用压缩后的数据进行重构(或者叫做还原, 解压缩), **重构后的数据与原来的数据完全相同**;
- ◆无损压缩用于要求重构的信号与原始信号完全一致的场合。一个很常见的例子是磁盘文件的压缩。根据目前的技术水平, 无损压缩算法一般可以把普通文件的数据压缩到原来的 $1/2 \sim 1/4$ 。一些常用的无损压缩算法有霍夫曼(Huffman)算法和LZW(Lenpel-Ziv & Welch)压缩算法。





# 有损压缩

- ◆有损压缩是指使用压缩后的数据进行重构，**重构后的数据与原来的数据有所不同，但不会让人对原始资料表达的信息造成误解。**
- ◆有损压缩适用于重构信号不一定非要和原始信号完全相同的场合。例如，图像和声音的压缩就可以采用有损压缩，因为其中包含的数据往往多于我们的视觉系统和听觉系统所能接收的信息，丢掉一些数据而不至于对声音或者图像所表达的意思产生误解，但可大大提高压缩比。





# 数据无损压缩的理论基础

## 信息论(information theory)

### ◆ 香农 (Claude Elwood Shannon)

□ Born: 30 April 1916 in Gaylord, Michigan, USA

□ Died: 24 Feb 2001 in Medford, Massachusetts, USA

◆ 1940年获得MIT数学博士学位和电子工程硕士学位。41年加入了贝尔实验室数学部，1948年发表《通讯的数学原理》，该文奠定了信息论的基础。

◆ 文中用非常简洁的数学公式定义了信息时代的基本概念：**熵**。在此基础上又定义了**信道容量**，指出了用降低传输速率来换取高保真通讯的可能性。这些贡献对通信工业具有革命性的影响。







# 给信息称体重——熵 (Entropy)

## ◆ 信息量(information content)

□ 具有确定概率事件的信息的定量度量，定义为

$$I(x) = \log_2[1 / p(x)] = -\log_2 p(x)$$

## ◆ 熵(entropy)

□ 按照香农(Shannon)的理论，在有限的互斥和联合穷举事件的集合中，熵为事件的信息量的平均值，也称事件的平均信息量(mean information content)，数学表示为

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





# 熵计算示例

◆ 字符串“aabbaccbaa”的长度为 10，字符a、b、c分别出现了5、3、2次，则 a、b、c三个字符出现的概率分别为0.5、0.3、0.2，则他们的熵分别为：

◆  $E_a = -\log_2(0.5) = 1$

◆  $E_b = -\log_2(0.3) = 1.737$

◆  $E_c = -\log_2(0.2) = 2.322$

◆ 整条信息的熵即表达整个字符串需要的比特：

◆  $E = E_a * 5 + E_b * 3 + E_c * 2 = 14.855$

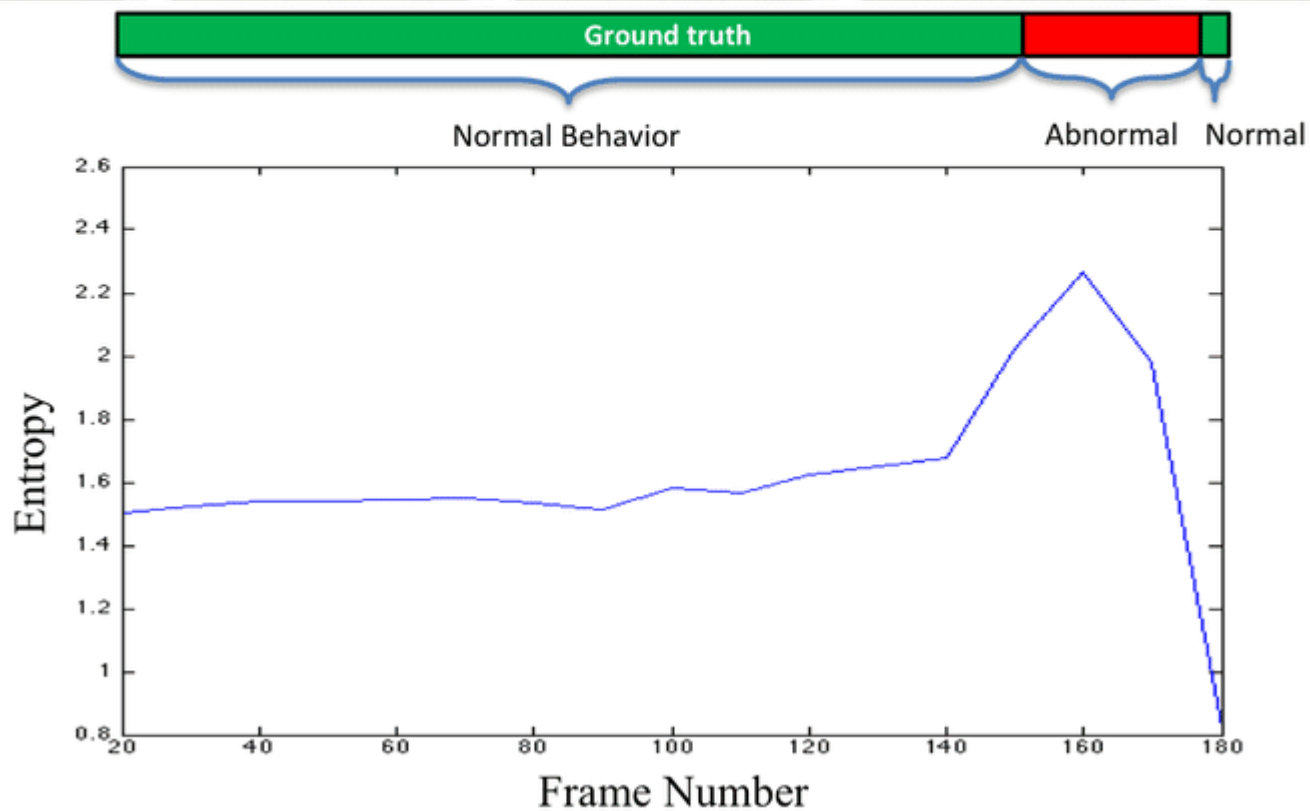
**10\*8=80比特**

$$H(X) = - \sum_x p(x) \log p(x) = -\mathbb{E}[\log(p(x))]$$





# 熵计算示例





# 联合国五种工作语言

各种文字的信息熵比较：

法文：3.98比特

西班牙文：4.01比特

英文：4.03比特

俄文：4.35比特

中文：9.65比特

表示一个汉字  
需要的最少比  
特数是9.65

汉字的信息量最大，因而，在信息管理和传递的时候，中文处于最不利的地位。





# 压缩算法的评价原则

## ◆ 压缩比

- 未压缩的比特流长度/压缩后的比特流长度

## ◆ 运算复杂度

- CPU或MPU完成压缩的运算时间
- 算法需要的临时存储空间

## ◆ 通用性

- 是否符合国际标准

## ◆ .....



- ◆ § 3.1 无损数据压缩
  - § 3.1.1 编码与压缩基本概念
  - § 3.1.2 压缩算法分类及评价原则
  - § 3.1.3 霍夫曼编码
  - § 3.1.4 算术编码
  - § 3.1.5 词典编码
  - § 3.1.6 RLE编码
- ◆ § 3.2 音频数据的压缩标准
- ◆ § 3.3 图像数据的压缩标准
- ◆ § 3.4 视频数据的压缩标准

◆ 有一幅16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，如表示

像素	A	B	C	D	E
个数	5	4	4	2	1

◆ 如果用3个位表示5个等级的灰度值，也就是每个像素用3位表示，编码图像总共需要 $16*3=48$ 比特



# 示例1熵

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

◆ 有一幅16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，如表示

像素	A	B	C	D	E
个数	5	4	4	2	1
频度	5/16	1/4	1/4	1/8	1/16
符号熵	1.72	2	2	3	4
	8.6	8	8	6	4

图像的熵： **34.6比特**



# 香农-范诺编码

## Shannon-Fano coding

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，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

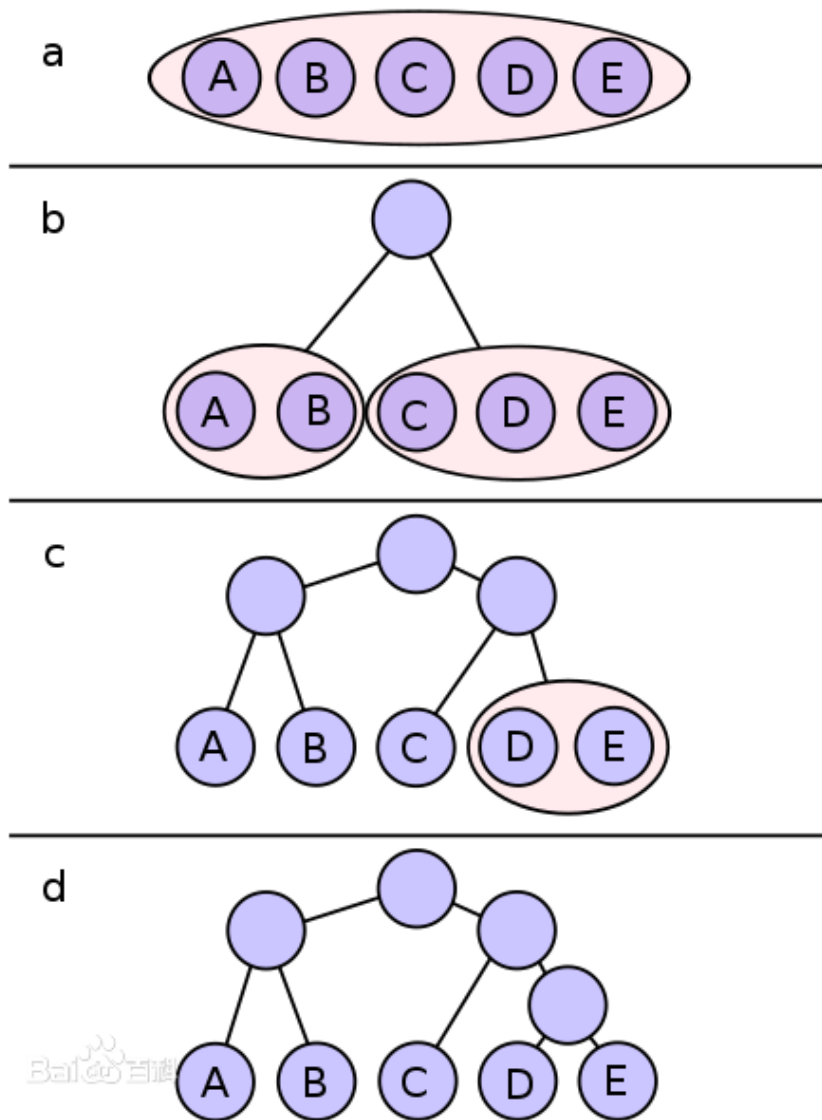
像素	A	B	C	D	E
个数	5	4	4	2	1
编码	00	01	10	110	111

需要的总位数为  $35 =$

$5 \times 2 + 4 \times 2 + 4 \times 2 + 2 \times 3 + 1 \times 3$

压缩比  $48:35 \approx \mathbf{1.37}$

平均码长 2.1875





# Shannon–Fano coding 算法评价

- ◆ 其名称来自于以Claude Shannon和Robert Fano
- ◆ 会产生相当有效的可变长度编码
- ◆ 但是，并不总是产生最优的前缀码
  - 概率{0.35, 0.17, 0.17, 0.16, 0.15}
  - 编码{00, 01, 10, 110, 111}
  - $0.35*2+0.17*2+0.17*2+0.16*3+0.15*3 = 2.31$ (平均码长)



◆ 有一幅16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，如表示

像素	A	B	C	D	E
个数	2	8	2	3	1

◆ 如果用3个位表示5个等级的灰度值，也就是每个像素用3位表示，编码图像总共需要48比特



## 示例2熵

像素	A	B	C	D	E
个数	2	8	2	3	1
频度	$2/16$	$8/16$	$2/16$	$3/16$	$1/16$
符号熵	3	1	3	1.585	4
	6	8	6	4.755	4

图像的熵: 28.755比特





# 霍夫曼编码

## Huffman coding

◆ 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。

□ 1952年, David A. Huffman在麻省理工攻读博士时发表了(A Method for the Construction of Minimum-Redundancy Codes)一文, 一般就叫做Huffman编码。

霍夫曼(Huffman)编码的核心思想:

**出现次数最多的符号用最短的编码**

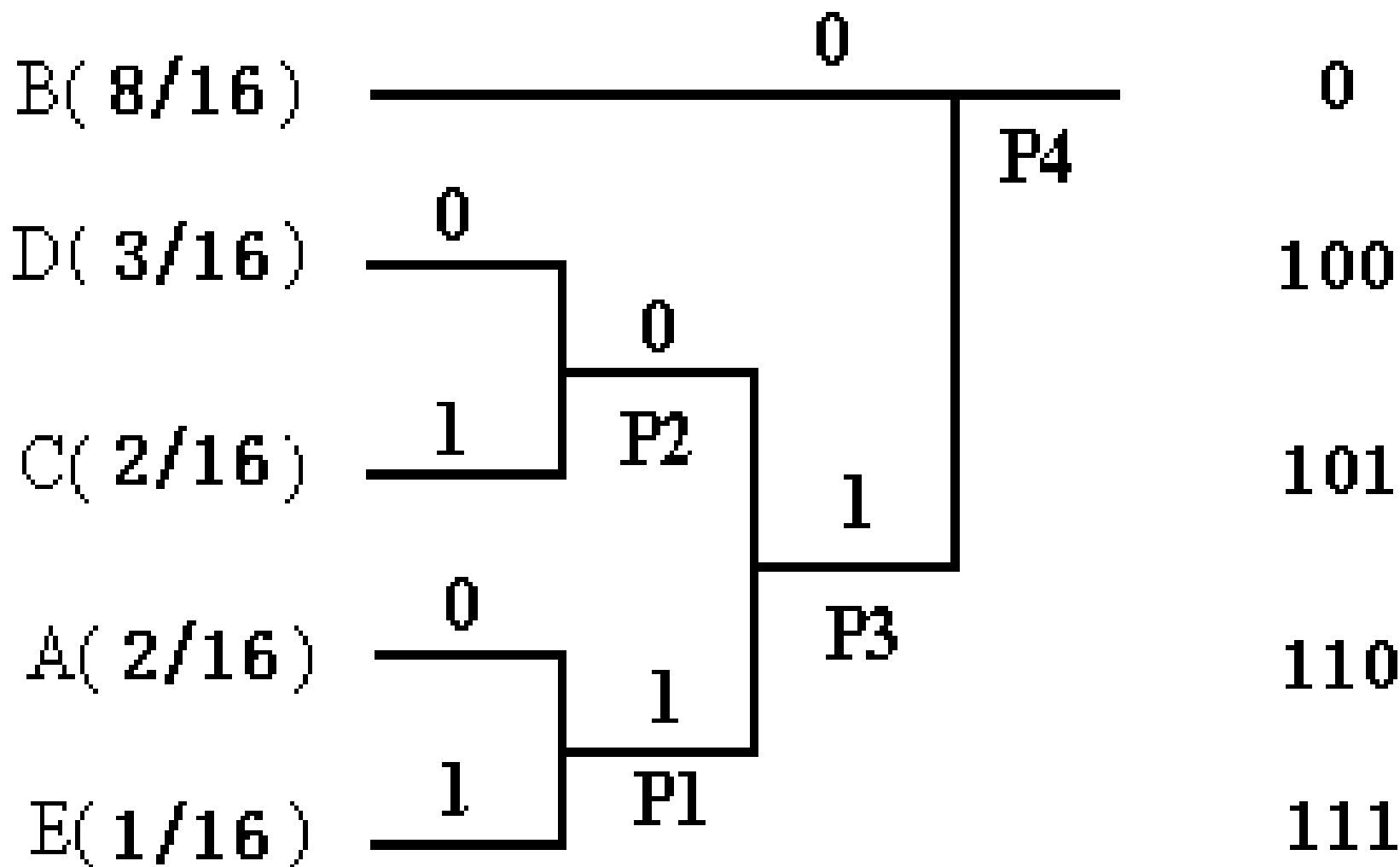
**出现次数最少的符号用最长的编码**

◆ 广泛用在JPEG, MPEG, H.26x等编码标准中





# Huffman编码过程







# Huffman编码结果

共需要32  
比特

像素	个数	频度	码	码长	比特数
<b>A</b>	<b>2</b>	<b>1/8</b>	<b>110</b>	<b>3bit</b>	<b>6bit</b>
<b>B</b>	<b>8</b>	<b>1/2</b>	<b>0</b>	<b>1bit</b>	<b>8bit</b>
<b>C</b>	<b>2</b>	<b>1/8</b>	<b>101</b>	<b>3bit</b>	<b>6bit</b>
<b>D</b>	<b>3</b>	<b>3/16</b>	<b>100</b>	<b>3bit</b>	<b>9bit</b>
<b>E</b>	<b>1</b>	<b>1/16</b>	<b>111</b>	<b>3bit</b>	<b>3bit</b>





# 算法评价

◆ 霍夫曼码的码长虽然是可变的，但却不需要另外附加同步代码。几个问题值得注意：

- 1. 霍夫曼码没有错误保护功能；
- 2. 霍夫曼码是可变长度码，因此很难随意查找或调用压缩文件中间的内容，然后再译码；
- 3. 接收端需保存一个与发送端相同的霍夫曼码表。

◆ 完美体现信息论中熵的思想

◆ 建立在概率统计的基础之上

◆ 其思想已在多种领域成功应用





# 个人观点陈述

◆畸对称性：世界上的绝大多数财富集中在极少的富翁手中

◆熵极限：当各种符号等概率出现的时候这些符号组成的消息携带的信息量达到极小值。这暗示了什么自然规律？

$$\frac{x_1 + x_2 + \dots + x_n}{n} \geq \sqrt[n]{x_1 x_2 \dots x_n}$$





## 小结： Huffman 编码

- ◆ 60 年代、70 年代乃至 80 年代的早期，数据压缩领域几乎一直被 Huffman 编码及其分支所垄断。人们努力寻找各种向熵极限逼近的编码方法。
- ◆ 从一个极限空间跨越到另一个极限空间？
- ◆ 从一种思路切换到另外一种思路？
- ◆ 寻求一种新的方法
- ◆ 需求一个新的起点
- ◆ 在寻找新的压缩编码方法的艰难历程中，70 年代末到 80 年代中期，算术编码开始崭露头角。



- ◆ § 3.1 无损数据压缩
  - § 3.1.1 编码与压缩基本概念
  - § 3.1.2 压缩算法分类及评价原则
  - § 3.1.3 霍夫曼编码
  - § 3.1.4 算术编码
  - § 3.1.5 词典编码
  - § 3.1.6 RLE编码
- ◆ § 3.2 音频数据的压缩标准
- ◆ § 3.3 图像数据的压缩标准
- ◆ § 3.4 视频数据的压缩标准



# 算术编码的起源

- ◆ 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。
- ◆ 怎样只输出 0.322 个 0 或 0.322 个比特？

数学家们在十几年前终于想到了算术编码这种神奇的方法！！！！





# 算术编码(Arithmetic coding)

算术编码在图像数据压缩标准(如JPEG, JBIG)中扮演了重要的角色。**在算术编码中, 消息用0到1之间的实数进行编码**, 算术编码用到两个基本的参数: 符号的概率和它的编码间隔。信源符号的概率决定压缩编码的效率, 也决定编码过程中信源符号的间隔, 而这些间隔包含在0到1之间。编码过程中的间隔决定了符号压缩后的输出。





# 算术编码示例

假设信源符号为  $\{00, 01, 10, 11\}$ ，这些符号的概率分别为  $\{0.1, 0.4, 0.2, 0.3\}$ ，根据这些概率可把间隔  $[0, 1)$  分成4个子间隔： $[0, 0.1)$ ， $[0.1, 0.5)$ ， $[0.5, 0.7)$ ， $[0.7, 1)$ ，二进制消息序列的输入为：10 00 11 00 10 11 01

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

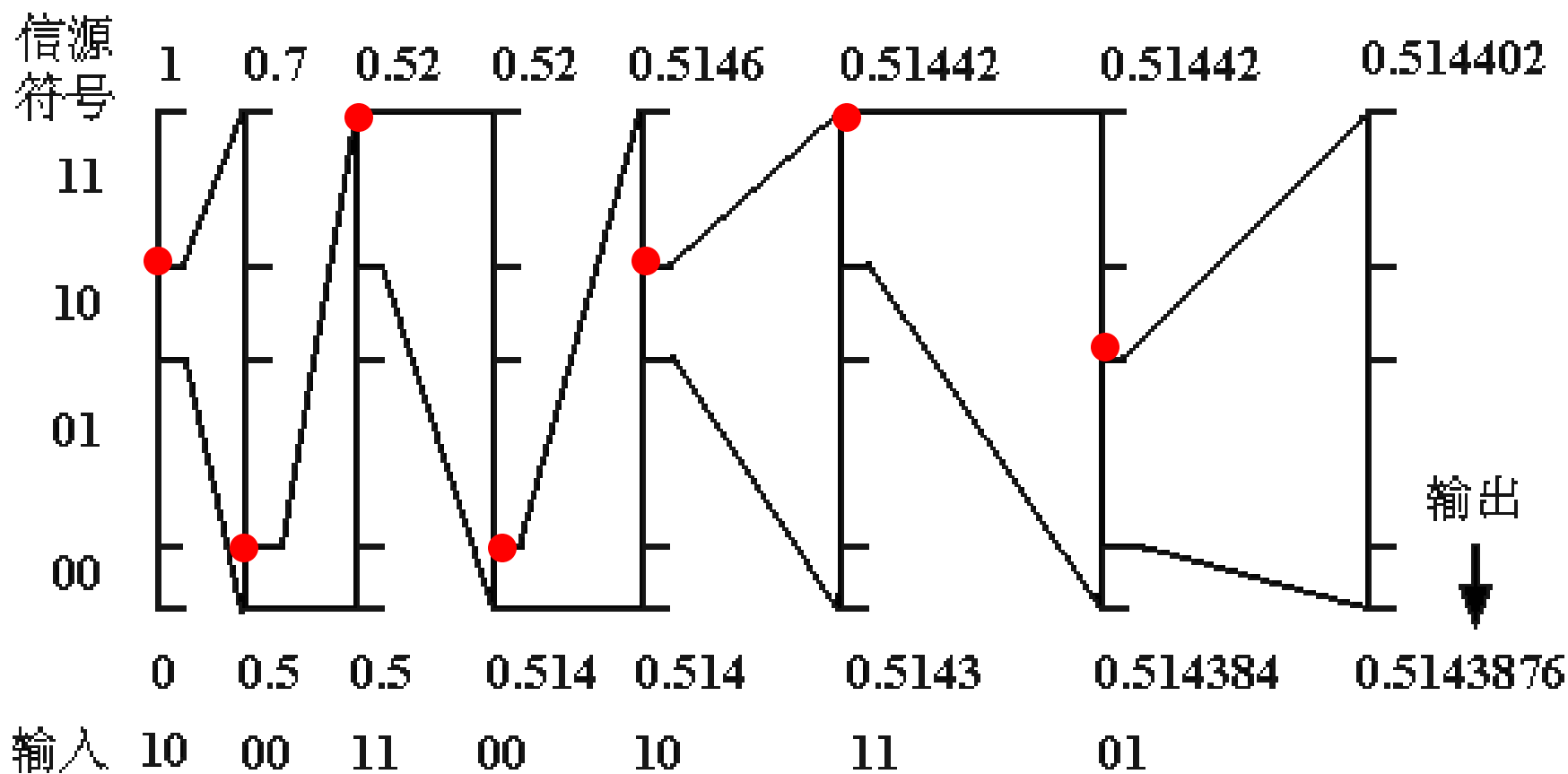






# 算术编码过程

编码间隔





# 小结：伟大的算术编码

- ◆ 用一个实数表示一个消息串
- ◆ 将任意一个消息串映射到 $[0, 1)$  区间中的一个点
- ◆ 当然由于计算机所能表示的实数精度有限，不能实现对无穷消息到一个点的压缩.
  - 算术编码体现了从模糊到精确的表现
  - 提示了对微观世界探索的重要性
  - 体现了算术的魅力
  - 人类想象力对极限的逼近



- ◆ § 3.1 无损数据压缩
  - § 3.1.1 编码与压缩基本概念
  - § 3.1.2 压缩算法分类及评价原则
  - § 3.1.3 霍夫曼编码
  - § 3.1.4 算术编码
  - § 3.1.5 词典编码
  - § 3.1.6 RLE编码
- ◆ § 3.2 音频数据的压缩标准
- ◆ § 3.3 图像数据的压缩标准
- ◆ § 3.4 视频数据的压缩标准



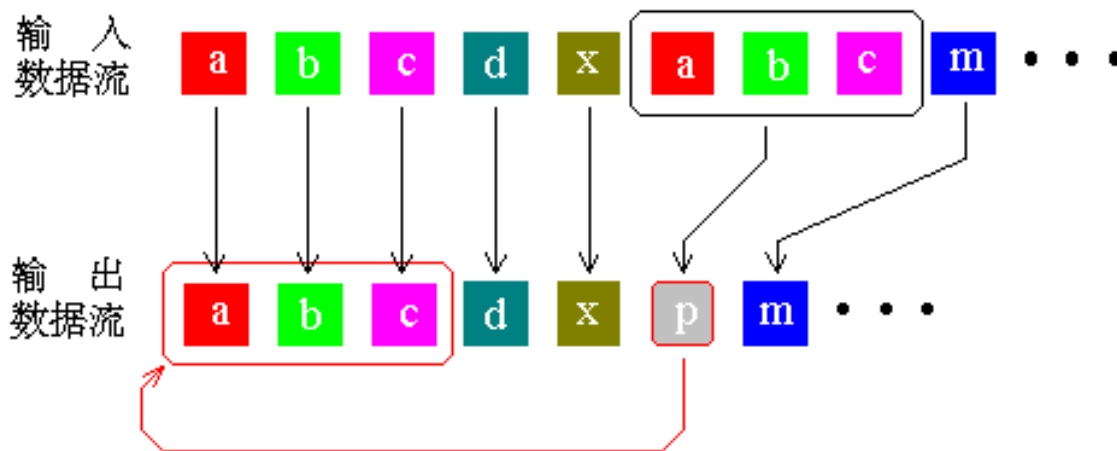
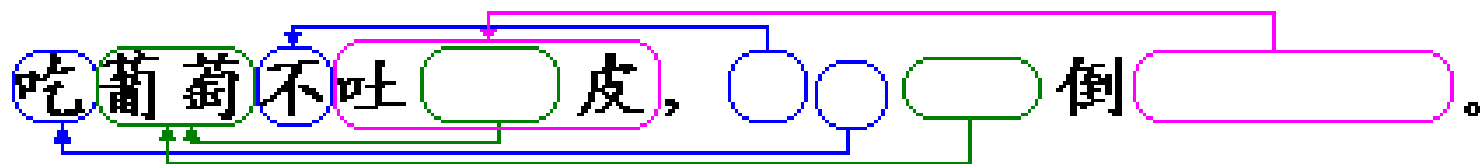
# 聪明的以色列人

◆ 以上讨论的压缩模型都基于对信息中单个字符出现频率的统计而设计的，直到 70 年代末期，这种思路在数据压缩领域一直占据着统治地位。一旦某项技术在某一领域形成了惯例，人们就很难创造出在思路与其大相径庭的哪怕是更简单更实用的技术来。在 1977 年，终于有两位聪明的以色列人，带给了我们既高效又简便的“字典模型”。至今，几乎我们日常使用的所有通用压缩工具，象 ARJ, PKZip, WinZip, LHArc, RAR, GZip, ACE, ZOO, TurboZip, Compress, JAR.....



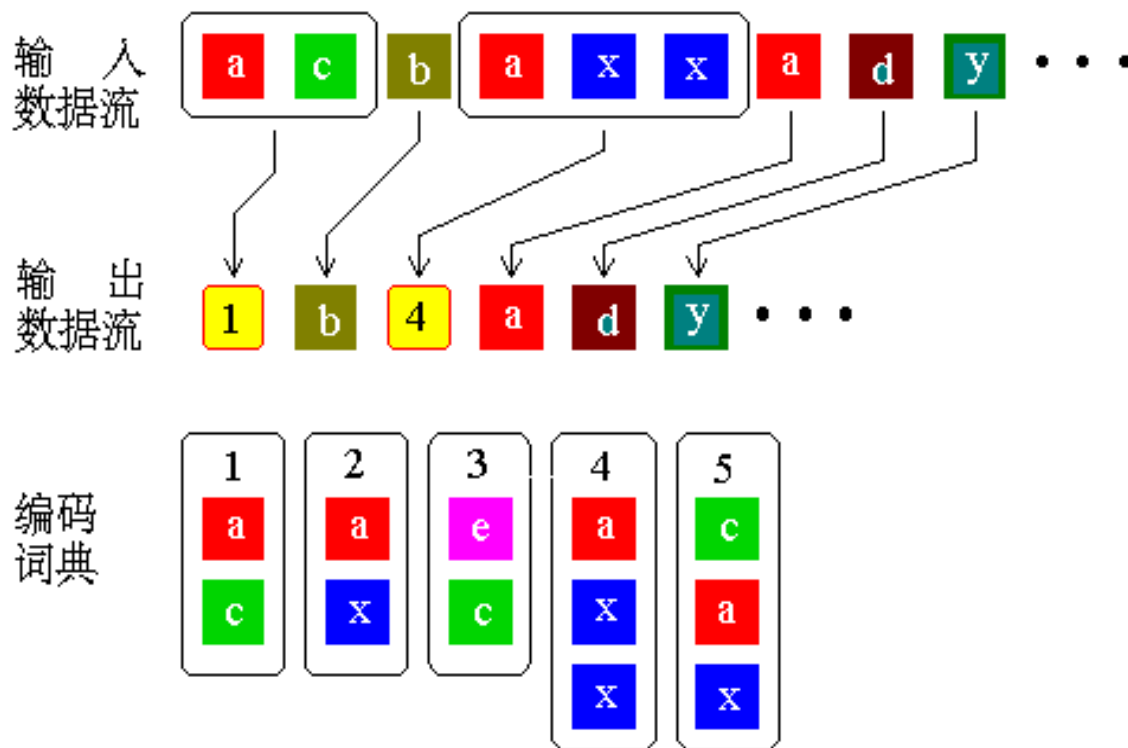


# 第一类词典编码



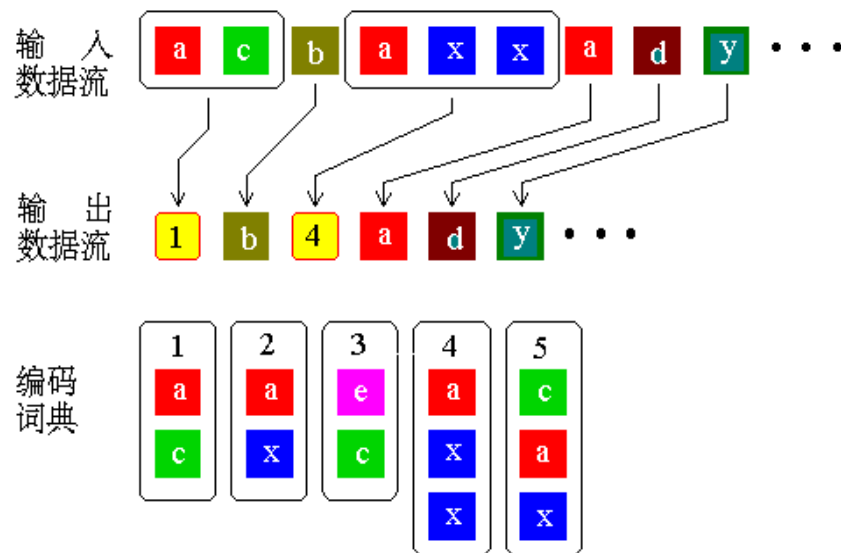
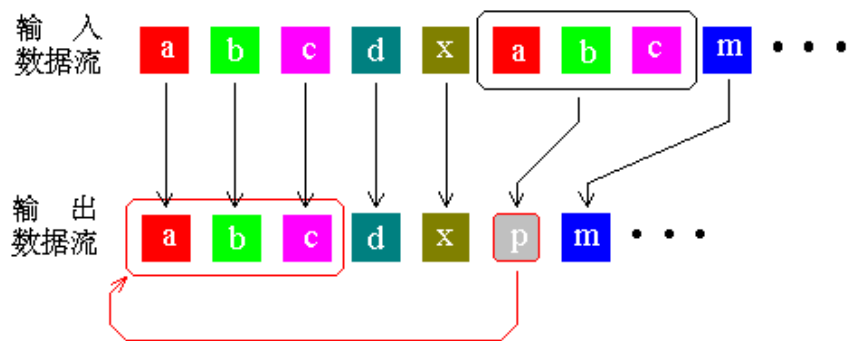
第一类词典法的想法是企图查找正在压缩的字符序列是否在以前输入的数据中出现过，然后用已经出现过的字符串替代重复的部分，它的输出仅仅是指向早期出现过的字符串的“指针”。





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)”，这种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。

## ◆思路：不直接存储重复出现的符号



## ◆类比：索引的应用

- 图书馆的索书卡
- 各种字典
- 伪彩色

- ◆ § 3.1 无损数据压缩
  - § 3.1.1 编码与压缩基本概念
  - § 3.1.2 压缩算法分类及评价原则
  - § 3.1.3 霍夫曼编码
  - § 3.1.4 算术编码
  - § 3.1.5 词典编码
  - § 3.1.6 RLE编码
- ◆ § 3.2 音频数据的压缩标准
- ◆ § 3.3 图像数据的压缩标准
- ◆ § 3.4 视频数据的压缩标准





# 比词典编码更简单

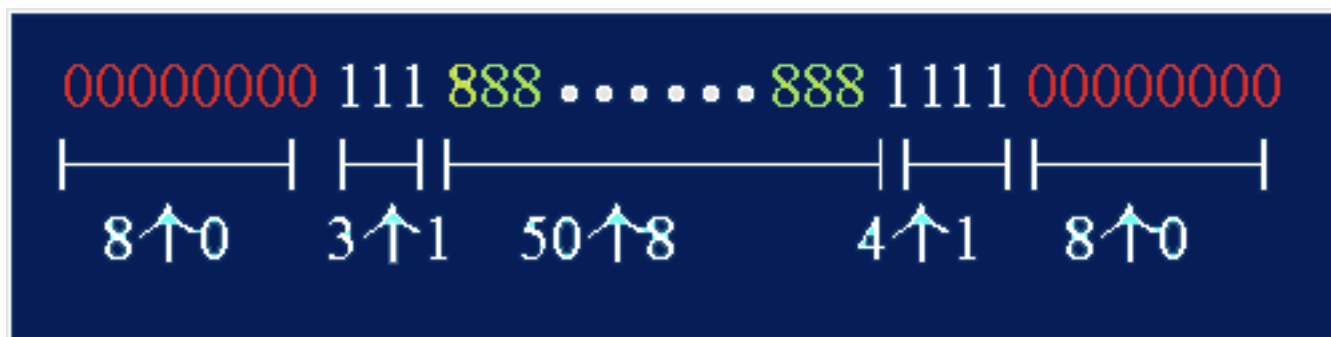
- ◆词典编码非常直观，容易理解。在寻求更简单编码的时候，有人发明了RLE编码（行程编码）
- ◆该编码虽然简单，但是应用十分普及，在著名的JPEG压缩算法中就有使用





# RLE编码

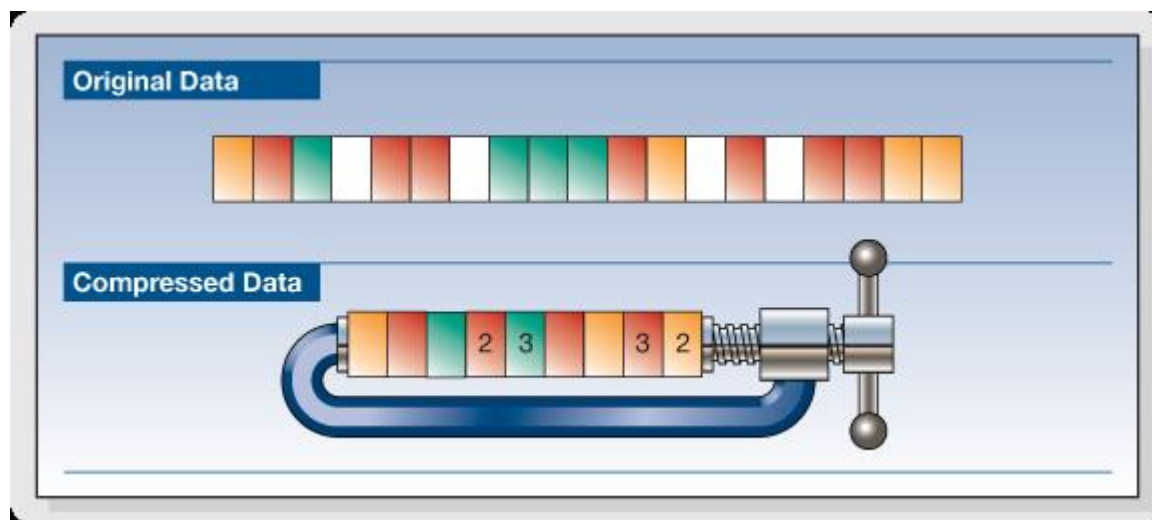
RLE(run length encoding) 编码的概念



用RLE编码方法得到的代码为：80315084180。代码中用黑体表示的数字是行程长度，黑体字后面的数字代表象素的颜色值。例如黑体字50代表有连续50个象素具有相同的颜色值，它的颜色值是8。



- ◆ 压缩编码的目标：向香农提出的熵极限无限逼近（挑战极限），向计算机的极限无限逼近
- ◆ 事实上在使用编码算法的时候必须考虑编码和解码的速度（结果和过程）
- ◆ 很多情况下会将多种压缩算法结合在一起使用（集众家所长）



# 多媒体技术基础

## 第三章 多媒体数据压缩

### § 3.1 无损数据压缩

谢谢！

