Research Institute for Future Media Computing
未来媒体技术与研究所

Institute of Computer Vision
计算机视觉研究所

多媒体系统导论

Fundamentals of Multimedia System

授课教师：文嘉俊
邮箱：wenjiajun@szu.edu.cn

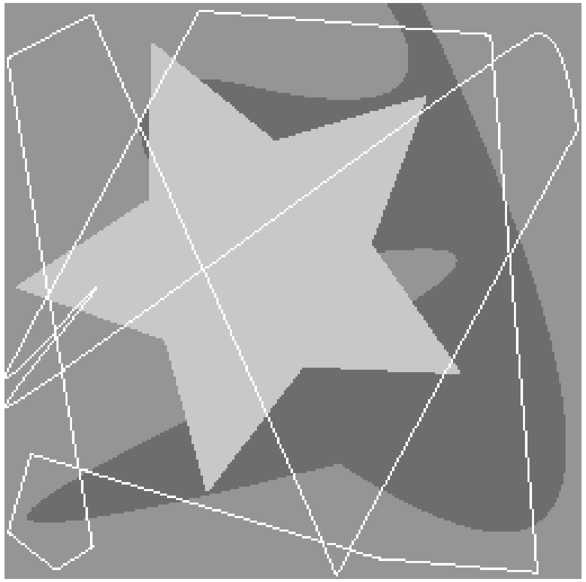
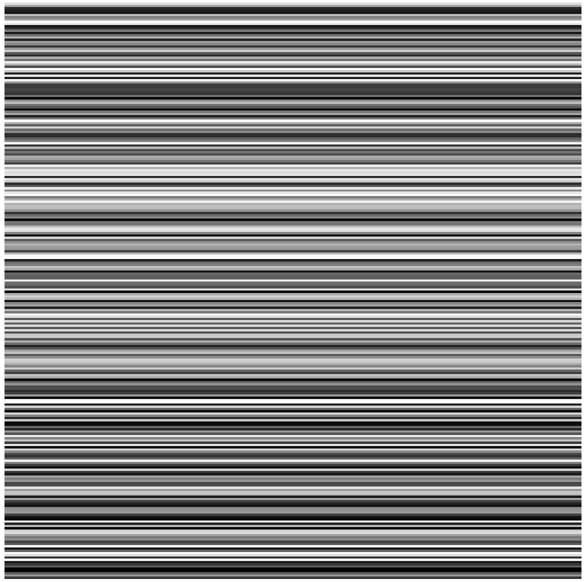
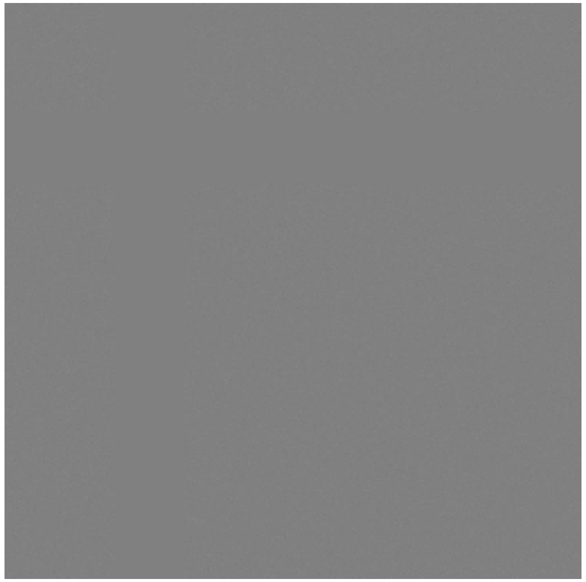
2024年春季课程

Outline of Lecture 06

- Introduction-简介
- Basics of Information Theory-信息论基础
- Run-Length Coding-游程编码
- Variable-Length Coding-变长编码
  - Shannon–Fano Algorithm-香农-凡诺算法
  - Huffman Coding-赫夫曼编码
  - Adaptive Huffman Coding-自适应赫夫曼编码
- Dictionary-Based Coding-基于字典的编码
- Arithmetic Coding-算术编码
- Lossless Image Compression-无损图像压缩
- Experiments-实验

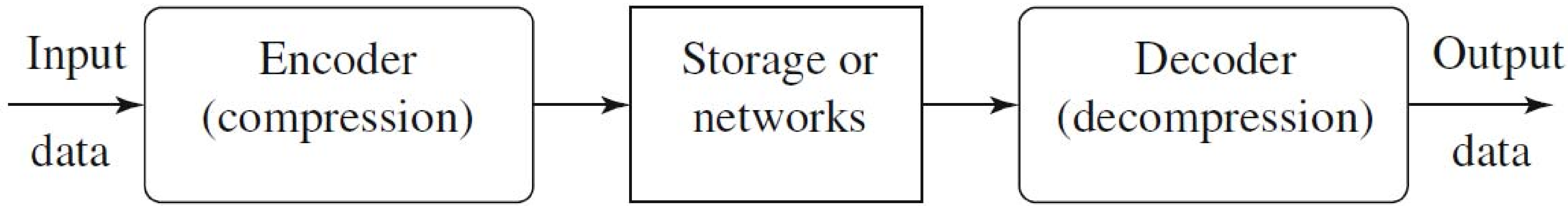
Introduction-简介

- What is Compression-什么是压缩?

Introduction-简介

- What is Compression-什么是压缩?
  - The process of coding that will effectively reduce the total number of bits needed to represent certain information-
  - 有效地减少表示某种信息所需的比特总数的编码过程.



输入数据　编码器(压缩)　存储或传输　解码器(解压缩)　输出数据

A General Data Compression Scheme.
一个通用的数据压缩方案

Introduction-简介

- What is Compression-什么是压缩?
  - If the compression and decompression processes induce no information loss, then the compression scheme is **lossless**; otherwise, it is **lossy.**
  - 如果压缩和解压过程无信息损失，则压缩方案是无损的，否则是有损的.
  - **Compression ratio-**压缩率:

$$Compression\ Ratio = \frac{B_0}{B_1}$$

B_0 - number of bits before compression-压缩前比特数

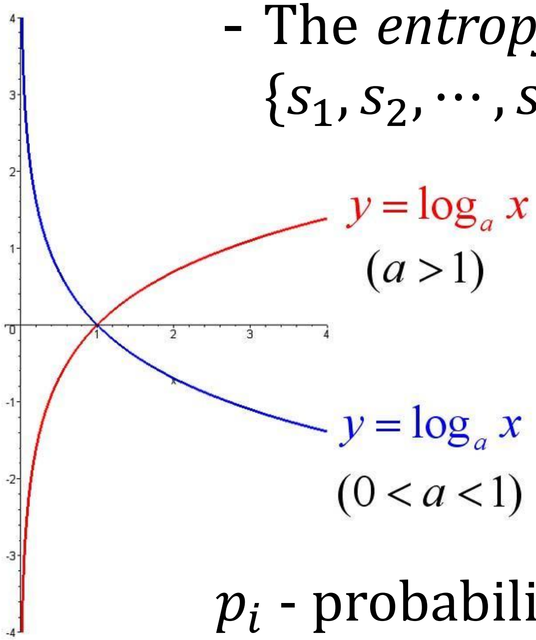
B_1 - number of bits after compression-压缩后比特数

Outline of Lecture 06

- Introduction-简介
- Basics of Information Theory-信息论基础
- Run-Length Coding-游程编码
- Variable-Length Coding-变长编码
  - Shannon–Fano Algorithm-香农-凡诺算法
  - Huffman Coding-赫夫曼编码
  - Adaptive Huffman Coding-自适应赫夫曼编码
- Dictionary-Based Coding-基于字典的编码
- Arithmetic Coding-算术编码
- Lossless Image Compression-无损图像压缩
- Experiments-实验

Basics of Information Theory-信息论基础

- Entropy-熵



- The *entropy* η of an information source with alphabet $S = \{s_1, s_2, \cdots, s_n\}$-一个具有符号集S的信息源的熵定义为:

$$\eta = H(S) = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i}$$

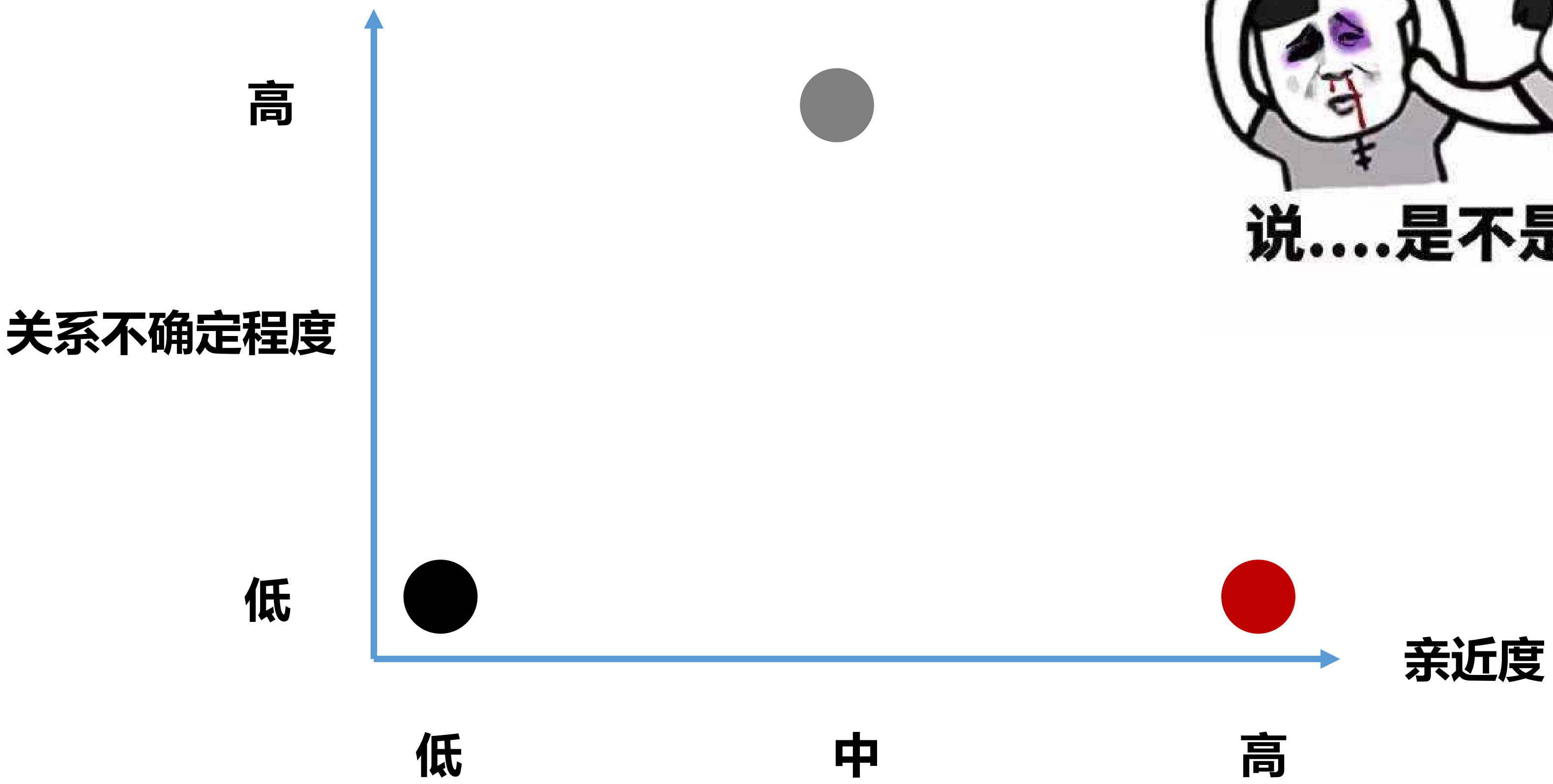
$$= -\sum_{i=1}^{n} p_i \log_2 p_i$$

p_i - probability that symbol s_i will occur in S-符号s_i发生概率.

$\log_2 \frac{1}{p_i}$ - indicates **the amount of information** (self-information as defined by Shannon) contained in s_i, which corresponds to the number of bits needed to encode s_i-符号s_i包含的信息量（也称自信息），反映出编码s_i所需的比特位数.

Basics of Information Theory-信息论基础

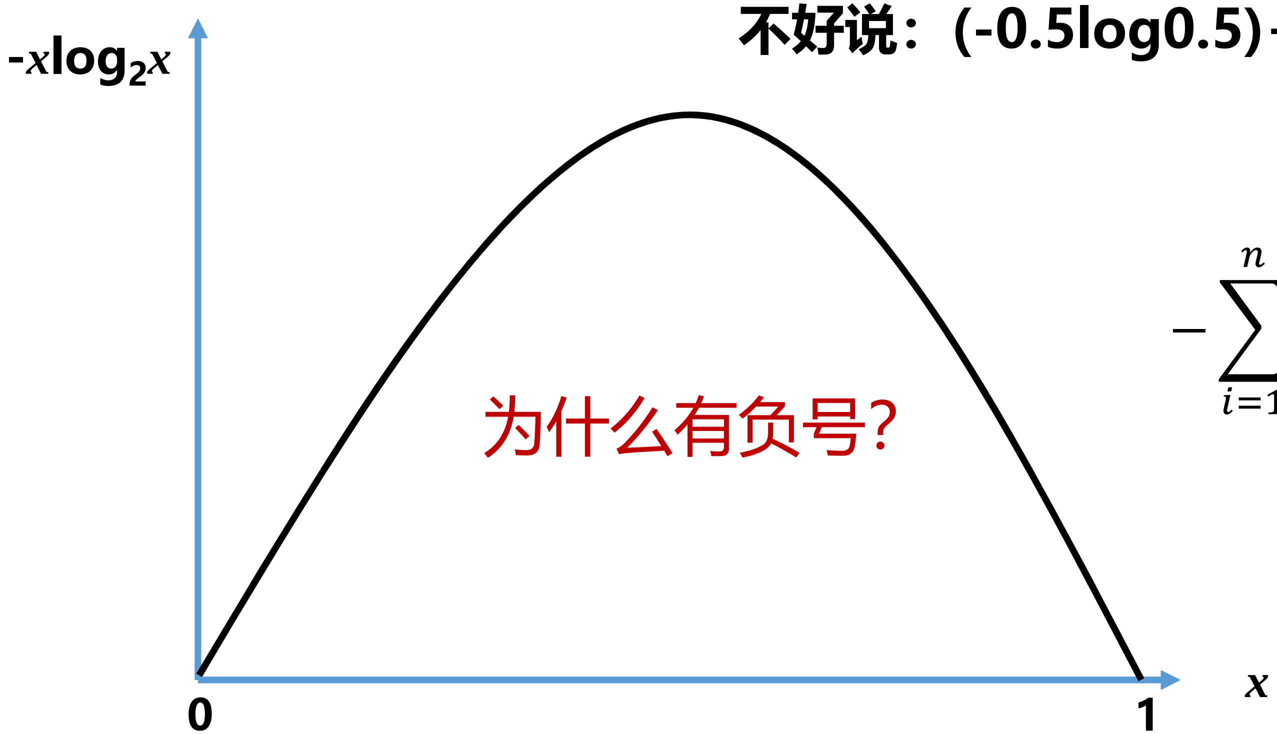
◆ Entropy-熵



Basics of Information Theory-信息论基础

- Entropy-熵

很亲近：-0log0-1log1=0,
不亲近：-1log1-0log0=0,
不好说：(-0.5log0.5)+(-0.5log0.5)=1



$$-\sum_{i=1}^{n} p_i \log_2 p_i$$

熵是指所有可能发生事件中所包含信息的期望平均值

Basics of Information Theory-信息论基础

- Entropy-熵
  - 实例：一段电报中有四个字符$S=\{A, B, C, D\}$，其概率分别为1/2，1/6，1/6，1/6，求各个字符需要多少位来表示，总体需要多少位来表示?

$$\frac{1}{2}\times 7+3\times\frac{1}{6}\times 7=7$$

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

A: $-\log_2 2^{-1} = 1$

B: $-\log_2 6^{-1} \approx 2.5850$

C: $-\log_2 6^{-1} \approx 2.5850$

D: $-\log_2 6^{-1} \approx 2.5850$

$$\log_2 \frac{1}{p_i}$$

$$4\times\frac{1}{4}\times\log_2\left(\frac{1}{\left(\frac{1}{4}\right)}\right)=2$$

$H(S) = -1/2 \times \log_2 2^{-1} - 3 \times 1/6 \times \log_2 6^{-1}$

≈ 1.7924

用ASCII传输需要多少位?

不采用编码技术四个状态需要多少个位?

Basics of Information Theory-信息论基础

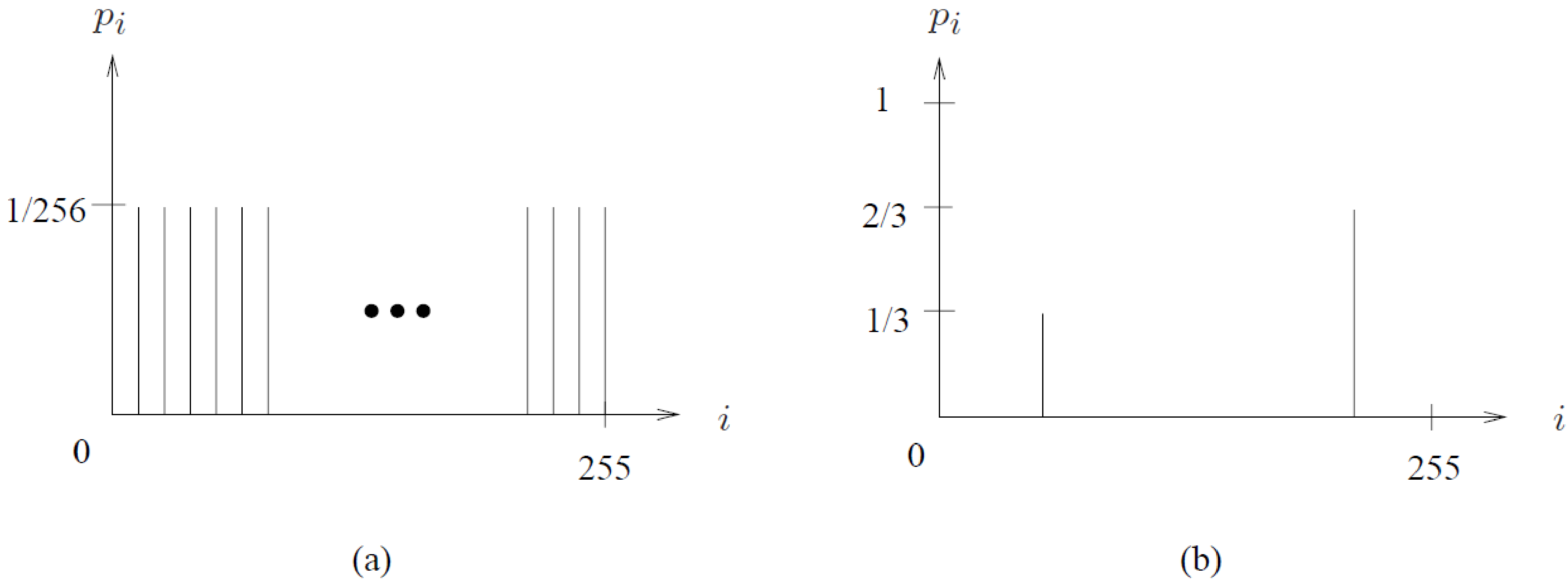
- Entropy-熵
  - The definition of entropy is aimed at identifying *often-occurring symbols* in the data stream as good candidates for *short codewords* in the compressed bitstream.
  - 寻找高频率符号，分配短码字.
  - If a symbol occurs rarely, its probability $p_i = 1/100$ is low, and thus its self-information $\log_2 100$ is a relatively large number. This reflects the fact that it takes a longer bitstring to encode it.
  - 低概率，自信息大，编码位长.

Basics of Information Theory-信息论基础

- Entropy-熵

$$\eta = \sum_{i=0}^{255} \frac{1}{256} \cdot \log_2 256 = 256 \cdot \frac{1}{256} \cdot \log_2 256 = 8$$

$$\eta = \frac{1}{3} \cdot \log_2 3 + \frac{2}{3} \cdot \log_2 \frac{3}{2}$$
$$= 0.33 \times 1.59 + 0.67 \times 0.59 = 0.52 + 0.40 = 0.92$$



Histograms for Two Gray-level Images.

Basics of Information Theory-信息论基础

◆ Entropy and Code Length-熵和编码长度

- The entropy η is a weighted sum of terms $\log_2 1/p_i$; hence it represents the *average amount of information* contained per symbol in the source S
- 熵反映信息源S中字符的平均信息量.
- Entropy specifies the *lower bound* for the average number of bits to code each symbol in *S.*
- 熵指明了对S中每个符号进行编码所需的平均位数的下界.

$$\eta \leq \bar{l}$$

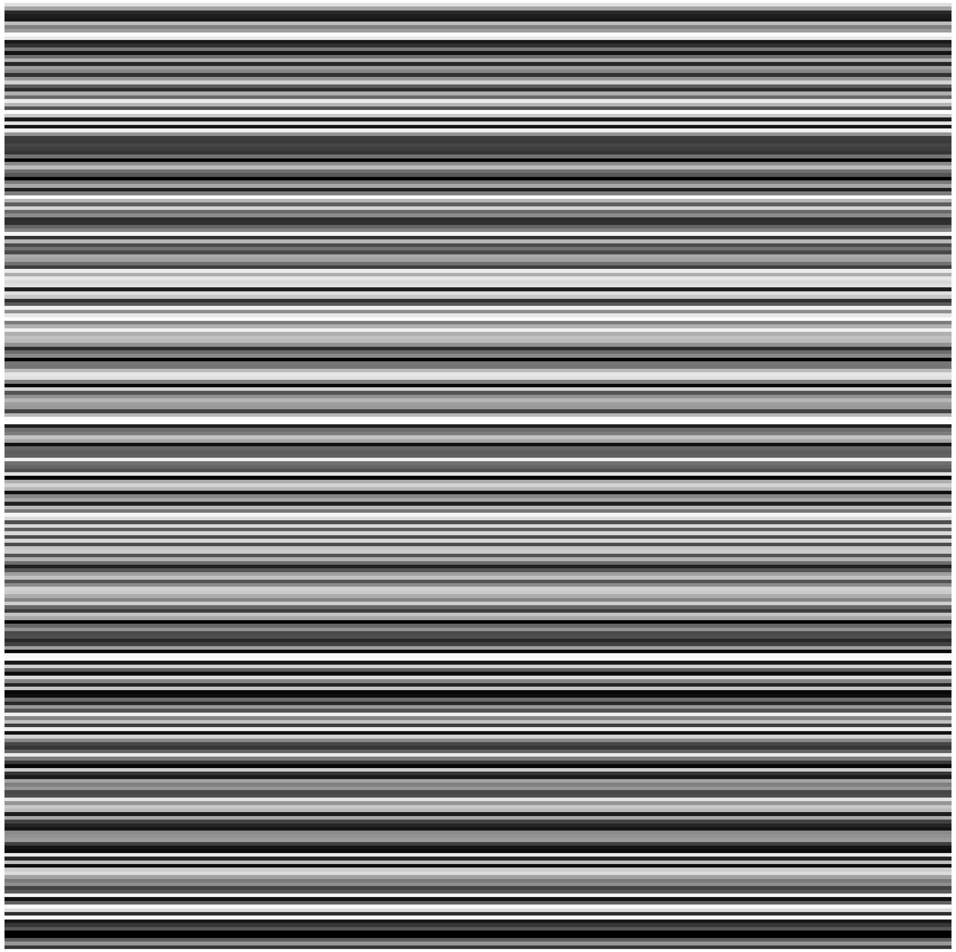
$\bar{l}$ the average length (measured in bits) of the codewords produced by the encoder-编码器码字平均长度. 理想的编码方法-尽量接近下界.

Outline of Lecture 06

- Introduction-简介
- Basics of Information Theory-信息论基础
- Run-Length Coding-游程编码
- Variable-Length Coding-变长编码
  - Shannon–Fano Algorithm-香农-凡诺算法
  - Huffman Coding-赫夫曼编码
  - Adaptive Huffman Coding-自适应赫夫曼编码
- Dictionary-Based Coding-基于字典的编码
- Arithmetic Coding-算术编码
- Lossless Image Compression-无损图像压缩
- Experiments-实验

Run-Length Coding-游程编码

- Run-Length Coding-游程编码



Run-Length Coding-游程编码

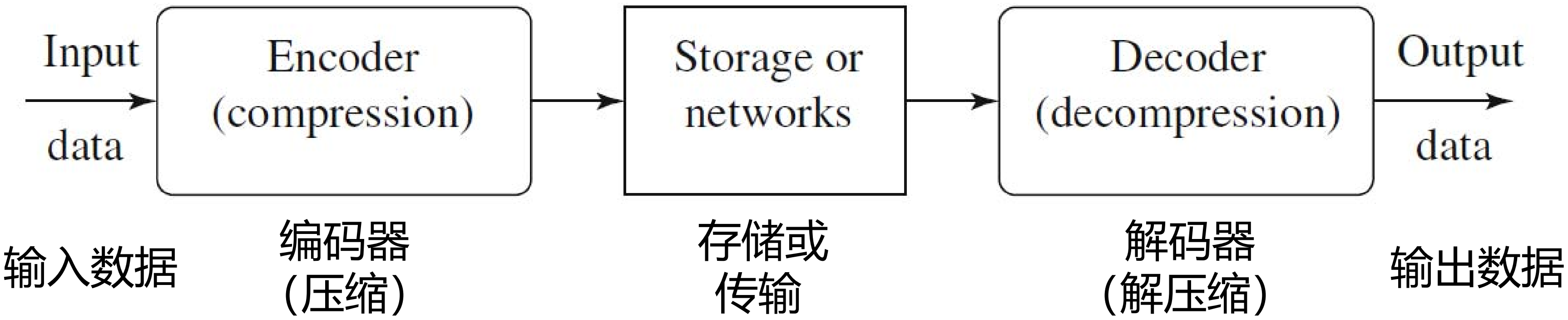
- Run-Length Coding-游程编码
  - It is one of the *simplest forms* of data compression.
  - The basic idea is that if the information source we wish to compress has the property that symbols tend to form *continuous groups*, instead of coding each symbol in the group individually, we can code *one such symbol* and *the length of the group*-字符连续出现，字符+连续出现长度.

    Source：”dfffffeeeeettttrrrrttttt”

    Coded Text: “d1f5e5t4r4t5”
  - Binary Image-One dimension and Two dimension

Introduction-简介

- What is Compression-什么是压缩?-review
  - The process of coding that will effectively reduce the total number of bits needed to represent certain information-
  - 有效地减少表示某种信息所需的比特总数的编码过程.



A General Data Compression Scheme.
一个通用的数据压缩方案

Introduction-简介

- What is Compression-什么是压缩? -review
  - If the compression and decompression processes induce no information loss, then the compression scheme is **lossless**; otherwise, it is **lossy.**
  - 如果压缩和解压过程无信息损失，则压缩方案是无损的，否则是有损的.
  - **Compression ratio-**压缩率:

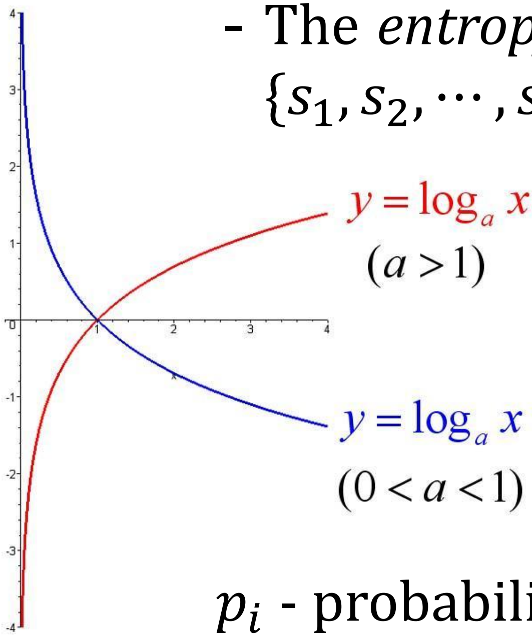
$$Compression\ Ratio = \frac{B_0}{B_1}$$

B_0 - number of bits before compression-压缩前比特数

B_1 - number of bits after compression-压缩后比特数

Basics of Information Theory-信息论基础

◆ Entropy-熵-review



- The *entropy* η of an information source with alphabet $S = \{s_1, s_2, \cdots, s_n\}$-一个具有符号集S的信息源的熵定义为：

$$\eta = H(S) = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i}$$

$$= -\sum_{i=1}^{n} p_i \log_2 p_i$$

p_i - probability that symbol s_i will occur in S-符号s_i发生概率.

$\log_2 \frac{1}{p_i}$ - indicates **the amount of information** (self-information as defined by Shannon) contained in s_i, which corresponds to the number of bits needed to encode s_i-符号s_i包含的信息量（也称自信息），反映出编码s_i所需的比特位数.

Basics of Information Theory-信息论基础

- Entropy and Code Length-熵和编码长度-review
  - The entropy η is a weighted sum of terms $\log_2 1/p_i$; hence it represents the *average amount of information* contained per symbol in the source S
  - 熵反映信息源S中字符的平均信息量.
  - Entropy specifies the *lower bound* for the average number of bits to code each symbol in *S.*
  - 熵指明了对S中每个符号进行编码所需的平均位数的下界.

$$\eta \leq \bar{l}$$

$\bar{l}$ the average length (measured in bits) of the codewords produced by the encoder-编码器码字平均长度. 理想的编码方法-尽量接近下界.

Run-Length Coding-游程编码

- Run-Length Coding-游程编码-review
  - It is one of the *simplest forms* of data compression.
  - The basic idea is that if the information source we wish to compress has the property that symbols tend to form *continuous groups*, instead of coding each symbol in the group individually, we can code *one such symbol* and *the length of the group*-字符连续出现，字符+连续出现长度.

    Source：”dfffffeeeeettttrrrrttttt”

    Coded Text: “d1f5e5t4r4t5”
  - Binary Image-One dimension and Two dimension

Outline of Lecture 06

- Introduction-简介
- Basics of Information Theory-信息论基础
- Run-Length Coding-游程编码
- Variable-Length Coding-变长编码
  - Shannon–Fano Algorithm-香农-凡诺算法
  - Huffman Coding-赫夫曼编码
  - Adaptive Huffman Coding-自适应赫夫曼编码
- Dictionary-Based Coding-基于字典的编码
- Arithmetic Coding-算术编码
- Lossless Image Compression-无损图像压缩
- Experiments-实验

Variable-Length Coding-变长编码

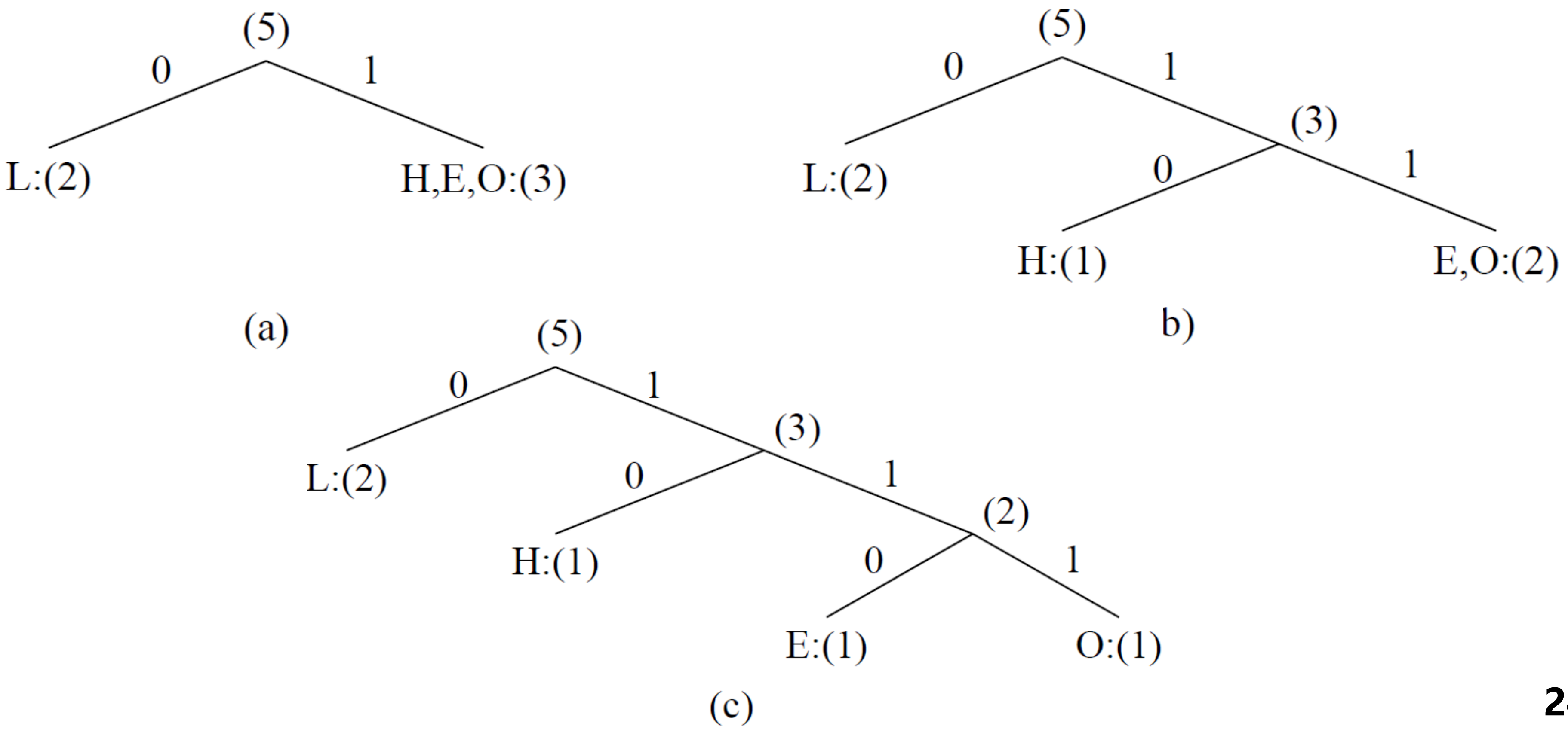
- Shannon–Fano Algorithm-香农-凡诺算法
  - A top-down approach-自顶向下方法.
  a) Sort the symbols according to the frequency count of their occurrences-符号频率由大到小排序.
  b) Recursively divide the symbols into two parts, each with approximately the same number of counts, until all parts contain only one symbol-迭代地将符号分成两部分，各部分中的符号频率的总和相近，直到所有的部分都只含有一个符号为止.
  - Binary tree with single symbol leaves-二叉树，叶结点为单个符号.

Variable-Length Coding-变长编码

◆ Shannon–Fano Algorithm-香农-凡诺算法

- An Example: coding of "HELLO"

Symbol	H	E	L	O
Count	1	1	2	1



Variable-Length Coding-变长编码

- Shannon–Fano Algorithm-香农-凡诺算法
  - An Example: coding of "HELLO"

(5)

0 1

$$\eta = p_L \cdot \log_2 \frac{1}{p_L} + p_H \cdot \log_2 \frac{1}{p_H} + p_E \cdot \log_2 \frac{1}{p_E} + p_O \cdot \log_2 \frac{1}{p_O}$$
$$= 0.4 \times 1.32 + 0.2 \times 2.32 + 0.2 \times 2.32 + 0.2 \times 2.32 = 1.92$$

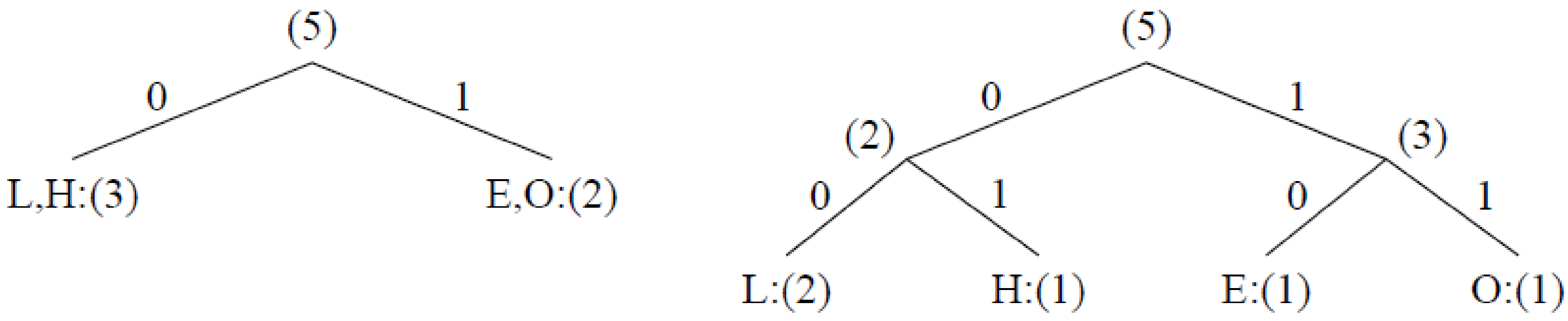
E:(1) O:(1)

Symbol	Count	$\log_2 \frac{1}{p_i}$	Code	Number of bits used
L	2	1.32	0	2
H	1	2.32	10	2
E	1	2.32	110	3
O	1	2.32	111	3
TOTAL number of bits:				10

理论值1.92,香农-凡诺算法10/5=2，与下界接近

Variable-Length Coding-变长编码

- Shannon–Fano Algorithm-香农-凡诺算法
  - Shannon–Fano algorithm is not necessarily unique-香农-凡诺算法结果不唯一.



(a) (b)

Symbol	Count	$\log_2 \frac{1}{p_i}$	Code	Number of bits used
L	2	1.32	00	4
H	1	2.32	01	2
E	1	2.32	10	2
O	1	2.32	11	2
TOTAL number of bits:				10

Variable-Length Coding-变长编码

- Huffman Coding-赫夫曼编码

  - A bottom-up approach-自底向上方法.

  1. Initialization: Put all symbols on a list sorted according to their frequency counts-符号列表降排序.

  2. Repeat until the list has only one symbol left-迭代:

  a) From the list pick two symbols with the lowest frequency counts. Form a Huffman subtree that has these two symbols as child nodes and create a parent node-选频率最小的两符号.

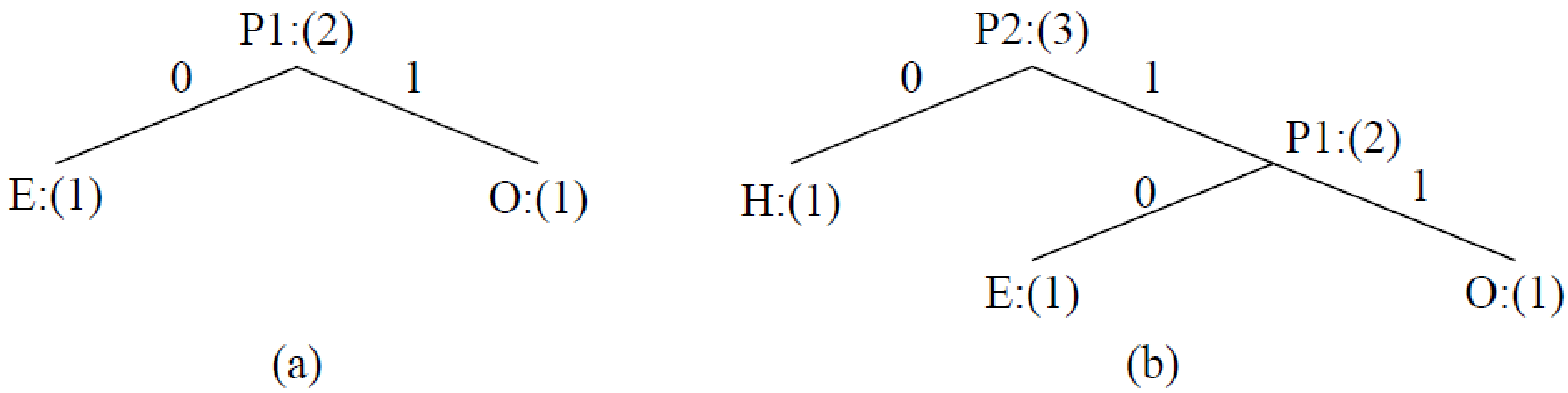
  b) Assign the sum of the children‘s frequency counts to the parent and insert it into the list such that the order is maintained-频数之和作为父结点，父结点有序插入列表.

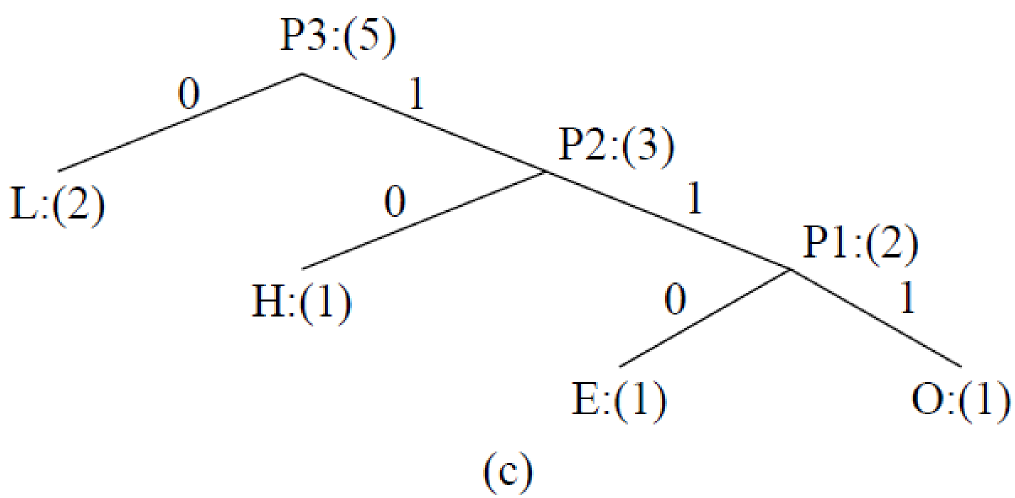
  c) Delete the children from the list-列表中删除选择的符号.

  3. Assign a codeword for each leaf based on the path from the root-根据根到叶的路径分配码字.

Variable-Length Coding-变长编码

◆ Huffman Coding-赫夫曼编码





After initialization: L H E O

After iteration (a): L P1 H

After iteration (b): L P2

After iteration (c): P3

赫夫曼算法:(1+1+2+3+3)/5=2，与下界接近

Variable-Length Coding-变长编码

- Huffman Coding-赫夫曼编码
  - 试计算如下字符集的香农-凡诺和赫夫曼编码总位数.

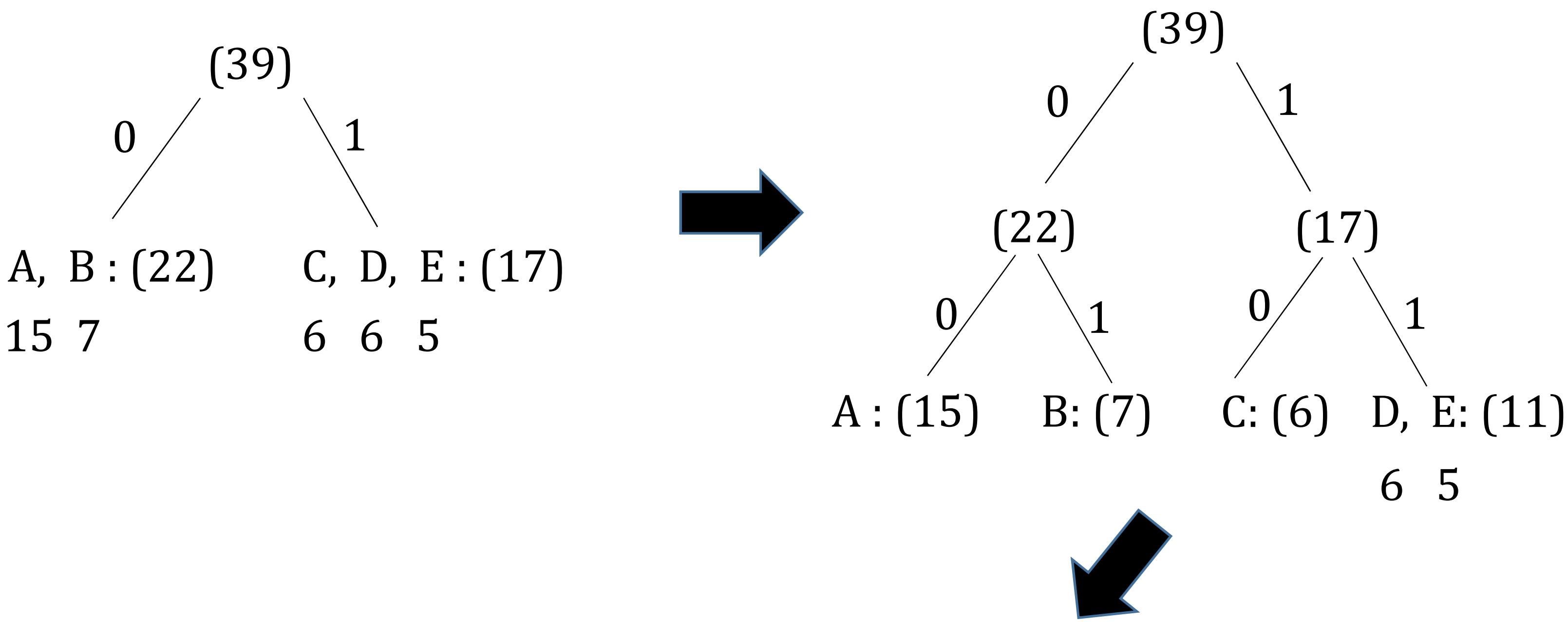
Symbol	A	B	C	D	E
Count	15	7	6	6	5

香农-凡诺算法：89位
赫夫曼算法：87位

Variable-Length Coding-变长编码

- Huffman Coding-赫夫曼编码
  - 香农-凡诺算法

A	B	C	D	E
15	7	6	6	5

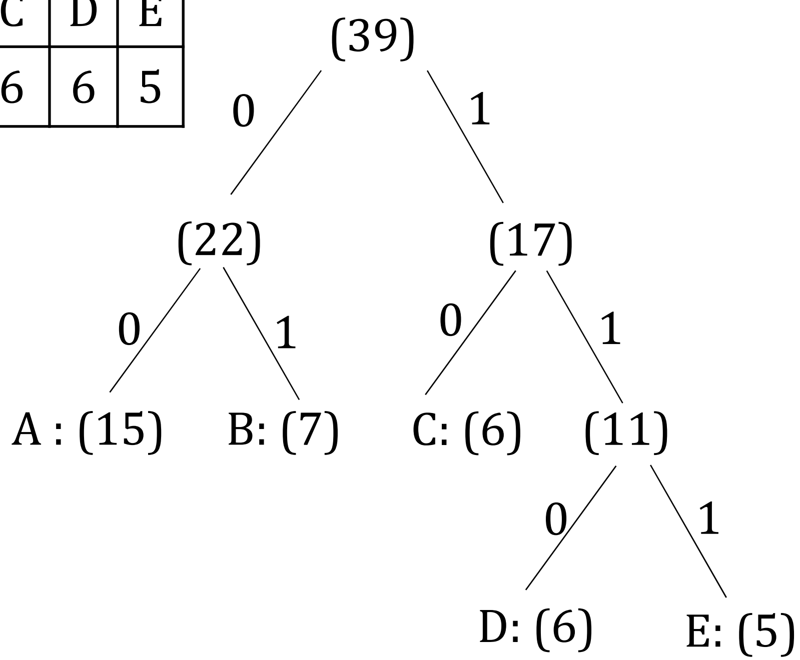


Variable-Length Coding-变长编码

- Huffman Coding-赫夫曼编码

  - 香农-凡诺算法

A	B	C	D	E
15	7	6	6	5



15	A	00	2
7	B	01	2
6	C	10	2
6	D	110	3
5	E	111	3

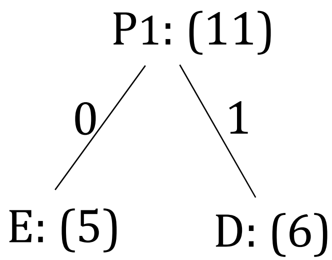
$15\times2+7\times2+6\times2+6\times3+5\times3=89$

Variable-Length Coding-变长编码

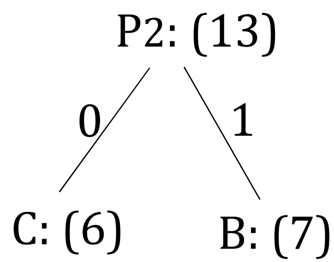
◆ Huffman Coding-赫夫曼编码

- 赫夫曼编码

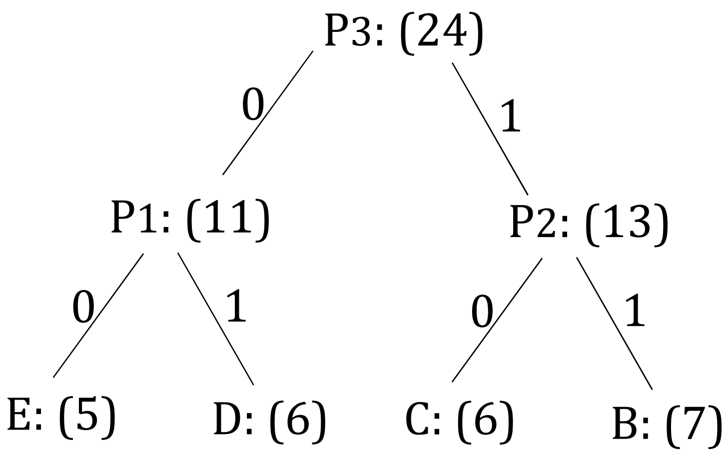
A	B	C	D	E
15	7	6	6	5



A	P1	B	C
15	11	7	6



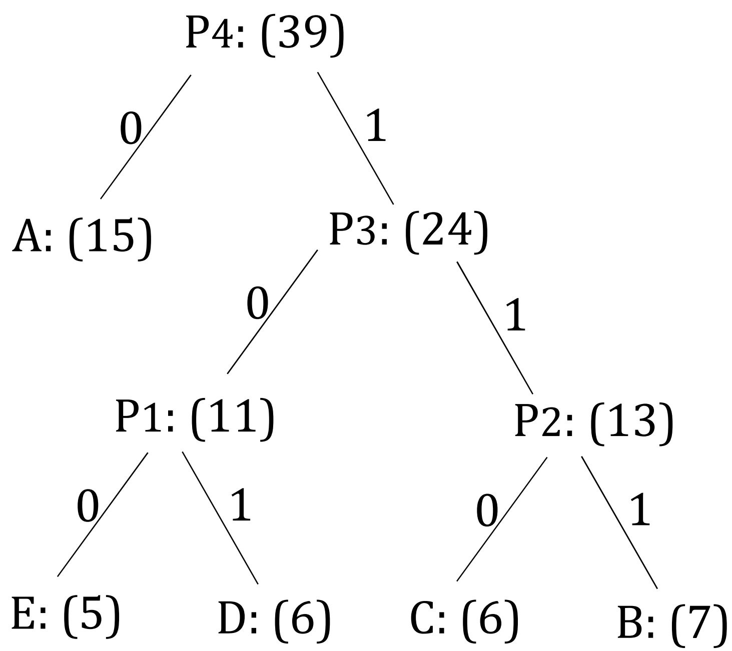
A	P2	P1
15	13	11



Variable-Length Coding-变长编码

- Huffman Coding-赫夫曼编码
  - 赫夫曼编码

P3	A
24	15



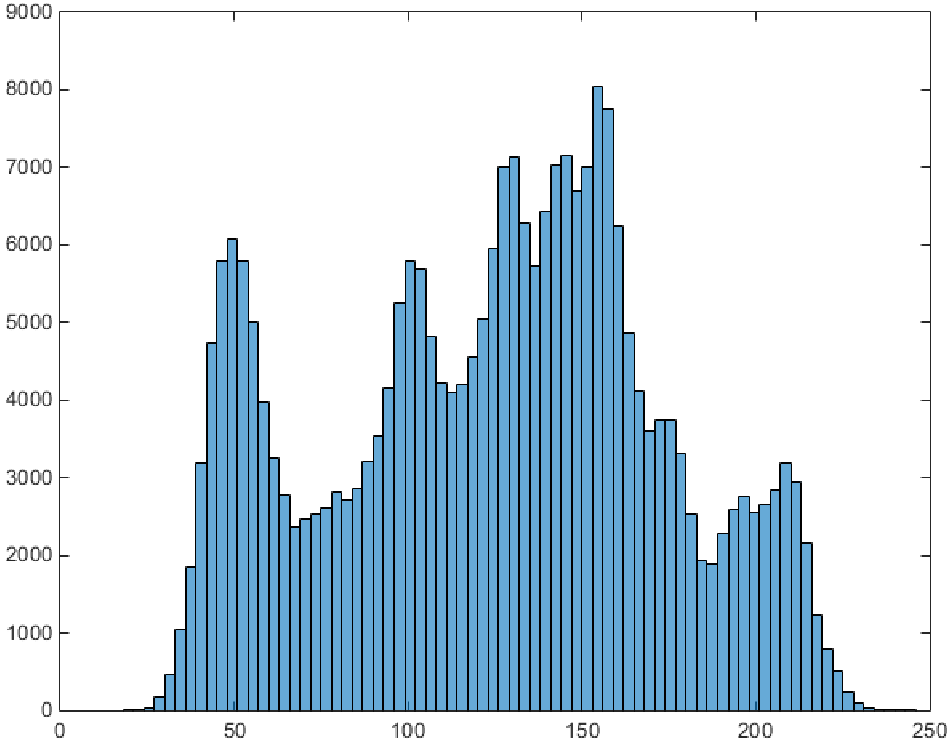
15	A	0	1
7	B	111	3
6	C	110	3
6	D	101	3
5	E	100	3

$15\times 1+7\times 3+6\times 3+6\times 3+5\times 3=87$

Variable-Length Coding-变长编码

◆ Huffman Coding-赫夫曼编码

- 图像压缩应用



压缩率（压缩比） = $\frac{B_0}{B_1}$

$\log_2 \frac{1}{p_i}$

编码效率=$\frac{\text{熵}}{\text{平均码长}}$

使用赫夫曼编码后的平均码长

Variable-Length Coding-变长编码

- Properties of Huffman Coding-赫夫曼编码特性
  - **Unique Prefix Property:** No Huffman code is a prefix of any other Huffman code - precludes any ambiguity in decoding-唯一前缀性(0, 10, 110, 111).
  - **Optimality**: minimum redundancy code - proved optimal for a given data model (i.e., a given, accurate, probability distribution): 最优性-最小冗余编码.

a) 两个频率最低字符：码长一样，码字最后一位不同.

b) 频率高，码字短.

c) 平均编码长度严格小于$\eta + 1$：

$$\eta \leq \bar{l} < \eta + 1$$

局限性：？1）单个字符编码；2）数据压缩前统计信息.

Variable-Length Coding-变长编码

- Extended Huffman Coding-扩展赫夫曼编码
  - **Motivation**: All codewords in Huffman coding have integer bit lengths. It is wasteful when p_i is very large-符号编码整数位，近似1的概率也需1位.
  - Why not group several symbols together and assign a single codeword to the group as a whole?-符号组编码
  - **Extended Alphabet**: For alphabet $S = \{s_1, s_2, \cdots, s_n\}$, if k symbols are grouped together, then the extended alphabet is:

$$S^{(k)} = \{\overbrace{s_1 s_1 \ldots s_1}^{k \text{ symbols}},\ s_1 s_1 \ldots s_2,\ \ldots,\ s_1 s_1 \ldots s_n,\ s_1 s_1 \ldots s_2 s_1,\ \ldots,\ s_n s_n \ldots s_n\}$$

the size of the new alphabet $S^{(k)}$ is n^k-所有可能的组合.

Variable-Length Coding-变长编码

- Extended Huffman Coding-扩展赫夫曼编码
  - It can be proven that the average # of bits for each symbol is -理论上下界:

$$\eta \le \bar{l} < \eta + \frac{1}{k}$$

  - An improvement over the original Huffman coding, but not much($\eta + 1 \rightarrow \eta + 1/k$).
  - **Problem**: If k is relatively large (e.g., $k \geq 3$), then for most practical applications where $n \gg 1$, n^k implies a huge symbol table -- impractical-扩展符号集过大.

Variable-Length Coding-变长编码

- Extended Huffman Coding-扩展赫夫曼编码

  - 实例：

Symbol	A	B	C
Probability	0.6	0.3	0.1

$$\eta = -\sum_i p_i \log_2 p_i = -0.6 \times \log_2 0.6 - 0.3 \times \log_2 0.3 - 0.1 \times \log_2 0.1 \approx 1.2955.$$

A: 0； B: 10； C: 11

平均码长：$0.6 \times 1 + 0.3 \times 2 + 0.1 \times 2 = 1.4$ 位/符号

Variable-Length Coding-变长编码

◆ Extended Huffman Coding-扩展赫夫曼编码

- 实例：

Symbol	A	B	C
Probability	0.6	0.3	0.1

$$\eta = -\sum_i p_i \log_2 p_i = -0.6 \times \log_2 0.6 - 0.3 \times \log_2 0.3 - 0.1 \times \log_2 0.1 \approx 1.2955.$$

A: 0； B: 10； C: 11

平均码长：$0.6 \times 1 + 0.3 \times 2 + 0.1 \times 2 = 1.4$ 位/符号

- 扩展赫夫曼编码 $k = 2$

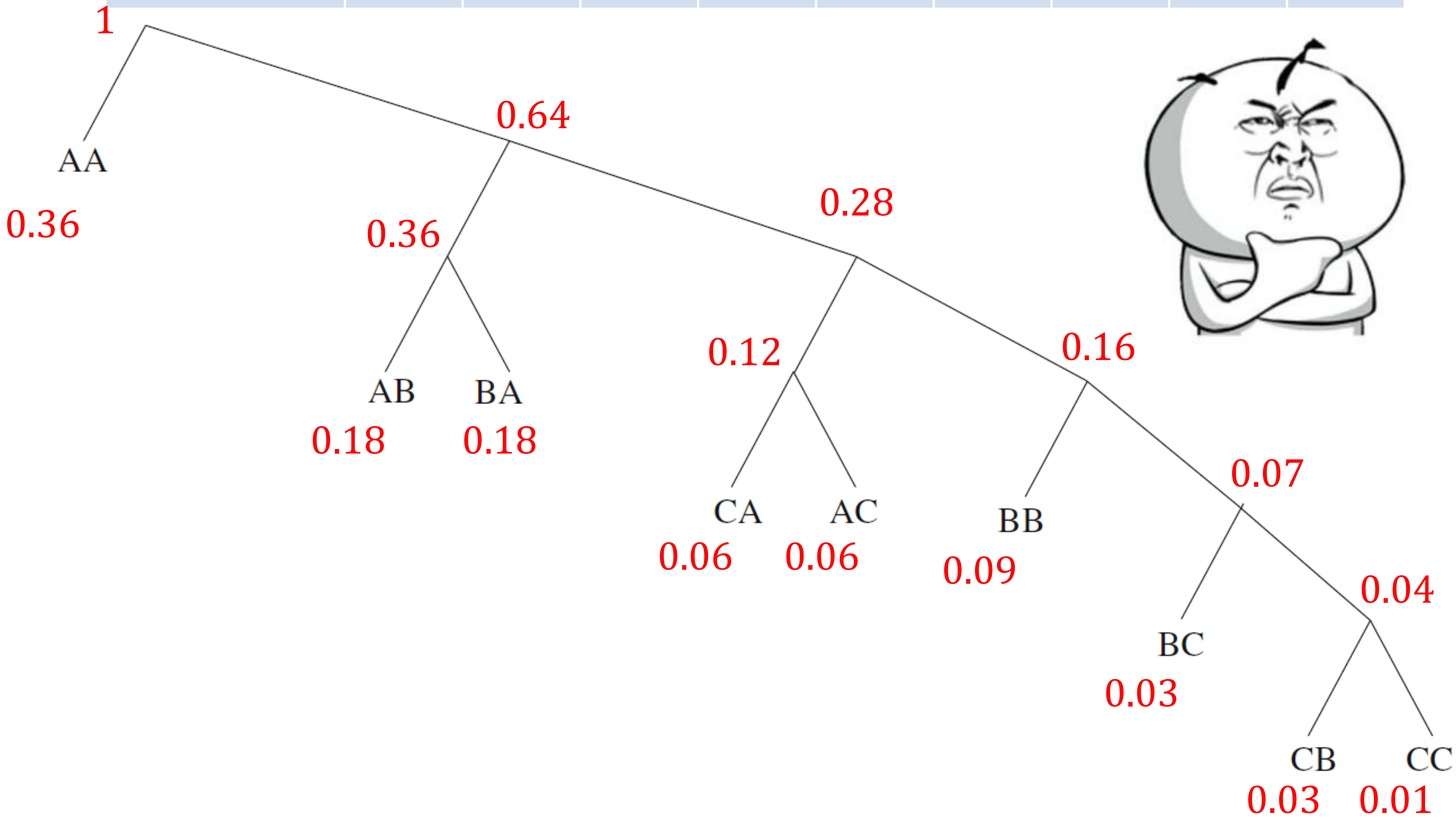
Symbol	AA	AB	BA	CA	AC	BB	BC	CB	CC
Probability	0.36	0.18	0.18	0.06	0.06	0.09	0.03	0.03	0.01

画出扩展赫夫曼编码的树型表示！

Variable-Length Coding-变长编码

- Extended Huffman Coding-扩展赫夫曼编码

Symbol	AA	AB	BA	CA	AC	BB	BC	CB	CC
Probability	0.36	0.18	0.18	0.06	0.06	0.09	0.03	0.03	0.01



Variable-Length Coding-变长编码

◆ Extended Huffman Coding-扩展赫夫曼编码

Symbol group	Probability	Codeword	Bitlength
AA	0.36	0	1
AB	0.18	100	3
BA	0.18	101	3
CA	0.06	1100	4
AC	0.06	1101	4
BB	0.09	1110	4
BC	0.03	11110	5
CB	0.03	111110	6
CC	0.01	111111	6

$$\text{Average} = 0.5 \times (0.36 + 3 \times 0.18 + 3 \times 0.18 + 4 \times 0.06 + 4 \times 0.06 + 4 \times 0.09 + 5 \times 0.03 + 6 \times 0.03 + 6 \times 0.01) = 1.3350.$$

理论最小值：1.2955, 赫夫曼编码：1.4

若事先不知道概率值怎么办?

Variable-Length Coding-变长编码

- Adaptive Huffman Coding-自适应赫夫曼编码
  - The Huffman algorithm requires *prior statistical knowledge* about the information source, and such information is often not available-需要先验统计信息.
  - The solution is to use *adaptive compression algorithms*. The probabilities are no longer based on prior knowledge but *on the actual data received* so far-不基于先验概率，而是当前收到的实际数据.
  - **Adaptive** - As the probability distribution of the received symbols changes, symbols will be given new (longer or shorter) codes-概率变化，码字变化.

Variable-Length Coding-变长编码

- Adaptive Huffman Coding-自适应赫夫曼编码
  - The statistics are gathered and updated dynamically as the data stream arrives-统计信息随着数据流的到达而动态地收集和更新.

```
ENCODER
--------

Initial_code();
while not EOF
  {
      get(c);
      encode(c);
      update_tree(c);
  }
```

```
DECODER
--------

Initial_code();
while not EOF
   {
       decode(c);
       output(c);
       update_tree(c);
    }
```

Procedures for Adaptive Huffman Coding

Variable-Length Coding-变长编码

- Adaptive Huffman Coding-自适应赫夫曼编码
  - **Initial_code:** assigns symbols with some initially agreed upon codes, *without any prior knowledge* of the frequency counts-分配共识码字，不含先验知识.
  - **Update_tree:** constructs an Adaptive Huffman tree.
  - It basically does two things:

  a) increments the frequency counts for the symbols (including any new ones)-字符频率更新.

  b) updates the configuration of the tree-树更新.

  - The encoder and decoder must use **exactly the same initial_code and update_tree** routines-初始码字和树更新完全相同.

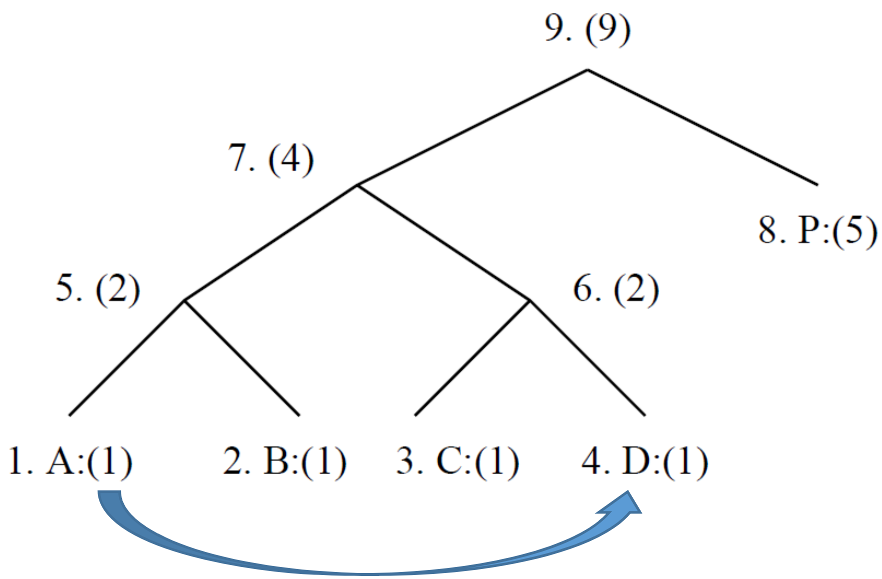
Variable-Length Coding-变长编码

- Adaptive Huffman Coding-自适应赫夫曼编码
  - Nodes are numbered in order from left to right, bottom to top. The numbers in parentheses indicates the count-从左至右，从底至上顺序编号.
  - The tree must always maintain its *sibling property*, i.e., all nodes (internal and leaf) are arranged in the order of increasing counts-赫夫曼树保持计数递增的兄弟特性.
  - If the sibling property is about to be violated, a swap procedure is invoked-违反兄弟特性，节点交换.
  - When a swap is necessary, the farthest node with count N is swapped with the node whose count has just been increased to N+1-计数增加节点交换至最远原相同值结点.

Variable-Length Coding-变长编码

◆ Adaptive Huffman Coding-自适应赫夫曼编码

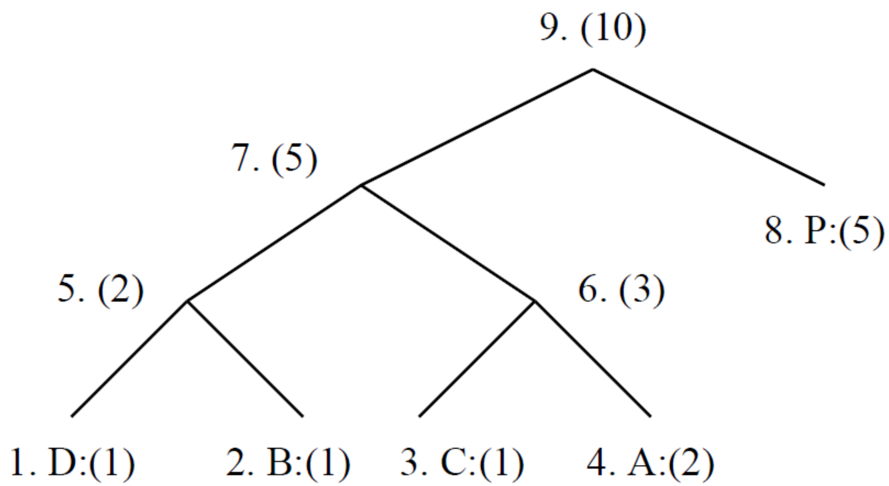
当前赫夫曼树



(a) A Huffman tree

顺序编号、兄弟特性

接收字符A

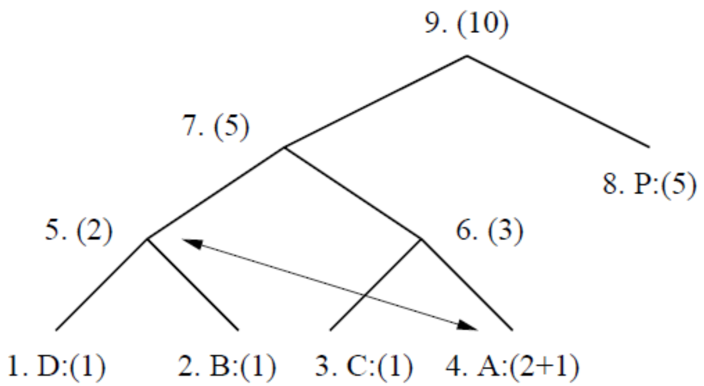


(b) Receiving 2nd ’A’ triggered a swap

违反特性、节点交换

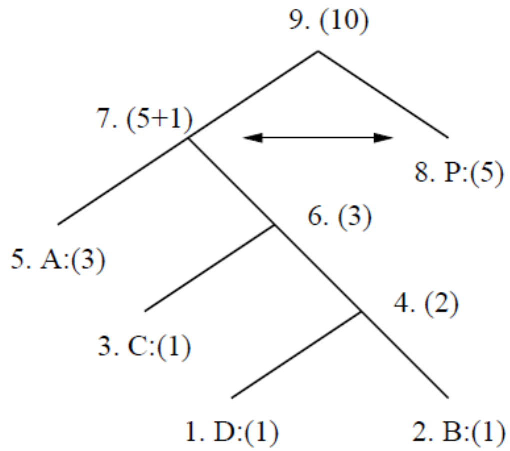
Variable-Length Coding-变长编码

- Adaptive Huffman Coding-自适应赫夫曼编码



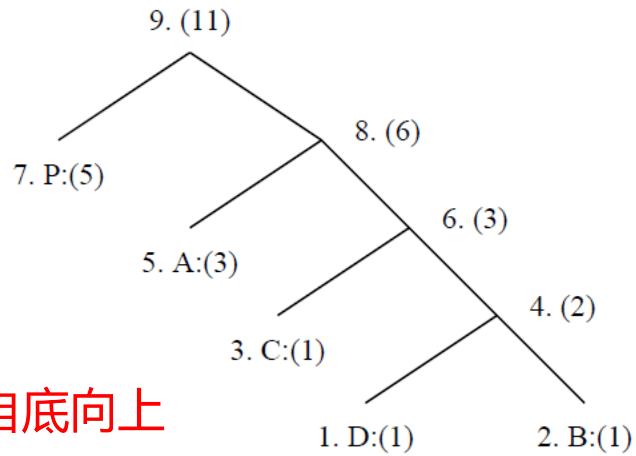
整枝下移

(c−1) A swap is needed after receiving 3rd ’A’



计数更新，保持次序

(c−2) Another swap is needed



整树更新、多次交换、自底向上

(c−3) The Huffman tree after receiving 3rd ’A’

Variable-Length Coding-变长编码

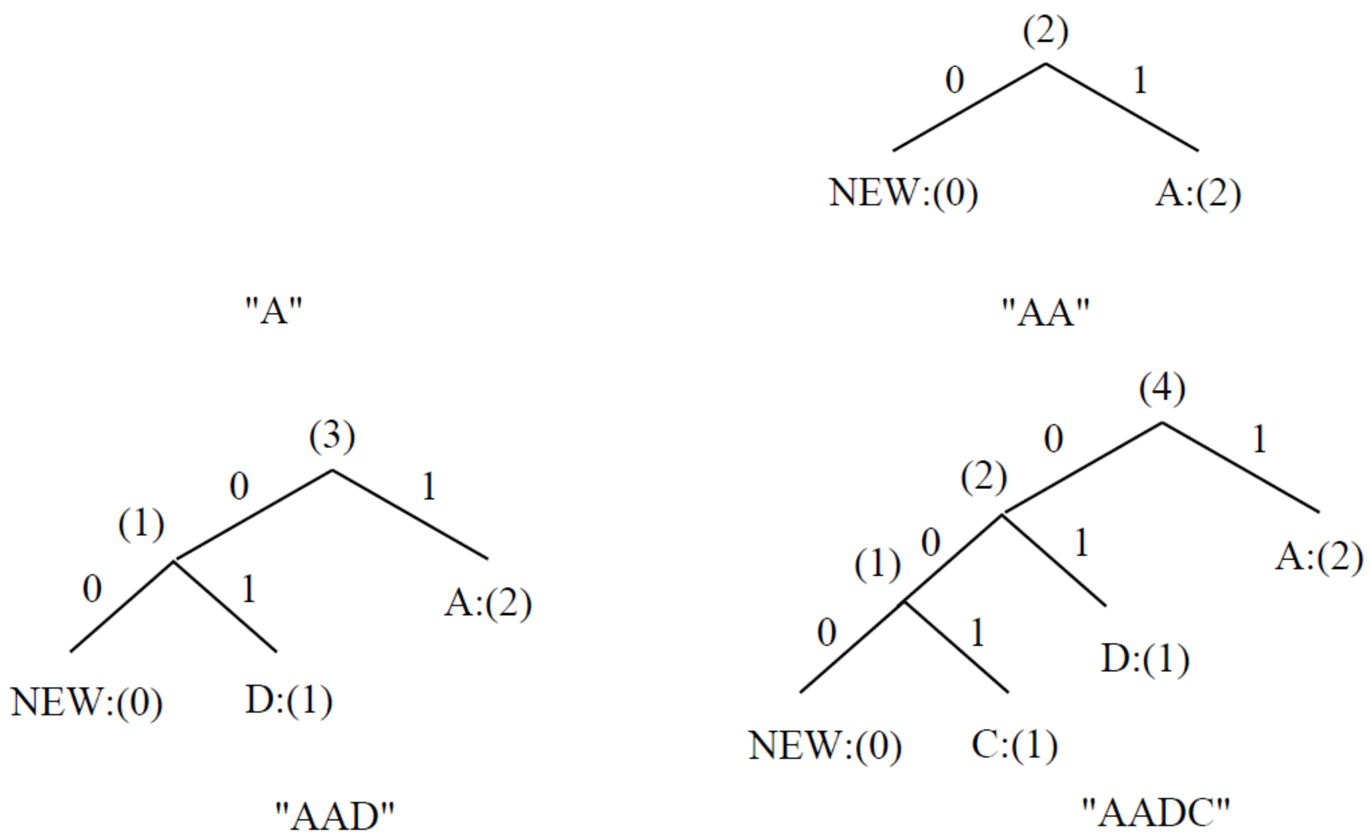
- Adaptive Huffman Coding-自适应赫夫曼编码
  - 实例：字符串AADCCDD的自适应赫夫曼编码
  - Initial_code：约定初始编码

Symbol	Initial code
NEW	0
A	00001
B	00010
C	00011
D	00100
⋮	⋮

  - **An additional rule:** if any character/symbol is to be sent the first time, it must be preceded by a special symbol, NEW. The initial code for NEW is 0-每个新字符发送特殊符号NEW，编码为0，计数始终为0.

Variable-Length Coding-变长编码

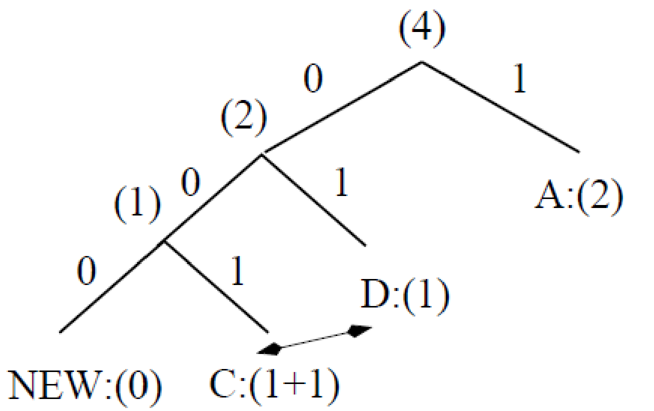
◆ Adaptive Huffman Coding-自适应赫夫曼编码

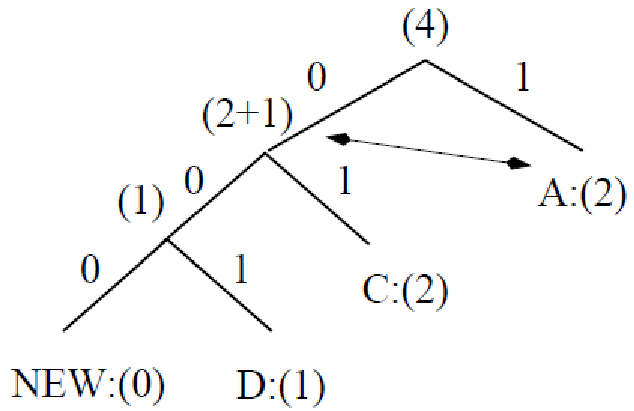


Symbol	NEW	A
Code	0	00001

Variable-Length Coding-变长编码

◆ Adaptive Huffman Coding-自适应赫夫曼编码

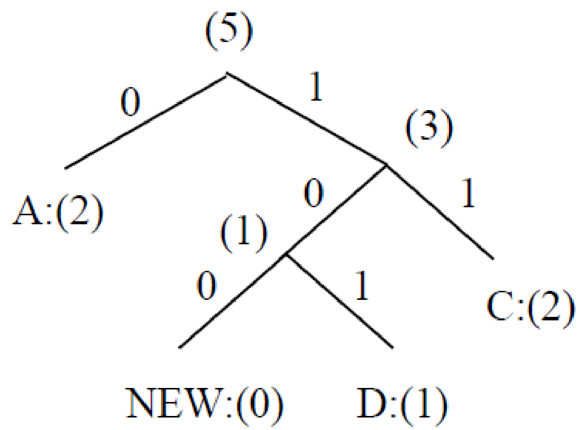




整树更新、多次交换、自底向上

"AADCC" Step 1

"AADCC" Step 2

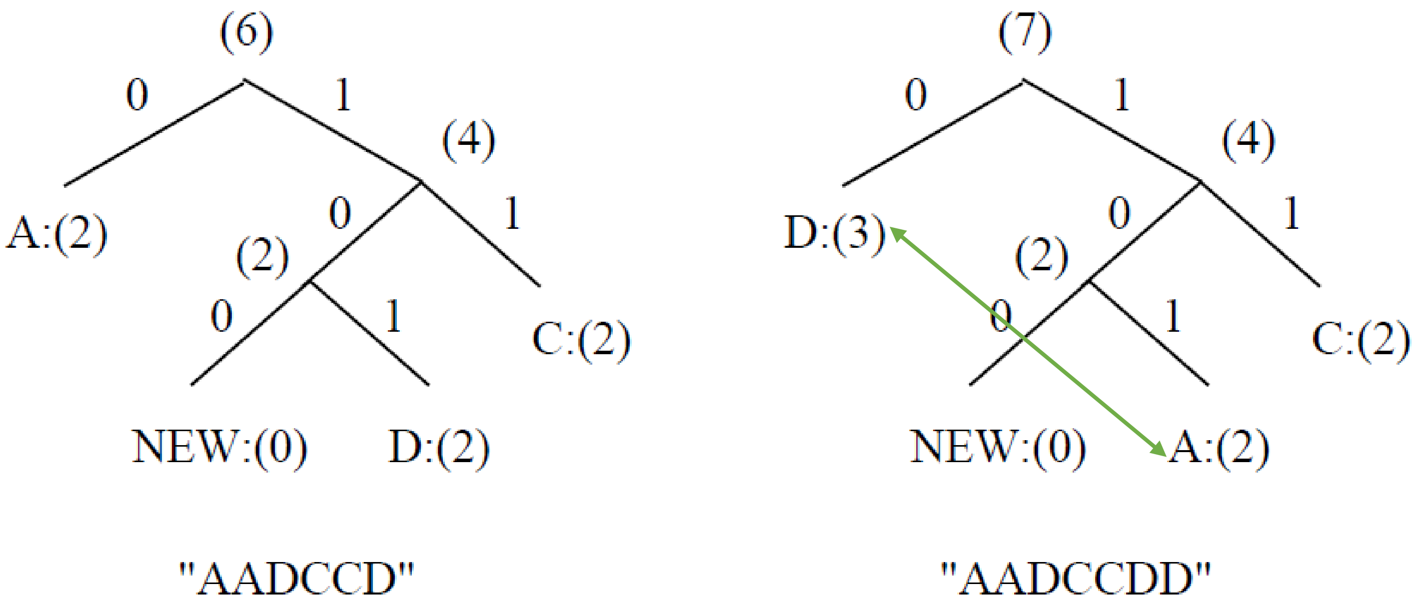


NEW	C	C
00	00011	001

"AADCC" Step 3

Variable-Length Coding-变长编码

◆ Adaptive Huffman Coding-自适应赫夫曼编码



Symbol	NEW	A	A	NEW	D	NEW	C	C	D	D
Code	0	00001	1	0	00100	00	00011	001	101	101

Sequence of symbols and codes sent to the decoder
发送给解码器的编码序列

Experiments & Class Assignments

- Experiments

  - Huffman Coding--*ch07_huffman_coding_demo.m*

- Class Assignments （作为作业提交）

  1、给定字符集S={A(15), B(7), C(6), D(6), E(5)}，利用香农-凡诺和赫夫曼算法计算编码树，并求各算法的编码总位数。

  2、给定字符集S={a,b,c,d}，约定初始编码a=00，b=01，c=10，d=11，画出传送字符串”aabbbacc”的自适应赫夫曼树及二进制编码。