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 07

- 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

Dictionary-Based Coding-基于字典的编码

- Lempel-Ziv-Welch algorithm-LZW编码算法
  - LZW uses fixed-length codewords to represent variable-length strings of symbols/characters that commonly occur together, e.g., words in English text-固定长度码字表示变长度字符串.
  - LZW encoder and decoder build up the same dictionary dynamically while receiving the data text-编解码器动态建立相同的字典.
  - LZW places longer and longer repeated entries into a dictionary, and then emits the code for an element, rather than the string itself, if the element has already been placed in the dictionary-将越来越长的重复字符串插入字典，发送编码而非字符串.

Dictionary-Based Coding-基于字典的编码

- LZW compression algorithm-LZW压缩算法

```
BEGIN
    s = next input character;
    while not EOF
        {
          c = next input character;

          if s + c exists in the dictionary
              s = s + c;  //当前没有发送等式右边的s和c的码字，等下次进入else分支的时候再输出或者读完字符串结束的时候输出
          else
            {
              output the code for s;
              add string s + c to the dictionary with a new code;
              s = c;
            }
        }
    output the code for s;
END
```

字典中存在字符串，
更新当前串(变长)

字典中不存在字符串，
发送旧串码字，增加新串到字典

Dictionary-Based Coding-基于字典的编码

- LZW compression algorithm-LZW压缩算法
  - An Example: LZW compression for string
  - Start with a very *simple dictionary*, initially containing only 3 characters, with codes as follows-3个字符字典:

code	string
1	A
2	B
3	C

  - if the input string is "ABABBABCABABBA", the LZW compression algorithm works as follows-新输入字符串"ABABBABCABABBA"，编码压缩过程如下:

Dictionary-Based Coding-基于字典的编码

- Lempel-Ziv-Welch algorithm-LZW编码算法
  - Input string "ABABBABCABABBA"

```
if s + c exists in the dictionar
   s = s + c;
else
  {
    output the code for s;
    add string s + c to the dict:
    s = c;
  }
```

s	c	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB

AB不存在，发A，加入AB
BA不存在，发B，加入BA
AB存在，不发送，串变长
ABB不存在，送AB，加ABE

总体发送9个码字，而不是
14个字符，无需传字典.
压缩率 = 14/9 = 1.56.
但字典迅速增大

Dictionary-Based Coding-基于字典的编码

- LZW decompression algorithm-解压算法

```
BEGIN
    s = NIL;
    while not EOF
        {
          k = next input code;
          entry = dictionary entry for k;
          output entry;
          if (s != NIL)
              add string s + entry[0] to dictionary
              with a new code; //为了构建字典中没有的字典
          s = entry;
        }
END
```

字典中查找解码

按照从左到右的顺序添加字符到字符串中进行字典构建，与压缩时的方式类似；entry[0]为了引入后续字符

新字符串，增加码字，

Dictionary-Based Coding-基于字典的编码

- LZW decompression algorithm-解码算法
  - An Example: “ABABBABCABABBA”
  - Codewords-1 2 4 5 2 3 4 6 1

```
s = NIL;
while not EOF
   {
     k = next input code;
     entry = dictionary entry for k;
     output entry;
     if (s != NIL)
        add string s + entry[0] to dictionary
        with a new code;
     s = entry;
   }
```

```
 s      k    entry/output     code   string
-------------------------------------------
                               1        A
                               2        B
                               3        C
-------------------------------------------
NIL     1         A
```

第一个字符查找解码，不增加码字

查找解码，增加码字

s+entry[0]

解码输出：ABABBABCABABBA

Dictionary-Based Coding-基于字典的编码

◆ LZW decompression algorithm-解码算法

- Counter-example-“ABABBABCABBABBX...” （压缩过程）

s	c	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	B			
ABB	A	6	10	ABBA
A	B			
AB	B			
ABB	A			
ABBA	X	10	11	ABBAX

s+c 字符串在字典里，需要连接再赋值给s

s+c 字符串不在字典里，直接把字符c赋值给s

```
if s + c exists in the dictionar
   s = s + c;
else
  {
     output the code for s;
     add string s + c to the dict
     s = c;
  }
```

编码器输出：1 2 4 5 2 3 6 10 ...

Dictionary-Based Coding-基于字典的编码

◆ LZW decompression algorithm-解码算法

- Counter-example-“ABABBABCABBABBX...”, (解码过程)

输出编号：1 2 4 5 2 3 6 10 ...

s	k	entry/output	code	string
			1	A
			2	B
			3	C
NIL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	6	ABB	9	CA
ABB	10	???		

```
s = NIL;
while not EOF
    {
      k = next input code;
      entry = dictionary entry for k;
      output entry;
      if (s != NIL)
         add string s + entry[0] to dictionary
         with a new code;
      s = entry;
    }
```

Dictionary-Based Coding-基于字典的编码

- LZW decompression algorithm-解码算法

  - Problem-问题　　又比如字符串“ABABBABCABABBA”，压缩后1 2 4 5 2 3 4 6 1

s	c	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

s	k	entry/output	code	string
			1	A
			2	B
			3	C
NIL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	4	AB	9	CA
AB	6	ABB	10	ABA
ABB	1	A	11	ABBA
A	EOF			

编码器遇到字符+字符串+字符情况，会产生新编码来表示
解码器还没来得及产生这个编码
编码器的动作先于解码器发生

Dictionary-Based Coding-基于字典的编码

- LZW decompression algorithm(improved)-解压算法改进

```
BEGIN
    s = NIL;
    while not EOF
        {
          k = next input code;
          entry = dictionary entry for k;

          /* exception handler */
          if (entry == NULL)
             entry = s + s[0];

          output entry;
          if (s != NIL)
             add string s + entry[0] to dictionary
             with a new code;
          s = entry;
        }
END
```

字典中查找解码

字典中未找到，字串调整

A BB A BB A

新字符串，增加码字

Dictionary-Based Coding-基于字典的编码

- LZW decompression algorithm-解码算法
  - Counter-example-“ABABBABCABBABBX…”,

编码器输出：1 2 4 5 2 3 6 10 …

s	k	entry/output	code	string
			1	A
			2	B
			3	C
NIL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	6	ABB	9	CA
ABB	10	ABBA	10	ABBA
ABBA				

```
s = NIL;
while not EOF
    {
      k = next input code;
      entry = dictionary entry for k;

      /* exception handler */
      if (entry == NULL)
         entry = s + s[0];

      output entry;
      if (s != NIL)
         add string s + entry[0] to dictionary
         with a new code;
      s = entry;
    }
```

Dictionary-Based Coding-基于字典的编码

- LZW algorithm
  - In real applications, the code length l is kept in the range of $[l_0, l_{max}]$. The dictionary initially has a size of 2^{l0}. When it is filled up, the code length will be increased by 1; this is allowed to repeat until $l = l_{max}$ –编码长度逐次增加，直至最大位数.
  - When l_{max} is reached and the dictionary is filled up, it needs to be flushed (as in Unix compress), or to have the LRU (least recently used) entries removed-字典大小有限制，达到最大值时，LZW将失去自适应性. 可字典刷新重新初始化，或删除一些最近最少使用的条目(码字).
  - 常用于TIF格式的图像压缩，平均压缩比在2：1以上，最高压缩比可达到3：1, 压缩和解压缩速度较快.

Outline of Lecture 07

- 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

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Arithmetic coding is a more modern coding method that usually *out-performs* Huffman coding-算术编码优于赫夫曼.
  - Huffman coding assigns each symbol a codeword which has an integral bit length. Arithmetic coding can treat the *whole message as one unit*-赫夫曼编码码字位长整数，整个消息一个单元，实现小数位码长.
  - A message is represented by a *half-open interval* [*a*, *b*) where *a* and *b* are real numbers between 0 and 1. Initially, the interval is [0; 1). When the message becomes longer, the length of the interval shortens, and the number of bits needed to represent the interval increases-区间表示消息，消息变长，区间缩小，位数增加.

Arithmetic Coding-算术编码

◆ Basic Arithmetic Coding - 基本算术编码

```
BEGIN
    low = 0.0;    high = 1.0;    range = 1.0;
    initialize symbol;           // so symbol != terminator

    while (symbol != terminator)
        {
          get (symbol);
          high = low + range * Range_high(symbol);
          low =  low + range * Range_low(symbol);
          range = high - low;
        }

    output a code so that low <= code < high;
END
```

Arithmetic Coding-算术编码

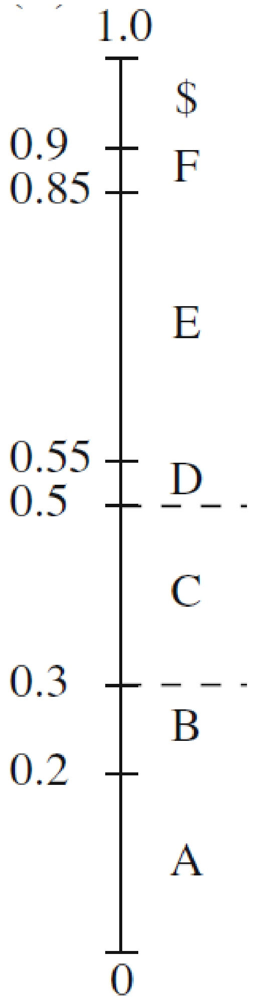
- Basic Arithmetic Coding - 基本算术编码
  - Example-[A, B,C, D, E, F, $]-$消息结束字符

Symbol	Probability	Range	Range_low	Range_high
A	0.2	[0,0.2)	0	0.2
B	0.1	[0.2,0.3)	0.2	0.3
C	0.2	[0.3,0.5)	0.3	0.5
D	0.05	[0.5,0.55)	0.5	0.55
E	0.3	[0.55,0.85)	0.55	0.85
F	0.05	[0.85,0.9)	0.85	0.9
$	0.1	[0.9,1.0)	0.9	1.0

各字符先验概率分布及区间范围

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Example-字符串C, A, E, E, $编码



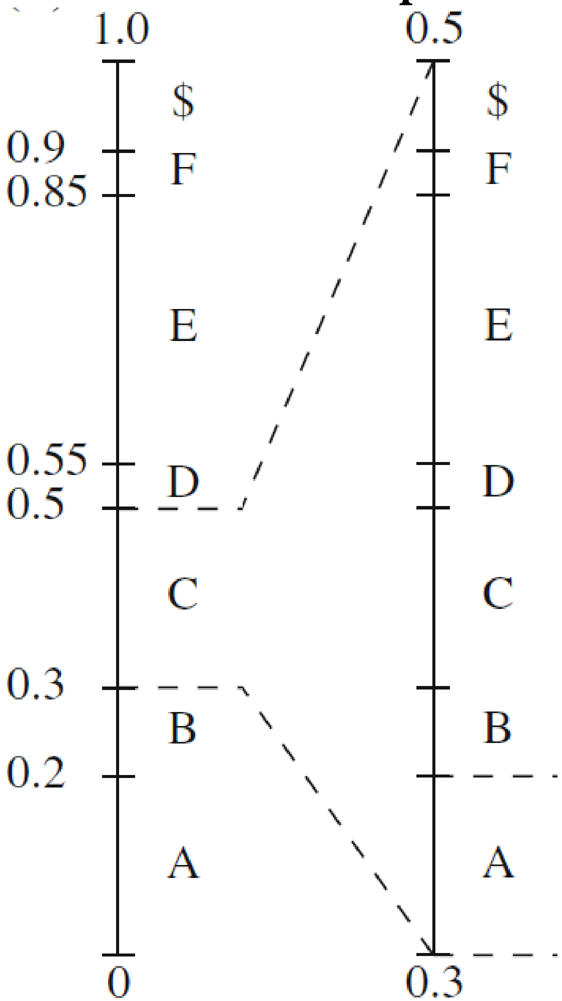
Symbol	Probability	Range	Range_low	Range_high
A	0.2	[0,0.2)	0	0.2
B	0.1	[0.2,0.3)	0.2	0.3
C	0.2	[0.3,0.5)	0.3	0.5
D	0.05	[0.5,0.55)	0.5	0.55
E	0.3	[0.55,0.85)	0.55	0.85
F	0.05	[0.85,0.9)	0.85	0.9
$	0.1	[0.9,1.0)	0.9	1.0

另一种解释：字符占编码区间的百分位，例C：30%-50%
区间不断变小，但字符所占相对百分位不变

L=0.0, H=1.0,R=1.0

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Example-字符串C, A, E, E, $编码




```
high = low + range * Range_high(symbol);
low =  low + range * Range_low(symbol);
range = high - low;
```

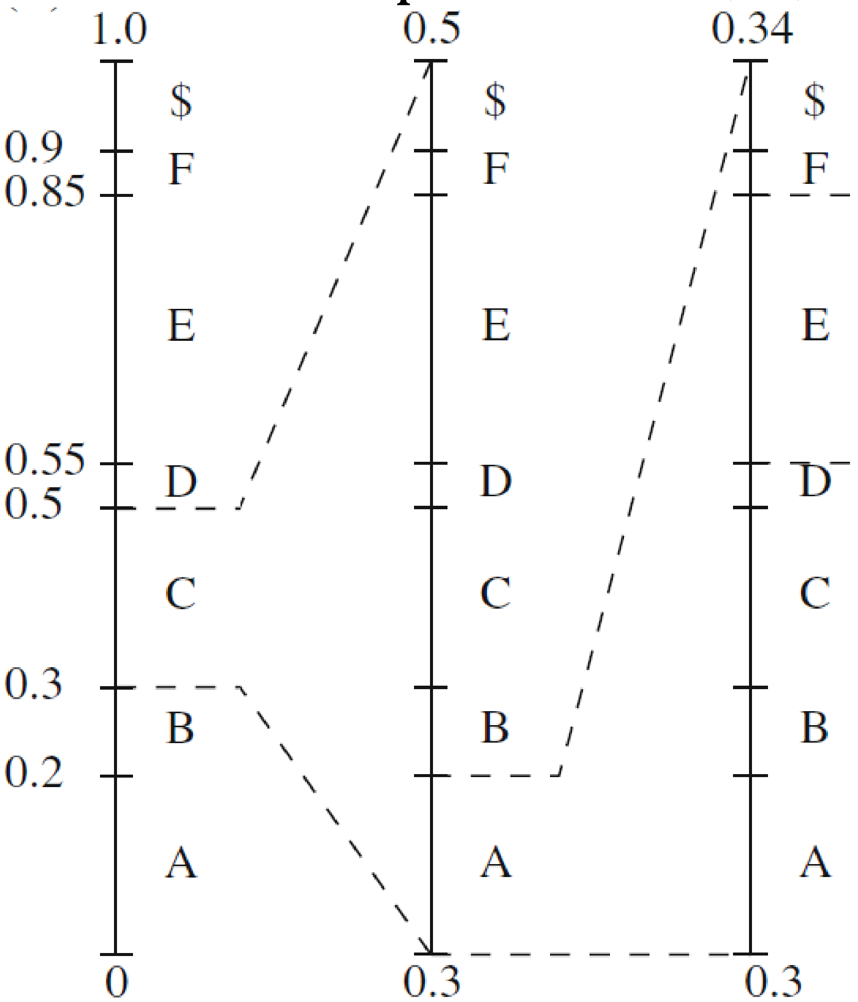
H=L+R*Range_hight(C)=0+1.0*0.5=0.5
L=L+R*Range_low(C)=0+1.0*0.3=0.3

L=0.0, H=1.0,R=1.0 字符C L=0.3, H=0.5,R=0.2

Arithmetic Coding-算术编码

◆ Basic Arithmetic Coding - 基本算术编码

- Example-字符串C, A, E, E, $编码




```
high = low + range * Range_high(symbol);
low =  low + range * Range_low(symbol);
range = high - low;
```

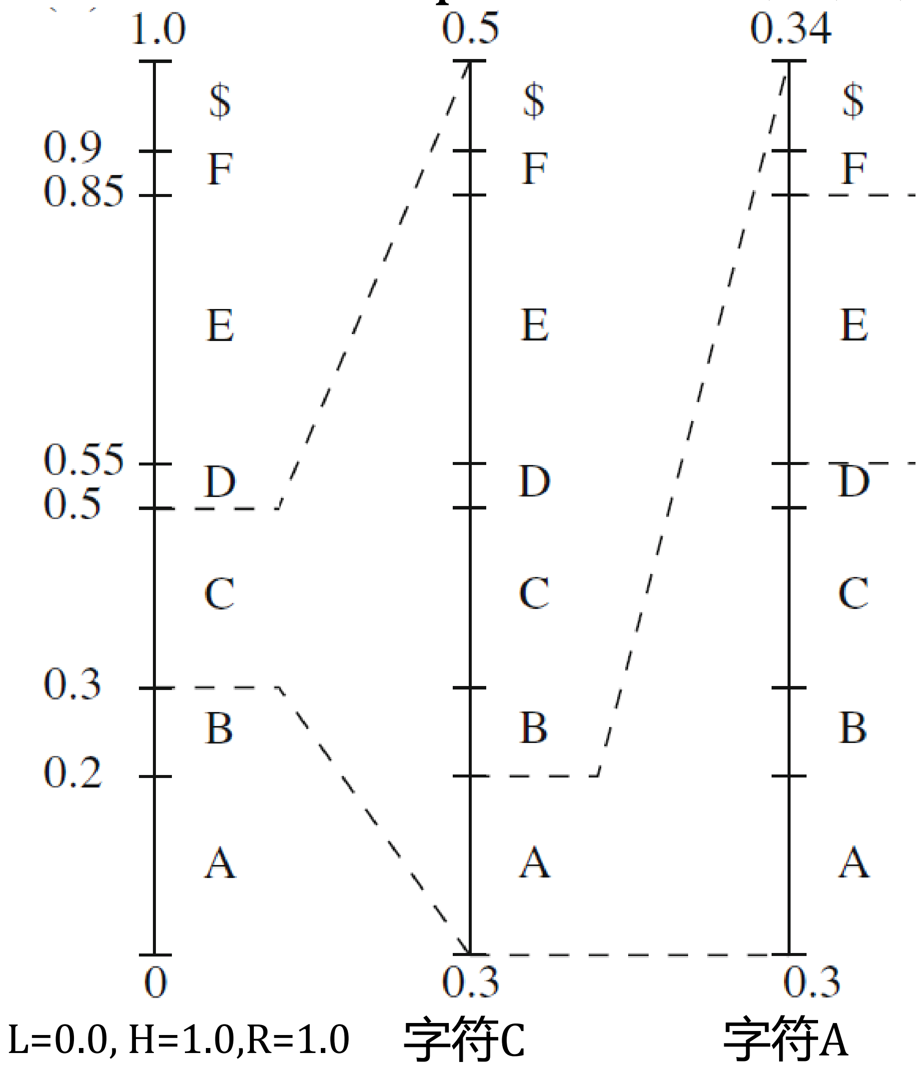
H=L+R*Range_hight(A)=0.3+0.2*0.2=0.34

L=L+R*Range_low(A)=0.3+0.2*0.0=0.3

L=0.0, H=1.0,R=1.0 字符C L=0.3, H=0.5,R=0.2 字符A L=0.30, H=0.34,R=0.04

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Example-字符串C, A, E, E, $编码



Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Example-字符串C, A, E, E, $编码

Symbol	Probability	Range	Range_low	Range_high
A	0.2	[0,0.2)	0	0.2
B	0.1	[0.2,0.3)	0.2	0.3
C	0.2	[0.3,0.5)	0.3	0.5
D	0.05	[0.5,0.55)	0.5	0.55
E	0.3	[0.55,0.85)	0.55	0.85
F	0.05	[0.85,0.9)	0.85	0.9
$	0.1	[0.9,1.0)	0.9	1.0

Symbol	low	high	range
	0	1.0	1.0
C	0.3	0.5	0.2
A	0.30	0.34	0.04
E	0.322	0.334	0.012
E	0.3286	0.3322	0.0036
$	0.33184	0.33220	0.00036

$$\text{range} = P_C \times P_A \times P_E \times P_E \times P_\$ = 0.2 \times 0.2 \times 0.3 \times 0.3 \times 0.1 = 0.00036$$

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Example-字符串C, A, E, E, $编码
  - Low=0.33184, High=0.33220, Range:0.00036-生成码字

```
output a code so that low <= code < high;
```

  - A binary fractional number-十进制数选择容易，但算法需要可转二进制的小数-0.1(?)

Low=0.33184,
High=0.33220

分配1(0.1);
十进制0.5，大于high
所以分配0(0.0)

分配1(0.01);
十进制0.25，小于high
所以分配1(0.01)

```
BEGIN
  code = 0;
  k = 1;
  while (value(code) < low)
    {
      assign 1 to the kth binary fraction bit;
      if (value(code) > high)
        replace the kth bit by 0;
      k = k + 1;
    }
END
```

当前码字十进制值过小

尝试分配1，过大则分配0

最终找到0.01010101，即$2^{-2}+2^{-4}+2^{-6}+2^{-8}$=0.33203125（最后一次分配1，小于high大于low）

Dictionary-Based Coding-基于字典的编码

◆ LZW compression algorithm-LZW压缩算法-Review

```
BEGIN
   s = next input character;
   while not EOF
      {
        c = next input character;

        if s + c exists in the dictionary
           s = s + c;
        else
          {
            output the code for s;
            add string s + c to the dictionary with a new code;
            s = c;
          }
      }
   output the code for s;
END
```

字典中存在字符串，更新当前串(变长)

//当前没有发送等式右边的s和c的码字，等下次进入else分支的时候再输出或者读完字符串结束的时候输出

字典中不存在字符串，发送旧串码字，增加新串到字典

Dictionary-Based Coding-基于字典的编码

- LZW decompression algorithm- 解 压 算 法 - Review

```
BEGIN
    s = NIL;
    while not EOF
        {
          k = next input code;
          entry = dictionary entry for k;
          output entry;
          if (s != NIL)
              add string s + entry[0] to dictionary
              with a new code; //为了构建字典中没有的字典
          s = entry;
        }
END
```

字典中查找解码

按照从左到右的顺序添加字符到字符串中进行字典构建，与压缩时的方式类似；entry[0]为了引入后续字符

新字符串，增加码字，

Dictionary-Based Coding-基于字典的编码

- LZW decompression algorithm(improved)-解压算法改进-Review

```
BEGIN
    s = NIL;
    while not EOF
        {
          k = next input code;
          entry = dictionary entry for k;

          /* exception handler */
          if (entry == NULL)
             entry = s + s[0];

          output entry;
          if (s != NIL)
             add string s + entry[0] to dictionary
             with a new code;
          s = entry;
        }
END
```

字典中查找解码

字典中未找到，字串调整

A BB A BB A

新字符串，增加码字

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码-Review

```
BEGIN
    low = 0.0;    high = 1.0;    range = 1.0;
    initialize symbol;           // so symbol != terminator

    while (symbol != terminator)
        {
          get (symbol);
          high = low + range * Range_high(symbol);
          low =  low + range * Range_low(symbol);
          range = high - low;
        }

    output a code so that low <= code < high;
END
```

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码-Review
  - Example-字符串C, A, E, E, $编码
  - Low=0.33184, High=0.33220, Range:0.00036-生成码字

```
output a code so that low <= code < high;
```

  - A binary fractional number-十进制数选择容易，但算法需要可转二进制的小数-0.1(?)

Low=0.33184,
High=0.33220

分配1(0.1);
十进制0.5，大于high
所以分配0(0.0)

分配1(0.01);
十进制0.25，小于high
所以分配1(0.01)

```
BEGIN
   code = 0;
   k = 1;
   while (value(code) < low)
      {
        assign 1 to the kth binary fraction bit;
        if (value(code) > high)
           replace the kth bit by 0;
        k = k + 1;
      }
END
```

当前码字十进制值过小

尝试分配1，过大则分配0

最终找到0.01010101，即$2^{-2}+2^{-4}+2^{-6}+2^{-8}=0.33203125$（最后一次分配1，小于high大于low）

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Example-字符串C, A, E, E, \$编码0.01010101(仅8位)
  - Entropy-理论下界(11.44)

$$\log_2 \frac{1}{P_C} + \log_2 \frac{1}{P_A} + \log_2 \frac{1}{P_E} + \log_2 \frac{1}{P_E} + \log_2 \frac{1}{P_\$} = \log_2 \frac{1}{\text{range}} = \log_2 \frac{1}{0.00036} \approx 11.44,$$

  - The upper bound of Arithmetic Coding-最坏上界

$$k = \lceil \log_2 \frac{1}{\text{range}} \rceil = \lceil \log_2 \frac{1}{\prod_i P_i} \rceil$$

  - Huffman coding would require 12 bits for CAEE\$-赫夫曼编码总计需要12位，而算术编码仅8位.
  - 算术编码的最坏上界近似赫夫曼编码最好下界
  - 注意！熵指明的是对S集合中每个符号进行编码所需的平均位数的下界

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Example-解码0.01010101-根据先验概率分布信息
  - $(0.01010101)_2=0.33203125$

Symbol	Probability	Range	Range_low	Range_high
A	0.2	[0,0.2)	0	0.2
B	0.1	[0.2,0.3)	0.2	0.3
C	0.2	[0.3,0.5)	0.3	0.5
D	0.05	[0.5,0.55)	0.5	0.55
E	0.3	[0.55,0.85)	0.55	0.85
F	0.05	[0.85,0.9)	0.85	0.9
$	0.1	[0.9,1.0)	0.9	1.0

Symbol	low	high	range
	0	1.0	1.0
C	0.3	0.5	0.2
A	0.30	0.34	0.04
E	0.322	0.334	0.012
E	0.3286	0.3322	0.0036
$	0.33184	0.33220	0.00036

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Example-解码0.01010101-$(0.33203125)_{10}$

二进制码字转十进制

```
BEGIN
   get binary code and convert to decimal value = value(code);
   Do
       {
         find a symbol s so that
              Range_low(s) <= value < Range_high(s);
         output s;
         low = Rang_low(s);
         high = Range_high(s);
         range = high - low;
         value = [value - low] / range;
       }
   Until symbol s is a terminator
END
```

根据先验概率匹配字符

调整区间

0.3≤value≤0.5, 解码C;
更新value=[value-0.3]/0.2
=0.16015625

0.0≤value≤0.2, 解码A;
更新value=[value-0.3]/0.2
=0.80078125

每一次除法表示在放大空间该value值占的比重是多少

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Example-解码0.01010101-$(0.33203125)_{10}$

Value	Output symbol	Range_low	Range_high	range
0.33203125	C	0.3	0.5	0.2
0.16015625	A	0.0	0.2	0.2
0.80078125	E	0.55	0.85	0.3
0.8359375	E	0.55	0.85	0.3
0.953125	$	0.9	1.0	0.1

Symbol	Probability	Range	Range_low	Range_high
A	0.2	[0,0.2)	0	0.2
B	0.1	[0.2,0.3)	0.2	0.3
C	0.2	[0.3,0.5)	0.3	0.5
D	0.05	[0.5,0.55)	0.5	0.55
E	0.3	[0.55,0.85)	0.55	0.85
F	0.05	[0.85,0.9)	0.85	0.9
$	0.1	[0.9,1.0)	0.9	1.0

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - **High-precision**: When it is used to code long sequences of symbols, the tag intervals shrink to a very small range. Representing these small intervals requires very high-precision(i.e., even **32-bit** or **64-bit**)-符号串长，区间小，需高精度数表示及编码
  - **Delay**: The encoder will not produce any output codeword until the *entire sequence* is entered. Likewise, the decoder needs to have the codeword for the *entire sequence* of the input symbols before decoding-整体编解码
  - **Solutions:**?

Arithmetic Coding-算术编码

- Basic Arithmetic Coding - 基本算术编码
  - Scaling and Incremental Coding -符号串长-缩放和增量编码
  - Integer Arithmetic Coding-整数算术编码(单位区间整数表示，避免浮点运算)
  - Binary Arithmetic Coding-二进制算术编码
  - Adaptive Arithmetic Coding-自适应算术编码

Arithmetic Coding-算术编码

- Scaling and Incremental Coding – 缩放增量编码
  - Although the binary representations for the **low**, **high**, and any number within the small interval usually require many bits, they always have the same MSBs (**Most Significant Bits**) - 数值二进制表示需多个位，但编码**最高有效位**相同.

		0	1	2	3	4	5	6	7	8	9
Low	0.5469	0.	1	0	0	0	1	1	0	x	x
High	0.5547	0.	1	0	0	0	1	1	1	x	x

  - Subsequent intervals will always stay within the current interval. Hence, we can *output the common MSBs* and remove them from subsequent considerations-后续区间落在当前区间内，最高有效位相同，后续编码不需考虑这些位.

Arithmetic Coding-算术编码

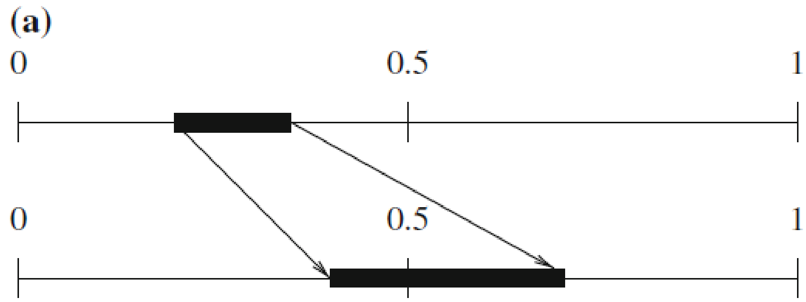
- Scaling and Incremental Coding – 缩放增量编码

  - **E1缩放:** high≤0.5

    a) 发送0至解码器

    b) low = 2 × low, high = 2 × high

    (即左移1位)

  - **E2缩放:** low≥0.5

    a) 发送1至解码器

    b) low = 2 × (low-0.5)

    c) high = 2 × (high-0.5)

    (即左移1位)

  - **E3缩放:** low≥0.25, high≤0.75

    a) low = 2 × (low-0.25)

    b) high = 2 × (high-0.25)

Arithmetic Coding-算术编码

- Scaling and Incremental Coding – 缩放增量编码
  - **Example-**Three symbols A, B, C
  - Input sequence: ACB
  - Coding process?

Symbol	A	B	C
Prob.	0.7	0.2	0.1

基本的算术编码

```
high = low + range * Range_high(symbol);
low =  low + range * Range_low(symbol);
range = high - low;
```

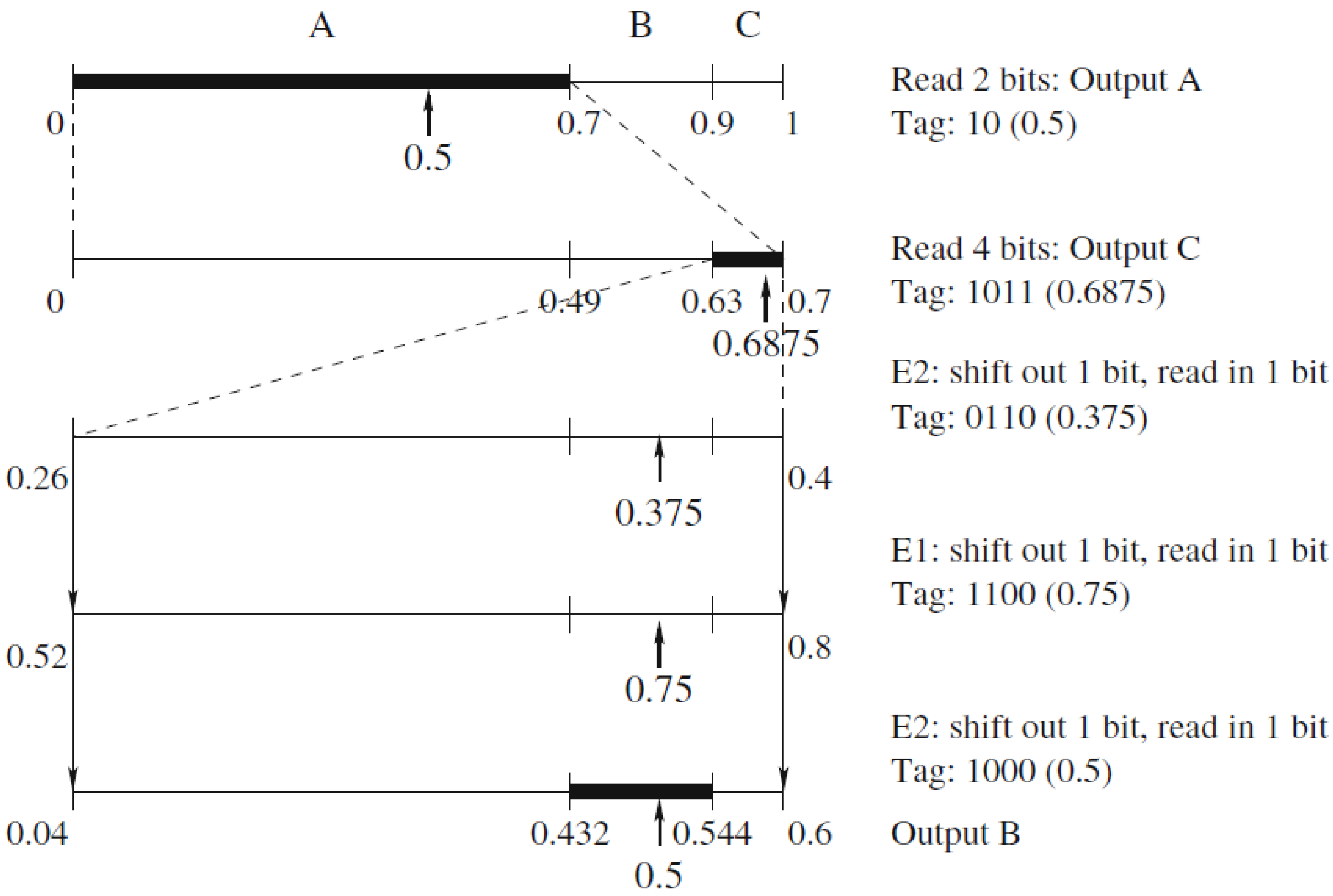
Arithmetic Coding-算术编码

- Scaling and Incremental Coding – 缩放增量编码
  - **Example-**Three symbols A, B, C- Input sequence: ACB
  - Scaling and incremental coding process?

1011(0.6875)

Arithmetic Coding-算术编码

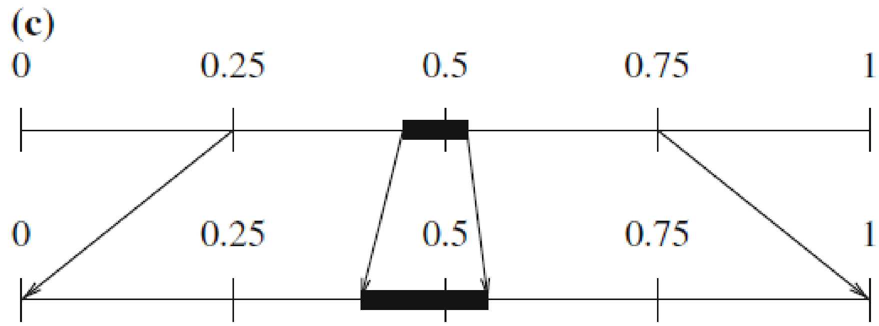
- Scaling and Incremental Coding – 缩放增量编码
  - **Example-**Three symbols A, B, C- Input sequence: ACB
  - Decode: 1011(0.6875)?



Arithmetic Coding-算术编码

◆ Scaling and Incremental Coding – 缩放增量编码

- E3 scaling :



a) low≥0.25 high≤0.75

b) low = 2 ×(low-0.25)

c) high = 2 ×(high-0.25)

- Signaling of the E3 scaling :

a) N E3 scaling steps followed by an E1 is equivalent to an E1 followed by N E2 steps.

b) N E3 scaling steps followed by an E2 is equivalent to an E2 followed by N E1 steps.

c) Postpone until there is an E1 or E2: 0111111, 1000000-延迟信号的发送，直至出现E1或E2，例6次E3后出现E1，则发送0<u>111111</u>.

Outline of Lecture 07

- 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

Lossless Image Compression-无损图像压缩

- Differential Coding of Images - 图像差分编码
  - Given an original image $I(x, y)$, using a simple difference operator we can define a difference image $d(x, y)$ as follows -简单差分图像定义:

$$d(x,y) = I(x,y) - I(x-1,y)$$

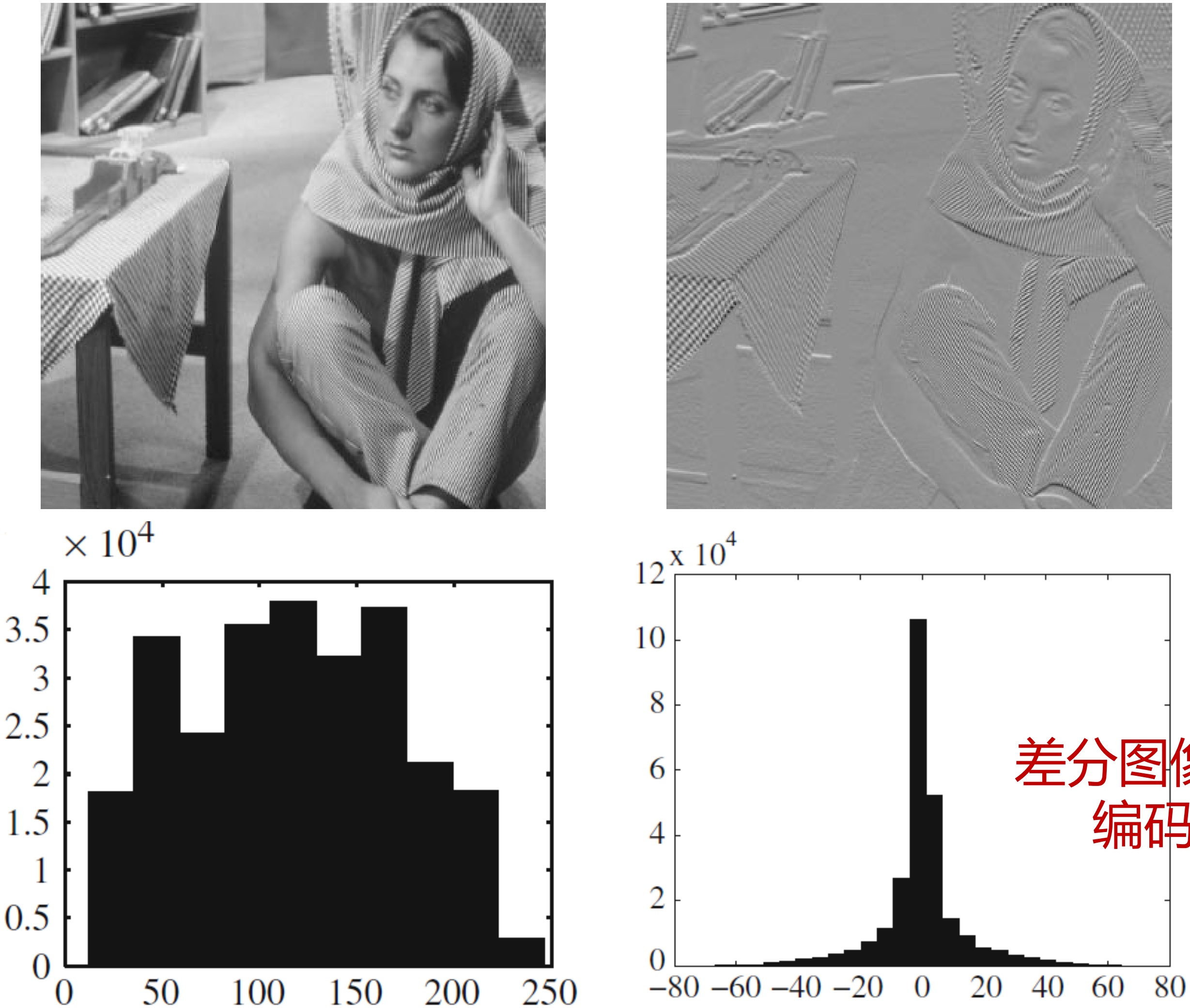
  - The discrete version of the 2-D Laplacian operator to define a difference image $d(x, y)$ as-2D离散拉普拉斯:

$$d(x,y) = 4I(x,y) - I(x,y-1) - I(x,y+1) - I(x+1,y) - I(x-1,y)$$

$I(x\text{-}1, y\text{-}1)$	$I(x, y\text{-}1)$	$I(x\text{+}1, y\text{-}1)$
$I(x\text{-}1, y)$	$I(x, y)$	$I(x\text{+}1, y)$
$I(x\text{-}1, y\text{+}1)$	$I(x, y\text{+}1)$	$I(x\text{+}1, y\text{+}1)$

Lossless Image Compression-无损图像压缩

- Differential Coding of Images - 图像差分编码



差分图像直方图更集中
编码压缩效果好

Distributions for Original versus Derivative Images. (a,b): Original gray-level image and its partial derivative image; (c,d): Histograms for original and derivative images.

Lossless Image Compression-无损图像压缩

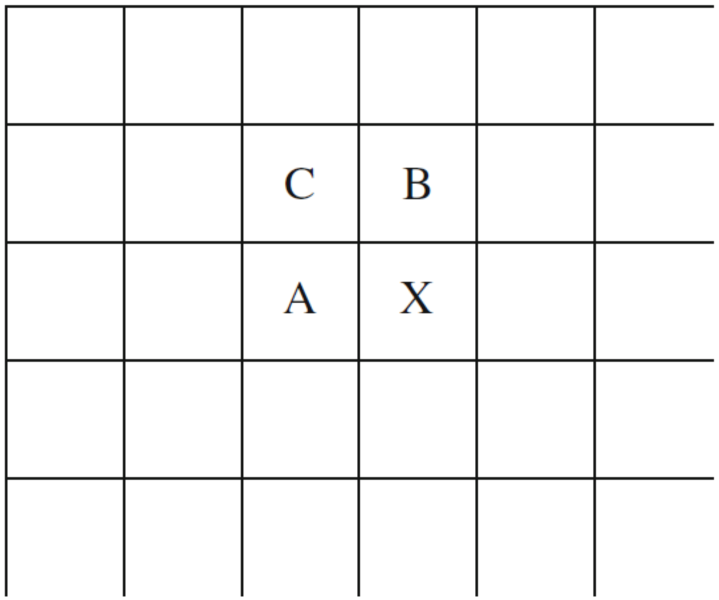
- Lossless JPEG - 无损JPEG
  - Lossless JPEG: A special case of the JPEG image compression-JPEG图像压缩的一个特例(100%质量).
  - The following predictive method is applied on the unprocessed original image (or each color band of the original color image)-彩色图像每个通过单独处理

  a) **Forming a differential prediction:** A predictor combines the values of up to *three neighboring pixels* as the predicted value for the current pixel-三个相邻像素值组合形成当前像素的预测值

  b) **Encoding:** The encoder compares the prediction with the actual pixel value and encodes the *difference* using one of the lossless compression techniques-对当前值与预测值的差值进行压缩编码

Lossless Image Compression-无损图像压缩

- Lossless JPEG - 无损JPEG
  - Differential prediction-差分预测.



当前值X的三个相邻像素A, B, C组合预测

Predictor	Prediction
P1	A
P2	B
P3	C
P4	A + B − C
P5	A + (B − C) / 2
P6	B + (A − C) / 2
P7	(A + B) / 2

预测器(7选1)

I(0, 0)传原值，第一行选P1、第一列选P2

Lossless Image Compression-无损图像压缩

- Lossless JPEG - 无损JPEG
  - Lossless JPEG comparison-效果比较

Compression program	Compression ratio			
	Lena	Football	F-18	Flowers
Lossless JPEG	1.45	1.54	2.29	1.26
Optimal lossless JPEG	1.49	1.67	2.71	1.33
compress (LZW)	0.86	1.24	2.21	0.87
gzip (LZ77)	1.08	1.36	3.10	1.05
gzip-9 (optimal LZ77)	1.08	1.36	3.13	1.05
pack (Huffman coding)	1.02	1.12	1.19	1.00

无损JPEG压缩率较低(1.0~3.0)
更高压缩离率需有损压缩方法

Experiments & Class Assignments

- Experiments

  - Arithmetic Coding--*ch07_arithmetic_coding_demo.m*

- Class Assignments

  1、假设符号表包含符号{0, 1}，当输入为0110011时，利用LZW算法进行压缩编码，给出其对应的字典(符号集合以及编码)和算法的输出。

  2、编程实现赫夫曼编码、扩展赫夫曼编码、自适应赫夫曼编码、LZW编码、基本算术编码、自适应算术编码和无损JPEG编码（至少选一种，语言不限）。