Lecture 24 Entropy and Huffman coding

- Scientific computing beyond the course
- Coding theory and methods for data compression
  - Entropy
  - Encoding/decoding
  - Huffman coding
- Final coverage and review

To read and practise

- Chapter 9 Random numbers and applications
  - How to generate pseudo-random number
  - Monte Carlo simulation, Brownian motion
  - Stochastic differential equation (SDE), and number method.
- Chapter 10 Trigonometric Interpolation and the FFT
  - Fourier transformation
  - Discrete transformation
  - Fast Fourier Transformation (FFT)
- Chapter 11 Compression
  - Discrete Cosine Transformation (DCT)
  - Hoffman coding
  - Applications in Jpeg

- Chapter 13 Optimization

Follow up CS courses

- CP411 Computer Graphics
  - Computing to generate computer images, knowledge of math and scientific computing applies
  - programming in C/C++ and OpenGL, industrial standard library.

- CP467 Image Processing & pattern recognition
  - Image is represented as a matrix
  - Image processing: filtering, FFT, compressing
  - Clustering, classifying, pattern recognition
  - http://bohr.wlu.ca/hfan/cp467/images/class2011/project/video.html

- CP463 Simulation
  - Math modeling, simulation computing

What is Data and Image Compression?

Data compression is the art and science of representing information in a compact form

Why do we need Image Compression?

Still Image

- One page of A4 format at 600 dpi is > 100 MB.
- One color image in digital camera generates 10-30 MB.
- Scanned 3”×7” photograph at 300 dpi is 30 MB.

Digital Cinema

4K×2K×3 ×12 bits/pel = 48 MB/frame or 1 GB/sec
or 70 GB/min, or 4.2TB/hour

2. Lossless and Lossy

Lossless compression: reversible

Lossy compression: irreversible

Rate measures

Bitrate:

$$\frac{\text{size of the compressed file}}{\text{pixels in the image}}$$

Compression ratio:

$$\frac{\text{size of the original file}}{\text{size of the compressed file}}$$

Distortion measures with lossy compression

Mean average error (MAE):

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - x_i\right|$$

Mean square error (MSE):

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - x_i\right)^2$$

4. Code

- Coding is to assign a binary string to represent a symbol, or parttern

  Encoding : message to bitstram
  Decoding: bitstream to message

- Fixed length code -- fix length binary string
  e.g. ASCII, 8 bits per symbol;
  Unicode, 16 bits per symbol

- Vairale length code -- shorter length bit string for higher frequent charaters, longer length string for lower frequent charaters

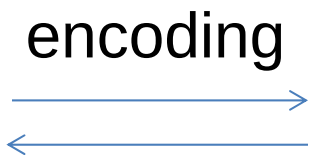
Example

Message: B A B A C A C A DE

ASCII

A - 01000001
B - 01000010
C - 01000011
D - 01000100
E - 01000101

B A B A C A C A DE

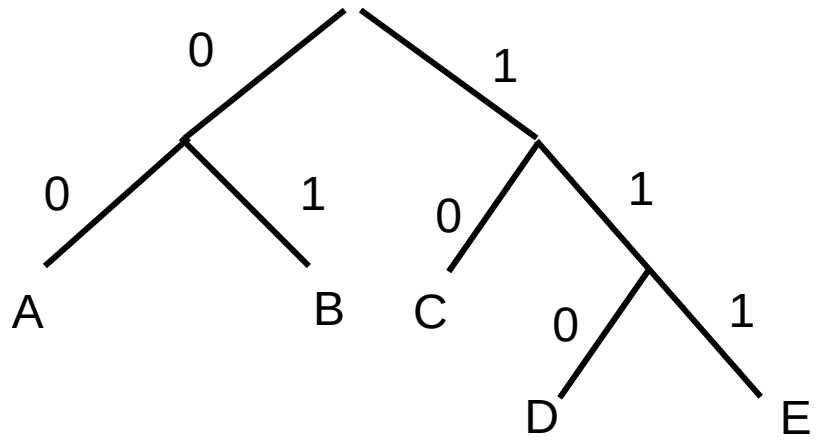


0100001001000001010000100100001000001101
0000010100001101000001010001001000101

Unique prefix property for variable length coding

Unique prefix property: no code is a prefix to any other code

With binary tree expression: all symbols are the *leaf* nodes



A - 00
B - 01
C - 10
D - 110
E - 111

Messaage: BABA CACADE

Bitstream: 010001001000100110111

Total: 22 bits, 2.2 bits per symbol

3. Entropy

Set of symbols (alphabet) $S=\{s_1, s_2, \ldots, s_N\}$,
N is number of symbols in the alphabet.

Probability distribution of the symbols in an information source is $P = \{p_1, p_2, \ldots, p_N\}$

According to Shannon, the entropy H of an information source is defined as follows:

$$H = -\sum_{i=1}^{N} p_i \cdot \log_2(p_i)$$

Entropy

The amount of information in symbol s_i, in other words, the number of bits to code or code length for the symbol s_i:

$$H(s_i) = -\log_2(p_i)$$

The average number of bits for the source S:

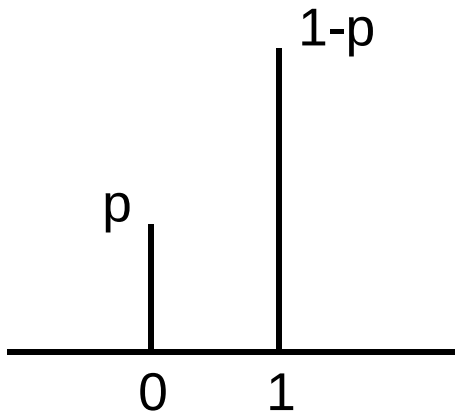
$$H = -\sum_{i=1}^{N} p_i \cdot \log_2(p_i)$$

Entropy for binary source: N=2

S={0,1}

p_0=p

p_1=1-p



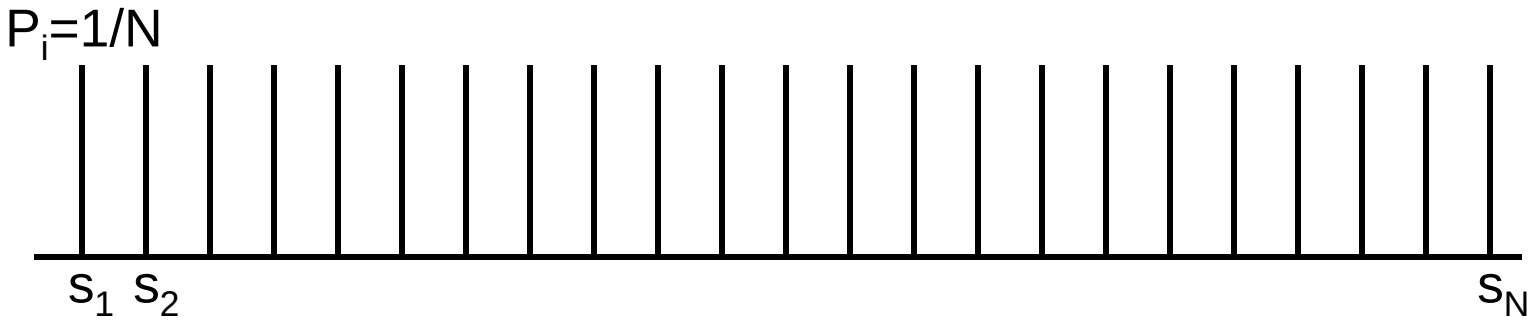
$$H = -(p \cdot \log_2 p + (1-p) \cdot \log_2 (1-p))$$

H=1 bit for p_0=p_1=0.5

Entropy for uniform distribution: p_i=1/N

Uniform distribution of probabilities: p_i=1/N:

$$H = -\sum_{i=1}^{N} (1/N) \cdot \log_2 (1/N) = \log_2 (N)$$



Examples:

N= 2: p_i=0.5; H=$\log_2(2)$ = 1 bit

N=256: p_i=1/256; H=$\log_2(256)$= 8 bits

Entropy of the message

B A B A C A C A D E

Symbol	p_i	$-\log_2(p_i)$
A	4/10	1.3219
B	2/10	2.3219
C	2/10	2.3219
D	1/10	3.3219
E	1/10	3.3219

H = -(0.4*log2(0.4) + 0.2 * log2(0.2) + 0.2 * log2(0.2) + 0.1 * log2(0.1) + 0.1 * log2(0.1))

H = 2.1219

The minimum bit rate for each simple is 2.12

Huffman Code: A bottom-up approach

INIT:
Put all nodes in an OPEN list, keep it sorted all times according their probabilities;.

REPEAT

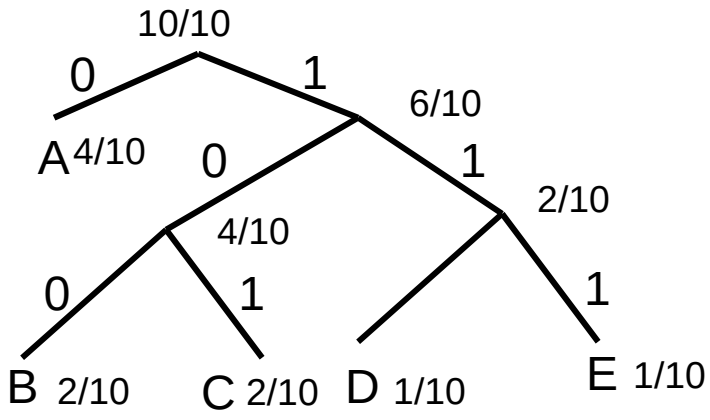
a) From OPEN pick *two* nodes having the *lowest* probabilities, create a parent node of them.

b) Assign the sum of the children's probabilities to the parent node and inset it into OPEN

c) Assign code 0 and 1 to the two branches of the tree, and delete the children from OPEN.

Huffman Code: Example

Binary tree by Huffman coding



Symbol	p_i	$-\log_2(p_i)$	Code	Subtotal
A	4/10	1.3219	0	1*4
B	2/10	2.3219	100	3*2
C	2/10	2.3219	101	3*2
D	1/10	3.3219	110	3*1
E	1/10	3.3219	111	3*1
Total:				22 bits

Huffman Code: Decoding

A - 0
B - 100
C - 101
D - 110
E - 111

Encoding

Message: B A B A C A C A D E

Codes: 100 0 100 0 101 0 101 0 110 111

Bitstream: 1000100010101010110111 (22 bits)

Decoding

Bitstream: 1000100010101010110111

Codes: 100 0 100 0 101 0 101 0 110 111

Message: B A B A C A C A D E