

Data Compression: Huffman Coding

10.1.2 in Weiss (p.395)

4/5/2004

CS216, Spring 2004

1

Why compress files?

4/5/2004

CS216, Spring 2004

2

What is a file?

4/5/2004

CS216, Spring 2004

3

Data Compression



- **Lossless** compression $X = X'$
- **Lossy** compression $X \neq X'$
- **Compression Ratio** $|X|/|Y|$
 - Where $|X|$ is the # of bits in X .

4/5/2004

CS216, Spring 2004

4

Lossy Compression

- Some data is lost, but not too much.

Standards:

- JPEG (Joint Photographic Experts Group)
 - stills
- MPEG (Motion Picture Experts Group)
 - Audio and video
- MP3 (MPEG-1, Layer 3)

4/5/2004

CS216, Spring 2004

5

Lossless Compression

- No data is lost.

Standards:

- Gzip, Unix compress, zip, GIF, Morse code

4/5/2004

CS216, Spring 2004

6

Lossless Compression of text

- ASCII = fixed 8 bits per character
 - Example: "hello there"
 - 11 characters * 8 bits = 88 bits
- Can we encode this message using fewer bits?

4/5/2004

CS216, Spring 2004

7

Huffman Coding

- Uses *frequencies* of symbols in a string to build a **prefix code**.
- **Prefix Code** – no code in our encoding is a prefix of another code.

Letter	code
a	0
b	100
c	101
d	11

4/5/2004

CS216, Spring 2004

8

Decoding a Prefix Code

Loop
 start at root of tree
 loop
 if bit read = 1 then go right
 else, go left
 until node is a leaf
 Report character found!
Until end of the message

4/5/2004

CS216, Spring 2004

9

Decode: 11100010100110

Letter	code
a	0
b	100
c	101
d	11

4/5/2004

CS216, Spring 2004

10

Huffman Trees

Cost of a Huffman Tree containing n symbols

$$C(T) = p_1 * r_1 + p_2 * r_2 + p_3 * r_3 + \dots + p_n * r_n$$

Where:

p_i = the probability that a symbol occurs

r_i = the length of the path from the root to the node

4/5/2004

CS216, Spring 2004

11

Letter	Frequency	code
a	.50	0
b	.125	100
c	.125	101
d	.25	11

4/5/2004

CS216, Spring 2004

12