Instructors

- Instructor
  - Richard Ladner
  - ladner@cs.washington.edu
  - 206 543-9347
  - office hours: WTh 11-12
- TA
  - Justin Goshi
  - goshi@cs.washington.edu
  - office hours - TBA

Prerequisites

- CSE 142, 143
- CSE 326 or CSE 373
- Reason for the prerequisites:
  - Data compression has many algorithms
  - Some of the algorithms require complex data structures

Resources

- Text Book
- 490gz Course Web Page
- Papers and Sections from Books
- E-mail list
  - Send mail to majordomo to subscribe

Engagement by Students

- Weekly Assignments
  - Understand compression methodology
  - Due in class on Fridays (except midterm Friday)
  - No late assignments accepted except with prior approval
- Programming Projects
  - Experimental comparison of compression methods
  - Modification of compression methods.
  - Build a decoder from an encoder.

Final Exam and Grading

- Final Exam - 8:30-10:20 a.m. Tuesday, March 19, 2002
- Midterm Exam – Friday, February 8, 2002
- Percentages
  - Weekly assignments (25%)
  - Midterm exam (20%)
  - Projects (15%)
  - Final exam (40%)
Basic Data Compression Concepts

- **Lossless compression** \( x = \hat{x} \)
  - Also called entropy coding, reversible coding.
- **Lossy compression** \( x \neq \hat{x} \)
  - Also called irreversible coding.
- **Compression ratio** \( \frac{|x|}{|\hat{x}|} \)
  - \( |x| \) is number of bits in \( x \).

Why Compress

- Conserve storage space
- Reduce time for transmission
  - Faster to encode, send, then decode than to send the original
- Progressive transmission
  - Some compression techniques allow us to send the most important bits first so we can get a low resolution version of some data before getting the high fidelity version
- Reduce computation
  - Use less data to achieve an approximate answer

Why Compress

- Conserve storage space
- Reduce time for transmission
  - Faster to encode, send, then decode than to send the original
- Progressive transmission
  - Some compression techniques allow us to send the most important bits first so we can get a low resolution version of some data before getting the high fidelity version
- Reduce computation
  - Use less data to achieve an approximate answer

Braille

- System to read text by feeling raised dots on paper (or on electronic displays). Invented in 1820s by Louis Braille, a French blind man.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>z</th>
</tr>
</thead>
</table>

and the with mother

th ch gh

Braille Example

Clear text:
Call me Ishmael. Some years ago -- never mind how long precisely -- having \% little or no money in my purse, and nothing particular to interest me on shore, \% I thought I would sail about a little and see the watery part of the world. (238 characters)

Grade 2 Braille in ASCII:
```
call me %s\%mael4, \%'s ye$\%$s ag o -- n' e m9d h[ t g
precisely -- hav+ \% or no m'\%y 9 my purses i & no?+
\%\%icu$\% 6 t\%f\% me on %\%ore1 \%, s\%$\% $\% wd sail
ab a \% i\% see \% wat\%d p \% w4 (203 characters)
```

Compression ratio = 238/203 = 1.17

Lossless Compression

- Data is not lost - the original is really needed.
  - text compression
  - compression of computer binaries to fit on a floppy
- Compression ratio typically no better than 4:1 for lossless compression on many kinds of files.
- Statistical Techniques
  - Huffman coding
  - Arithmetic coding
  - Golomb coding
- Dictionary techniques
  - LZW, LZ77
  - Sequitur
  - Burrows-Wheeler Method
- Standards - Morse code, Braille, Unix compress, gzip, zip, bzip, GIF, JBIG, Lossless JPEG

Lossy Compression

- Data is lost, but not too much.
  - audio
  - video
  - still images, medical images, photographs
- Compression ratios of 10:1 often yield quite high fidelity results.
- Major techniques include
  - Vector Quantization
  - Wavelets
  - Block transforms
  - Standards - JPEG, MPEG
Why is Data Compression Possible

• Most data from nature has **redundancy**
  – There is more data than the actual information contained in the data.
  – Squeezing out the excess data amounts to compression.
  – However, unsqueezing out is necessary to be able to figure out what the data means.
• **Information theory** is needed to understand the limits of compression and give clues on how to compress well.

Information Theory

• Developed by Shannon in the 1940’s and 50’s
• Attempts to explain the limits of communication using probability theory.
• Example: Suppose English text is being sent
  – Suppose a “t” is received. Given English, the next symbol being a “z” has very low probability, the next symbol being a “h” has much higher probability. Receiving a “z” has much more information in it than receiving a “h”. We already knew it was more likely we would receive an “h”.

First-order Information

• Suppose we are given symbols {a₁, a₂, ... , aₘ}.
• P(aᵢ) = probability of symbol aᵢ occurring in the absence of any other information.
  – P(a₁) + P(a₂) + ... + P(aₘ) = 1
• inf(aᵢ) = -log₂ P(aᵢ) bits is the information of aᵢ in bits.

Example

• {a, b, c} with P(a) = 1/8, P(b) = 1/4, P(c) = 5/8
  – inf(a) = -log₂(1/8) = 3
  – inf(b) = -log₂(1/4) = 2
  – inf(c) = -log₂(5/8) = .678
• Receiving an “a” has more information than receiving a “b” or “c”.

First Order Entropy

• The first order entropy is defined for a probability distribution over symbols {a₁, a₂, ... , aₘ}.
  \[ H = -\sum_{aᵢ} P(aᵢ) \log₂(P(aᵢ)) \]
• H is the average number of bits required to code up a symbol, given all we know is the probability distribution of the symbols.
• H is the Shannon lower bound on the average number of bits to code a symbol in this “source model”.
• Stronger models of entropy include context. We’ll talk about this later.

Entropy Examples

• {a, b, c} with a 1/8, b 1/4, c 5/8.
  – H = 1/8 * 3 + 1/4 * 2 + 5/8 * .678 = 1.3 bits/symbol
• {a, b, c} with a 1/3, b 1/3, c 1/3. (worst case)
  – H = 1/3 * (1/3)log₂(1/3) = 1.6 bits/symbol
• {a, b, c} with a 1, b 0, c 0 (best case)
  – H = -1log₂(1) = 0
• Note that the standard coding of 3 symbols takes 2 bits.
Entropy Curve

- Suppose we have two symbols with probabilities \( x \) and \( 1-x \), respectively.

\[
\text{max entropy at } 0.5 = - (x \log x + (1-x) \log (1-x))
\]

A Simple Prefix Code

- \((a, b, c)\) with \(a/8, b/4, c/5\).
- A prefix code is defined by a binary tree.
- Prefix code property:
  - no output is a prefix of another

Decoding a Prefix Code

- Repeat:
  - start at root of tree
  - if read bit = 1 then go right
  - else go left
  - until node is a leaf
  - report leaf
  - until end of the code

Decoding a Prefix Code

- 11000111100

Decoding a Prefix Code

- 11000111100

Decoding a Prefix Code

- 11000111100

Decoding a Prefix Code

- 11000111100

Decoding a Prefix Code

- 11000111100

Decoding a Prefix Code

- 11000111100

Decoding a Prefix Code

- 11000111100
Decoding a Prefix Code

```
0 1
0 1
1 0 0 0 1 1 1 1 0 0
CC
```

Decoding a Prefix Code

```
0 1
0 1
1 1 0 0 1 1 1 1 0 0
CC
```

Decoding a Prefix Code

```
0 1
0 1
1 1 0 0 0 1 1 1 1 0 0
CCa
```

Decoding a Prefix Code

```
0 1
0 1
1 1 0 0 1 1 1 1 0 0
CCa
```

Decoding a Prefix Code

```
0 1
0 1
1 1 0 0 1 1 1 1 0 0
CCa
```
Decoding a Prefix Code

1100011100
ccab

Decoding a Prefix Code

11000111100
ccabccca

How Good is the Code

bit rate = \( \frac{1}{8} \times 2 + \frac{1}{4} \times 2 + \frac{5}{8} \times 1 = \frac{11}{8} = 1.375 \text{ bps} \)
Entropy = 1.3 bps
Standard code = 2 bps

(bps = bits per symbol)