Essay

- Review your peers!
- Review at:
  http://www.crowdgrader.org/crowdgrader/rating/reviews/239
Your grade is
75% evaluations of your work
25% how good your evaluations were

To get your full 25% points:
• Complete 5 reviews or more
• Grade consistently and fairly
• Get “helpful” votes on your reviews
Ghosts in the Machine (part 2)
Formats: Proprietary vs. Open

- Microsoft Office (.doc .docx .ppt .xls)
- Open Document Format (.odt .odp .ods)
  - LibreOffice
- VHS versus BetaMax
- Blu-Ray vs. HD-DVD
- TCP/IP
Hiding Information

- Spam Wars
  - hiding from the filter
  - “Amaz!ng 0ffer”
- CAPTCHA
- Steganography
  - Hiding information inside other data
Encoding (Example)

000001 = 1
000100 = 2
001000 = 4
010000 = 8
100000 = 16

11001 = ?
Encoding (Example)

<table>
<thead>
<tr>
<th>Binary</th>
<th>ASCII</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>000001</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>000100</td>
<td>c</td>
<td>3</td>
</tr>
<tr>
<td>001000</td>
<td>e</td>
<td>5</td>
</tr>
<tr>
<td>010000</td>
<td>g</td>
<td>7</td>
</tr>
<tr>
<td>100000</td>
<td>i</td>
<td>9</td>
</tr>
<tr>
<td>100010</td>
<td>k</td>
<td>11</td>
</tr>
<tr>
<td>100011</td>
<td>l</td>
<td>12</td>
</tr>
<tr>
<td>100101</td>
<td>m</td>
<td>13</td>
</tr>
<tr>
<td>100110</td>
<td>n</td>
<td>14</td>
</tr>
<tr>
<td>100111</td>
<td>o</td>
<td>15</td>
</tr>
<tr>
<td>101001</td>
<td>p</td>
<td>16</td>
</tr>
<tr>
<td>101010</td>
<td>q</td>
<td>17</td>
</tr>
<tr>
<td>101101</td>
<td>r</td>
<td>18</td>
</tr>
<tr>
<td>101110</td>
<td>s</td>
<td>19</td>
</tr>
<tr>
<td>101111</td>
<td>t</td>
<td>20</td>
</tr>
<tr>
<td>110001</td>
<td>u</td>
<td>21</td>
</tr>
<tr>
<td>110010</td>
<td>v</td>
<td>22</td>
</tr>
<tr>
<td>110100</td>
<td>w</td>
<td>23</td>
</tr>
<tr>
<td>110110</td>
<td>x</td>
<td>24</td>
</tr>
<tr>
<td>111000</td>
<td>y</td>
<td>25</td>
</tr>
<tr>
<td>111010</td>
<td>z</td>
<td>26</td>
</tr>
</tbody>
</table>

Message: 01101 10101 10011 01001 00011
What was the message?

A. easy
B. abuzz
C. decoy
D. music
E. stone
Secret Messages

- http://youtu.be/L-v4Awj_p7g
Hiding an image inside an image
If you had to throw away some bits from each pixel in an image, which should you throw away so that the remaining image is as close to the original as possible?

A. The rightmost (low order bits).

B. The leftmost (high order bits).

C. It doesn't matter, only how many bits you throw away.
Data Just Won't Go Away

• Format a disk
  – “Warning!! This operation will erase all information on the disk.”
  – Actually...
    • Disk contains chunks of content indexed with tables
    • Format just erases tables (data is still there)

• Delete data in the cloud?
• Delete data in your phone?
The Cloud

• What is the cloud?
  – Google Docs
  – Google Drive, Dropbox
  – Microsoft Office
  – How are these different?
Will it Last?

- Floppy disks
  - Wear out in a year or two
- Home movies, digital photographs
- Domesday Book from 1086
  - A digital copy made in 1986 was unreadable by 2001
- Digital archives can be actively copied
  - If the equipment still exists...
Recap

- Ghosts in the machine
- Metadata
  - What you see is less than what you get
- Steganography
  - Hiding information in plain sight
- Erased data
  - Might still be around
    - On your hard drive
    - In the cloud
    - On devices
CS Concepts

- Document formats
- Interpreting bits
- ASCII, JPG, MP3, ...
- Metadata
- Representing digital images
- Modeling versus rendering
- OCR
- Sampling rate
- Cloud computing

- Data compression
- Spatial conherence
- Temporal coherence
- TCP/IP
- Processing power in audio/video
- Steganography
- Formatting disks
- Data persistence
Social Issues

- Choosing the right format
- What's really in a document?
- Risks of digital documents
- Power of digital documents
- Kiddie porn
- Handheld movies

- Spam
- Espionage
- Cheating
- Forgery
- Access to data years from now
- Destroying old records/data
• We've seen a few different ways
  – ASCII (8 bits per character)
  – A=1 B=2 ... Z=26 (5 bits per character)
  – UTF-8 (variable bits)
• Can we get any lower than 5 bits per character?
• Jean-Dominique Bauby
  – Editor-in-chief of French *Elle*
  – (Dec. 1995) Stroke and coma
  – (Jan. 1996) Wakes up paralyzed except for left eyelid
  – Writes book *The Diving Bell and the Butterfly* using left eyelid
  – (Mar. 6, 1997) Book published, sells millions
  – (Mar. 9, 1997) Dies of pneumonia
Writing a book with an eyelid

- How would you do it?
Indicating Letters

- ABCDEFGHIJKLMNOPQRSTUVWXYZ  
  - (Alphabet order)
- ESARINTULOMDPCFBVHGJQZYXKW  
  - (French)
- ETAOINSHRDLCUMWFGYPBVKJXQZ  
  - (English)
Comparing Orderings

*Plus ça change, plus c'est la même chose?*

- ABCDEFGHIJKLMNOPQRSTUVWXYZ

- ESPARINTULOMDPCFBVHGJQZYXKW
Comparing Orderings

Plus ça change, plus c'est la même chose?

- ABCDEFGHIJKLMNOPQRSTUVWXYZ

- ESARINTULOMDPCFBVHGJQZYXKW

68

32
An Algorithm

• Spelling by repeated comparison to letters is an algorithm
• Going through letters in a particular order follows a program
• Nurses didn't look this up, they invented it to make their work easier
  – Thinking computationally
• What is **CODING**?
  – It's the conversion of one representation into another.

• What is **COMPRESSION**?
  – It's a change in representation in order to reduce the size of the data (number of bits needed)

• Benefits
  – Reduce storage space needed
  – Reduce transmission costs
    • Latency, bandwidth, charges
What makes compression possible?

- **Redundancy** makes compression possible
  - Recognize repeating patterns
  - Save space by
    - Dictionary
    - Describe pattern
- **Limits of human perception**
  - Can discard data and humans won't notice
How easy is compression?

- Depends on the data
  - Random data is hard to compress
    - (No patterns)
  - Uniform data is easy
    - 01010101010101 is “01x8”
- No universally best compression algorithm
- Better compression is possible when algorithm is finely tuned to the data
Can you always compress without losses?

- Lossless compression is not always possible
  - Suppose it were always possible
  - For any large file:
    - Compress the file (reduces size by at least 1 bit)
    - Take output, compress again
    - Keep doing this
  - End up with 1 bit output, but there are more than 2 possible decompressions
Huffman Code: Lossless Compression

- Uses variable length encoding
  - Few bits to represent common symbols
  - More bits to represent uncommon symbols
- Works best when frequencies vary widely
Huffman Example

• “dog cat cat bird bird bird bird fish”

<table>
<thead>
<tr>
<th>Symbol</th>
<th>dog</th>
<th>cat</th>
<th>bird</th>
<th>fish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>1/8</td>
<td>2/8</td>
<td>4/8</td>
<td>1/8</td>
</tr>
<tr>
<td>Original encoding</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Huffman encoding</td>
<td>110</td>
<td>10</td>
<td>0</td>
<td>111</td>
</tr>
</tbody>
</table>

• Expected size:
  - Original: 1/8*2 + 2/8*2 + 4/8*2 + 1/8*2 = 2 bits/symbol
  - Huffman: 1/8*3 + 2/8*2 + 4/8*1 + 1/8*3 = 1.75 bits/symbol
Huffman Example

How many bits are saved by using the Huffman code above for “bird cat bird bird bird cat”?

A. 0  B. 1  C. 2  D. 3  E. 4

<table>
<thead>
<tr>
<th>Symbol</th>
<th>dog</th>
<th>cat</th>
<th>bird</th>
<th>fish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>1/8</td>
<td>2/8</td>
<td>4/8</td>
<td>1/8</td>
</tr>
<tr>
<td>Original encoding</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Huffman encoding</td>
<td>110</td>
<td>10</td>
<td>0</td>
<td>111</td>
</tr>
</tbody>
</table>
Huffman Code: Data Structure

- Binary (Huffman) tree
  - Represents the Huffman code
  - Edge $\rightarrow$ code (0 or 1)
  - Leaf $\rightarrow$ symbol
  - Path to leaf $\rightarrow$ encoding
Huffman Code: Overview

- Order symbols with least frequent first
- Build a tree step by step...
- Use binary tree to encode compressed file
  - For each symbol, output 0s and 1s on path from root to leaf with that symbol
  - Length of encoding is length of path to symbol
  - Save binary tree along with output
Building the Tree

• The algorithm (recipe)
  – Place each symbol in leaf
    • Weight is symbol frequency
  – Pick two lowest weight trees L and R (initially leaves)
  – Create new internal node
    • Left child L
    • Right child R
    • Weight on new node is sum of L and R
  – Repeat until there is just one tree
Huffman Tree (1)

EIAECICEIIHCECIAEHCIEIAEEC

A: 3
C: 6
E: 8
H: 2
I: 7
Huffman Tree (2)

EIAECICEIIHCECIAEHCIEIAEEC

A: 3
C: 6
E: 8
H: 2
I: 7
Huffman Tree (3)

EIAECICEIIHCECIAEHCIEIAEEC

A: 3
C: 6
E: 8
H: 2
I: 7
Huffman Tree (4)

EIAECIIEIIHCECEIAIIEHCEIIEIAEEC

A: 3
C: 6
E: 8
H: 2
I: 7
Huffman Tree (5)

EIAECICEIIHCECIAEHCIEIAEEC

A: 3
C: 6
E: 8
H: 2
I: 7
Huffman Tree (6)

EIAECICEIIHCECIAEHCLIEIAEEC

A: 3
C: 6
E: 8
H: 2
I: 7
Huffman Tree (7)

EIAECIIHCECIAEHCIEIAEEC

A: 3
C: 6
E: 8
H: 2
I: 7
Huffman Tree (8)

EIAECICEIIHCECIAEHCIEIAEEC

A: 001
C: 01
E: 11
H: 000
I: 10
Huffman Example

- A: 001
- C: 01
- E: 11
- H: 000
- I: 10

- Input
  - ACE

- Output
  - (001)(01)(11) = 0010111
Decoding (1)

- To decode, take stream of bits
  - Follow paths in tree from root to leaf

0010111
Decoding (2)

- To decode, take stream of bits
  - Follow paths in tree from root to leaf

0010111
To decode, take stream of bits
  - Follow paths in tree from root to leaf
Decoding (4)

- To decode, take stream of bits
  - Follow paths in tree from root to leaf

0010111

A
• To decode, take stream of bits
  – Follow paths in tree from root to leaf

0010111

A
Decoding (6)

- To decode, take stream of bits
  - Follow paths in tree from root to leaf

0010111
AC
Decoding (7)

- To decode, take stream of bits
  - Follow paths in tree from root to leaf

0010111

AC
Decoding (8)

- To decode, take stream of bits
  - Follow paths in tree from root to leaf

00101111

ACE
Huffman Code Properties

- **Prefix code**
  - No code is a prefix of another code
  - Example prefix:
    - dog → 01
    - cat → 011 //not valid prefix code!
  - Always know when code ends
  - No need to send end-of-code special marker

- **Not unique**
  - There can be multiple Huffman codes for same input
  - Sorting weights, some might be equal
• Algorithm is greedy
  – At each step chooses “best” choice
  – Combines trees with smallest weights
• Still yields best overall solution
  – Optimal prefix code based on frequencies
• Better compression is possible with other algorithms
  – Depending on patterns in data
What does 100111 decode to?
A. ICE
B. IAE
C. ACE
D. CEE
E. CAT
Another Example

• “TO BE OR NOT TO BE” (ignore spaces)
End