More Bits and Bytes
Binary Numbers & Huffman Coding

A General Idea

- Digital Information: Detecting the presence or absence of a phenomenon at a specific place and time: PandA
- Phenomena: light, magnetism, charge, mass, color, current, ...
- Detecting depends on phenomenon – but the result must be discrete: it was detected or not; there is no option for “sorta there”

What have we seen digital representations for?
- Colors
- Letters
- Logical Calculations
- Numbers

HW 7 due in a week. Last HW before midterm.

Positional Notation

- Binary numbers, like decimal numbers, use place notation

\[
1101 = 1 \times 1000 + 1 \times 100 + 0 \times 10 + 1 \times 1
= 1 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 10^0
\]
except that the base is 2 not 10

\[
1101 = 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1
= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0
\]

Binary Numbers

- Volunteers? I need 5.
More binary numbers

http://csunplugged.org/binary-numbers

Clickers: Which one represents different numbers?

A. 01 vs. 001
B. 11 vs. 110
C. 10 vs. 0000010
D. 110 vs. 0110

What about other bases for number systems?

Mayan Numbers

To calculate a Mayan number, you need to divide the number into powers of twenty.

What is binary addition? Let’s clicker this logic out!

What is the answer?

001111
+110111

A. 0000110
B. 1000110
C. 1000111
D. 1000101
E. 0000101
Last week: First guest lecture

- Guest Lecture Thursday: Lifelogging & Social Media
- Prof. Whittaker's Home Page
- Testing on invited talks:
  - No homework on them but possible clicker questions or midterm questions that demonstrate:
    - That you were there
    - That you were paying attention
    - That you understood the presentation
    - Come prepared to ask questions!

Clickers. Prof. Whittaker Guest Lecture.

- The main points of Professor Whittaker's guest lecture included that:
  A. Current capacity for digital storage make it possible to really do Lifelogging
  B. Lifelogging is ethically and morally wrong and should be stopped
  C. To make Lifelogs useful, all we have to do is just record everything
  D. All of the above
  E. None of the above

Clickers. ChattyWeb and Piccy Web

- This table from Prof. Whittaker’s guest lecture shows:
  A. Best predictor of student’s grade was GPA coming into the class
  B. How much you used Chatty & Piccy significantly improved your grade.
  C. None of the above.
  D. All of the above.

Professor Whittaker's lecture included this graph. Clickers

- What does the graph show?
  A. People often remember very little about an arbitrary day.
  B. People who have been wearing a SenseCam cannot remember everything.
  C. An interface that combines the information from both the SenseCam and GPS makes it possible to remember more about an arbitrary day.
  D. Surprisingly, when we are talking about improving human memory, an interface showing where you went in your day works better than an interface with information from the SenseCam.
  E. All of the above.

Huffman Coding:
http://en.wikipedia.org/wiki/Huffman_coding

UTF-8: All the alphabets in the world

- Uniform Transformation Format: a variable-width encoding that can represent every character in the Unicode Character set
  - 1,112,064 of them!!!
  - UTF-8 is the dominant character encoding for the World-Wide Web, accounting for more than half of all Web pages.
  - The Internet Engineering Task Force (IETF) requires all Internet protocols to identify the encoding used for character data
  - The supported character encodings must include UTF-8.
How many bits for all of Unicode?

There are 1,112,064 different Unicode characters. If a fixed bit format (like ASCII with its 7 bits) were used, how many bits would you need for each character? (Hint: $2^{10} = 1024$)

A. 10
B. 17
C. 21
D. 32
E. 40

Coding can be used to do Compression

- What is CODING?
  - The conversion of one representation into another
- What is COMPRESSION?
  - Change the representation (digitization) in order to reduce size of data (number of bits needed to represent data)
- Benefits
  - Reduce storage needed
  - Consider growth of digitized data.
  - Reduce transmission cost / latency / bandwidth
  - When you have a 56K dialup modem, every savings in bits counts, SPEED
  - Also consider telephone lines, texting

Coding can be used to do Compression!

What is CODING?
- The conversion of one representation into another

What is COMPRESSION?
- Change the representation (digitization) in order to reduce size of data (number of bits needed to represent data)

Benefits
- Reduce storage needed
- Consider growth of digitized data.
- Reduce transmission cost / latency / bandwidth
- When you have a 56K dialup modem, every savings in bits counts, SPEED
- Also consider telephone lines, texting

What makes it possible to do Compression?

- IN OTHER WORDS: When is Coding USEFUL?
- When there is Redundancy
  - Recognize repeating patterns
  - Exploit using
    - Dictionary
    - Variable length encoding
- When Human perception is less sensitive to some information
  - Can discard less important data

How easy is it to do it?

- Depends on data
  - Random data ⇒ hard
    - Example: 1001110100 ⇒ ?
  - Organized data ⇒ easy
    - Example: 1111111111 ⇒ 1×10

WHAT DOES THAT MEAN?
- There is NO universally best compression algorithm
- It depends on how tuned the coding is to the data you have

Huffman Code: A Lossless Compression

- Use Variable Length codes based on frequency (like UTF does)
- Approach
  - Exploit statistical frequency of symbols
  - What do I MEAN by that? WE COUNT!!!
  - HELPS when the frequency for different symbols varies widely
  - Principle
    - Use fewer bits to represent frequent symbols
    - Use more bits to represent infrequent symbols

Symbol | Dog | Cat | Bird | Fish
--- | --- | --- | --- | ---
Frequency | 1/8 | 1/4 | 1/2 | 1/8
Original Encoding | 00 | 01 | 10 | 11
Huffman Encoding | 2 bits | 2 bits | 2 bits | 2 bits
Huffman Code Example

**“dog cat cat bird bird bird fish”**

Symbol | Dog | Cat | Bird | Fish
--- | --- | --- | --- | ---
Frequency | 1/8 | 1/4 | 1/2 | 1/8
Original Encoding | 00 | 01 | 10 | 11
Huffman Encoding | 2 bits | 2 bits | 2 bits | 2 bits

- Expected size
  - Original ⇒ $1/8\times2 + 1/4\times2 + 1/2\times2 + 1/8\times2 = 2$ bits / symbol
  - Huffman ⇒ $1/8\times3 + 1/4\times2 + 1/2\times1 + 1/8\times3 = 1.75$ bits / symbol
Huffman Code Example

<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>1/4</td>
<td>1/2</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>2 bits</td>
<td>2 bits</td>
<td>2 bits</td>
<td>2 bits</td>
</tr>
<tr>
<td>Huffman Encoding</td>
<td>3 bits</td>
<td>2 bits</td>
<td>1 bit</td>
<td>3 bits</td>
</tr>
</tbody>
</table>

How many bits are saved using the above Huffman coding for the sequence Dog Cat Bird Bird Bird?
A. 0  B. 1  C. 2  D. 3  E. 4

Huffman Code Algorithm: Data Structures
- Binary (Huffman) tree
  - Represents Huffman code
  - Edge \( \Rightarrow \) code (0 or 1)
  - Leaf \( \Rightarrow \) symbol
  - Path to leaf \( \Rightarrow \) encoding
- Example
  - A = “11”, H = “10”, C = “0”
  - Good when ???
- Want to efficiently build a binary tree

Huffman Code Algorithm Overview
- Order the symbols with least frequent first (will explain)
- Build a tree piece by piece…
- Encoding
  - Calculate frequency of symbols in the message, language
  - JUST COUNT AND DIVIDE BY TOTAL NUMBER OF SYMBOLS
  - Create binary tree representing “best” encoding
  - Use binary tree to encode compressed file
    - For each symbol, output path from root to leaf
    - Size of encoding = length of path
  - Save binary tree

Huffman Code – Creating Tree
- Algorithm (Recipe)
  - Place each symbol in leaf
    - Weight of leaf = symbol frequency
  - Select two trees L and R (initially leafs)
    - Such that L, R have lowest frequencies in tree
    - Which L, R have the lowest number of occurrences in the message?
  - Create new (internal) node
    - Left child \( \Rightarrow \) L
    - Right child \( \Rightarrow \) R
  - New frequency \( \Rightarrow \) \( \text{frequency}(L) + \text{frequency}(R) \)
  - Repeat until all nodes merged into one tree

Huffman Tree Construction 1

Huffman Tree Step 2: can first re-order by frequency
UP to here was a review of what we did last Tuesday.
Huffman Decoding 1

Huffman Decoding 2

Huffman Decoding 3

Huffman Decoding 4

Huffman Decoding 5

Huffman Decoding 6
Huffman Decoding 7

Huffman Code Properties

- Prefix code
  - No code is a prefix of another code
  - Example
    - Huffman("dog") → 01
    - Huffman("cat") → 011 (not legal prefix code)
  - Can stop as soon as complete code found
  - No need for end-of-code marker
- Nondeterministic
  - Multiple Huffman coding possible for same input
  - If more than two trees with same minimal weight

Huffman Coding. Another example.

Huffman Tree Example 2. Step 1

Huffman Tree: TO BE OR NOT TO BE
Clicker: TO BE OR NOT TO BE

Which nodes can get combined next?

A. The brown node worth 2 (R, N) and the E node worth 2
B. The brown node worth 2 and the B node worth 2
C. The brown node worth 2 (R, N) and the B node worth 2
D. All of the above

Huffman Tree: TO BE OR NOT TO BE

Which nodes can get combined next?

A. The brown node worth 4 (R, N, E) and the B node worth 4
B. The brown node worth 4 (R, N, E) and the O node worth 4
C. The B node worth 2 and the T node worth 3
D. All of the above

Huffman Tree: TO BE OR NOT TO BE

Which nodes can get combined next?

A. The brown node worth 4 (R, N, E) and the B T node worth 5
B. The brown node worth 4 (R, N, E) and the O node worth 4
C. The B T node worth 5 and the O node worth 4
D. All of the above
Huffman Tree: TO BE OR NOT TO BE

UC SANTA CRUZ

Huffman Tree: TO BE OR NOT TO BE

N = 1110
R = 1111
E = 110
B = 01
O = 10
T = 00

0010.01110.101111.11
101000.0010.01110

32 bits

Huffman Code Algorithm Overview

- Decoding
  - Read compressed file & binary tree
  - Use binary tree to decode file
    - Follow path from root to leaf
- FIRST EXAMPLE. ACE

TO BE OR NOT TO BE

How many bits would it take to store this message if every letter was represented with the same number of bits? You should first figure out how many bits it takes to represent 6 different values/letters.

A. 26
B. 32
C. 39
D. 48
E. 52

0010.01110.101111.11
101000.0010.01110

32 bits
Huffman Decoding 1

3 2 1 5 8 10 20

1111001

Huffman Decoding 2

3 2 1 5 8 10 20

1111001

Huffman Decoding 3

3 2 1 5 8 10 20

1111001

Huffman Decoding 4

3 2 1 5 8 10 20

1111001

Huffman Decoding 5

3 2 1 5 8 10 20

1111001

Huffman Decoding 6

3 2 1 5 8 10 20

1111001
**Huffman Decoding 7**

1111001

**DECODING: 2nd example “BEBE”**

N = 1110
R = 1111
E = 110
B = 01
O = 10
T = 00

BEBE = 0111001110

---

**Huffman Code Properties**

- **Prefix code**
  - No code is a prefix of another code
  - Example
    - Huffman("dog") → 01
    - Huffman("cat") → 011 // not legal prefix code
  - Can stop as soon as complete code found
  - No need for end-of-code marker
- **Nondeterministic**
  - Multiple Huffman coding possible for same input
  - If more than two trees with same minimal weight

---

**No code is prefix of another**

N = 1110
R = 1111
E = 110
B = 01
O = 10
T = 00

---

**DECODING: Your turn**

N = 1110
R = 1111
E = 110
B = 01
O = 10
T = 00

1111001100 = ?

**DECODING: Your turn**

N = 1110
R = 1111
E = 110
B = 01
O = 10
T = 00

111100111000 = ?

A. ROBBER
B. REBOOT
C. ROBOT
D. ROOT
E. ROBERT
Huffman Code Properties

- Greedy algorithm
  - Chooses best local solution at each step
  - Combines 2 trees with lowest frequency
- Still yields overall best solution
  - Optimal prefix code
  - Based on statistical frequency
- Better compression possible (depends on data)
  - But needs “look ahead”, not prefix.

Homework 7: Data representations. Logic. Binary Numbers. ASCII. Huffman Coding.

Is ASCII 7 bits or 8 bits? We use the 8 bit table

From 0 to 127 inclusive is standard ASCII and that’s 7 bits. From 128 to 255 inclusive is known as extended ASCII and there are conflicting standards in existence. See here for some discussion of extended ASCII encodings.

http://en.wikipedia.org/wiki/Extended_ASCII