



**2007-04-17**  
**Lecture #17**  
**(Material on Test 3)**

**Today's Outline**  
**Compression**  
**Huffman Coding**

## Data Compression

- **encode information using fewer bits than the unencoded version**
- **encode**
  - **Longer -> Shorter**
- **decode**
  - **Shorter -> Longer**

### Why?

- **useful to save hard disk space**
- **useful to save bandwidth on a network**

## Data Compression

- **compression scheme tradeoffs**
  - **lossless vs lossy**
    - ex: lossless : text, spreadsheet, executable**
    - ex: lossy : picture, video, audio**
  - **speed of encode / decode (ex: streaming video)**

## Data Compression

### run length encoding

- lossless
- large runs of consecutive identical data values are replaced by a simple code with the data value and length of the run

### Examples:

WWWWWWWWWWWWWWBWWWWWWWWWWWWWWWBBBWWWW  
WWWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWW  
WWWW

RLE = 12WB12W3B24WB14W

The run-length code represents the original 67 characters in only 16.

## Run Length Encoding

- Used in Fax Machines because most information on a fax sheet is whitespace

00000000000000000000000011111000000000111111

22T5F9T6F

Notice it's lossless ... we can convert 22T5F9T6F back and get the EXACT same data we started with

When will this work good? Only if there's long runs ...

- Not good on most photos (too many color changes, not long runs)
- Good on black and white photos

## **huffman coding**

- **lossless data compression algorithm**
- **entropy encoding algorithm**
  - **assigns codes to symbols as to match code lengths with probabilities of the symbols**
- **developed by David A. Huffman at MIT in 1952**
- **prefix-free code (a bit string representing some particular symbol is never the prefix of a bit string representing another symbol)**
- **optimal with known input probabilities**
- **applications : often a modified version is used as a back-end to some compression methods like 'PKZIP/ZIP', multimedia codecs, 'JPEG', and 'MP3'**

## Huffman Coding

- Recall ASCII codes are “fixed-length codes”
- Therefore Each letter takes up the same amount of space
- But what if we were smarter?

The letter E, for example, occurs quite often, whereas the letter Z almost never appears ... so wouldn't it be smarter to store the letter E with a “short code” and letter Z with a “longer code”?

## Steps

- 1.) Organize the entire character set into a row ... order according to frequency with highest to lowest**
- 2.) Find 2 nodes with the smallest combination of frequency weights and join them to form a third node ... the weight of this new node is the combination of the previous 2**
- 3.) Repeat step 2 until all of the nodes on every level are combined into 1 single tree**

**Example:**

**S = 5%**

**U = 5%**

**I = 4%**

**D = 4%**

**M = 3%**

**C = 3%**

**G = 2%**

**K = 2%**

**let's use this subset of the alphabet to build a huffman tree**



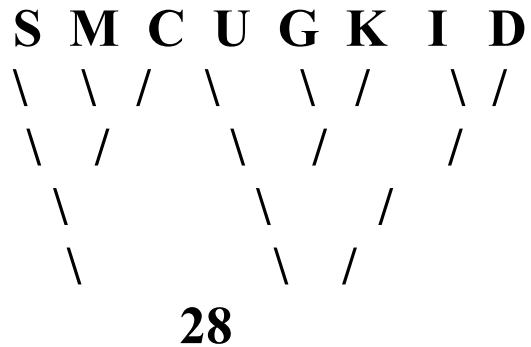
**Example:**

**S I D M C U G K**  
  \  
  /   \  
  /   /   \  
  /   /   /   \  
**05 08       06        09**

**I D S M C U G K**  
  \  
  /   \  
  /   /   \  
  /   /   /   \  
**08        11        09**

**S M C U G K I D**  
  \  
  /   \  
  /   /   \  
  /   /   /   \  
**11        17**

## Examples:



**Left = 1, Right = 0**

<b>Encode('S') = 11</b>	<b>Encode('M') = 101</b>
<b>Encode('C') = 100</b>	<b>Encode('U') = 011</b>
<b>Encode('G') = 0101</b>	<b>Encode('K') = 0100</b>
<b>Encode('I') = 001</b>	<b>Encode('D') = 000</b>

## **LZW Coding**

- lossless data compression algorithm**
- created by Abraham Lempel, Jacob Ziv, and Terry Welch in 1984**
- The algorithm is designed to be fast to implement but is not usually optimal because it performs only limited analysis of the data.**
- part of the GIF image format, optional in TIFF images**
- also a modified version exists in Adobe Acrobat (when it loads you see it say "LZW Algorithm")**

## LZW Coding

- TOBEORNOTTOBEORTOBEORNOT#
- Start by encoding all alphabet letters 1-27 including the #

Then add new symbols to the alphabet starting with 28

- start with 2 characters in input string ... pick 2 until get a repeat,  
then go to 3

28: TO	37: TOB
29: OB	38: OBE
30: BE	39: BEO
31: EO	40: ORT
32:OR	41: TOBE
33: RN	...
34:NO	...
35:OT	...
36: TT	

**Idea: Use these new symbols ...**

## **Other common compression algorithms to be aware of**

### **JAR**

- **java archive**
- **actually just a modified zip file**

## **bzip2**

- **free open source compression software, 1996**
- **compresses better than zip, but is slower**
- **compressor, not an archiver**
- **uses multiple compression algorithms on top of each other**
  - **run length encoding**
  - **burrows wheeler transformation (reorders buffer so it's more likely to contain large runs of symbols)**
  - **huffman coding**
  - **etc.**

### **mpeg-1 audio layer 3 (aka mp3)**

- **lossy compression for audio, 1996**
- **mpeg-1 standards don't include details about the encoder**
- **decoders though, are well-defined**
- **header and side information help the decoder decode the huffman encoded data correctly**
- **always a tradeoff between space used and sound quality**

## **mpeg-2**

- **lossy compress audio and visual data, 1998**
- **at the core of most Digital TV's and DVD players**
- **there are of course mpeg-3 and mpeg-4 which have improved**

## **mpeg-2 tricks ...**

- **A common (and old) trick to reduce the amount of data is to separate the picture into two fields: the "top field," which is the odd numbered rows, and the "bottom field," which is the even numbered rows. The two fields are displayed alternately. This is called interlaced video. Two successive fields are called a frame. If the video is not interlaced, then it is called progressive video and each picture is a frame. MPEG-2 supports both options.**
- **Therefore if the video is playing ... at any given “load” you only need either the top or bottom field ... not the whole thing**

## Mpeg-2 tricks

- Another trick to reduce the data rate is to **thin out** the two chrominance matrices. In effect, the remaining chrominance values represent the **nearby values** that are deleted. Thinning works because the **eye** is more responsive to brightness than to color. The 4:2:2 chrominance format indicates that half the chrominance values have been deleted. The 4:2:0 chrominance format indicates that three quarters of the chrominance values have been deleted. If no chrominance values have been deleted, the chrominance format is 4:4:4. MPEG-2 allows all three options.