Compression; Error detection & correction

- **compression: squeeze out redundancy**
  - to use less memory or use less network bandwidth
  - encode the same information in fewer bits
    - some bits carry no information
    - some bits can be computed or inferred from others
    - some bits don't matter to the recipient and can be dropped entirely

- **error detection & correction: add redundancy**
  - to detect and fix up loss or damage
  - add carefully defined, systematic redundancy
  - with enough of the right redundancy,

    can detect damaged bits

    can correct errors

Compressing English text

- **letters do not occur equally often**
- **encode frequent letters with fewer bits, less frequent things with more bits (trades complexity against space)**
  - e.g., Morse code, Huffman code, ...
- **run-length encoding**
  - encode runs of identical things with a count
  - e.g., World Wide Web Consortium => WWWC => W3C
- **words do not occur equally often**
- **encode whole words, not just letters**
  - e.g., abbreviations for frequent words

Lempel-Ziv coding; adaptive compression algorithms

- **build a dictionary of recently occurring data**
- **replace subsequent occurrences by (shorter) reference to the dictionary entry**
- **dictionary <u>adapts</u> as more input is seen**
  - compression adapts to properties of particular input
  - algorithm is independent of nature of input
- **dictionary is included in the compressed data**

- **Lempel-Ziv is the basis of PKZip, Winzip, gzip, GIF**
  - compresses Bible from 4.1 MB to 1.2 MB (typical for text)

- **Lempel-Ziv is a <u>lossless</u> compression scheme**
  - compression followed by decompression reproduces the input exactly

- **lossy compression: may do better if can discard some information**
  - commonly used for pictures, sounds, movies

JPEG (Joint Photographic Experts Group) picture compression

- **a lossy compression scheme, based on how our eyes work**
- **digitize picture into pixels**
- **discard some color information (use fewer distinct colors)**
  - eye is less sensitive to color variation than brightness
- **discard some fine detail**
  - decompressed image is not quite as sharp as original
- **discard some fine gradations of color and brightness**

- **use Huffman code, run-length encoding, etc., to compress resulting stream of numeric values**

- **compression is usually 10:1 to 20:1 for pictures**
- **used in web pages, digital cameras, ...**

MPEG (Moving Picture Experts Group) movie compression

- **MPEG-2: lossy compression scheme, based on human perceptions**
- **uses JPEG for individual frames (spatial redundancy)**
- **adds compression of temporal redundancy**
  - look at image in blocks
  - if a block hasn't changed, just transmit that fact, not the content
  - if a block has moved, transmit amount of motion
  - motion prediction (encode expected differences plus correction)
  - separate moving parts from static background
  - ...
- **used in DVD, high-definition TV, digital camcorders, video games**
- **rate is 3-15 Mbps depending on size, frame rate**
  - 15 Mbps ~ 2 MB/sec or 120 MB/min ~ 100x worse than MP3
  - 3 Mbps ~ 25 MB/min; cf DVD 25 MB/min ~ 3000 MB for 2 hours
  - regular TV is ~ 15 Mbps, HDTV ~ 60-80 Mbps
- **see www.bbc.co.ul/rd/pubs/papers/paper_14/paper_14.shtml**

MP3 (MPEG Audio Layer-3) sound compression

- **movies have sound as well as motion; this is the audio part**
- **3 levels, with increasing compression, increasing complexity**
- **based on "perceptual noise shaping":**

  use characteristics of the human ear to compress better:
  - human ear can't hear some sounds (e.g., very high frequencies)
  - human ear hears some sounds better than others
  - louder sounds mask softer sounds
- **break sound into different frequency bands**
- **encode each band separately**
- **encode 2 stereo channels as 1 plus difference**
- **gives about 10:1 compression over CD-quality audio**
  - 1 MB/minute instead of 10 MB/minute
  - can trade quality against compression

- see http://www.oreilly.com/catalog/mp3/chapter/ch02.html

Other audio compression algorithms

- **AAC, AC-2 (Apple iPod)**
- **WMA (Windows Media Audio)**
- **Ogg (open source)**
- ...
  - maybe 20:1 over WAV format

- **speech coding for cell phones, Internet telephony, etc.**
  - narrower frequency range (100 Hz - 4 KHz)
  - requires low delay
  - uses a model of human vocal tract

  - much higher compression than for general audio

Summary of compression

- **eliminate / reduce redundancy**
  - more frequent things encoded with fewer bits
  - use a dictionary of encoded things, and refer to it (Lempel-Ziv)
  - encode repetitions with a count
- **not everything can be compressed**
  - something will be bigger

- **lossless vs lossy compression**
  - lossy discards something that is not needed by recipient

- **tradeoffs**
  - encoding time and complexity vs decoding time and complexity
  - encoding is usually slower and more complicated (done once)
  - parameters in lossy compressions

    size, speed, quality

Error detection and correction

- **systematic use of redundancy to defend against errors**

- **some common numbers have no redundancy**
  - and thus can't detect when an error might have occurred
  - e.g., SSN -- any 9-digit number is potentially valid

- **if some extra data is added or if some possible values are excluded, this can be used to detect and even correct errors**

- **common examples include**
  - ATM & credit card numbers
  - ISBN for books
  - bar codes for products

ATM card checksum

- **credit card / ATM card checksum:**

  starting at rightmost digit:

  multiply digit alternately by 1 or 2

  if result is > 9 subtract 9

  add the resulting digits

  sum should be divisible by 10

  e.g., 12345678 is invalid

  8 + (14-9) + 6 + (10-9) + 4 + 6 + 2 + 2 = 34

  but 42345678 is valid

  8 + (14-9) + 6 + (10-9) + 4 + 6 + 2 + 8 = 40

- **defends against transpositions and many single digit errors**
  - these are the most common errors

ISBN checksum

- **checksum for 10-digit ISBN:**
  starting at leftmost digit:
    multiply corresponding digit by 10, 9, 8, ... down to 1 inclusive
      (a final X has value 10)
    add the resulting numbers
  result must be divisible by 11

  e.g., 0-201-61586-X is valid
    10*0 + 9*2 + 8*0 + 7*1 + 6*6 + 5*1 + 4*5 + 3*8 + 6*2 + 1*10 = 132 = 12*11

- **defends against transpositions and single digit errors**
  - and catches 90% of others

Parity & other binary codes

- **parity bit: use one extra bit so total number of 1-bits is even**

  `0110100 => 01101001`

  `0110101 => 01101010`

  - detects any single-bit error

- **more elaborate codes can detect and even correct errors**

- **basic idea is to add extra bits systematically so that legal values are uniformly spread out, so any small error converts a legal value into an illegal one**
  - some schemes correct random isolated errors
  - some schemes correct bursts of errors (used in CD-ROMs)

- **no error correcting code can detect/correct all errors**
  - a big enough error can convert one legal pattern into another one