CS216: Program and Data Representation
University of Virginia Computer Science
Spring 2006 David Evans

Lecture 15: Compression

http://www.cs.virginia.edu/cs216

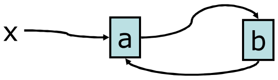
Menu

- Garbage Collection Puzzle
- Encoding: Huffman, LZW, GIF
- Representing Numbers

Fighting Finalizers

- finalize() – method in java.lang.Object
  - Class can override
  - It is called when GC determines the object is garbage (before collecting it)

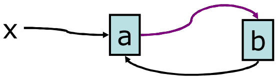
```
x = NULL;
GC  a.finalize ();
    b.finalize ();
```



Problem due to Paul Tyma, Google

Finality?

```
class A {         class B {
   B p;              A r;
}                    void finalize() {
                       r.p = this;
                  }
a = new A();
a.p = new B();
a.p.r = a;
x = a;
a.p = NULL;
GC – b.finalize ()
a.p.toString ()
```



Problem due to Paul Tyma, Google

Encoding

- Huffman Encoding
  - We proved there is no better encoding that is:
    - Prefix encoding (can divide coded message into symbols without looking ahead)
    - One-to-one mapping (Symbol→Bits)
    - Fixed mapping
- Can we do better without these constraints?

Lempel-Ziv-Wench (LZW)

- Terry Wench refined the L-Z scheme
- Fixed-length (typically 12-bits) codewords
- Dictionary maps each codeword to text
- Greedy scheme for building dictionary

LZW Encoding Algorithm

```
def LZWEncode (s):
  w = “”;
  res = “”;
  dictionary.initialize() # code for each alphabet symbol
  foreach k in s:
    if dictionary.contains (w + k):
       w = w + k;
    else
       # need to do something if dictionary is full
       dictionary.append (w + k)
       res = res + dictionary.find (w) ←w is already
       w = k;                            in dictionary:
                                         no need to send
                                         dictionary to decode!
```

Compression Bake-off

Declaration of Independence

Original	8586
Huffman (PS4)	5123 (60%)
Compress (LZW)	4441 (52%)
Gzip (not LZW)	3752 (44%)

Random Characters

Original	10000
Huffman (PS4)	9517
Compress (LZW)	10000 (“file unchanged”)
Gzip (not LZW)	8800

This is quite surprising!

GIF

- Graphics Interchange Format developed by Compuserve (1987)
- Algorithm:
  - Divide image into 8x8 blocks
  - Find optimal Huffman encoding for those blocks
  - Encode result using LZW

How is GIF different from JPEG?

Lossy/Lossless Compression

- Lossless Compression:
  - uncompress (compress (*S*)) = *S*
- Lossy Compression:
  - uncompress (compress (*S*)) similar to *S*
- For images, sound, video: lossy compression is usually okay
- For computer programs, declarations of independence, email: only lossless compression will do!

What’s wrong with GIF?

- Divide image into 8x8 blocks
- Find optimal Huffman encoding for those blocks
- Encode result using LZW

- 1978: LZ patented by Sperry
- 1984: (June) Welch’s article on LZW
- 1984: (July) Unix compress implemented using LZW
- 1987: Compuserve develops GIF (Graphics Interchange Format) image format, used LZW but didn’t know it was patented
- GIF becomes popular
- 1994: Unisys/Compuserve decide that developers who implement LZW (including in GIF) will have to pay a licensing fee
- 2003: LZW patent expired

PNG (“PNG’s Not GIF”)

Representing Numbers

Binary Representation

$$b_{n-1}b_{n-2}b_{n-3}...b_2b_1b_0$$

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 carry 1

Value = $\underset{i = 0..n-1}{\Sigma} b_i * 2^i$

What should n be?

What is n?

- Java:
  - byte, char = 8 bits
  - short = 16 bits
  - **int = 32 bits**
  - long = 64 bits
- C: implementation-defined
  - **int**: can hold between 0 and UINT_MAX
    - UINX_MAX must be at least 65535

      n >= 16, typical current machines n = 32
- Python? n is not fixed (numbers work)

Charge

- Is Java a “high-level language”?
  - Only if you never use numbers bigger than 231. If you have to worry about how numbers are represented, you are doing low-level programming
- PS4 Due Wednesday