6 File Compression

Content

- Introduction to Compression
- Methods in Data Compression
  - Run-Length Coding
  - Huffman Coding

Data Compression

- Fixed-Length fields are good candidates
- Decrease the number of bits by finding a more compact notation
- *Cons.*
  - unreadable by human
  - cost in encoding time
  - decoding modules ⇒ increase the complexity of s/w
    ⇒ used for particular application

Data Compression

- Reasons for data compression
  - less storage
  - transmitting faster, decreasing access time
  - processing faster sequentially

Suppressing repeating sequences

- Run-length encoding algorithm
  - read through pixels, copying pixel values to file in sequence, except the same pixel value occurs more than once in succession
  - when the same value occurs more than once in succession, substitute the following three bytes
    - ✓special run-length code indicator(e.g. 0xFF)
    - ✓pixel value repeated
    - ✓the number of times that value is repeated
    - Example:
    - 22 23 *24 24 24 24 24 24 24* 25 26 26 26 26 26 26 25 24

      RL-coded stream: 22 23 *ff 24 07* 25 ff 26 06 25 24

Suppressing Repeating Sequences

- Run-length encoding (cont'd)
  - example of redundancy reduction
  - cons.
    - not guarantee any particular amount of space savings
    - under some circumstances, *compressed image* is larger than original image
      - Why? Can you prevent this?

Assigning Variable-Length Codes

- *Morse code*: oldest & most common scheme of variable-length code
- Some values occur more frequently than others
  - that value should take the least amount of space
- *Huffman coding*
  - base on probability of occurrence
    - determine probabilities of each value occurring
    - build binary tree with search path for each value
    - more frequently occurring values are given shorter search paths in tree

Assigning Variable-Length Codes

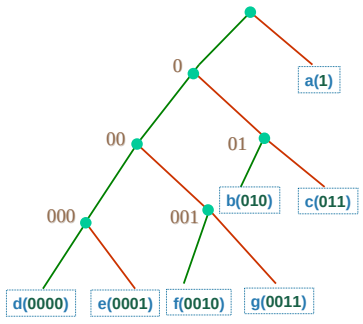
- *Huffman coding*

Letter:	*a*	*b*	*c*	*d*	*e*	*f*	*g*
Pr:	*0.4*	*0.1*	*0.1*	*0.1*	*0.1*	*0.1*	*0.1*
Code:	*1*	*010*	*011*	*0000*	*0001*	*0010*	*0011*

Example: the string "abde"

➔1010000000001

Huffman Tree



Lempel-Ziv Codes

- There are several variations of Lempel-Ziv Codes.
- We will look at LZ78
- Commands **zip** and **unzip** and Unix **compress** and **uncompress** us Lempel-Ziv codes

Example

- Let us look at an example for an alphabet having only two letters:

aaababbbaaabaaaaaaabaabb

- Rule
  - Separate this stream of characters into pieces of text so that each piece is the shortest string of characters that we have not seen yet.

a|aa|b|ab|bb|aaa|ba|aaaa|aab|aabb

1. We see "a"
2. "a" has been seen, we now see "aa"
3. We see "b"
4. "a" has been seen, we now see "ab"
5. "b" has been seen, we now see "bb"
6. "aa" has been seen, we now see "aaa"
7. "b" has been seen, we now see "ba"
8. "aaa" has been seen, we now see "aaaa"
9. "aa" has been seen, we now see "aab"
10. "aab" has been seen, we now see "aabb"

Index

- We have index values from 1 to *n*
- For the previous example

1 2 3 4 5 6 7 8 9 10

a|aa|b|ab|bb|aaa|ba|aaaa|aab|aabb

- Encoding

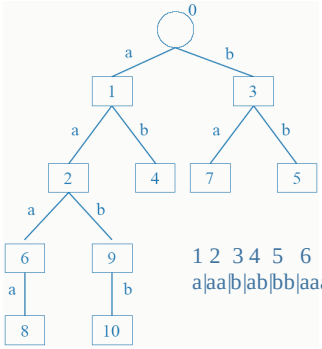
1 2 3 4 5 6 7 8 9 10

0a|1a|0b|1b|3b|2a|3a|6a|2b|9b

Lempel-Ziv Codes

- Since each piece is the concatenation of a piece already seen with a new character, the message can be encoded by a previous index plus a new character.
- A tree can be built when encoding

Encoding Tree



1 2 3 4 5 6 7 8 9 10

a|aa|b|ab|bb|aaa|ba|aaaa|aab|aabb

Exercise # 1

- encode the file containing the following characters, drawing the corresponding digital tree

“aaabbcbcdddeab”

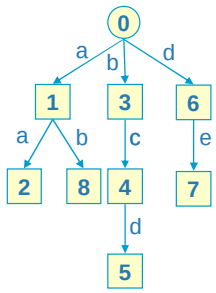
Solution

1 2 3 4 5 6 7 8

a|aa|b|bc|bcd|d|de|ab

0a|1a|0b|2c|4d|0d|6e|1b

Encoding Tree



1 2 3 4 5 6 7 8

a|aa|b|bc|bcd|d|de|ab

0a|1a|0b|2c|4d|0d|6e|1b

Exercise # 2

- Encode the file containing the following characters, drawing the corresponding digital tree

"I AM SAM. SAM I AM"

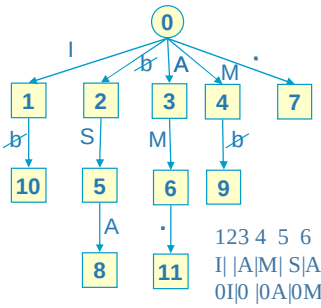
Solution

1 2 3 4 5 6 7 8 9 10 11

I| |A|M| S|AM|.| SA|M |I |AM.

0I|0 |0A|0M|2S|3M|0.|5A|4 |1 |6.

Encoding Tree



123 4 5 6 78 9 10 11

I| |A|M| S|AM|.| SA|M |I |AM.

0I|0 |0A|0M|2S|3M|0.|5A|4 |1 |6.

Lossy Compression Techniques

- Some information can be sacrificed
- Less common in data files
- Shrinking raster image
  - 400-by-400 pixels to 100-by-100 pixels
  - 1 pixel for every 16 pixels
- Speech compression
  - *voice coding (the lost information is of no little or no value)*