CS 240: Data Structures and Data Management
Module 10 Study Guide

Taylor J. Smith — Spring 2017

Key Concepts

- **Compression** takes a source text S and encodes it into a coded text C.
- The goal of compression is to minimize $|C|$, and this can be measured by the compression ratio.
- **Lossless** compression occurs when $\text{DECODE}(\text{ENCODE}(S)) = S$, and **lossy** compression occurs otherwise.
- **Huffman coding** uses character frequencies to build a variable-length code. Characters that occur more often get a shorter encoding.
- **Run-length encoding** encodes runs of identical characters by storing the run length. However, compression is only possible when runs in the string are of length ≥ 6 characters.
- **Lempel–Ziv–Welch coding** is similar to Huffman coding, but it considers frequent substrings instead of frequent characters.
  - The encoder reads a substring x, then adds xc to an adaptive dictionary (where c is the character following x)
  - The decoder reads two substrings x and y, then adds xc to an adaptive dictionary (where $c = y[0]$)
- Text **transforms** allow us to make a source text more compressible by rearranging its characters.
- The **move-to-front transform** rearranges the source alphabet, Σ_S, and converts source characters to their index in the alphabet. Recently-seen characters have a smaller index value.
- The **Burrows–Wheeler transform** rearranges the characters in the source text.
  - The encoder generates all cyclic shifts of S, sorts these shifts lexicographically, then returns the trailing characters of each shift as C
  - The decoder rebuilds the cyclic shifts and constructs S one character at a time directly from C
- Many real-world compression algorithms use a combination of the above techniques.
  - A typical encoding process may be BWT $\rightarrow$ MTF $\rightarrow$ RLE $\rightarrow$ encoder
  - The decoding process is simply the inverse of these steps

Suggested Readings

- **CLRS:** 16.3 (Huffman codes)
- **Goodrich/Tamassia:** 9.3 (Text Compression)

Practice Questions

CLRS

16.3-2. Prove that a binary tree that is not full cannot correspond to an optimal prefix-free code.

16.3-3. What is an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers?

a:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21

Can you generalize your answer to find the optimal code when the frequencies are the first n Fibonacci numbers?

16.3-5. Prove that if we order the characters in an alphabet so that their frequencies are monotonically decreasing, then there exists an optimal code whose codeword lengths are monotonically increasing.

16.3-7. Generalize Huffman's algorithm to ternary codewords (i.e., codewords using the symbols 0, 1, and 2), and prove that it yields optimal ternary codes.

16.3-9. Show that no compression scheme can expect to compress a file of randomly chosen 8-bit characters by even a single bit.

(*Hint:* Compare the number of possible files with the number of possible encoded files.)

Goodrich/Tamassia

R-9.12. Draw the frequency table and Huffman tree for the following string:

"dogs do not spot hot pots or cats".

C-9.20. Suppose A, B, and C are three integer arrays representing the ASCII or Unicode values of three character strings, each of size n. Given an arbitrary integer x, design an $O(n^2 \log(n))$ time algorithm to determine if there exist numbers $a \in A$, $b \in B$, and $c \in C$ such that $x = a + b + c$.

C-9.21. Give an $O(n^2)$ time algorithm for the previous problem.

Additional Practice Questions

1. (a) Encode the string TIN␣TIPTIP using Lempel–Ziv encoding. Assume we are using the standard ASCII character set (0–127) for our alphabet Σ.

   (b) Decode the sequence of numbers [66 65 68 65 128 132 129] which has been obtained from Lempel–Ziv encoding. Assume we are using the standard ASCII character set (0–127) for our alphabet Σ.

2. Show the result of encoding the string ONDON$ using:

   (a) The Burrows–Wheeler transform, where characters are ordered $ < D < N < O.

   (b) Move-to-front encoding, where the initial dictionary is

0	1	2	3
$	D	N	O

3. If the Burrows–Wheeler transform outputs the string E ␣ ␣ N R O D Y A O $ U E, then what was the original string?

   (*Note:* The original string ends with $ and you may assume that the space character ␣ comes after the character Z.)