COMP 360 Study guide

Francis Piché

January 14, 2019

Contents

Part I
Preliminaries

1 Disclaimer

These notes are curated from Professor Paul Kry COMP557 lectures at McGill University. They are for study purposes only. They are not to be used for monetary gain.

2 About This Guide

I make my notes freely available for other students as a way to stay accountable for taking good notes. If you spot anything incorrect or unclear, don't hesitate to contact me via Facebook or e-mail at `http://francispiche.ca/contact/`

Part II
Analyzing Key Algorithms

3 Data Compression

As humans utilize more and more data, techniques are required for handling and storing this data efficiently. Some examples of data transformations:

- Compression for storage (minimize file sizes on JPEG, mp3 etc)
- Compression & redundancy adding for secure transmission
- Sorting data
- Watermarking
- Discretizing continuous measurements
- Representing information as bits
- Natural language translation

3.1 Information Theory

First we need to think about what information is really needed? What information do we need to record to maintain the comprehensiveness of the original structure?

Lossy compression such as mp3 and jpeg files, results in *approximate* communication, since the compression is done by approximating some points. (Data is lost). In lossy compression,

the more data is compressed, the more it becomes incomprehensible.

Lossless compression on the other hand, such as gif, gzip, or tar files. As the name implies, the original data can be recovered without any loss.

A lot of information is redundant. For example, the phases: *The colour of the chair is red. THe colour of the hat is red. The colour of the table is red.* Could all be simplified to *The chair, hat and table are red.*

The optimal strategy depends on the type of data we are dealing with. For example, transmitting numbers between 0 and 31 we need 5 bits, but for trasmitting numbers between 9 and 31 we only need 4, since we can transmit x-9, and add it back on the other side. Suppose further that we know the first number is at most 31 and that all numbers differ at most 3? Transmit the first number x_1 needing 5 bits, and then each following number only 3 bits, since we could transmit $x_i - x_{i-1} + 3$, and then only send from 0-6. If the number is 0 then the previous was 3 greater than the current, and if the number is 6 then we know that the previous was 3 less than the current.

The optimal strategy also depends on the encoding technique. For example, if we are transmitting a 50x50 2D array of bits indicating the position of 10 mines, we could send the whole array (2500) bits, or we could just send the coordinates of each mine (12 bits per mine, so 120 bits total).

3.2 Variable Length Encoding

Suppose we know the frequency of the symbols in our message, but nothing about the structure of the message.

Say our alphabet is $\{A, B, C, D\}$. With 15 A's, 7B's, 6C's, and 5 E's for 39 total. We only have 5 symbols so we could just use 3 bits per symbol by letting A=000, B=001, and so on.

We can do better by building a **prefix code**. A prefix code is one such that no two symbols x and y can be encoded such that the encoding of x is a prefix of the encoding of y. So we can't have $x = 1000$ and $y = 100011$

3.2.1 Shannon-Fano Code

In this technique, we order the symbols by frequency, decreasing. Then, we partition the set of symbols so that the sum of frequencies on each side of the partition is as close to equal as possible. Then, we use 0's to encode everything on one side of the partition, and 1's for everything on the other side. We repeat until we can progress no further.

So for our previous example, A-15, B-7,C-6, D-6, E-5, if we group A+B=22 then it's as

close to equal to C+D+E=17 as we can get. Then we split A from B,as that's all we can do there. Then we split the CDE group into C, DE. And then D, E.

From this algorithm, we can see that if a node X is more frequent than Y, X cannot be deeper than Y. Similarly, there cannot be a node at the deepest level without a sibling. (By the structure of the tree, the parent is redundant). Swapping the leaves at the same level does not change the number of bits per symbol.

3.2.2 Huffman Coding

Improving on the Shannon-Fano Code, we can build the tree by using what we observed in the analysis above. Since the order on the same level doesn't matter, and the leaves will always be in pairs, we can build it up by taking the least frequent two symbols, and making their parent the sum of the two frequencies. We then add nodes to the tree in this manner until we can't anymore. (See my COMP251 study guide for more on Huffman Coding).

3.2.3 Entropy

The entropy of a probability distribution p on the letters of an alphabet is

$$\sum_{x \in S} -p(x) log_2(x)$$

The Huffman code is actually always within 1 bit per symbol of the entropy, and we cannot do better than entropy.

4 Google Page Rank

5 Simplex Algorithm for Linear Programming