Mahdiur Rahman

Functional Programming with Alexey Nikolaev

Project Report: Huffman Code Compression Algorithm

This project is initially composed of two files: compress.ml and Makefile. The actual algorithm is housed in compress.ml, which creates a file called ‘decompress.ml’.

The file ‘compress.ml’ is essentially composed of six parts:

1. READ FILE + MAKE DATA

   The objective here was simply to read a given file line by line, record each as a string, add a ‘\n’ at the end, and then append to a list. This list is essentially our data which is now in a form ready to compute.

2. MAKE TABLE

   The purpose of make-table was to create a list of each unique character found in the data, and then organize them from least occuring to most occuring. This is necessary to create the Huffman tree.

3. MAKE TREE

   The purpose of this section was to take the ‘table’ from the previous section, and create the Huffman tree, which propagates by making a node of the two smallest items in a list and continually doing so until everything conglomerates into one node. Hence, we have a useable tree.

4. MAKE CHART

   The chart is not an essential part of any Huffman algorithm, but for easiness on my part, I created a chart. The chart is essentially a global variable, a list of tuples of the form (char

* string). Instead of reading each character in the input file, navigating the Huffman tree, and then outputting whatever combination of 1's and 0's we get, I opted to save what each character corresponding binary code is, and simply navigating the list to find it when necessary.

5. OUTPUT COMPRESSED FILE

   I read each character of each string in our data, and created a larger string by inputting the corresponding code per character. This final large string is then output to the out-file. The out-file is essentially your 'compressed' file.

6. CONVERT TREE TO STRING

   In this section, I essentially take in the tree from section 3, and convert it into a string.

7. CREATE DECOMPRESS FILE

   Here, I essentially make a mega-string and then output it onto an ml or ocaml file. This ocaml file can then act as a decompression algorithm. Three strings get added together: 1. A string representing a node data type 2. The tree string created in section 6 3. A huge string which is essentially a string version of the decompression algorithm.

   Added together, the resultant string is a working decompression system with a unique Huffman tree.

If you were to open decompress.ml you would see not comments which differentiate sections. Nonetheless there are three parts:

1. The first part involves the Node data type, and the unique Huffman tree from compress.ml.

2. The second part reads the compressed file, which is essentially a single string of 1's and 0's, and saves it as a string.
3. The third part goes through each item of the string from part 2 and then navigates the Huffman tree. It goes left on 0 and right on 1, and whenever it reaches a Leaf node it appends the leaf's character to a central string. This central string is then output to an out-file. The out-file should be of the same contents as the very original in-file.

Problems:

The project is far from perfect and is essentially non operational. The reason is because I did not have the necessary tools/knowledge to express bits and save them in a file which would house only bits. However, this is minor compared to the implementation of the actual algorithm. To make this algorithm functional would require my mastering some of the bit packages and modules in oCaml.

Second problem was that I was unable to compile an ml file which used another ml file in its first line. In fact, I originally planned for decompress.ml to use a file called tree_.ml. Tree_.ml was supposed to be an ml file housing a single variable which was essentially the Huffman tree made in compress.ml. Since I could not get them to work, the work around was simply to make everything a string in compress.ml and then output it to as a file called decompress.ml.

Overall however this project was very rewarding and I appreciate the liberty it gave me. Most projects academic projects have very specific requirements which takes out a lot of the challenge. This project required some solid research and head scratching.