Programming Paradigms: Miniproject Two

Lukas Rønsholt
20166653
lransh16@student.aau.dk

Huffman coding done using haskell, the program will encode an given string using the huffman coding principle. first the unique characters is found and the occurrences of them. A binary tree is then build on these characters, with the characters of least occurrence being the lowest leafs. The binary tree is then traversed using depth first to calculate all of the characters bit values. At last the input string is then encoded by mapping the bit values on the string.

Decoding works by traversing the encoded string one bit at a time, looking up the bit in the calculated bit values. If a match is found the corresponding character is then added to a string, and we take the next bit in the bit string and lookup. If no match is found, the next bit from the bit string will be added to the current bit until a match is found.

To see the encoded string and decoded again it is possible to just run main in *ghci*, to do the encoding the following example can be followed:

```
text = "mississippi river"                  -- The text we will compress
chars = getDictFromString text              -- Get the uniqe characters and their occurences
tree = createBinaryTreeFromDict chars       -- Create a binary tree from the characters
prefix = getPrefixCodes tree                 -- Get the prefix codes for each character

compressed = compressString text prefix     -- Compress the text using the prefix codes

decompressed = decompressString compressed prefix   -- Decompress the text again
```

As an extension to the assignment a way to save a encoded string to a file was also made, along side with its counterpart to decode an string from a file. A function to encode text from a file and save it in a new file is also made. These can be seen below:

```
compressAndSaveToFile text      -- Compress the text and save to a file called "compressed"
decompressFromFile "compressed" -- Decompress the same file

compressFile "file path"        -- Compress text from a file
```

As an further extension a function was also made to make a graphical representing of the created binary tree. This function creates an file that can be read by the program *Graphviz*[1]. And a png can be created of the graph by running

```
dot -Tpng graph.gv -o Graph.png
```

References

[1] Graphviz. *Graphviz - Graph Visualization Software*. URL: `http://www.graphviz.org/`. (accessed: 01.09.2016).