Umeå University
Institution för Datavetenskap

Datavetenskapens byggstenar 7.5 p
DV160HT15

OU3 Huffman Coding

Submitted	2016-02-18
Author:	Simon Andersson (`dv15san@cs.umu.se`)
	Lorenz Gerber (`dv15lgr@cs.umu.se`)
Instructor:	Lena Kallin Westin / Erik Moström / Lina Ögren

Contents

1 Introduction

The aim of this laboration was to plan and implement a command line program written in C that accomplishes encoding and decoding of text files according to the *Huffman* algorithm.

The *Huffman* algoritm is used for data compression, in our case for text files. The basic idea is that instead of using 8 bytes for every character, fine unique variable length binary representations for every character where the most common used characters get the shortest binary sequences. In practice, this includes several steps: First a frequency count table of all characters has to be compiled. Then a binary tree is constructed where characters and their count frequency as leafs. Characters with high frequency count will be placed the closest to the tree root. A more detailed description of this process will be given in the method description.

2 Material and Methods

2.1 Datatypes

From the provided datatypes we used *prioqueue* (which is built on *list_2cell*), *tree_3cell*, and *bitset*.

2.2 Work Organization

On an initial kick-off meeting, we discussed the problem and possible solution strategies. Then we created repositories for the *code* and for the *report* on *github* and setup a team in a workgroup messaging app (*slack*). All further work and communication was done remote using the afore-mentioned tools.

3 Results

And our results looked like this...

4 Discussion

bla bla bla...

5 Contributions

Both authors were involved in every function with either writing or debugging/checking it. The initials in table 1 stand for the person who initially wrote the respective function.

References

Table 1 work contributionsn

planning and defining the strategy	SAN, LGR
setting up and managing git repos	LGR
initial code structure	SAN, LGR
main program and argument handling	LGR
char frequency count	SAN
compare tree function	SAN
build Huffman tree	SAN
tree traversal function	SAN
huffman code from tree traversal	LGR
encode function	LGR
decode function	SAN, LGR
file read/write	LGR
commenting and styling code	SAN, LGR
memory leak check	SAN
setting up LaTex document	LGR
writing report	SAN LGR