Huffman Coding

DCTI – IT Companies Seminary

Student: Samir-Constantin Prejbeanu

Coordinator: Cristian Mihaescu Ph.D

April,2014

Short Introduction

About Huffman Coding:

- Entropy encoding algorithm
- Used for lossless data compression
- Is using a variable-length code for encoding a source symbol (such as a character in a file)
- Is using a specific method for choosing the representation for each symbol, resulting in a prefix code ("prefix-free codes“)

Example:

Given the input text: **To be or not to be.This is not a question.**

Step 1: The algorithm starts by reading the text and counting how many times each characters is found. As a result we get a Character – Frequency table:

" "	9
" . "	2
" T "	2
" a “	1
" b "	2
" e "	3
" h "	1
" i "	3

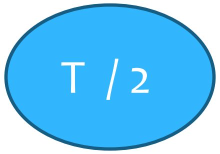
" n "	3
" o "	6
" q "	1
" r "	1
" s "	3
" t "	4
" u "	1

Example:

Step 2 : - **Creating Leaf Nodes**

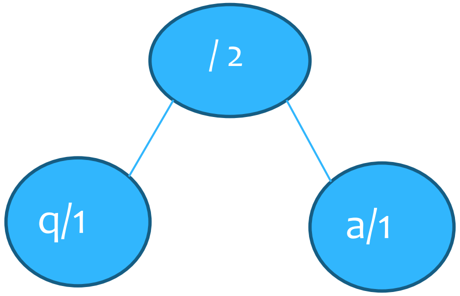
- **Creating Internal Nodes** → **The Tree**

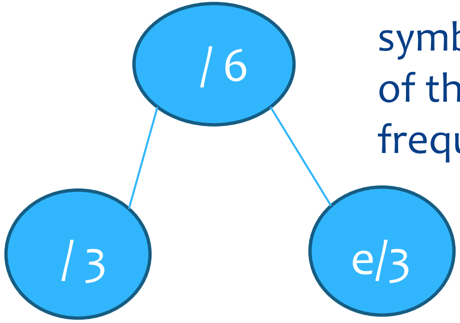
Leaf-Node example: -



* Leaf-Nodes contain Symbols and their frequency.

Internal-Node examples:





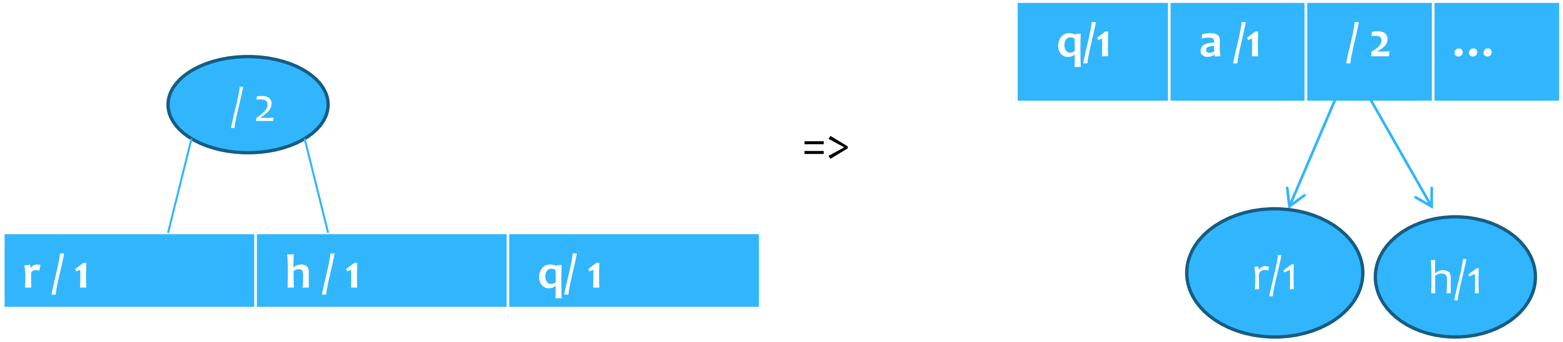
* Internal Nodes contain no symbol. They only contain the sum of their left and right children frequencies.

Example:

Every LeafNode is inserted into a priority queue ordered by their frequency.

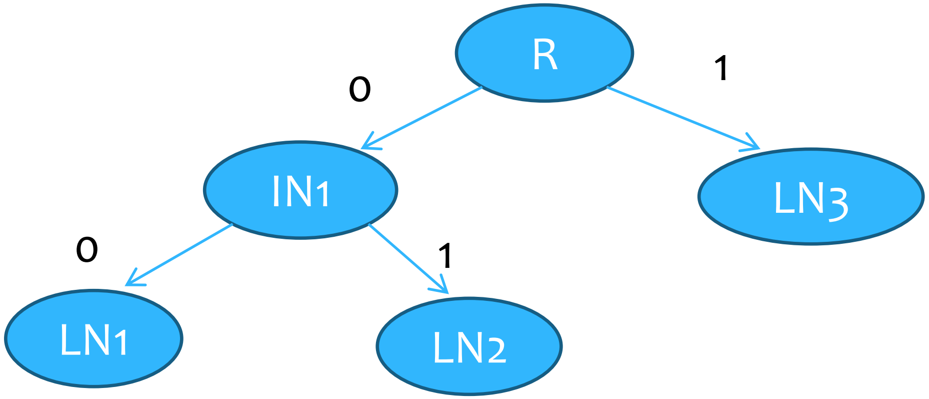
r / 1	h/ 1	q / 1	a / 1	. / 2	…

Internal Nodes are created by getting Nodes from the priority queue. We get it's left and it's right child and we form it. After that we insert it in the queue and re order the queue.



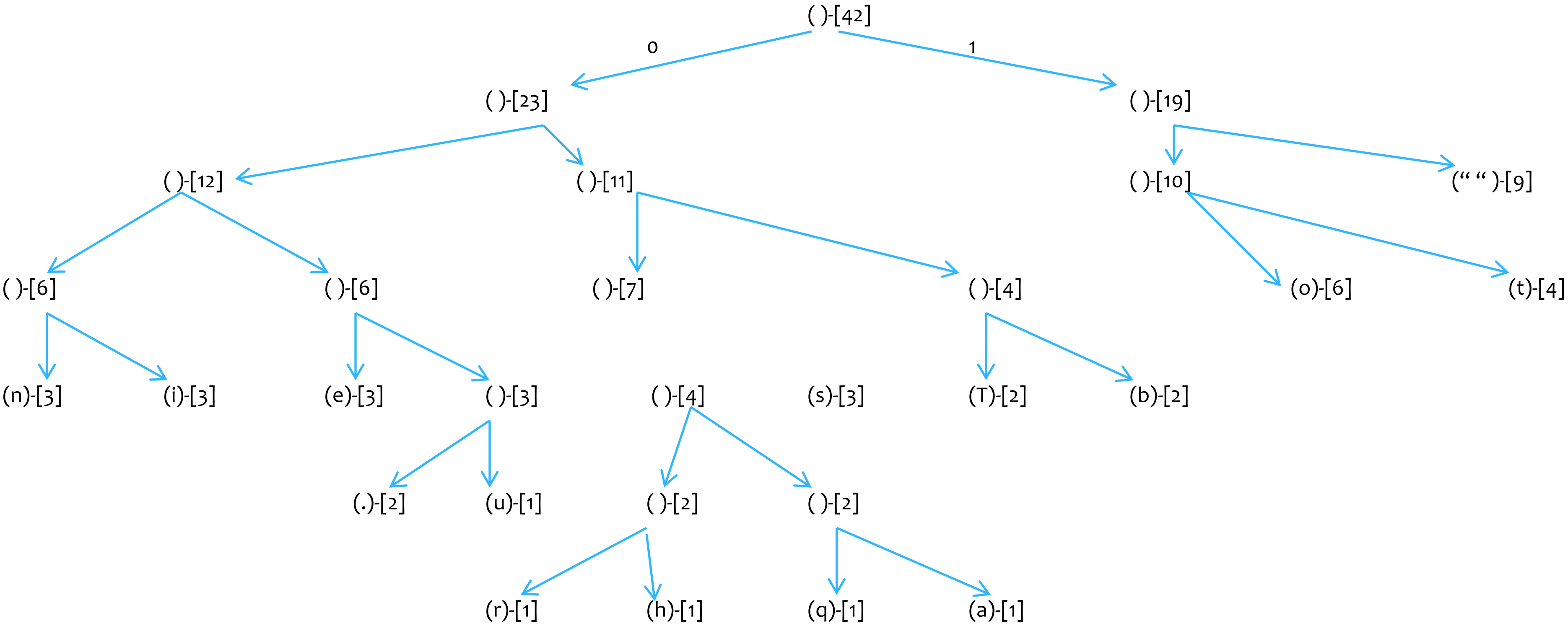
Example:

Step 3: After the tree is created it’s time to fill it’s branches in order to encode the symbols. Every left branch is coded with ‘0’ while every right branch is coded with ‘1’.



As a result for the Symbol in LeafNode 1 we have the following code : ‘00’ , for LN2: ‘01’ and etc.

The Huffman Tree



As a result we get the **Variable-Length Prefix-Free Codes** for every symbol found in the tree. The code is computed while walking the tree until a symbol is reached.

Prefix-Free Code - the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol.

" "	11
" . "	00110
" T "	0110
" a "	010011
" b "	0111
" e "	0010
" h "	010001
" i "	0001

" n "	0000
" o "	100
" q "	010010
" r "	010000
" s "	0101
" t "	101
" u "	00111

Result : Compression Ratio

Input	42 Symbols	336 Bits	2.23
Output	151 Symbols	151 Bits	

Other Results:

Input	592 Symbols	4736 bits	1.74
Output	2717 Symbols	2717 bits	

Input	1159 Symbols	9272 bits	1.65
Output	5627 Symbols	5627 bits	

Input	2092 Symbols	16736 bits	1.51
Output	11095 Symbs	11095 bits	

Input	4095 Symbs	32760 bits	1.51
Output	21707 bits	21707 bits	

Resources

Links:

en.wikipedia.org
stackoverflow.com
www.youtube.com
http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/
http://mathworld.wolfram.com/

Books:

1. Burdescu,Dan Dumitru,Mihaescu,Marian Cristian,Algorithms and Data Structures,Academica Griefswald,2012.
2.Kernighan,W. Brian,Ritchie,Dennis M.,The C Programming Language 2nd edition.
3.Drozdek,Adam, Data Structures Algorithms in C++ Second edition.
4.Cormen,Thomas H.,Leiserson Charles E.,Rivest,Ronald R,Stein,Clifford,Introduction to Algorithms 3rd edition

Thank you for your time!

The algorithm can be found at:
https://github.com/mihaescu/ADS/tree/master/Data-Compression