Algorithms and Data Structures 2010 - 2011

Lesson 8: *Huffman codes*

Luciano Bononi

International Bologna Master in Bioinformatics

University of Bologna

27/05/2011, Bologna

Coding

- Let's assume to have a **message** composed only of lower-case letters
- Our goal is to code the message in a very compact form, to **save memorization space**
- Without using compressing techniques, **how many bits are used to codify each letter**?
- If we assume to have an **alphabet** of lower-case letters (that is without numbers, symbols and upper-case letters). **How many bits are necessary to codify each letter?**

Coding

- Alphabet = 26 letters

Number of bits	Alphabet size
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
5	$2^5 = 32$
6	$2^6 = 64$

Coding

- Alphabet = 26 letters

Number of bits	Alphabet size
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
5	$2^5 = 32$
6	$2^6 = 64$

Fixed length codes

- **Simple solution**: a fixed length code is assigned to each letter in the alphabet. This solution is often very inefficient!

Letter	Code
a	00000
b	00001
c	00010
d	00011
e	00100
f	00101
...	...

Frequency of letters

- **Some letters are more frequent than others, that is the letters have a different frequency of use**
- Example:
  - the frequency of vocal letters is usually much higher than consonants
  - some consonants are more used than others (e.g. "c" vs. "y")
  - in a given message, a specific letter could not appear at all

Variable length codes

- **Idea**: a **variable length code** is assigned to each letter
- The length of each code depends on the frequency of the letter

Letter	Code
a	0
b	1
c	00
d	01
e	10
f	11
…	…

Variable length codes

- **That's all ok?** NO! The proposed code is **ambiguous**!
- Example, the received message:

  "**001000**"

- Can be decoded in many different ways:
  - "**0 0 1 0 0 0**" -> "aabaaa"
  - "**00 1 00 0**" -> "cbca"
  - "**00 10 0 0**" -> "ceaa"
  - "**00 10 00**" -> "cec"
  - … **The way you decode is not unique!!!**

Huffman codes

- The Huffman codes are **non-ambiguous variable-length codes**
- The algorithm used to define the code follows a greedy approach and is based on a binary tree

Definition of the code:

- **Step 1**: the frequency of each letter in the message is calculated
- **Step 2**: the set of letters is placed in **non-increasing frequency order (e.g. f(a) >= f(c) >= f(b)…)**
- **Step 3**: the two letters with **min f value (right side)** are selected
- **Step 4**: a new symbol is created, whose children are the letters selected before. Its frequency is the sum of child's frequencies
- **Step 5**: go back to step 2, until only one symbol remains

Huffman codes

- At the end of this algorithm, we have a **binary tree** that has as leaves the letters that are in our alphabet
- To obtain the code that is associated to each letter, we have to visit the tree (**traversal procedure**). Each left-edge (left-child) represents a "**0**" and each right-edge (right-child) is a "**1**"

Example of Huffman code

- Let's suppose that the message is: "**abracadabra**"
- Frequency table:

Symbol	Frequency
d	1
b	2
r	2
c	1
a	5

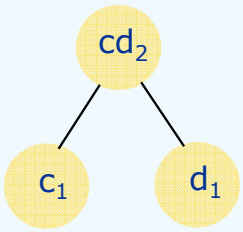
Example of Huffman code

- Ordering the frequency table we have:

Symbol	Frequency
a	5
b	2
r	2
c	1
d	1

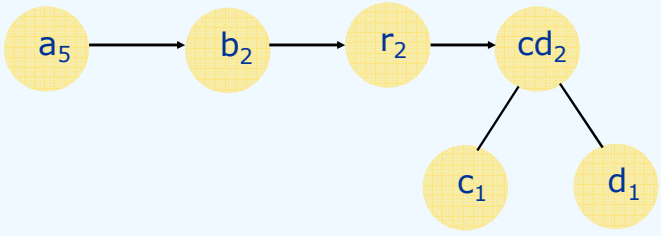
Example of Huffman code

- Fusion of the symbols with lower frequency values
- That is **c** (with frequency 1) and **d** (also with frequency 1), it is obtained a new node that is called "cd" (with frequency 1+1 = 2)
- Now, the new node "cd" is inserted in the data structure (ordered queue)



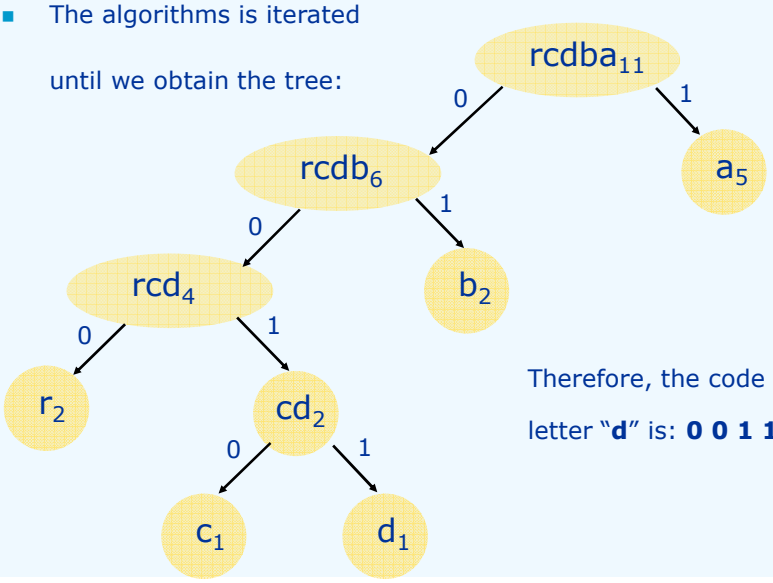
Example of Huffman code

- Table of symbols, represented as an ordered list (non-increasing)



Example of Huffman code

- The algorithms is iterated until we obtain the tree:



Therefore, the code of the letter "**d**" is: **0 0 1 1**

Example of Huffman codes

- Codes table:

Letter	Frequency	Codes
a	5	1
b	2	01
r	2	000
c	1	0010
d	1	0011

- The generated code is **non-ambiguous** and less frequent letters have long codes, and vice versa

Notes and bibliography

- **Huffman coding**
  - http://en.wikipedia.org/wiki/Huffman_coding
- **From ASCII Coding to Huffman Coding**
  - http://www.cs.duke.edu/csed/huff/info/

Algorithms and Data Structures 2010 - 2011

Lesson 8: *Huffman codes*

Luciano Bononi

International Bologna Master in Bioinformatics

University of Bologna

27/05/2011, Bologna