BBM 204: Algorithms Lab.
Instructors : Dr. Engin Demir,
Dr. Adnan Özsoy

Assignment 4
Trees

Fall 2020
Due Date : 31.12.2020 - 23:59

Abstract

In this assignment, you will implement a file compression algorithm based on Huffman coding with trees. At the end of this assignment;

1. your program will compress given file using Huffman algorithm for encoding,
2. your program will decompress given file using the same algorithm for decoding.

1. Introduction and Overview

Huffman algorithm is a powerful algorithm that is used for compressing textual data to make file occupy less space with respect to number of bytes. Intuitively, the algorithm works based on a frequency sorted binary tree to encode the input.
Normally, textual data is stored using ASCII encoding with 8 bits per character. 8 bits fixed encoding is not an efficient way to store large data, thus Huffman encoding uses binary encoding with variable sizes. The advantage of this idea is that characters with high frequency is given short encoding while ones with low frequency is given high encoding.

2. Huffman Encoding

In this assignment you are expected to implement Huffman *encoding*, *decoding*, *serialise* for transfer phases. In order to perform Huffman encoding the steps are;

1. Count the number of occurrences of each character.
2. Build a binary tree where each node represents a character and its count of occurrences in the given file.
3. Build the encoding map by traversing the produced tree to find the binary encoding of each character.
4. Encode given file and output the binary version to the destination file.

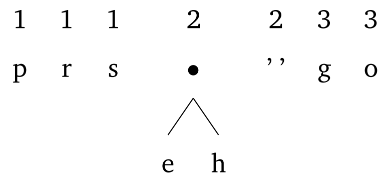
Let us construct the occurrence table, derive the tree and resulting encoding for the input text; "*go go gophers*". We first create the frequency table;

Character	Frequency
g	3
o	3
' '	2
p	1
h	1
e	1
r	1
s	1

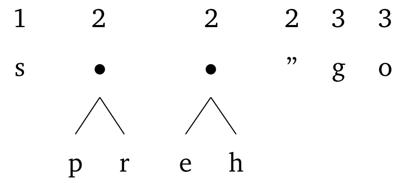
The initial list of tree will be ordered by increasing frequency;

frequency	1	1	1	1	1	2	3	3
character	e	h	p	r	s	’ ’	g	o

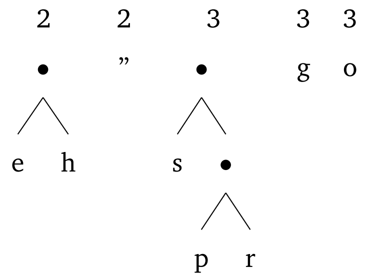
After performing the first merge;



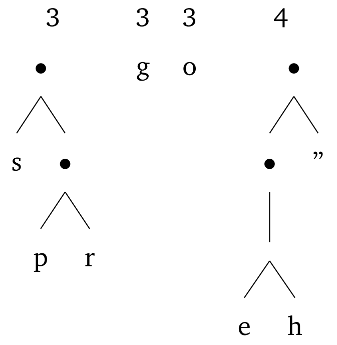
After the second merge;



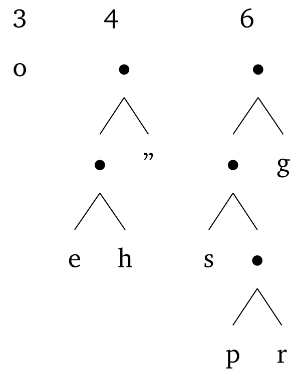
After the third merge;



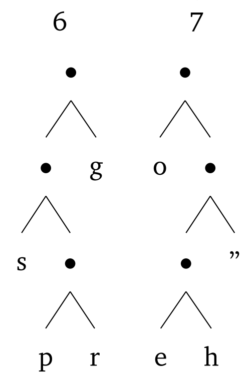
After the 4th merge;



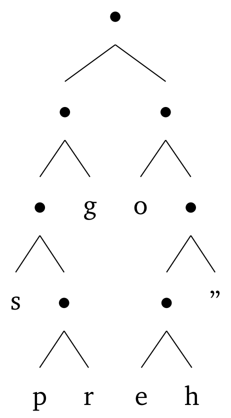
After the 5th merge;



After the 6th merge;



The resulting Huffman coding tree;



Code table obtained by in-order traversal tree;

character	encoding
s	000
p	0010
r	0011
g	01
o	10
e	1100
h	1101
”	111

The encoding will be;
01101110110111011000101101110000110000

3. Huffman Decoding

You are also expected to perform Huffman decoding using the Huffman tree created in the previous section. The decoding algorithm read each bit from the file, one at a time, and use this bit to traverse the tree. If the bit is 0, you move left and if the bit is 1, you move right. You do the movement until you go to the leaf node. Once you reach the leaf node you output the relevant character.

4. Structure of commands

Your code is expected to run with the following commands;

1. **-i input_file.txt -encode** – reads the given file and performs encoding and outputs the encoding [20 pts]

(a) **-s character** – given a character returns its Huffman encoding. The command must not be performed without encoding operation [10 pts]

2. **-i input_file.txt -decode** – reads the given file and performs decoding and outputs the decoded result. [20 pts]
3. **-l** – lists tree [10 pts]

5. Samples

Sample inputs and outputs are;

- input file contains; *BCCABBDDAECCBBAEDDCC*, the output can be;

  101111001101001010010001111101000100001011111 and vice-versa.
- input file contains; *BCAADDDCCACACAC*, the output can be;

  1000111110110110100110110110 and vice-versa.

6. Report

The report should explain the following sections in detail with giving references to the written code samples;

- Explain your encoding algorithm and code step by step in detail [15 pts]
- Explain your decoding algorithm and code step by step in detial [15 pts]
- Explain your list tree command (*-l*) in detail [10 pts]

7. Important Issues

- Projects without a proper Makefile won't be graded.
- Provide your executible(.exe) file.
- All terms in your tree must be in lower case
- Test your program on "dev.cs.hacettepe.edu.tr" before submission. Your submission will be compiled and executed on this machine and this machine only.
- The report will be 40 points, code will be 60 points.
- Please explain your design, data structures, and algorithms.
- Give necessary details without unnecessary clutter.

8. Notes

- Regardless of the length, use UNDERSTANDABLE names to your variables, classes, and Functions
- Write READABLE SOURCE CODE block, these will cost you points.
- Save all your work until the assignment is graded.

- The assignment must be original, INDIVIDUAL work. Duplicate or very similar assignments are both going to be punished. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms or source codes.
- Should you have a question, feel free to ask on Piazza. Be aware that the question has not been asked/discussed before.
- Copying without citing will be considered as cheating. Citing does not make it yours; cited work will get 0 from the respective section.

This file hierarchy must be zipped before submission (Not .rar, only .zip files are supported by the system) <student id>.zip;

- Report.pdf
- *.cpp
- *.h
- .exe
- makefile