SE232, Fall 2018

Lab 4: Huffman Encoding

Assigned: Dec 3rd

Due: Dec 30th, 23:59

TA: Weiye Chen (vorringer@sjtu.edu.cn)

Introduction

Huffman encoding is an example of a lossless compression algorithm that works particularly well on text and, in fact, can be applied to any type of file. It can reduce the storage required by a third or half or even more in some situations.

You are to write a program that allows the user to compress and decompress files using the standard Huffman algorithm for encoding and decoding.

Your program should compress a file by executing

```
./Compression inputfilename outputfilename
```

and decompress a file by executing

```
./Compression -d inputfilename outputfilename
```

where `Compression` is your executable file produced by your code.

Requirement & Hint

1. You are required to complete this lab on Linux.

2. You can find `Huffman Encoding.pdf` on our course website for detailed tutorial.

3. Compressing a file will require reading through the file twice: first to count the characters, and then again when processing each character as part of writing the compressed output. When writing the bit patterns to the compressed file, note that you do not write the ASCII characters '0' and '1' (that wouldn't do much for compression!), instead the bits in the

compressed form are written one-by-one.

You are responsible for freeing memory. Make sure not to leak any tree nodes, and if you allocate any extra memory, be sure to deallocate it!

Grading

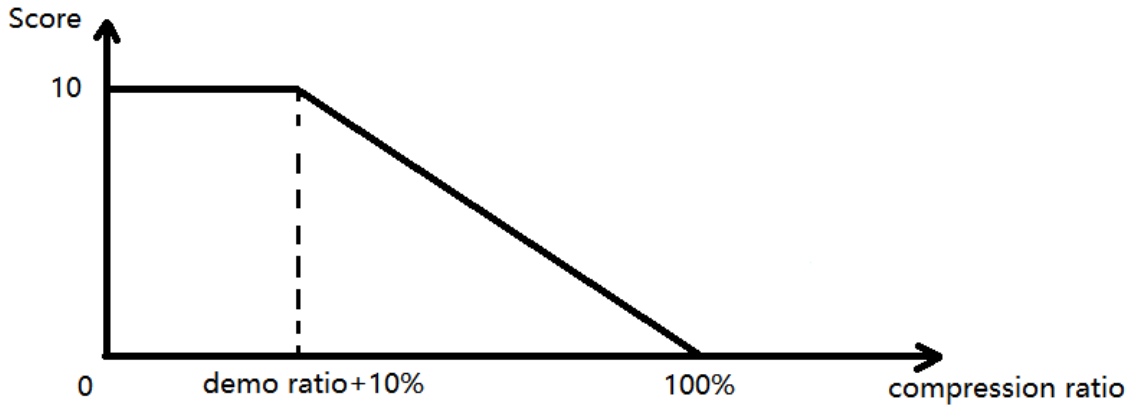
Your implementation will be evaluated using score in Test folder. You can evaluate your implementation by yourself. Try "./score " to evaluate your program. There are 10 test traces for 10 points each.

For trace 0, handle the empty file then you will get 10 points.

For trace 1 – 9:

Compression ratio = sizeof(compressed file) / sizeof(original file).

We have a full score threshold, which equals Demo's compression ratio + 10%. You will get full score if your compression ratio is less than the threshold. Otherwise your score varies linearly from 0 to 10 points.



Style points: We expect you to have good comments, good OOP design and good code style. You will lose up to 10 points for bad code style.

Hand-in

You should work in `Compression` folder. You may add or modify files in this folder. Only keep your `Makefile` can produce executable file named `Compression` correctly when we type `make` in this folder. You should use `handin` to compress your code marked with your student ID via following command:

```
./handin studentID
```

For example, if your student ID is 51703333333, the command should be:

```
./handin 51703333333
```

Hand in `studentID.tar.gz` to our course website.