Sunday, 9 Feb 2025

Hi All,

Your classmate, Lucas Nascimento, asked a couple of questions about proj1. We went back and forth. Then I wrote this to be more specific about the bit operations for step4 on slide 27. You might enjoy reading this approach.

Best regards,
Dr. DeBlois

Hi Lucas,

I'm not sure we're talking about the same things the same way. Check if you agree with these statements:

1. You start with a file of some number of bytes of data in ASCII codes, so each 'letter' has a one-byte ASCII code. You count how many of each character you have. You give short bit codes to frequent characters, and long bit codes to rare characters to get compression.

2. By looking at frequencies of characters in the input file, you figure out the mix of codes that lets you compress the file. Some codes are less than 8 bits, some are more. Making the tree gives you your code list. You store the codes and the code lengths in unsigned integers arrays. The length is in bits.

3. Did you run my executable huffman (copy it to your directory on the server) with the debug switch? To see the codes. They vary in length of bits for each 8-bit ASCII character. You can run it for completeShakespeare.txt by default or for NoDupFreq.txt using the -i switch or for a smaller file. The codes are left-justified.

4. THEN, once you have your codes, you read the file again and for each byte of ASCII data you put the left-justified code into the output buffer. The first one is simple — "or" it in. Use the code length in bits to keep track of the count of bits you put in the buffer. For the next code you add, rightshift the code past the code you have there before you add it so you don't clobber the first one (at the left end). Some codes are less than a byte, some codes are more.

5. Between adding codes, while the number of bits in the buffer is greater than or equal to 8, copy the buffer, shift the high byte to the low byte and write that byte out. Then because the high byte went out, leftshift the buffer to drop off the byte you sent and adjust the count of bits in the buffer down by 8 bits. Check if a second byte can also go out, etc.

6. Set it up to use a very small file to start — NoDupFreq.txt or even smaller. Keep in mind the codes start out left-justified and have a count in bits. You write a byte at a time so you don't have an endian issue.

There are lots of ways to do this. But do see how this way works consistently?

Best regards,
Dr. DeBlois