Q2 report

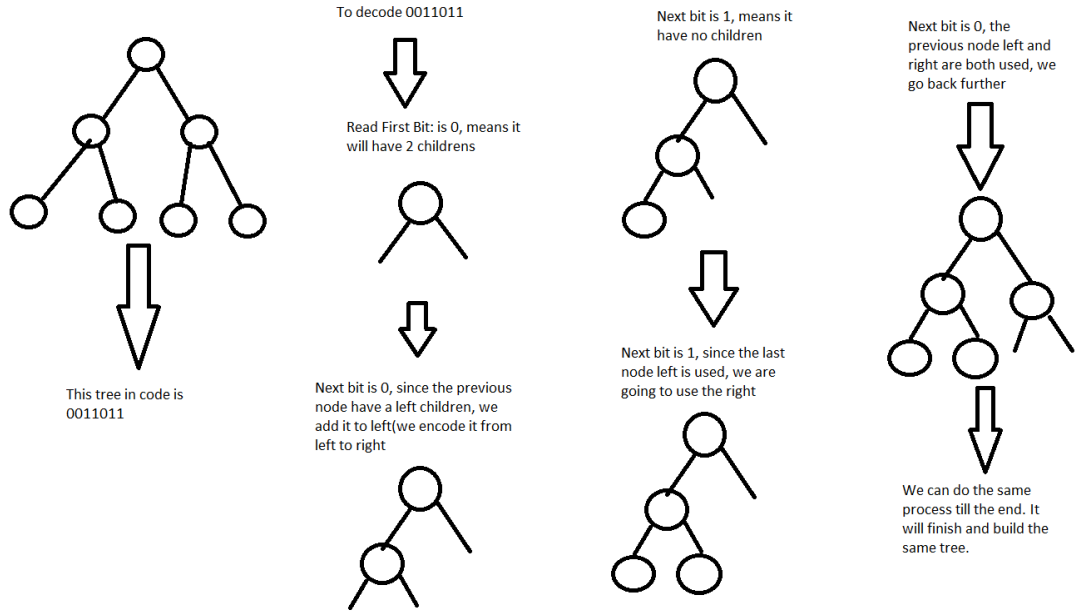
All result are in IMGFiles folder and screen shots are in screenShots folder.

1. Encode tree:

   How to encode the Huffman into binary form is important. First I search on the Internet, and there is a men said, we can use 0 to represent non leaf, use 1 to represent leaf. Below is that website.

   https://stackoverflow.com/questions/759707/efficient-way-of-storing-huffman-tree

   I only looked the idea of using 0 and 1 to represent the tree from self -> left -> right. And I think using this way to find the tree back.



   This is the process of rebuild the tree. It's hard to explain by word, so I draw it. I stored data from most left node to most right node in a list. After I build the Tree, I just put them back in the same order.

   Encode binary to byte:

   Read each 8 length "01" string as an integer.

   Like 00000011 is 3, then we use store (3 – 128) into byte, since the byte range is 127 to -128, so we -128 to avoid overload, and increase the size form 127 to 255.

   Header:

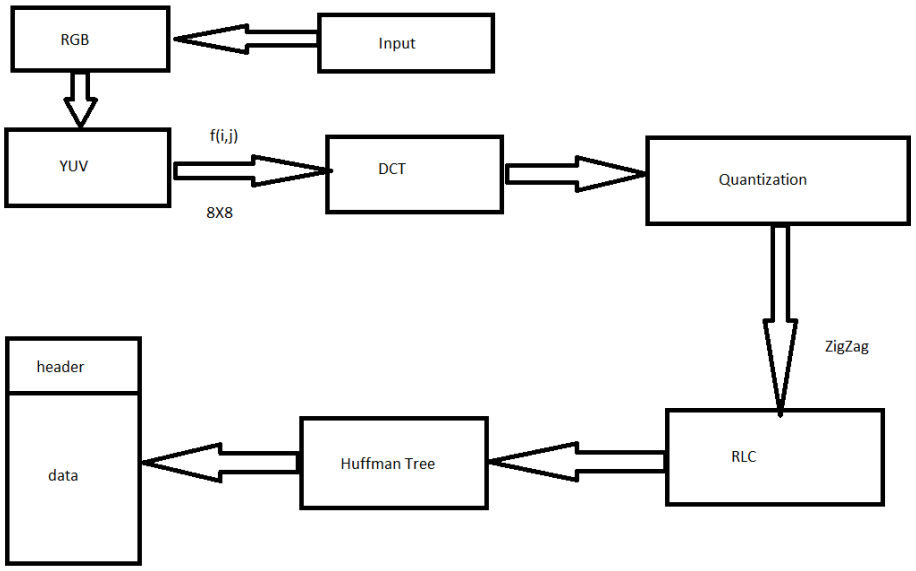
   I learned how wav file stored, there is some number of header, and after that are all data. I used the same way. I stored picture height and width. For all YUV, Output code size, and the number of fill zeros of output code, tree size, and the number of fill zeros of tree, and all keys in tree leaf. Each key is 2 bytes.

   The output is all binary, we store is as byte, so it might need to add some zeros for the last byte. The fill zero is to tell us how many bit zeros in last byte. My header is 29 bit.

   3 for data length, 1 for data fill zero, 2 for tree length, 1 for tree zero, 2 for key length, it times 3, since we have YUV, and 1 for image height and 1 for image width, total is 29bit.

Restore data:
This is all basic Huffman skills. 0 to left, and 1 to right to find data.

Processing flow:



2. Compress time & Decompress time
   BIOS:
   Compress time: 673ms
   Decompress time: 246ms

   Earth:
   Compress time: 463ms
   Decompress time: 187ms

   Fall:
   Compress time: 1500ms
   Decompress time: 362ms

   Nature:
   Compress time: 405ms
   Decompress time: 209ms

   Nature_2:
   Compress time: 1535ms
   Decompress time: 363ms

3. My compression ratio
   BIOS: 9.02
   Earth: 6.68

Fall: 4.04

Nature: 4.63

Nature_2: 4.12

https://convertio.co/zh/ is where I convert .bmp to .JPEG files

https://www.softpedia.com/get/Multimedia/Graphic/Graphic-Others/PSNR.shtml is where I download the .exe file to compare .bmp and .JPEG file PSNR

	original size(KB)	My compression size (KB)	My PSNR (dB)	JPEG (KB)	JPEG PSNR (dB)
BIOS	716	80	27.7	48.2	49.16
earth	368	56	34.2	29.9	48.23
Fall	813	202	27.4	135	41.33
nature	388	84	31.1	53.7	43.95
nature_2	706	172	27.8	113	40.36

JPEG file size is much smaller than my file (about 1: 2), and quality is much better. “PSNR close to 50 is almost the same, between 20 and 30 is able to see the difference by human eyes.” Form Wikipedia.4.

The JPEG file is almost the same when you look at it, all PSNR are above 40. But my file is able to see there are some data loss, all PSNR are about 30. The Wikipedia statement is correct.

5. The tree leaf data have negative number, and the data need two byte to represent, it took me sometime. I originally using one bit and only positive number, it make the decoder cannot get same data, and get the same picture.

   In fall, and nature_2 picture have more output code than 256^2, I couldn’t using 2 byte to represent, it made me these 2 picture fail.

   The compress and decompress is super hard to debug. It took me a lot time to find every single bug, since the data set is so large. But successfully rebuild the image is excited.

   Find a way to encode the Huffman tree and decode Huffman tree is kind fun actually.