Homework 1: Adaptive Huffman Coding

課程: 資料壓縮

姓名: 梁中瀚

學號: 104409017

系級: 資工三

1. Lena.raw compressed file size: 245152(bytes)

其實我在C++壓縮完後的原始大小是1961214(bits), 換算等同於245151.75(bytes), 會多出一些bit是因為C++要湊成1個byte才能輸出。

未壓縮的檔案大小262144(bytes)

Baboon.raw compressed file size: 242230(bytes)

First-order entropy:

原本的Lena.raw的entropy是7.44736

壓縮過後的entropy是0.998738

原本的Baboom.raw的entropy是7.35773

壓縮過後的entropy是0.998689

壓縮過後的檔案的entropy非常接近1是因為只會有兩種output 1跟0，entropy很接近1也代表1跟0出現的機率很相近，接近50%

以Lena.raw為例:

現在要送512\*512個message=262144個messages

最低所需bit數為262144\*7.44736

而壓縮過後的檔案大小為245152\*8=1961214 bits

平均一個message需要1961214/262144(bits)=7.481437

已就是原本的entropy 7.44736+0.034077=7.481437

以上式子符合Noiseless Source Coding Theorem

所以確定壓縮沒有錯誤

Noiseless Source Coding Theorem:

原本圖檔的entropy大概是7點多，所以Huffman coding(lossless coding)壓縮的極限就在7.44736 bits，壓縮過後entropy變成0.998738，lossless coding壓縮的極限就變成一個symbol所需的bits數量最少就需1個

1. Lena.raw DPCM file size: 166836(bytes)

原始大小為1334677(bits), 等同於166834.625(bytes)

Lena.raw DPCM file 的entropy為: 4.13627

Baboon.raw DPCM file size: 209189(bytes)

Baboon.raw DPCM file 的entropy為: 5.3812

Lena.raw DPCM與原始圖片當作symbol set最後輸出的差異為: 626537(bits)=78317.125(bytes)

Baboon.raw DPCM與原始圖片當作symbol set最後輸出的差異為: 33041(bytes)

1. Code的實現方法:

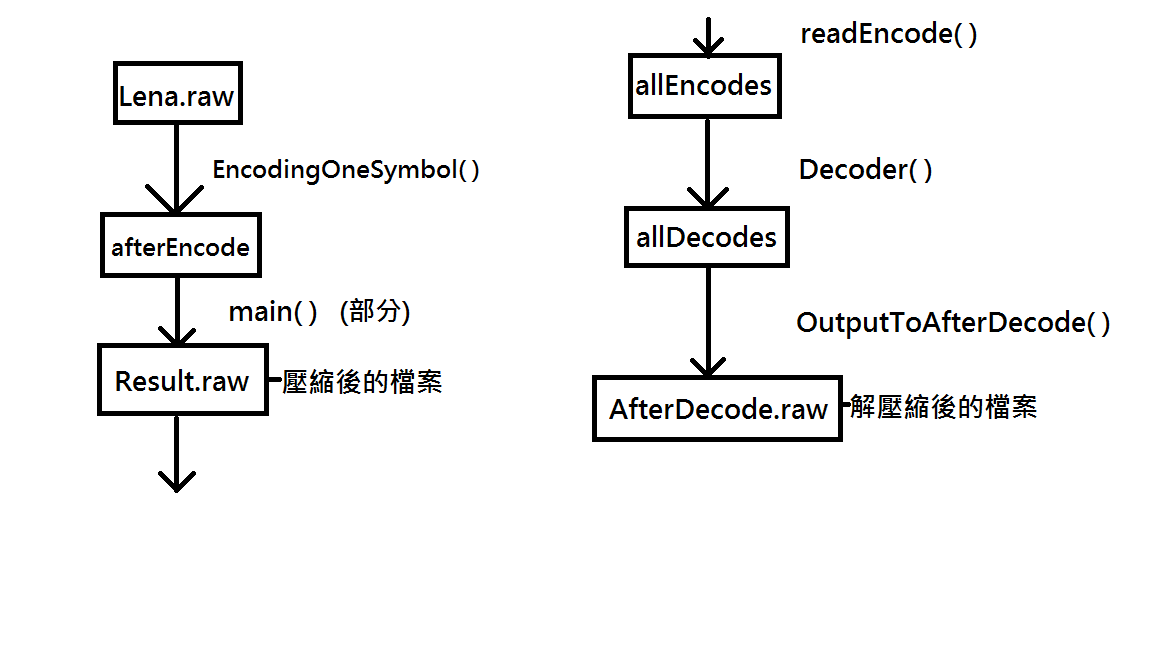
寫一個Class 命名為 chainNode，他是組成樹的基本單位，也就是每一個在樹當中的節點。

接著寫一個class叫做tree，他將會用chainNode時做出一個tree，並提供多種方法，讓caller能夠對tree做特定的操作，這些操作都會跟建立adaptive huffman tree有關。

chainNode簡介:

chainNode裡面除了左右小孩的指標外，還多了一個叫做parent的指標，她會指著該點的parent，目的是為了讓adaptive Huffman tree在維持sibling property的時候，會做的節點交換更容易更快速。

程式執行流程:



Tree裡面有兩種重要函數分別為AddNYT()以及AddNotNYT()，他們會讀入一個symbol並且對整棵樹做調整，類似於update procedure那一張流程圖。

而EncodeOneSymbol()就類似Encoding那一張流程圖，會接收一個輸入symbol，接著判斷它是否為NYT然後callAddNYT/AddNotNYT其中一個函數做數的更新，並且傳回該symbol當前的編碼

輸入的部分是實做readRAW()這函數，他會將Lena.raw以1個byte為單位做輸入，並且會存在img[512][512]裡面

輸出的部分是最麻煩的，因為要集成8個bits(1個byte)才能輸出，所以放在main函數裡面直接實做出來，實作方法為，用一個outBoolArray紀錄8個暫存的bit，當記錄了8個bit之後，就會輸出到檔案裏面。

然後用readEncode，讀入result.raw也就是壓縮後的碼到img[512][512]

再用decoder解壓縮，解壓縮出來的碼會存到allEncodes裡面，再用OutputAfterDecode()輸出成AfterDecode.raw檔案，就會輸出原本的圖片了

所以輸出的檔案有三個:

1. Result.raw (壓縮後的code)
2. AfterDecode (解壓壓縮後的code的結果)
3. ResultDPCM (圖片通過DPCM處理後再壓縮後的code)

Tree提供的變數有:

chainNode\* root; 他是一個指標，並且會永遠指著這棵樹的root node

chainNode\* curNYT; 他是一個指標，並且會永遠指著這棵樹當前的NYT node map<int, list<chainNode\*> > sameweight;

這個變數是類似python的dictionary變數，對他輸入key值，他就會返還一個list，list裡面有所有weight與key值相同的tree 的 nodes

Map<int , chainNode\*> appedsymbol;

這個變數會記錄下所有已經出現過的symbol

Int img[512][512];

記錄了整張圖片每個pixel的值

DPCMarray\* DPCM;

專門為了算DPCM而設的變數，他會儲存原始圖片轉成DPCM後的數值

List<bool> afterEncode;

他會儲存所有從原始圖片被壓縮的bit，也就是call玩encodingOneSymbol()512\*512次之後，afterEncode會儲存壓縮完的code

List<bool> allEncodes;

在call完readEncode()之後，allEncode會儲存所有在Result.raw檔案裏面的資料(bits)

List<unsign char> allDecodes;

儲存所有從decoder deocde出來的pixel