資料壓縮 HW1

學號：113522118
姓名：韓志鴻

一、Huffman

1. **huffman.py 執行流程**：
   i. 讀取 Interleave 和 Plane 內的 baboonRGB.raw 和 lenaRGB.raw。
   ii. 處理單一影像檔案：
      a、 讀取原始影像資料。
      b、 提取 RGB、YUV 和 Y 通道資料。
      c、 計算各種熵。
   iii. process_encoding 進行壓縮：
      a、 build_frequency_table：建立頻率表，計算每種符號的出現頻率。
      b、 build_huffman_tree：建立 Huffman Tree，這個樹反映了資料中各符號的相對頻率，出現次數少的符號會在樹的較深處。
      c、 generate_codes：遞迴遍歷 Huffman Tree，為每個符號分配一個二進位編碼。
      d、 encode_data：將所有編碼組合在一起形成一個長位元字串。
      e、 bits_to_bytes：將生成的長位元字串補到 8 的倍數，同時亦取得補齊的位數。
      f、 最後 write_encoded_txt 將 Huffman 編碼表、補齊資訊以及主要的壓縮內容 bitstream(一個僅包含 0 和 1 的超長字串)記錄起來並寫檔輸出。
   iv. 另外再做 DPCM：對 Y 通道數值進行 DPCM 轉換，每列第一個數值減去 128，之後計算每個數值與前一數值的差值。之後再回到 iii. 進行壓縮。

2. **huffman_decompress.py 執行流程**：
   i. 進行解壓縮：
      a、 parse_encoded_file：建立反向編碼表，取得每種編碼對應的數字，以及主要的壓縮內容 bitstream。
      b、 decode_bitstream：對於 bitstream 內的編碼，轉成對應數值。
      c、 inverse_dpcm：若當前是解經過 DPCM 的壓縮檔，每列第一個數值加上 128，之後的每個數值都要再加上前一個數值。
      d、 save_png：把剛剛得到的解碼資訊儲存成灰階圖片，即可還原成原圖的灰階圖。

二、Adaptive_Huffman

1. adaptive_huffman.py 執行流程：
   i. read_raw_image 讀取 raw 圖片，並且透過 rgb_to_y_channel 把 RGB 轉換成 Y 通道。
   ii. process_adaptive_encoding 進行壓縮：
      a、 對於第 1 次出現的新符號，先將其轉換成固定長的二進位並放入結果 bitstream(一個僅包含 0 和 1 的超長字串)。
      b、 在 NYT 節點處分裂出 2 個子節點：左子為新的 NYT，右子為此新符號的葉節點。
      c、 更新 mapping，下次遇到相同符號時可直接查找對應的葉節點。
      d、 更新 NYT 指向到新分裂出的 NYT 節點。
      e、 對於已經見過的符號，直接從對應葉節點取得動態 Huffman 編碼即可。

f、 update_tree：使用 FGK 演算法從下而上更新樹。

g、 bits_to_bytes：將生成的長位元字串補到 8 的倍數，同時亦取得補齊的位數。

h、 最後 write_encoded_txt 將編碼資訊記錄起來並寫檔輸出。

iii. 另外再做 DPCM：對 Y 通道數值進行 DPCM 轉換，每列第一個數值減去 128，之後計算每個數值與前一數值的差值。之後再回到 iii. 進行壓縮。

2. adaptive_huffman_decompress.py 執行流程：

i. 進行解壓縮：

a、 adaptive_huffman_decode：從 bitstream 中讀取固定長度位元並轉回十進位。

b、 將該十進位數值存在 decoded 陣列，後續這個陣列會回傳作為後續還原圖片的依據，若是先前已經出現過的數值就直接加入陣列就好。

c、 在 NYT 節點處分裂出 2 個子節點：左子為新的 NYT，右子為此新符號的葉節點。

d、 inverse_dpcm：若當前是解經過 DPCM 的壓縮檔，每列第一個數值加上 128，之後的每個數值都要再加上前一個數值。

e、 save_png：把剛剛得到的解碼資訊儲存成灰階圖片，即可還原成原圖的灰階圖。

三、執行方式與結果

1. Entropy：

Image / Entropy	Interleave - baboonRGB	Interleave - lenaRGB	Plane - baboonRGB	Plane - lenaRGB
H(R)	7.7067	7.2531	7.7067	7.2531
H(G)	7.4744	7.5940	7.4744	7.5940
H(B)	7.7522	6.9684	7.7522	6.9684
H(Y)	7.3583	7.4451	7.3583	7.4451
H(U)	6.5334	5.6531	6.5334	5.6531
H(V)	6.5134	5.6105	6.5134	5.6105
Joint H(R, G, B)	17.7368	16.8355	17.7368	16.8355
Joint H(Y, U, V)	17.1584	15.4941	17.1584	15.4941
Conditional H(RGB|Left)	0.2795	1.1510	0.2795	1.1510
Conditional H(RGB|Upper)	0.2798	1.1557	0.2798	1.1557
Conditional H(YUV|Left)	0.8552	2.4779	0.8552	2.4779
Conditional H(YUV|Upper)	0.8565	2.4792	0.8565	2.4792
Conditional H(Y|Left)	6.1442	4.8836	6.1442	4.8836
Conditional H(Y|Upper)	6.4209	4.5314	6.4209	4.5314

2. 壓縮率：

i. Huffman

Image	原始 Y 大小	壓縮後	壓縮率
Interleave - baboonRGB	262144 bytes	241848 bytes	92.26%
Interleave - baboonRGB (DPCM)	262144 bytes	208919 bytes	79.70%
Interleave - lenaRGB	262144 bytes	244697 bytes	93.34%
Interleave - lenaRGB (DPCM)	262144 bytes	166904 bytes	63.67%
Plane - baboonRGB	262144 bytes	241848 bytes	92.26%
Plane - baboonRGB (DPCM)	262144 bytes	208919 bytes	79.70%
Plane - lenaRGB	262144 bytes	244697 bytes	93.34%
Plane – lenaRGB (DPCM)	262144 bytes	166904 bytes	63.67%

ii. Adaptive_Huffman

Image	原始 Y 大小	壓縮後	壓縮率
Interleave - baboonRGB	262144 bytes	242263 bytes	92.42%
Interleave - baboonRGB (DPCM)	262144 bytes	209366 bytes	79.87%
Interleave - lenaRGB	262144 bytes	245100 bytes	93.50%
Interleave - lenaRGB (DPCM)	262144 bytes	167258 bytes	63.80%
Plane - baboonRGB	262144 bytes	242263 bytes	92.42%
Plane - baboonRGB (DPCM)	262144 bytes	209366 bytes	79.87%
Plane - lenaRGB	262144 bytes	245100 bytes	93.50%
Plane – lenaRGB (DPCM)	262144 bytes	167258 bytes	63.80%

3. 執行方式：

i. cd Huffman
python huffman.py
python huffman_decompress.py
cd ..
cd Adaptive_Huffman
python adaptive_huffman.py
python adaptive_huffman_decompress.py
cd ..

ii. 或是直接執行：**bash run.sh**

iii. 產生的壓縮檔與解壓縮的灰階圖皆儲存於 Results 資料夾內。