資料壓縮作業一

112522115 林語潔

一、 Lossless Coding

1. 程式說明:

Huffman 編碼的過程首先，統計符號的頻率，接著建立 Huffman 樹以及依序對 symbol 進行編碼。最後，encoder 將所有編碼以八個位元為一組轉換成位元組 (byte)，並將其輸出為文字 (txt) 檔案，同時傳送頻率表給 decoder。

Decoder 接收到一個儲存編碼的文字 (txt) 檔案以及頻率表。解碼器根據頻率表建立相同的 Huffman 樹，然後根據該樹對文字 (txt) 檔案中的編碼進行解碼。最終，解碼器將解碼後的資料還原成原始圖像。

DPCM 的作法: 在一張 256x256 的圖片中，保留第一個像素值，對於其餘的像素，減去前一個像素的值。若遇到整除 256 的像素，則減去該像素的垂直向上的像素值。

2. 執行

python huffman.py [-dpcm] {path}

[-dpcm]: option，表示使用 DPCM 編碼

{path}: 表示欲壓縮的檔案路徑

3. 實驗結果

表 1 圖片 lena Huffman 實驗結果

Filename	Compressed File Size (KB)	Entropy	Bits Average	Redundancy
lena_b	8	0.969	1.0	0.030
lena _b (DPCM)	9	0.460	1.075	0.615
lena _halftone	8	0.836	1.0	0.163
lena _halftone (DPCM)	12	1.341	1.381	0.0405
lena	61	7.595	7.624	0.029
lena (DPCM)	48	5.863	5.897	0.034

表 2 圖片 baboon Huffman 實驗結果

Filename	Compressed File Size (KB)	Entropy	Bits Average	Redundancy
baboon_b	8	0.969	1.0	0.0309

baboon_b (DPCM)	10	0.784	1.157	0.372
baboon_halftone	8	0.9613	1.0	0.0386
baboon_halftone (DPCM)	13	1.553	1.566	0.0127
baboon	59	7.240	7.267	0.0268
baboon (DPCM)	52	6.351	6.377	0.0253

4. 程式執行結果

```
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py ./Data/PNG/lena_b.png
=> Entropy: 0.9696836065955972
=> Bits average: 1.0
=> Redundancy: 0.030316393404402 84
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py -dpcm ./Data/PNG/lena_b.png
=> Entropy: 0.4601219671744273
=> Bits average: 1.0751800537109375
=> Redundancy: 0.6150580865365102
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py ./Data/PNG/lena_halftone.png
=> Entropy: 0.8368129054145714
=> Bits average: 1.0
=> Redundancy: 0.16318709458542857
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py -dpcm ./Data/PNG/lena_halftone.png
=> Entropy: 1.3413432324211818
=> Bits average: 1.3819427490234375
=> Redundancy: 0.04059951660225569
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py ./Data/PNG/lena.png
=> Entropy: 7.595364724994117
=> Bits average: 7.624542236328125
=> Redundancy: 0.02917751133400781
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py -dpcm ./Data/PNG/lena.png
=> Entropy: 5.863330892326578
=> Bits average: 5.8979034423828125
=> Redundancy: 0.034572550056234164
PS D:\NCU Course\112-2 資料壓縮\HW1>
```

圖 1 圖片 lena 程式執行結果圖

```
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py ./Data/PNG/baboon_b.png
=> Entropy: 0.9690876812687887
=> Bits average: 1.0
=> Redundancy: 0.030912318731211252
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py -dpcm ./Data/PNG/baboon_b.png
=> Entropy: 0.7845876606133153
=> Bits average: 1.1571502685546875
=> Redundancy: 0.3725626079413722
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py ./Data/PNG/baboon_halftone.png
=> Entropy: 0.961359072046256
=> Bits average: 1.0
=> Redundancy: 0.038640927953743964
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py -dpcm ./Data/PNG/baboon_halftone.png
=> Entropy: 1.5536534899390537
=> Bits average: 1.5664215087890625
=> Redundancy: 0.012768018850008778
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py ./Data/PNG/baboon.png
=> Entropy: 7.240645542146741
=> Bits average: 7.2675018310546875
=> Redundancy: 0.02685628890794689
PS D:\NCU Course\112-2 資料壓縮\HW1> python huffman.py -dpcm ./Data/PNG/baboon.png
=> Entropy: 6.351710552067522
=> Bits average: 6.3770599365234375
=> Redundancy: 0.02534938445591539
```

圖 2 圖片 baboon 程式執行結果圖

5. 結論

兩張圖片原始檔案(RAW)皆為 64 KB，因此壓縮後的檔案大小皆小於原始檔案。另外由表 1、2 可見，在非二值化的影像使用 DPCM，可以有效降低 entropy。

二、 Binary Arithmetic Coding, QM Encoder

1. 程式說明

壓縮的圖片分為三個類別，其中 halftone 和 binary 屬於二值化影像，讀入圖片後可以直接進行 QM 編碼。而一般原始檔案則是 0～255 的值，因此在轉成八位元二進位表示後，會先依照 index 拆分成 8 個 bit-plane，在依序進行 QM 編碼。

在 QM 編碼中，會依序讀入 0 或 1 的字元，來決定使用 MPS (More Probable Symbol) 或 LPS (Less Probable Symbol) 進行編碼，接著會依序更新 interval A 和 C，並執行 Estimate() 以及 Renorm_e() 兩個 function，其中 Estimate 是用來檢查是否要進行 MPS 和 LPS 交換，並查表更新 Qc 值，而 Renorm_e() 則是為了確保 interval A 不會縮小到過小的程度。

2. 執行

python qm_coder.py [-graycode] {path}

[-graycode]: option，表示使用 gray code 編碼

{path}: 表示欲壓縮的檔案路徑

3. 實驗結果

表 3 圖片 lena QM Encoder 實驗結果

Filename	Bit-stream length (bits)	Use Gray Code (bits)
lena_b.png	42120	
lena_halftone.png	55277	
lena.png	475951	423654
lena.png (1th bit-plane)	59399	39997
lena.png (2th bit-plane)	59524	42911
lena.png (3th bit-plane)	59371	49976
lena.png (4th bit-plane)	59621	53626
lena.png (5th bit-plane)	59306	58404
lena.png (6th bit-plane)	59440	59285
lena.png (7th bit-plane)	59705	59669
lena.png (8th bit-plane)	59585	59786

表 4 圖片 baboon QM Encoder 實驗結果

Filename	Bit-stream length (bits)	Use Gray Code (bits)
baboon_b.png	47440	
baboon_halftone.png	59353	
baboon.png	476930	426990

baboon.png (1th bit-plane)	59546	48856
baboon.png (2th bit-plane)	59567	29019
baboon.png (3th bit-plane)	59507	53102
baboon.png (4th bit-plane)	59504	57803
baboon.png (5th bit-plane)	59725	59432
baboon.png (6th bit-plane)	59616	59542
baboon.png (7th bit-plane)	59698	59592
baboon.png (8th bit-plane)	59767	59644

4. 程式執行結果

```
PS D:\NCU Course\112-2 資料壓縮\HW1> python .\qm_coder.py ./Data/PNG/lena_b.png
file : lena_b Bit-stream len: 42120
PS D:\NCU Course\112-2 資料壓縮\HW1> python .\qm_coder.py ./Data/PNG/lena_halftone.png
file : lena_halftone Bit-stream len: 55277
PS D:\NCU Course\112-2 資料壓縮\HW1> python .\qm_coder.py ./Data/PNG/lena.png
file : lena_bitplane_0 Bit-stream len: 59399
file : lena_bitplane_1 Bit-stream len: 59524
file : lena_bitplane_2 Bit-stream len: 59371
file : lena_bitplane_3 Bit-stream len: 59621
file : lena_bitplane_4 Bit-stream len: 59306
file : lena_bitplane_5 Bit-stream len: 59440
file : lena_bitplane_6 Bit-stream len: 59705
file : lena_bitplane_7 Bit-stream len: 59585
Total Bit-stream len: 475951
PS D:\NCU Course\112-2 資料壓縮\HW1> python .\qm_coder.py -graycode ./Data/PNG/lena.png
file : lena_bitplane_0 Bit-stream len: 39997
file : lena_bitplane_1 Bit-stream len: 42911
file : lena_bitplane_2 Bit-stream len: 49976
file : lena_bitplane_3 Bit-stream len: 53626
file : lena_bitplane_4 Bit-stream len: 58404
file : lena_bitplane_5 Bit-stream len: 59285
file : lena_bitplane_6 Bit-stream len: 59669
file : lena_bitplane_7 Bit-stream len: 59786
Total Bit-stream len: 423654
```

圖 3 圖片 lena 程式執行結果圖

```
PS D:\NCU Course\112-2 資料壓縮\HW1> python .\qm_coder.py ./Data/PNG/baboon_b.png
file : baboon_b Bit-stream len: 47440
PS D:\NCU Course\112-2 資料壓縮\HW1> python .\qm_coder.py ./Data/PNG/baboon_halftone.png
file : baboon_halftone Bit-stream len: 59353
PS D:\NCU Course\112-2 資料壓縮\HW1> python .\qm_coder.py ./Data/PNG/baboon.png
file : baboon_bitplane_0 Bit-stream len: 59546
file : baboon_bitplane_1 Bit-stream len: 59567
file : baboon_bitplane_2 Bit-stream len: 59507
file : baboon_bitplane_3 Bit-stream len: 59504
file : baboon_bitplane_4 Bit-stream len: 59725
file : baboon_bitplane_5 Bit-stream len: 59616
file : baboon_bitplane_6 Bit-stream len: 59698
file : baboon_bitplane_7 Bit-stream len: 59767
Total Bit-stream len: 476930
PS D:\NCU Course\112-2 資料壓縮\HW1> python .\qm_coder.py -graycode ./Data/PNG/baboon.png
file : baboon_bitplane_0 Bit-stream len: 48856
file : baboon_bitplane_1 Bit-stream len: 29019
file : baboon_bitplane_2 Bit-stream len: 53102
file : baboon_bitplane_3 Bit-stream len: 57803
file : baboon_bitplane_4 Bit-stream len: 59432
file : baboon_bitplane_5 Bit-stream len: 59542
file : baboon_bitplane_6 Bit-stream len: 59592
file : baboon_bitplane_7 Bit-stream len: 59644
Total Bit-stream len: 426990
```

圖 4 圖片 baboon 程式執行結果圖

5. 結論:

   由表 3、4 可見，相較於一般二進制編碼，使用 gray code 編碼方式明顯降低儲存的編碼長度，這是因為經過 gray code 轉換後，相鄰的 codeword 只有一位二進制數位不同，因此能大幅降低編碼長度。