Data Compression HW1

Huffman coding

壓縮作法

原始資料

- 使用C++的map來記錄原始資料各個資料出現的次數
- 並且用priority_queue來實作minHeap來建huffman tree
- 建完huffman tree之後就用DFS來走訪把各個資料會轉換成的編碼找出來之後就開始進行壓縮
- 在壓縮的時候用一個int的buffer來記錄現在裡面有多少bit如果塞滿了8個bit再寫進compressed file
- 壓縮結果

```
Enter the file to compress: baboon.raw
[+] The compressd file will be named "compressed"
[+] Compressing...
[+] Compress complete!!
[+] Original file size = 65536 byte
[+] Compressed file size = 59536 byte
[+] Compress rate = 90.8447 %
[+] Entropy = 7.24065
```

DCPM

- DPCM是應用相鄰的兩個pixel資料會很接近所以就用相鄰的資料來做壓縮而不直接使用原始資料來壓縮
- 這邊DPCM是使用當前資料左邊的資料來做預測
- 因為要用左邊的資料來預測所以必須事先給定圖片的長寬不然會拿到圖片最右邊的資料跟最左邊的資料導致算出來結果的差值很大
- 實作的流程如下
    - 使用當前的資料減去左邊的資料所以得到差值得結果會落在-255~255這個區間
    - 但是這樣要用9個bit來記錄所以就把他除2讓結果落在-127.5~127.5這樣變成只需要8個bit就夠了
    - 再來把它加上128這樣差值的結果都是正的會落在0~255之間
    - 只要把差值減掉128再乘以2加上左邊的資料就能得到重建後的資料了
    - 因為重建的時候沒有原始資料所以在算差值的時候是拿當前這個原始資料和左邊那筆重建的資料來算差值
    - 而最左邊的資料沒有左邊的資料所以就拿128當作是他左邊的資料

```
for (int i = 0; i < height; i++)
{
    for (int j = 0; j < width; j++)
    {
        if (j == 0)
        {
            errBuf.push_back((infile.get() - 128) / 2 + 128);
```

```
            rebuildBuf.push_back(128 + (errBuf.back() - 128) * 2);
        }
        else
        {
            errBuf.push_back((infile.get() - rebuildBuf.back()) / 2 + 128);
            rebuildBuf.push_back(rebuildBuf.back() + (errBuf.back() - 128) * 2);
        }
    }
}
```

- 因為只要有差值就能重建圖片所以就把所有的差值紀錄下來並把這些差值拿來壓縮
- 做huffman encoding的方式和原始資料的方式一樣都是去建huffman tree再來做encode
- 壓縮結果

```
> 2
Enter the file to compress: baboon.raw
[+] The compressd file will be named compressed_DPCM
[+] Compress complete!!
[+] Original file size = 65536 byte
[+] Compressed file size = 43736 byte
[+] Compress rate = 66.7358 %
[+] Entropy = 5.30714
```

解壓縮做法

原始資料codewords傳遞方式

- 一開始先傳送這個壓縮的檔案總共用用到多少資料像是如果只用到a,b,c就傳3
- 這樣就知道前面有多少資料是用在建huffman tree而不是真正原始檔案的資料
- 再來利用DFS去走訪huffman tree如果遇到的是interal node就輸出0如果是leaf node就輸出1並輸出這個node的資料
- 像是下面這個tree的leaf node是a, b, c

```
    *
   / \
  *   a
 / \
b   c
```

- 用DFS走訪之後就會輸出`001b1c1a`
- 這樣decoder收到之後就知道0是interal node要繼續往下走1就代表這個node是leaf node在這個node填上1後面的資料
- 像是上面例子1後面是b就把node填上b就這樣一直做到所有leaf node的數量跟一開始收到的數量一樣就知道huffman tree已經重建完後面的資料都是壓縮的資料
- 這樣decoder有跟encoder一樣的huffman tree然後就可以進行解壓縮

DPCM的codewords傳遞方式

- 因為DPCM適用相鄰的資料來做預測所以需要知道原始圖片的長跟寬
- 所以一開始先傳送長跟寬分別是多少
- 再來就跟用原始資料做壓縮的做法一樣把huffman tree的資料傳過去讓decoder去重建huffman tree這樣就能進行解壓縮
- 而因為DPCM是拿相鄰資料相檢的結果做壓縮最後解壓縮出來是兩兩相鄰資料的差值
- 把所有的差值按照長寬排好減去128再乘以2加上左邊那筆重建後的資料就能把整個原始資料重建回來

結果

	File	Original File Size (byte)	Compressed File Size without header (byte)	Compressed Ratio	Entropy
Origin Data	baboon.raw	65536	59536	90.8447 %	7.24065
	lena.raw	65536	62461	95.3079 %	7.59536
DPCM	baboon.raw	65536	43736	66.7358 %	5.30714
	lena.raw	65536	39611	60.4416 %	4.80015

- 從結果可以看到利用DPCM來做壓縮之後得到檔案明顯比裡用原始資料做壓縮的檔案還要小而且entropy也比較小

QM coder

- 這裡實做了三個部分
  - gray level image根據bit-plane來做壓縮
  - gray level image做gray code之後再根據bit-plane做壓縮
  - binary image做壓縮

gray level image

- 因為是灰階的資料所以每一個pixel的值都是0~255而0~255有8個bit所以就按照最高位到最低位分成8個bit-plane
- 為了要分成不同bit-plane來做壓縮所以一開始要先把所有的資料都讀進來再來按照不同bit-plane做壓縮
- 假設所有資料是123(01111011) 124(01111100) 125(01111101)
- 從每個資料的MSB往LSB取這樣進到encoder的bit stream就會是000111111111...101按照這樣來做壓縮
- 而每一筆資料取完他的最高位送進encoder之後就把資料左移1這樣第7個bit就會被丟掉第6個bit就會變成最高位這樣只要一直取最高位就可以了
- 再來就按照QM coder的規則去做壓縮一樣使用buffer來記錄encoder輸出的bit如果buffer滿了就寫到compressed file
- 用一個bitLength來統計這個bit-plane總共長度為多少所以當encoder需要輸出的時候就把bitLength加一

- 最後統計結果

```
Enter the file to compress: baboon.raw
[+] Start encoding gray level image...
7 bit-plane's bit-stream length = 51485
6 bit-plane's bit-stream length = 51260
5 bit-plane's bit-stream length = 60211
4 bit-plane's bit-stream length = 61023
3 bit-plane's bit-stream length = 61208
2 bit-plane's bit-stream length = 61478
1 bit-plane's bit-stream length = 61294
0 bit-plane's bit-stream length = 61336
[+] Encode complete!!
```

gray level image with gray code

- 一樣是灰階的圖片所以按照不同bit-plane來做處理
- 這邊另外對每一筆資料做gray code之後再把資料輸入到encoder裡去做處理
- gray code算法是把每一筆資料跟這筆資料除以2的結果做xor就能得到gray code算出gray code之後再輸入到encoder裡面去做處理
- 用一個bitLength來統計這個bit-plane總共長度為多少所以當encoder需要輸出的時候就把bitLength加一
- 最後統計結果

```
Enter the file to compress: baboon.raw
[+] Start encoding gray level image with gray code...
7 bit-plane's bit-stream length = 51485
6 bit-plane's bit-stream length = 64536
5 bit-plane's bit-stream length = 51199
4 bit-plane's bit-stream length = 54932
3 bit-plane's bit-stream length = 60435
2 bit-plane's bit-stream length = 61232
1 bit-plane's bit-stream length = 61462
0 bit-plane's bit-stream length = 61324
[+] Encode complete!!
```

binary image

- 因為binary image的資料直接就是0或是1所以直接把他送進encoder就好不用再另外做處理
- 但是為了方便所以把0或1左移7這樣他就會變成最高位就可以直接拿原本encode那部分的code來用不用再另外寫