

國立中央大學資訊工程學系

資料壓縮期末報告

LZMA 與壓縮圖片

梁中瀚

資工三

目錄

一、前言

二、LZMA 壓縮後格式簡介

(一)、header 內部構造

(二)、LZMA Compressed data

三、lzma、jpeg 與先壓縮成 jpeg 再放到 lzma 壓縮 Lena.raw 的比較

(一)、方法

(二)、數據比較

(三)、結果

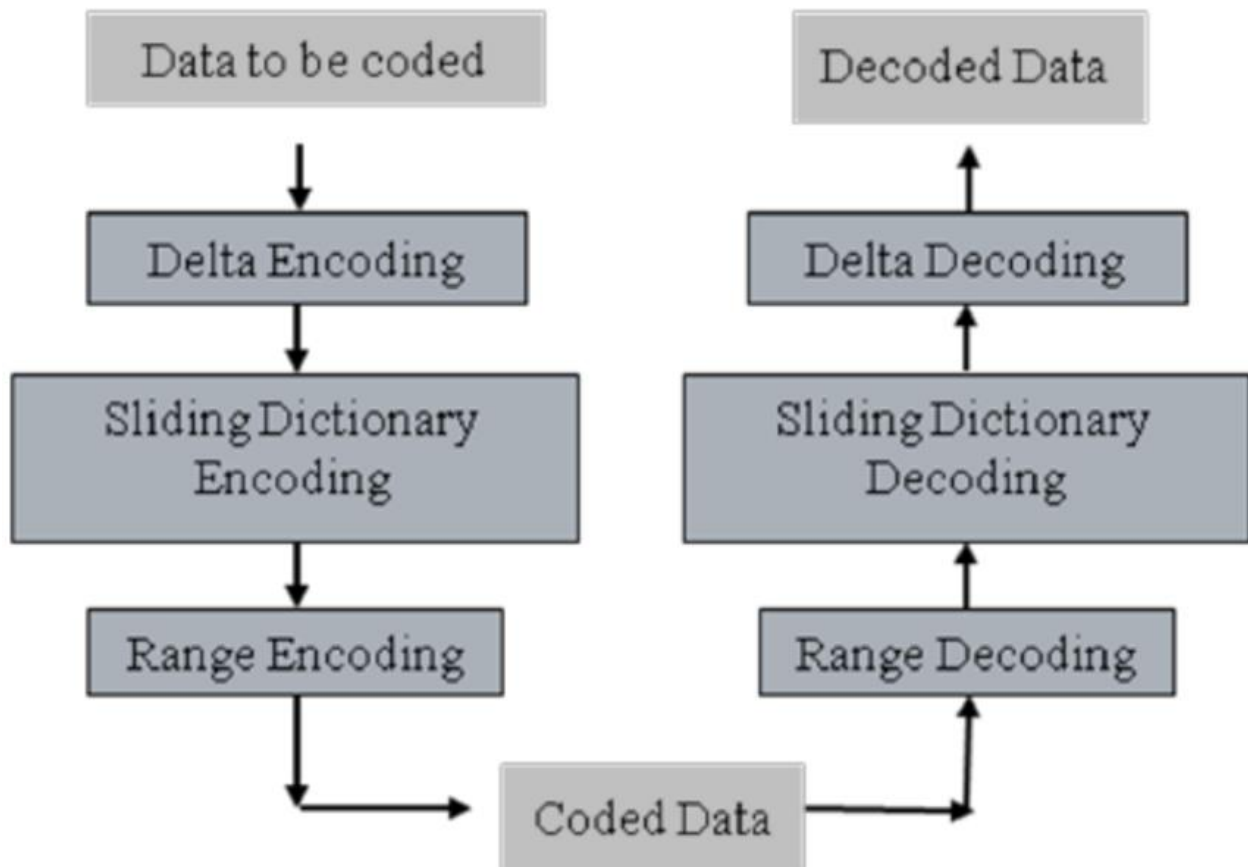
一、前言

Lzma 為 7z 壓縮格式所支援的壓縮演算法，而 7z 又是近期大多數人會使用到的壓縮工具，並且大多數人都是直接將檔案放到壓縮檔裡面壓縮，實際測試先將圖片壓縮一次後，再放到 lzma 壓縮，與只用一次圖片壓縮的方式比較，觀察其檔案的大小。

Lzma 是基於 lz77 的壓縮演算法，並且結合 range coding(類似 arithmetic encoding，但是可以用非二進位來壓縮)。

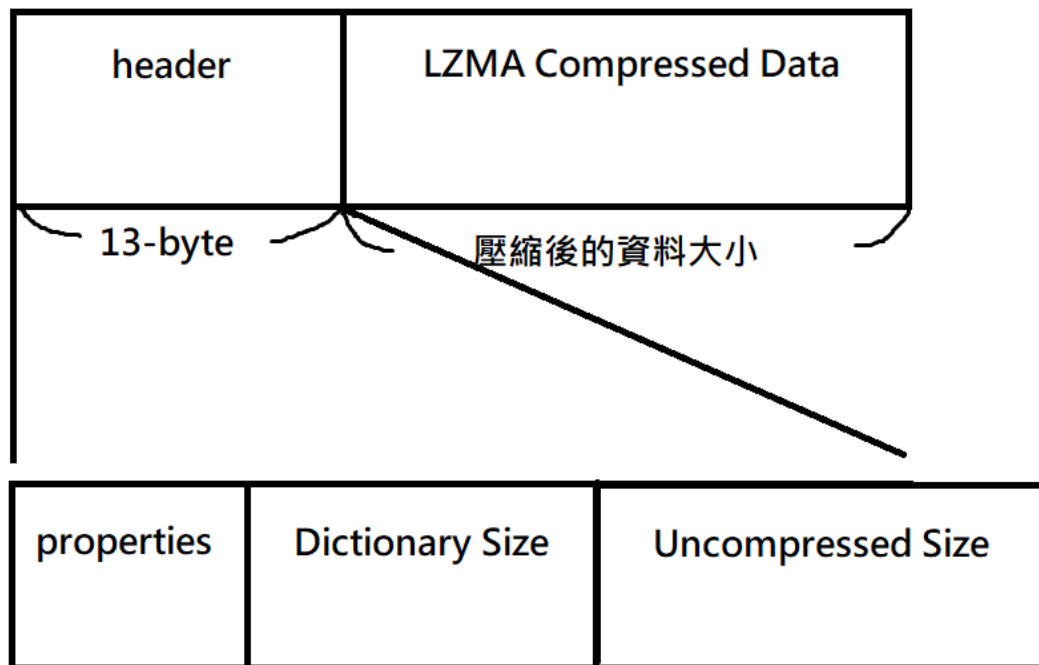
LZMA 與 lz77 的不同點在於 Lzma 會記錄下近期 4 個 match 用到的 distance 長度(在 window 哪裡找到 look ahead buffer 的字串)，讓 encoder 不會一直紀錄重複的資料，並且會用”很大”的 dictionary(window)紀錄近期 input 的字串，window 大小可以從 4KB 到 4GB，但是 7z file manager 只有提供 64KB 到 64MB 的字典大小。

壓縮流程:



二、Lzma 的壓縮後格式簡介:

.lzma 檔案就是被 lzma 壓縮後的資料格式，它包含了兩個部分，header 以及 LZMA Compressed Data，如下圖。



(一)、header 內部構造:

1. Properties:

Properties 內部由三個變數所組成:

(1) lc: literal context 的 bits 數量，範圍是[0, 8]

之前的 byte 當中有多少是 1 的 bit 會拿來當作 context(前後文)。

(2) lp: literal position 的 bits 數量，範圍是[0, 4]

(3) pb: position 的 bits 數量，範圍是[0, 4]

最後用這個式子把以上三個變數 encode:

$$\text{Properties} = (\text{pb} * 5 + \text{lp}) * 9 + \text{lc}$$

Decode 就會用以下的 sudo code 來達成:

```
pb = properties / (9 * 5);
```

```
properties -= pb * 9 * 5;
```

```
lp = properties / 9;
```

```
lc = properties - lp * 9;
```

2. Dictionary Size:

用 unsigned 32-bit little endian integer 來儲存字典大小，所以總共是 4 個 byte。

但是為了提供好的移植性(不同硬體)，應該只用 2^n 或者是 $2^n + 2^{(n-1)}$ 這兩種 size。

3. Uncompressed Size:

未壓縮大小是用 unsigned 64-bit little endian integer 來儲存的。

比較特別的是，如果 64 個 bit 都是 1 的話，代表未壓縮大小是未知的。

(二)、LZMA Compressed Data

經由 LZMA 壓縮過後的資料會存放在這個區塊。

壓縮後的形式為 bit stream，並且用 adaptive binary range coder 做 encode，而 bit stream 會被劃分成很多 packets，packets 的形式如下：

packed code (bit sequence)	packet name	packet description
0 + byteCode	LIT	A single byte encoded using an adaptive binary range coder.
1+0 + len + dist	MATCH	A typical LZ77 sequence describing sequence length and distance.
1+1+0+0	SHORTREP	A one-byte LZ77 sequence. Distance is equal to the last used LZ77 distance.
1+1+0+1 + len	LONGREP[0]	An LZ77 sequence. Distance is equal to the last used LZ77 distance.
1+1+1+0 + len	LONGREP[1]	An LZ77 sequence. Distance is equal to the second last used LZ77 distance.
1+1+1+1+0 + len	LONGREP[2]	An LZ77 sequence. Distance is equal to the third last used LZ77 distance.
1+1+1+1+1 + len	LONGREP[3]	An LZ77 sequence. Distance is equal to the fourth last used LZ77 distance.

1. LIT: literal，代表單獨的 byte 並且用 adaptive binary range coder encode，新出現的 character。
2. MATCH: 標準的 LZ77 格式，有長度(在 window 找到 look ahead buffer 的 code 的長度)還有距離(look ahead buffer 的 code 出現在 window 的哪裡)
3. SHORTREP: 一個 byte 的 LZ77 序列，距離等同於上一個 LZ77 使用的距離

4. LONGREP[n]: 一個 LZ77 的序列，並且距離等同於上 n 個 LZ77 的距離

長度的 encode 用以下的表格:

Length code (bit sequence)	Description
0+ 3 bits	The length encoded using 3 bits, gives the lengths range from 2 to 9.
1+0+ 3 bits	The length encoded using 3 bits, gives the lengths range from 10 to 17.
1+1+ 8 bits	The length encoded using 8 bits, gives the lengths range from 18 to 273.

Distance 用以下表格編碼:

6-bit distance slot	Highest 2 bits	Fixed 0.5 probability bits	Context encoded bits
0	00	0	0
1	01	0	0
2	10	0	0
3	11	0	0
4	10	0	1
5	11	0	1
6	10	0	2
7	11	0	2
8	10	0	3
9	11	0	3
10	10	0	4
11	11	0	4
12	10	0	5
13	11	0	5
14–62 (even)	10	$((\text{slot} / 2) - 5)$	4
15–63 (odd)	11	$((\text{slot} - 1) / 2) - 5)$	4

每一個 distance 都會用 6-bit distance slot 作為起始，6-bit distance slot 存的數字就是後面會再接多少 bit 的意思。

LZMA 在壓縮的時候要從 dictionary 快速地尋找 match，會使用 Hash chain 的方式記錄 dictionary (window)的內容。

三、lzma、jpeg 與先壓縮成 jpeg 再放到 lzma 壓縮 Lena.raw 的比較

(一)、方法

Jpeg 壓縮直接用 irfanview 把 Lena.raw 存檔成 jpeg 檔案並且把畫質調到 100(最高)，png 壓縮也是用 irfanview，而 lzma 壓縮採用的是 python 預設的套件"lzma"，.7z 檔案是用 7z file manager 壓縮(字典大小為 64MB)，最後再將三者的檔案大小計算出來。

(二)、數據比較

原始檔案大小(.RAW): 262144 bytes

壓縮後檔案大小

(1) jpeg: 154679 bytes (有損)

(2) png: 224248 bytes

(3) lzma: 180133 bytes (無損)

(4) 7z: 181034 bytes

(4) jpeg 再做 lzma 並存成.lzma 格式: 156012 bytes

(5) jpeg 再做 lzma 並存成.7z 格式: 154838 bytes

(5) png 再做 lzma 並存成.lzma 格式: 223698 bytes

(6) png 再做 lzma 並存成.7z 格式: 223905 bytes

(三)、結果

得到的結果是 jpeg 壓縮完之後再用 lzma 壓縮反而讓檔案越壓縮越大，而 png 壓縮完之後再用 lzma 壓縮則能降低少許檔案大小。

而使用官方提供的解壓縮 cpp 檔案，發現 python 壓縮的預設設定是，dictionary size 一律是 $8388608=2^{23}$ bytes，並且參數設定為 lc=3、lp=0、pb=2，試著調整 dictionary size 到 1GB=1000000000 Bytes，壓縮速度變慢了，但是壓縮出來的檔案還是一樣大。

而存成.7z 格式後檔案變大應該是因為標頭檔變大而導致壓縮完後的檔案變大了。

```
選擇 命令提示字元
LZMA Reference Decoder 15.00 : Igor Pavlov : Public domain : 2015-04-16
In header[0]: 93
In header[1]: 0
In header[2]: 0
In header[3]: 0
In header[4]: 96
In header[5]: 255
In header[6]: 255
In header[7]: 255
In header[8]: 255
In header[9]: 255
In header[10]: 255
In header[11]: 255
In header[12]: 255
lc lp dp is encode as: 93

lc=3, lp=0, pb=2
Dictionary Size in properties = 1610612736
Dictionary Size for decoding = 1610612736
End marker is expected

Read      : 180133 bytes
Written   : 262144 bytes
Finished with end marker

C:\Users\WMLAB>
```

上圖: dictionary size 調整成 1.5GB 壓縮出的結果依然一樣

Code:

用 python lzma 套件壓縮圖片，用 cpp 解壓縮並觀察 header 內容

https://github.com/Louislar/DataCompression_finalReport

參考:

[1]: <https://dev.twsiyuan.com/2018/06/how-to-compress-and-decompress-gamesaves-in-unity.html>

[2]:
<http://html.rhhz.net/BJHKHTDXXBZRB/20150302.htm#R-4>

[3]: <https://gautiersblog.blogspot.com/2016/08/lzma-compression.html>

[4]: <https://www.slideshare.net/LevanChelidze/7-zip-compression-settings-guide>

[5]:
https://people.cs.nctu.edu.tw/~cjtsai/courses/imc/classnotes/imc14_05_Dictionary_Codes.pdf