

探討無損數據壓縮

篇名：

探討無損數據壓縮

作者：

陳立中。國立基隆高中。214 班

蘇冠綸。國立基隆高中。206 班

張逸。國立基隆高中。214 班

探討無損數據壓縮

壹●前言

在細談這篇論文的目的之前，我們要先了解甚麼是「無損數據壓縮」。在電腦上，時常有一些程式可以將數據壓縮，使他們的大小變小。數據壓縮主要分成兩種——無損數據壓縮 (*Lossless Data Compression*) 與有損數據壓縮 (*Lossy Data Compression*)。其中，無損數據壓縮指的是當數據被壓縮並解壓縮後，與原來的數據是一樣的，而反之，有損數據壓縮指的是被壓縮並解壓縮後的數據與原來的有所不同，因有部份於壓縮過程中損失，或被扭曲以便壓縮。

俗說：「天下沒有白吃的午餐」。所以，我常思索著，原來需要這麼多儲存空間的數據，怎麼可能不付出甚麼，就把這些數據變小，其中想必有些奇妙的數學。因次，借此次的機會，我嘗試找出無損數據壓縮是如何做到的。

貳●正文

一、無損數據壓縮的演算法

有許多種演算法（方法）可以壓縮數據，而在本篇論文中將介紹幾個較為知名的壓縮演算法。

1. 字典式壓縮法

其中一種最常用的無損數據壓縮演算法就是字典式的壓縮法。字典是壓縮法的運作原理，就是找出數據中重複出現的資料，然後以較短的數據來代替他。

A. 游程編碼

游程編碼指的是將數據中連續且重複的資料用比較短的方式表達，例如說：（註一）

以下有一段數據：

```
0.999999999998
```

若用游程編碼，則以上數據可以用這個方式表達：

```
0.['9'x11]8
```

由於在原始數據中，「9」這個字重複了十一遍。所以，再游程編碼時，所有的十一個「9」都用 ['9'x11] 取代。游程編碼只是一種方法，所以每一個程式所使用的語法不一定相同。

游程編碼的優點是如果數據的性質符合，則游程編碼的壓縮比例可以很高。在數學上面，游程編碼這個演算法屬於 $O(n)$ 種類的演算法，因為運算所需要的時間與運算的原始數據大小成正比。所以用這種方法壓縮大且含有大量連續且重複的資料可以達到不錯的效率。

游程編碼的缺點是，很少有數據是含有大量重複且連續的資料。現實生活上，唯一有這種性

探討無損數據壓縮

質的數據就是黑白圖片，因為黑白圖片上的每一格都可以用 0 或 1 來代表，而且，一般圖片上，常有許多重複且連續的黑或白格子。

B. LZ77/LZ78

LZ77/LZ78 是一種稍微比遊程編碼應用更廣泛的演算法。LZ77/LZ78 中，需要有一個「字典」，這個「字典」裡，有一些數據中常出現的資料，並給每一筆資料一個代碼。（註二）則在其餘的數據中，將在字典裡出現的資料替換成那資料的代碼，以結省空間。例如說：以下有一原始數據：

```
I am doing this thesis on a train, because it is fun to do thesis
on a train, and a train likes to have a thesis done onboard it.
```

很明顯的，在以上數據中，重複出現的數據（字）有：'a train', 'thesis'

所以，字典裡的數據有：

```
1=a train
2=thesis
```

而其餘的數據是：

```
I am doing this $2 on $1, because it is fun to do $2 on $1, and $1
likes to have a $2 done onboard it.
```

值得注意的是壓縮後的數據也包含字典。

LZ77/LZ78 是目前最常用的無損數據壓縮演算法。市面上常見的 .zip 檔中所用的 DEFLATE 壓縮法就是使用 LZ77，還有 .gz 檔也是使用 LZ77，另外，壓縮率很强的 7-zip，也是使用一種 LZ77/LZ78 的變種，叫做 LZMA。

LZ77/LZ78 的優點就是可以壓縮很多種資料，可以被 LZ77/LZ78 壓縮的資料也很多，例如說最常見的文字檔，網頁，等。LZ77/LZ78 的程式也很好寫，一版編程的初學者大多都能寫的出來。

而 LZ77/LZ78 的缺點就是，它所需要的計算時間較長。數學上，LZ77/LZ78 屬於 $O(n^2)$ 的演算法，也就是說，壓縮所需要的時間與原始數據的大小平方成正比。如果，程式寫的很好，LZ77/LZ78 的效率可以進步成 $O(n \log n)$ ，也就是說，壓縮所需要的時間與原始數據大小乘以原始數據大小的對數成正比。

2. 其他壓縮演算法

A. B-W 轉換（Burrows-Wheeler Transform）

嚴格來說，B-W 轉換並不算是一種無損壓縮演算法。（註三）但是，由於 B-W 轉換可以將要被壓縮的數據中的字元的位置調換，使之後的壓縮效率更好。

B-W 轉換可以分成幾個步驟：（以下以 'papaya' 為例）

探討無損數據壓縮

a. 列出原始數據的所有旋轉轉變

旋轉轉變指的是將第一個字移到最後一個，重複直到回覆到原本的數據。

例：

```
papaya
apayap
payapa
ayapar
yapapa
apapay
```

b. 將第一列依字母順序排列

如果遇到相同的，就比較第二列，以此類推。

```
papaya
apayap
payapa
ayapar
yapapa
apapay
```



```
apapay
apayap
ayapar
papaya
payapa
yapapa
```

c. 取最後一列為結果

```
apapay
apayap
ayapar
papaya
payapa
yapapa
```

=> 結果為：'yppaaa'

從以上例子可以看見 *B-W* 轉換可以增加同樣的數據出現的次數，原本的 *p* 與 *a* 現在都放在一起了，所以壓縮起來效率會很好。

探討無損數據壓縮

B. MTF轉換 (Move-to-front Transform)

基本上來說，MTF轉換也不是一個無損數據壓縮演算法。向B-W轉換一樣，MTF轉換也是能將數據轉變，讓他們更容易壓縮。

在做MTF轉換時，有一個數組，它的大小等於所有可能字元的數量，在一般電腦上，一個位元組有八個位元，所以數組的大小 $=2^8=256$ 。一開始，這數組的每一個數字等於他在數組的位置 (0, 1, 2, 3, 4, 5... n)。之後，從數據的第一個元字開始，另那元字為 x，另 n 為 x 在數組的位置，然後，以 n 取代那元字，最後，在將數組中，將 x 一到最前面，重複直到最後一個元字。

例：(以 '31415' 為例)

數組狀態：0123456789

處理第一個元字：**3**1415 (數組狀態：**3**012456789)

處理第二個元字：3**2**415 (數組狀態：**2**301456789)

處理第二個元字：32**4**15 (數組狀態：**4**230156789)

處理第二個元字：324**3**5 (數組狀態：**1**230456789)

處理第二個元字：3243**5** (數組狀態：**1**230456789)

C. 哈夫曼樹狀編碼

哈夫曼樹狀編碼是利用元字在數據中出現的頻率來排列出一個樹狀資料結構，在利用這個樹狀資料結構來排出一個非等長編碼方式。(註四)例如說，如果原始數據是：
'ABABACAD'

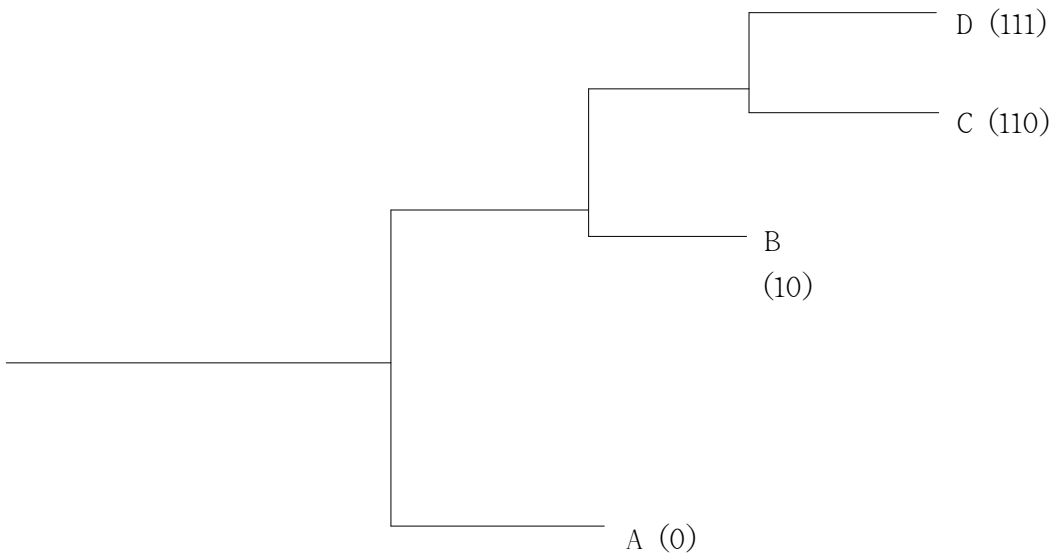
由於原始數據中，只有四種元字，所以每一元字為兩個位元，以上範例總共有十六個位元，也就是兩個位元組。

在原始數據中，元字出現的頻率如下：

元字	頻率
A	0.5
B	0.25
C	0.125
D	0.125

則所產生的樹狀資料結構為：

探討無損數據壓縮



因此編碼的方式為：

字元	碼
A	0
B	10
C	110
D	111

最後，將原始數據重新編碼：

ABABACAD =>

A	B	A	B	A	C	A	D
0	10	0	10	0	110	0	111

=> 01001001100111 (十四個位元)

結果比原始數據少了兩個位元。

參●結論

無損數據壓縮並不是向我們想的一樣，甚麼都不做，就可以減少數據的大小。而他是花了許多計算時間，還有許多人的智慧結晶，所換來的。

當然，從以上許多的無損數據壓縮中可以看出，不同的數據，需要不同的壓縮方法，才有辦法達到最佳的壓縮效率。所以，以後在壓縮數據前，一定要先考慮數據的種類，性質，在選擇最恰當的壓縮公式與軟體，而不是甚麼都用.zip 檔解決。

肆●引註

探討無損數據壓縮

註一、David Salomon。 *Data Compression — The Complete Reference*。 (United States of America : Springer , 2004) 20~21 頁

註二、English Wikipedia — Article on LZ77 and LZ78 。 <http://en.wikipedia.org/wiki/LZ77> 。 (檢索日期：民國九十七年三月二日)

註三、English Wikipedia — Article on Burrows-Wheeler Transform 。 http://en.wikipedia.org/wiki/Burrows-Wheeler_transform 。 (檢索日期：民國九十七年三月二日)

註四、The Squeeze Page 。 <http://www.cs.sfu.ca/cs/CC/365/li/squeeze/> 。 (檢索日期：民國九十七年三月二日)