壓出一片天空——
Arithmetic壓縮演算法實作

國中組應用科學科第三名

台北市立北政國民中學

作　　者：唐宗漢
指導教師：林淑娟

一、研究動機

Arithmetic乃由P.Elias所提出的壓縮理論，雖然其「理論壓縮率」頗高，但因所需之記憶體過多、速度太慢（因需大量計算），故難在早期電腦上實現，近期個人電腦速度大爲提高，且記憶體尚稱足夠，但ZIP壓縮法和其他壓縮法（如LZW，LZSS，ARJ等）已提供不錯的壓縮率和多樣的功能，故鮮有Arithmetic壓縮法之踪跡……

偶在一書上見到此壓縮之理論，驚於其高壓縮之率與特殊的壓縮方法，故興「實作一番」之念，以解其「冰封」之憾……

二、研究器材

IBM 80486 SX－33，DOS version 6.00，Quick Basic version 4.5(E)。

三、演算原理與評價

Arithmetic演算法之精神，不同於市面上「字典式」或「代換式」演算法；其精神在以一有理數區間，表示一串資料碼，舉一例云：

若吾人欲壓縮一字串“AABA”，則：

1.處理頻率：此例中A與B的頻率分別爲3/4和1/4。

2.處理區間：

第一區間：［0，1］代表未壓縮時之資料可能區間，在程式中以Aru和Ard來代表。

處理程式是一個迴圈：

FOR Loop Count＝1 TO LEN(Subject Code$)＜－－此例中爲4Aru＝Aru－Dstack(ASC (MID$(Subject Code$,Lm,1))) ＊ (Aru－Ard)/LEN (Subject Code$)

```
    Ard = Ard + UStack(ASC(MID$(Subject Code$,Lm,1))) * (Aru - Ard)/
    LEN(Subject Code$)
NEXT
```

（Ustack和Dstack分別是兩個陣列，用以儲存每個符號之前、之後其他符號的總出現數）

這裡是在將［Aru、Ard］［打］到這個符號的位置（此例中，遇A則取前區段的下3/4部份，B則是1/4）

Loop Count＝1時，處理的是“A”，故［0，1］取下方3/4，成爲新的區間：［0，3/4）or［0，0.75）or［0，0.11）＜－－二進制（注意，這個區段用來表示A。）

Loop Count＝2時，將［0，3/4）取下方3/4，得到

［0，9/16）or［0，05625）or［0，1001）＜－二進制（這個區段用來表示字串“AA”。）

Loop Count＝3時，處理的是B；因此改取頂端1/4部份，而有

［27/64，9/16）or［0011011，0，1011）＜－－二進制（這時是代表“AAB”）

Loop Count＝4時，最後還是一個“A”；亦取底部3/4，得到：

［27/64，135/256）or［0.421875，0.52734375）or［0.11011，0.1000111）（二進制）

這便是代表“AABA”的區間！那我們當然不須要將整個區間存入，只要存入區間中任意一數，要解壓時再Build Table就好了！容易看出，介於最終的區間［0.11011，0.100111］中最短（即要花最少空間表示）的數自然是0.1（二進位）；‘0’可以不計，於是四個Byte（即32Bit）被壓成一個Bit（此例壓縮率爲1/32，即3％）！而目前的Huffman和L－Z壓縮法決計不可能有此成積……

Arithmetic壓縮法的「整體壓縮率」確實非常高，不但其壓縮率距離整個資料的平均Entropy（資訊熵）非常近，且差「最差壓縮率」（即資料在「最不理想」狀況想得到的壓縮率）在我的壓縮程序中（當然没有檔頭）絕不會比100％差（這又代表什麼？這個的意思是：如果不考慮頻率資訊與長度資訊（兩個最長的檔頭），任何檔都可壓成6～24Bytes！！！），正好說明了每項資料的Entropy都在1以下（亦即所有資料都是「可壓的」）之道理……

但是，Arithmetic壓縮法真的是完美無缺的嗎？非也，缺點在於：

1.此壓縮法速度太慢；因其壓縮過程中需要大量計算，所以很耗時間，不過，好的「策略」（如好的資料結構、字串安排方式、搜尋法等）與愈來愈快的電腦應可以消除這個障礙。

2.此壓縮法與「資料之間」的關係完全無關；故它要多存一個「頻率表」，以表示各種Symbol出現的次數……

3.此壓縮法需要極高的「準度」；若是以1K作「單位壓縮區間」，則它必需有準到$1/10^{768}$左右的準度，而這個起碼要占去4K的「變數」來裝它，而這……似乎就QB來說極爲困難。

雖然有以上三缺點，但以目前的電腦來看，突破此三項障礙（尤其是第一、三項）並非難事，因此它的價值仍應受到肯定。

四、研究過程

此程式自6月便開始寫了一點點，但因是「理論性質」（亦即，壓出Code就完事，故無實用性（無法解碼），所以自93年11月起重寫……

遭遇的第一個問題：以什麼爲單位壓縮？最先想到的自然是以Byte來作，但QB本身只支到Double(2^{16})，$256=2^8$，每次只能壓二個Bytes來，全無實用性可言；也想過作一「大數器」（Big Num）以算到2^{64}，但又有「運算記體」大限，動不動就Overflow，只得放棄，而改以4 bits作爲單位；因4 bits只有16(2^4)種變化只有最多12種；東湊西補，也就可以用到6 byte當作一個「處理段」。

再來就遇到了麻煩：頻率表怎麼存？我換過四種方式，最後決定每次壓的時候就先試兩種存法的頻率，再加以比較並選出最好的；但仍舊佔用極大空間。（平均約3.2 bytes）

然後是致命的問題：壓縮碼的處理時間過久！在當時，6 bytes要16秒才能壓完……如此一來，一個普通的文件要20分鐘，一個中型要執行檔更要二個小時才能壓得完！我被這個問題搞得焦頭爛額，最後將二進位的轉換全部，成Function，並刪掉了許多不必要的處理函式（例如，計算Entropy、比較每個符號出現次數並與上次比較，以省空間），才有現在的6 bytes對3.4 second……（這個速度實在也慢的可以）

全部寫完之後，我才發現在檔案中，壓縮過的碼緊接在頻率表後面，根本無法讀取！所以又用連續的“1”，來表示其長度，及「其長度」的長度……

這樣子大費周章之後，得到的壓縮率爲90％，雖然不算好，也總算是知道了「資料壓縮」的實際方法，與Arithmetic的實際應用情形……

五、研究討論

這一次的程式之所以只有90％的Ratio，其最大的問題在於我只用6 bytes當一個「段」，故絕不可能有超過80％的平均壓縮率。

不過，若是使用C.C++，或其至組合語言，應可以作到超過10分之一的準度，屆時，依我計算可作到平均（設以1K爲單位）34％的Ratio（壓縮碼爲150 Bytes，頻率碼10＊16＝160 Byte，長度碼＝Log2 150（約7.2）＊4（2＊2）－＞30 Bytes，加起來340 Bytes……），所以我此次只是一個開始。

我希望未來能夠突破「以6 Bytes爲一段」的限制，則可以以比目前更好的速度，達到比現有市面上的壓縮工具軟體更好的效率（以純壓縮碼來說，測試十個檔案時，都比ARJ好至少八個百分點）。

六、參考文獻

活用Quick BASIC技巧篇　松崗　吳明哲、黃世陽

Microsoft Quick BASIC程式設計師的工具　松崗　John C. Craig著

資料壓縮實務技術　波心　沈文智著

評　語

1.本作品嘗試以QBASIC語言撰寫一壓縮軟體程式，展現理論與實作的比較，以國中一年級的年齡，誠屬難得。

2.作者對原理的掌握極爲清晰，表達能力強，示範清楚。

3.本作品可後續發展，具實用性的潛在價值。