數值方法導論與應用期末專題-檔案壓縮方法介紹

學號：B093040003 姓名：鄭璟翰

一、前言

先前我從作業三的內容來獲得的靈感，在 Final Project Preview 時，我使用「傅立葉轉換應用-檔案編碼&壓縮」做為介紹，但是在進行該題目時，發現了一些狀況(稍後會介紹到)，所以我選擇將原本的題目和老師在 Preview 提到「圖片檔案壓縮也有其他種做法」結合，改進行「檔案壓縮方法介紹」這個題目。檔案壓縮是一種將檔案或資料夾的大小進行減少的技術，通常用於節省存儲空間，進一步加快檔案傳輸速度。

二、檔案壓縮原理

檔案壓縮主要是利用資料的「一致性」進行處理，如果資料越一致，那就可以刪除重複的數據。首先尋找並刪除檔案中的重複數據，因為在許多檔案中，存在著大量的重複數據。相同的數據塊可以在一個檔案中出現多次或在多個檔案中重複出現。檔案壓縮算法通過將重複的數據塊替換為參照它們的指標或使用其他技術來減少數據的冗餘性。除了可以利用資料的「一致性」外，「資料編碼」也是檔案壓縮另一個常見的重要原理，可以利用資料編碼的技術來進一步減少數據的大小。資料編碼是一種將輸入數據轉換為更緊湊形式的過程。它利用統計、數學和字典等技術來將原始數據轉換為更短的編碼或符號序列。這些編碼可以使用較少的位元數來表示原始數據，從而達到壓縮的效果。

進行不同類型的檔案壓縮也有許多不同的技巧，如進行圖片檔案的壓縮，可以考慮圖片「空間上的一致性」、「頻率上的一致性」。「空間上的一致性」考慮相鄰的數值會比較相近，可以合併儲存；「頻率上的一致性」是表示圖片上的頻率代表的是變化，而一張圖片的頻譜會集中在低頻的部分，因為圖片較少有劇烈的變化。考慮「頻率上的一致性」可以使用作業三的 Low Pass Filter(LPS)來達成保留圖片低頻率的部分；但如果想要對文字檔案依照離散傅立葉轉換進

行運算的話，因為文字檔案的特性和圖片檔案的特性並不相同，即使轉換至頻率上，如果沒有找到文字檔案的規律，可能也沒有辦法進行有效的壓縮。因此，在考慮檔案壓縮時除了兩個基本的原理(刪除重複數據、資料編碼的技術)之外，針對檔案尋找

三、檔案壓縮方法介紹

檔案壓縮的方法依照結果可以分為失真壓縮（Lossy Compression）和無失真壓縮（Lossless Compression）是兩種。無失真壓縮是指壓縮檔案時能夠完整保留所有資料，表示著壓縮後的資料可以完全還原為原始資料，且壓縮前後的數據是完全相同的。無失真壓縮常用於對需要精確保留的數據進行壓縮，如文本檔案、程序代碼、無損音頻等。常見的無失真壓縮算法有 Huffman Coding、DEFLATE（用於 ZIP 壓縮）等；失真壓縮則是只壓縮檔案時可以更大程度地減少檔案大小，但在這個過程中會刪除一些相較不重要的原始資料，因此壓縮後的資料不能完全還原為原始資料，會有一定程度的損失。失真壓縮通常應用於對數據中的某些細節不敏感或可以容忍一定程度損失的情況下，以獲得更高的壓縮比。典型的應用包括圖像壓縮和音頻壓縮。常見的失真壓縮算法有：JPEG（Joint Photographic Experts Group）、MP3（MPEG Audio Layer III）、AAC（Advanced Audio Coding）、MPEG（Moving Picture Experts Group）

這次我選擇了三種壓縮(或編碼)方法進行介紹，第一種是使用原本的題目離散傅立葉轉換(Discrete Fourier Transform, DFT)來進行；第二種則是採用霍夫曼編碼(Huffman Code)來進行檔案壓縮；第三種則是透過 8*8 離散餘弦轉換(discrete cosine transform, DCT)來進行檔案壓縮，這也是 JEPG 的壓縮方法。

(一)、離散傅立葉轉換(DFT)

第一個部分是針對我原先預想的題目「傅立葉轉換應用-檔案編碼&壓縮」進行。首先，因為文字檔案當中是離散的數據點，因此需要使用到離散傅立葉轉換。

連續傅立葉轉換：

$$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-i\omega t}dt$$

離散傅立葉轉換：

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-ik\omega_0 n}, \omega_0 = \frac{2\pi}{N}$$

得到離散傅立葉轉換的形式後，可以觀察頻譜的第 k 項是由輸入 n 項和$e^{-ik\omega_0 n}$進行相乘所得到的，因此可以寫成以下矩陣的形式。

離散傅立葉矩陣：

$$\begin{array}{c c} & \begin{array}{cccc} n=0 & n=1 & \cdots & n=N-1 \end{array} \\ \begin{array}{c} k=0 \\ k=1 \\ \vdots \\ k=N-1 \end{array} & \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-i\omega_0} & \cdots & e^{-i2(N-1)\omega_0} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-i2(N-1)\omega_0} & \cdots & e^{-i(N-1)^2\omega_0} \end{bmatrix} \end{array}$$

但是因為可以看到其中有虛數 i 出現在指數次方上，這很難用計算機程式來表達這樣的運算，因此我用尤拉公式來對離散傅立葉矩陣進行轉換。

尤拉公式：

$$e^{ix} = \cos x + i \sin x$$

$$x = ik\omega_0 n, e^{-ik\omega_0 n} = \cos(-k\omega_0 n) + i \sin(-k\omega_0 n)$$

尤拉公式帶入離散傅立葉矩陣：

$$\begin{bmatrix} \cos(-0\frac{2\pi}{N}0) + i\sin(-0\frac{2\pi}{N}0) & \cdots & \cos(-0\frac{2\pi}{L}(N-1)) + i\sin(-0\frac{2\pi}{L}(N-1)) \\ \vdots & \ddots & \vdots \\ \cos(-(N-1)\frac{2\pi}{N}0) + i\sin(-(N-1)\frac{2\pi}{N}0) & \cdots & \cos(-(N-1)\frac{2\pi}{L}(N-1)) + i\sin(-(N-1)\frac{2\pi}{L}(N-1)) \end{bmatrix}$$

最後可得到一個由複數所組成矩陣，實數和虛數的部分就可以分開儲存。我把輸入得到的文字檔案以行作為單位，每一行作為輸入 x[k]乘上離散傅立葉矩陣(實際做法為將讀到的資料作為實部，虛部的部分先做為 0，得到結果去掉 i 後分開儲存)，就可以得到該行的頻率分布 X[k]。將每一行的頻率重新儲存於另一個檔案就完成編碼，而解碼的部分則將 X[n]乘上離散逆傅立葉矩陣，可以得到原本的 x[k]。

離散逆傅立葉轉換：

$$x[k]=\frac{1}{N}\sum_{n=0}^{N-1}X[n]e^{ik\omega_0 n},\omega_0=\frac{2\pi}{N}$$

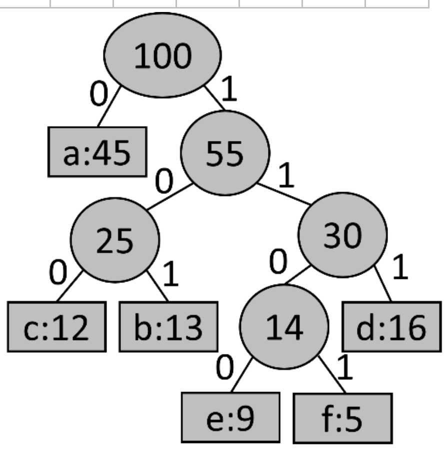
儘管這樣可以得到每一行的頻率分布，但是由於輸入的文字檔案頻率分布不一定有特殊意義，因此很難分析這些頻率。而且更糟糕的是因為原本的輸入因為只有實數的部分，轉換完的資料卻需要額外紀錄複數的部分，反而使得轉換後的檔案需要儲存更多內容，沒有辦法有效地進行壓縮，因此這個方法可能只能視作一種編碼，而沒有辦法進行壓縮。

(二)、霍夫曼編碼(Huffman Code)

Huffman Code 是一種常見的無失真壓縮算法，透過統計資料中的特徵來實現壓縮。Huffman Code 的核心做法是將出現頻率高的符號用較短的編碼表示，而出現頻率低的符號用較長的編碼表示。解壓縮時，使用相同的霍夫曼樹和編碼表，將壓縮後的編碼進行解碼，恢復原始資料。Huffman Code 的優勢在於對出現頻率高的符號進行較短的編碼表示，有效地減少了整體編碼的長度，實現了壓縮。它在無失真壓縮中廣泛應用於文字文檔、程式碼等領域。但是因為需要額外儲存編碼表，Huffman Code 的壓縮效果就受限於字母頻率的統計特性，而無法適用於圖片類型的檔案。

以下是關於 Huffman Code 的範例與虛擬碼。

	a	b	c	d	e	f
Freq	45	13	12	16	9	5




```
HUFFMAN(C)
  n = |C|
  Q = C
  for i = 1 to n-1
    allocate a new node z
    z.left  = x = EXTRACT-MIN(Q)
    z.right = y = EXTRACT-MIN(Q)
    z.freq = x.freq + y.freq
    INSERT(Q,z)
  return EXTRACT-MIN(Q)
//return the root of the tree
```

圖表一、Huffman Code 範例與虛擬碼

(三)、8*8 離散餘弦轉換(DCT)

8*8 離散餘弦轉換(DCT)是一種在影像壓縮中常用的轉換技術，特別在依照區塊進行壓縮方法（如 JPEG）中得到廣泛應用。DCT 將一個 8*8 的輸入分解為一組表示該區塊中不同頻率的頻譜，包括低頻部分和高頻的部份。分成 8*8 原因：1.因為藉由分割圖像可以進一步減少圖片當中高頻的區域，稍後進行壓縮時，可以減少被刪去的資訊；2.減少運算的量級和節省記憶體空間；3.可以進行平行化的運算，加快運算時間。

離散餘弦轉換：

$$X[k] = \sum_{n=0}^{N-1} x[n] \, cos \left[\frac{\pi}{n} m \left(k + \frac{1}{2} \right) \right]$$

離散逆餘弦轉換：

$$x[n] = \frac{1}{2} X_0 + \sum_{k=0}^{N-1} X[k] \, cos \left[\frac{\pi}{n} m \left(k + \frac{1}{2} \right) k \right]$$

可以觀察到和傅立葉轉換相比，餘弦轉換並沒有複數的部分，所以只需要考慮實數的部份。

使用 DCT 進行壓縮時的步驟：1.將 8×8 的圖像區塊分割為 64 個像素值； 2.將區塊的中心化（減去平均均值），以便更好地表示頻率的分布；3.為了進一步壓縮圖像，可以對這些 DCT 係數進行量化(將係數近似為一組有限範圍內的值的過程，以減少所需的位元數量)；4.將這些量化後的係數進行重新編碼，以進一步減少表示這些係數所需的位元數量。

而解壓縮時，可以反向操作壓縮的步驟來恢復原始圖像區塊：1.進行反向編碼和反量化以恢復量化係數；2.進行反 DCT 轉換，將這些係數轉換回原本的數值；3.進行反中心化（加上平均值）以恢復原始的 8*8 圖片區塊。

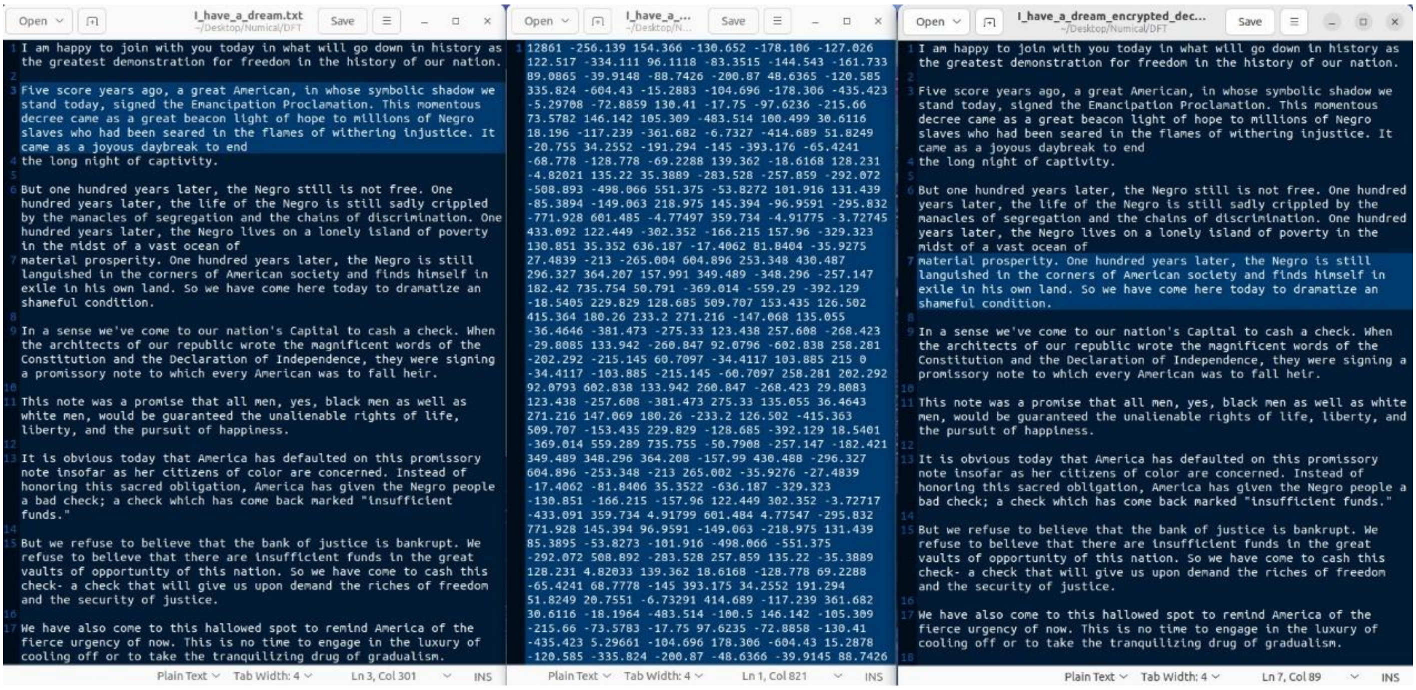
透過 DCT，影像壓縮可以通過保留重要的低頻成分和減少不重要的高頻成分來達到壓縮的效果，同時還能保留圖像的主要特徵和結構。這使得 DCT 成為廣泛應用於影像壓縮標準中的重要技術之一。

三、暫時結果

以下分別為三種編碼方式的暫時測試結果，還沒有經過整理，會在期末上台報告時重新整理呈現方式。

(一)、離散傅立葉轉換(DFT)

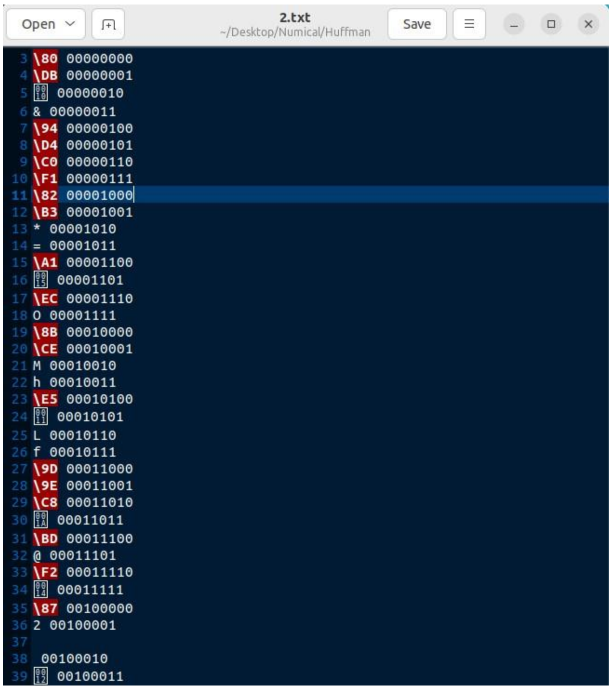
三個視窗分別是加密前、加密後、解密後的檔案，可以看到第一行的資料被轉換成頻率後的資料量大增，並沒有辦法做到壓縮的效果。



圖二、離散傅立葉轉換測試結果

(二)、霍夫曼編碼(Huffman Code)

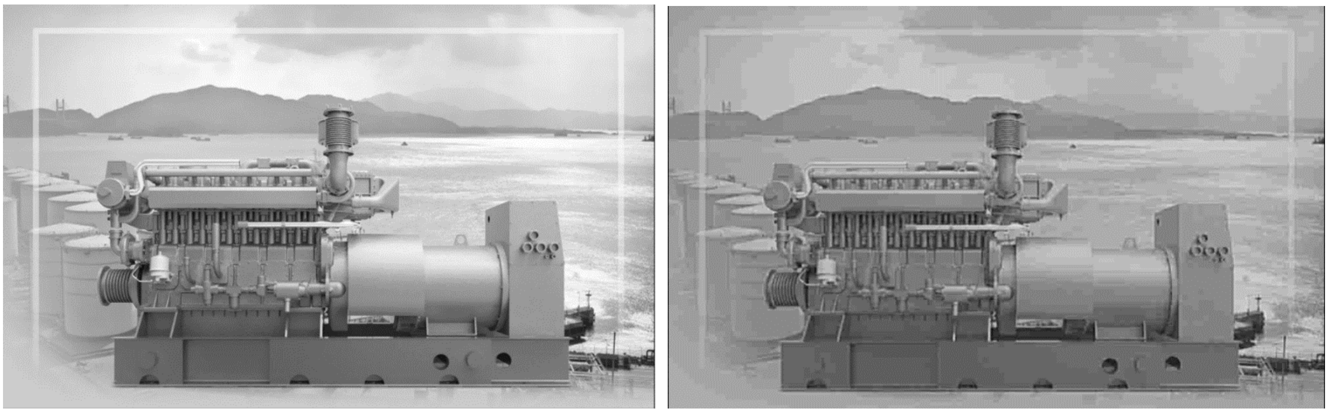
右圖三為某一圖片檔案經過 Huffman Code 編碼之後的所產生的 Huffman Table，左側(許多紅底)標起來的文字是原本的符號，而右側數字則是編碼之後的結果。可以看到大部分的字符的編碼方式都還是八個位元，因為圖片當中各自元分布的頻率相較平均，因此難以呈現其壓縮效果。



圖三、Huffman Code Table

(三)、8*8 離散餘弦轉換(DCT)

下圖四左側的圖片是原先的輸入，右側的圖片則是經過壓縮後的結果，可以看到有測的圖片僅有稍微失去一些明亮程度，大致上仍後左側相同。



圖四、DCT 壓縮前、壓縮後結果

四、參考資料

[1] 數值方法應用與導論課堂投影片 Week07_Fourier Analysis.pdf p29

[2] Wikipedia: Discrete cosine transform https://en.wikipedia.org/wiki/Discrete_cosine_transform

[3] Wikipedia:影像處理 https://zh.wikipedia.org/zh-tw/%E5%9B%BE%E5%83%8F%E5%A4%84%E7%90%86

[4] Kaggle: Image Processing: Image Compression using DCT https://www.kaggle.com/code/siddheshmahajan/image-processing-image-compression-using-dct

[5] Introduction to Algorithm p429~432