1. 107522050 熊偉年

|  |  |  |
| --- | --- | --- |
|  | compressed file sizes | the first-order entropy |
| Baboon.raw | 242,230 bytes | 7.3577344969276695 |
| Lena.raw | 245,152 bytes | 7.4473588478428665 |

1. DPCM

|  |  |  |
| --- | --- | --- |
|  | compressed file sizes | the first-order entropy |
| Baboon.raw | 209,360 bytes | 6.353183994011977 |
| Lena.raw | 167,422 bytes | 5.0653866464464485 |

1. 檔案說明

|  |  |
| --- | --- |
| main.js | Adaptive Huffman coding – FGK |
| maindecode.js | 幫main.js壓縮的檔案decode |
| mainDPCM.js | DPCM 的 Adaptive Huffman coding |
| maindecodeDPCM.js | 幫mainDPCM.js壓縮的檔案decode |
| compare.js | 比較兩個檔案內容是否一樣 |
| calculate.js | 計算entropy |

程式解釋

首先一般的Adaptive Huffman coding – FGK 我的流程大概是這樣

1: **begin**

2: create node *ZERO*

3: **readSymbol**(**X**)

4: **while** (**X**!=*EOF*) **do**

5: **begin**

6: **if** (first\_read\_of(**X**)) **then**

7: **begin**

8: **output**(*ZERO*)

9: **output**(**X**)

10: create new node **U** with next nodes *ZERO* and new node **X**

11: update\_tree(**U**);

12: **end**

13: **else**

14: **begin**

15: **output**(**X**)

16: update\_tree(**X**)

17: **end**

18: **readSymbol**(**X**)

19: **end**

20: **end**

21:

22: **procedure** update\_tree(**U**)

23: **begin**

24: **while** (**U**!=*root*) **do**

25: **begin**

26: **if** (exists node **U1** with same value and greater order) **then**

27: change **U1** and **U**

28: increment value of **U**

29: **U** := parent(**U**)

30: **end**

31: increment value of **U**, update leaf codes

32: **end**

這是參考 <http://www.stringology.org/DataCompression/fgk/index_en.html>

網站寫的.

只是我寫成JS 的版本

JS的讀檔(fs.readFileSync)會有 加入一些function 跟 offset

所以在壓縮的時候必須要避開他

然後update再找 order比他大但是value一樣的時候要直接找order最大的。

然後再換的時候如果發現是自己的root 就不換。

然後記得最後不滿八個編碼要flush 我把它塞滿1 湊到8個bit

這有可能造成小bug 就是 剛好在讀這些塞滿的bit的時候會造成不小心讀到symbol. 不過這次的兩份sample 都沒出現此問題，故沒有特別做處理。

最後DPCM的處理比較特別一點，照理來說要儲存負的數字，因為一個value有可能0~255，所以差值的範圍可以到-255~255，照理來說需要9個bit才能儲存，但是稍作觀察發現我只要繼續照樣存8個bit，舉例-130的表示法應該是101111110，但是我只存01111110到buffer中。到時候讀資料的時候直接照樣的加上去，但是要忽略最後第九個carry bit，這樣在decode的時候算出來的值會剛好一樣，這樣就只要8個bit照樣可以處理-255~255的範圍。

我再舉個例子 假設今天的值是

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 255 | 255 | 0 |

那DPCM的差值算出來是 (假設初始值是0)

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 255 | 0 | -255(100000001) |

那我壓縮後儲存到buffer的值是

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 255 | 0 | 1(00000001) |

那在讀取(解壓縮)的時候

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 255 | 255 | 0 (因為11111111加上1之後忽略第九個carry bit) |

如何執行

我的程式讀檔都必須要，更動程式碼才可以讀取不同的檔案，這點請注意一下謝謝！

```

node main.js

```

其他檔案也是前面加個 node 加程式名字 即可執行。

電腦裝需要 node.js