

Assignment #3

Observation1: txt -> encode(編碼) -> decode(解碼)過程分析

Huffman code 的編解碼結果會在 terminal 出現，而執行時間只在 terminal 出現。而 Arithmetic code 的編解碼結果、程式運行所花費的時間，都會出現在 encode、decode 的檔案中！

原始檔案：

```
text2 huffman U x encode huffman M text huffman A text arcd text2 arcd
huffman > text2
1 Hello this is Ivvvvy speaking!!!!@^3^
```

編碼結果：

Arithmetic code

```
decode arcd encode arcd x decode huffman M encode huffman M decode (工作樹狀) M
arcd > encode
1 H%;s000
2 kD0i070QF00C500v0m0000F-0{0<
3 TIME COST IS: 0.000103 secs
4
```

Huffman code

```
text2 huffman U encode huffman M x text huffman A text arcd text2 arcd
huffman > encode
```

⚠

因為檔案為二進位檔或使用了不支援的文字編碼，所以未在編輯器中顯示。
繼續開啟

編碼結果：

Arithmetic code

```
decode arcd x encode arcd decode huffman M encode huffman M decode (工作樹狀) M
arcd > decode
1 Hello this is Ivvvvy speaking!!!!@^3^
2 TIME COST IS: 0.000128 secs
3
```

Arithmetic code 的 terminal 畫面

```
(base) ivy@ivy-VirtualBox:~/Desktop/109208001_assignment_3/arcd$ ./build/main.o -e <text2 | tee encode
kD0i070QF00C500v0m0000F-0{0<
TIME COST IS: 0.000103 secs
(base) ivy@ivy-VirtualBox:~/Desktop/109208001_assignment_3/arcd$ ./build/main.o -d <encode | tee decode
Hello this is Ivvvvy speaking!!!!@^3^
TIME COST IS: 0.000128 secs
```

Huffman code

```

text2 huffman U  encode huffman M  decode huffman M  text huffman A  text arcd  ▶ 🔍 📄 ...
huffman > decode
1 Hello this is Ivvvy speaking!!!@@@ ^3^
  
```

Observation2: 兩個壓縮演算法的效能比較

Arithmetic code

編解碼 花費時間 (sec)	encode	decode
短的句子測試一	0.000108	0.000152
短的句子測試二	0.000086	0.000105
平均	0.000097	0.0001285
長的句子測試一	0.000103	0.000128
長的句子測試二	0.000105	0.000119
平均	0.000104	0.0001235

Huffman code

編解碼 花費時間 (sec)	encode	decode
短的句子測試一	0.000270	0.000262
短的句子測試二	0.000274	0.000216
平均	0.000272	0.000239
長的句子測試一	0.000262	0.000229
長的句子測試二	0.000270	0.000224
平均	0.000266	0.0002265

1. 由數據可以看到：arithmetic code 的平均編碼時間小於解碼時間，而 huffman code 則是相反。
而從整體時間來看，arithmetic code 的編解碼效率高於 huffman code ！
2. 除此之外，我還發現，不管是哪種演算法，如果是初次編碼，程式都會編碼最久，等多次編碼後，編碼時間會趨向平均值。