Assignment #3

1. 編譯結果

```
(base) zhouchou@zhoudeMacBook-Air assignment_3-Mondayhs % sudo chmod +x build.sh
./build.sh

gcc -Wall ./src/arth.c ./src/bitstream.c ./src/huffman.c ./src/main.c -I ./inc -c
gcc ./obj/arth.o ./obj/bitstream.o ./obj/huffman.o ./obj/main.o -o ./bin/main
```

2. 執行結果

```
(base) zhouchou@zhoudeMacBook-Air bin % ./main
Erro!
 Please input follow :
 if u want encode use '-e', decode use '-d'.
./main -[d,e] [input file] [output Arithmetic file] [output Huffman file]
```

Error Run

```
(base) zhouchou@zhoudeMacBook-Air bin % ./main -e /Users/zhouchou/github-classroom/NCU-DSA-111-1/assi
gnment_3-Mondayhs/data/data10/test.txt /Users/zhouchou/github-classroom/NCU-DSA-111-1/assignment_3-Mo
ndayhs/data/data10/ec_arith.txt /Users/zhouchou/github-classroom/NCU-DSA-111-1/assignment_3-Mondayhs/
data/data10/ec_huff.txt
       start encoding by Arithmetic ...
       time cost: 0.000188  ms
       start encoding by Huffman ...
       time cost: 0.000039  ms
```

Test data: 10 words

```
(base) zhouchou@zhoudeMacBook-Air bin % ./main -e /Users/zhouchou/github-classroom/NCU-DSA-111-1/assi
gnment_3-Mondayhs/data/data100/data100.txt /Users/zhouchou/github-classroom/NCU-DSA-111-1/assignment_
3-Mondayhs/data/data100/ec_arith.txt /Users/zhouchou/github-classroom/NCU-DSA-111-1/assignment_3-Mond
ayhs/data/data100/ec_huff.txt
       start encoding by Arithmetic ...
       time cost: 0.000332  ms
       start encoding by Huffman ...
       time cost: 0.000259  ms
```

Test data: 100 words

```
(base) zhouchou@zhoudeMacBook-Air bin % ./main -e /Users/zhouchou/github-classroom/NCU-DSA-111-1/as
gnment_3-Mondayhs/data/data1000/data.txt /Users/zhouchou/github-classroom/NCU-DSA-111-1/assignment_
Mondayhs/data/data1000/ec_ari.txt /Users/zhouchou/github-classroom/NCU-DSA-111-1/assignment_3-Monda
s/data/data1000/ec_huff.txt
       start encoding by Arithmetic ...
       time cost: 0.003426  ms
       start encoding by Huffman ...
       time cost: 0.000870  ms
```

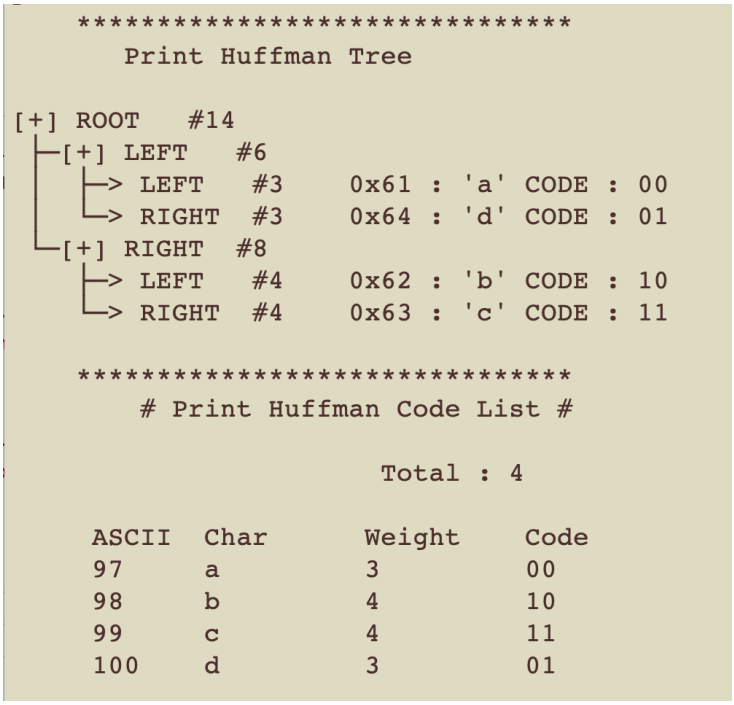
Test data: 1000 words

3. 原理

Huffman:

將欲壓縮之字串，先讀一遍，將字串中的每一相異單字元（Single Character）的出現頻率，做成統計，依此建構霍夫曼樹（Huffman's Tree）。每一相異單字元，用 0 與 1 予以編碼，出現次數逾多者，給予較少的位元編碼。霍夫曼編碼法的特點在於所編碼出來的檔案具有唯一碼性質的即時碼。也就是各個相異字元所編碼出所位元串並不相同，解碼時能立

即解出。



Arithmetic:

將一段 message 利用一個 0 到 1 間的區間來表示，當這個 message 越長，用來表示這個 message 的區間就越小，那要表示這個 message 的 bit 數就變多。相同的文字在 message 出現越多，區間變小的速度比較慢所以可以達到資料壓縮的效果。隨著符號序列長度之增加，其算數編碼長度之 entropy 趨近於無雜訊編碼理論之極限(效率愈高)。

4. 分析

   當字數變多 **Arithmetic** 耗時就越長，**Huffman 編碼時間較 Arithmetic 快很多。**

5. 未來

   將兩者解碼的部分補上（能夠一起呈現）。

6. 參考資料

   Huffman:

   https://www.foxzzz.com/Huffman-Code-Demo/

   Arithmetic:

   https://par.cse.nsysu.edu.tw/~homework/algo01/8934609/index.html