Assignment #4

110503517 通訊二 游晉陽

Ps.因個人電腦編譯器有點問題，故請同學代為執行

1. Arithmetic Coding 編譯＆執行結果

```
bingchen@LAPTOP-34JN212B:~/assignment_3-JacobTsai0107/arithmetic_coding$ ./a
rcd -e <text_1 | tee text_1_encode
C)
the coding time is 0.000154 sec.
```

```
bingchen@LAPTOP-34JN212B:~/assignment_3-JacobTsai0107/arithmetic_coding$ ./a
rcd -d <text_1_encode | tee text_1_decode
Classification of VPN based on the OSI model layers recognizes three types o
f VPNs: data link layer, network layer and application layer VPNs (Malik, 20
03). Algorithms used for encryption can be classified into partial encryptio
n, direct encryption and compression-combined encryption (Lian, 2009). Accor
ding to the number of keys used, algorithms can also be classified into asym
metrical and symmetrical algorithms.
the coding time is 0.000282 sec.
```

2. Huffman Coding 編譯＆執行結果

```
bingchen@LAPTOP-34JN212B:~/assignment_3-JacobTsai0107/huffman_coding$ cmake
.
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/bingchen/assignment_3-JacobTsai01
07/huffman_coding
bingchen@LAPTOP-34JN212B:~/assignment_3-JacobTsai0107/huffman_coding$ make
Scanning dependencies of target huffman
[ 25%] Building C object CMakeFiles/huffman.dir/huffman.c.o
[ 50%] Linking C static library libhuffman.a
[ 50%] Built target huffman
Scanning dependencies of target huffcode
[ 75%] Building C object CMakeFiles/huffcode.dir/huffcode.c.o
[100%] Linking C executable huffcode
[100%] Built target huffcode
```

```
bingchen@LAPTOP-34JN212B:~/assignment_3-JacobTsai0107/huffman_coding$ ./huff
code -i text_1 -o text_1_encode -c
the coding time is 0.000113 sec.
bingchen@LAPTOP-34JN212B:~/assignment_3-JacobTsai0107/huffman_coding$ ./huff
code -i text_1_encode -o text_1_decode -d
the coding time is 0.000044 sec.
```

從執行結果我們可以得到以下表格

	Huffman	Arithmetic
encode	0.000113 sec	0.000154 sec
decode	0.000044 sec	0.000282 sec

從上表可以發現，使用 Huffman Coding 對資料做 encode 和 decode 比使用 Arithmetic Coding 快上許多，尤其在 decode 時更為明顯（相差 6 倍多）