

## ▸ 程式解說

## ● compress()

```
1  struct node{
2      char c;
3      int value;
4      node *left, *right;
5
6      node(char c, int value, node *left, node *right) :
7          c(c), value(value), left(left), right(right)
8      {
9      }
10 };
11
12 struct cmp{
13     bool operator()(node const& a, node const& b){
14         return a.value > b.value;
15     }
16 };
```

```
1  void compress(){
2      ifstream INPUT_TXT;
3      ofstream OUTPUT_TXT, COMPRESS_BIN;
4      stringstream ss;
5
6      string input, output_code = "", output_compress = "";
7
8      map<char, int> cnt; cnt['\0'] = 1;
9      map<char, string> char_code;
10
11     priority_queue<node, vector<node>, cmp> pq;
12
13     // ...
14 }
```

一開始我們現宣告node資料型態與幾個變數

- node: 這是為了之後建立Huffman Encoding Tree所設立的型態，以及給priority queue所寫的compare function
- ifstream INPUT\_TXT: 用來讀取input.txt
- ofstream OUTPUT\_TXT: 用來寫入code.txt
- ofstream COMPRESS\_BIN: 用來寫入compress
- string input: 將來自input.txt讀取的資料內容存入
- string output\_code: 存入要寫入code.txt的所有內容
- string output\_compress: 存入要寫入compress的所有內容
- map cnt: 記錄每個字元所出現的次數
- map char\_code: 記錄每個字元的編碼
- priority\_queue pq: 建立tree時所需要的priority queue

```

1  void compress(){
2      // ...
3
4      INPUT_TXT.open("input.txt");
5      OUTPUT_TXT.open("code.txt");
6
7      ss << INPUT_TXT.rdbuf();
8      input = ss.str();
9      INPUT_TXT.close();
10
11     // count the number of each character
12     for(auto ch : input){
13         if(cnt.find(ch) == cnt.end()){ // if not found
14             cnt[ch] = 1;
15         }
16         else{
17             cnt[ch]++;
18         }
19     }
20
21     // push all the characters into the priority queue
22     for(auto data : cnt){
23         node tmp(data.first, data.second, nullptr, nullptr);
24         pq.push(tmp);
25     }
26
27     // build the huffman tree
28     while(pq.size() > 1){
29         node *left = new node(pq.top().c, pq.top().value, pq.top().left, pq.top().right);
30         pq.pop();
31
32         node *right = new node(pq.top().c, pq.top().value, pq.top().left, pq.top().right);
33         pq.pop();
34
35         pq.push(node{'\0', left->value + right->value, left, right});
36     }
37
38     // ...
39 }

```

- L4-L9: 我們先將檔案打開，首先將input.txt的內容讀入我們所設的stringstream裡，在丟給input，在把我們的code.txt先關起來
- L12-L19: 接著我們從input裡擷取一個個字元，並計算每個字元出現幾次
- L22-L25: 我們將map cnt裡的字元以及字元所對應的次數，以node的資料型態——丟入priority queue裏
- L28-L36: 將資料丟入priority queue後，開始建立Huffman tree，我們先拿出最上面的node，而這是我們的左節點，指向left，並將他pop，接著再拿出最上面的，而這是我們的右節點，指向right，並將他pop，最後我們將兩者次數加總，分別指向left right，最後在push進入priority queue，並直到我們的priority queue只剩一個node時，再進行編碼

接著，我們要進入將每個字元做編碼的環節

```

1 // build code
2 template <typename T>
3 void buildcode(T &char_code, node *root, string code){
4     if(root->left == nullptr && root->right == nullptr){
5         char_code[root->c] = code;
6         return;
7     }
8
9     buildcode(char_code, root->left, code + "0");
10    buildcode(char_code, root->right, code + "1");
11 }
12
13 void compress(){
14     // ...
15
16     // build the code
17     node *root = new node(pq.top().c, pq.top().value, pq.top().left, pq.top().right);
18     pq.pop();
19     buildcode(char_code, root, "");
20
21     // ...
22 }

```

- 我們開始建立Huffman tree，我們先宣告一個node pointer，並指向priority queue最上面的node，進入buildcode進行編碼，以遞迴的方式建立，如果向左走則將原本的的字串code+"0"，而向右走則將原本的的字串code+"1"

```

1  void compress(){
2      // ...
3
4      // write the code into the code.txt
5      for(auto data : char_code){
6          stringstream ss;
7          ss << (int)data.first;
8          output_code = output_code + ss.str() + " " + data.second + "\n";
9      }
10     OUTPUT_TXT << output_code;
11     OUTPUT_TXT.close();
12
13     // write the compressed file into the compress.bin
14     COMPRESS_BIN.open("compress", ios::binary);
15     for(auto ch : input){
16         output_compress += char_code[ch];
17     }
18     output_compress += char_code['\0']; // add the end of file
19
20     // padding the last byte
21     if(output_compress.size() % 8 != 0){
22         output_compress += string(8 - output_compress.size() % 8, '0');
23     }
24
25     // write the compressed file into the compress.bin
26     int bytes_string[output_compress.size() / 8 + 1];
27     int index = 0;
28     for(int i = 0; i < output_compress.size(); i += 8){
29         string tmp = output_compress.substr(i, 8);
30         bytes_string[index++] = stoi(tmp, nullptr, 2); // convert binary string to int
31     }
32
33     for(int i = 0; i < index; i++){
34         COMPRESS_BIN.put(bytes_string[i]); // write the compressed file into the compress.bin
35     }
36
37     COMPRESS_BIN.close();
38 }

```

- L5-L11: 我們要將已經編好的char\_code把資料轉換成一長串的字串，首先將char\_code的first也就是char強制轉換成數字(題目要求)，丟進stringstream裏，接著接上之前所存的所有內容，分別加上將ss轉成字串(ascii-code)、空白、second(我們的編碼)、換行，最後存入output\_code，並寫入code.txt裏
- L13-L18: 我們將compress打開，將字串內容input，將一個一個字元進行編碼，最後再補上\0
- L20-L23: 最後我們將全部的資料，最後若不足8的倍數，則補上0
- L25-L31: 我們設立一個bytes\_string的陣列，剛剛是二進位的字串，我們每八個八個擷取，把他轉成數字，並放入陣列裡
- L33-L35: 我們把剛剛轉成數字的陣列，寫入compress
- L37: compress檔案關閉

- decompress()

```
1  void decompress(){
2      ifstream CODE_TXT, COMPRESS_BIN;
3      ofstream OUTPUT_TXT;
4
5      string code, output = "", output_compress = "";
6      map<string, char> code_char;
7      vector<int> bytes_string;
8
9      stringstream ss;
10     string s_code, s_char;
11
12     // ...
13 }
```

- ifstream CODE\_TXT: 讀取code.txt
- ifstream COMPRESS\_BIN: 讀取compress
- ofstream OUTPUT\_TXT: 寫入解碼完後的結果
- string code: 用來存入目前讀取到的字串內容(編碼)
- string output: 存入已解碼完的內容
- string output\_compress: 存入讀取完的compress並已轉成二進位邊把字串
- map code\_char: 建立編碼本 <string, char> -> <編碼, 所對應的字元>
- vector bytes\_string: 將讀入的compress內容轉為數字
- string s\_code: 存入來自code.txt裏, 最右欄的編碼
- string s\_char: 存入來自code.txt裏, 最左欄的ascii-code

```

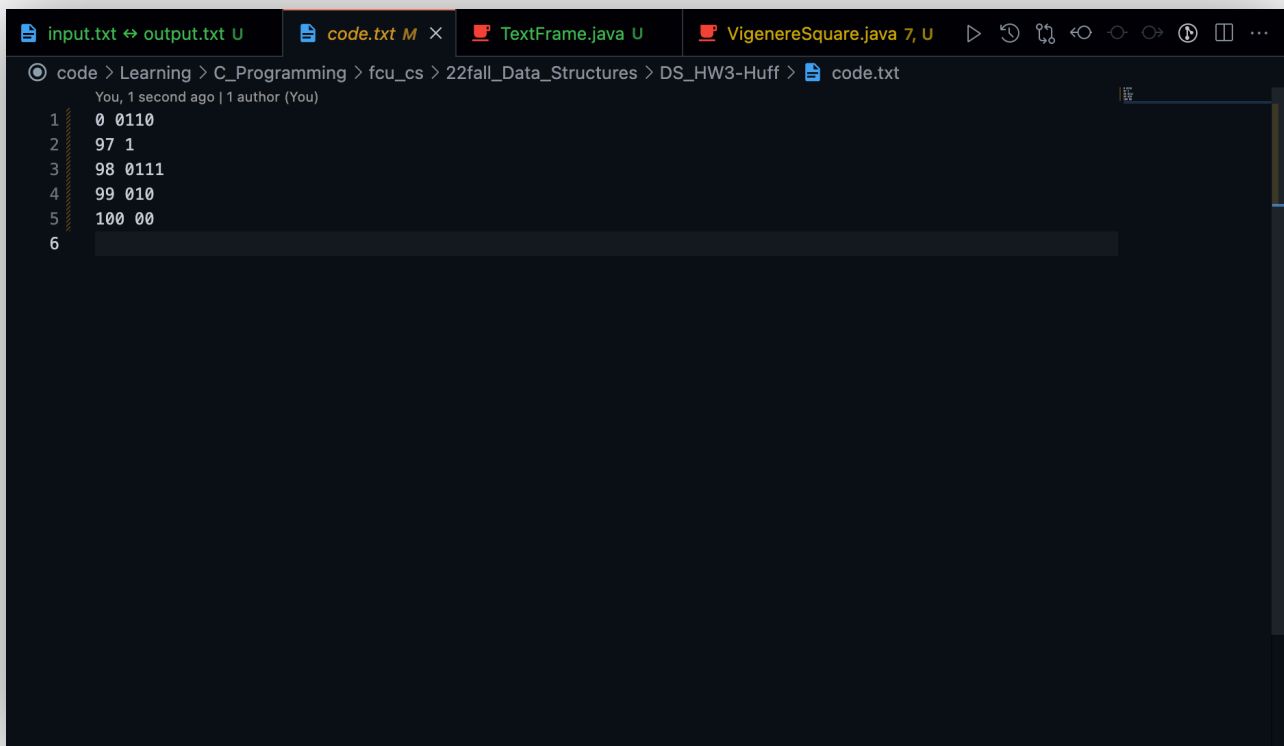
1  void decompress(){
2      // ...
3
4      CODE_TXT.open("code.txt");
5      COMPRESS_BIN.open("compress", ios::binary);
6
7      // read the compressed file
8      char ch;
9      while(COMPRESS_BIN){
10         COMPRESS_BIN.get(ch);
11
12         bytes_string.push_back((int)ch);
13     }
14     COMPRESS_BIN.close();
15
16     for(auto data : bytes_string){
17         string tmp = bitset<8>(data).to_string(); // convert int to binary string
18         output_compress += tmp;
19     }
20
21     while(CODE_TXT){
22         CODE_TXT >> s_char >> s_code; // read the code.txt
23         // cout << s_char << " " << s_code << endl;
24         code_char[s_code] = (char)stoi(s_char); // build the code
25     }
26     CODE_TXT.close();
27
28     for(int i = 0; i < output_compress.size(); i++){ // decode the compressed file
29         code += output_compress[i];
30         if(code_char.find(code) != code_char.end()){ // if the code is found
31             if(code_char[code] == '\0'){ // if the character is the end of file
32                 break;
33             }
34             output += code_char[code]; // decode the compressed file
35             code = ""; // reset the code
36         }
37     }
38
39     // write the decompressed file into the output.txt
40     OUTPUT_TXT.open("output.txt");
41     OUTPUT_TXT << output;
42     OUTPUT_TXT.close();
43 }

```

- L4-L5: 分別將code.txt, compress開啟
- L8-L14: 利用get()讀取資料，而是以字元讀入的，所以將資料存入ch，接著將讀到的字元轉成數字存入bytes\_string
- L16-L18: 我們將剛剛存在bytes\_string裡的數字轉成8bit的二進位編碼，再轉成字串的形式，並存入output\_compress
- L21-L26: 將開好的CODE\_TXT分別讀入資料給s\_char(ascii-code)和s\_code(每個字元對應的編碼)，並且寫入我們的map裏，<s\_code(key), s\_char(value)>，因為我們現在要用編碼去找本來的字元，因此key, value，這裡會和剛剛compress相反，最後關閉檔案
- L28-L37: 現在我們要開始解碼，我們將output\_compress裡的字元一個個讀取，並且先存到code裡，然後去找此編碼是否存在，若沒有則繼續加在先前字串後面，若有，先觀察是否為'\0'，如果是，則結束迴圈，若不是則從code\_char尋找本來的字元，將字元加進output裏，最後將code清空
- L39-L42: 開啟output.txt，將剛剛解碼完的字串寫入檔案裡，接著關閉檔案，最後結束程式

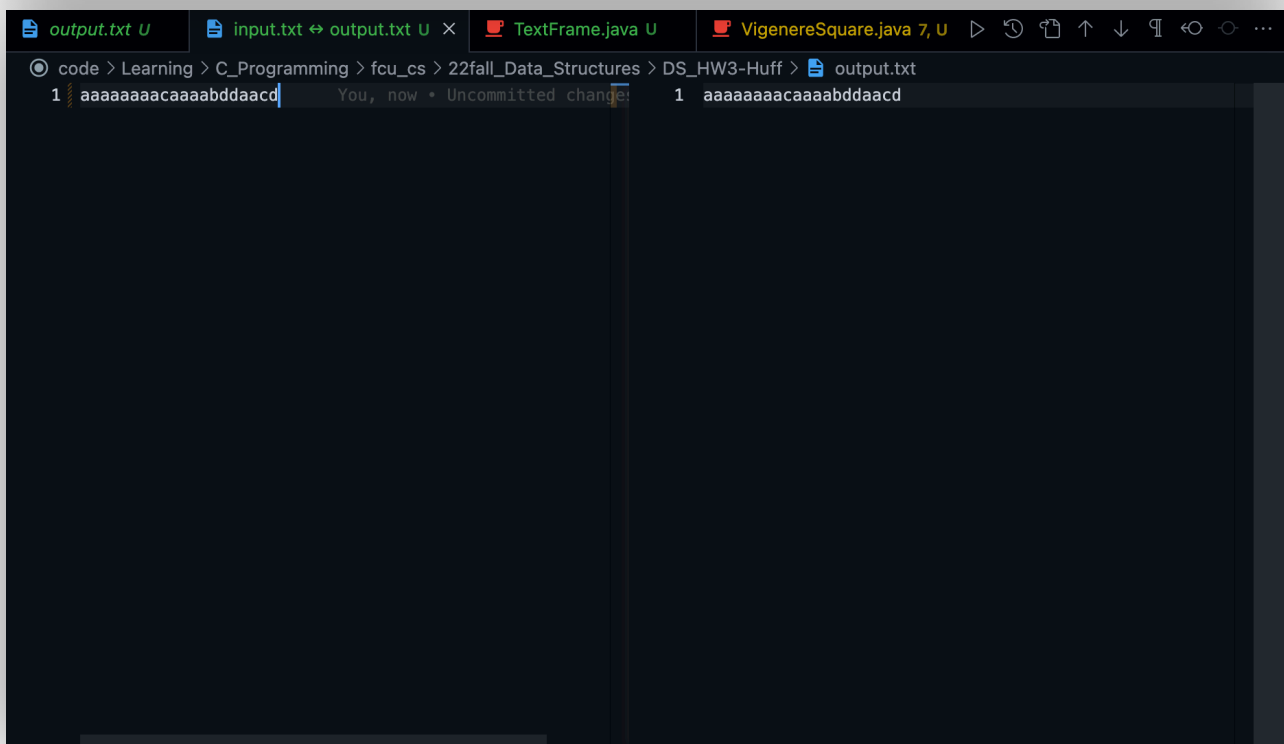
▸ 程式輸出結果 - 1

- Test case: “aaaaaaaaacaaaabddaacd” (講義上)



A screenshot of the Visual Studio Code editor interface. The top toolbar shows icons for file operations and running code. The file explorer on the left shows the project structure: code > Learning > C\_Programming > fcu\_cs > 22fall\_Data\_Structures > DS\_HW3-Huff > code.txt. The main editor window displays the content of code.txt, which contains a list of numbers and binary strings:

```
1 0 0110
2 97 1
3 98 0111
4 99 010
5 100 00
6
```



A screenshot of the Visual Studio Code editor interface showing a comparison between two files. The top toolbar includes icons for file operations and running code. The file explorer on the left shows the project structure: code > Learning > C\_Programming > fcu\_cs > 22fall\_Data\_Structures > DS\_HW3-Huff > output.txt. The main editor window displays a side-by-side comparison of input.txt and output.txt. Both files contain the same text:

```
1 aaaaaaaaaacaaaabddaacd
```

The comparison shows that the two files are identical, with no differences highlighted in red.

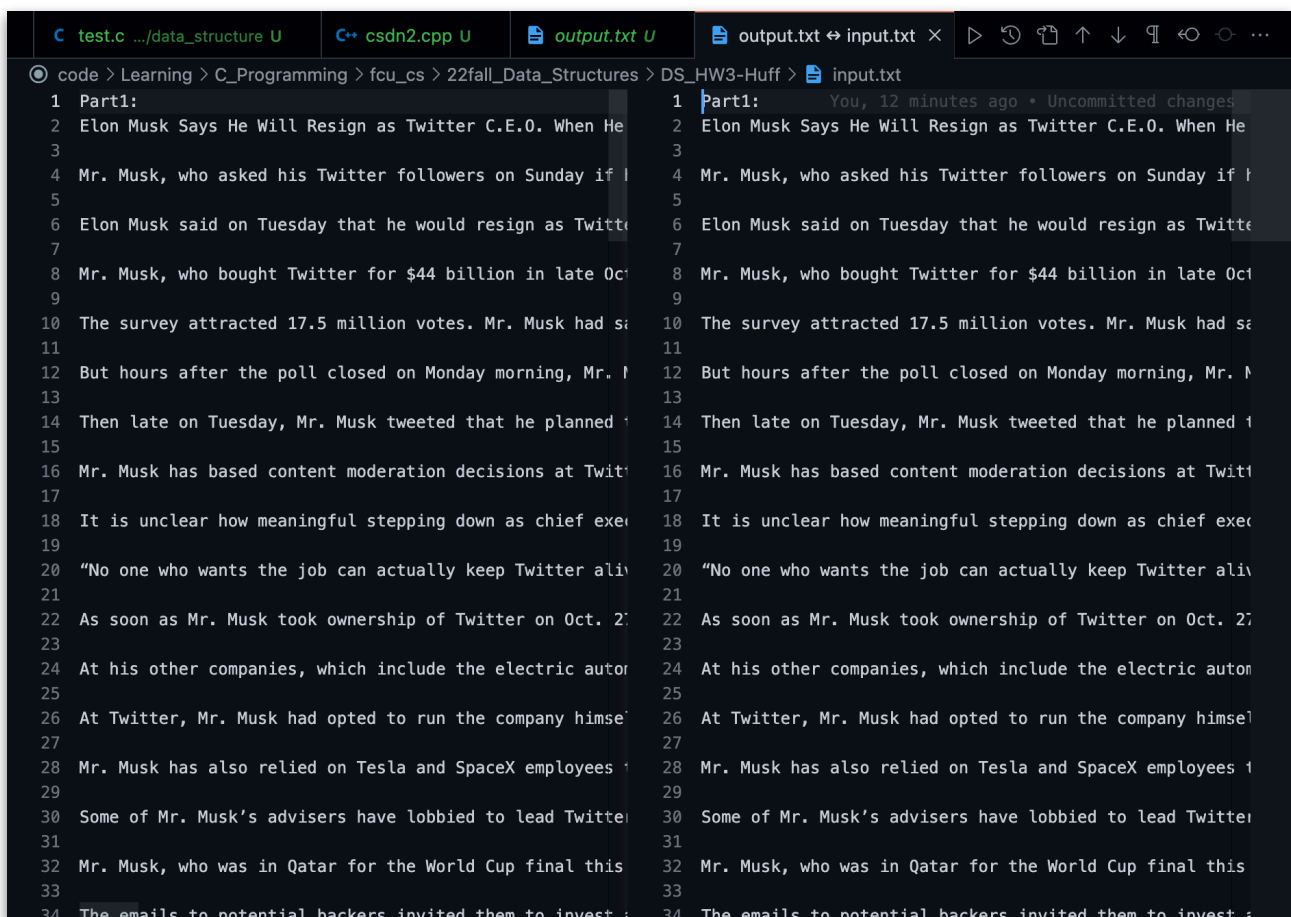
- 上圖：code.txt
- 下圖：input.txt 與 output.txt 的比較
- 我們利用VS Code內建的比較功能，做比對（編解碼正確，無出現異常紅色標注）

▶ 程式輸出結果 - 2

- Test case: 內文來自New York Times擷取的兩篇報導（共17832字元）
  1. Elon Musk Says He Will Resign as Twitter C.E.O. When He Finds Successor
  2. From Zero Covid to No Plan: Behind China's Pandemic U-Turn



```
code > Learning > C_Programming > fcu_cs > 22fall_Data_Structures > DS_HW3-Huff > code.txt
You, 16 hours ago | 1 author (You)
1 -128 1011000
2 -108 101110100001
3 -104 0010110110000
4 -103 00101010
5 -100 101110101
6 -99 101110110
7 -87 0010110110011
8 -61 0010110110001
9 -30 1001110
10 0 00101101100100
11 10 1011001
12 32 110
13 36 1001111000111
14 38 1001111000100
```



```
code > Learning > C_Programming > fcu_cs > 22fall_Data_Structures > DS_HW3-Huff > input.txt
You, 12 minutes ago • Uncommitted changes
1 Part1:
2 Elon Musk Says He Will Resign as Twitter C.E.O. When He
3
4 Mr. Musk, who asked his Twitter followers on Sunday if l
5
6 Elon Musk said on Tuesday that he would resign as Twitte
7
8 Mr. Musk, who bought Twitter for $44 billion in late Oct
9
10 The survey attracted 17.5 million votes. Mr. Musk had s
11
12 But hours after the poll closed on Monday morning, Mr. M
13
14 Then late on Tuesday, Mr. Musk tweeted that he planned
15
16 Mr. Musk has based content moderation decisions at Twit
17
18 It is unclear how meaningful stepping down as chief exe
19
20 "No one who wants the job can actually keep Twitter ali
21
22 As soon as Mr. Musk took ownership of Twitter on Oct. 2
23
24 At his other companies, which include the electric autor
25
26 At Twitter, Mr. Musk had opted to run the company himse
27
28 Mr. Musk has also relied on Tesla and SpaceX employees
29
30 Some of Mr. Musk's advisers have lobbied to lead Twitte
31
32 Mr. Musk, who was in Qatar for the World Cup final this
33
34 The emails to potential backers invited them to invest ;
```

- 上圖：code.txt
- 下圖：input.txt 與 output.txt 的比較
- 我們利用VS Code內建的比較功能，做比對（編解碼正確，無出現異常紅色標注）



## ▸ 心得

這次的作業跟以往有些不同，過去都只是基本輸出而已，但這次作業是要利用所學會的編碼演算法，對實際的文字檔做編解碼，這次作業也花了一些時間去寫，我第一個版本，寫得比較不好，方法也點笨拙，後來與朋友討論，才發現有個更好的方法，透過不斷地討論可以學習到他人較好的解法，互相琢磨，而這次跟以往不同的是，這次不僅教別人，也讓別人教我，使我學到更好的方法，這是最後一次資料結構的作業，而同時也是學到最多的一份作業。